Faster Onboarding of Developers in Existing Codebases

Master's Thesis

Sander van den Oever

Faster Onboarding of Developers in Existing Codebases

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Sander van den Oever
born in Spijkenisse, the Netherlands

Software Engineering Research Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

bunq
Naritaweg 131-133
Amsterdam, the Netherlands
www.bunq.com

Faster Onboarding of Developers in Existing Codebases

Author Sander van den Oever
Student ID 4135067
Email tudelft@sandervdo.nl

Abstract

When a new software developer joins bunq, a Dutch bank, (s)he will need some time to get familiar with the existing codebase. Preferably, the time necessary for the familiarisation is as short as possible. The faster the developer is up to speed, the faster (s)he can contribute to new features and help solving bugs. This research develops a prototype for a tool that aims to support developers in the comprehension of the PHP backend of bunq. This tool has been evaluated by interviewing multiple developers. Furthermore we asked the developers whether the tool would be able to replace a senior developer regarding question asking. During the interviews feedback was acquired on the current prototype. New developers found the tool useful, although there was also room for improvement. More experienced developers indicated that they found the tool less useful, but saw opportunities to use the tool for more managerial-like tasks.

Thesis Committee

Chair	Dr. A.E. Zaidman, Faculty EEMCS, TU Delft
University supervisor	Dr. A.E. Zaidman, Faculty EEMCS, TU Delft
Company supervisor	Ir. W. Van, Android Developer, bunq
Committee Member	Dr. M. Aniche, Faculty EEMCS, TU Delft
Committee Member	Dr. C.C.S. Liem MMus, Faculty EEMCS, TU Delft

Preface

This document is the result of my graduation project to obtain my Master degree in Computer Science. It would not be here without the help of several people. First and foremost I want to thank bunq for allowing me to work on this project in a very dynamic and interesting environment. Secondly I want to thank my supervisors, Andy Zaidman and Wessel Van. Both have provided me with a lot of valuable feedback to further improve my work. Lastly I want to thank all my friends, family and everyone else involved, but in particular Shirley, Mathijs, Emile, Jelle, "OG" Kevin, Christian, Meriç and Julita. You helped me through the hard parts and provided me with an unforgettable time at bunq, thanks for that!

Sander van den Oever

Delft, the Netherlands
October 10, 2018

Contents

List of Figures

Chapter 1

Introduction

Onboarding of new software developers within a company can be challenging, especially with large and/or complicated codebases. This challenge can cost both time and thus money. For companies it can hence be important to shorten the onboarding process. This will save them both time and money that would otherwise have been spent on the onboarding process. This chapter provides an introduction to the problem we are aiming to solve in this thesis. Afterwards, research questions will be formulated and explained.

1.1 Problem definition

When a company hires a new developer this developer needs some time to get familiar with the existing code and setup. During this time the developer will be less efficient. It has been estimated that 60% of the time for maintaining a software project is spent on acquiring a proper *understanding* of the codebase [11]. Limiting this time will result in more time being spent on actual fixes/new features, which is a profit for the company. bunq is a relatively new Dutch bank, with a relatively small team. The first commit to the backend projects' git repository was on November 11th, 2013. Between the first commit and June 2018, over 90.000 commits have been pushed to the repository. As of December 2017 the bunq backend repository contains over 1700 different folders (which can be seen as "modules"). To keep all information in one place, bunq uses the issue tracking capabilities of GitLab. GitLab is a web-interface around the Version Control System (git) that is being used to store the code. The issues are used to document the development process and design decisions. Every commit has an explicit reference to such an issue. Every issue thus has one or multiple commits linked to it, from which the related files can be obtained. Using this information a first filtering can be applied on the codebase to find the code that is actually relevant for a particular bug. For instance, when working on a bug in the iDeal part of the backend the code to upload a new avatar is most likely irrelevant. Only when files for uploading new avatars have been changed they would appear in the filtered set of modules/files.

The goal of this research is to support developers with the complexity of the backend

and to reduce the time necessary to get up to speed. A tool will be developed that supports developers in navigating within the framework and getting insights in how entities relate to one other. To help developers trace down the origin of a problem, this tool will be extended with extra information from the available documentation such as related features and issues.

1.2 Research Questions

The main need for this research is fed by the question *How can new developers be brought up to speed faster?*. The main goal of this thesis is to investigate an initial direction for a tool that allows developers to onboard faster within an existing codebase. The aim is to have the setup use a generic foundation, such that the theory could be applied to others contexts than bunq as well. However, for this use-case, the tool will be completely dedicated to bunq in order to be able to test its full potential.

RQ1 What requirements should be met by a tool to support new developers at bunq with understanding the business logics?

RQ2 What do developers think about a tool replacing a senior developer regarding question asking?

RQ3 What is the influence of having experience with coding in the bunq backend on the usefulness of a tool as defined in RQ1?

The hypothesis for RQ1 is that a tool has the ability to support (new) developers by providing the relevant information to get started. To understand the context of the problem, bunq and its codebase will be introduced in Chapter 2. Then in Chapter 3 we will look into relevant literature to see what has been done in the field already. The design process and implementation decisions of a prototype for such a tool will be discussed in Chapter 4. The design and setup of the experiment investigating RQ2 and RQ3 will be explained in detail in Chapter 5. The tool has been evaluated using a round of experiments of which the results can be found in Chapter 6. Finally, in Chapter 7, the validity and reliability of the results of the study will be discussed.

Chapter 2

bunq: start from scratch

The research questions as discusses in the previous chapter will be studied at bunq. This chapter will provide an overview of bunq, currently the newest bank in the Netherlands. It will give insight into what bunq is, how the company is structured and what their software systems look like.

2.1 What is bunq?

bunq[1] is an (officially licensed) Dutch bank since 2015. The company has been founded in 2012 by Ali Niknam. Niknam is also the founder of TransIP[2], a successful webhosting company. At this point, Niknam is the only shareholder. This allows bunq to make their own plans without intervention of other shareholders. bunq likes to think of itself as a tech-company with a banking license rather than a bank [23]. The employees at bunq ($\pm$90 as of January 2018) are far from the established stereotype bankers. They do not wear suits and do not apply a formal setting to every occasion. bunq only has a single office which is located in Amsterdam Sloterdijk, where everyone works on-site.

bunq originates from frustrations about the "old" banks that had systems and procedures that did not fit in the modern ages [23]. For example, payments should be instant, visiting branches should belong to the past and mobile devices should play a central role. All of these ideas were translated into bunq. bunq has no branches. Everything is managed through bunq's mobile app, from address changes to opening new accounts and ordering new debit cards. In bunq's book, "Breken met Banken", Niknam also explains the risks that old banks take by investing your money to make more money [23]. One of the core principles at bunq is thus that users' money is never invested (other than the minimum as required by law). Consequently, no credit services are provided at all; you can't withdraw / pay more than you actually have in your accounts. At the same time you also will not get any interest for

[1] https://www.bunq.com
[2] https://www.transip.nl

your money. This smartphone only, no investments and privacy focussed approach is what defines bunq.

bunq is also a company that cares about software developers. According to bunq themselves, they are the first Dutch bank that opened up a Public API[3] Using this API developers can do almost everything that would be possible through the official bunq app as well. This allows for new innovative solutions. For example: for every mile you go jogging 5 cents is transferred into your savings account. Or every time you spend money your LED light turns red.

2.1.1 The team

As mentioned before, bunq has around 90 employees by January 2018. These employees are spread over several teams. Examples are Customer Service, Compliance, Marketing, Product and Development. The software development team at bunq can be divided in four categories;

Backend implements business logics and endpoints to which the mobile apps connect.

Frontend implements the websites (among which the payment pages that redirect to the app) and the administration dashboard for support employees (internally called "guides").

iOS and Android implement the apps that are used by the end-users.

DevOps automating parts of the development process (Continuous Integration flows), infrastructure management and otherwise network optimisations, etc.

At bunq, as of January 3rd 2018, there are three senior backend developers, excluding Niknam, who (from time to time) helps out as well with some development. For the backend there are four junior developers and some interns. The frontend team consists of three developers and there are five developers that work on the mobile apps (Android and iOS). This makes the team relatively small compared to the total number of employees at bunq and there is a constant pressure on the teams to push out new features as fast as possible. New developers join every now and then, but the team only has limited time to guide the new developers. They therefore have a great deal of responsibility from day one. They need to figure out most things on their own.

2.2 Built from scratch

bunq is the first new bank in the Netherlands after DSB bank about ten years ago. [3] In 2012 Knab was launched, but they used the banking license of their founding mother: Aegon. Because bunq is an entirely new bank they were not held back by old legacy software systems like old banks [30], where the software commonly runs on older hardware and uses older languages like COBOL. Instead they could develop the entire system from scratch, which is exactly what they have done. This gave them the freedom to set up the system

[3] Application Programming Interface; allowing you to connect software systems to each other.

as they wanted, using modern technologies. They have built their own backend framework in PHP, a commonly used language for web development[4]. bunq does not offer a web-interface; bunq's ideology is that everything should be accessible from the smartphone. This way you can arrange your banking affairs always and anywhere you have access to the internet.

2.2.1 Backend structure

As mentioned before, the backend is almost entirely written in PHP, in a Java-like Object Oriented way. PHP has some perks but also advantages. According to bunq, it allows for fast development, it is readable and it is single threaded [5]. Disadvantages are the fact that PHP is loosely typed and fatal errors. The first is solved by wrapping almost everything explicitly in objects. The latter is solved by checking types and values to avoid unexpected behaviour. Apart from the backend language bunq uses a MySQL database to store the data next to several other technologies. Big chunks of the framework can be auto-generated from JSON specifications. When bunq wants to add a new table to the database, they specify the model in a JSON specification and from that the bare classes are generated.

All PHP classes live in their own file. These files are grouped by functionality, mostly driven by the Model-View-Controller (MVC) design pattern [7]. Each model has its own folder/module in the root folder of the repository. Folders are grouped by prefixes like `core_` or `tools_`. Suffixes are used to separate the controllers, models, views and libraries. An example of a valid folder name would be `core_user_model`. Considering the amount of models that live within the codebase the number of folders can be explained. For new developers it can be challenging to see how everything comes together however. B The Models, Views and Controllers are discussed in more detail below.

Workflows

The controllers in bunq's backend make sure that all information is properly retrieved and updated upon request. A bank has a lot of information to store, both from a service as a legal perspective. There are various regulations describing events that should be reported to authorities (suspicious account activities for example). Besides that, bunq offers additional features (for example avatars and comments) that other banks often do offer. The data for these additional features needs to be stored as well. Because of all this information the database consists of a lot of models. To make the use of all the information as easy as possible, bunq developed so called "workflows". Workflows are a structured description of business processes. Examples are creating a new user model and reading the details of a transaction model. The code for these workflows is mostly generated based on a JSON specification. Using this approach developers do not have to implement basic functionalities for every workflow. The JSON specification contains a structured sequence of steps which are executed in a specified order. It also specifies what input is required to execute the workflow and it specifies what data is returned by the workflow. Since workflows are used

[4] https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2018
[5] https://www.bunq.com/en/news/phava-by-bunq - Accessed Jan. 24th 2018

for nearly every automated business process there are a lot of them (4000+ as of June 2018). As depicted in Figure 2.1, workflows can be chained together to get to the final desired result. This can be used to focus every workflow on a single task and make the workflows individually reusable. An example would be the process to order a new debit card. This can be achieved by executing multiple smaller workflows.

Models

bunq stores (most of) its data in a MySQL database. This database has tables where each table corresponds to a Model in the codebase. The implementation of the models is mostly generated based on JSON definition files. These files describe what fields the Model should contain, and transforms this into for example the PHP Model classes and the necessary SQL queries to create the tables and the code to retrieve/update/delete the models from the database.

Views

The Views part of the MVC-pattern is mostly implemented in the Post-Processors. bunq makes use of so called Pre-Processors and Post-Processors. To explain these concepts we will look at the general approach of processing a request to the APIs. This can be divided in a few steps that have been visualised in Figure 2.1.

For every request these steps are taken;

1. **Routing**
   Map an API call (for example `/api/user/1`) to the right workflow,
2. **Pre-Processing**
   Validate the input for the workflow and prepare it for the workflow,
3. **Workflow**
   Perform the necessary logics,
4. **Post-Processing**
   Process the result from the workflow, e.g. filtering and structuring it, such that it can be presented to the user.

Note that in order to get a Response, multiple Pre-Processors, Workflows and/or Post-Processors can be utilised, as depicted in Figure 2.1.

Daemons

Some tasks need to happen on fixed times, or after each period of time (say, every minute). This is what Daemons are used for in the backend. The schedule (crontab[6]) ensures that the Daemon Master scripts are always running and reboots them when they are not. The Master in turn spawns one or more Workers that perform the actual logics. These logics are again one or more Workflows that can modify the database, communicate with external services, etc. This construction has been visualised in Figure 2.2.

[6] crontab is a utility to schedule jobs to be executed at specific times[21]

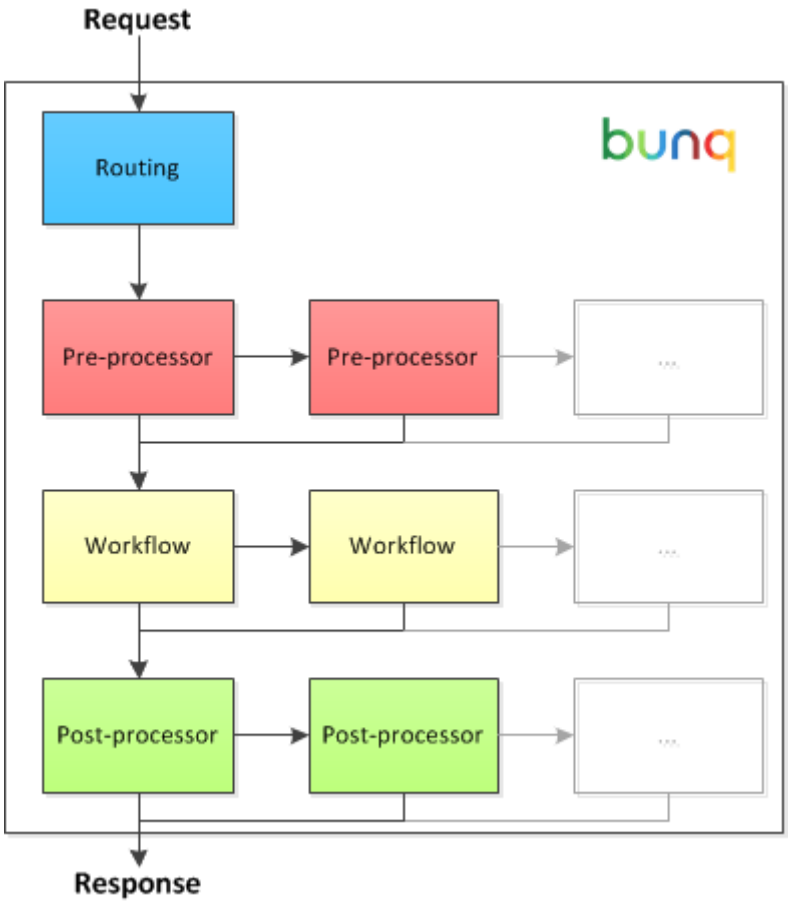


Figure 2.1: Overview of API call processing in the bunq backend.

2.2.2 Central Issue Tracking (CIT)

bunq uses a *single source of truth* principle. This means that all information/documentation is stored only once and only in a single place, where every employee has access to the data he/she needs. In the case of bunq's issue tracking this single source is a self-hosted instance of GitLab [7]. Within this installation there's a project called `CIT`. This project contains many repositories like `bugs`, `incremental`, `design`, `editorial`, etc. The issue tracking functionality of GitLab is used to store most of this information. The (somewhat simplified) setup is depicted in Figure 2.3. In Figure 2.3 you will see the GitLab server on the far left. On this server there are multiple Projects that in turn can contain git repositories. For the blue-colored repositories it holds that they do not contain code. For these repositories only the issue functionalities[8] are used for documentation purposes. The red-colored repositories do actually contain code and Merge Requests, but do not contain issues. Instead, the code

[7] https://www.gitlab.com/
[8] https://docs.gitlab.com/ee/user/project/issues/

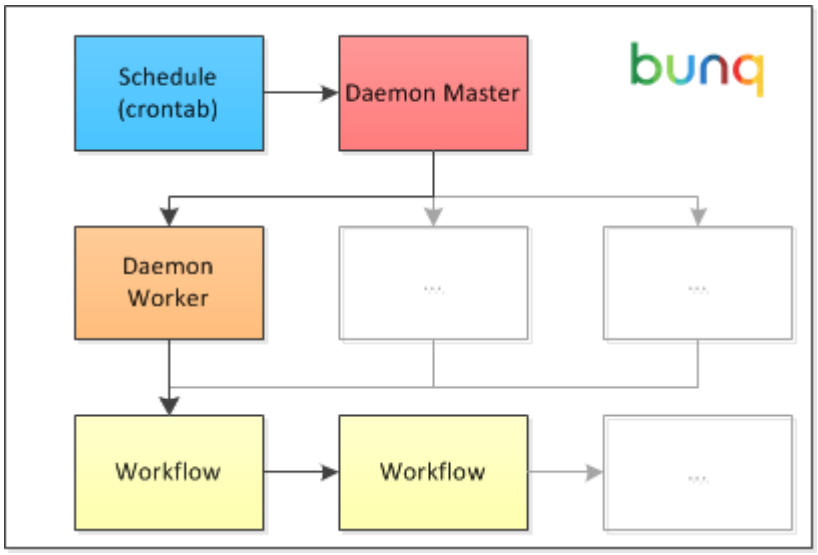


Figure 2.2: Overview of the Daemon setup.

and Merge Requests refer to the issues within the CIT project. Apart from the listed repositories there are more, which have been left out for clarity. The issues within the CIT project are created and maintained by support and testing employees. Developers use them as an information resource / backlog. Only developers work on the code and support employees do not have access to the code.

The internal procedures at bunq state that everything should link to each other. This is checked manually, when a missing link has been identified, it will be added after all. As this is a manual process we should consider that there might be incorrect links, or missing links that have not been found yet. For example, when someone proposes a new product in its designated repository, and it gets approved, new issues for implementation will be created linking back to the original product proposal. This way someone can always trace back the entire process. This is also visible in the source code. Test cases that arise from a bug always refer to the GitLab issue through an annotation like `@CIT CIT/bugs#123`. The same holds for the commit messages, they also always point to the issue they belong to. The messages include a reference like `(CIT/bugs#123)`. The git history then looks like;

```
397d**** MR points (CIT/incremental#2141)
dd80**** UserError instead of 500 when *** CIT/bugs#7343
c751**** Merge branch 'develop' into CIT/bugs#6970_avatar_***
8bdb**** add signup and validation *** types (CIT/incremental#2106)
5554**** Merge branch 'feature/user-***-finalized' into 'develop'
6b1d**** CIT/bugs#7349 TIN event should remain ***.
ca15**** MR points (CIT/bugs#6970)
```

The issues within the CIT project have labels assigned, indicating their priority, platform (mobile apps, backend, frontend, etc.), status (in progress, done, unreproducible, etc.)

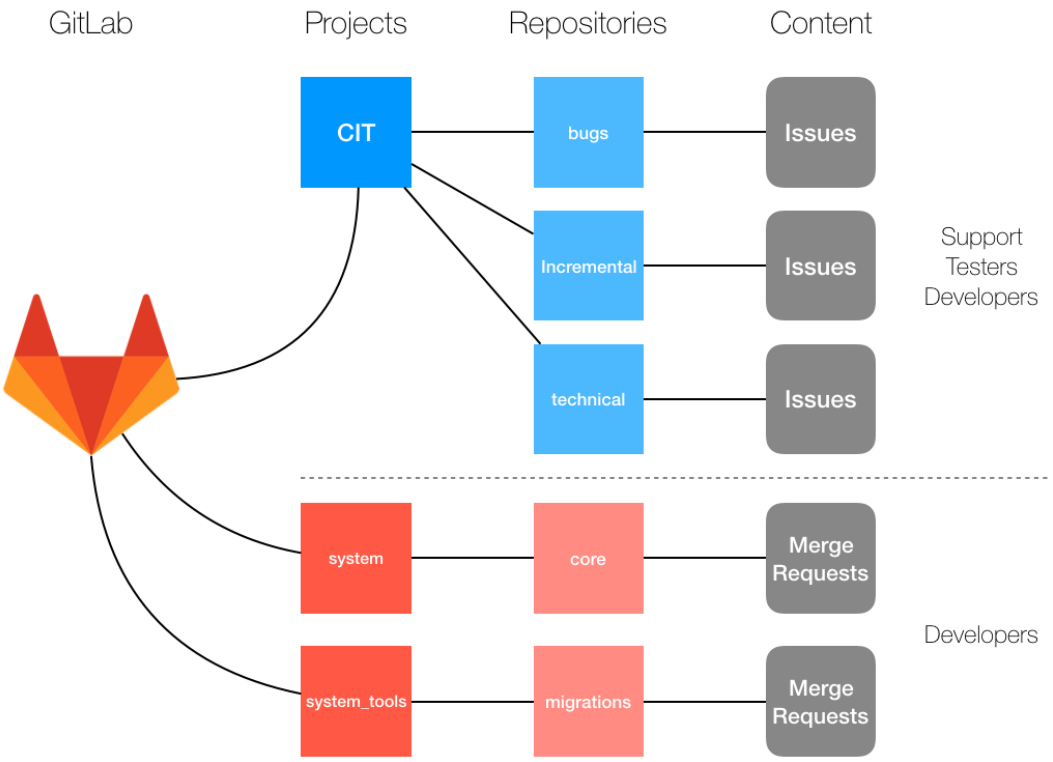


Figure 2.3: Overview of the GitLab (CIT) structure (simplied).

and what feature they relate to. Examples of these feature labels are `Feature: Cards`, `Feature: Direct Debits` and `Feature: Apple Pay`. One issue can have multiple feature labels. These labels are used internally to easily find all issues related to the implementation of a feature. Typically a feature has a backend issue and issues for the mobile apps. The backend will expose the API for the new feature, where the apps will implement that newly exposed API. There will hence be issues for the backend and the apps separately. For example, for `Feature: Payment Notes` the backend exposed an API that allowed the apps to submit user-written notes to be attached to one of their payments. Once the API for this was released, the mobile apps implemented their side, making use of the newly released API for this. All the issues for this feature, both for backend and mobile apps, have the same label so that they can be found easily. The labels are applied to the issues manually, meaning that there is a chance that labels are applied incorrectly. In practice this happens rarely, because multiple people look at the issues. It is typically noticed early on and resolved immediately.

Chapter 3

Background and Related Work

In order to onboard developers faster we will need to research the actual blocking points and existing techniques that are available to aid developers. A commonly used technique is software visualisation. We will explore what has been tried in this field up until now. Furthermore we want to look at traceability options in order to link source code parts to features/documentation.

3.1 Factors that slow down new developers

Over time software systems tend to grow as more features are being added. Typically when a developer starts a job (s)he joins an existing team where a product is being developed for a while already. This is recognised by multiple bigger companies that work on/with larger code bases [4]. It is known that a lot of time is spent on getting these developers up to speed, but it is also important to look at their *needs*. In a observation study performed by Begel and Simon [5] multiple issues came to light regarding the learning process for an existing software system.

Begel and Simon noticed that new developers (Novice Software Developer, NSD) often struggled with the available tooling (for large-scale development) and finding the right part of code to fix a particular issue. Furthermore they noted that NSDs had difficulties collecting all information, which could be scattered throughout multiple places. NSDs are not always allowed or able to ask experienced developers for help.

Yet, it has been shown by Dagenais et al. that asking your co-workers questions is most helpful when learning how to work in a corporate organisation [13]. Furthermore they explained that NSDs without domain knowledge were less efficient, which is similar to the findings of Zhou and Mockus [42]. Many processes in the financial industry are not documented for the general public (partially due to security risks). This means that not all domain knowledge can be obtained other than by joining a company within the domain. Being able to deduce domain knowledge from the source code could alleviate this burden.

In order to understand the domain/product it is useful to look at the smaller chunks first

[6]. Binder et al. described that it is easier to understand smaller chunks and master those pieces. Mastering the entire project, however, again would require some more time [43]. At bunq developers often work on specific features, which means that they should focus on looking at as few resources as possible.

3.2 Code Analysis techniques

In the previous section we have seen that NSDs typically experience difficulties when trying to understand an existing project. In the case of bunq, it takes them time to learn where to find the relevant code for a feature. Guesses are made based on folder names for instance (B.5). Having a proper analysis of the project that can indicate the relevant code entities upfront can save the NSDs time. In order to remove some of the difficulties we should ensure that the NSDs can look at smaller chunks of code. For this we need to be able to analyse which parts of the code are currently relevant. There are two common ways for analysis of source code; static and dynamic. Where static analysis considers the code itself, dynamic analysis considers the code when executing. Static analysis interprets the code as written and tries to deduce information from that. Dynamic analysis considers stack traces and (late) binding information [12]. Both approaches have up- and downsides. Static analysis is typically not able to capture the dynamicity of (web) applications while dynamic analysis on the other hand requires storage of information (logs) and might not exercise all targeted parts of the code [15]. Also, with dynamic analysis only the code that has been executed is analysed. To get an analysis of the entire system, every possible execution path should be examined. For bunq we primarily want to use static analysis as this can be ran once and then incrementally be updated with local file changes. As many concepts in the source code use strict JSON definitions it will not be difficult to visualise program flows in a static way.

3.3 Visualisation techniques

Visualisation of traceability links between software entities is known to help developers recover, browse and maintain the links effectively and efficiently [8]. Visualising these relationships is not a straight-forward thing to do [10, 31]. To visualise the code it is necessary to see what methods are available to create the visualisations. Visualisation of source code has been an ongoing research domain for some time now. Consequently multiple technologies have been developed and implemented. One can for instance choose to visualise the execution of an program, the structure of the source code, etc. When looking at the execution of a program it is possible to reveal the interactions between objects and the communication/behaviour of different components within the system [14]. These kind of visualisations are mostly dynamic, whereas static analysis can result in interesting visualisations as well. For instance an evolution view can be used to show parts of the system that change frequently according to Lommerse et al. [28]. The work of Lommerse et al. describes an evolution view where versions numbers are mapped against line numbers that changed, creating an overview of what parts changed in what version. This allows develop-

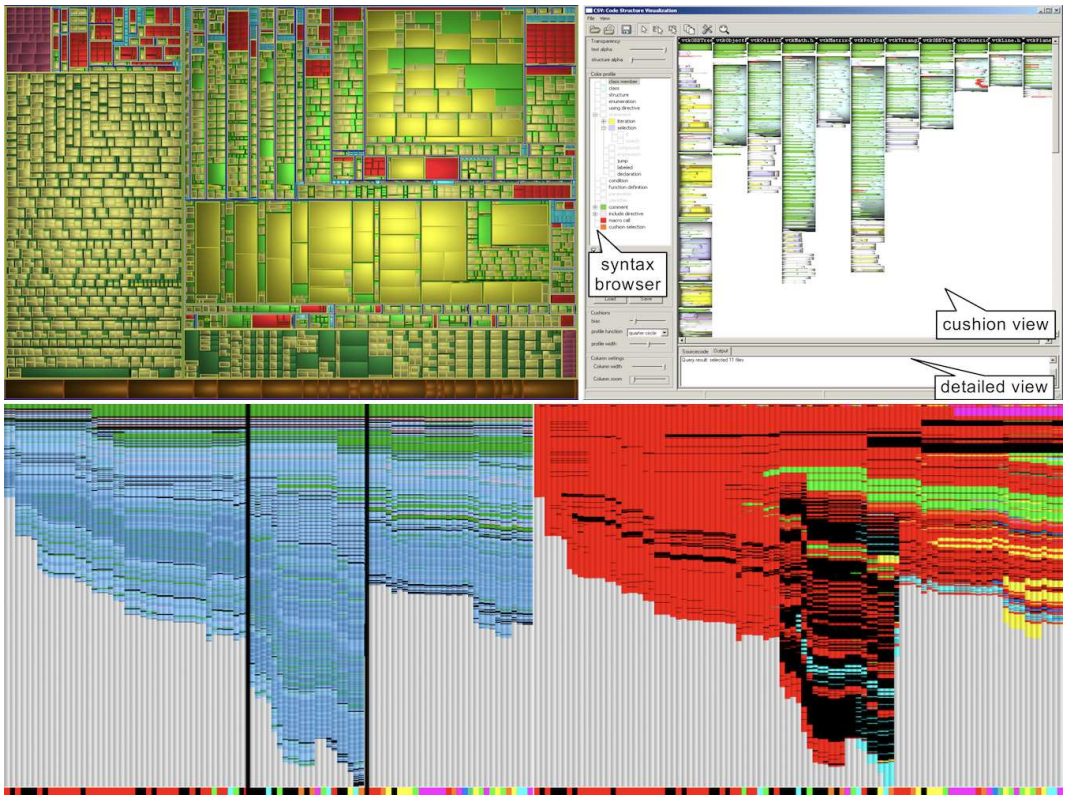


Figure 3.1: Example of the views by Lommerse et al.

ers to see the stability of the code and what parts belong together (as they likely require to be changed together). This can be interesting when determining which version of the code introduced a bug.

Lommerse et al. described two different views as well; the symbol view and the syntactic view. The former shows an overview as a linker would see (the global scope objects in C/C++; basically everything but actual implementation). The Syntactic view eases navigation by visualising code constructs / depth. For-loops, conditionals, etc. are depicted as blocks ("cushions") that are stacked on top of each other to simulate an optical depth. Figure 3.1 shows the different views by Lommerse et al. The top-left view is the Symbol view, top-right the syntactic view (with the "cushions") and on the bottom it shows two evolutions views, left is evolution of line-types and right evolution of authors. The visualisations in this work are mostly intended to aid the developer with navigating the code. Using these cushions could help to determine what kind of class is being dealt with. Classes that contain a lot of conditionals most likely contain business logics.

Another way to visualise code structures is making use of 3D space. Fronk et al. looked into 3D visualisations in their study [19]. They have build a toold that can visualise dependencies within Java programs. Although they had the impression that their tool, for bigger or more complex programs, the visualisations went beyond what was cognitively manage-

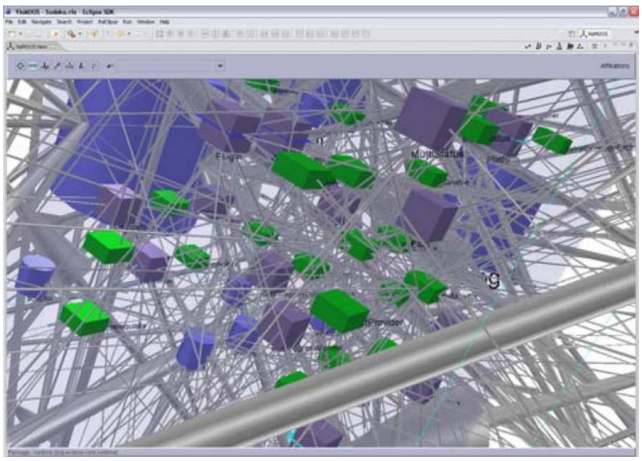


Figure 3.2: Example of the 3D view by Fronk et al.

able due to the amount of dependencies. An example of such a 3D visualisation is visible in Figure 3.2. Hence, it is important to keep in mind, when designing a visualisation, that it stays simple enough to be cognitively manageable.

When a developer at bunq wants to find the origin of a bug this can provide additional insight into where certain decisions were made. bunq has a tool available to get stack traces from errors. These traces are logged in the database. Using the tool, called `logview`, the traces are retrieved from the database and shown to a developer. These logs are completely textual and might not directly give a clear impression of the code that is touched by the trace. Hence, having a (more visual) overview could help to see faster where the problem could have occurred.

3.4 Feature location and tracing techniques

Feature location is known as the identification of source files that compose a specific feature [16]. A commonly performed task for developers is the localisation of a feature in order to extend it. This fits with the ideology of incremental design, which is described as the addition of new functionalities and properties to existing software [36]. In that sense it is important that developers do spend as less time as possible on these tasks and rather spend this time on actual development. The survey by Dit et al. describes multiple feature location techniques [16]. The techniques operate on different levels; static, dynamic and textual (documentation, etc.). Data sources range from source code artifacts to the documentation of the system. Visualization is one of the possible outputs of such a feature location tool. Knowing where a feature is implemented will reduce the time NSDs need to find the place where they need to make changes. Given the current size of the bunq code base it can be hard to see what folders are relevant for the current task. Being able to filter this down to only specific features can probably make navigating through the project easier.

3.4.1 Dynamic feature location

An example of a dynamic feature location tool is STRADA, this tool locates features by means of scenarios and trace analysis [18]. By running a scenario that triggers the feature it is possible to see what code is executed. By running another scenario that does *not* trigger the feature one can exclude the common source code, this is also known as Software reconnaissance [35]. An extension of this technique considers traces, but ranks methods based on the amount of occurrences in the traces. The main drawback of all dynamic analysis approaches seems to be the overhead; a lot of information has no value, but needs to be recorded anyway. Edwards et al. [17] reported results where the performance degradation was quite visible; their optimised tool still achieved 10% performance decrease and other tools performed way worse (on the `httpd` project only a 1% degradation was observed). For time-sensitive businesses it will be unacceptable to have too much performance degradation, even on running tests. bunq is such a time-sensitive business. Multiple systems have requirements on the response times, for instance the Instant Payment system. Having additional overhead is undesirable and should be avoided.

3.4.2 Static feature location

For static analysis a starting point is required. This point can be indicated by a developer, it can be selected randomly, it can be the main method, etc. Commonly, the starting point is specified by a developer [16]. Several tools have been developed to create so called Abstract System Dependencies Graphs (introduced by Chen and Rajlich [36]). These tools use dependency graphs between code artefacts (methods, statements, etc.) to locate features, given an initial starting point. Although these tools often use dependency graphs to perform the analysis, there are other possibilities as well. For instance the static data flow can be analysed. By indicating a variable/value it is possible to identify the source code that touches the variable. An approach like this was introduced by Trifu [39]. Static feature location does not require trace logs to find related files. Hence, there is no overhead at runtime as would be the case for dynamic feature location. This approach is better suitable for bunq. Starting points for bunq can be multiple. Examples are; the currently active Model (JSON definition, PHP class, database table, etc.), the active / used View (via the selected API call, JSON definition, etc.) and the currently active Controller (JSON definition or a related workflow). From these starting states all other related entities can be determined by following dependencies and usages.

3.4.3 Textual feature location

The aim of textual analysis is a little different from the dynamic and static approaches described above. Textual feature location aims to map textual descriptions of features to the actual source code that implements those features. Prior research was performed to the usage of `grep`, Information Retrieval techniques and Natural Language Processing (NLP) [35, 40, 16]. The usage of `grep` is quite limited, although it can still provide some benefits. The grep utility can be used for string pattern matching, but the programmer's name for a feature must match with the identifier in the code [35]. Using Information Retrieval

techniques works already better; by building an index of all terms that occur in the source code you can create a nice vector space that can be matched with a query vector [40]. NLP is closely related to that; it tries to map a human text to a query string. This provides great flexibility in the input, while keeping a nice precision for the output [16]. This approach requires to create an index to which queries can be mapped. It will allow to search for generic keywords in order to find the related implementations. For bunq this could be useful, but the bigger part of this feature is already available through the editor that is used; PhpStorm (this feature is also mentioned in interview B.3).

3.4.4 Traceability

Traceability can be described as following requirements throughout the evolution of a software product in a forward and backward direction [20, 33]. This enables developers to go from a high-level document (documentation/requirement) to the related low-level software entities [33]. These traceability links are important for various tasks including program comprehension [24]. Artifacts can be scattered all over the code base. As such it can be hard to collect and maintain all links (the latter is a critical problem) [9]. Another challenge is that, for example, web applications typically utilise multiple languages; in the case of bunq PHP and JavaScript. This means tracing throughout multiple files *and* multiple languages. According to Ratanotayanon et al. developers do not want to spend time on arbitrary tasks like managing traceability links, they do not have the time and/or patience for that [37]. The work of Ratanotayanon et al. uses the "diffs" of the files to determine what changed and they developed an algorithm that would update the links based upon that information.

Commit logs

Platforms like GitLab[1] and GitHub[2] allow referring to issues in the repository from the commit message [3]. Ideally a commit message is descriptive of the contents of the given commit [29], but unfortunately many commits lack proper descriptions [25]. At bunq, however, it is mandatory to refer to the related issues in CIT. Therefore it can be used to attempt to extract the feature that was touched. The other way around works as well. Given a feature, it is easy to see what commits are related and from there you can easily list all artifacts involved. One important thing here is that the links are created manually, meaning that they are prone to (typing) mistakes.

Folder and File names

There are also several other static approaches to map features to their corresponding files. Some examples are listed in the work of Liu et al. [26]; execution traces are used to determine the files that are touched by a specific execution. These traces are produced by running

[1] https://www.gitlab.com
[2] https://www.github.com
[3] https://about.gitlab.com/2016/03/08/gitlab-tutorial-its-all-connected/

tests that exercise specific behaviour. Giving the number of folders in the bunq code base it can be useful to map a feature to the proper folder(s).

Chapter 4

Design and implementation of the prototype

In order to support developers in their onboarding process a prototype of a tool will be developed. First we will set up some requirements for this tool. Then we will describe the way the data is collected from different data sources. Once we know how to collect the data we will explain how the data will be processed and stored. Finally we will discuss how the frontend will be set up and how it will query the stored data.

4.1 Design requirements

In order to understand what needs to be built all junior developers have been interviewed. During these (one-on-one) interviews the developers were asked about their approach in the first weeks and what obstacles they had to overcome. In total six developers have been interviewed, coming from both the backend and frontend team. The questions asked during the interviews and the complete transcripts of the conversations can be found in Appendix B. From the interviews we can draw the conclusions that most software developers struggle(d) with [redacted]. Another cause for confusion were [redacted]. Those issues would only come to light during code reviews. As there is a limited number of developers, it is hard to provide more direct guidance. However, an attempt can be made to make the available documentation easier to find.

In order to design a supportive system, first, some guidelines needed to be drafted. The goal is to provide newly onboarded developers with information to understand the codebase faster. To this end, the system will need to provide insights into the setup of the code-base and framework (DR1). The developers at bunq indicated that the size of the code base is one of the struggles. The assumption is that a tool that can pinpoint features in the codebase can help when searching for something specific. The system should therefore be able to point developers to the right place for a given feature (DR2). Often new developers start

with small bugfixes to get familiar with the code-base. The system should thus, given an issue, be able to point to related files (DR3). To be able to ask questions to the right person it is necessary to see who worked on those files as well (DR4). The workflows, as explained in Chapter 2, are defined in JSON files. These files describe some sort of flowchart. When there are multiple conditional jumps this becomes less clear. Hence, for the logic parts the system should visualise these workflows, such that it is easier to determine more specifically what part is affected by a bug/issue (DR5).

The system must be responsive (DR6). According to the work of Palmer users are only willing to wait a handful of seconds for a response in (web) applications [34]. When the tool is not responsive enough, developers might not want to use it. Therefore the relevant data should be queried from a simple data structure, that allows for quick lookups. Ideally the system is live; it reflects the current state of the project (or the `develop`-branch as it is in GitLab).

4.2 Data collection

The git repository already contains a lot of information. The different commits (almost) all have links to the issues in CIT. This way, the files touched by a commit can be linked to a specific issue.

We wrote a PHP script that runs on a local repository. The choice to run the script on a local clone of the repository was made for performance reasons. Alternatively the script could run directly against the GitLab API. This would mean, however, that the script needs to make a call for every commit. Given the amount of commits this would create a lot of requests to the API and slow down the process, whereas when the repository lives locally, the script has the data available immediately.

The script runs the following command;

```
git log --name-status # Adds committer name to the output.
        --no-merges   # Exclude merge commits.
        --after="%s"  # Analyse commits starting from this date.
        --before="%s" # Analyse commits up until this date.
        --format="%s" # Format the output given a string.
```

Where the `after` and `before` parameters get values of the format `2018-01-01 00:00:00`. The `format` parameters get the following value;

```
:::::COMMIT:::::%n
%H%n
:::FIELD:::%n
%aN%n
:::FIELD:::%n
%aE%n
:::FIELD:::%n
%s%n
```

```
:::FIELD:::%n
%b%n
:::FIELD:::%n
%cI%n
:::FILELIST:::%n
```

The commands generates an output as follows;

```
:::::COMMIT:::::
5c49**********************************
:::FIELD:::
N*** ***
:::FIELD:::
n***@bunq.com
:::FIELD:::
Improved code style CIT/internal#834
:::FIELD:::

:::FIELD:::
2018-04-27T00:19:43+02:00
:::FILELIST:::
M       bunq/***_lib/code/***.php
M       bunq/***_lib/test/***Test.php
```

This fixed structure allows to parse the data quite easily by splitting the output on the newline symbols (`\n`) and afterwards on the separators like `:::FIELD:::`. The data that is extracted contains;

Commit hash (`%H`) A unique identifier for the commit.

Author name (`%aN`) Name of the author of the commit.

Author email (`%aE`) Email address of the author.

Commit subject (`%s`) The header line of a commit, per style guide this should contain a reference to a GitLab issue in the format of `CIT/[project]#[number]`.

Commit body (`%b`) Rarely used in the bunq backend repository, yet sometimes contains an explanation or additional references to issues.

Commit date (`%cI`) Date the code was committed.

File change type indicators (e.g. `M` for "modifier"). This indicates whether the file has been created, modified, deleted, etc. Only files that have been created or modified are considered for now.

File paths The relative path to the file, seen from the root of the repository.

The script does only consider file creations and modifications. When file deletions / moves are considered it would increase the complexity of the data structure. Instead, files are checked for existence when they are retrieved from the database. This prevents complicated queries and still gives a complete overview of all files that are and have been involved with a feature. All the information will be parsed in PHP objects and temporarily stored in the working memory. The git repository is analysed month by month so that memory will not overflow. Once all commits for a given month are parsed, they will be uploaded individually into a database, as will be explained in detail in Section 4.4.

4.3 Enriching the data using GitLab

Right now the data in the database only contains the raw references to the issues. We want to add the actual issue titles, as visible on GitLab, to make it easier to see what an issue is about. To this end the database is queried for all `Issue` nodes that do not have a title set and that do not have a mark that they have been checked already. For all of these `Issue` nodes the GitLab API is queried using the unique issue reference. In the response from the GitLab API we can also find any `Labels` applied to the `Issue`, if any. Those are added to the dataset as well. Furthermore, both the issue title and body (the field where a reporter can write a detailed explanation of the issue) are scanned for further issue references. Some issues are linked to a so-called PARP-issue. PARP stands for Product Approval & Review Process. When a the implementaton of a feature can be split over multiple teams, every part will get its own issue. All these separate issues will link back to the original PARP-issue, which functions somewhat as a summary. Commonly when an issue has a PARP-issue, it is linked by referring to the PARP in the child issue and vice-versa. These newly found (related) issues are also added to the dataset and also enriched with their titles and labels.

4.4 Setting up the database

The data as retrieved from the git log is stored into a database. By pre-analysing the data the system can be made responsive to developer inquiries (DR6). When the system would need to parse the log on the fly it would mean that the tool would not return a response within seconds, rather it would take hours or time-out. The main reason for this is the amount of commits. The full analysis of the repository took between 7 and 8 hours. The analysis considered commits ranging from the very first commit in this repository on November 11th 2013 09:57:25 +0100 until June 14th 2018 09:00:00 +0100, resulting in 50,388 commits.

4.4.1 Identifying data entities

A few entities have been identified from the available data;

Commit Represents a Commit on the project under analysis.

Issue Represents an Issue in GitLab (can also be refered to as “ticket”, “bug report”, etc.).

Author Represents the author of a Commit. Can be seen as a "Developer".

Label Represents a Label that can be attached to an Issue on GitLab.

File Represents a file that is or was present in the repository.

The main chunk of this data, knowingly the `Commit`, `Author`, `File` and `Issue` nodes can be created from the raw git log data. The `Commit` entity will contain the commit's hash, subject, time and when available its body. The commit hash is considered to be an unique element for the entity, which it is for git itself as well. The `Author` entity will contain the author's name and email address. Here, the email address is considered the unique identifier for the entity. Sometimes a different displayname is set up, but GitLab only considers the email address to link commits to a certain user. For `File` the entity only contains the relative file path, which is also the unique identifier for the entity. The `Issue` entity only contains a reference to GitLab at first. This reference will have the format `CIT/[project]#[number]`. The `project` part can be one of the categories of issues, like `bugs`, `incremental`, etc. The reference is considered an unique identifier for the entity as every combination of `project` and `number` can only exist once in GitLab.

As mentioned, the `Issue` entity will in a later stage be enriched with additional information, retrieved from GitLab. This includes its Labels and its written content (the title and the body). The Labels for an Issue are entities on their own. That describes all the data entities that will be stored for use within the tool.

4.4.2 Choosing database type

One consideration for the database was the type of database. There are currently multiple types of databases available. Examples are relational and graph databases. In this section we will investigate which type and structure fits best with the design requirements of the tool.

Relational databases (e.g. MySQL) The data entities as described above are actually suitable to be mapped into a relational database. The downside here comes when data needs to be obtained together (when a lot of JOINs happen). Other than that there are no real downsides for our data structure.

Graph database (e.g. Neo4J) The data fits well in the graph-like structure. All entities already have a clear connection between them. Relations between the entities can directly be mapped into the database structure. This also allows to retrieve all related entities by following the edges between nodes in the graph.

Document-based database (e.g. MongoDB) Document-based databases basically store a JSON structure. They can efficiently search through all the data, scanning all these documents. However, the cost for this is data duplication.

Key-value-based database (e.g. Dynamo) Stores keys with their respective values, but does not really provide a predefined data-structure. It is highly efficient for looking up a value given a certain key.

The Relational and Graph databases seem to fit best with our data structure. There are clear relations between the different entities and information will often be collected together. To retrieve all information that can show up multiple joins would be necessary. Here the benefit of a graph database shows.

4.4.3 Choosing database software

This section describes the approach towards selecting a specific database engine. Multiple graph databases have been considered. We mainly focussed on performance and popularity of the software. The performance is important to ensure a responsive system. The popularity is important for technical support in case of difficulties and for the availability of supporting libraries. After the initial filtering two candidates remained;

Dgraph Modern and relatively new distributed graph database that it said to have an outstanding performance.

Neo4J The current market leader, around here for years and having a big community to turn to for support and guidance. Several tools and libraries are already available for Neo4J.

An initial attempt was made to implement the Dgraph database for the tool. However, as Dgraph is relatively new, there are not yet a lot of libraries available. An attempt to implement a custom library was stopped early-on because it was not worth the time. The estimation was made that the gained speed from Dgraph would not justify the time required to implement it in the other parts of the tool. Hence, the choice was made to go for Neo4J, which was set up within minutes.

4.4.4 Choosing database structure

In Section 4.4.1 we have identified five different data entities;

- Author (indexed on email address)
- Commit (indexed on commit hash)
- Issue (indexed on GitLab reference)
- Label (indexed on name)
- File (indexed on path)

These entities have been mapped directly to corresponding nodes in the graph database. Between the nodes relationships have been defined;

`Author -> Commit` `COMMITTED` Links an author (developer) to a certain Commit node.

`Author -> Issue` `WORKED_ON` Links an author (developer) to an Issue, indicating that the developer has contributed code that relates to the referenced issue.

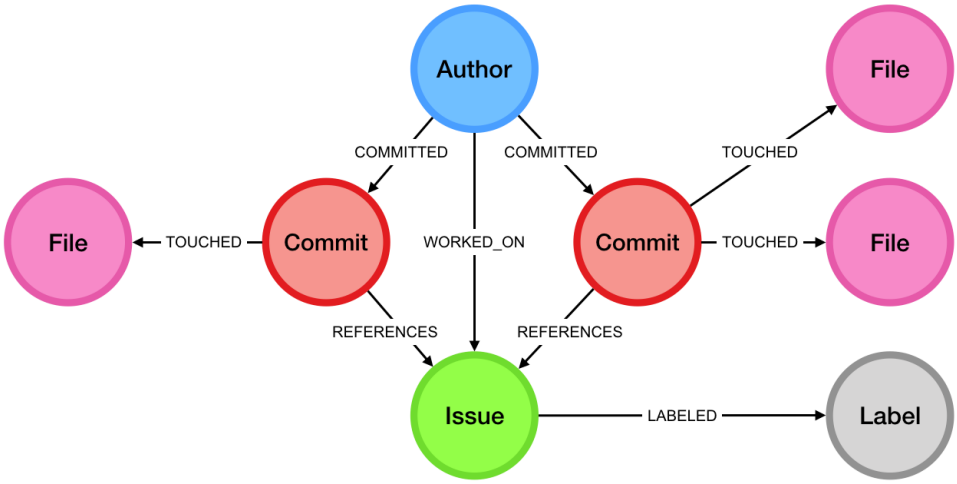


Figure 4.1: Structure of the Neo4J graph.

`Commit -> Issue` `REFERENCES` Links an commit to an Issue, based on the reference in the commit message. This indicates that the Commit is related to / (partially) resolved the references issue.

`Commit -> File` `TOUCHED` Links a commit to a File. This indicates that a certain file was either created or modified in the Commit.

`Issue -> Label` `LABELED` Links an Issue to a Label. This indicates that a certain Issue was labelled with the corresponding Label in GitLab.

When determining the relations the aim was to introduce minimal relationships, yet connect all nodes. Furthermore the aim was to make use of Neo4J's powerful Cypher queries rather than defining explicit relations between each pair of nodes. The only exception that has been made is the `WORKED_ON` relation. This would simplify some queries, improving their maintainability enough to be worth it.

In Figure 4.1 an example of the structure is shown. Using the Cypher query language it is now possible to run more complicated queries on this dataset. Examples of queries that can be executed easily;

- What files have been touched by some developer D?
- What files were touched for issues labeled by some Label L?
- What commits refer to some Commit C?
- What Labels have been worked on most by some Developer D?

4.5 Interface design

In this section the requirements for the user interface as well the implementation will be discussed in detail.

4.5.1 Choosing technologies to use

As bunq already has many (custom) web-based tools it made sense to follow this pattern. The ability to have the tool as a plugin for the editor was considered, but mostly because of lacking documentation the decision was made to go for a web-based application. As most of the (web) tools at bunq use JavaScript to render their information it seems logical to keep the same approach for this tool. This ensures that the team will be able to maintain the tool later on. Furthermore there's years of development experience, which can be very useful during the implementation.

The frameworks that have been considered are;

Angular Fast, but complex syntax.

ReactJS Lightweight and easy syntax.

VueJS Detailed documentation, albeit not entirely complete (in English). As this is a newer framework it comes with less resources at hand.

Considering the more complex syntax of Angular and the sometimes incomplete documentation of VueJS we opted to go for ReactJS. Added benefit of using ReactJS is the experience that one of the bunq developers has with ReactJS.

4.5.2 Design of the interface

The main idea is to visualise what files belong to a certain feature (DR2, DR3). So the main part of the interface should focus on that. Additionally we should show some related information, including:

Developers the developers that have contributed to a given Feature (for DR4).

Related Features features that have a lot of files in common. (for DR2)

Related issues issues that relate to this feature (either the implementation specification or bugs/improvements) (for further exploration and info retrieval) (DR3)

Workflow visualisation a visual representation of the logics in the workflow (DR5)

Each of these items aims to reduce the problems that new developers at bunq experience as identified in Section 4.1. Showing the developers of a feature should make it easier to identify who to address your questions to. Related features and issues on the other hand, aim to provide insight into what parts of the code base might be affected as well by the changes in focus, reducing/removing the need to ask where to look. The related issues

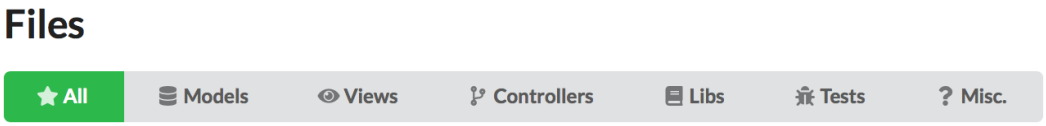


Figure 4.2: Filter to narrow down the scope of files shown.

can give an insight in bugs that occurred during the development of a feature, providing more related documentation to the current feature or bug. For DR5 the tool will include a visual representation of the workflows. As the JSON definitions of workflows describe a flowchart-like structure, the definitions will be parsed into actual visual flowcharts.

After some initial tests it turned out that there are features that return long lists of files (100+). To that end some filtering was introduced, as is illustrated in Figure 4.2 (DR1). Because bunq enforces a structured filename/path convention. Therefore the filters could be implemented using Regular Expressions on these conventions. In the case of Models the files are located in folders that end with `_model`. Likewise for Controllers (`_controller`), Views (`_view`) and Libraries (`_lib`). Looking at the the files that were still grouped under "All", we could identify one more category that can be filtered easily with a regular expression, the tests. Tests can be spread over multiple folders, but in almost all cases there is a subfolder called `test`. This is used as the regular expression to filter here. Finally there's the Miscellaneous category. Anything that did not fit in one of the previous categories or where no simple regular expression could be written, end up here. This also includes files that should have been placed in one of the earlier-mentioned categories. In the case of old commits it occurs that the filepaths still follow the old conventions. In such cases almost all files appear under the "Misc." category. However, after looking at a random sample of features, this appears to be only applicable to a fraction of the features.

4.5.3 Queries to the database

The data shown in the interface is retrieved ad hoc from the Neo4J database. Within the interface a query can be initialised by making use of one of the search options. There are three options to search by; features, developers and CIT (issue tracking) references. Each of these options will be explained in detail in the following sections.

Searching by Label

The search option consists of a dropdown (with auto-completion) that shows all Labels that are present in the Neo4J database. Once a feature has been selected a search will be initiated, knowingly the following Cypher query;

```
MATCH (l:Label) WHERE l.name =~ {label}
MATCH (l)<-[:LABELED]-(i:Issue)<-[:REFERENCES]-(c:Commit) ...
     ... <-[:COMMITTED]-(a:Author)
OPTIONAL MATCH (c)-[:TOUCHED]->(f:File)
```

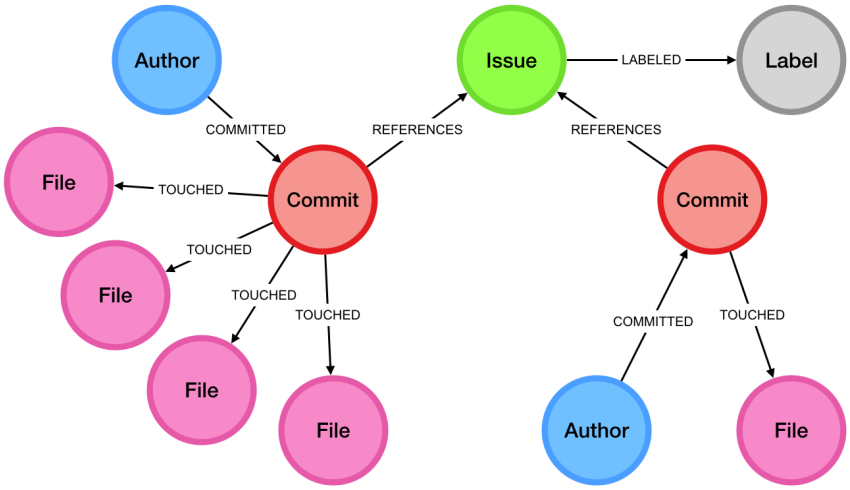


Figure 4.3: Graph structure of the result of a "Search By Label" query.

```
OPTIONAL MATCH (i)-[:RELATES_TO*..5]->(i2:Issue {parp: true})
RETURN a,i,c,f,i2;
```

The starting point of the query is the Label with a certain name `{label}`. For the matching label (which should be one, as each Label has a uniqueness constraint on the name) the database will follow all relationships of the type "LABELED". Meaning that now the result set includes the Label itself, and all issues that have this label attached to it. In turn, for all Issues, all commits that reference these Issues are extracted. And for each of these commits the Author information is retrieved. Finally the last two pieces of information are added to the result. First, for all commits in the result it will attach the files that have been changed. Secondly, for all issues it will find related issues (because they have a reference to the other issue). In the end there will a structure similar to the example in Figure 4.3. In this figure, the green node depicts an Issue with its associated Commits (red nodes). These Commits have been authored by their Authors (developers, blue nodes) and touched several Files (purple nodes). By performing this query on a certain feature (Label) one would thus obtain all files, developers and issues that relate to the Label.

Searching by Developer

This search option consists of a dropdown with all developer names and email addresses. Searching is possible on both data points. The dropdown is populated with a generic query to the Neo4J database, selecting all possible Author nodes. Once the user submits an inquiry in the tool for a certain developer, the following query is ran;

```
MATCH (a:Author {email: {username}})-[:COMMITTED]->(c:Commit) ...
     ... -[:REFERENCES|RELATES_TO*..10]->(i:Issue) ...
     ... -[r:LABELED]->(l:Label)
RETURN DISTINCT l, count(r) ORDER BY count(r) DESC;
```

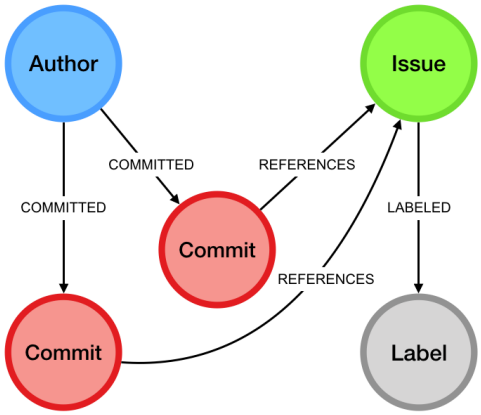


Figure 4.4: Graph structure of the result of a "Search By Developer" query.

The first thing that happens it that all Authors are retrieved that match the unique identifier (email addresses are used as identifier). Once the author node is found, the Commits by that Author are retrieved. For every Commit the references Issues are then added to the result. Now, for every Issue the Labels are fetched. As Author and all reachable Labels are known, we count the possible paths to each Label. Meaning that if an Author worked on 2 issues that relate to the label "Feature: iDeal", there are two possible paths to the Label. One via each of the issues. This way Labels are ordered by the amount of issues resolved within them. For users of the tool this will show them what a developer has worked on most.

The graph that is used to perform the count is depicted in Figure 4.4. In the graph the blue node depicts the Author, the one that was searched for. The red nodes are Commits by this user, with their referenced Issues (green). From the Issue one can obtain the Labels (grey nodes) that belong to them. As can be seen in the figure, there are two possible paths to the Label node in this example. The count for this Label would hence be '2'. This number is calculated for all Labels and the results are ranked according to it.

Searching by CIT Reference

This search has three input fields that, together, form a complete GitLab reference. GitLab references have the format `[group]/[repository]#[issue_number]`. Nowadays the group is always `CIT`, but in the past this structure wasn't followed yet. Hence, there were other groups that contained issues as well. To allow users to search through these older issues as well, the group name can also be entered. Once the reference has been provided, the system will find the node for this issue, given that it exists. For the found issue it will then retrieve all Commits that reference this issue. Then, similar to searching by Label, the touched Files and related Issues are appended to the result. The difference between this search and the "By Label" search lies in the scope. The scope of the CIT search is smaller, giving a more narrow result. The aim of this search is to be used for debugging purposes, to easily track down who worked on a bug and what was changed to solve it.

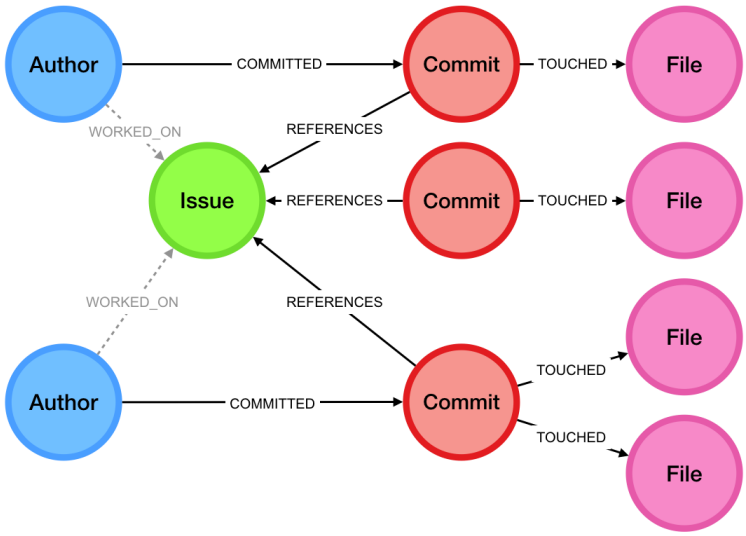


Figure 4.5: Graph structure of the result of a "Search By CIT" query.

```
MATCH (i:Issue { reference: {ref}})<-[:REFERENCES]-(c:Commit) ...
     ... <-[:COMMITTED]-(a:Author)
OPTIONAL MATCH (c)-[:TOUCHED]->(f:File)
OPTIONAL MATCH (i)-[*]->(i2:Issue)
RETURN i,c,a,f,i2;
```

The graph that visualises this search is shown in Figure 4.5. The node that is queried first is the green Issue node. From there the Commits that reference the Issue are retrieved. For these commits the Files touched and their Authors can be obtained, resulting in the final result of the query.

Combining the three searches there are multiple ways to dig through the available data. Firstly, users can search by Label, giving a complete overview of a Feature, all its Files, Authors and Issues. Additionally it shows Features that are (closely) related to the current Feature. Secondly there is the ability to search by Author (Developer). This allows one to have a broad overview of the expertise of a developer, for example to determine whether the person is the best fit to answer some questions. Lastly, there is the search by CIT. This allows a developer to quickly dive into a pretty specific part of the codebase. The aim is to quickly show what happened to solve particular issues, like bug fixes.

4.6 Final Interface

The complete interface (Figure 4.6) now consists of the following components;

1. Search box
2. Info box

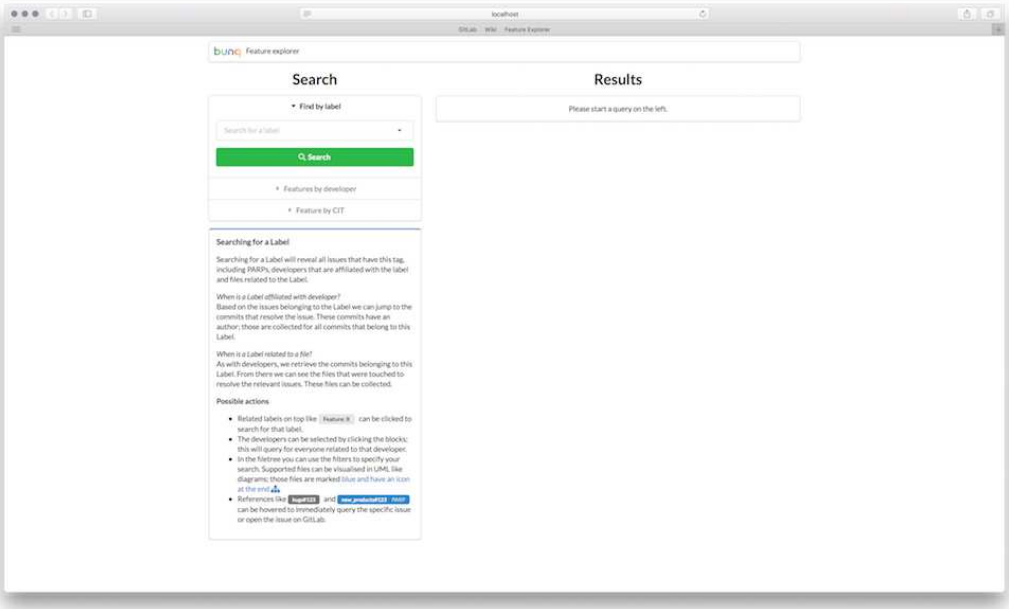


Figure 4.6: Overview of the complete interface.

3. Result window

The components will be discussed in more detail below. The main part of the view is reserved for the results (the right part of the screen). The left column is used for the controls and explanations. When the user performs a "Search By Label" query, the results will be displayed in the right-hand side section of the screen. An example is visible in Figure 4.7. It shows the related labels, the Authors (developers), the related files and the related issues. When the user performs a "Search By Developer" query, the results will show in the same section. Instead, now it will show the Issues worked on by this Author and all Labels that this Author worked on, as can be seen in Figure 4.8. Lastly, there is the "Search By CIT", visualised in Figure 4.9. It contains the same information as searching by a Label, yet, retrieved with a different query, and its scope is more specific.

4.7 Pilot study

A pilot study was performed to evaluate the experiment. This pilot was performed by a single person, due to limited availability of test subjects. This person was comparable with the targeted audience for the experiment, specifically the person is an external developer. The target audience will be discussed in more detail in Section 5.4. It consisted of the full experiment, including the interview in the end. From this pilot no peculiarities regarding the experiment itself came to light. Only a connectivity issue occurred because the pilot was conducted off-site. This has been resolved before performing the actual experiments.

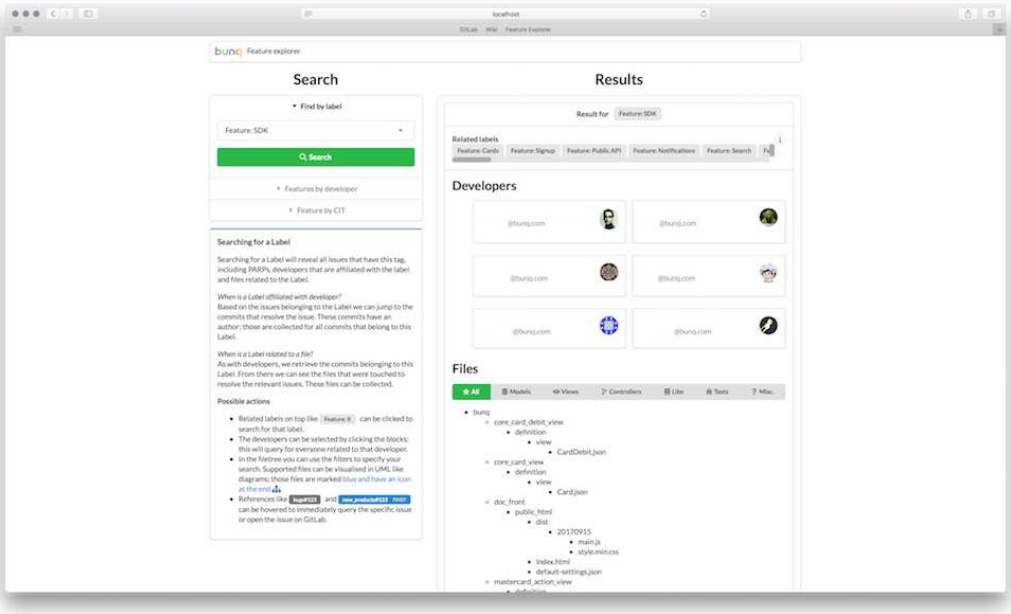


Figure 4.7: Overview of the complete interface with a Search By Label result overview.

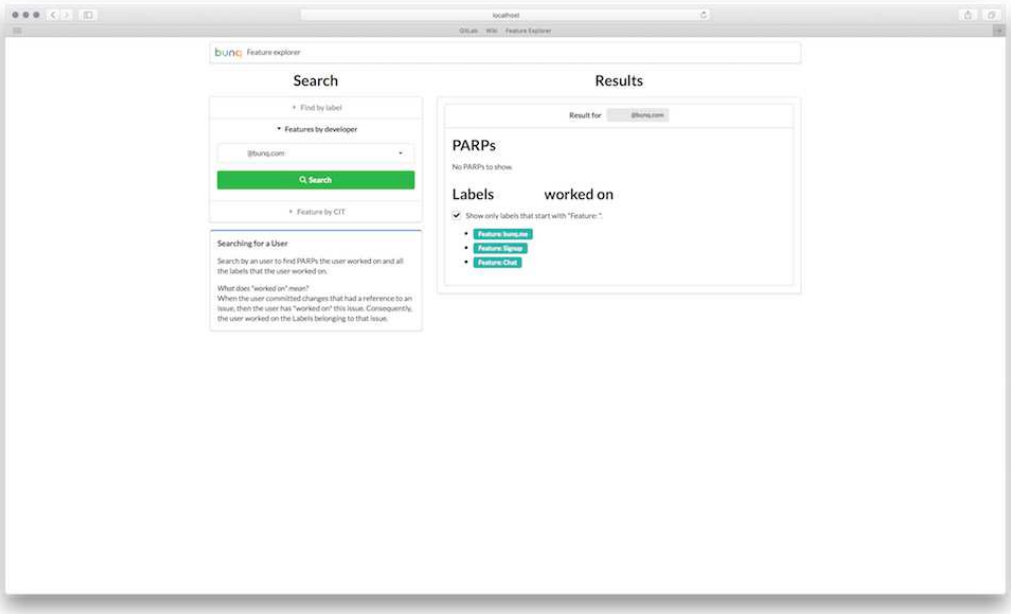


Figure 4.8: Overview of the complete interface with a Search By Developer result overview.

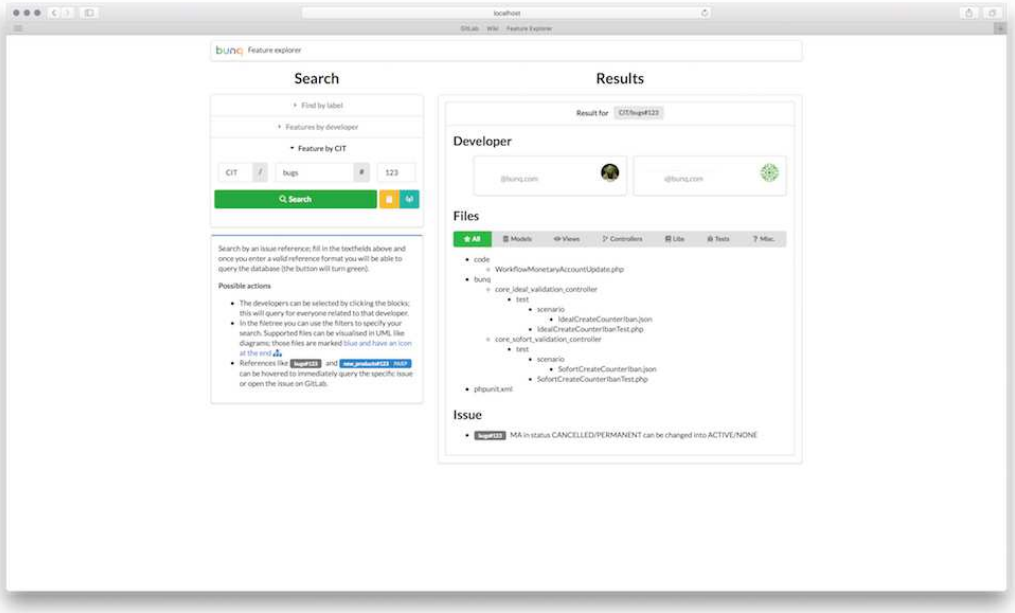


Figure 4.9: Overview of the complete interface with a Search By CIT result overview.

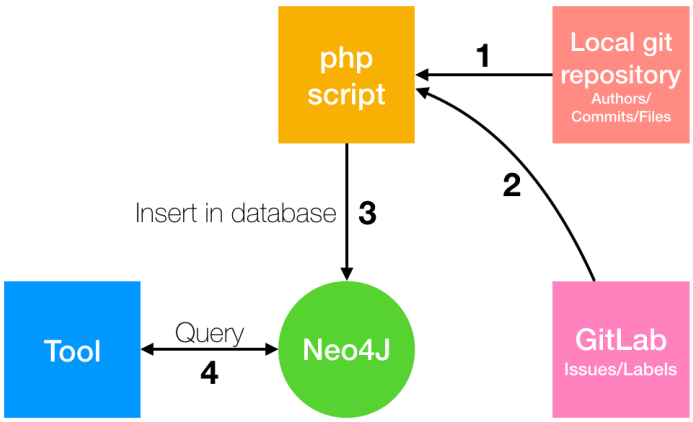


Figure 4.10: Overview of the complete interface with a Search By CIT result overview.

Furthermore the experiment setup has been discussed with both supervisors. This resulted in some small comments on the assignment text (mostly on clarity), which were then improved.

4.8 Summary

An high-level overview of the resulting system is depicted in Figure 4.10. First, the PHP script collects the starting information from a local clone of the backend repository. This process has been explained in Section 4.2. The data retrieved in the first step is then enriched with information from GitLab as explained in Section 4.3. When all data is available the data is uploaded into the database, following the structure as decided upon in Section 4.4.4. Finally, when the database is populated and all possible nodes have been enriched with data from GitLab, the tool is able to query the database and show the information to the user. This has been explained in Section 4.5.3.

Chapter 5

Experiment design

In order to answer the research questions the tool will be evaluated by means of an experiment. This experiment will be used to gather feedback on the tool. In this chapter the type of experiment will be discussed, as well as its setup and the assignment that was designed for this experiment.

5.1 Type of Experiment

To see what developers think of the tool an experiment has been set up. The choice was made to perform a qualitative research rather than a quantitative research. This choice was made for multiple reasons. Firstly, because the number of available developers at bunq is limited and external participants would need to sign the strict Non-Disclosure Agreement because of the sensitiveness of the data/code at bunq. Secondly, a qualitative research allows us to develop a deeper understanding of the problem [1]. Instead, focus will be put on feedback to test this prototype and identify possible points of improvement. The tool will be presented to people currently working at bunq (either as a developer or at least with some coding affinity). Furthermore, people that might be working at bunq in the future will be considered. These people will have to learn the framework from scratch, and as such are ideal candidates to see whether the tool helps to get insight into the codebase. The experiment will be done individually as to not influence each other. Furthermore, a fixed environment will be provided to ensure that all developers have the same tools/settings available at the start. This environment will be reset for every experiment run. Due to the limited number of developers available at bunq the decision was made to focus on getting feedback on the tool. For the feedback we conducted interviews with the subjects. The aim of the interview is to get an impression of the developers' thoughts while working with the tool and its usefulness for them.

5.2 Assignment design

The experiment was guided by an assignment, of which a copy can be found in Appendix A. The subject first was guided through an exploration assignment, to get familiar with the tool. Then, the subject had to make a change to an isolated feature within the bunq backend. Due to the fact that the author of this research developed such a feature within the backend, that feature was selected. Using this feature would ensure that its scope was isolated, it touches only a limited number of other features (less than 5). Furthermore the data had been labelled by the author himself, so there would be no dependency on the quality of labels of others. This would ensure that for this feature the accuracy of the tool would be as high as possible. The feature that was selected is a Payment Method, used to authorize online payments. The subjects were asked to change the View of one of the endpoints (the JSON response that is returned by the API) to also include the name of the user that initiated the payment. This required the user to look at the existing Models and Controllers to figure out where the reference to a User was stored. Additionally the subject would need to find at what point in the execution flow the name should be retrieved and find where to add the name to the resulting JSON output of the endpoint. The third (and last) part of the experiment was a design assignment. After acquiring some knowledge of the bunq backend already, they were tasked to see what elements would be necessary to implement a similar endpoint. The aim being to have them dive into the system and see the relations between components.

In between the second and third parts there was a quick (5 to 10 minutes) interview; basically asking the subject what (s)he had done so far and some early on feedback. This was done to avoid the subject mixing up things for the second and third assignment. Furthermore it could be used to see whether subjects were actually using the tool or not. When they were not they were asked/hinted to use the tool slightly more and if that wouldn't work, then provide reasons as to why the tool did not help them. This way we ensured that we could also know why the tool would *not* help developers.

Furthermore, as some of the participants had no experience at all with backend development, they had an assignment sheet available. This sheet contained an basic explanation on the setup of the backend. It explained, high-level, the flow from an actual API call arriving at the servers to sending back the response. It also contained a small explanation on CIT and the reference to it as used throughout the tool and backend.

5.3 Interviewing method

There are plenty of methods that can be used to perform interviews. We will consider multiple methodologies to structurally analyse the interviews as defined by Babbie [2, 32, 27].

Structured interviewing Although easier to analyse it does not allow flexibility in the order and questions of the interview (e.g. asking follow-up questions or having a discussion).

Self-completion questionnaires Convenient for subjects and no interviewer effects, yet you can not ask questions and there is a greater risk of missing data.

Structured observation Direct observation of what a user does provides insights in how the tool can be used, but ensuring reliability can be more problematic.

Content analysis This methodology is about how humans communicate, which is irrelevant for this study.

Secondary Data Analysis Requires access to data by others, which for a closed environment like bunq, is unsuitable.

Ethnography / participant observation Although this provides more understanding as to what a subject does rather than what (s)he says (s)he does, it can be unpleasant and impractical.

Semi- or unstructured interview Flexible, deeper understanding and detailed qualitative data. Difficult to replicate/compare however due to uniqueness of interviews.

Focus groups Requires groups, which can suffer from 'group effects' (dominance / groupthink). The subject pool is relatively small however.

Conversation or discourse analysis Easy to collect too much data and there are problems when transcribing recorded conversations.

Documentary analysis Data should already be available, which is not the case. Qualitative content is often accused of being subjective or biased.

Systematic review Avoids selective bias but data must already be available.

Narrative review Limited scientific value because of selective bias.

In the end, the interviews have been set up following the Semi- or unstructured interviewing method. Upfront, a question list was prepared with questions that should be asked, at least. Yet, the flexibility of this method allowed the interviewer to ask more questions, to get a deeper understanding of the subjects' opinions. The interviews have been recorded and transcribed manually by the interviewer to make sure that the interpretation of the recordings only has a small effect. The questions that were asked during the interview are;

1. How did you like working on the assignment?
   To get an overall impression of the appreciation of the assignment.
2. Have you used the "Find By Label" feature; why (not)?
   To get specific feedback on the feature.
3. Have you used the "Find By Developer" feature; why (not)? *To get specific feedback on the feature.*
4. Have you used the "Find By CIT" feature; why (not)?
   To get specific feedback on the feature.

5. Were there any benefits to using the tool, if so, which one(s)/why?
   To find out what people valued in the tool.
6. What should have been different in the tool and why?
   To find out what people missed/thought was wrong in the tool.
7. Do you have any general feedback regarding the tool?
   To give the ability to give any other feedback.
8. What is your impression of the bunq backend?
   To find out what the first thought are when seeing the codebase and find points of improvement
9. Do you think it is feasible to implement/improve features using this tool, without the help of senior developers? Why?
   To find out whether developers would be open to replacing senior developer guidance with a tool.
10. What are, according to you, the most important sources of information when learning an (big) existing code base? Why?
    To find new inspiration for information sources that could enrich the current tool.

In the list above, each question has its goal listed. From these goals the questions have been formulated. The end goal was to get feedback on the tool, and specifically per feature, hence questions two to seven. Additionally some questions have been asked to understand a bit more about possible other factors that could influence the result (unclear assignment for instance) and questions to find possible points of improvement. Together these questions will provide a complete insight in how the subjects experienced the tool.

5.4 Experiment audience

The tool is targeted to relatively new developers at bunq and those that have not worked there yet. bunq recruits (among others) from university campuses, for this reason some recently-graduated / near-graduation Master students of Computer Science have been asked to participate. At the time of the research there were multiple Computer Science interns active within bunq. Those have been asked to participate too. Furthermore all employees at bunq that did coding have been asked to do the assignment and give their feedback. Having these differences in experience with the bunq backend allows to see whether the tool could potentially also be used for internal purposes (like an information database for instance).

In the end the following persons were interviewed;

External (no backend knowledge). 2.

Interns (limited to none backend knowledge, $<$10 weeks at bunq). 8.

Juniors (Basic to extensive backend knowledge, $>$10 weeks at bunq but $<$1 year). 8

Seniors (Extensive backend knowledge, $>$1 year at bunq). 2.

5.5 Setup of Experiment

The experiment consisted of a few stages. First there was a pre-survey to be filled in. Then the subjects could start with the first part of the assignment. After roughly 30 minutes the subjects would wrap up and tell the interviewer what they have done and how they came to their solution. Then, they started with the design part. Approximately 45 minutes could be spend on this. Afterwards there was room for 30 to 45 minutes of interview. All interviews have been recorded (after asking permission).

The experiment was conducted on a dedicated laptop. The browser/editor/tool would all be reset at the beginning of each session. Code changes had been moved to a hidden git repository, and the working copy of the project had been reverted to a clean slate. All subjects started with the same commit being the "clean slate". Furthermore the database was reset for every subject. This ensured the same environment for all subjects.

The experiments were conducted in closed meeting rooms, or otherwise rooms where (barely) no interferences were made. The interviewer was present all the time, but across the table and not observing the subject actively. The idea here is to let the subjects feel as comfortable as possible. The subjects would have access to the tool, a code editor PhpStorm (set up with the backend code), Sequel Pro (database tool) and GitLab (to see more details of issues).

5.6 Analysing interview feedback

In order to structurally analyse the feedback, the feedback has been sorted in different categories. Reading over all the interview transcripts, feedback was highlighted using different marker colors.

Tool Any feedback on the tool itself, having sub categories;

- **Feature requests** For things that are currently missing
- **Search By Label** For all feedback on searches performed on Labels
- **Search By Developer** For all feedback on searches performed on Developers
- **Search By CIT** For all feedback on searches performed on CIT references

Assignment Feedback on the actual assignment (duration, difficulty, assignment text, etc.)

Other Containing all other feedback, including subcategories;

- **bunq** Feedback related to bunq's code, methodologies, etc.
- **code** Feedback related to coding practices and other general coding aspects.

After this initial highlighting all feedback points have been written on small cards and sorted according to their category. At this point we also re-evaluated whether the item really fitted with the category. This resulted in multiple feedback points in every category, which then were grouped by an open card sorting strategy [38]. This was done twice, once by the

author and once by another employee at bunq, that did not participate in the experiment nor the interviews. The two sets were compared and discussed to converge to a jointly agreed solution.

Chapter 6

Results

The results of the experiments will be presented in this chapter. First we will look at the general feedback that was given, we will list the most frequently occurring categories and the number of feedback points we received in those categories. Likewise we will iterate through the feedback for the specific features in the tool and the feedback regarding shortcomings/possible improvements of the tool. Lastly, we will take a look at the answers that we received when we asked developers whether they think the tool could replace senior developers to address their questions to. The transcripts of the interviews can be found in Appendix C.

6.1 General feedback

All feedback that concerned the tool as a whole and all feedback that was related to multiple features is considered 'general feedback'. This general feedback can be spread over the following categories;

- **Overview of the backend** (19x)
  Where subjects mentioned that the tool created overview of the backend.
- **Visualisations of worksflows being useful** (12x)
  Where subjects indicated that the visualisations were useful.
- **Seeing what belongs together** (9x)
  Where subjects indicated that the tool showed them what belongs together.
- **Where subjects indicated that the tool was useful** (7x)
  Where subjects gave feedback on the usefulness for new developers vs. more experienced developers.
- **Inability to determine where to acquire a certain model instance** (6x)
  Where subjects mentioned the inability to find how to retrieve a specific model that they needed.

- **Other**
  Any other general feedback.

The biggest category of general feedback was on having the overview of the backend. Most subjects agreed that the tool could provide an useful overview of the relevant parts of the backend. The feedback also indicated that the tool might be more useful for new developers than for experienced developers. Those that said to prefer to use the IDE were typically the developers that are used to bunq's backend. Especially the newer developers indicated that it would have helped them in their first days. Part of the overview that the tool provided was the ability to discover non-obvious relationships in the code. When the name of a feature is present in the filename it is easy to find them using a simple file search. However, when the filename does not at all include the feature name, it might be harder to see that the file belongs to a feature.

6.2 Feedback per feature

In this section we will evaluate the three main features within the tool. Knowingly "Search by Label", "Search by Developer" and "Search by CIT". For each of the features we will list the categories as found after the card-sorting phase and give a summary of the feedback and peculiarities in the data.

6.2.1 Search by Label

After the card sorting, the following feedback categories have been identified;

- **Acquiring an overview** (14x)
  Where subjects mentioned the ability to get an overview of the relevant files/classes for a feature.
- **Not all Features are properly labelled** (5x)
  Where subjects noted that the quality of the results was dependent on the Label and the application of that label within GitLab.
- **Related issues are unclear** (3x)
  Where subjects indicated that they did not understand the related issues (e.g. How are they determined? What does it mean?).
- **Other**
  Any other feedback on the Search by Label feature.

The biggest feedback category for this part was the "Acquiring an overview" category. Approximately half of the subjects indicated that the tool gave them an overview with the information (classes mostly) they required at that point. The biggest relevant remark was that the quality of labels could influence the quality of the results given back by the tool. For instance, the iDeal feature has been labelled less consistent than the [redacted] feature. Hence, those searching for [redacted] would be able to retrieve most to all relevant files,

whereas those who searched for iDeal could not find all relevant files as not everything was labelled. The remaining categories are small non-functional changes mostly; unclarity about related labels, information that was not shown like authors of specific files and the need to explicitly click a search button after making a selection in a dropdown. The labelling is something that will need attention, as the tool strongly depends on the quality of the labels.

6.2.2 Search by Developer

After the card sorting, the following feedback categories have been identified;

- **No added value** (14x)
  Where subjects indicated that they did not see added value for this feature or did not know how to use it.
- **Fallback option to other search methods** (6x)
  Where subjects indicated they would use this option as a fallback for the other search options.
- **Information is not specific enough** (5x)
  Where subjects indicated that they received too much information / non-specific information.
- **Tracking progress** (4x)
  Where subjects indicated that the feature could be used to keep track of the progress of tasks/projects.
- **Other**
  Any other feedback on the Search by Developer feature.

The biggest category out of these listed above was “No added value”. People could not find specific information in the results of this search, and hence did not use it. The tool would spit out all features a developer had ever worked on. Especially for the senior developers at bunq this would result in a long list of features. There were no additional filter possibilities to narrow it down, so subjects could not find what they were looking for. Some subjects provided suggestions of possible use cases, that are worth further exploration. These include using the results to determine the next tasks for a person and showing the most recent changes.

6.2.3 Search by CIT

The following categories have been identified during the card sorting phase;

- **Specific overview of an individual issue** (13x)
  Subjects indicated that they could obtain an overview of one specific issue and the related files.
- **Requirement for CIT information** (6x)
  Subjects indicated that they needed to know the CIT reference upfront.

- **Not more information than Feature by Label** (5x)
  Subjects indicated that this feature did not provide more information than the Feature by Label search.

- **Other**
  Any other feedback on the Search by CIT feature.

The biggest category here was the one where people explained that they could get more specific information out of the tool (compared to the other search options). For instance, the Search by Label would return all files related to the entire feature. In the case of a small change to a feature one might want to lookup the issue reference instead and only see the files affected for this small change. The drawback according to some, was that it required the user to know the exact CIT reference. Some argued that the more specific view did not provide a full overview, while others argued that this was nice for bug hunting.

6.3 Shortcomings and improvements

After categorising the feedback regarding shortcomings and possible improvements we found the following;

- **Missing documentation/explanations** (11x)
  Multiple subjects indicated that they would like some short explanations with the entities that are shown in the tool. For instance, not all Workflows/Models have clear names. The subjects would like to see a short explanation then of what the purpose of a class is.

- **More/other search possibilities** (11x)
  Some subjects proposed new search options, like “Search by Endpoint name”.

- **See more details about Models and Views** (10x)
  Subjects indicated that they would like visualisations for the Models and Views as well and not just for the Workflows.

- **Combining the search options** (7x)
  Subjects wanted to see more fine-grained search options. The current search options could result in big overviews, with too many files. The subjects indicated that they wanted to combine searches, like a Label and then select a Developer, to filter down the results even more.

- **Visualisation of entire request flow** (4x)
  Subjects indicated that they would like to see the processing of an entire API call visualised, from received request to sent response.

- **Seeing actual changes (diff)** (4x)
  Subjects indicated that they wanted to be able to see *what* changed, rather than what files were touched.

- **No immediate search initiated after making a selection in the drop-down** (3x)
  Where subjects indicated they expected a search to be performed, right after making a selection in the dropdown. Currently they have to click the Search button explicitly.

- **Other**
  Any other shortcomings and improvements.

A commonly requested option was the ability to combine search options. Subjects wanted to narrow down their searches even more than what was currently possible, to get more specific information. Also, a common remark was that the naming of elements was not always clear, so a short description (taken from documentation where possible) could help understand what an element is about or what its function is. The visualisations should be expanded upon, to include the entire flow instead of just the workflows. The end-to-end visualisations should provide a better overview of the entire process.

6.4 Can the Tool replace the senior regarding question asking?

After card sorting we found the following categories of feedback;

- **It can not replace all questions** (10x)
  Subjects indicated that they think the tool will not be able to replace *all* types of questions.

- **It can not replace proper mentoring** (6x)
  Subjects indicated that they think that the tool will not be able to replace the senior developer when it comes to giving coding advise (structure / style).

- **Still need a starting point** (6x)
  Subjects indicated that with this tool they would still need to know first what to search for.

- **Absence of Natural Language Processing in a tool** (5x)
  Subjects indicated that the tool is incapable of parsing Natural Language. This means that the tool could misinterpret a question and that the tool might not be able to explain the answer in such a way that the subject understands it.

- **Other**
  Any other feedback on the tool as a replacement for senior developers regarding question asking.

Almost all subjects indicated that seniors could not be replaced (completely). Most agreed that the tool could take away some questions about exploring the codebase, yet, the seniors are still needed (for a starting point or mentoring for instance). One point where the seniors are in favour is when it comes to asking for help and getting explanations, multiple subjects indicated that the tool is a statical tool, not able to parse Natural Languages and not able to explain something as a human could do.

Chapter 7

Discussion

This chapter will discuss the results as well as the validity and reliability of this study. Also, an advise for bunq will be presented, based on the findings in this study.

7.1 Answering the Research Questions

Looking at RQ1, we want to know the requirements that the tool should meet to help new developers at bunq understanding the business logics. The initial requirements as designed in Section 4.1 were not sufficient yet. The feedback showed that developers like the visualisations and overview, but it needs to be expanded. Especially the database structure (Models) is something that was mentioned often. Subjects did not know where they could retrieve the specific instances of a Model, and could not find confirmation for that in the tool. Furthermore, a much requested feature is to visualise an entire flow, rather than just the workflow. These requested visualisations are different than most common software visualisations that are around including tools that show change frequencies [14] and software evolution [28] as we have seen in Chapter 3.

RQ1 The current prototype of the tool is helpful to understand the business logics (workflows). However, to increase the understanding, expansion are necessary (e.g. bigger scope for visualisations, improved searching/filtering)

For RQ2 we asked whether subjects think the tool can replace a senior developer when it comes to asking questions. The majority did not think the tool would be able to do so. The most common arguments being that a tool would not be able to help you when stuck or when you need advise/tutoring. Also, sometimes it is easier to explain your problem in spoken language and the same holds for getting an answer. Since the tool is not capable of Natural Language Processing, this is a drawback.

RQ2 Developers do not think a tool like the prototype is able to replace a senior when it comes to asking questions. At most, the tool would be able to take away some basic questions, when the developers do have a starting point already. If not, it should be provided in some way or by someone.

For RQ3 we looked at the influence of a developer's experience on the usefulness of the tool as perceived by said developer. From the results we can see that the (experienced) senior developers did not think that the tool would help them much. However, a noteworthy point made by experiment Subject 20 (see Section C.20), was that the tool could potentially be used to investigate pitfalls for a developer and identify areas that require targeted guidance. For instance, if numerous bugs are introduced in model relationship, a senior could sit with the developer and give an explanation on the material. This allows seniors to be more targeted as they can see from the (adjusted) tool where most bugs arise for a developer. Other than that the experienced developers mostly indicated that they could have done it without the tool.

RQ3 The usefulness of the tool degrades with the experience that a developer gains over time. When the structure of the backend is known to developers they tend to argue that navigation is straight forward. However, for new developers they claim that it could be very useful to have.

Concluding we can see that the tool mostly has potential for new developers, when they are exploring the backend. For more experienced developers it becomes less useful as they will know better how to navigate throughout the project and where to find their files.

7.2 Reflection on the Assignment

The assignment (as in Appendix A) was checked by two supervisors and the CEO of bunq. However, still we noticed that the subjects seemed really pushed towards using the Find By Label feature. The foremost reason seems to be that the explicit feature name immediately gave a hit in the auto-completing dropdown for the Label search. It would have been better to not state the explicit feature name or at least give equally explicit entry points to do the other searches. Also, the Find By Label search box was the default one, when none would be expanded by default this might influence the choice of which search option to use.

The assignment could have been more explicit about the usage of the tool; three out of nineteen participants did not realise they were expected to use the tool for the second and third part of the assignment. They were give hints to use it more in the "in-between" interviews, as mentioned in Section 5.2.

For the third part of the assignment subjects were asked to design a fictive feature called "AliPay", which could be based on another existing feature. The idea was that subjects would explore the full existing feature to see whether they would be able to find all related code. However, due to the formulation of "design AliPay", subjects did not know exactly at what level they needed to design this feature. As a consequence some only looked into

Workflows while others looked as far as the method level, naming some of the methods that would be needed to implement the feature.

Last remark is about the explicitness of the endpoint, the full endpoint name was given in the assignment. However, this name is very similar to the folder that contained the Models/Views/Controllers. By scrolling in the list subjects could relatively easily find the folders for the feature with its code. It would have been better to choose an endpoint that could not be found this easily.

7.3 Reflection on the Interviews

For the interviews there are two main flaws that can influence the data as obtained; firstly the interviewer effect and secondly interpretation errors. The interviewer effect [41], where the interviewer's way of asking question can be influencing the responses, might have played a role in the obtained responses. Especially considering the semi-structured setup, all interviews were unique but comparable. Furthermore there is the interpretation effect; where the interviewer does simply not correctly understand a subjects' response. This has been avoided by repeating the response in the interviewers interpretation/words, and asking for confirmation whether that is what was meant by the subject.

7.4 Analysing the Results

When analysing the results, the feedback was categorised. This categorisation process has been done by two persons, the interviewer and another person that was not involved in the interviewing process or the experiment. There is a slim chance that both of them placed feedback in a different category then someone else might do. Given enough time an extensive inter-rater agreement analysis can be performed to gain insight in the rate of agreement between the two persons [22].

7.5 Limitations

As there is a limited inflow of new developers at bunq (some months no new developers start) it is hard to obtain enough data for a quantitative study. Also, developers at bunq are typically very busy, meaning they can not easily participate in long experiments. This is especially true for the senior developers. Due to these time constraints only two senior developers have been interviewed about the tool, in a slightly altered setting. As they have already more than a year of experience with the backend the implementation assignment was skipped as it would not make sense to let them spend their time on that. Instead, they were introduced to the tool and then asked for their opinion on it, following the same question list as for the regular interviews.

7.6 Advise to bunq

Based on the result as discussed earlier in this chapter, the advise to bunq would be to use the assignment as developed for this study as an introduction assignment for new developers. Because of the predefined scope of this assignment it can be used to get developers familiarised with the concepts in the bunq backend, while knowing what the outcome should be. The assignment would need some tweaks to include more parts of the backend as currently it focusses somewhat on the Views. Furthermore the assignment text (the explanation on the backend structure) can be shared with developers to provide a high-level overview of the architecture of the backend. This was missing up until now. Lastly, if such an introduction assignment were to be introduced, it would be useful for a new developer to go over the outcome with a senior developer. This way any unclarities can be taken away immediately.

Chapter 8

Conclusions and Future Work

The results of this research, as discussed in Chapter 7, will be summarised in this chapter. Afterwards some ideas for future work will be discussed.

8.1 Conclusions

In this study a prototype has been developed to see what developers thought about a tool that could help them with development. We looked at their needs based on the experiences of recently joined developers. From there we developed a prototype. This prototype aimed to give an insight in the files related to a feature, the developers of a feature, the features that relate to it and furthermore it wants to provide insight into the business logic components (workflows) that are used at bunq. The prototype was already considered useful, especially for new developers, but also requires expansion to be able to use it in daily work.

Senior developers at bunq fulfil multiple roles. Not only do spend most of their time writing code, they also have to teach the junior developers next to that. For the guidance they can possibly make use of the tool, as proposed by one of the senior developers. They could use it to get an insight into the parts of the backend that need extra explanation.

The developers think the tool can not replace a senior developer when it comes to asking questions. Yet, the tool can reduce the amount of questions when sufficient information is shown, as discussed in Section 7.1. Things that can not be replaced according to the developers are the mentoring and guidance. Especially when resolving specific problems the developers indicate that they would still need a clue or pointer to be able to use the tool currently. The tool then can help them understand the structure of a feature, but not necessarily where something is wrong.

8.2 Future work

Future studies can investigate the actual effect of a tool as described in this study. Currently, only a qualitative research has been performed due to limited time and the limited inflow

of new developers at bunq. A more extensive study that compares the onboarding time of developers with and without this tool can be used to measure the actual effect. Apart from a more extensive study, possible alterations / improvements of the tool can be looked into as well. One improvement would be to visualise the Models and Views as well, providing a full visual overview of what happens after a certain API call. Other possibilities include the more detailed filtering (combining filter options) and integrating actual source code (diffs for instance) into the tool. These options could enhance the usefulness of the tool. It would be interesting as well to talk with more teams aside the Software Development team. For instance, it was mentioned by some, that the Product team might be able to make use of this tool to assign tasks to the most suitable person. Or to get an overview of what is going on. This might, indirectly, also cause the software developers to waste less time on issues that could better be solved by someone else.

Bibliography

[1] Karim Abawi. Qualitative and quantitative research. *World Health*, 2008.

[2] Earl R Babbie. *The practice of social research*, volume 112. Wadsworth publishing company Belmont, CA, 1998.

[3] Banken.nl. Nieuwe bancaire speler in nederland: Bunq. http://www.banken.nl/nieuws/3653/nieuwe-bancaire-speler-in-nederland-bunq Accessed: 2018-10-04.

[4] Andrew Begel and Beth Simon. Novice software developers, all over again. In *Proceeding of the fourth international workshop on Computing education research - ICER '08*, pages 3–14, 2008.

[5] Andrew Begel, Microsoft Way, and Beth Simon. Struggles of New College Graduates in their First Software Development Job. pages 226–230, 2008.

[6] Carl Binder, Elizabeth Haughton, and Barbara Bateman. Fluency: Achieving true mastery in the learning process. *Professional Papers in special education*, pages 2–20, 2002.

[7] Steve Burbeck. Applications programming in smalltalk-80 (tm): How to use model-view-controller (mvc). *Smalltalk-80 v2*, 5:1–11, 1992.

[8] Xiaofan Chen, John Hosking, John Grundy, and Robert Amor. DCTracVis: a system retrieving and visualizing traceability links between source code and documentation. *Automated Software Engineering*, pages 1–39, 2018.

[9] Jane Cleland-Huang, C.K. Chang, and Mark Christensen. Event-based traceability for managing evolutionary change. *IEEE Transactions on Software Engineering*, 29(9):796–810, sep 2003.

[10] Bas Cornelissen, Danny Holten, Andy Zaidman, Leon Moonen, Jarke J Van Wijk, and Arie Van Deursen. Understanding execution traces using massive sequence and

circular bundle views. In *Program Comprehension, 2007. ICPC'07. 15th IEEE International Conference on*, pages 49–58. IEEE, 2007.

[11] Bas Cornelissen, Andy Zaidman, and Arie Van Deursen. A controlled experiment for program comprehension through trace visualization. *IEEE Transactions on Software Engineering*, 37(3):341–355, 2011.

[12] Bas Cornelissen, Andy Zaidman, Arie van Deursen, and Rainer Koschke. A Systematic Survey of Program Comprehension through Dynamic Analysis. *IEEE Transactions on Engineering Management*, 35(2):684–702, 2009.

[13] Barthélémy Dagenais, Harold Ossher, Rachel K E Bellamy, Martin P Robillard, and Jacqueline P de Vries. Moving into a new software project landscape. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, volume 1, page 275, New York, New York, USA, 2010. ACM Press.

[14] Wim De Pauw, Richard Helm, Doug Kimelman, and John Vlissides. Visualizing the behavior of object-oriented systems. *ACM SIGPLAN Notices*, 28(10):326–337, 1993.

[15] Giuseppe A. Di Lucca and Massimiliano Di Penta. Integrating static and dynamic analysis to improve the comprehension of existing web applications. *Proceedings - Seventh IEEE International Symposium on Web Site Evolution, WSE 2005*, 2005:87–94, 2005.

[16] Bogdan Dit, Meghan Revelle, Malcom Gethers, and Denys Poshyvanyk. Feature location in source code: a taxonomy and survey. *Journal of Software: Evolution and Process*, 25(1):53–95, jan 2013.

[17] Dennis Edwards, Norman Wilde, Sharon Simmons, and Eric Golden. Instrumenting time-sensitive software for feature location. *IEEE International Conference on Program Comprehension*, pages 130–137, 2009.

[18] Alexander Egyed, Gernot Binder, and Paul Grunbacher. STRADA: A Tool for Scenario-Based Feature-to-Code Trace Detection and Analysis. In *29th International Conference on Software Engineering (ICSE'07 Companion)*, pages 41–42. IEEE, may 2007.

[19] Alexander Fronk, Armin Bruckhoff, and Michael Kern. 3d visualisation of code structures in java software systems. In *Proceedings of the 2006 ACM symposium on Software visualization*, pages 145–146. ACM, 2006.

[20] Orlena C Gotel, Anthony C W Finkelstein, and London Sw. An Analysis of the Requirements Traceability Problem Imperial College of Science , Technology & Medicine Department of Computing , 180 Queen ' s Gate. pages 94–101, 1994.

[21] The Open Group. Base specifications issue 7, 2018 edition. http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html Accessed: 2018-09-24.

[22] Kevin A Hallgren. Computing inter-rater reliability for observational data: an overview and tutorial. *Tutorials in quantitative methods for psychology*, 8(1):23, 2012.

[23] S. Huizinga and T. van de Put. *Breken met banken*. Ronde Tafel, SU De, 2016.

[24] Huzefa Kagdi, J.I. Maletic, and Bonita Sharif. Mining software repositories for traceability links. In *15th IEEE International Conference on Program Comprehension (ICPC '07)*, number January, pages 145–154. IEEE, jun 2007.

[25] Tien Duy B. Le, Mario Linares-Vásquez, David Lo, and Denys Poshyvanyk. RCLinker: Automated Linking of Issue Reports and Commits Leveraging Rich Contextual Information. *IEEE International Conference on Program Comprehension*, 2015-Augus:36–47, 2015.

[26] Dapeng Liu, Andrian Marcus, Denys Poshyvanyk, and Vaclav Rajlich. Feature location via information retrieval based filtering of a single scenario execution trace. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering - ASE '07*, page 234, New York, New York, USA, 2007. ACM Press.

[27] Rik Loijmans. Overzichtsartikelen. *Huisarts en wetenschap*, 57(10):519–519, 2014.

[28] Gerard Lommerse, Freek Nossin, Lucian Voinea, and Alexandru Telea. The Visual Code Navigator: An interactive toolset for source code investigation. *Proceedings - IEEE Symposium on Information Visualization, INFO VIS*, pages 25–32, 2005.

[29] Walid Maalej and Hans-Jörg Happel. Can Development Work Describe Itself? *7th IEEE Working Conference on Mining Software Repositories (MSR 2010)*, pages 191–200, 2010.

[30] Patrick Maes et al. International payments: building a single payment architecture using soa. *Journal of Financial Services Technology, The*, 2(1):71, 2008.

[31] Andrian Marcus, Xinrong Xie, and Denys Poshyvanyk. When and how to visualize traceability links? In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pages 56–61. ACM, 2005.

[32] University of Southern California Libraries. Organizing your social sciences research paper: Types of research designs. https://libguides.usc.edu/c.php?g=235034&p=1559832 Accessed: 2018-09-14.

[33] Rocco Oliveto, Giuliano Antoniol, Andrian Marcus, and Jane Hayes. Software artefact traceability: the never-ending challenge. In *Software Maintenance, 2007. ICSM 2007. IEEE International Conference on*, pages 485–488. IEEE, 2007.

[34] Jonathan W Palmer. Web site usability, design, and performance metrics. *Information systems research*, 13(2):151–167, 2002.

[35] V. Rajlich and N. Wilde. The role of concepts in program comprehension. *Proceedings - IEEE Workshop on Program Comprehension*, 2002-Janua:271–278, 2002.

[36] Václav Rajlich and Prashant Gosavi. Incremental change in object-oriented programming. *IEEE Software*, 21(4):62–69, 2004.

[37] Sukanya Ratanotayanon, Susan Elliott Sim, and Derek J. Raycraft. Cross-artifact traceability using lightweight links. *Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, TEFSE 2009*, pages 57–64, 2009.

[38] Donna Spencer. *Card sorting: Designing usable categories*. Rosenfeld Media, 2009.

[39] Mircea Trifu. Using dataflow information for concern identification in object-oriented software systems. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, pages 193–202, 2008.

[40] Jianguo Wang. Supporting developer-onboarding with enhanced resource finding and visual exploration. 2012.

[41] Brady T West and Annelies G Blom. Explaining interviewer effects: A research synthesis. *Journal of Survey Statistics and Methodology*, 5(2):175–211, 2017.

[42] Minghui Zhou and Audris Mockus. Developer Fluency: Achieving True Mastery in Software Projects. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering - FSE '10*, page 137, New York, New York, USA, 2010. ACM Press.

[43] Minghui Zhou, Audris Mockus, and David Weiss. Learning in offshored and legacy software projects: How product structure shapes organization. In *ICSE Workshop on Socio-Technical Congruence*, 2009.

Appendix A

Experiment - Assignment text

bunq

Backend Exploration Assignment

Assignment for thesis research

Owner: Sander van den oever

Date: 20180622

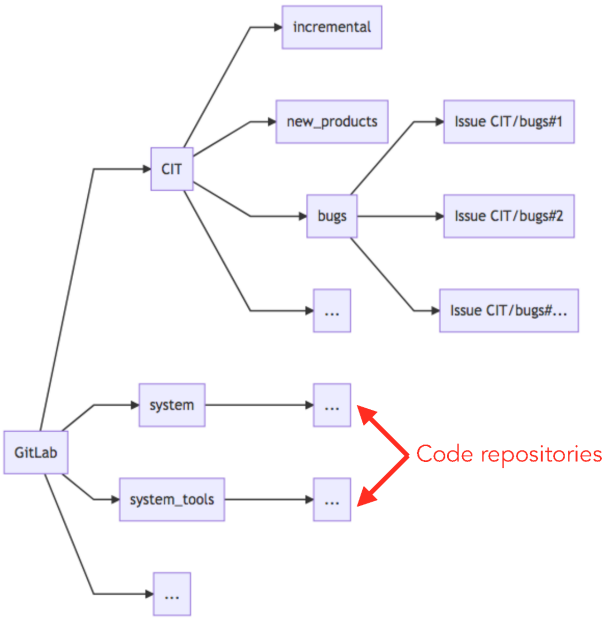
Version number: 1.2

Confidentiality level: **Confidential**

Introduction

The aim of this assignment is to get you familiar with the bunq backend. The backend is a project that consists of several components/modules. You will have the possibility to use a tool that allows you to explore the structure of the backend.

GitLab and the Central Issue Tracking (CIT)
bunq uses a privately hosted GitLab instance (https://gitlab.bunq.net) for its source control and issue tracking. The issue tracking system is also used for documentation / research for new products (`new_products`), incrementally improved features (`incremental`), recurring tasks (`recurring`) and more. CIT exists next to code repositories like '`system`' and '`system_tools`'. CIT holds no code, just the issues. The general structure is as follows;



CIT stands for Central Issue Tracking. This is a central source of information regarding design decisions. All code changes will refer to an issue in one of the CIT projects (`bugs`, `new_products`, etc.). This happens by referencing the issue in the commit messages.

You can (almost) always see a reference to an issue in the commit messages;

The "`CIT/recurring#53`" refers to issue #53 in the group CIT and repository "recurring". The issues on GitLab are typically labelled with "Feature" labels, indicating to what feature an issue belongs. E.g. when there's a bug in Apple Pay an issue exists in "CIT/bugs" which is labeled with "Feature: Apple Pay". Some issues are labelled as **PARP**. This is a special label to indicate bigger projects; the PARP issue is the main issue of the project and commonly all teams have a dedicated sub-issue that links to the PARP. For instance you can have a PARP with "child"-issues for Android, iOS and backend.

Global overview of the back-end

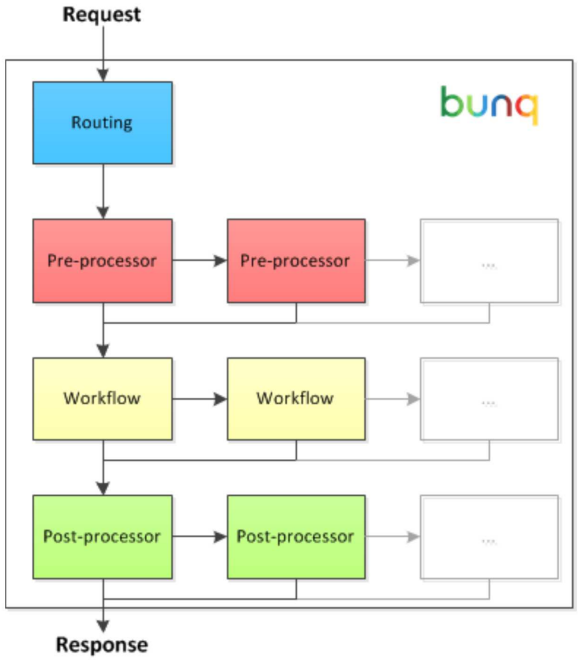
The backend code is stored in GitLab as well (`system` ➡️ `core`). The backend follows an Model-View-Controller pattern.

- **Model** - responsible for database interaction.
- **Controller** - retrieves/manipulates data; within the backend they are called *Workflow*.
- **View** - present data to the user; in bunq's case these are commonly JSON structures.

There are four steps when processing a call to the backend;

- Routing
- Pre-processing
- Running workflows (controllers)
- Post-Processing

What happens when someone sends a request to the backend is that it will first be picked up by a **router**. The router determines the destination of a request (e.g. it will forward a request for iDeal to the appropriate classes). Before a request reaches any important code it will be validated first in one or multiple **pre-processors**. When the request has been validated it is passed on to the appropriate **workflow** (this workflow can call other workflows in turn). Finally, when all workflows have done their bit one or multiple **post-processors** will be ran. This includes formatting the response as expected by the initiator of the request and transforming the data into a JSON structure for instance.



A good thing to be aware of is that a lot of the backend code is generated based on JSON definitions. Every model/controller/view has its own folder within the backend project. E.g. the folder `core_user_model` contains the code and definition for the User model. `core_user_model/definition` contains the JSON definition of a model (properties (columns) and relations to other models). `core_user_model/generated` contains generated code for this definition (code that's really trivial like getters and setters is generated there). Lastly there's a folder `core_user_model/code` contains the actual User model class. A likewise structure is visible for controllers and views.

Generating code / Running tests

To make things a bit easier you can trigger the code generation from within PhpStorm. In the top menu bar (see screenshot) you will see two bunq logos. The left one will (re)generate the PHP code for the currently opened/active file. The right button will (re)generate the PHP for the entire project (this can take somewhat longer). The buttons are marked by a red box in the screenshot. You need to regenerate a JSON every time you make a change to it.

You can also run tests from this menu; one test is pre-configured (green box). The debugger works as well if you'd like to use that during the assignment.

Part 1 - Exploring the Feature Explorer tool

Accessing the code and tools;
- **Code**: use PhpStorm to navigate the code, it's preconfigured to open the right project and the debugger is configured to run the tests.
- **Database**: you can use Sequel Pro to explore the database, it will automatically connect to the right environment when you start it ("bunq_test_data").
- **Feature Explorer**: you can access the tool by opening your browser and going to https://localhost:3000. There's a bookmark available in the browser.

Please look around for the possibilities in the tool to have a basic understanding of what's possible with the tool, try some buttons and click through the different parts/features to familiarise yourself. Try to do the following (no need to write anything down);
- Find out how the feature *Account Closure* has been built; search for the label "Feature: Account Closure",
- Find out what features have been worked on by "Geert-Jan" (search for his name),
- Find out what files were changed to resolve bug "CIT/bugs#1714" (search by CIT).

Please use the Feature Explorer for the following assignments.

Part 2 - Adding a new field to an existing endpoint

You are going to add an additional field to an existing endpoint (= view).

-
-
  -
  -
  -
  -
  -

Please create a branch with the following format [yyyymmdd_name].
Locate the endpoint (view) in the backend and implement all that is needed to show the name of the user in a new field in the endpoint response.

Part 3 - Designing a new payment method

Currently bunq has an implementation to pay using [redacted] iDeal. Next to that we want to support a new payment method, let's call it "AliPay". It works similar to [redacted] and iDeal. The flow is as follows;

- Webshop makes an API call to initiate the AliPay transaction.
- The backend creates a new pending transaction and returns the UUID[1] of it.
- The webshop retrieves the QR code from the backend using the UUID.
- The user scans the QR and the backend assigns the user to the transaction.
- The webshop polls the backend every 5 seconds for the status of the transaction.
- The user confirms/denies the payment.
- As soon as the webshop notices the changed status it will proceed.

Find out what is necessary to implement the above and write it down. **You do not need to actually implement the feature**, a design plan is sufficient. Try to be as complete as possible as to what elements you need to build and how they work together. Try to include what classes you need and describe the most important functions/functionality within the classes.

Hint:

- You can base your design for Part 3 on the current implementations of [redacted] iDeal issuer. [redacted]
- You do not need to focus on optimising the code like removing code duplication, improving class hierarchies etc.

[1] Universal Unique Identifier

Appendix B

Preliminary Interviews

These interviews have been conducted during the initial phase of the research. The aim is to get an impression of the factors that slowed down progress of new software developers at bunq. The questions asked are;

- When did you join bunq?
- What is your educational background?
- What were the main obstacles when working/navigating around in the project?
- What took most time when working on a task?
- What was your approach to resolve issues in the first weeks?
- How does this approach compare to your current approach.

Names and other sensitive information has been hidden, this is indicated by means of squared brackets ([...]).

B.1 Subject 1

Translated from Dutch.

Questioner (Q) *Okay, do you remember when you joined bunq?*

Subject (S) September 17th, if I recall correctly. In any case; halfway September.

Q *So, that would be, around 5-6 months ago?*

S Yes, say half a year ago.

Q *Did you have any prior experience as a developer, and if any, what experience did you have?*

S Yes, I started reading books about development around the age of 12-13. I worked on a lot of personal projects. At the age of 17 I started doing some website maintenance for a research company together with some backend related work, mostly bug fixes. Afterwards I worked with C# for a start-up, where we built a bigger Excel[1] toolbar/training kit. Then I started working for a company that builds MVPs[2] for bigger companies like De Correspondent and MoneYou. Development has always been a hobby and I spent quite some time on reading about it, blog posts, books, just because I find it very interesting. And I also do some projects on the side with some friends.

Q *And what is your educational background?*

S I started Pre-University secondary education[3] first, but did not complete and changed to "higher general continued education"[4]. Then I started with Computer Science at Hogeschool Amsterdam. Currently I'm halfway into that programme.

Q *Then now some more bunq-related question. Please try to think back of your first days here. What were your main struggles when trying to find your way in the codebase?*

S On a really small level, the conventions, that took some time to get familiar with. For instance the explicit "if-nothing-else"[5]. At macro-level it was mostly about the level of coupling. [redacted].

Q *Do you have a explanation for that?*

S I don't know, those are the design choices that have been made in the past. In some way it makes sense. [redacted].

Q *And what did cost you the most time in the first few weeks?*

S In terms of learning curve?

Q *Say, you would get a task, what would you spend most time on?*

S In the beginning I would get very simple tasks, say change TIN patterns[6]. For that a migration[7] was necessary, that was a big obstacle. Also, relations between different entities were not always clear. Also because we can "embed" Models in other Models, you can end up with somewhat confusing database structures.

[1] Microsoft Excel: https://products.office.com/en/excel
[2] Minimum Viable Product ("proof of concept")
[3] Dutch: VWO, highest level in the Netherlands
[4] Dutch: HAVO, second-highest level in the Netherlands
[5] bunq code guidelines require explicit if/else branches even when they are empty.
[6] TIN: Tax Identification Number
[7] Migration: Database alteration

Q *Can you elaborate a bit on what you found unclear about the database structure?*

S For example, we have the `User` table, in which each row is a `User`. But it can be any of the subclasses of `User`. So these are all different Models but they are stored in one main table. Which is being extracted at the application level again. And I found it difficult to see which Model was embedded in a different Model.

Q *So you mentioned that in your first day you would get an issue assigned, like the TIN issue, what was your approach to solve the issue? How did you find where to start?*

S In this case it was about a TIN having a valid or invalid format. This was already implemented but required changes. So I just started looking for where it happened. You end up with a library that does some regex[8] pattern checks pretty soon. Then you know that that is the place to make your changes. Then [a senior developer] comes in, saying "There should be a migration for this", making the entire assignment a huge thing in the end.

Q *I see. And when we look at a more generic level, when you get a random issue, what do you often do first to get started? What is your strategy?*

S I first try to determine what Models are touched by this issue. Then I try to think about the flow, for instance when the app was the perform this action, what would be a logical thing in their opinion? And I commonly check the code base for similar behaviour, 9 out of 10 times we already have something similar. Then you already get a pretty good impression.

Q *Okay clear. And does your current approach differ much from the initial approach?*

S I don't think so. I mostly know faster where I should look because I know the code base better by now.

Q *Okay, last question, what would have really helped you in terms of resources, something that would have saved you a lot of time?*

S Maybe a little more visual and clear representation of our Models and what domains we have.

B.2 Subject 2

Translated from Dutch.

Questioner (Q) *When did you start at bunq?*

Subject (S) Back in May 2017, something like 8-9 months ago.

Q *Did you have any prior working experience?*

[8] regular expression

S Yes, but that was mostly with iOS and Ruby, PHP was only with a webshop that had zero standards.

Q *But you did have some programming experience?*

S Yes, for sure. Also in PHP.

Q *What experience did you have with those languages?*

S Some app development and for PHP I worked with Laravel[9].

Q *What is your background regarding education?*

S HBO[10] Computer Science. For my graduation project I joined bunq. Before I also completed MBO[11] Media technology, which was also mostly web development.

Q *Then some questions that focus more on your first weeks/tasks. What did you find most difficult when you saw the backend and had to work with it?*

S In the beginning it's a lot of information to absorb. And, what I found difficult, during the reviews you would get a lot of comments about standards that you didn't know about. That's the biggest thing I think. All kinds of things that are explained when you hit them instead of upfront.

Q *Okay, and when you think back to your tasks back then, generic, what would take most time when you started with a task?*

S Well, for me, the only task was the graduation project, the app analytics systems. I just read a lot of code in the beginning. That is what I spent most time on.

Q *And were those random classes? Or specific classes?*

S Actually, random in the beginning, to get a feeling of the system. Like the views, models, etcetera. Later more specific like "How do I do *A* or *B*?".

Q *Clear. So you have mostly worked on your graduation project, but I can imagine that the project could be split in subtasks..-*

S -.. It was actually quite difficult to split it up. At least the backend part, that was just one big MR[12].. -

Q *-.. ah, in that case I'm curious about your approach, how did you get started?*

S To implement it you mean?

Q *Yes.*

S Just YOLO[13] actually. I opened an MR early and I sat down as well with [a senior

[9] PHP framework; https://laravel.com
[10] Dutch: for University of Applied Sciences
[11] Dutch: Middle-level applied education
[12] Merge Request; request to get changes added to the product
[13] "You Only Live Once"; just try and see what happens

developer] to discuss the structure. Then [the developer] told me to just create the Model and the View and then we squash everything necessary in between. So, that is what I did, I reviewed it with [the developer] and continued from there on. I think that works well, first create a design, the Model and View, and then build the remaining parts from there.

Q *So, is that the same as your "YOLO" approach or was it different?*

S Yeah, actually it started like that, but other than that I did not really have an approach.

Q *Clear. You started, as [the developer] told you, with the Model and the View and added the Controllers afterwards. Do you still do this the same way? Or did you find a better way?*

S When I started working on an entirely new feature I still do that, yes. First I set up the structure. We don't use UML, but our Model and View are basically UML, which can be generated to code.

Q *And when you would not work on a brand new feature, how would you approach it then?*

S When I need to change something in an existing feature then I just starting reading the code to find where I need to make my changes. Then I make the changes and throw it to [a senior developer] or I ask [that developer] what his approach would be.

Q *To determine where to make your changes, what do you do to find that position?*

S Can I explain that in a generic way?

Q *Yes, you may also provide an example if that is easier for you.*

S The Limits-feature is a nice example. I first went to [a senior developer] to ask how Card payments are processed and how they differ from SCT or bunq-to-bunq payments. So when he explained all that I knew which Workflow everything ends up in. That gave a better picture on how we do payments, I had not touched that part until then. Then I found the right place in the Workflow, made a plan for the change and then actually applied the change.

Q *I am still a little curious, you say you 'just made the change', but how did you know that that particular Workflow was the right one?*

S Well, I had to change the calculation of the limits and that Workflow contains the entire Limit calculation logic. Payments contain a lot of sub-workflows, I read them and then it became clear that this workflow was the right one.

Q *Clear! When you think back of the very first days here, what would have helped you a lot to understand everything faster/better?*

S More guidance from the mentors. [redacted]. But I am not sure whether that would really help, [redacted]. I think you better just get started, or that is how I work at least.

B.3 Subject 3

Translated from Dutch.

Questioner (Q) *When did you join bunq?*

Subject (S) January 22nd.

Q *You work part-time right?*

S Yes, two days a week, well 12-16 hours a week.

Q *Did you have any prior working experience?*

S Yes, I worked as a PHP developer for two years in Delft.

Q *Ah, nice, what kind of company was that?*

S Just web design, so a small web development company. Not that special. I just picked up upon the basics.

Q *What about your educational background?*

S I studied Computer Science in Delft, completed my bachelor, I chose the specialisation of Knowledge and Data Engineering in the second year. After that I started my masters in Artifical Intelligence at UvA[14].

Q *Okay, for you it will be easier as you are still in your first weeks; I will ask some questions about your first experiences. bunq has a big code base, what was most difficult for you when navigating the code base?*

S [redacted].
What I noticed there is that I'm really happy with the ⌘ + ⇧ + F[15] function in PhpStorm. You can make quite some progress with that. But what you see sometimes, because you want to keep the code very readable, is that functions get split in very small chunks that call each other. In the end you have 10 methods chained together. That makes it less convenient when you want to see where an error originates from. And sometimes you get another error, which you don't understand, but then it does not necessarily have to be your code. It can also be the environment then. Other than that I think it is quite easy to find where you need to be in this code base, mostly because of the consistent naming of functions.

Q *When you think back of your first task, generically speaking, what costs most time when you start with such a task?*

[14] UvA: University of Amsterdam
[15] Searches through the entire project

S In the beginning, and I think this holds for everyone, you don't know the complete code style yet. So you will need to master that. You need to know how they want you to write your code here, which is different from what I am used to. What takes most time currently is writing tests.

Q *Any clue why that takes so long?*

S Yes, because I have to work on entirely different things all the time. So then I need to find out again how, for instance, Authentication works. Or something else is slightly different, like making the balance of a User negative. Then I need to find out how to put that in a test. In the end you will get there, but you need to get used to it. You learn a lot from it though, and it goes faster by experience.

Q *In your first days, what was your approach when you started working on an issue?*

S Well, for the first feature I had to work on sending a notification to Users with a negative balance after *x* days. The problem was that the balance was not shown in the notification, so a User would not know how much to pay. What I did there; I took the existing test, I changed it completely until I could test the negative balance and notification time treshold. How I typically write PHP is first writing the unit test and then adding `print_r` statements to show whether the expected result shows up. That what works best for me, there are people that don't like it at all, but I like it. I keep adding small chunks until I have the output I want. What you also see is that people write their tests and add some `assertEquals`, but then you only see that your test fails.

Q *Did your approach change meanwhile? Do you do things differently now?*

S Not really, I've always worked like this. After a while you get a bit of feeling for it, you know *where* to print and basically it is the same as what the debugger does. Only I forgot to remove breakpoints for the debugger because I loose track of them. For some reason I do not have that when I use print statements.

Q *When you think back of your first days, is there something of which you say "If I had that,.. that would have helped me a lot", then what would it be?*

S Documentation is always useful. Here it is more like, this is the issue, do your thing. You will learn more that way. Which is true. So documentation would be nice but otherwise I can not think of something.

B.4 Subject 4

Questioner (Q) *First, I would like to know when you joined.*

Subject (S) 4th of December.

Q *Okay, so that would be two-and-a-half months now?*

S *Yeah.*

Q *Did you have any prior experience?*

S No, well, I had some freelance jobs that I worked on together with friends. I had some internships and the freelance jobs but nothing full-time.

Q *Okay, and what work did you do there?*

S Mostly front end development for the last year of my studies. Also worked a while with React Native for a tool where I developed the frontend and a friend the backend. That were, like, simple SaaS[16] platforms.

Q *Okay, and what is your educational background?*

S I double majored in Chemical Engineering and Computer Engineering. I wanted to do Computer Engineering actually, but the school I went in.., I did not get the scholarschip for Computer Engineering. But I did get it for Chemical Engineering. Then I entered from there, that was a fun story. I spend five years because of doing both of them, but yeah. So no masters or anything.

Q *Nice! Some more questions about the code now, you've worked a bit in the backend right?*

S Yes.

Q *Did you have any difficulties in getting around for your first task? Did you hit any bumps on the road?*

S Yes, the first two weeks were the most confusing times of my live. Because I did not know what I was doing. I did not know what I was reading, what I was looking at. [redacted]

Q *And can you elaborate a bit on what made it hard to understand? What was the difficulty in that?*

S Might be because of my mentor as well because it is very much his style to let me figure things out by myself. But yeah, a little more documentation could have helped maybe. [redacted] I am talking a little bit abstract, but I think you get what I mean?

[16] Software as a Service

Q *I think I understand, let me try to rephrase it; basically your mentor had some difficulties looking from your perspective. He could not really replace the mindset to explain it properly at "your level" so to say. Is that right?*

S Yes, exactly. Although, it is not just the mentor, like the code base and projects are also a lot and complicated.

Q *Can you maybe explain what makes them complicated for you?*

S What makes them complicated; there has been years of work in there. And we are expected to understand what is going on everywhere in it in just a few weeks of time.

Q *If you think back on your tasks in general, what takes most time when you start working on an issue?*

S When I start or do you mean in the whole process?

Q *In the whole process. What takes most time?*

S The iteration of reviews. So yeah, when I think I am done, someone else has to review it and there are things to fix. Then they review it again until it gets merged, there is a meaningless loss of time in between.

Q *Where do you think this comes from? The reason behind it taking so long for it to get approved basically?*

S It is also a part about me too, because I am doing too many iterations of reviews. Like I should be able to get it done in a few. I am trying to reduce that. But I guess it is because we are very few people and everyone is trying to get lots of shit done. So they just can not find the time. If there were more people to review stuff that could already help.

Q *Do you have any specific plan on how to reduce the number of iterations?*

S Read my code more carefully.

Q *Let's go to the next question; When you think back of your first days and you had to implement something, what would be your approach to attack something? Where would you start looking etc.? Your strategy basically.*

S In my first weeks I was getting very much confused in the code base and I was just jumping from function to function to figure out what is going on. And it was too complicated for me in the beginning and it would have been much better if [redacted].

Q *Okay, cool. And how do you currently approach issues when you get an issue assigned?*

S I ask people.

Q *You ask people, okay, that is a clear difference. Another question; thinking back on your first day what would really have helped you back then to understand everything faster? What did you miss?*

S Well the first week was all setting things up, you mean when starting with the projects right?

Q *Yeah well, not necessarily the setup of your development environment but when you start working on a task, what could have helped getting it done faster or help understand the problem faster?*

S [redacted].

Q *In what sense should they be better?*

S [redacted].

Q *What kind of documentation would you have liked to have then?*

S Like the backend structure for example, you know?

Q *You mean the high-level overview of the components, you did not have that for the frontend?*

S No.

B.5 Subject 5

Questioner (Q) *When did you join bunq?*

S I joined bunq almost four months ago.

Q *Cool, did you have any prior experience with development?*

S Yes, a little bit. My previous job I did like fifty percent coding and fifty percent research. It was the place where I did my internship and I just stayed there and I was researching some algorithms and coding a bit. So, not a lot of coding experience. I also did two internships, but one of them was the research one and the other one was A-B testing with JavaScript, and also research.

Q *Was the programming all in JavaScript or also other languages?*

S The first internship was with JavaScript and then second one and the work I had was a bit of PHP, JavaScript as well and Python.

Q *And your background? Like what did you study?*

S So I studied Maths and IT, so that is like 50 percent Maths and 50 percent IT. And before that I actually did Art High School. So something completely unrelated. Without IT background I basically started at the university without knowing much at all. In High School the only thing I did was HTML and the standard stuff.

Q *Also scripting?*

S Yes, but like basic stuff, simple websites.

Q *And then during university you learned the real stuff?*

S Yeah.

Q *So let's go back to your first weeks/days. You did do some stuff in the backend right?*

S Yes, but not in the first weeks.

Q *No, okay, but try to think back like, the frontend is also pretty big as far as I know, so try to think back of that period. What were difficulties you had when navigating around in the project? What bumps did you hit to understand things?*

S In the project or in general?

Q *In the project, with the code.*

S On specific projects or..?

Q *More generic, like if you have an example that is fine, are there any specific things that were a hurdle for you to overcome to understand?*

S Let me start very generic. So, in the beginning it felt a bit overwhelming. There was a lot to read, but not a lot of time to read.

Q *Do you mean reading code or documentation?*

S Documentation on the wiki and so on. I got pushed into making issues and so on, without understanding much. [redacted]. So that was a bit weird for me in the beginning. The difficulty I think with the issues, they know how to build it up, so they do not put too difficult things in the beginning. At least, that is my experience. But then it was a bit unclear to me how the communication works, what were the

ownerships for the issues and so on. So coding wise, well with backend we have all of these Daemons that you can run and so on, and that was also unclear for me. I think we also want to ask for help, but we do not want to bother people. And that is a big thing, you are stuck and you are stuck and you just do not want to bother anyone.

Q *Is there similarity in the things you get stuck at?*

S Quite some backend things.

Q *Do you maybe have a few examples?*

S I know that sometimes setting things up for me takes a while. Because some of the explanations just in a way that I do not get it. It requires a bit more base level of understanding and that is what I do not have for example. Once I have it set up I can work with it, but some of the things I just wouldn't know how it's done.

Q *Okay. So this is somewhat related, when you think back of your first tasks, what did you spend most time on?*

S Trying to understand the whole project for sure. So, trying to understand the bigger picture of the issue. So that was the biggest thing for me I believe for the frontend. Oh, and also debugging PHP. In JavaScript you can just print something and you see the errors everywhere. That was a thing I was stuck at for a long time now that I think of it. I would get errors and I wouldn't not know where they were coming from. It would be super vague and it turned out that it is something completely different than it was indicating.

Q *Going forward on what you just said, the 'getting the bigger picture', how would you approach that? Like how would you try to find the context around it?*

S First I would check the wiki, because sometimes there are explanations on the flow. Also I can use the app, so that is one way to just try it and see what happens. I also looked in the triage app and there are also test-runs[17] so you can see the sample workflows.

Q *Then you would look at workflows in the app, not necessarily the code?*

S Yes, I think it's more of the frontend approach.

Q *When you need to do something in the backend, how would you approach that then?*

S First I would look at the description of the comments. Because sometimes the comments differ from the description. Then try to understand it from the frontend way. So for example, when it is an admin thing I try to search for a user and see if it's there. I look for the requests being send and so on. If it's like app, well,.. the same. But if it is really like a backend thing then I just go really simple. I look for keywords for example. Like 'debt collection', looking for a folder like that and just looking for

[17] Pre-defined test scenarios used in manual testing

the controllers and view, model, etc. Looking for that and then broader. So, a lot of searching in the project I would say. But that is when you do not know what you are doing, I would say. When you know what you are doing you know to go to that view and so on.

Q *That's actually also my next question, what you differently now compared to in the beginning.*

S Yeah, now I know what to look for. I do not necessarily look for keywords but I look for folders.

Q *And how do you know what folder you need to use?*

S Well, many things come from the models like `User` or `MonetaryAccount` so you get a starting point and you just go on from that. Like the frontend repository is way smaller, so it is way easier. You need to deal with iDeal? Well, iDeal folder. bunqme? bunqme folder. So for frontend it is easy to know, but for backend, I am still not for a very long time here, but I can somewhat find my way already.

Q *So you would go with the foldernames and then work your way further to the controllers, ..-*

S -.. also folder names but also file names. So look on the files.

Q *So if I would say 'you need to do something in the user approval workflows', how would you approach it then?*

S I could type either 'user approval' or literally search for keywords or double-tab Shift and look for the folder or the names. I use a mixture of both I do not necessarily pick for one.

Q *I see. And how do you decide, when there are multiple workflows, which is the one you need?*

S I would look for the keyword and see where the approval appears.

Q *I have one last question, when you are back at day one, with your current knowledge, what would have really helped you back then? Something of which you say 'If I had that, I would be so much better in understanding or faster..'.*

S For a backend developer, there are way more things, that I did not necessarily read all. But I think just having the feeling that you can just sit and take a look around instead of being rushed into something. Just so that you get somewhat a journey for it, at least one day or so at work. And asking. Maybe not just one day, but the first week a bit easier.

Q *And how would it be easier? You mention more time basically and asking questions?*

S Yeah, just being able to read in piece. My first day was kinda rushed. It was not a nice experience.

Q *Are there other ways to make it easier you think?*

S More documentation maybe.

Q *What kind of documentation would have helped you?*

S More basic explanation, but they are changing a lot, so I'm not sure how useful it is. And the structure. Mostly the first week would help. People learn in different ways. Some people just like digging into code, some people like to read up. Maybe even really understanding the app, the admin, a proper journey for that.

Q *You mean like using it first?*

S Yeah, like 'this is what we are doing'. They say you need to know it beforehand.

Q *You mean before you start the job?*

S Yes. I think it is also nice to maybe have an opinion from them. How they use it, how they see it from their side as well.

Appendix C

Experiment Interviews

This chapter contains the transcripts of the interviews conducted as part of this research. Some transcripts have additional annotations, to clarify what a subject is referring to or to indicate that something could not be heard properly on the recordings for instance. Those are put between squared brackets [like so].

C.1 Subject 1

Questioner (Q) *Cool, what have you done and what do you think still needs to be done?*

Subject (S) So, I've been looking at the view. And then I've seen the counterparty name, which in the beginning I thought was the actual thing, because I was also looking in the model. But then I realised it's actually the merchant so I was like, nope. And then I was like 'Okay, can I add the user label' like in the rest of the places, this is how we do it. So, okay I started looking in the endpoint and I just didn't see any place in the endpoint where it touches the bunq user. It was basically just creating the Model. And so on, just updating it and so on. Then I figured I must look outside the endpoint and just check to the point where it does something with the bunq user and grab the name and create a label from it. Or just grab the label and pass it along. And then update the model like it was stored in the database and then pass it in the view.

Q *Yeah cool. That's basically the idea yeah. At some point you create the Request Inquiry basically, and that you scan and then the user is attached to it. So it's somewhat deeper, but that was also to try to see how you can dive into it basically.*

S The thing is I was confused because I really thought it has to be within this endpoint that I'm only supposed to look so I just really got stuck in it.

Q *Yeah, I see. I think it would be useful to look at the following assignment, to not waste your entire afternoon. I have paper here if you need it, if you prefer to type that's fine with me as well. It's just basically implementing an endpoint, or the steps.*

S Let me read it then.

-

Q *So, what have you created?*

S Okay, so we would have like a AliPay model and this would hold the main details, like for example the amount, the currency, the counterparty details, status, return url. I saw also language was being used quite a lot. That would be the model more or less. Then we have the controller. It starts with making a POST call and that would be using the Create class. There we would create the model with required fields. So, the fields would be taken also from the view. And the view would be the amount and so on and details to the model. Then we would create the POST. Then, I would create it with status pending. Then it would return the UUID of it, with the UUID I think it would make another GET for the QR. And possibly more details of the transaction. And then whenever the users scans, I know there is mobile part involved and I do not know exactly how that -

Q *Yeah you can ignore that part of the app.*

S But then, yeah, depending on the action what the user does the transaction can be either accepted or denied. So at that point it gets updated. The update would also go through the Process class. If needed, like if it's more complicated it needs to do more things than just update a few fields. Well, we can also do Listing and so on but in the view we have also the pre and post processors. For the pre processor all the things like the user sends, fields for the create, and then for the post processor all the fields that we want to display for the user and place it there. I'm not sure how technical I'm supposed to go in this assignment.

Q *This is fine! So, were you able to use the tool during the assignments?*

S Yeah, like I think more in the beginning, I was a bit confused to be honest. Like what I expected to see when I searched for a Feature is maybe somewhere highlighted the actual endpoint in here. So I see all the places it is being used from what I understood. I'd also think want to see like the ones that are relevant like the most relevant. Which have [redacted] in the name or something. Because then I know I can just go here, cause now I have to like really scroll and look I think. Or it would do this [computer typing.. (⌘ + F)] and now I still need to see for a name.

Q *Yes I see, and had you seen the filters on top?*

S Those are very nice for example to get to know the workflow.

Q *The graphs you mean?*

S That's something I was missing, when you look at the JSON it looks like [ugh] and I think it's also interesting if it goes like this at some point if it's like true or false.

Q *That's actually in there. This workflow doesn't have it, but had you seen that you can filter here for the, you can click that and well this is an endpoint that doesn't match the regular expression but. -*

S Ah yes, this I also checked.

Q *And does that help already?*

S Yeah, a little bit but I think it'd still be nicer to narrow it down. Like, to me it so much things at once it just gets kinda confusing. Maybe even, imagine this was toggled/hidden so you would only see this and then open.

Q *So basically it would be collapsed.*

S I think that would be nice.

Q *Ah fair point.*

S Then you see just the names and you're like "okay, I wanna go to this one" and then [bam] click. Because now I just feel overwhelmed. But this, this is awesome.

Q *So I see you have used the Find By Label feature for [redacted], could you think of any other use case for this feature? To search by the name of a feature basically.*

S What do you mean?

Q *Well now you used it to -*

S Like label?

Q *Yeah, but what was the goal of your search? When you searched here for [redacted].*

S I was looking for [redacted], first of all, but -

Q *Eh sorry, yeah.*

S Just to get an overview of the endpoint and how it's connected to other endpoints and so on.

Q *And can you think of like a different goal for such a search when you search on a Label?*

S Kinda, but - this is supposed to help developers right?

Q *Yeah*

S I think if I'd like to search I would want the path to be displayed. So, where is the endpoint located.

Q *And it's not clear currently from this graph then you mean? Because that is the path somewhat, but it's split in a tree view here.*

S Path is [computer things], this is kinda the path.

Q *Yeah, exactly.*

S I think there would be like a main one on the top or something. Like, this was a thing I thought that shows me the path as the first thing.

Q *So, in case of a feature that has multiple endpoints then -*

S Show the model, or I don't know. Some of them don't have -

Q *But the main parts of the feature should be shown more explicit on top?*

S Yeah or even just the names of the controller/model/view.

Q *Yeah, I see. Okay clear. Have you used the find by developer feature?*

S Only in the first one, the first exercise.

Q *And what do you think would be a use case of that feature, what could be a use-case?*

S For example, one person was working on iDeal and you kinda expect he worked with things closely related to it maybe. Which won't necessarily have the name iDeal in it. Maybe something like this. Or if you just want to see how much someone worked, how much he did or how much he developed I think it could be interesting thing to also give new tasks to a person. Like, you see what the person worked on so you can make a better match on what he can work in the future. I think it is also interesting for Product Planning.

Q *Hmm, yeah, fair point.*

S It kinda sucks like this can monitor the users but I think it would be perfect for them. The progress.

Q *Good point actually, I had not thought about that yet. Okay, cool. The Find By CIT, have you used that one?*

S Only in the first. For the bugs I think, can you also write like - this is just a number of the bug right?

Q *Where?*

S Here. Bugs and then just the issue number.

Q *Yeah the number so that it is the complete reference as you write in your commit messages.*

S It would be cool if there was also a field for the name of the bug. All bugs related to [some featurename].

Q *Ah, like that.*

S Then you can see what feature is buggy what needs to maybe possibly rewritten or something.

Q *Ah, that's actually a nice one, because I have all the bug issues that somehow relate to a feature in the end or should relate to a feature, then you can see how many bugs belong to certain feature.*

S Also I think it would be interesting also for product.

Q *For anyone basically then.*

S As a developer do you care how many bugs there are for something, you just care that this is your bug. This is your bug and if there is a lot of bugs then it really sucks I think. Okay, I'm curious how many bugs there are in this but I believe I care less than Product.

Q *Yeah, that I believe as well. Because they want to deliver the perfect Product. You want to get shit done.*

S What else for this?

Q *Had you also seen that you can search from these buttons?*

S No, but it makes sense I was just.

Q *You have to click the explicit icons on top, something everyone click on that thing, should have thought of that.*

S I did not realise it's clickable. It's clear now.

Q *Had you read the sidebars?*

S This? No.

Q *I already expected that it wouldn't be read.*

S It's so much, I just like [sigh]

Q *I know, have the same problem myself but I wanted to have at least some minimal documentation there.*

S Maybe make it collapsible or so.

Q *Yeah, I already tried to make it like with small headers so that if you're like what's this that you could see immediately on the left where you would find the explanation for it.*

S Yeah but I really think like if you put a lot of text at first it's,.. annoying. Like, it's just makes you not want read it. I think when if is collapsed and you see there's more and you really want to you will click but..

Q *So that you would only see like the headers of the sections and then open them.*

S Yeah yeah, simple like that that.

Q *Okay, fair. Okay, cool, are there any benefits to using the tool and if so which ones? Or why?*

S You can really see the overview better than the other way. Like, you can see all the places it was used. I think it's really great, like when you look in PhpStorm it's a pain in the ass. Like you really have to scroll down, okay you can also see but you really have to scroll down the workflows just to see a JSON. Here you get it visualised and so on, and the authors I think it's also really cool and you can click on them. I think it goes to GitLab right?

Q *No this will search for the Find By Developer.*

S Even better in that case actually.

Q *Then you can see what that developer else has developed.*

S [Loads overview of a senior developer] PARPs? What is PARPs?

Q *The PARPs are the product propositions, so for instance there was one PARP for Apple Pay and then all the backend, frontend, whatever issues point to the PARP. There is one central issue with the documentation. I actually took out the part of the PARP explanation.. - oh no it's in here, with one small line.*

S Ah yeah, so if you see the author you know who you can ask for help.

Q *Nice, that was actually the intention of the feature.*

S Oh, I forgot, that was also missing. You have to look for the, like, you right-click and look at the annotations and know who to poke if you need. And you see multiple authors

Q *Yeah, that was kinda the intention of that thing. Okay, what should have been different to the tool and why? What did you miss?*

S Clarity I think. I was feeling overwhelmed by the amount of information. What I was missing was just yeah clarity. To see what I should be looking at. So I get everything, but I don't know what to focus on. And I'm like,.. kinda frustrated and confused.

Q *And that's mainly the tree-view or also other components?*

S What do you mean? Ah, tree view, —

Q *Ah yeah, tree view, with all the files.*

S Nah, I just didn't understand the word. [Reloading interface meanwhile] Yeah I think that's for the tree view and the thing on the left.

Q *Ah, the text as well.*

S Yeah, too much information. Maybe this should not even show up until you click it [sidebar with explanation] or something. Maybe something like this.

Q *Ah, makes sense. Or not have the all categories so that you can only see subsets, that's already smaller.*

S That would also work, but just leave it empty until actively clicking maybe I think. I don't know, I'm really fan of "the less you show the better" kind of approach.

Q *Yeah. But you also don't want to show like too few information. You need to find a balance, but I agree that this list is really long.*

S This is nice also, the labels, related labels. [mumbling: cards].

Q *[laugh] There's a lot of authors. And a lot of files. And in terms of insights, did you miss anything there or.. say it would be really cool if this was added as well or..?*

S Like, what could be additional?

Q *For instance.*

S Not sure. Search by label, developer, CIT. Search by endpoint maybe. I think that would be great. That's what I also kinda tried to do within the exercise 2 I saw, [was it here,.. the endpoint] I really missed some of it. I typed and then it shows me everything as well. I would have loved that.

Q *Because then you can see exactly for an endpoint,.. — that might actually be really nice for the app guys when we implement something that they can generate their own documentation basically.*

S That would be another nice thing then. Like, show the POST / GET, like that. Really visual. I know the example for the POST. Definitely make an example, for the test you have the v1/user/blabla. [A developer] made something for the documentation.

Q *Yes, that was really nice! You can now add like the create example —*

S Yeah it seems nice.

Q *Yeah, I have used it already once, it works well, only that it prints the documentation twice. But yeah, I delete one and —*

S Worth it.

Q *Or the app boys delete one. Whatever. Let's see. Do you have questions?*

S Maybe actually also like take the description from the JSON.

Q *Ah yeah right. Right now, I don't take anything from the JSONs, only when I visualise the workflows then I parse the JSON of those workflows. But other than that I don't do anything with the filecontents. So this is all based on the git history.*

S Ah, like that.

Q *Yeah, so I index every commit, and for every commit I check who authored the commit and what files were touched. So, and we have the reference to CIT in the commit message always, or we should have, so that way I know these files below to this issue and then I have a different script that goes for every issue in my database and it will fetch the labels on GitLab and the related issues, stuff like that. So, but it starts with the commits rather than the files.*

S Okay that changes a lot then. Cause that would also would have been great if it gets the point like model and you see the model in the endpoint here. Or you see the view, the fields you know.

Q *More like the concepts rather than the files or the authors.*

S Also. Both. This is great. We see how far this issue goes, not the endpoint.

Q *Do you have any general feedback apart from what you have already mentioned.*

S Yeah, I like it. Really great initiative.

Q *Yeah this question does not make much sense for you I guess, but what's your impression of the bunq backend?*

S Very organised, comparing to the frontend. It's always like, the first time I looked at it I was kinda like really impressed. Like, how well you can manage a backend. So, yeah, positive.

Q *Okay cool. Do you think it's feasible to implement or improve features using this tool, without the help of a senior developer?*

S Like, you're asking if this kinda replaces a senior developer?

Q *If it could.*

S No. I don't think so.

Q *Why?*

S Because sometimes you really don't understand. How something is processed or something. And this tool can give you an overview. But other files in related to the authors and — okay, you know who to ask with this tool, you can see the senior but I don't think so. Like, you really need to sometimes to go to someone to talk face-to-face and they have to draw something on the board. You don't — like this will definitely help but I don't think it can replace him.

Q *And that's then basically because of the understanding when you want to explain something.*

S Yeah, I don't know. Maybe when a coder would be much more experienced then I would maybe be able to get more out of it. But for now no I don't think so. I just need the [senior developer] sometimes. With the blackboard you know.

Q *And a more general question, what do you think are the most important sources of information when you want to learn how to work with an existing codebase, like what's most important to have to learn something like that?*

S To learn programming in a place or what?

Q *No, say you join bunq as a new developer and you have to learn this entire framework, there's barely documentation but what sources of information are valuable to you, can be any source of information.*

S Wiki. A wikipage. That's pretty useful. If something is documented.

Q *So documentation is general would be the source then.*

S Yes, but not fully. Just trying out stuff, trying chunks of code and not the same just random chunks. But like if it's something you are working on you can run the test or you can just look for the workflow here it is also visual so you can. Well asking people, if in doubt. Junior group -yeay-. So wiki, code itself, groups, yeah I think that's it.

Q *Okay, cool. Those were my questions.*

C.2 Subject 2

Questioner (Q) *Did you like the assignment a bit?*

Subject (S) Yeah, it was nice.

Q *Yeah? Okay, cool. So, I'm going to ask you about the tool specifically about the different features in the tool basically. Because I'd like to have your feedback on that. The first feature, the Find By Label, the top one, have you used it?*

S Yes.

Q *Can you explain me for what you have used it and why?*

S I've used it for, here obviously [exercise 1], for the first task and then here I didn't need to [exercise 2] and for the third one I was looking for a similar thing I did before and I tried to use a Label for it but it didn't take me to the right place and then I went to the code and I found my commits and issue number and looked by this and I found it.

Q *Ah, so there was a different label on the issue or something?*

S Yeah, it was - I was looking for something more specific and it was more generalised.

Q *Ah I see. Yeah the labels are taken from GitLab so if it doesn't exactly match then - yeah it won't retrieve your commits obviously. Okay, the Find By Developer, have you used that one. Apart from assignment one then.*

S Same thing actually. I searched for my commits in there

Q *But you were not able to find it?*

S I was, but I still wasn't sure if it was the same Label. Because, what I was looking for was the Tinker thingy and I was supposed to look by Public API apparently. That's the label it has. And everything is in that, a lot is in that at least. And this is the CIT I later came into which helped me find the thing.

Q *Okay, cool. So yeah the Find By CIT you have also used. Can you think of another scenario where the Find By CIT would be useful for you?*

S It's actually nice to see all the files related to solely one problem. Like, in GitLab you don't see this because - okay, in the MRs you see the changes and stuff but neither in PhpStorm nor in GitLab you can find all the related files about one issue / about one feature. And that's really nice. Because for this - very good example I think - one thing to note here is - I immediately skipped the recommendation here -

Q *The hint? yeah.*

S Because I thought of something similar I did, and that felt more familiar for me, and I wanted to search for that as an example.

Q *Yeah that's fine.*

S I had no idea what files I had touched for that and seeing all of this together immediately helps me visualise how the whole thing works.

Q *Oh that's really cool, yeah. And did you also use the visualisations of the workflow?*

S Yes, I did, I really liked that!

Q *And what information did you use from it?*

S Mainly played around with it. Nothing - didn't find any specific information that I used. Oh no, that's not true. For the UserDeny thing - was it a thing here? Ah, Account closure - that was very specific, that was very good as a workflow. I knew how [another developer] build it and I had read the workflow before but still, when you read it here it's like very clear what we do to deny the user. I actually read this one, it was really nice.

Q *Okay, cool -*

S So, if the function names are really clear it can be enough to understand what a workflow does.

Q *Yeah. That's - the system itself is really nice but the JSONs don't read that nicely I think.*

S I think, by the way, for one of these tasks I was reading a workflow definition and I found it boring and I went back to this [the tool].

Q *Ah that's really good.*

S Yeah. And then I went back [code] for the details and the arguments and so on.

Q *How did you find it here then? That specific workflow? Did you search for it?*

S I was already on the same thing.

Q *Ah okay, so you did* ⌘ + F *or —*

S I actually found the workflow name here first I think. And then I went to the code.

Q *Ah I see, nice nice. Okay, that's good. Are there any benefits that we haven't discussed yet to the tool?*

S I made a note when I was playing with this in the first part. Maybe this is too much from the frontend perspective, but it could be interesting to see not the lifetime of the workflow but the endpoint. So, when you make a call it goes to pre-processor and then it goes to some workflow and then maybe some more workflows/libs and some post-processor. We know this connection because we know how our codebase works, but could be interesting to see which files are touched when you make a call until you get the response.

Q *So basically you'd get like HTTP method including endpoint name and then see everything that's being touched?*

S Yeah!

Q *That would be cool indeed.*

S For this here for example for the second part, how I found related files was also just searching by the endpoint name and going through the files through PhpStorm.

Q *Yeah, that's also what I typically do nowadays, but you need to know how to search and then the links between the Workflows and the endpoints which is quite difficult to visualise it with the current setup I think, because I base the index on the Git log so I don't look at the contents of the files basically. And for that chain, if a workflow calls a workflow, I won't see it basically.*

S In this context actually, even though you don't see the tree, when you look at all the files in that specific issue it's clear what goes where. Because of the naming also.

Q *Yeah, but that's also because you have the inside knowledge, for a fresh developer that would be more difficult to see. Okay, cool. What should have been different about the tool according to you?*

S Can I comment on the UI?

Q *You can say anything.*

S This should definitely be wider, this is the main thing that you want to show and search and whatever else is not really needed. I want to see this [main part].

Q *Okay, cool.*

S And what I was looking at one point was a combination of the - I'm asking more and more features, but you know that's what people do —

Q *Haha, no no, it's fine, that's actually what I need.*

S At one point I was looking at - I was trying to filter a feature by the developer. I went to the feature and I clicked the developer and I kinda expected it to be combined here.

Q *Ah I see yeah. I wanted to do that at first but that would take me so much more time so.*

S Yeah I can image.

Q *But it would be a nice addition.*

S Not really sure if it would be worth it but that was the natural feeling when I clicked it.

Q *And what was your intention of filtering it down to first feature and then a developer? What did you..?*

S I think for the account closure thingy there were two different issues or features there. And I was trying to separate them. One is the denial from admin and the other is the block flows.

Q *Aaah, and they're like somewhat intertwined, so yeah.*

S Yeah, they both have the same label but admin user deny thingy is done by [another developer] and the other [a senior developer] I think. Or maybe just different issues I'm not sure.

Q *Oh yeah, I had not considered that. Yeah, fair. Okay, cool, useful. Do you have any other general feedback about the tool?*

S I haven't really read this, I don't know if I should have [info-boxes].

Q *Well, it's a help-box right? "Have you read the manual"?*

S Nobody really reads this stuff I think. Not when I write them anyway.

Q *So, in terms of functionalities, are there things that should have been in here that would have made it more useful or things that could have been left out because it doesn't help you or..?*

S I was actually quite impressed. I wasn't expecting it to be so progressed since I last saw it and I just.. - I still don't know how it works in the background, like how you connect it with GitLab and find everything but it's pretty cool.

Q *Basically what I do is that I go over the Git log and I take every commit one by one, and every commit has like the CIT link, or it should have and there are a lot that haven't but okay, most of them have it. So then I have a commit with the changed files and the author and the commit tag. So that I put in a graph database and then I have a script running after that after that indexing process for each issue in the database goes to GitLab, retrieves the title, the related issues, —*

S Labels —

Q *and then also ties that into the database. So it's two stages basically, first the Git log annotation and then retrieving additional information from GitLab and the graphs is basically — I have the full file path and I go to GitLab and I just extract the file. So if you rename a file later on, or you add a different file that now how has the contents, then it might be that it just shows an error like 'File doesn't exist'. But that happened only one or two times, so yeah.*

S That's cool.

Q *But that's also why the file contents are not considered now, that's just, — Like I could do it when you click this rendering thing but then it would take a long time to render because it would need to fetch the first file and then second file and if you have like a long chain then you get a lot of requests to build up the entire graph. So, but it's something that I tried to do. But I didn't put it in the thesis. I first had a menu on top here as well so you had like visualiser, file search, and whatever and I had like great ideas, but no time. That was a pity.*

S I think once you're familiarised with the actual code base you don't really need this. Because you know how to search things. It's just nice to see the workflow graph but other than that I could have done everything in the code as well. Except for searching stuff by commits and like issue tags. That's really cool. For that purpose I go to GitLab, open the issue, open all the related MRs and check out all the files on my own.

Q *Yeah, way more work then. Yeah well, that was kinda one of my questions as well. Because I wanted to ask whether it would be feasible to implement or improve features using this tool, without the help of a senior. But yeah, you kinda gave an answer already but maybe you can answer explicitly as well?*

S Can you repeat the question?

Q *Whether you think this tool could replace the senior basically, as like help guide or . . . .—*

S That's tough. It could take some work of the senior. But initially, when I'm given a task and I need to build something I still need to know what other similar stuff exists. So I would ask in #juniors[1] or maybe to a senior developer "can you give me an example to look at for this feature?" And once he gives me the name this would then be useful, because I can search and see the related files and go read them.

Q *Yeah exactly and then you know like what you should approximately build.*

S Yes, but I do still need that name to begin with.

Q *Yeah exactly, so then the senior would need to give an entry point and then you could continue from the tool. Okay, cool. A bit general question; what's your impression of the bunq backend. What do you think about it? The backend in general?*

S This is the first time I write PHP, this is the first time I work full-time with big team. I have only done backend projects with my close friend with random naming and like random commits and not structured at all. Well not unstructured but we knew about it internally, but it wasn't really a structure. Here I really like that there is this naming and you can do this there thing going on because it helps me structure my thoughts in how to create new big projects. Like I'm doing now with admin and it's like, it could have been a whole naming mess if it wasn't for my first six months.

Q *Yes exactly. But now you have this experience.*

S Kind off.

Q *Yes I see, okay makes sense.*

S It taught me a lot but I am not experienced enough to comment on it further.

Q *Well no, it was more meant like 'What do you think when you see the backend?' and so, that's then the structure apparently.*

S I don't hate it.

Q *I don't hate it either, that makes two of us. I think most people won't hate it as long as they know how to work with it.*

S Yeah. Makes sense. I could have hated it in my first months.

Q *Yes, exactly. Last question; what are according to you the most important sources of information when you start learning such a framework?*

S Example code.

Q *Example code. Like other features?*

[1] Chat group containing all junior developers at bunq

S I usually just — if it's a frontend thing I play around with it, change some things in the code and see what it does. If it's a backend thing I just jump around in the methods and files and read it and try to see what it does. After a while I come up with questions and then ask some senior or something. And then my questions make a little more sense. And yeah, then people are also very important in this sense. But, I think I prefer learning from the existing code.

Q *Learning by example. Okay, very cool. That's was my part. Do you have any other questions for me?*

S No, I'm good.

Q *Then I'm going to wrap this up and I won't bother you anymore about the thesis for now.*

S It was really nice.

C.3 Subject 3

Questioner (Q) *What have you tried to search, what obstacles did you encounter..?*

Subject (S) I first tried to find the feature in the tool. Then I looked at the Models, Views and Controllers and then I started searching myself in PhpStorm. There I first checked the controller, all the transaction create and process things and read / update. I tried to find where it retrieves all the data, for what exists already at least. I thought, maybe I can then find somewhere where I can also add the name. But I don't really know where it retrieves the user so I wasn't really able to find that.

Q *And say you had found the User, do you have an idea how to pass it in the response in the end?*

S The first thing I tried is adding it here —

Q *Let's see, where is this, ah in the view, yes. —*

S Okay and then I added it to the response, [searches in IDE; "I thought I had to add it in Create, or was it…"]

Q *This is the workflow* [redacted]*TransactionCreate, yeah.*

S Oh, I wanted to add the user here then. But I don't really know — well, anyway, I was searching where — I should look at the views as well. But I didn't do that much.

Q *No, okay, but you already found quite a lot so that's good. How did you find these views?*

S Well initially I checked here [tool] and I searched immediately for [..mumbling..]

Q *So you have searched for label* [redacted]*, then clicked views —*

S Yes and there [searches initial file]. — Ah, by the way, sometimes when I click here [points at search field suggestions] I think it will search immediately. — But they're al below each other so

Q *Yeah, via the controllers you can also go to the views then. Good, but you were able to find that you had to change the view. Adding an additional field, yeah then it's a matter —*

S Yeah but it is still empty, I didn't know where to find the info.

Q *No okay but you were able to find how to indicate that there will be an additional field. So that's good already. Did you encounter any other obstacles?*

S Not really, just that it is a lot. Really a lot. A lot of code, I had not really expected that. I knew upfront but, now I scroll through it myself I feel kinda lost sometimes.

Q *Yes, I can imagine. I would propose that you continue with the third part.*

-

Q *Okay, tell me, what have you found?*

S So we need at least the model view and controller. I had some difficulties with the models, in this case I didn't really know how to approach it. Anyway, normally it being used for database interaction. That's how we [team] do it at least. What we need then is; we need to request the information, when someone initiates such a transaction. So that would include the name and the amount when I'm right. And then a transaction should be created I think in a controller. So I need a class that creates it. But that class then sends it back to, or well, here it says this [points at exercise 3 steps, UUID] —

Q *Yes, it returns an ID, or how do you say, when we literally return the ID as in the database someone could easily see how many users we have for instance. So we generate a unique identifier with all letters and digits mixed up, that's a UUID.*

S Okay. I also think we need a class to update all these transactions — well, I'm not entirely sure whether you need a separate one for status because in my head you can also update the status in the update [workflow] but I'm not sure whether it should be separate.

Q *You can, it a design decision.*

S And also we need the ability to read a transaction. For when you get it back [the UUID]. The user should have a screen like you showed me. And thus that (s)he can scan the QR code but that (s)he can also see the status of the transaction and that (s)he can choose the deny or approve. I feel like everything could fit in one view, but that's not how it works I think.

Q *Yeah, our views are slightly different — I think you're thinking of Laravel views — our backend, we have views, but our views are basically only JSON because —*

S Ah yeah I saw that indeed, for me in my head it's really a webpage, but —

Q *In Laravel terms, instead of doing the return view and then the name of a view, you can immediately return an array and then Laravel will transform it nicely into JSON. So you don't wrap it in HTML, it's a separate page that makes a call to an API endpoint and it extracts the info we want. While in your system [translations] you render the entire page with the JavaScript inside of it. So that's a slightly different approach. Furthermore the high-level structure seems right. We have the model and we have the transactions create, update and read, those we all need indeed. Status, — have you tried to select it in the tool? The workflow.*

S Yeah I tried.

Q *It was the only one that didn't work probably.*

S These you mean? [pointing at visualisation]

Q *Yes! But that one does not want to load, because it's an old remnant, it's some pollution in the data. So indeed, what you said is correct, —*

S But how is this created? [visualisation]

Q *It's being generated, I wrote a script for it. — What did you think of the assignment?*

S It was interesting. A bit difficult, but that's mostly because I know nothing or well very little about the bunq backend. But it's a good assignment.

Q *Yes? Okay. Have you used the features from the tool a bit?*

S I think I've used everything yes.

Q *That's good, then you have a lot of feedback for me. — You should hit the "Escape" button*

S [Tries to close visualisation] Ah yeah what I try every time is click somewhere outside such that it closes, but makes sense.

Q *yeah I'm used to doing everything with the keyboard, so [another developer] also mentioned this already as well like 'I miss the close button' and I was like 'but I always use Escape'. For me it makes sense.*

S Yeah I click mostly so yeah.

Q *Find by Label feature, have you used it, and for what purpose or why not?*

S I feel like all features are in there, or at least all labels. So I used it to find it. But because everything is already here I expected when I clicked it that it would continue immediately. Could be me though.

Q *Someone else mentioned the same so,.. it's not just you.*

S Anyway, searching is easy, also when you do not start with the first letter, I always like that.

Q *And the information that you can extract from this feature? When you searched for something —*

S For only this or..?

Q *For the Search By Label, and then the information that you get back. For instance when you randomly click a feature you'll get —*

S Ah then you also got the users, I actually clicked on a user right away to check what it was. So these then were the bigger projects [developer search view]. Here it lists all the features but is it relevant to know when someone worked on something?

Q *Well that's a nice question for you, is it relevant to know this?*

S [mumbling] — What order does it use anyway?

Q *That answer I do not know honestly. I get it back from the database.*

S It would be useful when it would order on most recently worked on.

Q *Most recent?*

S Yes!

Q *Ah. I thought of 'worked most on', —*

S But this seems like a super old feature. I'm not really sure whether all features are being actively developed. Because this is basically everything that exists within bunq right?

Q *Basically.*

S Is it ever "done" done?

Q *No, you will always encounter new bugs or new features.*

S What you mention is also possible, hmm —

Q *Yeah then I'd need some more toggles here.*

S Yeah such that you can choose.

Q *Yeah, good suggestion. Now it's a really basic toggle that hides the non-Feature Labels. That helps a lot already. But there are also priority labels so — But anyway, still looking at Find By Label, what do you think of the information you get there? Is it useful to you?*

S You mean this? [points at label view] Yeah I think this is nice, that you can see all those models, views and controllers.

Q *That you can filter them you mean?*

S Yeah, for me it was easy to find it back here then [PhpStorm] otherwise I would have no clue where to start searching, or well with a lot of scrolling.

Q *Yes, exactly, so you used this to find the filenames and then you go to the code —*

S Yes then I go to the code. And then immediately — you probably work on a feature when you work here — you can easily find it. I didn't really use this [library filter] by the way. But otherwise it was nice. It was nice to know who worked on it. Oh and the related.. — what was it based on again? Ah it's here, that's useful. I can't really think of how people would use it, but hmm.

Q *You can say whatever you think, no worries.*

S No, I don't really know, when I work on a feature it's useful to know what's related because probably I also need information about this where I should work on.

Q *That's exactly the idea behind it yes. When there's a lot of overlap indeed, then —*

S Yeah that's necessary indeed.

Q *We often see that when you work with Payments for instance then you have to create Mutations, Transaction Monitoring has to process it, and when you can look up for a certain feature what other features are touched then you can continue your search like that. — Okay. Cool. The Find By Developer, have you used it?*

S Only for that assignment [exercise 1], because furthermore the assignment explains clearly. It was a feature, and from what I've seen there were not that many and I think it looks clear, I find it useful to just click someones name.

Q *As in searching a feature first and then clicking someones name?*

S Yes, because then you know who belongs to which feature. But if you really want to look specifically for then this is really easy.

Q *Would you be able to think of a case where this would be useful?*

S Maybe when you know of someone specifically that he changed something, a specific commit or something, I don't really know. — I don't know whether it's necessary,.. you can only see the PARPs and Labels right?

Q *Yes. It first showed all files etc. as well, but when you search for instance for [a senior developer], he has been here for quite some years now so that became a long list and the overview was gone.*

S I think I mostly use the Label one.

Q *And the Find by CIT? Did you use that one? So say, specifically one issue. Instead of one feature.*

S Yeah that is useful I think. When you really work on it you can immediately search for that issue specifically.

Q *And what's useful about that?*

S [looks up `bugs#1714`] —

Q *Yeah that button doesn't work, it should have jumped to GitLab but [someone else] noticed it —*

S Oh but when it works that's really useful.

Q *You can hover the issue here and then still —*

S Oh I did that as well!

Q *I saw that GitLab had been opened indeed. The button I killed somewhere, but I never noticed. But this works as well, normally you could have multiple issues in this list though, and then it's not really trivial to click on the right one. But in this case it's just one. What would you use this for you think?*

S When you know that you have to fix a specific bug or when you need information about it. Because then you can jump to the issue immediately.

Q *And what kind of information would you look up here then? The issue you can also open via GitLab of course.*

S But then you can't find it this easily I think.

Q *Well it's being assigned to you, so in theory you can get an overview of your issues.*

S Ah. Yeah I don't really use GitLab issues, so I don't really know what to do with this. — These are empty [looks at other issues].? Or, no, views is empty.

Q *Could be that no views have been touched for this issue.*

S But can you see that as well here then? [GitLab]

Q *Yeah if you go to for instance [some senior developer's] issue then you can see the commits, and you see what issues are referenced, but then you have to check out commit by commit before you can see the related files. So it's possible, but it's not an easy overview.*

S Ah when it's in here [tool] then then that's useful. [looks at tool]

Q *Yeah the models and views are empty, which makes sense because bugs typically touch controllers. That's where the logics are. — What are the advantages, if any, of this tool?*

S You can get an overview of an entire feature fast and easily. That's something I find really useful. Especially that you can find all those models etc. here [points at filters]. And the people that work on it. You can just see everything at once. That's nice I think. And also immediately the related things, I think it's nice.

Q *What should be different about the tool according to you, what is not good about it?*

S The one thing I mentioned before, that it goes on when you click. But furthermore, the tool itself, no I think it's fine.

Q *And in terms of information, do you think "I would like more or less information"*

S I typically pay attention to commits because then I see the changes etc., but that might be something that should stay within GitLab, but that I do not know.

Q *You would like to see a list of commits?*

S Yes, commits with the changes actually, because then you see what has happened.

Q *So you want to see the list of commits and then — you really want to see the actual diffs or the files that have been changed within a commit?*

S No, diffs actually.

Q *Okay, the diff logs. —*

S That I find useful.

Q *You would like to see those added in the tool so that you can see it in one look?*

S Maybe in a tab, so that you know when this feature was last worked on. That's what I have in my head.

Q *Ah yeah, that seems useful.*

S Like, all commits get a label anyway, —

Q *Commits get an issue reference and —*

S The branch name then?

Q *That also includes it yes, and the issues then are connected to the labels.*

S Ah okay. I'm not sure whether it's relevant but okay.

Q *No okay, but I can consider it. Maybe more people mention it. Do you have any general feedback on the tool that you haven't mentioned before?*

S I think it looks good. It works well, we've been working on a tool ourselves but this looks kinda impressive.

Q *Sorry?*

S We've been working on something the entire time, so this looks already kinda impressive.

Q *Yeah, this took quite some time of course, when you'd have more time you'd also get more fancy —*

S Only the bunq logo is..

Q *Yes, I know, that menu bar is a bit messed up, I had four different menu items there before because I was really eager to add a lot and in the end I finished one of the tabs. And I had to wrap up, so I disabled all menu items and then the logo started looking weird. So that was a bummer. — Had you seen that when you view a feature that there's also a list of issues and search further from there on? [Click a generic label like "App: bunq"] — I love these labels, those really generic Labels.*

S From here I clicked yes. But when it's this long then you could better put it somewhere else. Can you only find it here or also per tab [filter bar]?

Q *Yes per tab it's still below it. The list below doesn't change, it's only the blob of files that changes.*

S Ah down here, — this is how I navigated to one of the issues before.

Q *Yes exactly.*

S Yes this I have used, also when I was scrolling, when I was searching.

Q *What have you used it for?*

S When making the second assignment

Q *Ah okay, to find more specific issues?*

S I thought I could get more information then.

Q *Okay, cool. What is your impression of the backend?*

S It's a lot. And I think it's a bit messy. But, in one way it's really structured, but maybe because it's this much that it feels messy.

Q *Do you have a clue why it feels messy?*

S Because this is such a long list [list of folders in PhpStorm]. I'm not sure whether it will be better when you put this in subfolders. It's just a lot.

Q *That's definitely true. Do you think it is possible that this tool could replace the help of seniors? Currently, as juniors, we often go to seniors for explanations. Do you think that this tool could reduce/replace this?*

S I don't think so, because I think that people will always have questions. But this really helps. I think you will have less questions because you are able to find more on your own. You can find what relates to a feature, but I think you will always need information from a senior; you can ask questions to a senior even if it's about this.

Q *Do you have an idea what kind of info you'd still ask a senior for?*

S Mostly how everything is set up and how it works because that I do not get from the tool.

Q *How it is set up; relations between the different components you mean?*

S Yes, yes, but more specific, globally I understand it, but when talking about a specific feature then then, at least in the beginning when you don't know that much yet, then you might need some more information. Just per feature or so. But I've only done this for two hours, I don't know when you work here whether you will understand better or not. I think you should always be able to ask some questions.

Q *Yes, okay. I'm not going to say you are wrong, not at all. Last question, what do you think is most important, being a developer, you are being assigned to a new framework and they tell you to study, what are the most important information sources then?*

S I am new and a developer, oh and, —

Q *You are new in a company and they say 'this is the backend, good luck'. What would you need to be able to learn it properly? What kind of information (sources)?*

S Something like this [tool] and someone I can ask all my questions to. Because — you probably have the repo but I wouldn't understand what all this is. At first. I wanted to say something like 'what if I had not been here and I would have had to work on this' — do people start working on a feature when they start working here or what?

Q *Yeah, often very simple tasks in the beginning such that you can* ⌘ + F *for a string, and then change it. You'll get more difficult projects soon enough, I got this [*[redacted]*] project after a month or something. I didn't even know that many of the PHP classes were generated from a JSON, it became clear after like three weeks of struggling, I had to realise that I had to click 'Generate' and then -poof- everything would exist.*

S It seems useful if there would be a feature that included everything. So that the person could understand that entirely then it would probably be easy to derive the others [next tasks] from that. Someone who is knew doesn't know anything, probably.

Q *Pretty likely, as it is no public framework. Those were my questions, do you have any questions for me?*

S Are they going to use this later? Or are they already using it?

Q *No they are not using it yet.*

C.4 Subject 4

Questioner (Q) *Can you tell me what you have approximately done and what you have searched for etc.?*

Subject (S) First I looked at this, —

Q *What is this? Ah, searching for Feature [redacted], yes —*

S Then I looked a bit at what's all there. Read the assignment. It was about the pre and post [processors], so I searched where those two are.

Q *Yes, so you searched for the pre and post processor in the list, yes —*

S And then I looked what workflow belongs to it, because I did not know which of the pre and post I was supposed to use. And then I noticed oh you can look only at post. And then I just opened some of those files to check what is in them. It was very old initially because I searched for files where [redacted] was in the name actually. And then I saw "addCommonFields", but you POST something, so I thought 'this looks interesting'.

Q *Where is this, ah ViewPreProcessorBody, okay —*

S Then I thought, okay maybe I should add the name here then, so I started looking where it was being used. And then I ended up in here I think.

Q *What's that, RequestResponse, yes —*

S That's being used somewhere I thought —

Q *Ah yeah a little above there is addCommonFields.*

S And then I saw the name user, and then I thought where is it mentioned, and then I thought 'here I should add the name then'.

Q *Ah, yeah. I had not realised you can reach it using this approach.*

S And this is where I got stuck a bit, like where it got send to, so I checked where it got used and then I saw the names here and checked which names occurred in both places.

Q *You are really going in the right direction, you already found where to add the field, and you already found where you can actually retrieve the user. I can show you the flow actually, that will make it a little more visual. [redacted] demo]. Are there any other things that you have searched for in the tool?*

S Yeah, assignment one. —

Q *Yes, the examples right?*

S Yes, and then I looked here whether I could see which Pre and Post processor belonged to it or which workflow.

Q *Okay, super cool. Then I think you can start working on assignment three.*

-

Q *Okay, tell me, what is needed to implement this?*

S The first two steps, 'returning an ID'-workflow and then step three is Create, TransactionCreate. That one transforms it into a the QR and then you need to match the QR to the person and afterwards create transaction.

Q *Do you know in which workflow it happened for — ehm, the matching, where does it happen, for* [redacted] *for instance?*

S DetermineSenderAndReceiver, hmm.

Q *There's another one related but this one also plays a role yes. And then?*

S Then in the same step you have to do the create transaction. And then it goes on to the next step. Where it checks whether it has already been paid and then the ProcessUpdate where it basically asks "is it already there". Which also happens in the last step but then the answer is "yes". And in between it performs an update with a Transaction.

Q *And did you identify other components next to the workflows? —*

S No, I thought that the workflows would contain the functions that I need.

Q *Okay cool. Then I also have a list of questions that I wanted to ask, what did you think of the assignment?*

S Yes Interesting, because I didn't know what I should think of it as I never really looked into the backend code. And it was confusing that when I first opened it it was like 'wow, a lot of folders'. So that was useful. And then I first checked here [tool] and then looked in the code so that you can link code to what is written here [assignment text].

Q *Yes exactly, okay, cool. Have you been able to use the tool?*

S Yes, especially in the beginning, and afterwards, especially with assignment two, I started looking at this [code]. So you first look and then you check where it is defined.

Q *When you have found the code that belongs to it [mumble..] So then I assume you've used the Find by Label feature a bit? — I think this even is a Feature By Label-search? Yes, what did you think of it? What did work, what did not work well for you?*

S Yes, it is more when you do not know the label name, that you then can see labels that are similar because now it says not found. When there's no result maybe a Label like, — wait I'm not sure whether payment is possible —

Q *It should auto-complete, unless you found a bug.*

S Then you have to choose.

Q *When everything is right then yes. I pull a list from GitLab of all available labels and those I show there. I first had a free input there, but when it doesn't exist it is not nice. And what did you think of the information that was shown?*

S When I saw it for the first time I thought 'okay, interesting', and then I started clicking around a bit, there are a lot of those bullets and I know that it refers to how the code is positioned, but it doesn't give information — you really need to read the name and when the name is unclear you know nothing. When they would have clear code [documentation] on top of the classes like 'this class does this' and 'this class does that' that would have been useful. But they don't have that, so I don't know how that could be improved. When you understand the name then you know what should happen but when you don't know what is supposed to do .. —

Q *So the naming of the classes is not always clear, then it's just a long list of bullet points which do not necessarily mean anything.*

S But I don't know how you could improve that.

Q *The idea now is that it follows the folder structure. And now you can see what files are related and then you can go back to the editor [PhpStorm] such that you can see the actual contents of the files. But there is no documentation as you have mentioned.*

S We have it everywhere.

Q *I know. The policy is though that the code should be self-explanatory. — Have you used the Find by Developer as well?*

S Yes, only for iDeal because I saw that a couple of things mentioned and there was also a name. And then I checked what he had done, so I knew what he had done, because I did not know which of the iDeal I needed [labels] as there were a couple of options. I didn't know which one to pick. So I expected that to be more clear when searching for his name, then I assumed that it would be Deposits.

Q *It did not become more clear from developer search?*

S Yeah well you saw everything that he had done, but I like the labels best because then you see everything with the same name. And the same name is often more related to each other. And with the developer you also have other components that are not related to each other.

Q *Yes exactly. So when you think of the developer part do you think that there are useful parts or do you think it can be removed entirely?*

S It could, actually the label is most — I didn't use this [CIT] one purely because I didn't know what to search — but developer might be useful for yourself, when you're a developer yourself to have a quick look at your own stuff. But to really search for something, not really.

Q *So you'd basically always use the "By Label"? Okay. Well I can skip the next question as you have not used the "By CIT" feature. Had you seen that on the bottom of the page there's a list of issues for this feature?*

S Oh yeah, but not really.

Q *Didn't use it?*

S No.

Q *Okay, well you can click to the next screen there, for instance for one of the issues you want to see all details. Then it will automatically fill in the fields, and then you can see what has been implemented for this specific issue. —*

S Oh, that's useful actually.

Q *May I ask what you think is useful about it?*

S Because then you have a single issue, which is a bug often, so then you can see what contributed to the fix, or what helped to fix it.

Q *Yes, my idea here was that whenever you need to continue on some bug, because it was not solved correctly in the end, then you can see what has been tried already for instance. You can search slightly more specific. Okay, did you have — what are according to you the advantages of the tool?*

S I think that you see what classes belong together. Because here you see that a lot of different models are being used with things that belong together and I wouldn't not think of looking into different folders for code that I'm looking for, because I expect it to be together.

Q *Ah yes, but it is a little more spread in this case.*

S That the most useful thing I think, seeing what belongs together.

Q *Okay, cool, are there also things of which you say it is not useful or it should be different or things you miss..?*

S Yeah, what I missed is not possible I think, a small explanation of what certain things are. The Find By Developer I found least useful, it didn't add much.

Q *Okay, cool. Do you have any general remarks on the tool, that you haven't mentioned yet?*

S I don't think so.

Q *Okay. What is your general impression of the backend, having seen it like this?*

S A lot.

Q *A lot?*

S Yes, a lot of code, big, it is — I don't code that often — but when you see it like this, like one line, really small methods, looks intense.

Q *Yeah, they really split it in small methods here yes. As soon as something can be considered as a separate step you are told to extract it in a private method.*

S Yeah, that is like 'wow, okay, interesting'.

Q *Do you think it is nice to read such code? All those small methods?*

S I like it when it has been written in slightly bigger methods. But that's purely because you will have more names and — I don't know — I like it when I can see what something is —

Q *The value you mean? —*

S Yes.

Q *I agree, the policy is though to not use magic values here, so yeah. Okay, what we do often as juniors is that we go to the senior developers like "I'm stuck here, please help" do you think this tool could prevent that, like the tool replaces a bit the senior.*

S Replacing I don't know, I think it will take longer before you need to go there because you know now that there's a different option. So you can first look there and then when you get stuck — because you should spent some efforts, they don't just give the answer, that it I think.

Q *And for what kind of information would you still need to go to a senior developer do you think?*

S Well, for instance when I was looking for the user, I didn't know at first where I could start looking best; in the preprocessor/postprocessor/view, etc. In what kind of class I should look. I think that when you hear the first step then it might be useful, but that might also be because I don't know the entire process yet, the pre/workflow/post, I didn't really see it yet, only later I saw it.

Q *Yes, okay, somewhat more generic questions, say you are thrown into such a big codebase you get an assignment and they tell you to do your thing. What would be good sources of information for you to learn such a framework? When you're thrown in fresh like that.*

S I'd Google first how the framework works.

Q *Okay, we have a custom framework, bummer.*

S Okay, then I would like to know the basic information [points at assignment text] telling me how the things work.

Q *Yes, so some written documentation?*

S Yes, so you know how the process works, it is divided into smaller parts in the code, and using this you can find what code. And then you know preprocessor that belongs to this step and then you can figure out yourself what steps there are. So then you know 'this is how it works'. That's useful to know.

Q *Yes okay. Do you have any questions for me?*

S No, I think?

C.5 Subject 5

Questioner (Q) *Okay, what have you done?*

Subject (S) I've tried to change that the model accepts the username in `core_[redacted]_[redacted]_transaction_model`. But I didn't see anything there so then — let's see what did I do — then I went into the definition, there I saw a JSON that contained all possible fields that were being returned. So I have added the username there, — or did I do it in the controller? [searches the changes] —

Q *I see here a User added in the Transaction Create, there another time —*

S And in the definition of the model I have copied the merchant name and then did the same for the user name. And then I have here, in the controller, added to the function execute and createNew[redacted]Transaction the ability to pass a new parameter as UserModel and make a dedicated field for that model. And then I tried generating but that didn't work out that well.

Q *Okay, had you also checked out the view definitions?*

S Here you mean?

Q *[nods]*

S No, I had checked out the workflows here, or well what would work better for me in how to solve it because it [editor] told me that in the paths it was undefined what a User was so I thought I had to change it somewhere else too. But I didn't know exactly.

Q *Okay, have you been able to use the tool with this?*

S Yes I checked what was related. And I saw here for instance this [?] so I knew this belonged to it, and that this was the definition then. And also with the views, which ones were related. So I did check it out, —

Q *Only the views you had not used yet?*

S Well I had tried in the views — oh that's what I tried initially, so in the view I tried here [?] username which I took from the counterparty name. Such as minified "cpn" and that kind of stuff. And type "string".

Q *Do you have any clue where the username should come from in the end or did you not get that far?*

S No, Well yeah I thought some model and then take the display name from there, but I wasn't sure how to get it from a specific user.

Q *Yes exactly. Then you can start with the last part of the assignment, you do not need to implement anything for that, you can use the code of course.*

-

Q *Tell me what have you done?*

S Okay my idea was to, just like with [redacted], at least to create a controller. And then I think we need a Create controller workflow in which it creates a transaction in the database and that one will return an ID then, in the view I guess. So then you have the view that return the ID with a JSON to the webshop. Then the webshop had the UUID and then it can go the view — or well, retrieve the JSON in which the status of the transaction can be found. That's a second view then. Then when the user scans QR it should update the transaction using another workflow with the user ID or a reference to the user. And when the user presses a button then another workflow should change the status in the view to for instance APPROVED or DENIED. Then the webshop will notice that. So it updates the view basically. And that works with a similar model for the AliPay such that it will be the same for [redacted] iDeal. So it has the same fields basically.

Q *I don't have much to add to that. That's basically it. Impressive. Did you like the assignment?*

S I liked it, I liked figuring out what I would do. But it kills me that it is sort of uncertain that I might be doing something that is really weird or something not-thought-through. But I liked it.

Q *No, you're right, there are some small things that are much deeper, so those are some specifics. But this is in high level exactly what you should do. Were you able to use the tool with it?*

S Yeah I checked it out to compare what [redacted] iDeal were doing. Although I could basically only get the information I wanted for [redacted]. For iDeal I wasn't really sure. I had searched for iDeal and then I saw only those three; deposits, incoming and outgoing. For incoming I didn't see anything and deposit is not really relevant for what I was looking for, —

Q *Ah this is that old that we probably used an old structure back then causing it to not be indexed, that's a bit of a bummer, probably when you use the All-filter you will see a lot of files. This probably uses a different file-structure —*

S Yeah I did see this here, and then I just checked out in the code what it was. I just searched by name basically.

Q *It uses regular expressions that filter this list and when there is no exact match then they won't show up, that's something others bumped into as well. The codebase is now structured like this, so for the new code it works perfectly but that has been introduced at some point and everything before then works less good. Anyway, you've been able to use the Find By Label feature?*

S Yes.

Q *And what did you think of it? Were there useful things, things that were not useful or?*

S I think it's nice you can just start typing something so you don't have to go through the entire list of possibilities. And I find it useful that — well everything actually, so I can what is possible basically.

Q *And in terms of information that you see?*

S I find it well structured, but when you're completely new to it it could be a bit unclear — ah that's also a little confusing that I have to click explicitly on search — normally it would immediately show the results, but okay.

Q *Initially the idea was to have multiple filters per block and then you do not want to search immediately. But I didn't have the time for that.*

S This [filelist] is a nice overview when you know about the structure. But some things have the same name, for instance when looking at the big list, so you can easily look at the wrong one. You see that this is more but —

Q *Ah yeah, the folder name is slightly different yes —*

S When you search for this and you look a bit too high

Q *Then you easily go to the wrong file yes, indeed. Had you used the visualisations, the graphs?*

S Yes, this one I used particularly to check what the workflows were.

Q *Do you think that this graphical representation helped you?*

S A little bit to know exactly what the workflows were for. For instance here I found out that it was intended for expiration and status updates. I mostly used it do gather some ideas for my own solution.

Q *Okay cool. Have you used the Find By Developer feature?*

S No only for the first assignment, but other than that not really. I only searched for iDeal [redacted].

Q *On the labels?*

S Yes. And mostly because I don't really know the developers. And I don't know what they have done exactly.

Q *Would you be able to think of a use case where it would be useful to search by developer?*

S When you know that they are specialised in some features within the application, so you can think 'what would he do in this situation'.

Q *Have you used the Find by CIT?*

S No, mostly because I didn't know the bug numbers.

Q *Ah I see. Had you seen that when you search for an label that on the bottom all related issues are shown?*

S Eh here? No I had not seen that. Oh wait, no I had seen this. I found it useful that I could jump straight to GitLab.

Q *When you use the left button it will immediately search by CIT and fill in the fields.*

S You know what I thought? Because of the placeholder 'bugs' I thought that only bugs could be entered.

Q *Ah yes. I used the placeholder to show what you should enter in which field.*

S Yeah I thought 'bugs, I don't know any bugs', so I was not really interested.

Q *Could you think of a situation where it would be useful or would you say that you do not know when you could use it?*

S When you are building something or in general?

Q *In general.*

S Well, when you want to know the current status of a feature I guess.

Q *Wouldn't you want to search by label then? Because then you can search for a feature specifically.*

S Oh no, I mean like an actual bug or something. That's the idea of this right?

Q *The idea is that can find everything related to one specific issue or bug.*

S So ain't that to look for the status of something? Like the progress of implementation?

Q *Depends on how often we index of course because it indexes after the facts. I can quickly explain a bit, the git history is used as a basis. Because then we have the commits with an issue reference. And then we also know the files that have been changed and the author of the changes. Using that I build up the initial part of my graph. Then I check what issues have been mentioned in the commits that I've seen. For those I extract additional info from GitLab, so the Labels, related issues, etc. That I connect all together, such that it becomes one big graph. And that's what's being queried.*

S That's how you build the overview then.

Q *Exactly.*

S So it is afterwards basically? And how often is that done?

Q *Now, manually. So two weeks ago it was indexed completely but I know that* [redacted] *hasn't been changed meanwhile so I know it is up to date for this feature. The idea is though to run it every hour or every time you merge something in develop or you name it. You need to think of a trigger then.*

S When you have fixed a previous issue, or it is fixed then you can look back.

Q *That was the intention yes. What are the advantages of using this tool according to you?*

S Quick overview for specific things, so when you look up for example iDeal that you have an overview of all models, views and controllers. So you know immediately what belongs to it.

Q *Are there also things that should've been different or that you did not find useful? Things that you miss?*

S Yes, features by developers, I feel like it won't be used that often. I can not imagine that a developer wants to know what some other person has done. Unless like with my use case you want to know how he approached something. Then you already know what the feature is about, what that developer has implemented, so then you could search for that.

Q *Okay. Do you have general feedback on the tool that you haven't mentioned before?*

S I think it looks nice! It looks neat and structured. And I think it's good that you split it in models, views and controllers and that there's also the 'all' option to see everything.

Q *Yeah the list become massive for some things. What do you think of the backend after this assignment?*

S I find it very big but it also feels structured. Good folder structure, everything belongs to each other and everything has clear names. But it is just quite impressive when you have to look into it and implement something out of the blue. A lot of experimenting. But it looks nice.

Q *Do you think that this tools could replace a senior as a Helpdesk towards the juniors? Now we often visit the senior like 'hey, how does this work', would this tool be able to reduce/replace that?*

S Partially I think yes. What I can't get from here is best-practices to implement something, like code structures. Not necessarily in terms of models but within the methods, deep inside the code, when you just joined a company as an amateur, that I would not be able to get from this tool.

Q *For that you still would need to go to a senior?*

S Yes, as a mentor basically, like 'how would you approach this?'.

Q *Okay, cool. A more generic question, you as a developer are thrown into a framework, it's new, you have no clue where to start. What would be your most important sources of information, or your most favourite sources of information? Where would you prefer to get your information?*

S When I know absolutely nothing?

Q *It's a new framework, you want to learn it, where would you prefer to get the information?*

S A very detailed description from a senior which explains what a model does, what a view does, etc. Such that you know 'i have to look there' when you want to implement something.

Q *A face to face meeting or?*

S Yes, I think so.

Q *And why?*

S It's easier to ask questions face to face compared to using a statical application. That person understands a bit better what I do not understand. So it's a bit easier.

Q *Cool, those were my questions, do you have any questions for me?*

S No. How many test subjects did you have yet?

Q *You are number five including the pilot person. You are number four that counts. Many to go.*

C.6 Subject 6

Questioner (Q) *What have you tried?*

Subject (S) I first went to the Feature, here —

Q *So you have searched for [redacted], yes. [confirming on-screen info]*

S Yes, and then I searched for this [? - context unclear] and then you get Issuer Transaction so you will see the model. So I started searching for that [in PhpStorm]. So I scrolled a little through everything and searching where I could maybe put a "getName". That's where I got stuck actually.

Q *Yes, exactly. Have you also tried to find some controllers and views in the tool instead of only the model?*

S Yes, I clicked a bit [controller visualisations], but I wasn't sure what I could do with the information.

Q *Do you know a bit about what happens in these workflows?*

S Yes, here you check whether everything is correct and then you execute it [certain [redacted] workflow].

Q *Yes, basically. And when you look like this, do you have a clue at what point the data is being returned to the user? So basically where the JSON for the user is constructed? Not necessarily in the current file.*

S It was somewhere in here [assignment text], there was some information about it. Where was it again — ah, here. So you mean when it is being returned, that happens in the post-processing then right?

Q *Yes, that one transforms it into JSON. So the idea is that the workflow has an output and that is transformed into a JSON. Are you now able to identify which workflow you would need for that, which workflow provides the output for [redacted]? — There are a couple of them listed here [tool].*

S Update I guess? Or Create? I don't know.

Q *When you read those names, what would you expect from those workflows in terms of what they do?*

S Well this creates a Transaction [Create] and this reads a Transaction [Read]. This updates one [Update], this gives the Status [Status; removed file], I think. It doesn't want to load this one [Status workflow visualisation].

Q *Oh, might be that the file doesn't exist anymore. There's a small thing that when a file did exist but does not exist anymore now, then it can't find it anymore. Because it pulls the latest version of the source code from develop. I did not yet find a solution for that. Yeah, but that is basically correct then. So when* [redacted] *now initiates a Transaction, that's the flow that is shown here [assignment], then* [redacted] *reads whatever they have initiated, because upon initiation they only get the ID back. So* [redacted] *says 'okay, we want the information of id' and then you say 'okay, for id it can be put to accepted/rejected'. When you now look again at the workflows would you be able to identify which workflow is being used by* [redacted] *to retrieve the data? — We have Create, Read and Update, and Status you can ignore for now.*

S I don't really know, Create?

Q *In this case it is the Read workflow, I think this is also partially a piece of mapping. In our Router when you, for example, make a POST request, it is always being forwarded to a Create workflow. When you make a GET request, to GET information, it always goes to a Read workflow. That's a bit of mapping that is not clear enough yet. So that's good to know. But in principle the Read workflow is used to read information from the database. And that workflow should then also return the name of a person. However, the name of the User is not present in the Model itself, it's one step deeper. The assignment was also about finding a flow; I have to pass it from here to there to there, I agree that that is not easy.*

S Yeah I had done something in the model. But I didn't really know what I was supposed to do.

Q *No exactly. Well, that's useful feedback as well.*

S So you would need to do it in the controller then?

Q *In the end the controller should pass along the information. But then you need to fetch that name from somewhere first of course. What I'd like to propose is that you start looking at the next assignment, that's more of a design experiment where you will be looking at what you need; what components you need. You have explored it a bit already, and I gave some additional explanations as well. So let's see whether you can manage to see like this whatever you need. I think it's useful to know about the mapping between POST and Create workflows and GET and Read workflows and Update and PUT workflows. Let's see whether you are able to figure it out and if not that's no issue, that just means my tool should be improved.*

S Or that I fail..

Q *Nah, no. —*

-

Q *Okay, what have you found?*

S I was reading about the steps here [assignment text] and first I had to make an API call so I started looking at the images of the controllers. And then here you create a Token for a QR request [TokenQr workflow]. And then I started looking at the code of Create. That one returns a model. So I thought it should return an AliPayTransactionModel in that case. And the Read controller takes the QR code of that model, which comes from the database then. And then, I was not sure, but I thought that the Update controller should check the status — when I look at the image [visualisation of workflow] — it checks whether it should be expired or not or what should happen to the status. But I'm not sure if.. —

Q *Yeah sounds correct globally.*

S And then the Status controller would update the status to what the user selects. Those were my thoughts basically.

Q *I see you also listed some Models?*

S Yeah, I looked at the Model, I looked at it a bit and we should also have such a JSON file in which everything is stored. Or well, all fields.

Q *Okay, and the views you have left empty? Did you not have time for that?*

S No I did not really have time for that.

Q *Do you have an idea of what should happen in there?*

S Where are the views [looking at screen] — ah here, yeah I'm not entirely sure how the views fit in. Because it starts with a view right? And that one calls..

Q *Yes, more or less, the API endpoint is defined in the view, —*

S And that one then retrieves something from the controller and that one asks for information from the model and that is being returned to the view.

Q *Yes, and then you get the story with transforming into JSON and sending it to the user. So what you see in the view is on one hand when you make a request what you should supply. So when you make a POST call in your system [Translations system], you should for example pass the translation. Well here it tells you what fields to supply when you make a call to this endpoint.*

S Should I have added a name field here as well for the second assignment?

Q *Here you should, — or well there's one level below even, here, the response, what you get returned —*

S Because I started typing here but I didn't know what I should put for 'minified' etc. So, I didn't know if that was supposed to be done.

Q *Maybe it's a thing you don't know about. Minified is to minify the amount of data that you return. So for instance when you're on your phone you want to send as less data as possible. It just saves money. So what we do is that we put an abbreviation of the field name here, such that instead of the full name we can send the minified version and that saves us almost half the letters.*

S Oh that makes sense.

Q *So it's not like — typically we use the first few letters, or when we use underscores the first letter of every part.*

S Yeah I got stuck a bit there because I wasn't really sure what to do there — I thought this was not the idea.

Q *It's good that you ended up there anyway. It's something quite specific that you didn't know about the minified. That's something that could be put in documentation —*

S Yeah I had the impression that I had to mess around with the JSON somewhere.

Q *Yeah that is right indeed. We work from the JSONs and from there we generate all the classes that we have. That's the idea indeed. In the previous assignment you had added a field here —*

S Yeah I had ⌘+Z'ed it, but I had it there indeed.

Q *Ah I see. But I recorded now that you had found it, so that's good. Anyway, then you add a field here, you only need to make sure that the field actually gets a value. And that happens in the processors. Then I have some questions about your usage of the tool. How did you like the assignment itself?*

S Difficult.

Q *Difficult, yes?*

S The third one was somewhat easier, but I think partially because of the explanation of you after the second assignment. Giving some first pointers.

Q *And was it mostly the mapping from workflows to POST etc.?*

S I was looking wrong at first, the images [visualisations of workflows] help a bit to understand what happens. I think that's a useful addition.

Q *Okay, cool —*

S Did you create those images?

Q *It's generated.*

S Oh nice.

Q *Like, what happens is that it reads the JSON data and with that it generates a graph. Have you seen the JSON of a workflow by any chance? [opens a workflow definition] When you look here then you see a 'start state', which you'll find back here and then there are transitions to these states. And between the brackets you can put a condition, for instance 'value === 5'. That's what you see here in the Update, these are the conditions on which it will transition. But yes, this is really generated. It's not a real image either, you can select all elements in it like this. It's a nice system.*

S It's a nice addition, maybe only a button to be able to return back, now you have to click on the side.

Q *I'm used to using the Escape button on my keyboard. You really notice that you're really used to things, you don't really think how other others.. — anyway. I saw you used the Find By Label. For what reason have you used it and what did you think of it?*

S This one right? Yeah, to look up those things [assignment] and it's okay, I can scroll through or type. And I don't really have comments.

Q *And what kind of information can you extract from here that you considered useful? When you search for this, what kind of information do you think 'this is really useful for me'?*

S Well when you click on [redacted] that see all the controllers etc.

Q *So the files that belong to it?*

S Yeah, this is also useful when you want to know it [other files], I haven't really used it.

Q *Related labels not used, developers —*

S Are they [related labels] really related to this? How does it work?

Q *How it is determined is by looking at the files touched for the implementation of this feature. Then it checks those files for what other features they have been touched, and then it ranks how many times they have been touched for another feature. And the more they have been touched by another feature then that feature is more likely to be related because they also caused file changes in the same files. That's the general idea. — The Find By Developer feature, have you used that one?*

S Yeah, for the first assignment. Worked fine as well.

Q *And what would you use that one for you think?*

S When you want to find out what you [author] have been doing for instance.

Q *And would you be able to think of a use case where that would be useful within bunq?*

S I think to know what people are working on, right?

Q *Yes, that's possible of course. I'm trying to find out what people would it for.*

S Maybe to find out who are working together or something. But you can't really see that clearly here. Maybe you can show here the people that are related as well. Like you did with Labels.

Q *Ah yeah that would be a possibility. Okay, cool. The Find By CIT feature, have you used that one?*

S Yes, also for the first assignment. One small remark that this didn't work [search for a PARP, get a 404-error]. Maybe that you have a dropdown here as well. Might be nice.

Q *Ah yes, so that you can see —*

S Does it show here what you entered the last time or?

Q *No, it's a placeholder.*

S I didn't really check it out thoroughly.

Q *Do you think it would be useful to be able to search for an issue like this in GitLab?*

S Yes, I think so.

Q *What would be the (dis)advantages then according to you?*

S Advantage is that you have a useful overview of what is involved with that bug. What belongs to it. I don't really see any disadvantages.

Q *Say, I throw a task on your plate, 'go work on this', for instance 'there's a bug in Card Payments', how would you approach that?*

S Find the bug, find out what's involved. — Is there like a bug description somewhere, by the way?

Q *Yes, it's in GitLab but then you need that button that's not working now.*

S Maybe you could add that, I'm not sure how big the text would be — and then I would start looking at the files I think.

Q *And say, that doesn't give any search results, would you have a different solution to search further?*

S How do you mean no search results?

Q *Say, you search and —*

S The bug doesn't exist or something?

Q *— for instance. Or it is not indexed yet. But you want to know already how things are tied together. For instance Card Payments or whatever.*

S I think I would go to the Card Payments based on the label.

Q *Okay.*

S I think.

Q *That's possible.*

S But that's when there's no GitLab issue yet or something.

Q *For instance, or it is not indexed yet. What happens now is that the tool looks at the Git history and for each commit it knows the time and CIT tag; that was mentioned in the assignment description as well. We, of course, have the author of a commit, and we know what files have been changed. Using that I know what CIT issue relates to what files and persons, and then I can check in GitLab what other issues it might relate to, what labels it has and that's how I build this information set. But I start with the commits. So, not all issues that are in GitLab are in my tool necessarily. Mostly what has been done in the past. Do you think there are advantages to using this tool?*

S Yes, I think so.

Q *If so, which ones?*

S I think mostly the overview is useful, and the visualisations seem useful. To get a better, especially in the beginning. I think when you know a bit better how everything works, that I would use it less.

Q *So, in the beginning to find the files?*

S Yes.

Q *Okay, cool. Are there also disadvantages of the tool or things that are missing? Things which you say of 'this should be different' / 'this should be added'?*

S At the developer screen, like I mentioned, you could add the people that are related to it. Other then that it looks nice.

Q *You mention the related developers, is that something that would be useful to yourself as a developer?*

S For my current project, not really, as I just worked together with them [other girl-interns]. But when working in a team I think I'd find it useful that when I visit an issue and I can see who's working on it, that I can see who else could be related to it.

Q *But, in principle, when you search for a bug —*

S Ah then you see everything already.

Q *— Exactly, so that's actually what you're referring to already?*

S Yeah, okay.

Q *As I understood from you now, you wanted to be able to search on a person and then being able to see which whom that person was collaborating. Did I understand that correctly?*

S Yes. [Looks through some users in tool; ends with author]. Maybe,.. would it be possible to immediately go there when clicking instead of clicking here [search button].

Q *Yeah, that would be possible. I tried to use the explicit button to not have the interface change when you did not want it to yet. The original idea was also to be able to enable multiple filters, and then it would suck when it would update every time you update a filter.*

S Are you still going to implement that? That would be useful.

Q *Yeah but not in the scope of my thesis. So, actually, no. Any general feedback —*

S Ah you also have other labels that are not for Features

Q *— Yes, that's why those are hidden by default. Not everything is useful. —*

S But the remainder [hidden items] is not interesting or something?

Q *Well it doesn't really tell what features I've worked on. I mean, "Priority: Hotfix", okay, I work a lot on Hotfixes. It can be useful, but it doesn't tell me what features I've worked on. That's why it's being filtered. Any general feedback that you haven't mentioned yet?*

S I think the design is good. I think the info box is useful. I think it's a good tool.

Q *Cool! I think this is an interesting question for you: what is your impression of the bunq backend? When you look at it, what do you think?*

S Complicated. But I think when you know how it's organised that it is nicely structured.

Q *Okay. Do you think it is possible, using this tool, to kinda replace seniors? Now juniors often go to seniors for questions, would this tool be able to reduce that?*

S I don't think so, maybe for the small things. I would not be able to understand the backend, just with this tool.

Q *Okay, and for what information would you then still go to the seniors you think?*

S Eh, everything.

Q *You say that for small things I can use the tool, so*

S Yeah, you know a bit the flow, but how everything works exactly that's not entirely clear.

Q *Yes, okay, so more how the different parts work together?*

S Yes.

Q *Then a more generic question, what are, according to you, the most important sources of information when you need to learn to work with such a framework?*

S Most important sources of information?

Q *Yes, where would you like to gather your information. Say, you are thrown into "here, this is a new framework, you've never worked with it", where would you like to be able to get information from?*

S Maybe like a help page. Which would shortly explain the backend. Because I read this now, which was helpful, but with the tool alone you don't have this [assignment text]. Well, I knew what Model, View and Controllers were, but when it's entirely new and you know nothing yet then you will need some more information still.

Q *So a help page you'd say —*

S A kind off start-page or something.

Q *Okay. Clear, those were my questions, do you have any questions for me?*

S Not really.

C.7 Subject 7

Questioner (Q) *Did you like the assignment a bit?*

Subject (S)

Q *Cool, wat heb je gedaan?*

S Nou, "zoals er stond" [not sure] een keer die tool gebruikt om deze op te zoeken. Toen heb ik die endpoint opgezocht met de tool, dus dat is deze. —

Q *De transaction view, yes.*

S En er stond aangezien het in de endpoint response moest zitten, heb ik dus de die post processor opgezocht. Dus hier. En wat ik eigenlijk gedaan heb is dat ik in issuer transaction een field heb toegevoegd voor username. Die dan hier in die response object body weer een nieuwe field aanmaak en dan die meegeven. Maar ik zou dus niet weten waar ik die naam vandaan zou moeten toveren. Maar stel dat we het mochten hardcoden dan gewoon, —het mocht gewoon een string zijn— dus dan zoiets [hard coded in the post processor].

Q *Had je nog gekeken naar de modellen die er bij komen kijken hier?*

S Ja, tenminste ik heb naar deze gekeken. En er zit verder niks in maar, even naar al die fields gekeken, maar dat was puur om te kijken of ik die naam ergens vandaan kon toveren. Maar achteraf gezien is het ook best logisch dat er geen naam in staat, want dat zou raar zijn als dat in iets anders dan UserModel zou staan volgens mij.

Q *Ja, in some way wel ja. Had je nog meer informatie uit de tool gehaald of heb je die view eruit gehaald en verder niks?*

S Ja, ik heb de tool alleen gebruikt — ja ik heb gewoon letterlijk die endpoint opgezocht met de tool. En toen vervolgens gewoon kijken die Response en dan gewoon in dinges [PhpStorm] zo opgezocht.

Q *Ja, oke, cool. Als je naar de tool gaat en bij Models, als je daar specifiek op zoekt —*

S Dus gewoon hier?

Q *Zie je hier nog modellen staan dat je denkt van 'Oh hier had ik ook nog wel wat mee gekund'?*

S Ja, achteraf — tenminste ik begon pas echt een beetje naar dat model te kijken zodra ik — toen ik echt specifiek echt iets eruit nodig had. Maar voor hier — in eerste instantie voor deze opdracht zou ik niet zo snel hebben van ik moet echt weten wat er in die models zit. Ja, je kan gewoon een beetje hier kijken naar de voorbeelden natuurlijk. Dus nee, ik had hier niet zo snel — achteraf gezien had het wel handig geweest maar nee voor de rest heb ik daar niet aandacht.

Q *Oke cool, duidelijk, dan stel ik voor dat je met de volgende verder gaat. En daar is het eigenlijk de bedoeling; maak een AliPay, basically hetzelfde als* [redacted] *maar dan dat je gaat kijken welke klasses, of welke klasse, welke componenten komen er allemaal bij kijken en dat mag je opschrijven, je mag het typen, dat maakt me niet zoveel uit en dat we dan straks door kunnen spreken dat jij zegt van dit en dit en dit moet er allemaal gebeuren om dit geïmplementeerd te krijgen.*

S Okay, maar je hoeft het dus niet te doen want dat zou heel veel werk —

Q *Dat zou heel veel tijd kosten, dus dat hoeft niet van mij.*

S Oke, dus gewoon aan de hand van de tool en alles wat ik hier beschikbaar is kijken hoe ik het dan aan zou pakken?

Q *Yes!*

-

Q *Oke vertel, waar ben je op uitgekomen?*

S Ook gewoon het hele denkproces erbij? Dat lijkt me wel het handigst?

Q *ja sure!*

S Oke goed, ik ben dus gewoon dit stappenplan gevolgd, goed. Zodra de user dus het eerste — die heeft dus die pagina die je liet zien met die transactie. Dat is dus zo'n HTTP request die wordt geroute naar de eerste view. Oftewel die komt in een PreProcessor terecht. Dus toen keek ik hier bij [redacted] [redacted], aangezien het daarop moet lijken. En eigenlijk het eerste dat ik zag was de TransactionView —

Q *De PostProcessor. [turned out to be PreProc]*

S Oke, goed, dan komt ie dus bij die PreProcessor terecht, waar dus eigenlijk alles -hoe zeg je dat- alles omgebouwd wordt naar een mooi Object, en dus die workflow gecalled wordt. Dat had ik dus voor het grootste gedeelte hieruit gehaald. Aangezien het in de backend moeilijk is om een overzicht te krijgen was het wel mooi dat dat hier dan zo stond. Dus was dan uit deze pakte ik die. Dus dat wordt dan aangemaakt. Vervolgens heb je dus die workflow nodig in een controller. En wat ik daar zag — wat mn gedachtegang telkens was was kijken wat hier nodig is en kijken hoe de structuur hier is en dan vervolgens in de backend code opzoeken van wat gebeurd er hoe ziet het er precies uit, hoe werkt het ongeveer. Dus toen keek ik bij die ene workflow, deze volgens mij, ja. Of nee, die QRGenerate, naja goed, daar vond ik dus die workflow en dan met dus die UUID die is gegenereerd in die preprocessor. Die wordt dus doorgestuurd naar die WorkflowQRGeneration die daar dus een QR code mee generate waarvoor die een aantal models gebruikt waarvan dus een aantal bestaande en aantal die je dan dus nog zelf zou moeten aanmaken voor dit. En d'r worden natuurlijk een hoop messages gestuurd dus je zal ook specifieke message models nodig hebben. Ja goed, dus die models daar haalt ie die QR code ook vandaan — of die wordt aan de hand daarvan gegenerate in ieder geval zoiets. Dus dat gebeurt dan in die workflow en via de PostProcessors wordt die QR code dan weer aan de User laten zien. Dus voor stap 4 —

Q *Ja, retrieval of the QR code —*

S Ja precies, dus die krijgt ie dan terug te zien. Ja goed, dan is dus eigenlijk het tweede stukje, dus dan wordt ie constant gepolled, hoe dat polling precies werkt kon ik er niet echt uithalen, vooral met die 5 seconds, maar goed —

Q *Ah, ja dat is ook niet echt —*

S Nee, dat is niet heel belangrijk. Dat gaat dan —wat ik zag— via een tweede view. Dus dan krijgt ie weer bij die PreProcessor krijgt ie die binnen en dan —

Q *Welke preprocessor?*

S Ja, dat is TokenQrRequest[redacted], ja — en die zag er vrij simpel uit dus het enige wat daar hoeft te gebeuren volgens mij is dus eigenlijk dat de workflow wordt gecalled die eigenlijk al het echte werk gaat doen voor zover ik zag. Dus daadwerkelijk de hele [redacted] nu AliPAy transactie gaat uitvoeren. Dat was dus een hoop GET dingen om voor wat is het allemaal - de GET request en de GET transaction en weet ik het allemaal. Oh ja, en hier ook vooral hebben we dus inderdaad dat de transaction PENDING is en kijken dus hoe staat de transactie ervoor en is ie geconfirmeerd of wat dan ook. Voor zover ik zag tot dan dat als ware gefixed is, dus dat ie daadwerkelijk confirmed is door de user. Waarna — en daarna is het hetzelfde verhaal met andere users die weer die models af en aan voor van alles en nog wat. Waarna die eigenlijk weer terug gaat naar die post processor die dus bij die andere pre processor hoort. En

dat dan weer naar een HttpRequest, terugsturen wat je daar dus aan het einde zag. Dat is een beetje wat ik hieruit kan

Q *Ziet er vrij compleet uit. Top. Helder. Dan heb ik nog wat vragen specifiek over de tool. Hoe vond je de assignment?*

S Ja prima, leuk, ook best wel goed om mijn backend kennis een beetje te testen, want dat — ik zat af en toe wel een beetje van heb ik er nou wel wat van opgestoken in de afgelopen weken maar op zich viel het niet tegen. Maar ik moet wel echt zeggen als wij deze tool aan het begin van ons project hadden gehad dan had het echt — dan waren we echt in week 2 qua kennis even ver geweest als in weet ik weel, week 5 of 6 ofzo.

Q *En waar komt dat dan voornamelijk door denk je?*

S Voor mij persoonlijk, ik denk vaak niet zo heel veel na, of ik doe niet echt heel veel onderzoek voordat ik ergens aan begin. Ik ga gewoon kijken en dan gaat het honderdduizend keer fout weet je wel en dan bijna op de meest hacky manier en dan gewoon net zo lang totdat je gewoon snapt hoe het werkt. En dit is eigenlijk de Too Long Didn't Read versie van een uitleg en zeg maar, dus ja ik vind dit zelf heel chill. Want je ziet gewoon eigenlijk best wel snel hoe het in elkaar steekt. Mits je ongeveer weet waar die controllers en views voor gebruikt worden. Maar het geeft wel snel een best wel goed overzicht, dus ik denk zeker voor nieuwe mensen dat dit echt heel chill gaat zijn.

Q *Oke helder. Dan wil eigenlijk stap voor stap er doorheen gaan, we hebben hier drie features; find by label, developer en CIT. De Find By Label, heb je die gebruikt, or not en why?*

S Ja, naja er staat hier al meteen het moet eigenlijk hetzelfde zijn als [redacted] ongeveer. Of gebruik het als inspiratie. Dus ja ik dacht meteen ik gooi hier in Find By Label "[redacted]" en dat pakte goed uit want dit was wat ik zocht. Hier heb ik eigenlijk alles mee gedaan. Ja, voor de rest, even kijken, omdat ik eigenlijk hiermee al alles had gevonden wat ik nodig had heb ik de rest niet echt gebruikt. Wat wel, het was waarschijnlijk wel handig geweest om — stel ik had vragen gehad over een bepaald onderdeel of zo van 'hey hoe werkt deze view of wat gebeurt er allemaal in deze controller?' dat je dan — ja dat je dan aan de hand van dat je weet dat het [redacted] is dat je dan hier — of dat je zeg maar dus zo kan kijken van 'he wie hebben daaraan gewerkt' dat je dan op die manier bij diegene terecht kan komen, bij de owner. Dus dat is fijn. Voor dit gedeelte heb ik alleen de eerste gebruikt.

Q *Oke, en was dat omdat je bij de andere onderdelen voor developer en CIT geen nuttige informatie kon vinden?*

S Nou, ja zoals ik al zei voor wat ik eigenlijk gewoon gedaan heb om het te maken had ik alleen die eerste nodig. Als ik op een gegeven moment hierop vastgelopen was van 'oke ik kan echt niet meer vinden wat ik nodig heb' dan had ik vast wel een van die

andere twee gebruikt. Maar puur omdat ik alles hier kon vinden wat ik nodig had heb ik het bij die eerste gelaten.

Q *Oke cool. Duidelijk. Wat zijn de voordelen van de tool, if any?*

S Naja, wat ik al zei, ik vind het heel chill dat het veel overzicht geeft. Zeker als je vrij nieuw bent is het fijn dat je zeg maar alles heel gescheiden van elkaar ziet. Dus dat je dan — naja — dat hebt. Wat ik fijn vond is dat als je — zoals bij die derde opdracht — echt iets gebruikt als leidraad om iets nieuws te maken, dat je heel — zeg maar snel een goed overzicht hebt dat je — stel ik heb iets nodig wat hierop lijkt — dan zou je in principe gewoon Control-C en dan PhpStorm Control-V en dan ben je er. En dat, die mate van overzicht haal je hier niet uit. Want dit is gewoon een bak met oneindig mappen dus dan — ja, hoe ik het normaal zou doen, dan zou ik hier gewoon [redacted] intypen [double-shift] en dan gewoon een beetje rond spitten tot ik iets krijg wat er een beetje op lijkt. Dus qua overzicht is dit veel chiller vind ik. Wat was de vraag ook alweer?

Q *Ja, wat je de voordelen vindt.*

S Naja, dat voornamelijk. Dat vind ik een groot voordeel. En je kan makkelijk — ja goed je kan het ook wel uit de klassen halen wie de owner en de maker is enzo maar dit is een fijn overzichtje dat je meteen ziet van wie er te maken hebben gehad met het geheel om zo maar te zeggen. Dus dat is ook wel fijn.

Q *Cool, zijn er nog ook nog dingen die anders hadden gemoeten aan de tool volgens jou? Dingen die misten, dingen waarvan je zegt 'dit had ook nog wel handig geweest als ik dit erin had kunnen vinden'?*

S Ja, ik dacht dat ik op een gegeven moment wat had maar.. — ik zou het zo snel niet weten eigenlijk, voor wat ik nu heb moeten doen was het eigenlijk heel erg goed te gebruiken. Dus zo snel,.. — ik kan er natuurlijk nog wel even over nadenken, want daar heb je natuurlijk het meeste aan — maar voor dit zou ik zo snel niet iets verbeteringen kunnen bedenken. Want het is natuurlijk ook — je komt er natuurlijk ook achter als je het in andere situaties gebruikt, weet je wel. Als je specifiek aan het zoeken bent op een bepaalde bug ofzo en dan kom je erachter van 'oh ik mis dit' maar zo ver zou ik het niet kunnen zeggen.

Q *Oke, heb je toevallig gebruikt gemaakt van deze issue jumps hier [bottom of find by label view]? Of had je dat niet gezien?*

S Nee, ik heb ze gezien, dat was op zich wel nice dat je dan een soort van overzichtje had van 'he hoe verhoudt zich dat nou?' want dat CIT is natuurlijk wel chill voor alle niet-backend mensen dat linkt het een beetje aan elkaar. Ik heb er voor de rest geen gebruik van gemaakt voor deze opdracht want ik zag niet echt hoe ik het nodig zou hebben voor nu.

Q *Ja je kan hiervandaan als je die groene klikt kan je direct zoeken op de Feature by CIT. Dus je kan dan wel gelijk doorklikken. En dan zie je dus voor een feature — of voor een issue wat er gechanged is.*

S Oke, dat is nice. Ja, goed zoals ik zei ik had het hier niet echt voor nodig maar ik kan wel begrijpen dat iemand dit . . . —

Q *En waar zou iemand dit dan voor nodig kunnen hebben denk je?*

S Het is moeilijk om nu een beetje buiten de scope van deze opdracht natuurlijk wat te bedenken. Misschien om het gewoon weer terug te linken aan een bepaalde — voor zover ik het nu zie is het vooral chill als een shortcut om het zo maar te zeggen. Dat je niet zelf weer handmatig het erbij hoeft te zoeken. Dus, qua bugs, ja, ik zou niet zo snel iets gaan debuggen wat door iemand anders gemaakt is. Of dat je dat dan. Het is dan gewoon als je echt gaat inlezen misschien dat je dan kan kijken van wat was een bug en als je iets vergelijkbaars maakt hoe kan je dat dan vermijden dat je dan gewoon meteen in een oogopslag zo daar naartoe kan en dat kan gaan doorspitten op GitLab. Dat lijkt me nice. En op diezelfde manier dat je gewoon — ik zou zeggen — het voornaamste voordeel is dat je snel doorgelinked kan worden dat je niet zeg maar zelf moeite hoeft te doen om het erbij te zoeken.

Q *Ja. Oke, duidelijk. Heb je in general nog feedback op de tool die je nog niet hebt genoemd?*

S Ja, zo snel nog niet eigenlijk. Waar, naar wie is het gericht? Is het gewoon gericht om het leven van developers makkelijker te maken?

Q *Het is initieel bedoeld om nieuwe developers sneller op weg te krijgen. Zodat ze sneller aan de slag kunnen. Maar ik ben dus ook aan het kijken of het eventueel voor juniors of mensen met al iets meer ervaring of het dan ook nog steeds nuttig is. En welk deel dan nuttig is.*

S Ah oke. Ik denk dat je dat heel goed bereikt hebt. Zoals ik dus ook al zei, als wij dit aan het begin hadden gehad dan had dat ons leven echt veel — jij hebt ons toen op een gegeven moment even geholpen met de backend uitleggen — als je dit hebt — misschien als het echt voor nieuwe developers is bedoeld dat je dan zeg maar dit verhaaltje dat je hier [assignment] aan het begin vertelt dat je dat ook beetje hierin zou kunnen includen. Want, dit is heel nice om te gebruiken, mits je wel weet wat alles doet. Er zullen een heleboel mensen zijn die al ervaring hebben met Model View Controller. Maar als je precies weet hoe dat bij bunq gebeurt dus eigenlijk wat er hier in deze introductie uitgelegd staat, dat je dan een heel compleet — zeg maar for beginners ervan maakt.

Q *Dus nog wel de uitleg erbij zeg maar?*

S Ja, kijk, bijvoorbeeld ik had zelf nooit echt zo'n strenge model view controller structuur aan de slag — was ik echt aan de slag geweest dus ik zat aan het begin van oke —

what the f*, dit duurt wel even voordat ik dit snap, maar als ik dit gewoon had gelezen aan het begin en daarna dit had gehad dan — om een beetje dingen door te kunnen kijken in plaats van gewoon van 'nou, hier is ie' weet je wel, dat was echt — ja, dat had het hele proces wel een stuk sneller gemaakt. Ja, dus een soort van introductie waarna dit komt als het ware.

Q *En zou het denk je helpen als die structuur, zeg maar dat schemaatje dat in die introductie staat, wat meer terugkomt in die interface? Zou dat ook al helpen?*

S Ja, ik denk het wel. Wat je dan misschien krijgt is dat het voor mensen die dit gewoon kunnen dromen allemaal dat die dan een beetje dat weer telkens moeten zien. Net zoals in een spel wat je heel vaak gespeeld hebt telkens een introductie krijgt bij iets. Dat dat misschien een beetje storend wordt. Dus dat je —

Q *Ja ik zie het dus nu voor me net zoals nu — ik weet niet, heb je dat gezien dat je van de workflows schema's kan... — ik weet niet of je er eentje hebt aangeklikt toevallig?*

S Oh ja, ik heb volgens mij op een gegeven moment — ja, dit was cool dit was echt —

Q *Ik zie dat nu dus voor me dat je dus dit schemaatje hebt en dan gecombineerd met de inhoud; welke workflows. Dat je het op die manier nog duidelijker kan maken voor nieuwe mensen.*

S Ja, ja hier kan misschien nog wel het een en ander aan toegevoegd worden qua verduidelijking of uitleg. Op zich als je zeg maar echt weet wat je wil dan is dit best wel voldoende. Achteraf gezien had ik dit ook heel goed kunnen gebruiken voor dit.

Q *Je had het niet gebruikt?*

S Nee, ik heb er op geklikt maar, maar ik heb me niet gerealiseerd, ik heb gewoon dit gebruikt en toen opgezocht in de dinges [PhpStorm]. Ja dan heb je alles gewoon, methoden met alle code erbij in plaats van een blokje. Dus ja, ja misschien dat je hier misschien een wat meer completer — of een, naja het is duidelijk dus ik zou het niet echt duidelijker willen noemen maar meer gewoon meer informatie erbij of zo. Zoiets.

Q *Nee, helder. De bunq backend, wat is jou indruk daarvan?*

S Ja, ik vind het best wel nice. Ik moet zeggen dat ik best wel groot fan ben van als het ware dat de code zelf vertelt wat het doet. Dus dat het in een opslag dat je kan zien wat de functionaliteit is, want ik was ook net aangenaam verrast door hoe gemakkelijk ik dit hele process kon doorlopen zonder het echt zelf geschreven te hebben, met gewoon toch — ik bedoel, mijn backend kennis is toch minimaal in vergelijking met de rest die hier rondloopt, namelijk 10 weken. Dus toch met vrij beperkte kennis dat ik er zo vrij gemakkelijk nog doorheen kon. En het proces, in ieder geval high-level, best wel goed kon begrijpen. Ja, dus daar ben ik best wel over de spreken. In plaats van dat je zeg maar — stel deze methode was met weet ik veel, UserModel X en dit Y, en dan

met zo'n lap text erboven in een comment, moet ik wel zeggen dat ik dit wel een stuk fijner vind. Dus dat vind ik wel.. —

Q *Heeft de tool daar ook nog aan bijgedragen, die leesbaarheid in that sense?*

S Qua leesbaarheid van de code, nou niet echt. Het fijne is dat je door de tool weet waar je moet zoeken zeg maar. Dat je weet waar je aan toe bent ongeveer. Als je het echt gaat uitpluizen zoals nu. En ja dan op een gegeven moment als je de tool gebruikt dan kom je bij de code terecht en de code op dat moment spreekt dan voor zich. In mijn ervaring. Dus ja, leesbaarheid niet echt, maar echt je weg weten te vinden in de code heel erg.

Q *Oke, helder. Wij hebben nu natuurlijk als juniors dat we vaak naar seniors gaan voor uitleg, denk je dat de tool dat deels zou kunnen opvangen slash vervangen?*

S Ja dat hangt er een beetje vanaf wat voor uitleg het is. Bijvoorbeeld toen wij naar jou toe kwamen van 'he hoe werkt die backend eigenlijk' dan ja absoluut. Dit had dat algemene praatje echt perfect op kunnen vangen. In combinatie met wat informatie wat hier nog in stond. Voor de rest, waarvoor zijn wij naar seniors gegaan? Dat waren echt wel een beetje de vage dingen weet je wel, van 'he ik haal nu een grote CSV file binnen, hoe kan ik die het best opslaan?'. Dat soort dingen. Dus dat is een beetje weer te specifiek voor de tool denk ik. Dus ik denk dat je in principe tot op een code review wel heel ver hiermee kan komen. En ja dan code style enzo dat heeft er natuurlijk weinig mee te maken. Die gaan in die reviews eruit komen. Dus ja, het kan denk ik voor een heel groot gedeelte kan het voorkomen dat je zeg maar naar een senior gaat van 'he moet ik hier een workflow voor maken', want je kan makkelijk voorbeelden van — zeg maar het enige wat je misschien zou vragen van helemaal aan het begin 'he ik wil implementatie X maken, is er iets waar ik naar kan kijken wat ik als voorbeeld kan gebruiken?'. En dan vervolgens kun je de tool gaan gebruiken om die uit te spitten.

Q *Oke, helder. Jij moest hier aan een nieuwe codebase gaan werken, die was compleet onbekend voor je, wat zijn dan voor jou de belangrijkste informatiebronnen, waar haal je het liefst je informatie vandaan om dat te leren?*

S Je bedoeld nu gewoon in deze context?

Q *Ja, of een ander framework, als je zegt van daar haal ik altijd heel graag mijn informatie vandaan dan..*

S Ik weet niet, qua wat ik ook zelf gedaan heb met gewoon programmeren bij bunq is dat het eigenlijk dat ik heel weinig internet gebruikt heb. En heel veel gewoon proberen, vragen en voorbeelden kijken. En echt voor bepaalde PHP specifieke dingen dan wel gewoon opzoeken in de PHP documentatie online en voorbeelden, StackOverflow enzo. Stel ik zou een hele nieuwe implementatie moeten maken zoals ik al zei, dan zou ik als die tool zou bestaan of als die er was geweest dan had ik waarschijnlijk daar wel meteen naar gegrepen. Gewoon om puur om te zien van hey — naja wat ik

zei, dat je eerst vraagt 'waar lijkt het op' en dan vervolgens de tool gebruiken om te kijken hoe dat dan in elkaar zit. En gewoon dat hele ding uitvogelen. Ja, dat is hoe ik het nu zou aanpakken. Meer kan ik er eerlijk gezegd niet van maken.

Q *Oke, hartstikke cool. Dan waren dat mijn vragen, ik weet niet of jij nog vragen voor mij hebt?*

C.8 Subject 8

Questioner (Q) *Cool, can you tell me what you have done?*

Subject (S) Not that much really. I've added 'username' here, —

Q *— in the view yes —*

S And then here in the 'createNewResponseBody' I tried to get the username from somewhere, in the Transaction model, but I couldn't find it anywhere from which I knew.

Q *Okay, and where have you tried to find the username?*

S I tried the RequestResponse and.., there were a couple of things you could use to get a UserModel from which you could get the name then. But this one turned out to be null often [RequestResponse]. And I could've written a function where it checks whether it's null or not, and you do not only have 'createdBy' but also a similar thing. I could have tried that, but I'm not sure whether it would've made a difference.

Q *You're going in the right direction. You have found the right model. How did you find this RequestResponse model?*

S Actually just this [autocomplete]. —

Q *Ah, with the autocomplete —*

S — and see what's in the list. Yes. I saw it in this list here as well.

Q *Ah, here in the model definition yes. So you were able to find some information in the tool as well?*

S Ja, alleen toen het bleek dat ie ook null kon zijn begon ik twijfelen of dit wel de manier was. Dus toen ben ik er uiteindelijk mee gestopt.

Q *Zeg maar het idee is de flow die ik je liet zien, die transactie wordt aangemaakt, dan wordt er een RequestInquiry aangemaakt, een betaalverzoek, op dat moment weten we nog niet welke user er aan vast hangt (die Inquiry). Zodra hij hem scanned dan assigned ie de inquiry aan een user en als je dan op een van die twee knoppen klikt dan krijg je de response. En dan pas is het ook niet meer null pas zeg maar. En daar komt het dan uiteindelijk vandaan. Zolang we nog niet weten welke user het is staat ie op null.*

S Maar, hoe had ik het dan moeten programmeren in dit geval?

Q *In principe was dit de goede manier, en je geeft gewoon null terug als er nog geen naam bekend is.*

S Dus er hoeft geen check omheen?

Q *Precies, want je weet niet.. — pas op het moment dat de QR gescand is weet je pas wat de username is. Je moest inderdaad naar het RequestResponse model en daarvandaan kon je de name vinden. Dus dat heb je er goed uit weten te halen. Heb je de tool daar verder nog bij gebruikt? Ook voor andere dingen, of?*

S Nah, barely I think. To find this [Transaction model]. Hier heb ik wel even naar gekeken. Maar verder, waar ik die view kon locaten, ik werk natuurlijk al tien weken in de backend dus dat was gewoon even dubbelklikken of dubbel shift en toen had ik die wel.

Q *Ja, dus dat wist je uit je hoofd? — oke.*

S — Verder geloof ik niet dat.. — Toen ik vast liep ben ik wel gaan kijken daar, maar toen kwam ik ook niet echt verder daarmee.

Q *Wat had je nodig gehad om verder te kunnen komen dan, wat mistte je?*

S Ja, goede vraag, eigenlijk vooral de bevestiging dat de user die ik eventueel uit de response zou kunnen halen dat dat wel de juiste user was hiervoor. Als ik daar bevestiging op de een of andere manier op had gekregen dan had ik er wel even een methode mee geschreven van.. Maar ik dacht ja ik heb ergens een UserModel vandaan gehaald maar dat kan van alles zijn. En die is op dit moment ook null dus ja.

Q *Ja oke, op die manier.*

S Maar of dat in de tool —

Q *Dus dat zou dan een soort van — je had willen weten wat een RequestResponseModel is eigenlijk, wat het voorstelt —*

S Ja precies.

Q *Ja oke, fair. Oke, nee helder, dan — ik denk dat je dit nice hebt gedaan. Goed gevonden.*

S Ja, nja, ik heb de afgelopen dagen ook in views gewerkt dus dat maakt het —

Q *Ja natuurlijk, maar ik ben nu aan het vergelijken van juniors met externals met seniors en dan kijken wie welke informatie nodig heeft. Dus waarschijnlijk een senior zegt ook 'pff, daar heb ik dat niet voor nodig, dat weet ik zo ook wel'. —*

S Ja natuurlijk —

Q *Maar dat ben ik gewoon aan het vergelijken. Okay, right, dan denk ik dat je naar de volgende —*

-

Q *Oke, wat heb je allemaal nodig?*

S Het eerste is eigen view, AliPayTransaction, waarmee 't begint is dat ie een request ontvangt, de eerste keer dat ie dat ontvangt is dat een CREATE request, bij een CREATE request roept ie de CreateController aan die maakt een nieuw model, een TransactionModel en die wordt dan weer teruggestuurd. Vervolgens de volgende request vanaf de webpage dat zijn dan READ requests, die zenden, ja ook via Controller natuurlijk, dat model gewoon terug. Model heeft onder andere een status. Op het moment dat die status veranderd is dan merkt die webshop dat en zal daar dan mee dealen. Tot slot heb je nog een update model controller. Die zal aangeroepen worden op het moment dat de user danwel bevestigd of denied request heeft of als ie out timed is en eventueel nog andere dingen. Maar daar heb ik verder niet over nagedacht. Het enige dat ik lastig vond is waar die precies aangeroepen wordt. Dat heeft iets met die RequestResponse of die QR Request te maken. Dus ik dacht RequestResponse als die confirmed is dan update de model en hetzelfde als ie denied is en de QR bepaald ongeveer of ie gevalidate kan worden of zoiets. Maar dat snapte ik niet precies.

Q *Ik denk dat je 99% van de flow hebt, inderdaad RequestResponse die — zodra je de bevestiging hebt, dat is een RequestResponse confirmed type, dan propagate ie dat naar het Transaction model van 'oke, ik heb hier een bevestiging, dus deze is nu completed'. Dat is wat je dus zei eigenlijk. En de QRRequest is basically voor het QR token dat je voor een RequestModel een QR entry de app in hebt zegmaar. Dat is eigenlijk die QRRequestToken. Ja,.. that's it. Ik heb er niet veel aan toe te voegen. Ziet er goed uit. Cool, dan heb ik wat vragen nog achteraf over het gebruik van de tool. Hoe vond je de assignment zelf?*

S Ja, prima. Goed te doen.

Q *Oke, mooi. Dan hebben we hier drie features bovenin zitten, je had de find by label, developer en CIT. En dan wilde ik beginnen bij de Find By Label, of je die hebt gebruikt en zo ja voor wat, waarom,..?*

S Ja, die heb ik gebruikt, onder andere om dit te vinden [[redacted]TransactionModel], [redacted], de structuur daarvan. Tenminste, dat is voor die derde opdracht, voor de tweede opdracht heb ik er nog wel even naar gekeken naar die — ja eigenlijk om hetzelfde te vinden. —

Q *Om de structuur te vinden? —*

S Ja. —

Q *Heb je er verder nog informatie uit kunnen halen?*

S Uit dit gedeelte [find by label] of..? —

Q *Van deze feature, ja.*

S Ja, ja inderdaad die structuur. Wat wel handig was geweest vond ik is, je hebt hier die bijvoorbeeld die update controller, alleen kon ik niet echt vinden waarvandaan die werd aangeroepen. Dus als daar een feature voor zou zijn dan..

Q *Zodat je ziet hoe de workflows aan elkaar gekoppeld zitten.*

S Ja, vooral. Ik weet eigenlijk nog steeds niet waar ie aangeroepen wordt. Waar die op een gegeven moment gecalled wordt inderdaad. Want die andere die zijn duidelijk, die Create and die Read komen duidelijk uit die views want -dat zie je hier misschien niet- maar

Q *Ja in de code —*

S Ja in de code. Maar die update niet aan diezelfde views. En dat zou niet logisch zijn voor wat ik bedacht heb dus. Dat zou nuttig zijn. Ja, verder viel het me hier onder andere op na die structuur ben ik naar dit soort klassen gaan kijken van wat doet het nou echt en wat heb ik nu nodig.

Q *Zat er veel overbodige informatie in?*

S Ja weet ik niet zo goed. Ja, ik weet niet of ik dit bijvoorbeeld nodig had, maar dat is daarvoor wel nodig dus dat is dan niet noodzakelijk overbodig. Nee, nee niet echt denk ik.

Q *Nee, oke, dus het was in ieder geval wel dat het nog een beetje samenhing en dat je niet dacht van dit heeft er echt niks mee te maken. —*

S Nee klopt. Ja ik weet niet of dat op een of andere manier mogelijk is maar iets meer dependencies van hoe dit hiermee te maken heeft. Zoiets zou wel nuttig zijn.

Q *Waarom het allemaal samen in een view staat hier —*

S Ja precies.

Q *Oke helder. The find by developer, had je die nog gebruikt?*

S Nee. Niet voor de opdracht. Want ik het volgens mij gewoon vinden via de labels toch?

Q *Ja, dus je had niet het idee dat je daar informatie kon vinden?*

S Nja, niet meer dan dat ik nu gevonden had.

Q *Oke, duidelijk, de Find by CIT?*

S Ja, daar geldt het zelfde voor.

Q *Had je gezien dat je onderaan via die issues kon doorklikken naar de Find by CIT?*

S Had ik gezien ja, ik heb het niet gedaan maar, —

Q *Oke, had dat nog een reden dat je dat niet had gedaan of?*

S Ik heb op geen moment gedacht, ja zo'n issue is meer wat er per keer geüpdatet is toch? En ik had meer de totale overview nodig. Dus niet gevoel dat ik het nodig had.

Q *Wat zijn volgens jou de voordelen van het gebruik van deze tool? If any?*

S Naja, het is een stuk overzichtelijker dan wanneer je die 1800 classes in de — of die 1800 mappen in de code moet bekijken. Het staat nu wel duidelijk gegrouped. Dat is het grootste voordeel denk ik. Dat het overzicht creëert.

Q *Cool, je noemde net al een paar die dingen die je wel graag erbij had willen zien. Zijn er nog meer dingen dat je zegt 'dat had anders gemoeten', of 'die mis ik nu' aan informatie of 'als ik die informatie had gehad dan had ik het nog beter kunnen oplossen'?*

S Nee, niet perse eigenlijk.

Q *Dus alleen wat je al genoemd hebt?*

S Ja eventueel ook nog dat er iets is om die views en models te visualisen ofzo. Want bij de controllers kun je dit doen [clicks open] en dat helpt wel al een beetje dat je een beetje die flow ziet, dus als je bij de Models de fields zou zijn bijvoorbeeld, zoiets.

Q *Heb je in general nog feedback op de tool?*

S Nee, niet meer dan ik net genoemd heb.

Q *Cool! Wat is je indruk van de bunq backend? Als je dat zo ziet?*

S Ja wel — ik heb er niet heel veel ervaring mee met andere bedrijven — maar wel, zit wel mooi in elkaar vind ik. Toen ik begon met programmeren 10 naja 9 weken geleden toen hadden we maar gewoon wat dingetjes geschreven en uiteindelijk is het zo erg veranderd en daardoor is eigenlijk automatisch het backend stuk code veel beter geworden en veel logischer.

Q *En dat mooie zit hem dan in?*

S Naja, omdat er zo'n bepaalde structuur en checks in zitten zorgt dat ervoor dat je code kwalitatief wel goed moet zijn. Je kunt niet gewoon een beetje bij elkaar hacken. Dus dat vind ik wel mooi. En verder, model view controller dat is redelijk basic voor bedrijven denk ik. Maar, zit wel goed in elkaar volgens mij.

Q *Deze tool, we zijn nu vaak als juniors naar een senior gaan van 'he hoe moet dit', denk je dat deze tool die seniors, die hulp, zou kunnen vervangen of in elk geval deels kan opvangen?*

S Ja, vind ik moeilijk in te schatten, aangezien ik zelf natuurlijk ging ik eerder naar jullie [juniors] toe

Q *Je mag het ook daarop reflecteren —*

S Ja klopt maar dat waren vaak nog meer vragen die nog meer basic zijn dan je in deze tool kan vinden. Ik zou het niet met zekerheid durven zeggen.

Q *En wat voor informatie zou er dan nog in de tool bij moeten staan om het wel daarvoor usable te maken?*

S Even denken, ja, weet ik niet. Het is soms zulke basic informatie maar als je ergens voor de eerste keer naar kijkt dan weet je dat nog niet. Niet meteen. De vragen die we soms gesteld hebben.. Maar ja, ik denk niet dat je als deze tool de wikipedia wilt vervangen bijvoorbeeld voor die basic vragen.

Q *Dus het zou dan een beetje de style guides of conventies zijn?*

S Ja, en gewoon hoe je dingen gebruikt, dat soort dingen. Ik denk dat het nuttiger is als je al iets meer van de code snapt dan wanneer je net begint.

Q *Helder, dan een wat generiekere vraag. Je wordt zo in een nieuw framework gegooid, compleet onbekend voor jou, wat zijn voor jou dan de belangrijkste informatiebronnen, waar haal je het liefst je informatie vandaan?*

S Dus je bedoelt in de backend, dat framework?

Q *Niet specifiek dit framework maar als jij een nieuwe framework moet leren waar wil je dan het liefst je informatie vandaan halen.*

S Het liefst heb ik eigenlijk dat het zoveel mogelijk voor zich spreekt. Dat ik er gewoon doorheen kan bladeren en dat het duidelijk is. Wat waarbij hoort en hoe dingen werken. En anders, iets van een tooltje of iets van wikipedia waar gewoon heel duidelijk schematisch staat hoe iets in elkaar zit. En niet hele lappen text want dan weet ik wel dat ik dingen ga overslaan.

Q *Ja, dus toch een beetje de schema's het visuele meer dan,.. — Ja oke. Helder, dat waren vragen ik weet niet of je nog vragen aan mij hebt?*

S Nee hoor.

Q *Nee? Oke, dan bedankt voor je tijd!*

C.9 Subject 9

Questioner (Q) *Vertel, wat heb je gedaan?*

Subject (S) Nou, ik ben natuurlijk eigenlijk gewoon gaan kijken van 'waar zit die view?'. Maar dat weet ik wel.

Q *Dat wist je al omdat . . . ?*

S Naja, backend ervaring in feite.

Q *Dus hoe heb je hem gevonden..?*

S Dus ik heb uiteindelijk — naja daar heb ik uiteindelijk wel alvast een extra veld username toegevoegd en toen zag ik in de POST van 'he, dit wordt op deze manier gedaan' dus laat ik in ieder geval hier eerst mee beginnen. —

Q *Ja, maar hoe had je de view zelf nu gevonden? Want je zegt backend ervaring maar..?*

S — Nja, Shift Shift deze [filename of view] heb ik daar ingevuld uiteindelijk —

Q *Ah oke op die manier.*

S — en toen kwam ik er dus. En toen had ik die view. Toen dacht ik van in dat model moet ik 'm dan erin krijgen. Naja, die staat hierboven zeg maar. Toen ben ik gegaan voor de relatie alleen toen gaf ie eerst nog een error. Dus die heb ik uiteindelijk weer weggehaald. Maar toen ben ik eigenlijk gaan nadenken van 'he waar moet ik hem er nou uithalen?' Want ik zag hier inderdaad in die — naja in het model zelf zag ik de dingen die erin moeten. Je hebt onder andere de andere IBAN en naja daar was ik een beetje op vastgelopen.

Q *Ja, je had dus wel de relations gevonden? Die had daar toegevoegd, —*

S Die had ik hier toegevoegd inderdaad.

Q *En dan kreeg je waarschijnlijk een error dat de kolom niet in de database zat of iets dergelijks -*

S Ja dat inderdaad.

Q *Ja want je moet dan een migration nog runnen. Dus dat klopt wel.*

S Toen dacht ik van naja dan moet ik dus ergens de ID eruit vinden waarschijnlijk en die staat dan daar ergens maar die heb ik nog gevonden. Dus toen toen dacht ik van 'nou oke dan kan ik de tool wel gebruiken'. En toen ging ik naar [redacted], en daar ben ik inderdaad gaan kijken van 'nouja oke waar maakt ie dan gebruik van'. Dat zag ik hier wel. Ik dacht misschien hier ApiClient, dat je daar misschien iets kan inputten of outputten. Daar was ik uiteindelijk mee bezig.

Q *Ah oke, daar was je uiteindelijk geëindigd.*

S Ja ik was eigenlijk — want eigenlijk gebruik je als een soort van zoeken naar de backend voor het probleem van je database of van je uiteindelijke ding. Nou dat kwam ook eigenlijk wel bijna overeen met de search everywhere voor sommige dingen.

Q *Ja, dus daarmee kwam je er ook al?*

S Ja daar kwam ik op die paar dingetjes uit, zeg maar, misschien is het ook wel een beetje gewenning hoor dat ik daar gelijk.

Q *Ja dat kan.*

S Ik had zoiets van 'ohja moet wel..' —

Q *Dat was ook een reden voor jou om niet de tool eerst te gebruiken om te kijken van waar zit het?*

S Ik denk gewenning inderdaad. Ja dat.

Q *En heb je hier uiteindelijk nog verschillende dingen geprobeerd om wel te vinden wat je moest vinden want je zegt ik heb naar die ApiClient gekeken, had je eerst nog verder gezocht in de tool of gewoon die staat bovenaan die kijk ik eerst, zat daar nog een idee achter?*

S Naja, de benaming. Natuurlijk. Ik ben daarbij nog verder gaan kijken inderdaad voor andere dingen maar toen keek ik naar de functies en kon ik het ook al niet direct eruit halen zeg maar. Het was benaming waar ik vooral naar keek. Wat heb ik verder nog in de tool gedaan? Ja ik vond het wel nice dat je dus snel een overzicht hebt van wat iemand heeft gedaan zeg maar. Dus dat is wel, ja.. dat kan wel handig uitpakken.

Q *Oke, helder. Ik zou willen voorstellen dat je naar die design opdracht gaat kijken. Het liefst probeer een beetje de tool te gebruiken. Misschien zeg je van 'ja het werkt gewoon echt niet voor mij', dan ben ik erg benieuwd natuurlijk waarom het niet voor je werkt. Maar probeer er even mee of het lukt. En dan ben ik erg benieuwd hoe compleet het overzicht gaat zijn dat je mij geeft voor AliPay.*

-

Q *Oke, brand los.*

S Nou, mijn eerste idee was zo van wat is minimale in en uit want je hebt natuurlijk services waarmee je communiceert. Dus ik ben een beetje gaan nadenken naja dit lijstje [assignment] van wat is daar nou een view van?

Q *Ja.*

S Toen ben ik inderdaad hier nog gaan kijken naar — naja het is "[redacted]", dan kan je dus ook in die views kijken. Dit gaf al even wat minder —

Q *Ja hij is nog niet activated [filter menu bug]*

S Ah ja.

Q *En welke informatie heb je hieruit gebruikt?*

S Uiteindelijk eigenlijk niet heel veel. Maar ik heb wel inderdaad gekeken van wat — maar hoe zit het dan met — ik kom zelf eigenlijk op vier hoe heet het.. dingen. Zeg je specifiek van je moet eerst die UUID nemen, dan moet daarna nog de QR code er los van halen. Naja op zich al zou je dat in een keer doen dat ie dan al gelijk een QR code kan doen dan kan je dat natuurlijk in een call doen.

Q *Ja*

S Ja, dus dat was het enige waar ik aan het twijfelen was. Dan dacht ik inderdaad het tweede ding moet zijn dat je de status moet zeggen of je betaald hebt of niet. [...] heb ik het nu maar even genoemd. Het derde is inderdaad van het pollen van dat ding om de vijf seconden. Moet je inderdaad gewoon weten welke status ze zijn. Naja dan kwam ik dus inderdaad op de [redacted] object, daar zag je dus je hebt een transaction model die dat doet.

Q *Dat heb je dus in de tool gevonden?*

S Ja ik denk vooral een beetje inspiratie ofzo van oja er is inderdaad een soort van handler die dat doet en voor de rest dan welke dingen je zou moeten hebben, dat kon ik verder zelf ook wel bedenken van nja er moet dus een status hebben dus die — zo'n ID en een QR code. Op het moment dat je die allereerste [call] doet dan zeg je gewoon 'ik wil hem maken' [transaction model] dan create 'ie uiteindelijk die UUID zelf en die QR. Als we er van uitgaan van deze los [zie boven] dan creëert ie eerst die UUID los en dan kan je daarna nog inderdaad zeggen van oke ik stuur deze UUID naar de server, met de benodigde authenticatie misschien, waardoor hij die QR code dus gaat aanmaken, die stored ie dan lokaal op de server zelf en die returned ie dan ook nog. Hij zou daarna nog van — als je daar de goede status van wilt updaten dan zou je dus een soort van call moeten hebben op de server waarin je aan de hand van een QR code en een bepaalde token die eraan mee is gegeven kan de bijbehorende [redacted] kan halen of updaten. Op het moment dat je die geüpdatet hebt en je hebt ergens in je view een functie die dus de status kan returnen van een bepaalde [redacted] dan kan je dus ook checken van of ie betaald is of niet.

Q *Oke.*

S Moet je meer de QR codes hebben of niet, maar, ik ga er van uit dat eentje wel goed is.

Q *Ja eentje volstaat.*

S Maar waar ik vooral de tool een beetje voor heb gebruikt is denk ik vooral een beetje inspiratie ofzo, maar niet perse —

Q *En wat heb je dan gebruikt als inspiratie?*

S Ik denk eigenlijk alleen maar het controller hoe-heet-het de benamingen misschien. Dan heb je dat sowieso overeenkomt met de andere componenten. Ik vond het wel moeilijk om de tool te gebruiken zonder dat je — want je gaat heel snel zelf dat je denkt van —

Q *Want je voelde je meer geneigd om gewoon naar de code te kijken of..?*

S Naja, omdat dit een soort van schets moest zijn heb ik niet per definitie heel erg naar de code gekeken.

Q *Dus je had het niet nodig om te kijken hoe* [redacted] *in elkaar zat om deze op te kunnen zetten?*

S Vooral omdat dit een — naja omdat het wat high-level is dat je niet helemaal — Zeg maar uiteindelijk als je het echt gaat implementeren dan zal je er wel wat meer in —

Q *Want wat voor informatie zou je er dan — of wat voor,.. — Zou je dan informatie uit de tool kunnen halen die je kan helpen bij de implementatie?*

S Als je gaat implementeren?

Q *Ja. Als je net een level dieper gaat.*

S Ja, dan wel denk ik. Ik denk dat je dan heel veel dingen gaat overnemen die er al zijn. Omdat je al weet 'hier is een view die al inderdaad die vergelijkbaar materiaal heeft' en daar kan je inderdaad naar kijken van naja welke files zou je dan nog meer naar moeten kijken.

Q *Dus wat hiervan zou je dan bijvoorbeeld kunnen overnemen voor AliPay? Welke zijn nodig? Weet je dat nu een beetje aan te wijzen of?*

S Die Request views die niet zeg maar, want dat zijn volgens mij gewoon dingen die er naast gebeuren maar je hebt hier in dat [redacted] Transaction view naja dat zijn gewoon de endpoints die zo'n transactie pakken. Bij die models — bij de controller kwam ik die QR dingen tegen. Ja hier.

Q *Ja, de WorkflowTokenQrRequest, ja.*

S Nou waarschijnlijk kan ik dat perfect overnemen want er wordt al een QR gemaakt die gebruik je daar ook voor. Misschien omdat het een workflow is kun je misschien zelfs nog de workflow aanroepen. Maar dat ligt natuurlijk aan de structuur daarvan. Ja en verder in die transacties — want je maakt hier geen transactie eigenlijk, je geeft alleen maar informatie door. Als ik het goed heb.

Q *het is wel bedoeld als echt een payment method zeg maar. Gelijkwaardig aan iDeal* [redacted] *. In that sense maak je wel echt een transactie.*

S Als je ervan uit gaat dat het naar een bunq rekening gaat dan — of

Q *Dan is het officieel nog steeds een transactie. Alleen dan is het een interne transactie. Nee oke, klinkt goed. Dan heb ik nog wat vragen. Hoe vond je de assignment zelf?*

S Muah. Was wel op zich duidelijk denk ik.

Q *Ook leuk of..? Mag je gewoon eerlijk in zijn hoor, ik moet er nog meer mensen mee 'vervelen'.*

S Eh, ja wat vond ik er van.. Ja ik vond het wel lastig inderdaad om het goede gebruik voor de tool binnen dit te doen. Maar misschien komt dat ook al inderdaad doordat je dan al misschien al wat dingen van de backend al soort van — klein beetje weet. Dat je daar de manier van opzoeken daarmee gewend bent. Dat zou kunnen.

Q *Je bent al te veel geïnfecteerd?*

S Ja, misschien wel.

Q *Cool, maakt niet uit. Normaal —*

S Voor vreemde problemen kan het wel inderdaad —

Q *Zoals? Kun je daar een voorbeeld van geven?*

S Als je compleet nieuw bent misschien. Met backend ofzo.

Q *En wat zou er voor die mensen dan met name in de tool zitten volgens jou?*

S Weten waarnaar je moet zoeken. Eigenlijk. De relatie tussen de — . Oh en daar zat ik ook nog wel over na te denken inderdaad, want ik was een soort van relaties van uit mezelf aan het tekenen. Maar, je kunt wel bijvoorbeeld in heel veel andere text editors kan je dus bijvoorbeeld zo'n UML laten genereren. Dat soort dingen.

Q *Had je hier gezien dat je de workflows aan kon klikken?*

S De workflows? Nee had ik niet gezien. Maar dat is wel nice.

Q *Met die wetenschap, is dat nuttig, of. . . ?*

S Ja.

Q *Een leuk voorbeeld denk ik is als je Transaction Update uit mn hoofd — deze, als je die aantikt. Daar zie je als het goed is ook de branches [in uml] zo dan zie je echt de tool aan het werk.*

S Ja dit is een mooie visualisatie wel.

Q *Dit zou jou kunnen helpen, of..?*

S Ja, ik denk dit wel. Ik denk om code sneller te kunnen begrijpen misschien.

Q *Vergeleken met?*

S Vergeleken met deze file openen zeg maar. Zo'n workflow.

Q *De definitie?*

S Dit maakt het net iets, ik denk dat dit net iets sneller is om een globaal idee te hebben van —

Q *Oke, nou ga ik normaal stap voor stap door de tool heen, want er zitten eigenlijk drie main features in. Als je hierboven kijkt, je hebt search by label, developer en CIT reference. Heb je de search by label - die heb je gebruikt zo te zien?*

S Ja.

Q *Wat was je doel van het zoeken daar? Wat verwachtte je te vinden qua informatie?*

S Ik verwacht te vinden welke componenten relevant waren voor het probleem.

Q *En heb je die kunnen vinden?*

S Naja, voor die derde, hoe heet het, ja ik heb ze wel kunnen vinden. Dat wel denk ik. Alleen voor die tweede dus niet met die User hoe ik —

Q *Ja. Dat ontbrak dan nog? Dat was niet duidelijk genoeg hoe je naar die User kon?*

S Ja, maar misschien had ik gewoon anders moeten kijken, dat zou ook kunnen.

Q *Ja ja, dat kan, maar misschien dat het dan niet duidelijk is weergegeven in de tool. Dus misschien dat er iets ontbreekt voor jou. Of iets dan anders weergegeven had moeten worden om het duidelijker te maken?*

S Ja, ik weet niet. Even terugdenken. Ja het wordt natuurlijk ook niet, want die relaties zitten hier ook niet —. Misschien inderdaad relaties ook voor models. Omdat Models natuurlijk ook andere models bevatten, dat je daar misschien ook iets als je er op klikt, dat je dan, —

Q *Dat je zo'n soortgelijk diagram —*

S Nah, dat je in feite gewoon de definition file krijgt.

Q *Ah zo.*

S Dit, maar dan op een grafische manier zeg maar. Ik denk dat dat misschien nog wel heel nice zou zijn.

Q *Ja oke, fair. Had je verder nog iets gebruikt aan informatie of dat je zegt van dit had hier nog toegevoegd kunnen worden als je zoekt op een label?*

S Ja alleen dat je misschien alleen op — ja je zoekt natuurlijk op hardgecode label inderdaad, dus dan wil je iets — vaak ben je meer kernwoorden aan het intikken zeg maar. Dus dat is — vaak, weet je als je google'd zeg maar naar een probleem; dan ga je inderdaad gewoon wat er in je opkomt ga je dat proberen in te tikken,..

Q *Natural language.*

S Ja, en dan krijg je uiteindelijk Google die zegt 'bedoel je dit' en nu moet je natuurlijk zoeken naar Label dus dat is misschien ook weer een andere manier van —

Q *Ja ik heb hier echt de labels zoals ze in GitLab staan gebruikt. Die komen ook rechtstreeks vanaf de issues. Die labels die worden gewoon hergebruikt.*

S Ja op zich vult ie hem wel aan natuurlijk. Kijk want [..demonstrates] en dan heb je hem al zeg maar. Als je weet '[redacted]'...

Q *Maar het zou inderdaad nice zijn als je ook een soort suggesties krijgt van 'bedoelde je dit of...'.*

S Misschien file search ofzo.

Q *Ah ja, en waar zou je dat voor gebruiken dan, file search?*

S Ja, ook relevantie vinden.

Q *Relevantie in welk opzicht?*

S Of er stukjes code zijn die hetzelfde al hebben gedaan als het probleem waar jij nu tegenaan loopt. Bijvoorbeeld, kijk, ik moest iets halen uit die user en dat kon ik bijvoorbeeld kijken waar ie allemaal gebruikt werd of zo.

Q *Dat je een soort voorbeeld had om vanaf te werken?*

S Ja, dat je snel voorbeelden kon vinden voor het resultaat, voor het probleem bedoel ik.

Q *Yes, I see. Had je de find by developer gebruikt?*

S Nee, ik het begin eventjes gekeken van oh dit is wel grappig. Ik zag dat ik er nog niet in stond.

Q *Nee klopt.*

S [name of former developer] is die niet —

Q *Die is ook van de TU Delft geweest.*

S Ja, dat zeg maar. Die staat er wel al in zeg maar.

Q *Die hebben code gemerged in develop, ik kan je straks wel uitleggen hoe het werkt. Maar als het gemerged is je code dan kom je in principe hier ook in te staan automatisch. Maar je hebt deze niet echt gebruikt?*

S Nou vooral voor dit probleem dacht ik 'is niet echt relevant wie die het spul heeft gedaan'. Je kan natuurlijk wel al — het is natuurlijk wel zo als je ziet van ;'he, die persoon heeft heel veel op deze label gedaan' of zo, en ik wist dat eerst niet dan zie je dat wel een stuk sneller via die tool. Of wie er allemaal nog meer aan gewerkt heeft. Ik denk dat dat wel — ik denk dat het wel heel nuttig is maar nu heb je maar een uurtje dus dan... En in RocketChat gebruiken om dan die developers te ...

Q *Ja precies. Alleen in welk opzicht zou het dan verder nuttig zijn om dat te kunnen zien?*

S Naja, bijvoorbeeld ik ging dan — stel ik zou dit bijvoorbeeld echt moeten doen dan zou ik inderdaad gaan zoeken naar [redacted] en dan zie ik jou naam eronder staan omdat jij daar heel veel dingen in hebt gedaan en dan weet ik wel gelijk inderdaad van 'oh ja dan kan ik die aanpoken voor vragen erover'.

Q *yes, maar dan zoek je eigenlijk op label* [redacted] *right?*

S Eigenlijk wel ja. Ja, dat is die bij label natuurlijk.

Q *Maar het zoeken echt op developer zie je daar nog nut voor?*

S Naja waar iemand mee bezig is. Of ie geschikt is voor — als je een bepaalde taak hebt die je wilt overdragen aan iemand.

Q *Ah ja.*

S Dat je dan kan kijken van 'heeft ie dat soort dingetjes ook al eerder gedaan'?

Q *Ja, dat is wel een goeie, had ik nog niet aan gedacht. Oke, clear. De find by CIT had je die gebruikt?*

S Nee, want volgens mij ga je daar bij — ja je zoekt hier op getallen, —

Q *Ja, je geeft echt een issue reference —*

S Ja, dus je moet echt heel specifiek moet je zeg maar zoeken inderdaad. Want in CIT worden daar ook problemen — ja daar worden ook gewoon problemen neergezet waar ze tegen aan lopen toch.

Q *Ja, de bugs repository daar staan alle bugs in die users rapporteren bijvoorbeeld of die we zelf tegenkomen met testen. En je ziet hier ook deels —*

S Ja, maar ook de problemen waar developers tegenaan lopen zeg maar?

Q *Wat voor problemen?*

S Nouja bijvoorbeeld iemand heeft een looptijd probleem vastgesteld en die vraagt andere mensen bijvoorbeeld om ook naar die branch te kijken.

Q *Dat gebeurt niet nee. Dat is gewoon via RocketChat, nee CIT is echt documentatie van bugs in de app, nieuwe producten die we nog gaan ontwikkelen, enzovoorts. Wat je hier nog wel had in de tool op zich, want je zegt van moet je inderdaad specifiek weten, dat klopt inderdaad wel. Maar dat je hier een overzicht hebt van de issues, je hebt dan hier nu bijvoorbeeld* [redacted] [redacted] *aanstaan zie je hier dus alle issues, bugs die bij deze feature horen. En hiervandaan kan je dan doorzoeken op zeg maar. En dan zie je nu dus van deze issue heeft deze bestanden aangeraakt, ja dat een beetje. Weet niet of je nu je dit zo ziet heb je dan nog zoiets van dit is wel nuttig of dit zou ik eigenlijk nooit gebruiken?*

S Tis natuurlijk zo een beetje van als je dus — dat het bugs zijn die al een soort van opgelost zijn als ze daar een oplossing voor hebben dan — voor die specifieke bug dan is het lastig om — misschien als recent iets als er een grote verandering is geweest ofzo. Dat je dan kan kijken van welke bestanden heeft ie aangepast. Want ik begin nu opeens problemen te krijgen ofzo. Dat je een beetje daarin kan kijken. Of een heel snel grafisch overzicht wilt hebben van een probleem waarvan je hebt gehoord dat het opgelost is. In plaats van dat je de CIT doorheen hoeft te spitten. Als het misschien een hele grote issue is. Maar ik zou het wel lastig vinden inderdaad voor directe development doeleinden hier.

Q *Ja dat kan. Waren er voordelen in het gebruiken van de tool volgens jou? If so welke?*

S Ja, overzicht denk ik.

Q *Overzicht in welke sense?*

S Overzicht van kleine stukjes code die je kunt opzoeken of naja overzicht van over bepaalde modellen of views die met een process zijn betrokken of zien wat iemand al heeft gedaan. Dat kan misschien tijd winnen. En die controller diagrammetjes die vind ik wel nice.

Q *Zat het eigenlijk in die eerste assignment, ik weet niet eigenlijk.*

S Ja, misschien wel maar

Q *[searching] nee ja, het staat er niet expliciet in inderdaad. Ik wel zo mooi links een help dingetje getypt maar —*

S Ja, daar kijk je niet naar natuurlijk.

Q *Ja daar kijkt niemand naar. Nou, dat is niet helemaal waar, die meiden hebben dus wel allemaal het gelezen. Maar hun feedback was ook veelal van 'oh het ziet er mooi uit', oke en hoe werkt het? Zijn er nog dingen dat je zegt van 'deze informatie zou ik toevoegen aan de tool / dit mis ik nu / dit werkt niet'?*

S Wat zei ik eerder nou? Oh, die relatie dingen. Met de modellen.

Q *Ah ja. Noted.*

S Ja, en misschien zoeken op file. Ik weet niet of dat nog, omdat ik daar wel mee bezig was bedoel ik. Dus op zich.

Q *Oke, helder.*

S In feite zou je tool alles een beetje moeten kunnen doen wat je anders normaal met shift-shift doet of is dat ook weer niet ..?

Q *Ja en nee, er zit overlap in zeg maar. Het is niet bedoeld als een vervanging van de search.*

S Ja je hebt natuurlijk niet de code.

Q *Nee klopt. Had je in general nog feedback op de tool dat je nog niet hebt genoemd?*

S Nee. Niet echt denk ik.

Q *Je hebt nou de backend zo gezien, wat denk je daarvan? Wat is je impression als je die backend,.. je werkt er nu al een tijdje mee, —*

S Veel folders. Ja ik vind het wel heel logisch. De backend. In het begin waren we natuurlijk echt even aan het inkomen van hoe zit het nou in elkaar ofzo en uiteindelijk toen was het toen we die flow wel begrepen toen dan gaat alles ook vanzelf soort van.

Q *Dus het deel dat het logisch maakte is dan, hoe zeg je dat..*

S Nja, soort van het framework dat dan, gewoon soort van het PHP framework dat dan opgebouwd is. Wat er allemaal in zit. Maar gewoon het Model View Controller aspect en bijbehorende met de strikte coding guidelines zeg maar.

Q *Oke, dan hebben we deze tool gezien. Nu hebben we vaak dat juniors hier die vragen de seniors om raad van 'joh, hoe doe ik dit' weet ik veel wat. Heb je het idee dat deze tool dat zou kunnen vervangen slash deels opvangen? Of zeg je van 'nee, dit gaat niet een senior verlossen' soort van?*

S Het vervangt denk ik de vraag, als de senior als antwoord geeft 'hey je moet in die file kijken' bijvoorbeeld voor dat probleem, dat zou deze tool dan eigenlijk ook kunnen doen. Of je moet op deze locaties kijken. Of deze views.

Q *Dus dan geeft de senior aan van hier moet je kijken en dan ga jij hier verder bedoel je?*

S Dan ga je naja als junior ga je dat dan inderdaad inkijken. In plaats van dat je dan nog die vraag hoeft te — gesteld hoeft te hebben zeg maar. Maar ik denk in dat opzicht echt puur — als je dus echt uitleg nodig hebt of zo dan is het wel lastig te vervangen natuurlijk.

Q *Heb je voorbeelden van informatie die echt alleen een senior zou kunnen geven?*

S Eigenlijk heel veel natuurlijk. Algemene coding tips.

Q *Coding tip als in style of in structure meer?*

S Beide denk ik. Stijl wordt hier niet door opgepakt.

Q *Nee ja klopt.*

S Naja de senior kan natuurlijk niet razendsnel tienduizenden files opzoeken en indexen. Dat is wat de senior niet kan doen. Maar ik denk dat het verduidelijken zeg maar dat dat kan natuurlijk ook door een senior gebeuren, met dat ie als ie de informatie wel weet zeg maar door het uit te schetsen ofzo. Maar dat kost natuurlijk tijd van de senior, dus — En zoals andersom, diagram maken dat kost, ja dat kost tijd.

Q *Ja dat is absoluut waar. Je bent nu met deze backend aan de slag gegaan. Daar kwam je helemaal nieuw in natuurlijk. Je wist niet wat voor backend het was. Wat is dan voor jou de meest belangrijke informatiebronnen? Als je zo'n nieuw framework moet leren?*

S Documentatie natuurlijk, maar ja, op welke frameworks het gebaseerd is misschien. Dat je het kan koppelen naar iets dat je kent.

Q *Ah, dus stel het zou gebaseerd zijn op Laravel dat je dan in de Laravel documentatie wat dingen kan terugvinden ofzo bedoel je dan?*

S Ja of dat je daar ervaring mee hebt ofzo. Dat je dan zelf gaat nadenken van het lijkt hierop of zo.

Q *Kon je dat hier toepassen? Op deze backend?*

S Model view controller aspect, het algemene deel met een API die je programmeert. Zeg maar dat zijn natuurlijk structuren die waarin je kleine dingen wel terugziet. Maar volgens mij is dat heel erg algemeen overal wel zo zeg maar. Ja ik heb niet zo heel veel ervaring met heel veel verschillende frameworks zeg maar dus daarom is het ook lastig om dat te kunnen zeggen. Maar misschien dat er andere PHP frameworks zijn die wel wat meer erop lijken. Maar dat weet ik niet.

Q *Ik weet honestly ook niet of er een ander framework als basis is gebruikt hoor, dat durf ik echt niet te zeggen. Volgens mij zijn ze echt gewoon gaan bouwen wat zij prettig vonden en dat is uitgegroeid tot dit.*

S Volgens mij is dat, dat klinkt wel logisch ja.

Q *Dat zou me ook niet verbazen met hoe ze hier werken.*

S Dat het [a developers]'s visie was tot een perfecte framework.

Q *Ja exactly. Nee ja oke, helder. Dat waren mijn vragen, had jij nog vragen aan mij?*

S Nee, niet echt.

C.10 Subject 10

Questioner (Q) *Cool, so what have you done?*

Subject (S) So, in the JSON file I've added a new field called 'username'.

Q *Which file — this is* [redacted]*TransactionView, how did you find this file?*

S So, yes, so first I have I went to the localhost:3000 [tool] and then I searched for the feature '[redacted]', which is this one, and in here I went to the views in the views tab. And since the endpoint was called [redacted]-transaction I went to the

[redacted]_transaction_view, and then I see that there are two files, the pre-processor and the post processor. So you want us to change the field in the response, so I went in to the — well, firstly I went into the folder, in the PhpStorm, and then in here I see that I can change, in the response, this file. Which is ViewPostProcessor. And so in here I can see that there's a method called getResponseObjectBody. Which is, I guess, the method that gives the response. So I guess that it has like a lot of fields, in this method, and a field `user_name`, which is a constants which I created in the post processor base.

Q *Yes.*

S Like the other ones. Then I get the username, I guess this is right, the owner user name only for ACL. I don't know what exactly this means.

Q *The ACL is actually the Access Control Layer, so basically every model in our backend has an owner user assigned and if you're not the owner of an object you need special permissions before you can access whatever is in it. So basically now it says 'give me the user that owns this model', it's fine for now, no worries, because you don't have the entire context, but typically this is used to determine whether you have access to this model. But I'll explain afterwards anyway. Okay, so you added that field here, yes.*

S Yes, and I also added the username field in the response of the JSON file.

Q *How did you realise or think of adding the field here [in the JSON]?*

S So, from the documentation that you have give me [assignment text] you explain —where is it— that every model has a definition inside of the JSON file. And then I guess that by defining the field here, well I first need to compile this JSON, [clicks generate, error pops up, some mumbling].

Q *Yeah, you can ignore this. It's probably some small thingy. Oh, there's a comma. Now I see. And now we get, — Yeah, just let it be. Like the idea is right so. You should add it here and when you compile it you also don't need to change the base class by the way. You need the constant there, but that's taken care of by the generation.*

S Yeah, exactly. And then it's generated I guess in the post processor.

Q *Yes, exactly.*

S Yeah, that's basically it. So, adding the field in the JSON file in the response attribute, and then adding well like here.

Q *Okay, yeah. That's basically it. Well done. I would suggest that you go on with the next one then. So that's basically seeing what you need to do to build the AliPay alternative to* [redacted].

-

Q *Okay, so what did you find?*

S Okay so basically I went through that we need to go through to be able to get the flow of the user. So, for example, so firstly you mentioned that this AliPay payment method is very similar to the [redacted] iDeal ones. So I went to the tool and I in the same feature that I was in, which was [redacted], I see that there are different folders created to be able to this [redacted] implementation. So, from this one I went into the first step of the flow which is, well makes a call to initiate the AliPay Transaction.

Q *Yeah.*

S And yeah, okay — that's the only one I haven't done. Let's, I guess, for that one you need to create a new endpoint into the TransactionCreate. Inside the `core_alipay_transaction_controller`, for example. You will need to process that into .php file and create the JSON file to be able to get the request and response template.

Q *Yeah.*

S And then, for example, the next thing you want to do is to get the QR code based on the UUID. So for that you need to create, I guess, a new file called WorkflowTokenQrRequestAliPayCreate, which will generate the QR code based on the input of the request. And that will be set into the AliPayCreate JSON file. Inside that folder. The next one will be to add the user to the transaction that has been created. For that, I guess, you'll have to create a PHP file in the `core_alipay_transaction_controller`, and create a new PHP file called WorkflowAliPayTransactionUpdate, in which you update the transaction with a username that you have in the request. And that you added in the JSON file. The username of the user. And then to confirm the payment the user needs to make a request to the ProcessUpdate PHP file, which is again the same folder, `core_alipay_transaction_controller`, and the template for the request and response is done in the same folder with called for example WorkflowAliPayTransactionProcessUpdate.json. And finally to be able for the webshop to check this transaction status you need to make another endpoint, for example called WorkflowAliPayTransactionStatus where you tell the webshop what's the status of the transaction. If it's good, if it's not. And also make a JSON file to be able to know the request and the response objects you want to receive.

Q *Okay, sounds good.*

S So, that's it. Or did I messed up some things?

Q *No no no. Like, there are some things that should've been different, of course, but given that you have zero backend experience I think this is a really good approach already. You will find that during implementation that some things are positioned somewhat different. But the main idea you got pretty well. So that's really good.*

S I mean, the whole thing, I wouldn't be able to have done this so quickly without the tool because if you want to do this in PhpStorm you have no way to like "what's the files that needs to be done for [redacted]", there's no tool to be able to do that. So, using this search tool I have basically all the files that I need to know what should I do to be able to replicate this in the system.

Q *Yeah. Have you also checked out for models that you need?*

S No, but I guess that it will be pretty much the same, but I haven't done that. Sorry.

Q *No, no worries.*

S But yeah for example in the models step I guess I will have to create an implementation of the MessageInModel, MessageOutModel of the AliPay. And then again a TransactionModel for the transaction.

Q *Yeah, that's basically it indeed. Yeah there are some small — I can explain in a bit, but I think you covered pretty much it. Okay, cool. So, how did you find the way to start?*

S So the first thing that I did was I searched in the tool like [redacted]. Firstly I searched "Feature: " —

Q *You can do partial search by the way, so if you just start writing [redacted] then it will also work.*

S Ah nice.

Q *Saves you some typing.*

S I see that there is [redacted], what we would like to do is to make a payment so I went ahead and searched for that. That gives me the results for the files that I want to — that was touched when they created the [redacted]. So I went to the controllers tab and see that there are different payments, so for example in the `core_token_qr_request_controller` there are different files. And some of them are called [redacted] iDeal and then one that doesn't have any name. So I guess that is, for example, a base controller for the other ones. I'm not sure, but that's my guess. Then basically based on the naming of this file and also if you can for example check the contents of the file in this beautiful graph, you can see that you basically have the logic of what you want to do. For example if you want to create a transaction you simply want to make all of this same steps but with the AliPay transaction payment. Yeah, so basically create the same files and the same structure but based on the [redacted], because — I guess it's pretty much the same.

Q *Yeah that was the idea, so that you dive in into the [redacted] thing and see what's all related to [redacted], so what do I need to actually build a similar system. Have you also looked into PhpStorm, like you have filelist here, have you checked some files the source code to see what it actually does or..?*

S Well, I wanted — for example this one.. —

Q *This is the TransactionCreate workflow, yeah.*

S In the `core_request_inquiry_controller`, I guess this is also used to be able to implement it. But since I didn't have any knowledge of the backend or what this really does I wasn't really expecting to change anything here.

Q *Yeah, I see.*

S But sure, probably, I don't know, maybe. I will have to take a — look more into this.

Q *I see, yeah, we do use like — when we have an iDeal transaction or* [redacted] *what happens behind the scenes is that we initiate a Payment Request. As if you request money from a friend basically, you now have a request from a company to pay that company. So that's what happens, that's why you have the RequestInquiry and RequestResponse things in the list, because we have the RequestInquiry from your account to the company and you confirm that with your response basically.*

S Yeah, makes sense.

Q *So that's what happens and as soon as that response is set to Accepted, we know 'Okay, the user accepted, so the transaction succeeded'. And then we can communicate that back to iDeal or whatever. So that's basically where this part comes in. But that indeed is like context knowledge. So that's basically out of scope. Okay. Cool. Then I have some questions that I want to ask you. Did you like the assignment?*

S It was really nice. Yeah.

Q *Yeah? What did you like about it if I may ask?*

S Well basically I haven't used like the developers tab or the issues so much since this task is basically centered in the files, I guess. You could probably use the other ones, but I think the most useful one is the default one. I like that they are ordered based on their folders. That's not something you come up with very often in similar tools. Or in this case in the IDE environment tools. I haven't looked into the other tabs, but I guess it is pretty much the same, like the only — I guess dependency is that you depend on the structure of the project. So if you don't have a really good structure then probably you will be a bit misleading. I mean, you will still get the files and you still get the names of the files but if the structure of your project is not very good you probably waste a bit of time to handle like that layer —

Q *Yeah okay.*

S Where are these files, where should I create this files? But that's basically part of the project, it's not part of the tool.

Q *Yeah, the tool is really specific for bunq, I should say. Because bunq has a really structured codebase and we have the structured commit messages, because the tool runs entirely on the git history. That's all I use as a data source, well yeah and GitLab for enriching the data but — So if it weren't this structured this tool would not have existed. So it makes sense that the structure —*

S I mean, it would still work right?

Q *Yeah, well, depends as you said on the structure of the project then. And then you need to build it on the structure of that particular project of course. Now it — like for the workflow thing I really map whether the folder ends with the controller thing or whether there's a definition workflow folder, stuff like that you can't do in every project. If you were to run this on a random project then it would crash tremendously. Okay, cool. So you said you have used the Find By Label feature. For what purpose was it or what was the goal?*

S To be able to find the [redacted] related files. Yeah, basically that.

Q *Did you expect to see more information? Or like, do you say like is there any information that doesn't need to be there or..?*

S I don't know, maybe if it's possible, maybe add the PHP documentation into this tool. Will it be possible?

Q *What documentation do you mean?*

S For example the little comments in each method, here — I don't know, maybe, sometimes there are useful comments.

Q *Well, the backend itself doesn't have comments. Like, we are not allowed to write any comments. At all. That's a policy. So our code should be explaining itself which is okay but — only for really complex stuff we are allowed to put comments basically. So, yeah, I see what you mean, but we don't have comments so,.. —*

S Yeah if you build that feature it will be a bit pointless..

Q *Yes, exactly. So but, yeah, I see your point. So that you have a bit more of a feeling what it does I think?*

S Yeah. Like, I mean, the same thing goes here, if you write a bit of code with beautiful methods, self-explanatory then okay great. But if you write poor code then maybe you wouldn't be able to have so much information.

Q *Yes exactly. Cause these graphs, without knowing what's going on, are these understandable? Like, the steps?*

S Yes, for sure.

Q *Okay, cool. Did these graphs also help you to understand what you were supposed to do — understand?*

S Well, I based my steps into the naming of these nodes. I wasn't exactly [..] flow of the method. Since they're pretty much very simple. For example, they create one linear graph. So there's no much difficulty in that. But I can see where there are cases where you will be able to use — it will be useful to have this graph condition thing. For example when you have methods more complicated. But since this case is — doesn't have any overcomplicated methods, I didn't have the needs to use it.

Q *Yeah, I see, okay cool. You mentioned that you didn't need to use the Developer and CIT searches. Do you have a reason for that maybe, that you didn't need it?*

S Well my first step was just to look for [redacted]. And the first thing it came up with was the files. I think that was basically what I was looking for. But if I didn't have the chance or if I didn't find anything useful in the files, I guess I will try the feature by developer. See what developer has worked on this, the [redacted] iDeal payments. And see how they connect with the — what files they have touched to be able to implement that. In case the CIT — I will probably be able to search for the [redacted] issue. And I guess there I will be able to also to find very useful stuff. But since the files was the first thing that came up, I was glad to use it.

Q *Cool. Can you think of a use-case where it would be useful to search for developer or CIT.*

S For developer maybe if you have a developer that is doing a lot of random stuff. Maybe it will be useful to keep track of him, like he's doing right now or what he has done. Because if he is not centering one thing then maybe it's very useful to have a history of what the user has done. Because if you do one thing all the day then you're not going to have much more information. Like, if a guy is doing [redacted] in iOS then he's doing that — you don't need to that's he's doing another thing because he's doing that. [?]

Q *I think I get your point.*

S And then for the CIT then I guess you will have to — you will need to have information about the issues that you have in the CIT group. But, besides that maybe for example for the bugs, I guess, well, for anything related, but for the bugs it will be very useful. Because in the bugs you need extra documentation to be able to know how to reproduce these bugs. And in that case it will be very useful I think.

Q *Okay. Are there any specific parts of the information that's then extra useful to you or..?*

S Extra useful like?

Q *Yeah, like, the part that's really useful to you, like —*

S Well the files was the main thing that I used. Something that, for example, the related tags I'm not sure really how does it work?

Q *Ah yeah there's a really small info icon but apparently I should increase it or something. Well, it doesn't explain too well what it does actually. I wrote somewhere what it means, so basically when you have a feature then it touches files. But another feature might also have an overlap with that because there's some generic base in your code. Say for instance this feature uses the Requests, I know bunq.me also uses Requests. So then they use the same code basically. So this basically ranks others features in terms of how many overlap there is in terms of how many files that they touch.*

S Ah okay.

Q *So, apparently the Signup flow uses a lot of the* [redacted] *stuff. Which makes sense because in the Signup flow you can confirm your identity by making a* [redacted]*. Deposit in this case. But there's still overlap. So that's what that feature can do.*

S Yeah, makes sense. I mean there's a lot of features that I haven't really used. But yeah, I'm not really sure which stands out more.

Q *Which one is more..? Sorry?*

S Which one stands out more. I mean, I have used the files and the issues. But not as much the other ones, because of this task probably. It's centering using that instead of the other ones.

Q *Yeah, okay, cool. What are the main benefits according to you, if any, of using this tool?*

S Well I think all of these features are beneficial. Because you don't have these kinds of tools in your environment usually. It's true that for example in GitLab you can search the author, and get his commit history, but you don't get for example — to get a whole list in the same view. Like the files and authors that contributed to this exact feature. And yeah, I mean, the whole thing, like implementing all the features just together. That in itself is a great advantage.

Q *And in what sense is that an advantage?*

S Well, firstly because you don't have to move to another application. You have to have thousands of applications open to do different things. Or depends on — well I guess this could be called a third-party application — but if you're developing yourself you can change it to whatever you want.

Q *Ah, customisability?*

S Yeah, basically.

Q *Okay, cool. Are there any other points that could've been—should've been different, things that you missed in the tool, what could've helped you or anything?*

S I feel like this tool could have a lot more features, but in the bright side, like it's so useful that you could be able to add a lot more features that will make it more useful.

Q *So there's room for extension?*

S Yeah. But this in itself is very useful. I mean, it's not that this is not useful, this is very useful, but the area in which this is useful it could be expanded more.

Q *Yes, exactly. So would you have examples of where it could expand into?*

S I don't know. Maybe try to — get some ideas from another — because I know from another, for example there are some tools in GitLab that you do not have in your environment. In your IDE. Maybe try to merge some of those features that are in GitLab, into here. Because that's one of the advantage of having this tool is to have all of these tools and all of information right locally instead of depending on GitLab for example.

Q *Is there any specific feature in GitLab that you are hinting at?*

S I'm not sure. Not right now. I say GitLab but I could say GitHub, I could say BitBucket, or whatever. I don't know, Atlassian. I don't know.

Q *So basically you would like to have all the different information sources in one tool, summarized?*

S Yeah.

Q *Okay, that makes sense. Any other general feedback that you have on the tool that you haven't mentioned before?*

S I don't know, that's pretty much it I guess. Yeah, is there way to implement this inside the IDE, inside PhpStorm?

Q *I wanted to do that initially, but then it took we one week to get the Hello World plugin to run. And then I gave up and I just build a web interface, because it was not worth the efforts I think. So, but it should be possible. It was the initial aim, but within my thesis not feasible timewise.*

S Yeah, I get it.

Q *But that was indeed — that you just had the PhpStorm shortcuts and just —*

S I mean, the only disadvantage is that when someday a lot of the users change from PhpStorm to Netbeans or Eclipse or whatever —

Q *Or I could use an iframe in PhpStorm.. — yeah that doesn't really solve.. — it's not really native. Cool. What's your impression of the bunq backend? What do you think?*

S Wow, it's massive. I mean, I was looking through the methods and it was so confusing everything. I mean, I have worked in PHP and my experience with PHP is basically using I don't know, using frameworks like Yii, Symphony, Laravel but this doesn't have any similar things compared to them from what I see. Like yeah you have the — I mean, it's very well structured. That's one thing. But having thousands of folders in the same folder is a bit overwhelming.

Q *It used to be all separate repositories.*

S Oh really?

Q *They merged it like one - one and a half year ago. In one core repository. But it used to be all github repositories — Gitlab sorry.*

S I mean instead of using one folder for all — change the structure to be able to, for example, if you want to get into, I don't know, push notifications I guess, —

Q *You can click once and write 'push' [PhpStorm project structure sidebar] — now you can go through all the folders that have 'push' in it, it's a really useful feature, that's how I normally —*

S See, you depend on the IDE to be able to do this. If your IDE is different or you don't have this tool, then having different structure will be very useful. In that case you don't need to depend in the IDE. I mean, this is great if you use PhpStorm I guess.

Q *I guess we will have to stick with that then. Okay cool. Do you think it's feasible to implement/improve features with this tool, but then without the help of a senior developer?*

S I mean, you still need to know what the code is about. I had no idea what the code — or where I was something. It's more useful to have knowledge about it but if you don't know anything about it it will give you more insights for sure.

Q *Okay, cool. And how did you phrase it — Never mind, I wanted to grab onto something you said at first.. yeah, never mind — So, when you are assigned to work on such a huge framework or a code base and you have to learn it basically, what are the most important sources of information for you then? In an ideal world?*

S What do you mean? In tools?

Q *Tools, or any kind of resource, where would you get the information from preferably? What works best for you?*

S Well usually the code is usually is explaining itself normally. In this case where you don't have any comments, in that case that will be very complicated. Even though the code is self-explanatory. Or it wants to. I will guess the second source will be documentation which write outside the code.

Q *Like the wiki?*

S Like the wiki. And then the third one is probably my last resort, is to waste someones time to ask him about the code.

Q *Ah yeah, I see. Okay, cool. Anything else that you would like to mention? Or maybe you have a question for me?*

S How long did it take to do this?

Q *To build? I've been working on this for months. I started in November, first of course literature study. I think I started actual development in January/February somewhere. And then a few months. I had to learn React, graph databases.*

S Ah it's React? Nice.

Q *Yeah, it's an React interface. Yeah I really fell in love with React actually, like I had never worked with it, I was planning on doing everything jQuery and then I found React and [a frontend developer] is really fond of React so I was like 'okay whatever, I'll try it', and I found a nice CSS framework. So it even looked okay. So yeah, that was cool.*

S Yeah the best thing is just leave it simple. Don't add any fancy features.

Q *Yeah it should be quite straight-forward.*

S Yeah because, at least from the coders perspective you want as least text as possible and get into the tools like the useful information.

Q *Yes, exactly. If I may ask, did you read the information on the left or..?*

S Oh no.

Q *Yeah, there's information on the left. It even changes depending on the search thing that's expanded. That also explain what these things mean —*

S Well I didn't had to need to read. So I guess that's good.

Q *Yeah that's a good thing, but if you wanted to know 'what does it mean that this tab is in this list' then it would've given a little more insight. Or well, that was the idea at least.*

S I was so inside in the tool that I did not really had to —

Q *Well then at least the interface worked. Cool. You have no further remarks?*

S No. That's everything.

C.11 Subject 11

Questioner (Q) *Oke, kan je me vertellen wat je dan zou moeten doen / wat je hebt gedaan?*

Subject (S) Eerst wat ik zie, ik moet een endpoint voor [redacted] zoeken. Dus als ik jou tool gebruik, wat ik gedaan heb, is Find By Label en toen heb ik [redacted] ingetypt. Nou, ik weet dat het Payments zijn in ieder geval dus, als ik dan op search klik dan zie ik automatisch de views en controllers. Nou wat ik eerst heb gedaan, ik moest een endpoint vinden. Het was in dit geval dat ik aan de naam goed kon zien dat [redacted]TransactionView redelijk overeenkwam met de "[redacted]-transaction" waaraan je kon zien dat die het was. Wat misschien wel handig is om de eventuele endpoints in de views hier erbij te zetten.

Q *Ah, dus zeg maar wat in de view zelf staat in de JSON —*

S Exact. Misschien dat het dan te vol raakt maar dat zou het hele proces iets kunnen vergemakkelijken. Vervolgens heb ik die map geopend en kijk ik bij [redacted]`_transaction_view`, in ieder geval, ik heb het dus voor Create geïmplementeerd. Wat eigenlijk voor Read had gemoeten. Wat ik eerst heb gedaan, ik heb eerst even de definition erbij gepakt en dan heb ik een username aan de response toegevoegd. Maar omdat ik het voor Create heb gedaan heb ik het ook aan de request toegevoegd. Vervolgens heb ik bij de ViewPreProcessor gekeken. Daar zag ik dat Workflow[redacted]TransactionCreate werd gebruikt. Nou ja, dat zag ik gelukkig ook hiertussen staan, in jou tool, dat ie hier er ook bij stond. Wat ik toen gedaan heb is, ik heb dit ding eventjes aangeklikt en toen kon ik makkelijk zien van hoe wordt ie gemaakt. Nou ja, ik zie dat er een createNew[redacted]Transaction function is. Wat ik heb gedaan is, ik ben dus naar de klasse gegaan. En dan, had je daar de createNew[redacted]Transaction functie. Ik zag, daar mistte de username, nou ik heb gekeken hoe is dat bij counterpartyName gedaan? Dat heb ik exact hetzelfde geïmplementeerd. Dus wat ik heb gedaan, ik heb toen vervolgens naar de definition gegaan hiervan. — Dus die staat hier —. En toen zag ik van aan het begin van 'oh ik moet hier nog een username toevoegen' nou hoppa username erbij. Generate gerust. Vervolgens die username er hier weer bijgezet. — Ik zie hier nu wel dat ik een dingetje mis. Toen vervolgens zag ik inderdaad weer bij die createNew[redacted]Transaction functie van 'oke hier moet ik een NameSimple username in doen', die kan ik vervolgens in de create die username weer meegeven door automatisch dus username in het object hebt. Vervolgens ben ik weer teruggegaan naar die ViewPreProcessor en toen heb ik de username hier dus toegevoegd. Toen moest ik er nog wel een parse functie voor schrijven. Die was eigenlijk exact hetzelfde als counterpartyName alleen dan het veldnaam veranderen. Vervolgens ben ik naar de post processor gegaan. Hier heb ik het field name toegevoegd, dat ie nu vanuit de Transaction model de username kan ophalen. Dat is in principe hoe ik het gedaan heb. D'r zal waarschijnlijk ook wel een andere manier geweest zijn waarop je de user vanuit een van de attributen van de [redacted]TransactionModel kon krijgen, en een nadeel hier was ik kon ook niet echt heel makkelijk zien welk model ik

hiervoor zou moeten — naja, ik weet dat het dit model is, maar wat misschien handig had geweest is als je net als bij die Workflows, als je hierop klikt dat je dan gelijk die gevisualiseerde versie hebt. Ik denk dat je dat voor models misschien ook wel kan doen, dat je misschien die relations met andere tabellen doet. Want dan kun je in ene oogopslag heel makkelijk zien 'hier kan ik een User vandaan halen' of niet. En [redacted]Transaction heeft dan wel de User, maar die is alleen voor ACL dan. Het is niet de bedoeling dat je die gaat gebruiken. Maar dat vond ik met deze tool nog wel een beetje lastig te zien. Ik kon niet echt in een oogopslag makkelijk zien 'waar kan ik die username vandaan halen', want die moest ik nu wel weer toevoegen. Want dit is heel veel scrollwerk en wat dan misschien ook handig is is als je dan relations hebt dat je dan op de model kan klikken en zo weer een beetje door kan klikken totdat je ergens misschien een user vindt. Dat leek me wel handig. Ik weet niet of het nog met de huidige attributen dat je daar een user van kan krijgen..

Q *Het is mogelijk ja, je moet inderdaad naar de relations kijken van — het zit wat verder weg gestopt. En dat komt met name op het moment dat je de transactie aanmaakt dan is nog niet bekend wie die transactie gaat betalen. Dat weet je pas op het moment dat iemand met zijn telefoon, die ingelogd is, de QR code scanned. —*

S Oh, oke.

Q *Dus het zit net een stapje verder in het proces. Maar het ging mij er met name om dat je wel zag van dat je die view kon vinden dat je de models moest hebben, en dat had je wel allemaal naar boven dus. En in hoeverre heeft de tool je daarbij geholpen of juist niet geholpen?*

S Ja, hoe ik het normaal gesproken zou doen is, ik zou een ⌘ + ⇧ + F op het endpoint doen. Dat heb ik altijd zo gedaan. Het enige nadeel bij [redacted] is dat het er ontzettend veel zijn, dat PhpStorm crashed. Dus, dan is het wel handig als ik hier gewoon op de F.. — het is meer dat het nu een automatisme is geworden om automatisch command-shift-F te doen. Maar als ik hier op [redacted] zoek dat ik hier heel makkelijk kan zien deposits of payments, dat ik weet payments die is handig dan kan ik search doen en hoppa dan heb ik gelijk die views erbij. Het zijn wel veel views. Dus zoals ik al zei, misschien is het ook handig om ergens een overzichtje van de endpoints erbij te maken. Dat zou het proces nog iets makkelijker maken. Daarnaast is het misschien ook überhaupt handig om misschien een Search By Endpoint te doen. Ja, ik weet niet of het heel veel toegevoegde waarde zou hebben maar toevallig voor deze opdracht wel. Maar misschien dat je ook nog wat keywords gewoon ergens kan verwerken. Maar ik kan me natuurlijk ook voorstellen dat als het alleen — als het alleen voor endpoints is dat je dan het gewoon command-shift-F kan doen.

Q *'T kan, maar op zich heb je hier wel meer vrijheid in hoe je je informatie geeft.*

S Klopt, dat is zeker waar. Wat hier nu wel is, is ik kan in een oogopslag zien welke onderdelen hier allemaal bij betrokken zijn. Kijk, en nu is het zo dat die [redacted]`_transaction_controller` die is vrij voor de hand liggend dat die er bij

zit, maar als je hier kijkt naar `core_token_qr_request_controller`, daar zal je dan waarschijnlijk de user vandaan moeten halen dan.

Q *Dat is om de QR token aan te maken dan. Zeg maar die de User te zien krijgt.*

S Ja nou waarschijnlijk in een van deze andere controllers kan ik dan misschien die User ophalen en dat kan ik dan misschien ook in de Models zien. Maar misschien is het handig ook om te zien welke model — hoe deze models onderling met elkaar gelinkt zijn. Want ik weet wel dat deze en deze een relation hebben, volgens mij. Ik bedoel deze en deze en nog eentje.

Q *Vast de Inquiry, ik weet niet of die hier tussen staat.*

S Ja, want wat ik dan handig zou vinden is dat misschien bij die search dan ook Find By Model misschien erbij staat.

Q *Dat je meer expliciet op de Model View Controller structure kan zoeken?*

S Precies, dan hoef je eigenlijk maar een tabje bij te maken dat je als het ware hier selecteert Model Controller of View. Vervolgens een keyword, waarmee je een beetje op die naam kan filteren. En dat je dan een lijstje krijgt en als je dat dan nog kan visualisen, dat zou het nog veel makkelijker maken. Want uiteindelijk wat je nu dan toch doet is terug naar PhpStorm en dan bestand openen of op die naam — omdat je dan wel makkelijker kan zien.. Maar als je die functionaliteit zou kunnen toevoegen dan zou het echt — laat ik het zo zeggen, dan zou ik hem de hele tijd gebruiken.

Q *En waar zou dan met name het voordeel in zitten volgens jou als je die feature zou hebben? Wat is de grote toegevoegde waarde?*

S Ja, het vervelende aan PhpStorm is dat — maar dat ligt waarschijnlijk wel meer aan onze code-base — is als ik een Model nodig heb wat ik niet ken dan als ik een Model open dat ik een hele waslijst krijg aan allemaal attributen waarvan ik werkelijk geen idee heb hoe ze onderling met elkaar zijn. En als dat opgelost kan worden dan scheelt dat enorm veel tijd. Want nu bijvoorbeeld als ik had geweten hoe ik die User had kunnen ophalen dan dat ik dan bij deze Models hier ergens op een Model kon klikken, nouja ik weet sowieso ik moet die TransactionModel hier hebben, maar als ik daarop had kunnen klikken en dat ik dan net als bij die Workflows een popupje krijg met wat de relations zijn en dat ik daar ook op die models kon klikken en dan vervolgens daar weer hetzelfde voor geldt, dan was ik er waarschijnlijk wel achtergekomen. Nu heb ik de makkelijkste oplossing gebruikt denk ik eigenlijk. Het is niet de ideale oplossing.

Q *Nee, maar je hebt wel kunnen vinden waar je moest zijn, dat was het einddoel.*

S Ja, ik had waarschijnlijk ook wel die Library kunnen gebruiken. Laat ik het zo zeggen, misschien mist een toegevoegde waarde alleen al een lijst van de functies.

Q *Ja, precies. Dat je kan zien wat erin zit.*

S Ja, want PhpStorm heeft dat volgens mij ook —

Q *Ja je kan onderin op 'Structure' klikken en dan kan je zien wat er in de huidig geselecteerde klasse zit.*

S Ja, precies. Nou, zoiets in die tool dat zou echt ideaal zijn. Denk ook dat het niet zo heel veel moeite is.

Q *Op zich niet, de tool die draait nu niet op de codebase dat is met name het ding. Hij draait op de git history.*

S Oooh.

Q *Zeg maar die grafiek nu wat ik heb gedaan is hij pakt de filepath, die haal ik uit de changelogs zeg maar van git, ik weet welke bestanden worden getouched. En als er een workflow tussen zit dan whatever dat wordt allemaal in een database gedonderd, die paths. En als ik zie, deze path matcht een regex voor workflow dan ga ik gewoon proberen voor dat path de current verzon uit develop te pullen. En dan parsed ie de JSON. Om de grafiek te maken.*

S Ja, oke. Nee, dat verhelderd het al inderdaad.

Q *Maar dat zou op zich ook wel toegevoegd kunnen worden voor de Models, Views maar dat heb ik gewoon niet erin zitten nu.*

S Nee, nee, precies. Maar dan — Ja ik werk eigenlijk op 1 scherm, omdat ik 1 scherm heb, ik heb er wel 2 nodig alleen — maar als ik dan op dat andere scherm die browser zou kunnen hebben en op het andere scherm gewoon PhpStorm dat ik dan op structure hier kan klikken en dan dit zien, dan is het gewoon prima. Ja, en dan eigenlijk als je het op 1 scherm zou doen — maar de meeste developers werken wel op meer dan 1 scherm — dan is natuurlijk een PhpStorm plugin ideaal. Maar ik weet hoeveel werk dat is.

Q *Nou, ik was ermee begonnen maar zelfs de Hello World kreeg ik niet draaiende na een week dus toen had ik zoiets van 'wacht maar, ik begin wel gewoon een web interface, dat werkt'.*

S Ja, die PhpStorm plugins die zijn wel [mumble].

Q *Ja, en er is geen goeie documentatie voor.*

S Nee, exact.

-

S Nou in ieder geval zoals ik al zei, omdat het hetzelfde werkt als [redacted] pak je gewoon Feature [redacted] erbij, nou dan zie je in een oogopslag van `token_qr_request_controller` nou dan kunnen we er een beetje vanuit gaan dat joh — dat staat er ook van 'the user scans the QR code and the backend assigns the user to the transaction'.

Nou dat, dan kun je nu dus direct al zien dat `core_token_qr_request_controller` en dan kan ik dus zien dat ik dat ik een soortgelijke workflow TokenQrRequestIdeal-Create [redacted], maakt verder niet zo heel veel uit. Ik kan hier ook in ene keer zien wat ik nodig heb. Dus dit zal redelijk hetzelfde zijn allemaal. Naja, het is wel allemaal redelijk hetzelfde natuurlijk, alleen dat stukje update hier [update the related transaction] zal net verschillen, omdat dit over iDeal gaat en dan zou je dus hier over AliPay moeten hebben. Dit blijft waarschijnlijk onveranderd.

Q *Wat is dit? TokenQrRequestGenerate? Ja dat is een generiek — tenminste hij is wel gewijzigd natuurlijk, bij het zoeken naar deze feature.*

S Ja, maar nu kan ik dus wel makkelijk — misschien is dat trouwens ook wel goed om toe te voegen, omdat je toch altijd alles uit de git data haalt, uit de git diff — dat je kijkt 'wat is eraan toegevoegd?'. Als 'ie niet nieuw is maar gechanged is dat je dan de changes kan bekijken. Dat zou ook wel nice zijn.

Q *Ah zo, ja.*

S Wat je voor een Feature kan zien wat je aan een bepaalde workflow hebt toegevoegd. Waarschijnlijk zou het nu voor beide wel —

Q *Waarschijnlijk kan je in de git log de diffs erbij trekken, dat kan sowieso weet ik, en dan kan je die vast wel filteren op commit messages die dan die reference hebben naar een issue number — oh maar je wilt meerdere issues hebben.*

S Ja, je wilt het allemaal een beetje bij elkaar. Als je dat met een druk op de knop zou kunnen doen, dat zou echt perfect zijn. Dat je gewoon echt heel makkelijk kan zien wat er is toegevoegd of is weggehaald. Nja, dit geld dus ook hier voor, voor de QrCodeGenerate, dit ook, —

Q *Wat is dat?*

S De RequestInquiryDetermineSenderAndReceiver, hier had ik dus ook weer waarschijnlijk de User uit kunnen halen denk ik, als ik nu zo kijk. Want die heeft die sender en receiver. Ja ik had waarschijnlijk wel —

Q *De receiver is de webshop in dit geval. —*

S Ja, precies, maar die sender die zit in die Decree toch?

Q *Sorry?*

S Die sender zit in zo'n Decree toch?

Q *Eh, allebei dacht ik. Ik moet je heel eerlijk zeggen, de decrees heb ik niet zelf aangeraakt.*

S Ik vond dat ook altijd een beetje een opmerkelijk ontwerp.

Q *Volgens mij is de Decree wordt getranslate naar een message die het netwerk over gaat. Dus bijvoorbeeld als je een Mastercard payment doet dan krijgen we een XML binnen, raw data,*

S Mastercard heb ik van gezien hoe het eruit ziet. Het is echt alleen maar getallen. Met spaties en —

Q *Ik heb het testdata script gezien voor Mastercard payments, en alle stappen die je daar moet doorlopen om een transactie op te zetten..*

S Ja, dat heb ik ook gedaan, ik heb er een utility_script voor geschreven. Dat is een drama, jongen, echt,..

Q *Ah je hebt er een script voor geschreven? Nice.*

S Even kijken, nou ik kan dus, ik moet waarschijnlijk iets als een WorkflowAliPayTransaction(Controller) maken. Naja, dan weet ik dus ook dat ik daar de model en view voor moet maken. Nja, die kan ik hier ook makkelijk zien. De `core_[redacted]_transaction_view` een zelfde soort voor AliPay maken. Dit moet misschien `token_qr_request_view`, is misschien ook wel handig, wat ik al zei, als ik hierop kan klikken dat ik kan zien hoe de request en response eruit zien, dat ik dan makkelijk kan zien wat ik hier moet aanpassen. Hetzelfde geldt voor bunq.me en deze. RequestResponseViewPostProcessor in iets hoeft aangepast te worden. Bij models kan ik direct de AliPayIssuerModel aanmaken, Certificate model er voor aanmaken. Transaction model voor aanmaken. Waarvan die TransactionModel de belangrijkste is. Dat je daar echt die transactie in hebt. Volgens mij ben je er dan al bijna. Nja, misschien nog — naja er zijn eigenlijk ook niet echt libraries. [...] Nou, ja hier kan ik — wat misschien ook handig is je zoekt hier op een bepaald Label, als ik hier op [redacted] zoek misschien handig als ie die ook highlight.

Q *Ah ja.*

S Zodat je ziet van oh [redacted], oh die is belangrijk, die is specifiek voor deze Feature. En het is niet zo'n global — je kan het zo prima ook zien, maar het is misschien ook wel nice.

Q *De pest is alleen dat het label is natuurlijk '[redacted]', dus hoe ga je dan alleen bepalen dat je alleen [redacted] moet highlighten?*

S Misschien is het dan handig dat je ook nog ergens een keyword kan meegeven.

Q *Ja, of dat je hier inderdaad een search bar zet van 'dit moet gehighlight worden'.*

S Dat zou sowieso nice zijn, want stel je zou bij online payments kijken dat je dan iDeal kan doen [redacted]. Dat je alles in een oogopslag hebt, dat je dat kan highlighten of filteren of wat dan ook. Dat zou echt handig zijn, dat je kan zien wat specifiek voor deze Feature is en wat gedeelde functionaliteit is en wat gedeelde functionaliteit is binnen de verschillende payment methods. Ik mis hier wel daemons, kan dat kloppen?

Q *Oh ja, die zitten er niet in. Nou, misschien dat ze bij Miscellaneous staan trouwens. Kijk hier staat 'daemon'.*

S Ja, oke.

Q *Maar die staan niet in een apart tabje.*

S Ja, dat wou ik net zeggen, dat is misschien wel handig om die in een apart tabje te doen als dat mogelijk is.

Q *Ja, moet wel de interface wat breder maken dan.*

S Migrations missen ook volgens mij?

Q *Ja, dat is een andere repository. En ik doe alleen maar core indexen.*

S Oh ja. Klopt. Misschien ook handig. Dat je kan zien welke velden er zijn toegevoegd. Dan hoef je namelijk niet meer het hele model te parsen. Dat is eigenlijk misschien wel makkelijkere manier voor die models visualisen.

Q *Maar dan moet je de SQL parsen.*

S Daar heb je vast wel een library voor.

Q *JSON parsen is echt kapot makkelijk. Want het is JavaScript al, dus —*

S Ja, dus dat is gewoon bijna native.

Q *Ja, en PHP kan je op zich ook best wel goed —*

S Het zal vast mogelijk zijn, maar ik zou zeggen, het is misschien een makkelijkere manier dat je niet die hele SQL zooi hoeft te pullen. Want dan moet je namelijk ook weer een generate runnen want die staan niet standaard op de repository, op de core, de SQL?

Q *Oh zo, nee.*

S Ja, hij staat wel in die migrations natuurlijk.

Q *Nee wat je kan doen natuurlijk, als je een migration schrijft dan zet je de SQL files die zet je erbij, die staan er gewoon in.*

S Ja, precies en dat zou echt heel nice zijn.

Q *Ik hoef die hele repository dan niet eens te pullen want ik kan gewoon wederom over de git log heen gaan en dan heb ik alle file paths en dan kan ik gewoon op het moment dat jij een bepaald filepats oproept, kan ik zeggen 'oke pull nu de contents van develop en kijk maar wat de change was in die file'.*

S Dat zou ook echt heel handig zijn voor bij Models dat je die misschien kan splitten dat je hier twee tabjes hebt dat je models hebt en dat je ook de migrations hebt en daarin kun je waarschijnlijk zonder al te veel moeite heel makkelijk visualisen wat er voor deze feature aan het model is toegevoegd. Ik kan bijvoorbeeld nu al zien `core_token_qr_request_model` die is aangepast, of die is aangemaakt, dat weet ik niet, dat is misschien trouwens ook handig om erbij te zetten, een labeltje ofzo —

Q *De status?*

S De status inderdaad, of 'ie is toegevoegd of dat 'ie is aangepast.

Q *Dat is lastig, want je kijkt naar commits natuurlijk maar het kan zijn dat er binnen een feature meerdere commits zijn geweest die een bestand eerst create, dan modified, dan weet ik veel wat. Dus wat houd je dan aan, want als je op een andere manier zoekt kan het weer zijn in een andere scope kan het zijn dat 'ie alleen maar gecreate is, terwijl als je een kleinere scope neemt dan kan het alleen maar een modify zijn.*

S Maar wat is de scope van de hele feature net?

Q *Als je de scope van de hele feature neemt dan heb je dus waarschijnlijk van creation tot live-gang. Maar als je bijvoorbeeld zoekt naar een bug in een issue, dan wil je niet laten zien 'hij was nu created' nee hij was alleen maar modified nu ik zoek op dit issue. Dus dat is nog wel een challenge dan.*

S Ja ik er misschien ook zo 1, 2, 3 niet echt een.. —

Q *Het is wel iets om even over na te denken.*

S Want het zou wel toegevoegde waarde hebben dat — want bijvoorbeeld hier kan zien — naja, ik kan hier wel een beetje beredeneren — ja dit is waarschijnlijk de hele feature, dus het is wel aangemaakt hier. Maar als je dan kijkt naar die subproblemen dan kun je wel een beetje beredeneren dat deze niet aangemaakt wordt, maar alleen aangepast wordt. Maar in ieder geval die migrations lijkt me wel echt handig om hier bij te hebben, dat je — hoeft niet eens helemaal gevisualiseerd te worden die SQL maar dat je echt alleen — aan de query kun je ook al zien wat er is gebeurd.

Q *De raw contents een beetje? Dat je alles in een view hebt. Zullen we maar — tenzij je nog meer had nu trouwens?*

S Ik denk dat dat het wel een beetje was. Misschien dat je hier nog descriptions van de test erbij kan zetten —

Q *Ja, die heb je er niet altijd bij.*

S Oh ik dacht dat dat een code convention was.

Q *Anyway, even snel nog een — wat vond je van de assignment?*

S Assignment was duidelijk, nice, laat ik zo zeggen, het past goed die tool toe, en dat moet ik zeker bij part 3 zeggen want het is nu, wat ik net ook al tegen je zei, als ik dit had geweten bij een van mijn eerdere issues dan had ik dat gewoon op deze manier gedaan. En had het zoveel sneller af geweest.

Q *En het grootste voordeel daarbij had dan geweest? Of de voornaamste reden om dat dan erbij te pakken?*

S Dat ik in ene oogopslag direct kan zien, als ik een feature soortgelijk aan iets, zoals dat voorbeeld dat was Pack membership, dat ik in een oogopslag kan zien 'wat moet ik toevoegen?'. Naja het is altijd meer dan je denkt. En je mist altijd iets. Zeker in zo'n grote codebase met allemaal verschillende mapjes en ook niet alles heeft Pack in de naam zitten enzo. En dat vond ik heel fijn hier aan. Je kan in een oogopslag zien "wat moet ik hier toevoegen als ik een soortgelijk membership wil maken?".

Q *Ja oke, helder. Find By Label die heb je gebruikt, met welk doel heb je die gebruikt met name? Wat was een beetje je verwachting?*

S Ik zal eerst even voor 2 kijken, het ging eerst over [redacted], dus toen dacht ik 'nou, dan zoek ik op [redacted] en dan kijk ik wel wat eruit komt'. Nou, dat vind ik dan heel handig dat ik die auto-complete heb, want niemand weet de features uit zijn hoofd. Nouja, dan zag ik dus de Deposit, dacht ik, en de payments. Dan weet je, payments. Nou dan wil ik even kijken van wat is er allemaal nodig geweest voor die feature? En daar kan ik dus wel gewoon redelijk makkelijk zien wat er allemaal nodig was. Moet wel zeggen dat dit wel redelijk specifiek was, het is echt een bepaald endpoint. En daar is deze tool misschien iets minder geschikt voor.

Q *Als in?*

S Nou als ik een endpoint [redacted]-transaction moet vinden dan is het waarschijnlijk sneller om hier ⌘ + ⇧ + F en dan endpoint.

Q *Ik had misschien de naam van het endpoint niet zo expliciet—*

S Als je het niet zo expliciet had gezet dan was het misschien iets duidelijker geweest. Als ik nu gewoon endpoint zoek en dan zet ik erbij [redacted]-transaction, dan heb ik 'm sneller. Dan klik ik op dit dingetje en dan weet ik ook gelijk 'oh het zijn deze'.

Q *Maar dat weet jij, maar zou een junior of nieuw developer dat ook kunnen?*

S Als 'ie niet weet hoe die endpoints gemaakt worden dan nee. Dit kan wel helpen in het licht van het leerproces. Maar zoals ik net al zei bij part 3, als je een Feature hebt die heel oppervlakkig is, waarbij je eigenlijk alleen een naam weet en wat het moet doen, dan is dit ideaal. Dat scheelt zoveel tijd. Ja echt, als ik dat had geweten voor die [redacted].. —

Q *Dan had ik je wel wat eerder laten testen.. De Find By Developer had je die gebruikt? Wat? En if so, why?*

S Ik heb die niet gebruikt. Ja ik heb het alleen voor die eerste gebracht omdat ze expliciet vroegen —

Q *Ja, precies. En met welke reden heb je de developer niet kunnen gebruiken.*

S Ik zag de toegevoegde waarde er niet echt voor.

Q *Wat ontbreekt er in die view om het voor jou van toegevoegde waarde te laten zijn.*

S Ja, laat ik het zo zeggen, ik zou deze tool eigenlijk alleen gebruiken om de architectuur van bepaalde features te bekijken. En dan ga ik even heel plat zeggen, dan interesseert het mij geen reet wie het gemaakt heeft. Ik weet niet of jij een voorbeeld hebt waarvan je zegt 'daarvoor is het echt handig'?

Q *Ik heb er wel ideeën bij maar dat ga ik je nu niet in je hoofd praten natuurlijk. Dat beïnvloed. CIT had je die gebruikt?*

S Nee, maar dat wat ik hier had kunnen doen, dat vind ik heel fijn dat hier aan de onderkant wel de issues erbij staan en dat was misschien ook wel handig geweest als ik er nu achteraf naar kijk, nu ik de [redacted] zie en dat ik hier [redacted] transactions endpoint zie, dat was me net nog niet opgevallen. Dan zie ik incremental#716,

Q *Ja, dan kom je bij de issue maar dat is basically het main issue van dit endpoint, dus dat geeft praktisch hetzelfde als op de feature zoeken.*

S Dit is handig, dit had ik beter voor die tweede [opdracht] kunnen gebruiken.

Q *Wat precies is handig? En hoe had je dat —*

S Dat ik nu niet echt die hele waslijst van al die view files die aangepast zijn krijg. Het zijn er nog steeds wel —

Q *Je hebt nu gezocht op een specifiek issue? Welk issue? Oh, #716, dat is dat main issue?*

S Ja, dat is die endpoint. En in Principe, naja, die 500 [ander issue] kan je misschien ook wel al zien van wat is hier gebeurt voor — die endpoint geeft een exceptie, oh welk endpoint is dat? Dan zie je nu hier al direct van dit is de `core_[redacted]_transaction_view`. Het was wel halverwege dat ik erachter kwam dat die issues hier stonden, ze vielen niet zo op.

Q *Ze stonden eerst bovenaan maar bij sommige features werd het zo'n lange lijst, en ik kreeg als feedback 'ja die bestanden zijn more important' zeg maar. Dus naar beneden gegooid. Het zou eigenlijk een beetje side-by-side, maar ja —*

S Wat ik nu wel denk dat by Feature By Developer wel handig is, is als ik ergens aan een feature aan het werken ben dat ik kan zien wie hem gemaakt heeft en als ik ergens vast zit dat ik diegene kan vragen.

Q *Dat was ook de intention die ik erachter had zitten.*

S Maar dan is het misschien handiger om het andersom te doen, want dan zoek je alsnog op Feature en niet op developer.

Q *Ja, maar je kan hiervandaan kan je nu doorklikken naar een developer.*

S Ja, precies maar dan heb je niet perse Feature By Developer nodig.

Q *Nee precies.*

S Maar daarvoor is het wel handig om erbij te hebben staan, want anders zou je de annotate moeten gebruiken want dit.

Q *Ja precies, en dat is ook niet heel chill. Laten we naar de volgende vragen gaan. Voordelen van de tool? Ik weet niet of je er nog meer hebt naast wat je al genoemd hebt? Je hebt er al veel genoemd.*

S Ik denk dat dat wel was.

Q *Tekortkomingen van de tool? Heb je die nog? Naast wat je al —*

S Die heb ik hier op het papiertje staan, maar dit hier heb ik ook allemaal al —

Q *Ja oke. Als dat allemaal genoemd is dan prima.*

S Het is niet zozeer tekortkomingen maar voornamelijk verbeterpunten —

Q *Ja, precies. Het is altijd lastig om dat in te pakken dat mensen niet denken van 'shit ik wil het niet zeggen op deze manier'.*

S Nee precies, maar er zijn wel een aantal dingetjes waarvan ik denk van 'ja dat mist wel echt'. En dat is dan vooral dat stukje met die models. Dat mis ik wel. Want nu heb ik er verder niet heel veel aan. En dat lijkt me niet al te veel werk om dat toe te voegen en dat kan wel heel veel —

Q *Het is wel eventjes werk, het was voor mij nu te veel werk nog, maar —*

S Ja, nee, dat begrijp ik, maar in principe zou dat —

Q *Het is doable. —*

S Voor de tijd die je erin steekt is de reward wel groot genoeg.

Q *Ja. Had je in general nog feedback die je nog niet hebt genoemd?*

S Nee, ja het ziet er nice en strak uit en niet zoveel, weet ik veel wat, friemeltjes aan. Het is lekker gebruiksvriendelijk. En ik weet dat informatica-mensen daar nog wel eens problemen mee hebben..

Q *Meer knopjes, meer knopjes,..*

S Ja precies, nee maar dit vind ik heel duidelijk. Wat ik wel vind,.. dit, is misschien niet ideaal. Die related labels. Misschien zijn het er ook gewoon te veel.

Q *Dat zou kunnen, dat er gewoon een limit op moet zitten.*

S Ja precies en dat je dan op extend kan klikken en dat je ze dan allemaal ziet.

Q *Ja, dat is wel een slimme inderdaad, daar had ik niet aan gedacht. Ik had eerst had ik een lijst die zo naar beneden liep net zoals in die developer page. En dat was ook niks, dus ik dacht nou ik doe het wel zo.*

S Oh maar dit is wel handig op zich. Oh ik kan hier ook gewoon op klikken.

Q *Ja, je moet wel op het label en niet op de scrollbalk klikken. Maar ja, dat —*

S Oh en dan kan je ook een beetje zien waarmee het samenhangt.

Q *Ja want je ziet dan vaak dat bestanden voor meerdere features worden aangeraakt. En hoe meer dat in common is, zeg maar, dan wordt ie hier hoger. Dat is het idee daar. Oke, nog een paar generiekere vragen. Je indruk van de bunq backend, wat denk je daarvan als je dat ziet?*

S Groot. [redacted]. Wat ik wel moet zeggen is dat het consistent is, dat zou het ook in de commit messages moeten zijn en daardoor is een tool als dit dan mogelijk. Want ik weet wel bijvoorbeeld bij andere projecten kun je niet zomaar zo'n tool gebruiken want heel veel mensen hebben dan een commit message als 'implement changes', ofzo waarmee je dus eigenlijk helemaal niet weet wat er gebeurd is.

Q *Messages als 'what the hell, work', 'start working', 'ci testing', 'ci still testing'..*

S Boos worden en weet ik het allemaal. Dat is hier handig aan. Wat ik heel irritant vind aan de bunq backend wat dit wel oplost is al die mapjes. Ik weet niet hoeveel het er zijn maar het zijn er wel meer dan duizend denk ik.

Q *1800 zoveel geloof ik?*

S Naja, daar vindt je niks tussen, en dan is het heel handig om dit overzicht te hebben. Van welke map heb ik überhaupt bij deze feature horen. Want laat ik het zo zeggen, als ik het had gemaakt dan had ik wel die mapjes wel weer genamespaced, per feature. Maar dat is wat je nu eigenlijk ook hebt.

Q *Soort van ja.*

S Hoe bedoel je verder nog wat van de backend?

Q *Gewoon wat jou indruk was toen je de backend zag zeg maar.*

S Vooral veel. [redacted]. En dan duurt het wel even voordat jij door hebt waar wat gebeurd en daar helpt dit wel bij. Het is nu omdat ik wel al wat meer gewend ben aan die backend dat ik dan automatisch al Command-Shift-F en dan die endpoint ga doen, maar laat ik het zo zeggen, als ik hier m'n eerste dag zou werken dan had ik geen idee gehad, echt niet. Dan is dit the way to go eigenlijk, dat je een beetje kan verkennen.

Q *Oke. Cool. Denk je dat deze tool — denk je dat het mogelijk is dat deze tool de taak van een senior, we gaan vaak naar een senior toe om vragen te stellen, zou deze tool dat deels kunnen opvangen slash vervangen?*

S Als ik weet wat ik voor vragen stel dan denk ik het niet. Nou, het kan het wel verlichten. Maar niet vervangen. Want wat ik vooral heel erg had is omdat [redacted] dat ik geen idee had hoe ik die workflows, die JSONs definitions, daar had ik gewoon geen idee over. En dat waren meer de vragen die ik aan seniors stelde of hoe kan ik een IBAN van een bepaalde user ophalen. En dat is misschien wel handig om toe te voegen, maar dat is natuurlijk ook weer met de git history weer iets lastiger om te doen. Maar die libs die kunnen zoveel, dat je daar misschien een zoekfunctie voor hebt. Dat je —

Q *Zoekfunctie voor libs?*

S Ja voor de methodes die in een lib staan. Als je bijvoorbeeld hebt getIbanByUser dat je autocomplete krijgt met getIbanBy weet ik veel, Monetary Account, dat soort dingen. Als ik dat had gehad dan had ik [a senior developer] zelf wat minder vragen gesteld.

Q *Ja, je weet gewoon de structuur dan niet.*

S Ja, ik denk niet zozeer dat het echt, want als jij hier net begint ga je echt niet aan zulke grote features werken.

Q *Niet direct nee, je wordt eerst —*

S Je wordt eerst een beetje klaar gestoomd inderdaad. En zodra je dan aan die grote features gaat werken dan verlicht dit wel echt de workload van de senior, maar ik denk als jij helemaal aan het begin zit dan is het meer echt heel low level wat je niet begrijpt. En ik denk dat daarin deze tool nog wel verbeterd kan worden. Maar dat hangt ook van af wat het hele doel is van deze tool was, is het om—

Q *Vertel ik je zometeen. Nog één vraag en dan ben je in principe wel verlost en dan kan ik je vertellen .. — Wat zijn volgens jou de belangrijkste informatiebronnen als je een bestaand framework of codebase moet leren?*

S Examples van documentatie.

Q *Examples van documentatie?*

S Ja want ik weet nog waar ik hiervoor mee heb gewerkt was Yii, da's een ander PHP framework, en het is waardeloos gedocumenteerd, maar, het is dat de examples erbij staan, die het weer goede documentatie maken. Dus dat je kan zien hoe je het gebruikt. Want wat je meestal hebt is dat je, je hebt dan een bepaalde functie daar staat dan in 1 zin bij wat ie doet, welke parameters ie nodig heeft en wat de return type is enzo. Dat is opzicht handig om te hebben, maar dat kan je ook wel uit de code halen. Wat daarentegen wel heel handig was aan die documentatie pagina is dat er bij staat hoe je het in daadwerkelijke situaties zou gebruiken. Dat heb je hier totaal bij bunq heb je dat totaal niet. Als jij dan — het is meer dat het consistentere codebase is waardoor als je het een keer weet je voor de rest ook weet. Maar, laat ik zo zeggen, als jij hier je eerste dag doet dan is het echt handig om een aantal van die examples te hebben.

Q *Ja, helder, had je verder nog opmerkingen of vragen?*

S Nee.

C.12 Subject 12

Questioner (Q) *Even kijken, waar zitten we nu, in de .. —*

Subject (S) [redacted] view response.

Q *Yes, en hoe had je dit bestand gevonden?*

S Eh, op die endpoint gezocht. Hier staat die endpoint. Dus gewoon eh.. —

Q *PhpStorm search? Oke, cool. Dus daar heb je hem toegevoegd.*

S Yes, en dan natuurlijk in dat eigen bestand zit dan de request en response pre en post processor. Dan heb je natuurlijk die nieuwe generated field. Dus omdat, ik heb ook meteen de model opgezocht. Want die staat ook hierin. Hierboven staat welke model je gebruikt [in de definition file]. Dus pak je die model, en kijk je in dat model. Daar staat dus een relatie met RequestInquiry. En daarin zit dus de data wie is de request aan het opvragen. Dus dan ging ik naar die Model, RequestInquiryModel, en daarin staat createdUser — [interruption by colleague] — ja dus in de RequestInquiry staat meestal gegevens van de persoon of bedrijf die een request aanmaakt. In RequestInquiry zelf staat alleen maar de counter details, maar die extend dus de RequestModel en daarin staat wel mooi wie de request heeft aangemaakt en eventueel merchant gegevens. Dus wie de Request heeft aangemaakt, en daarvandaan pak je de naam van de user die de request aanmaakt. Dus wie wil, — aan wie het betaald moet worden. [Navigates to [redacted]Preprocessor] Dus ja, dus uiteindelijk wat ik dan doe is als er RequestInquiry is, want het is optional, doen we null, zo niet dan doen we de naam van de createdUser pakken. En voor nu heb ik even testdata, maakt niet zoveel uit.

Q *Nee, exactly. Oke helder. Dit heb je nu — ik zag je helemaal door alle relaties heen vliegen, dat was omdat je de backend kende of heb je dat op een andere manier gevonden?*

S Nou, toevallig weet ik dat alles in JSON staat en dat staat ook hier ergens [assignment text]. Dus, het lijkt me logisch dat informatie die hieruit — dat de informatie die gebruikt wordt om bodies te maken uit modellen komt. En ik zag dat, deze view processor, deze view, deze model gebruikt. En van daar dus ging ik zoeken van waar slaat — kan ik de data uit deze model pakken. In dit specifieke model staat het niet wat ik nodig heb. Maar ik zie wel dat ie relaties heeft dus dan ga je zo uiteindelijk door totdat ik de data die ik nodig heb heb.

Q *Oke, cool. Dus je hebt de tool nu in eerste instantie niet gebruikt bij deze assignment, niet nodig gehad?*

S Nee, eigenlijk niet nee. Maar ik zou ook niet weten waarvoor ik deze tool zou gebruiken om dat te zien. Dus, de usage voor in mijn geval zou ik doen als ik iets moet — als ik weet dat in feature balblabla is dit geïntroduceerd en ik wil backtracen wat — .. ja dan had ik dus ook dit kunnen gebruiken. Nee, ik heb de tool uiteindelijk niet gebruikt dus nee.

Q *Nee, dat maakt niet uit, maar dan probeer ik uit te vissen van waarom je die dan niet hebt gebruikt want dan was er blijkbaar in de editor iets of je wist het al of..*

S Ja, precies. Editor navigatie was goed genoeg — of mooi genoeg om dit specifiek gegevens terug te pakken.

Q *Ja, maar er zit ook wel een deel in dat je weet waar je het kan vinden — want je jumped best wel snel — je weet er staat een Model bij, dat staat dicht bij elkaar —*

S Want wat ik in de models dus ook niet zie is bijvoorbeeld, ik zie wel dat er een model is maar ik kan niks meer zien. Bijvoorbeeld de relaties zou ik niet hieruit kunnen pakken.

Q *Nee precies.*

S Maar met de workflows is het wel een stuk handiger — hier zie ik wel alles mooi wat er allemaal gedaan wordt. Maar zoiets met de model zou misschien wel sneller geweest zijn dan springen springen springen. Als het model dan mooi uitgewerkt wordt in relaties zeg maar. Dus dat zou wel missen als ik deze tool had gebruikt. Dan had ik nog steeds toch in de IDE moeten gaan zoeken naar uiteindelijk alle references.

Q *Dat is nuttige feedback. Cool. Dan stel ik voor dat je met die volgende verder gaat. Probeer nog even een beetje of het ook met de tool wil lukken, en als je zegt het is gewoon echt rommel daar kan ik gewoon echt niks mee, dat is ook prima, dan ben ik erg benieuwd waarom natuurlijk. Dus, als je het wilt proberen met de tool dan graag. En dan spreken we zo nog wel even verder over die eerste assignment.*

-

Q *Oke, vertel.*

S Dus, punt drie was die AliPay gebeuren. Ik heb de tool deze keer wel gebruikt. Best wel beter. Dus ik heb op [redacted] tag gezocht. En wat ik eerst gedaan heb is alle views kijken wat hier gebruikt wordt.

Q *Is er een specifieke reden dat je eerst naar de views bent gaan kijken?*

S Ja, want ik wilde kijken wat binnen en wat uit komt zeg maar. En wat ik snel kon zien was dat wat er binnenkomt is details over de betaling, dus merchant, amount, dus gewoon de details van de betaling. En wat eruit komt is uiteindelijk authenticatie gegevens voor de user. Dus een manier om de user te authenticeren of de betaling accepteren zegmaar. Dus dan weet ik wat erin en eruit moet, dan weet ik dus ook wat ik moet opslaan en wat uiteindelijk getriggered moet worden. Dus wat het dan wordt is een initial view. Dat dan de merchants kunnen doen om de transaction flow te kunnen initiaten. Dus dat wordt een of andere endpoint, weet ik veel, ali-pay-initiate ofzo. Dan sturen hun dus dan heel moderne JSON de amount, merchant id, en wat references voor zichzelf. Dat wordt dan opgeslagen in een model. Uiteindelijk pakt dus de workflow of de controller in dit geval dit op. Dan gaat dus deze model op een of andere pending status zetten waardoor dit dan uiteindelijk opgepikt wordt. Uiteindelijk dus dat wordt op Pending gezet, dan gaan we een al bestaande workflow afkicken. Bijvoorbeeld, een user authenticate workflow. Als ik het goed zag zat hier iets van TokenQr, ja. Dus dan gaan we — dus zo'n flow bestaat al dus die kunnen we grotendeels kopiëren en nabootsen. Dan wordt dus dat — dat komt er dan dus uit in principe, UUID en de base64 van dat ene QR code. Dat de merchant dan in de webpagina kan laten zien. Uiteindelijk wordt dat dan in de app opgepakt. Er wordt een PUT gedaan met — om de status van dat ene ding te veranderen zodat het laat zien dat het user heeft geauthenticate. Dan in de app komt er dan te zien Request Details. Dat dan in dat ene model is opgeslagen. Nou, in dat model wordt de RequestInquiry opgeslagen, en die wordt dan weer geshowd in de app via references. Of via de relatie zeg maar. Accept of Deny. Dan trappen we ook weer de bestaande RequestInquiry accept of Deny — of Update workflow, ik weet niet meer precies hoe ie heet, maar er bestaat al een workflow die RequestInquiry accepteert of denied. Dus dan kicken we die gewoon af. En dan gebaseerd op de 2 paren als het geaccepteerd wordt dan update deze model ook zodanig dat het geaccepteerd wordt en dan geven we de merchant ook meteen response van 'het is geaccepteerd'. Zo niet, dan wordt de request ook weer gewoon gedenied. En deze model wordt dan ook op denied gezet en dan kan de merchant ook meteen zien — of returned naar de merchant ook meteen denied en dan kunnen hun dus gebaseerd op accepted of denied beslissing nemen over wat ze uiteindelijk gaan doen. Dus ik denk dat dat de flow gaat worden.

Q *Heb je — de webshop die polled in Principe onze backend —*

S Ja, dat is dus gewoon dat ene model. Alleen ik heb geen specifieke endpoint er voor gemaakt. Maar dan is er natuurlijk een specifieke endpoint voor. Maar ja, die deze model returned en daarmee kunnen ze dan ook zien als deze op status accept wordt, doe dat, op status reject doe dat en dat. Dus basically wanneer de user een accept or deny doet in de app dan worden 2 dingen dus gedaan. De RequestInquiry bestaande workflow wordt gekicked om deny te doen en deze ene model die hun gaan pollen wordt dan ook op accept of denied gedaan.

Q *Ja, exactly.*

S Dus dan heb je een soort van duplicate data eigenlijk. Bij wijze van spreken.

Q *Ja en nee. Want er zou misschien ook transaction modellen kunnen zijn die je niet perse met een request hoeft te accepteren in de toekomst. Omdat die op een andere manier worden geaccepteerd. —*

S Fair enough.

Q *Dus ik denk dat het gewoon een manier is en je kunt nu heel duidelijk zien in RequestInquiry van 'he wanneer is die assigned en accepted/rejected' —*

S Fair enough.

Q *En wat ook kan, een Request van zichzelf expired niet automatisch geloof ik. Terwijl je dat hier wel weer makkelijk in kan zetten. En als je dan nog een RequestInquiry accept waarvan de transaction die er al aan vast zit dan expired is kan je weer een error gooien van he —*

S Mja, oke, dus je kan in principe hier wel meer details aan toevoegen inderdaad.

Q *Oke, had je verder nog dingen gezien of — niet dat het moet hoor maar, ik weet niet of je al klaar was met je verhaal?*

S Nee, ja, ja wat ik — alleen wat ik aan de tool gemerkt heb in deze proces dan is — de workflows kan ik wel mooi zien wat er een beetje gedaan wordt. En de namen zijn goed genoeg om te beschrijven wat — zonder de method code te zien wat er uiteindelijk gedaan zou moeten worden. Alleen de model en views zijn een beetje kaal, dus daarvan moest ik wel steeds switchen tussen IDE en tool. Dus als dat switchen op een of andere manier of verminderd wordt of smoother gaat zeg maar, dan is het makkelijker om deze tools samen te gebruiken. En zo niet kan je ook alleen maar de IDE gebruiken. Want dit kan je nog niet alleen gebruiken. Nog niet.

Q *Stel dat je hem alleen zou willen gebruiken, of jou ideale tool, hoe zou ik dat dan — hoe zou je dat dan voor je willen zien? Visualisatie van de workflow bijvoorbeeld, maar wat zou jij dan graag willen zien?*

S Wat mooi zou zijn is als je zoiets [graph of workflow] krijgt maar dan van heel het proces. Dus een of ander request komt binnen, en dan zie je bijvoorbeeld en dan zie je bijvoorbeeld request — eh, je had het ook hier ergens heel mooi eigenlijk, zoiets [schema in assignment text]. Dus Router misschien hoeft niet, dat is te — dan begin je bijvoorbeeld bij preprocessor ofzo en dan kan je heel mooi zien de view data, want die heb je al, dus kan je mooi met een clickable JSON kan je de view misschien expanden. Dus dan weet je wat er in komt, en dan kan je in een workflow ook zien wat in een workflow — of welke workflow hoort bij deze preprocessor, kan je ook zien wat er in de workflow erin gestopt moet, want nu kan je niet zien wat in de workflow komt. Dus kan je ook meteen zien wat dus van de preprocessor uiteindelijk in de workflow gaat, en dan zie je uiteindelijk dit van de workflow, wat er gedaan wordt, wat er uit de workflow komt en wat dus weer in de postprocessor geduwd wordt en wat de postprocessor weer eruit gooit. Dus zo'n hele flow zou een manier zijn en ook misschien een model die hoort bij deze flow, want de pre en post processors gebruiken meestal een model, of een main object die uit — meestal uit de Create Read Update workflows komt een — of een model of een object met meerdere models. Dus als dat object of model ook gedefinieerd wordt dat zou ook chill zijn. Of, alles clickable maken zoals de workflow. Maar ik denk dat de hele flow misschien handiger is

Q *Dat je alles in ene keer ziet?*

S Ja en dat je dan gewoon mooi erin kan schuiven of zo. Want alles op een site is misschien niet want dan is het misschien net te klein, maar dat het een soort playground wordt dat je gewoon — dat je gewoon op een mooie manier alles wat erin komt, al het proces en alles wat eruit komt. Want dan zou je bijvoorbeeld opdracht 3 een stuk makkelijker kunnen doen, want dan doe je bijvoorbeeld alleen maar de [redacted] flow open, en kan je meteen zien wat mensen al sturen voor [redacted], wat in een workflow gaat, wat uit een workflow komt, wat hun dan weer als response krijgen. En dan kan je misschien voor de voorbeeld ook meteen de polling flow van [redacted] pakken, en dan zie je ook weer wat voor request ze doen, wat in de workflow gebeurd, wat eruit komt, wat weer in de response eruit komt.

Q *Ja I see what you mean.*

S En dan hoef je dus minder te denken aan wat zou de merchant moeten — of dan hoef je niet te switchen in IDE en tool.

Q *Ja precies, want je moet nu omdat je de models en de views eigenlijk geen informatie verder van hebt moet je nog steeds.. —*

S Ja, heen en weer.

Q *Oke, helder. Verder had je hem denk ik heel erg compleet. Ik heb eigenlijk niks dat ik dacht van dat heb je echt gemist. Cool. Dan had ik nog een vragenlijstje voor achteraf. Wat vond je van de opdracht?*

S Deze opdracht? Op zich prima. Wel te doen. Alleen, ja ik weet niet of — ik heb de tool niet — ik heb het gevoel dat ik je tool niet goed gebruikt heb of goed kon gebruiken voor deze opdracht.

Q *En waar komt dat mostly door dan?*

S Ja, dus puur door lack of information voor model en view. Ja ik weet niet of je misschien iets in de libs wilt doen, maar ik weet niet — maybe de method namen in de lib of zo, maar deze drie zijn cruciaal zeg maar, en ik kan alleen maar wat zien van controllers, maar controllers is ook wat er in en uit komt. Maar de main factor is gewoon een mis — informatie van de drie most used classes.

Q *Ja precies, het zijn echt de drie main concepts in de backend eigenlijk. Oke, fair. De Find By Label die heb je uiteindelijk wel gebruikt nu.*

S Ja, ik wilde alles van [redacted], want — een andere manier zou zijn om 1 commit te pakken maar dan zie je niet meteen alles over [redacted]. En als GitLab niet alles mooi gelabeled heeft dan kan je ook weer van alles missen. Dus ik dacht, misschien de [redacted] label zou mijn best bet zijn om alles van [redacted] te krijgen. Ik zag geen andere manier om alles te krijgen voor een bepaalde tag of feature te pakken.

Q *Dit scherm, we hebben het net gehad over dat die informatie mist bij model en view, zitten hier nog andere dingen in dat je zegt 'dat zou je hier echt goed kunnen toevoegen' of dit werkt juist goed of ..*

S Wat goed werkt is alles referencen, for sure.

Q *Alles references als in? Hoe bedoel je?*

S Dus dat via commit message dat ik alles meteen kan zien, wat in dat issue allemaal — alle commits zien die dat issue referencen. En alle files die getouched zijn. Dat ziet er best wel cool uit. Alleen ik weet niet of er een of andere reden is dat je dus ook, — is het misschien outdated of zo? Dat ik dus die ene model zag die nog niet bestond?

Q *Oh ja, als er een file deleted is in een commit dan neemt ie dat nog niet mee. Dan ziet ie het nog steeds als file touch zeg maar. Wat klopt, want het is een touch maar het is een touch waardoor ie hier weg zou moeten zijn. En ook file moves, renames, dat zou echt een ramp worden als ik dat steeds moest updaten in mijn database.*

S En het openen van de uiteindelijke GitLab issues is ook super handig, want dan kan je meteen discussies zien of zo. Wat ook cool zou zijn is misschien direct de merge requests. Want bijvoorbeeld nu zie ik in [redacted] dat [a senior developer] wat gedaan heeft, maar ik heb het gevoel dat [this developer] alleen maar gemerged heeft.

Q *Ja, dus dat zou dan meer zijn dat je wilt kunnen zien per persoon wat ze hebben gedaan?*

S Ja. In principe, ja precies. Want misschien heb alleen jij echt flink code zitten schrijven en hebben deze twee eigenlijk alleen maar op de merge knop gedrukt. Dan is het nogal hard om [a senior developer] even te poken van hey waarom heb je dit dit dit gedaan terwijl hij dat helemaal niet heeft gedaan.

Q *Oh, moet ik eerlijk zeggen, de merge requests heb ik geexclude van analysis, ik heb zeg maar git log en dan —no-merges gedaan. En dan eroverheen gelopen.*

S Dus [this developer] is in dit geval er dus wel echt —

Q *Hij heeft ergens iets, een commit — oh hij heeft volgens mij ook een* [redacted] *test ooit verbeterd. Die faalde af en toe.*

S Maar to be able om basically dit — als dit erin kan op de een of andere manier, —

Q *De annotations?*

S Ja, dus dan kan je zien — of git blame,

Q *Ik zou op zich aan de hand van de git blame kunnen laten zien van welk bestand door wie is gewijzigd door die avatars bijvoorbeeld klein hier achter te zetten, zoiets.*

S Bijvoorbeeld. Maar dat, dat mist inderdaad. Want normaal met bug hunting gebruik ik in ieder geval ook dit. Dus wie heeft er een line veranderd, dus oke dit was in dit geval [a senior developer], dus, [the developer] poken waarom is dit blablabla. En via deze tool alleen, kan ik —ik zie dat deze drie wat gedaan heeft maar ik zie niet precies wie dit kapot heeft gemaakt.

Q *Ja precies. Ja ik zit nu even te denken want je wilt dan wel — je kan namelijk blame kijken, maar het kan ook zijn dat voor een andere feature een bestand is aangeraakt, dus dat er mensen tussen komen te staan in de git blame die niet hier staan zeg maar. Dus dan zou je er ook weer een sorting in moeten zetten dat deze priority krijgen over anderen en dan — je hebt de set die wel hier staan en de set die niet hier tussen zit en die kan je dan weer ranken op het aantal lines dat ze hebben bijgedragen aan een bestand of zo. Ik zit even beetje brain farts, maar dan kan je inderdaad wel laten zien van — als je heel de tijd ziet dat ik vooraan sta dan krijg je wel een impression van okee die heeft er het meest aan bijgedragen.*

S Maar dat zijn misschien twee aparte use cases, denk ik. Wie heeft deze bestand aangeraakt en wie heeft gewerkt op deze issue.

Q *Ja, exactly. Dan zou je hier meer laten zien wie met dit bestand heeft gewerkt. Ja, oh wacht, ik snap hoe je hem bedoeld. Ja, ik trek het meer uit elkaar dan — ik laat ze allebei naast elkaar zien, ik laat enerzijds zien wie heeft er dit bestand aangeraakt maar vooraan zet ik degene die het hebben aangeraakt terwijl ze aan het issue hebben gewerkt. Maar je zoekt in principe op een feature niet noodzakelijk issue.*

S Ja, maar deze mensen hebben op de issue gewerkt, op deze tag gewerkt zeg maar. Maar hierin zou ik bijvoorbeeld ook [a senior developer] kunnen zitten die aan [redacted] heeft gewerkt.

Q *Ja precies, als ik iets van iDeal of iets generieks aanraak, ja. Oke, ja dat is een hele goede inderdaad.*

S Maar dat zijn dus mijn drie, of mijn paar main feedback over de tool.

Q *Yes, had je de find by developer nog gebruikt?*

S Yes, ik was curious wat ik gedaan had en ik zag alleen deze opkomen. Alleen dit vind ik wel een beetje misleading want hier staat dus Labels waar ik op gewerkt heb, alleen ik zie dus niet wat ik op deze label gedaan heb. Dat is wel een beetje..

Q *Ja, het is ook een heel groot label 'Cards', [some mumbling about other labels]*

S Dus dit is wel heel leuk maar ik weet niet precies —

Q *Ja en dan krijg je dus dat je wilt filteren, dan zou het dus helemaal nice zijn als je dan eventueel erachter laat zien wie eraan gewerkt hebben dat je boven nog kan filteren 'toon alleen bestanden'*

S Precies.

Q *Ja, oke, er komen allemaal ideetjes op nu. Leuk is dat, thesis is bijna voorbij en nu krijg ik ideeën. Oke, cool. Ja, helder. En wat voor informatie is voor jou met name nuttig als je zoekt op die developer of zeg je van wat kan hier nog bij, wat zou je helpen?*

S Misschien most recent changes? Qua bestanden. Nee maar dat is misschien weer dat git blame gebeuren. Want dat heeft dus ook niet perse met een issue te maken.

Q *Nee, dat is gewoon globaal. En dan kan ik de git log op zich wel gewoon de top 10 — of de laatste 10 commits pakken voor jou mailadres, —*

S Bijvoorbeeld.

Q *Dat kan.*

S Dus ja, ik vind het profiel een beetje kaal met alle data die je beschikbaar hebt. Dus je kant bij wijze van spreken ook laten zien hoe actief — dus een beetje statistiek, wat voor de meest gewerkte project of de meest, project waar de meeste commits in zaten of de meeste tijd in zat of zo. Of een paar recente bestanden.

Q *En hoe zouden die dingen, bijvoorbeeld recente commits of activity, hoe zou jou dat helpen als developer om dingen te kunnen begrijpen bijvoorbeeld? Of heeft het een ander doel?*

S Om dingen te begrijpen, misschien niet echt, ik denk dat dat statistiek geen nut heeft voor bug tracking of feature tracking. Misschien is het meer of een nice to have. Want je hebt zoveel data beschikbaar bij wijze van spreken uit de git repo dus je kan best wel een rich user profile maken qua stats. Dus waar heeft ie op gewerkt, recente commits, etc.

Q *Oke, clear. Had je de Find by CIT nog gebruikt?*

S Ja, ook. Totdat ik gewoon random — oh deze heeft niet eens eentje..

Q *Nee ja, er zitten echt heel veel pollution in de database. Dat is waarom ik dat script had geschreven dat commit messages valideert. Maar niemand die had hem geïnstalleerd. Dat was ook een beetje jammer.*

S Ik had, omdat ik wist dat ik recent op Insolvency gewerkt had, had ik wel deze gebruikt. Wat ik wel heel cool vond is dat copy paste heel soepel gaat, dus ik kan hier gewoon paste doen en het vult gewoon alles in, dat is wel echt supermooi. Dus ik heb hem wel gebruikt in exploring zeg maar maar niet gebruikt voor opdracht 2 of 3. Puur opdracht 1.

Q *Oke, en zie je hier wel een use-case voor of zeg je van naja dit zou ik echt nooit gebruiken?*

S Ja, voor bug hunting, dus dan weet ik — net zoals die ene voorbeeld over de API key die niet expired — ging ik in de annotations zien wie voor het laatst aangeraakt heeft, dan de issue reference gepakt en dan alle files kijken en dat die issue reference tevoorschijn kwamen. En dan zo kijken in welke files is het meest logische dat deze bug in zit. Er was een workflow die API key gaat activeren dan is het hoogstwaarschijnlijk dat daarin dus invalidatie mist of het geactiveerd of expired is of niet. Dus zoeken bij CIT zou ik sowieso wel gebruiken om bug hunting of om curious te zijn van wat is er allemaal veranderd in deze feature? Ik denk dat dat de meest gebruikte feature is, voor mij dan.

Q *Oke, en zou je dan nog meer informatie nodig hebben of heb je dan informatie je dan nodig hebt in beeld nu?*

S Naast de missing van models en fields op zich niet echt. Ik denk dat dit wel genoeg is. Dus naast wie wat heeft gedaan en de models en de views, die hele flow, is het op zich wel genoeg. Ik begrijp alleen niet nog niet heel erg wat dit is..

Q *Dat zijn alle issues die weer via dit issue related zijn, die bijvoorbeeld dit issue getagged hebben of als er ooit een commit is die twee issues heeft getagged, I don't know.*

S Naast dat, dan is het inderdaad wel genoeg.

Q *Oke, cool. Tja, de voordelen van de tool die heb je inmiddels al flink genoemd, wat er anders moet denk ik ook wel, ik weet niet of je nog andere toevoegingen had dan wat je al had genoemd?*

S Nee, ik denk het niet nee.

Q *Cool. Nog feedback in general over de tool die je nog niet hebt genoemd?*

S Nee.

Q *Je indruk van de bunq backend als je die ziet, de backend, wat is dan jou indruk daarvan?*

S Het heeft een terugkomende structuur, of design pattern wat je 't wilt noemen. Die best wel duidelijk is. Je kan heel de code doorlezen best wel makkelijk. Dus, als je iets wilt uitzoeken, bijvoorbeeld ik wil weten hoe wordt [redacted] geprocessed zeg maar, als je zo'n workflow definition open doet dan is het heel makkelijk te begrijpen. Wat er gebeurt, wat de flow is. Dus code lezen is op zich, gaat prima, er gebeurt niet te veel — er zit niet heel veel clever coding in. Dus dat je een one-liner if hebt of wat PHP magic, enzovoort. Wat dat betreft is het wel goed te begrijpen. Ik weet niet meer wat ik meer qua indruk heb. Nou, als ik dus wel de view kijkt, zonder te weten wat er gebeurt — ik weet niet wat er gebeurt voor een view en wanneer een view eindigt, maar de view op zichzelf is — ja, ik weet niet, er zit een soort blind spot. Dus ik kan alleen maar zien wat de view zelf doet maar ik heb geen idee wat er voor de view of na de view gebeurd. Dat is een soort blank, heel veel in de backend is wel blank space. Dus ik heb een workflow net aangemaakt, of voor opdracht 3 aangemaakt, en ik heb geen idee hoe die workflow überhaupt gecalled wordt door die controller — of door die pre-processor. Dus die pre-processor die returned een request, op ene of andere magische manier komt het in een workflow terecht, die workflow returned some output, op een of andere magic manier komt het in de post processor terecht. Post-processor doet return response, en op een of andere magic manier komt er dus gewoon uiteindelijk in de request.

Q *Maar dat is ook wat je ziet met het endpoint, die naam, van de definition, daar wordt Workflow voor geplakt, en afhankelijk van de request method Create, Read, whatever en dan probeert ie gewoon die workflow te instantiaten. Dat is de magic.*

S Ja, precies. Ik weet ook inmiddels wel hoe het inderdaad uiteindelijk gaat, maar als eerste indruk, dat is dus van de backend. Dus de code is wel mooi gestructureerd, er komt een mooie terugkomende pattern, code lezen gaat op zich best makkelijk, alleen er is wat black holes wat betreft waar request doet return request op de een of andere manier komt het in de workflow, workflow doet return blabla. Dus dat waren zo wel mijn eerste indrukken over de backend.

Q *Cool. Denk je dat deze tool een senior developer gedeeltelijk of geheel zou kunnen vervangen als een beetje orakel voor de juniors?*

S Ik weet niet, want de meeste keren dat je naar een senior stapt is om te vragen van 'hoe zou ik het best abc implementeren' of 'hoe zou ik het best dit aanpakken' of 'ik heb een structuur conflict' etc etc. Ik denk niet als je die meeste vragen alleen kan beantwoorden. Dus ik denk dat je toch input zal moeten hebben van een senior. Want ik denk niet dat ie het geheel kan vervangen. Ik denk wel dat er sommige vragen

vervangen kunnen worden, bijvoorbeeld als je voorbeeld wilt hebben hoe iets gedaan moet worden. Bijvoorbeeld je moet een transactie flow maken, dan hoef je niet meer te vragen 'yo, heb je een voorbeeld van hoe kan ik zoiets bouwen'? Dan weet je dus dat je iDeal transactions hebt, maar je hebt ook [redacted], dus dan kan je hier heel makkelijk alles over [redacted] of alles over iDeal vinden en doornemen. Dus voor zulke requests, voor sommige vragen kunnen seniors wel vervangen worden, maar ik denk niet dat ze in hun geheel vervangen kunnen worden. Want sommige vragen heb je toch echt input nodig. Bijvoorbeeld structuurproblemen of de beste manier om abc te doen, voor zulke vragen.

Q *Oke, helder. Wat zijn volgens jou de meest belangrijke informatiebronnen als je een groot codebase gaat leren, als je daarmee gaat leren werken, grote codebase of een groot framework? En waarom?*

S De beste leerbron, dat is een goeie. Mijn ervaring is, ik heb eigenlijk gewoon als ik wist dat ik iets moest doen dan ging ik op zoek naar een stukje code die ongeveer hetzelfde aan het doen is. Of in de goede richting duwt en dat stukje code lezen en dan proberen zo goed mogelijk na te doen. En dan natuurlijk met een paar details veranderen. Dus dan een goede manier om — of een goede bron zou dan zijn om makkelijk een relatie te kunnen vinden tussen dingen die bijna hetzelfde zijn. Dus ik zag bijvoorbeeld met [redacted] dat daar ook RequestInquiry in zat en RequestInquiry zit ook weer in iDeal en zo heb je een hele grote hierarchy zeg maar. Maar zo'n bron bestaat niet. Dus dat is allemaal zelf in de IDE beetje springen springen zoeken voor iets wat er op lijkt. Dus een ideale bron zou zijn als je een soort internal wiki hebt waar gewoon alle relaties en alle flows een beetje gevisualiseerd of geshowd worden.

Q *Een soortement documentatie, visuele documentatie dan.*

S Ja zoiets, want alleen maar tekst is misschien niet heel handig, —

Q *En die moet je ook weer updaten.*

S Ja precies. Dus gewoon — ik zie wel potentie voor bijvoorbeeld deze tool om dat te kunnen bereiken. Dus als je heel makkelijk kan zien dat RequestInquiry wordt gebruikt door [redacted] iDeal en je moet dus ook zoiets extra gaan bouwen dan kan je dus heel makkelijk vanaf dat andere zien en gebruiken en kopieren om jou AliPay te implementeren, bijvoorbeeld. Dus,tot nu toe is er niet echt een goeie bron voor zo'n grote codebase. Anders dan de huidige code lezen en zoeken wat iets er op lijkt en een beetje na te doen.

Q *Maar goed, voor jou werken de examples bijvoorbeeld, die noemde je, die werken normaal wel goed om zoiets te leren. Dus dat is inderdaad wel een goeie bron. Okay, cool. Had jij zelf nog vragen voor mij?*

S Wat determined of dit een faal of een success is?

Q *[...]*

C.13 Subject 13

Questioner (Q) *Okay, vertel.*

Subject (S) Nou ik heb de workflows of in ieder geval de class namen bepaald die ik denk nodig te hebben. Dat zijn voor de controllers allemaal WorkflowAliPayIssuerTransaction en dan vervolgens Create Read Update Status and ProcessUpdate. Ik heb nagedacht over de models AliPayIssuerTransactionModel en een Message model. Verder heb ik een router nodig die nou ja, route. En een daemon die transactions continue checked.

Q *Yes. En kan je een beetje toelichten wat je waarvoor nodig hebt? Wat de rollen zijn van de verschillende onderdelen.*

S Ja, de workflow create die gaat als het ware een transactie initialiseren en daar een QR code voor genereren. Die vervolgens door de Read endpoint kan worden opgehaald later, door de webshop.Nou de update gebeurd volgens mij als je daadwerkelijk dat ding gaat scannen, gaat betalen, zodat ie dat status veranderd in een accepted transactie. Met de status kan de webshop pollen of ie al betaald ie. En ProcessUpdate weet ik niet helemaal zeker.

Q *Okay, cool. —*

S Eigenlijk heel simpel en heel dom nagedacht. Ik heb gewoon gekeken naar de iDeal [redacted] implementaties. En die naast elkaar gelegd en gekeken van 'wat zijn de overeenkomsten tussen die 2'. En op basis daarvan heb ik eigenlijk dit opgesteld.

Q *En hoe heb je dat naast elkaar houden gedaan?*

S Ik was eerst van plan om het met die tool heel slim te doen. Om gewoon die hiërarchisch, twee aparte plops te zetten en alle overeenkomsten tussen die twee maar ik merkte dat de iDeal issuer dat dat gedeelte via de tool niet heel goed in te zien was. Omdat ie niet goed gelabeld was binnen GitLab. Dus uiteindelijk heb ik het iDeal verhaal gewoon in PhpStorm gekeken met een zoekactie. Maar wel de tool gebruikt voor [redacted]. Ja en dan eigenlijk gewoon daar doorheen gestepped. En alle gelijkenissen opgeschreven. Ja, heel dom nagedacht maar —

Q *Ja, nee, maar dat was op zich wel het idee dat je gewoon kon baseren op een andere feature, dat je die goed kon doorgronden en dan die kennis toepassen in een nieuwe flow. Waar je eigenlijk niet veel bij hoeft na te denken maar dat je wel makkelijk ziet wat je nodig hebt.*

S Ja, dat is inderdaad wel hoe ik zo tot hier gekomen ben.

Q *Oke, cool. Dan heb ik wat vragen over de tool zelf, wat vond je van de assignment?*

S Echt van de assignment?

Q *Ja.*

S Nou oke. En dan heb je het over de design opdracht, part 3?

Q *Het geheel. Het hele experiment.*

S Allright. Ik vond het leuk om te doen. Interessant om na te denken over hoe dit soort dingen daadwerkelijk geïmplementeerd zijn. Ik heb er zelf nooit echt eerder mee te maken mee gehad. Maar inderdaad wel cool om het een beetje op design level te bekijken. En de code assignment is redelijk een routine klusje als je een tijdje bij bunq hebt gezeten. Maar het was geinig om die modellen weer een keertje door te spitten.

Q *Ja, je had de tool wel een beetje gebruikt right? D'r zitten drie features eigenlijk een beetje in.*

S Label, developer en issue.

Q *Yes, exactly. De bovenste heb je die kunnen gebruiken.*

S Ja.

Q *En waarvoor en waarom heb je die kunnen gebruiken?*

S Om alles dat bij het [redacted] verhaal komt kijken te checken. Daar was het heel nuttig voor. Om inderdaad te kijken welke classes involved zijn. En dat was heel makkelijk omdat gewoon goed gelabeld was. Ja, zoals ik net al zei dat iDeal gedeelte was niet zo chill gelabeld. Dus dat was — hier zie je het eigenlijk al wel.

Q *Ja, exactly.*

S Dus daar had ik er niet zo heel veel aan.

Q *Ja, maar je had het op GitLab opgezocht dat het aan die labels lag of..?*

S Naja, ik merk gewoon — ik zoek op iDeal en dan zie ik wel een paar opties, naja, ik had dan — ik zocht op iDeal. Dan vindt je deposit, dat is het sowieso niet, naja topup ook niet, dus ik dacht outgoing, heb ik daarop gezocht. Toen zag ik twee klassen. Toen zag ik al direct dat dit niet is wat ik hebben moest. Dat ik hier niet zoveel aan zou hebben.

Q *Nee precies.*

S En dan heb ik wel overwogen om zo te zoeken op [a senior developer] zijn contributions, maar naja ik heb het niet eens geprobeerd. Ik weet vrij zeker dat het een flinke lijst is. [searches] Dan vind ik in principe weer diezelfde labels die ik anders ook — hoewel, oke, dit helpt wel iets meer.

Q *Wat was de reden dat je in eerste instantie niet op [a senior developer]s naam had gezocht meer?*

S Omdat ik er vanuit ging dat ik dan zo'n gigantische lading met klassen zou krijgen dat er weinig aan te doorgronden was. Maar ik had me eigenlijk gerealiseerd dat ik dan nog op Labels kon kijken. Dus dat is stom.

Q *Dat is niet stom, —*

S In ieder geval, ja.

Q *En wat was de meest nuttige informatie, want je hebt gewerkt met verschillende informatie dingen, wat was het meest nuttig voor jou?*

S Überhaupt aan het gebruik van?

Q *In dit scherm waar je nu gezocht hebt op een label. Welk van de informatiedingen die getoond worden is voor jou het meest relevant nu.*

S De klassen en met name de UML grafieken daarbij, ik vond het heel nuttig om die flow gevisualiseerd te zien. Wat ik vroeger wel regelmatig deed was in die JSON definitie checken hoe dat ging, maar dan moet je ook echt gaan nadenken van 'oh als deze conditie dat is dan springt ie daar heen en anders dan gaat ie daar heen'. En dan moet je eigenlijk heel veel continue in je hoofd houden van 'oh ja ik zit nu in die flow'. Terwijl die visualisatie zegt van 'oh als dat zo is dan gaat ie daarheen en ..' Veel easier.

Q *Ja, oke, helder. De Find By Developer, die heb je wel gebruikt of naja, je wilde [a senior developer] zoeken maar.. —*

S Nee, die heb ik in principe niet gebruikt.

Q *En dat was dan omdat je bang was dat je te veel informatie zou krijgen?*

S Ja ja. Of in ieder geval niet specifiek genoeg.

Q *Ja, precies. De Find By CIT had je die nog gebruikt?*

S Nee. Ik dacht ik kan die wel gaan gebruiken maar dan zou ik waarschijnlijk eerst naar GitLab gaan om dat issue te vinden. Terwijl de resultaten daarvan naar mijn idee overeen zouden komen met wat gelabeld is. Snap je wat ik bedoel.

Q *Ja.*

S Dat ik in feite eigenlijk hetzelfde zou doen maar dan via een omweg, via GitLab.

Q *Ja, en had je gezien dat als je hebt gezocht op een label bijvoorbeeld nu, en je gaat bijvoorbeeld helemaal naar beneden, dat je hiervandaan kan doorzoeken op die issues? Zeg maar als je nu hier klikt dat 'ie m dan voor je invult.*

S Ja. Had ik gezien.

Q *Dat maakte het niet makkelijker of dat je zei van nu wordt het nuttig?*

S Ik heb daar niet zo — ik heb het niet zo overwogen om het zo te doen. Ja, want ik weet dat heel veel, dat eigenlijk zo'n groot project eigenlijk nooit echt 1 issue is. Het is altijd verspreid over meerdere issues. Dus dan zou ik het idee hebben dat als ik op 1 bepaalde issue zoek dan krijg ik misschien een paar klassen maar dan weet ik niet of het echt compleet is en of ik wel alles zie dat er mee te maken heeft.

Q *Ja. Oke. Ja, helder. Zou je wel use-cases weten voor het zoeken op developer of CIT? Waar het wel nuttig zou zijn bijvoorbeeld?*

S Ja, sowieso als je met CIT kan zoeken dan weet je wie daadwerkelijk voor dat issue code heeft geïmplementeerd. Dus als je dan al weet wie — bij wie moet ik zijn voor een vraag over deze klasse — in ieder geval bij dit onderwerp, dan kan je makkelijk het issue vinden en de developer daarbij. En dan kan je daar je vraag bij neerleggen. Ja, By Developer, ja dat is nuttig als je gewoon echt wilt zien 'waar heeft iemand aan gewerkt' en dat heeft wel zijn use-cases maar ik zou niet zo snel zelf op developer zoeken als ik iets over code wil weten, want ik hoef nooit echt te weten wie waaraan werkt. Maar meer andersom. Hele vage omschrijving.

Q *Ja, precies. Je wilt weten van een Feature wie daaraan heeft gewerkt, maar je hoeft niet te weten wie aan welke features heeft gewerkt per se.*

S Aan welke features iemand heeft gewerkt nee.

Q *Exactly. Oke, helder. Wat zijn de voordelen aan het gebruik van deze tool if any?*

S Ja, ik vind dus die visualisaties, die grafiek, die vind ik heel fijn. En inderdaad, mits het in GitLab goed is gelabeld, kan je, zeker omdat je alle gerelateerde klassen ziet en daarop kan klikken om die grafieken te bekijken — kan je echt heel snel inzicht krijgen in hoe zo'n feature werkt. Zoals ik net al zei ik heb hem echt gebruikt om een high-level overview te krijgen van hoe dat [redacted] endpoint is opgebouwd. En daar vond ik dat heel nuttig voor.

Q *Dus dat werkte dan wel goed om een understanding te krijgen van hoe het in elkaar zit dan? Oke, cool. Zijn er nog dingen dat je zegt van 'dit zou echt superniche zijn om toe te voegen' of 'dit hoeft er niet in te zitten' of noem het maar, verbeterpunten zeg maar?*

S Ja, het zou ook fijn zijn als je ook echt op een klasse naam kan zoeken. Dat je zeg maar al in PhpStorm zit met code en je komt een controller of een class tegen — zoals ik net al zei, vroeger zou ik dan altijd die JSON definitie zelf gaan uitpluizen, maar nu zou het heel fijn zijn om die naam te kunnen invullen en die grafiek erbij te krijgen.

Q *Ja. Makes sense.*

S Ja misschien ook nog wel iets specifieker, of juist minder specifiek, dat je bijvoorbeeld echt op een endpoint kan zoeken. Want je hebt natuurlijk wel `core_blabla_model`, core_blabla_controller, maar dan zou het mooi zijn als je echt op die endpoint naam

kan zoeken. Zoals hier die "[redacted]-transaction". En verder nog een ander verbeterpunt, ik zei het net al, dat je bij models kan zien welke velden erin zitten. En welke relaties het heeft naar andere modellen.

Q *Ja. En even terugkomend naar je vorige punt, zoeken op endpoint. Wat verwacht je dan te zien? Wat voor informatie?*

S Dan verwacht ik in principe diezelfde result overview te zien. Maar dan alleen maar de classes die specifiek zijn voor dat endpoint. Dus, ik zal het laten zien. Dat je zoekt op [redacted]-transaction, en dat je dan deze als resultaat krijgt. Transaction Controller, Transaction Model, Transaction View.

Q *Dus echt alleen de klassen die echt involved zijn bij de actual flow en niet de dingen die eromheen hangen?*

S Precies.

Q *Oke. Cool. Had je verder nog feedback die je nog niet hebt genoemd? Dat je zegt dat wil ik ook wel kwijt?*

S Even kijken, wat gebeurd er als ik hierop klik?

Q *Dat is een Search By Developer, dan kan je zien wat ie verder heeft gemaakt.*

S Oke. Naja, dat zou ik niet zo snel gebruiken, want als ik nu echt zoek op inderdaad een label, [redacted], dan zie ik al die developers die iets met die code hebben gedaan. Maar ik kan me voorstellen dat dat heel snel een hele grote lijst wordt. Zeker omdat ie ook libraries die ie gebruikt ook naar binnen haalt. Dus ik denk niet dat dat heel veel toevoegt wat dat betreft. Want als iemand echt een zo'n heel kleine change in een ver weg liggende library heeft gedaan dan komt ie er ook tussen te staan, denk ik. —

Q *Eh, ja, hoe het nu gemaakt is — misschien wel handig om te weten — hij gebruikt de git log als een basis. Dus die commits hebben allemaal een tag naar issues toe. En ik ga gewoon over de git history heen en ik kijk voor welke commit, 'oke, wie heeft die commit gemaakt, welke bestanden zijn gewijzigd, en voor welk issue was dat?'. En op die manier koppel ik ze aan elkaar. Dus als iemand een library heel ver weg aanraakt, dan kan hij hier wel tussen komen maar alleen als die commit daadwerkelijk een commit message bevat die weer refereert aan dit issue. Dus als er voor een ander issue datzelfde bestand wordt aangeraakt, dan wordt ie niet hier toegevoegd zeg maar. Tenminste, hij stond er dan waarschijnlijk al in, maar —*

S All right.

Q *Dus als het goed is dan zou dat echt alleen developers die voor dit issue of voor de feature waarop je zoekt —*

S Ah oke, dat is inderdaad wel wat efficiënter.

Q *Ja, maar ik ben met je eens dat als je bijvoorbeeld hier zoekt op "App: Bunq", nou dan is 'ie sowieso even bezig want dat is zo generiek, en dan gaat dit inderdaad niet veel meer zeggen. [name-matching discussion].*

Q *Wat is jou indruk van de backend? Als je dat zo ziet voor je?*

S Ik zie een hoop consistentie, dat is fijn. Kleine, — je bedoelt echt de code ook of niet?

Q *Ja als je daarnaar kijkt, wat is dan het eerste dat in je opkomt, wat denk je daarvan?*

S Goede naamgevingen, veel consistentie, korte heldere functions, afwezigheid van comments —

Q *Dat is een goed of een slecht iets?*

S Dat is in de context van de bunq backend een goed iets, denk ik. Omdat inderdaad de functions en klein zijn en goede namen hebben, beschrijft 't zichzelf al en is het in mijn optiek ook niet nodig. Ja, ik vind het zelf heel prettig dat de directories echt genamed zijn op basis van welk endpoint het is, op die manier kan je vrij snel vinden welke code als het ware response maakt voor een gegeven endpoint. Ik denk dat dat de grootste dingen wel zijn.

Q *Okay cool. Denk je dat deze tool — als juniors gaan wij vaak naar seniors toe met vragen enzo. Denk je dat deze tool dat deels kan opvangen, vervangen? Of helemaal?*

S Of het dat echt vervangt weet ik niet, maar ik denk vooral dat het — want als ik weer eventjes terug denk dan zou je eerst zelf de code in duiken en als je het dan echt niet begrijpt dan ga je naar iemand toe om vragen te stellen. Maar ik denk dat het gewoon voornamelijk dat eerste deel van zelf je inlezen, dat het daar voornamelijk tijd bespaart. Maar goed, of het echt vragen volledig weg neemt, dat — ten opzichte van zeg maar de code — dat weet ik niet.

Q *Oke, dan heb ik nog een laatste vraagje; stel je moet nou in zo'n nieuw framework leren, wat zijn dan de meest belangrijke informatiebronnen voor jou in een ideale wereld zeg maar?*

S De wiki.

Q *De wiki?*

S Nee, ehm.. —

Q *Het hoeft ook niet specifiek voor de bunq backend hoor, maar gewoon als je een groot, nieuw, bestaande codebase moet leren.*

S Ah ja. Ja dingen die echt idiomatisch zijn voor dat framework, want ik denk dat de meeste mensen die bij bunq binnenkomen wel eerder in een MVC framework hebben gewerkt. Een Laravel of een Rails of zo. En, in ieder geval de bunq backend heeft best wel wat dingen die afwijken van dat pattern. Denk dat het prettig zou zijn om een

document te hebben die dat beschrijft. Ja, en misschien bij iets complexere endpoints een README of iets erin. En wat ik net al zei, comments, de afwezigheid van comments. Meestal is dat prettig. Maar er zitten ook wel eens dingen in die — waarbij ik denk van 'hier had wel even iets van een toelichting bijgekund'. Bijvoorbeeld bij een — die [redacted] Transaction Create, daar moet je echt in de code gaan lezen wat het doet. Terwijl als je er boven een comment schrijft van 'oh ja, deze initialiseert de transactie en genereert een QR code'. Dan lees je dat en weet je direct wat ie doet. Dus dat. Ja, en ik moet zeggen dat ik dit document in ieder geval dat grafiekje enzo en de manier waarop dit is uitgelegd dat ik dat al veel helderder vindt dan wat er nu — of elk geval toen ik er was — aan documentatie is.

Q *Ja, nee oke. Helder. Ik weet niet of je nog vragen had voor mij?*

S Binnen deze assignment eigenlijk niet, helder.

C.14 Subject 14

Questioner (Q) *Okay, cool, can you explain wat je hebt gedaan?*

Subject (S) Ik had eerst, want dat stond hier dat er een label was, voor waar stond het 'de feature is called [redacted]', dan zoek je naar het label [redacted]. En dan vindt je deze shit. Ik moest in een view iets aanpassen dus nou, dan ga je naar de views, dit is bunq.me, dit other shit you changed. Deze leek promising, dit is QR shit, toen ben ik naar deze gegaan toen ging ik naar de verkeerde want ik deed dit en toen kreeg ik eerst de model. Maar dit was inderdaad [redacted]-transaction, wat ook hier gezegd wordt. Dat dat het endpoint is. Toen was het simpel als add shit.

Q *Nieuw veld toevoegen ja.*

S Dan in de ViewPostProcessor shit toevoegen, en done?

Q *Yes. Had je gevonden waar je de naam van de user vandaan kon halen?*

S Ehm, ja dat dat heb ik gewoon een beetje semi op de — maar ik wist dat omdat je het hier over Request en Response had, gokte ik dat we gewoon een standaard Request gebruikten, en toen ben ik getRequestInquiry, getAnsweredByUser, leek mijn meest logische manier om daar te komen.

Q *Ah ja, fair. Ik zag alleen deze naam en dacht dat het een andere functie was, maar daarin zit de RequestInquiry.*

S Yes. Staat ie ook nog ergens anders of?

Q *Eh, RequestResponse heeft volgens mij ook een reference naar de user die Response heeft aangemaakt. Dus dan had je daarvandaan ook naar de user model kunnen gaan.*

S Maar dan had het misschien wel mooier geweest om dat RequestResponse omdat deze, dan ben je deze waarschijnlijk kwijt, want die is waarschijnlijk null, —

Q *De RequestResponse bestaat volgens mij niet zolang hij nog niet geaccepteerd is —*

S Ja, precies, dus dan heb je deze blabla, [searching in code]

Q *En hoe had je die relation naar de RequestResponse gevonden?*

S Dat is mijn automatische gotta-love-PHP autocomplete shit. [searching for getAnswered-ByUser from RequestResponse]. En dan kom je op hetzelfde uit. This should do the same and not break everything. Dus ik keek gewoon, — ik vermoedde dat er in een transaction iets van dingen zat, er zat in ieder geval geen user shit in, ik was eventjes door het model heen, en alleen de counterparty zit er hier in. En nou toen ging ik er van uit dat ik in de auto-complete waarschijnlijk die [RequestInquiry] moest hebben, terwijl die [RequestResponse] waarschijnlijk netter had geweest.

Q *Ja, het kan allebei.*

S And you end up at the same thing.

Q *Okay cool. Heb je hierbij wat gehad aan de tool?*

S Eh, het was wel makkelijk om gewoon in 1x hier uit te komen, gewoon in 1x dit te kunnen doen. Anders had ik waarschijnlijk — ik had er denk ik ook wel gekomen. Want op zich [types [redacted]-. . .] dus uiteindelijk kom je er ook wel. Maar, op zich is het wel makkelijk om gewoon in 1x keer te kunnen zien van 'oh hey, dit is hoe het ..'. Hoewel die soms denkt dat er niks in zit. Maar het is op zich wel nice om het in 1x zo te zien.

Q *Okay, cool. Dan zou ik zeggen, begin aan de derde, daarvoor hoef je niks te implementeren.*

S Ik mag schrijven toch?

Q *Ja je mag schrijven, je mag ook typen op de computer als je dat fijner vindt, dat maakt me niet zo veel uit.*

-

Q *Cool, can you explain what you have done?*

S Well, again I looked in the [redacted] [redacted] thing because I thought that one was the most up to date and I don't think we have this nicely for the iDeal one. Because that's way way back when we weren't this strict with labels for new features and shit. So we need a model to actually hold the transaction information, something similar to this one. So you create that with all the shit that you need. But that's unknown so we'll just glance over that I guess. We need a new TokenQrRequest type. Which is — you actually extended it with [redacted], so this would add — [some searching] —

which would give you a different type of request. Which would also, you'll also have to create, I forgot one view, you need to create one view for — but I'll come to that later, then you need to add a controller to actually create the thing. Which is actually accepting. [searches something]. Okay, wait, let's go back, so okay.

Q *You added the type?*

S Yes, we need to adjust this amazing file with the avatar, like the same you did for — this one is for iDeal, [redacted] — we would also need one then for AliPay, I found that one through your thingy as well. But okay, you changed that, what did you change there and see what I need to change there. Some aren't applicable because you probably split some things up which I don't need to do anymore because —

Q *It's already split up now.*

S So that's some convenience for me. We need the controllers, we need pretty much similar to this, we'll need the workflows but then for the AliPay one. This is used for splitting it up I assumed. That's done. What else did I do. Let's go to this. We need to create a new view for the AliPay shit. Which will be similar to what you have but you don't know what AliPay actually wants you to have or will send so it will be similar but different. And we need a new endpoint, which was here, which wasn't there, [searching..] Because we don't want [redacted] to actually do API calls with their user or whatever, they just need to be able to create the request, and I think — I see some certificate shit here, which you'll probably have to fix as well but I think those are more like details.

Q *Yes exactly for* [redacted] *we needed to have like a secure channel. So we implemented it with the certificates. Might be different for instance for iDeal. Yeah, cool. That covered I think pretty much everything.*

S Yeah right? Only, what am I missing here? [some mumbling] So this one is called through this one.

Q *This is TokenQrRequest*[redacted]*Create.*

S This one posts the QR, agreed? This is API actually — this is when you scan the QR code.

Q *Yes, exactly. And then it's assigned.*

S Does it only then create — Oh then it creates the Request of course. Because they create the transaction and then this actually creates the Request which is attached to the transaction — yeah I get it, okay. I'm not confused anymore.

Q *It makes sense in the end.*

S But I think it's pretty straight-forward, yeah.

Q *Okay, cool, then I have some more questions that I want to ask you. Did you like the assignment?*

S Yes it was fun. I think it's a bit like, you have to— are you also putting this for people who don't know bunq?

Q *Yeah. That's why there's a lot of explanation in front. You had to start at the basics basically. So for them it's already quite tough. But for instance, someone else was like 'oh I do this on the terminal blamblam' and I was like 'you should use the tool'. So that's difficult to find a balance.*

S Yes.

Q *I wanted to discuss the features in the tool. So there are three main features basically. The different filters that you have over there. The Find By Label you have used, for what reason have you used that feature?*

S For — what was this assignment again — just to get a general overview I think of certain features, of what they — what changed. So you know like where to start looking. Because sometimes it can be like 'what is this called' and you're like '[redacted]' and then you're like 'does that look like it?', 'maybe?', while here you'll have — if it's, because that's always the thing, if it's properly documented, [searched OAuth], I think this one will be less nice..

Q *It's still okay-ish.*

S Yeah, it's okay-ish. Controller, — anyway. If the project sucks so to say, if it wasn't maintained properly and some shit gets in, you won't find it or you'll be misguided anyways. But if it's nicely managed I think it's very handy to be like 'ah these are the things which touch it' and 'ah, that sounds like something I need to look into'. And then you can go from there instead of guessing what it would be named.

Q *Yes exactly. Okay. Does it meet your expectations? When you search do you expect more information, less information?*

S Ehm, no I think pretty much everything is there. Let's see. No that works. Developers are the developers that ever committed something to that — within any of the issues in the thing I assume —

Q *Exactly.*

S And [senior developers] are in everywhere because they merge shit I guess.

Q *No, I excluded all merge commits.*

S But how is [this senior developer] in here then? Oh he probably fixed something while I was off or something.

Q *Probably a test or something really small. [Another backend developer] is also in the [redacted] thing while he did, well he fixed the view documentation, we have the documentationGet, Post, that he added everywhere. That's how he ended up in there. So when you touch it then you're in there.*

S But no, I think it's shows what you want it to show.

Q *Okay cool. Have you used the find by developer feature.*

S Well I did this, which is pretty much the same thing. And then I didn't do much afterwards.

Q *It's the same thing as?*

S [Someone from Product] has misnamed my thing by calling it Project: OAuth. So it's not in here, but it will be in here I assume.

Q *Should be.*

S Yes, at the bottom. So, but yeah I didn't really use it because I knew I would be going — because for the second time [part 2] it didn't make sense because it's pretty clear here that the feature is called [redacted] so you look for the feature and not by who made it. Even though I knew it was you. And for this one I knew that I had to make something similar to the [redacted] thing. So it made sense to go over the [redacted] thing again. So I don't think there was much use in going over a developer, at least in these cases.

Q *Could you think of a case where it would be useful to use a developer search.*

S Not that much. Just mostly out of interest. Because also, from a DevOp perspective, like sometimes shit is broken and we need to know who to contact, but that's also — you'd go over the label and look for the feature and see, well it will sometimes be difficult, but if it's [redacted] which is broken — I'll poke anyone of these. I'll just call [a senior developer], '[developer] it's broken, fix it!'. That might be, for that it's too broad anyways. So I'll have to go to the wiki and find the wiki because I can never remember when you need it.

Q *You should put a shortcut here.*

S Yes. Yes! But yeah, besides that I don't think it's that much of an added value. But that might just be me not being able to think of a reason.

Q *Yeah, that's possible. The Find By CIT, have you used that one?*

S Ehm, mostly just in the part 1 I think, getting to know the feature, because I think you need to either make this — adjust this because I don't think there's much use for it in these assignments. That might be my —

Q *No yeah, I know it's steered kinda towards the Label at the moment. It was also kinda hard to come up with assignments that fit in the time.*

S [searches an incremental issue]

Q *It could also be that it's — like the PARPs are not showing data most of the time, or the references are missing in the commit messages. It's not connected to GitLab immediately. Oh that button is broken by the way. Sorry.*

S [goes to GitLab for issue that misses in tool] So you ran the tool two weeks ago right. Something like that yeah. Two-and-a-half maybe, I don't know.

Q *I did see that [an employee] renamed the Labels by now. Now it's called Feature OAuth actually.*

S Oh wait, [redacted],... Can I like [mumbling].. CIT, issues, [redacted]. Feature Cards, of course!

Q *Yeah that's a big feature.*

S Yes, because it's also everything with Cards. But this is a newer one, we also have an older one.

Q *Maybe it's closed or something?*

S Incremental 1548,—

Q *You can copy-paste the reference, it works.*

S Too late.

Q *That one [issue] is in here. Apparently.*

S So yeah. But that might be interesting. If you actually have the — go from the issue and want to look what we actually did for, what we have done. Because this is just — there's actually not much in here. Interesting. That the merge request didn't get pushed here. Or don't I have read access to those?

Q *Oh no, I don't have read access to the core repository codebase.*

S Ah okay. Because normally you have a shit ton of random commits here which don't tell you anything. So at least here you can see what I actually changed, or what we touched I guess. So yeah.

Q *Would that help you to see what was actually touched for a certain issue?*

S Sometimes it can. For example for the part 3 where you have to pretty much duplicate a feature for something else. It can be handy, but you can then either go over the labels or you can — depending on which one you get — you can go over either one. And if you — yeah, I think this is more interesting for the small bug fixes like what changed.

On the other hand those GitLab issues are most of the time better readable with only merge request and you can just open the merge request and be there as well. That might be an interesting as well by the way, because you do — to have a list of the merge requests.

Q *For an issue or for a label?*

S For an issue. Or well, either one I guess because you get pretty much the same overview right?

Q *Fair point.*

S And then just linking those to GitLab I guess, because it will be a pain to show them here. But just like 'okay these merge requests are related to this' that might be interesting.

Q *Yeah it's already somehow — well the issues are already in here so it shouldn't be hard to add—*

S Yeah you have the issue here yeah, yes.

Q *There the GitLab button does work by the way.*

S Yes I tried yes.

Q *It's just that one that's messed up. It tries to use a relative URL instead of a absolute URL, I don't know. Okay anyway. Let's proceed. If any, what were the benefits from using this tool?*

S Have a quick overview of all the files which hold a certain project. Especially for the part 3 thing in which you see 'okay, these are the files that you used or touched for this project' and then you know where to look.

Q *Okay, cool, are there also things that should've been different? Or that are missing?*

S Well the merge request thing might be a nice thing to have. But besides that it looks pretty straight-forward and handy already.

Q *Cool, any other feedback that you might have?*

S I don't like the bunq logo up there, it looks wrong.

Q *Yeah it looks squeezed, I know. [explains]. What's your impression of the bunq backend? When you see it, what do you think?*

S It's ginormous. And it's nice, but it also has its drawbacks at some point. But most of it is fairly nice.

Q *What would you call drawbacks?*

S This is your, — this is not in the code I hope?

Q *No, Experiment Participant right? That's generated.*

S Oh okay. Of course.

Q *I set a recognisable username for this laptop, so.*

S What was the question again? How do I like the backend?

Q *Yeah. Or well what are the drawbacks?*

S Well it's not drawbacks, but — it can be daunting at first sight. I mean all the controllers, all the views and all that kind of crap. It can be quite daunting. Also, because we already have a shit ton of it, but yeah, it's very powerful with all the generation it does. And like it takes a lot of shit out of your hands so yeah I do like it.

Q *Okay, so right now the junior developers often go to seniors when they have questions. Do you think that this tool can reduce that or replace it?*

S It can at least help with finding shit because that's also — that's an issue I've had as well like 'where the *** do I find this?', cause sometimes the naming is not what you'd expect it to be. Because I would always , I always get messed up with Issuer and Merchant and I always — I never know which one is which and God knows and with this one I was like 'ah it's Issuer'. So yeah. Those kinds of small things, it makes it actually easier to find shit. So at least those questions can be fixed.

Q *And what kind of questions would still require a senior's attention?*

S Well, code-related questions. Because it doesn't really show how the code works or that kinds of shit. But at least structure-wise it shows like how it works. Which is good.

Q *Okay. So what are, according to you, the most important sources of information when learning a big existing codebase or any codebase for that matter?*

S Well for our codebase the wiki can be quite handy as well. Well for working on our codebase the Junior FAQ is actually quite handy. I've been on there quite a bit as well. Mostly to reset my *** ElasticSearch. But, yeah, I think it's mostly just getting used to the structure and like how that's set up with the views, the controllers and the models. If you know that you know how the backend works and you can work from there. You have some weirdness with the Routers and shit, which most people don't actually have to touch. But yeah I think if you would just have a basic understanding, most of it will like show itself.

Q *Yeah, makes sense. Okay. Do you have any questions for me?*

S Is your [redacted] thing working?

Q *[[redacted] discussion]*

S It would be nice if you could click on the Models as well to see like a general overview or the fields that a Model has.

Q *Okay, so the fields would be useful to see then.*

S Yeah, not like very much in detail, but just the field and the type that's associated with it. Because then you have like one — because then you can see the controllers with what they do, and then also — having this is nice, but then you still have to go like this, this, this —

Q *Yeah, you have to go to the code to actually see — Would you also like to see the relationships then? In that same view or?*

S You mean the extended model part or? Or the relations actually—

Q *The relations, like when you do the getRequestResponse or something, that you actually see that there's a relation between* [redacted] *Transaction and the RequestResponse.*

S Yeah that might be nice, just as a like, — because it's pretty much just a property. It's like a special property because it has another model attached to it, but, yeah that would be nice as well.

C.15 Subject 15

Questioner (Q) *Okay. Cool.*

Subject (S) Right, so honestly I didn't use the thing, the Feature Explorer, too much because as soon as I looked for it here, I found it. So I didn't — and I already knew some of the structure of things so I didn't really need to dig into anything. So I don't know if I was supposed to use it. So the first thing I did was I went into the JSON and I added the username. I didn't really know what minified meant, I looked it up and decided to go for the same as the counterparty names. Since it's a string and it's a name, cpn. Something name. Add the name of the user. Then I generated the whole thing. Then I went here and I, — so in the beginning I kinda tried to understand what the transaction model was, I went into Sequel Pro in what we had and the IssuerTransactionModel. At the beginning I tried to go through MutationReference, filterByModel, but then I realised that if I get the RequestResponse you can get directly the labelUser and the DisplayName. But, it was giving me an error that it was called on null. I didn't had the time to dig into it so I just hot fixed it, like if it's null then let it be, if not then get the display name. And then in the test I basically said 'let's put a username here' and added to the getValidTestEnvironment and then I just printed it in a random test. I could've done an extra test but like.. I just put it here.

Q *Okay, really cool. How did you find the RequestResponse model?*

S Yeah, it was okay. I didn't look too much again into it. So I knew what I had to do, so for instance I completely ignored everything that I knew I didn't need.

Q *But how did you find the —*

S Ah, *how* did I find, ah the RequestResponse for example. Well I had a similar problem when I was trying to get data for TransactionMonitoring from other Models, I don't really remember, I think MastercardAction. So in the beginning I actually did a MutationReferenceQuery createQuery, and then I was doing a filterByExternalModelName and then I was doing [redacted]TransactionModel::class. And then at that point I was like, okay, and then we filterBy the Id. And at that point I did this and filtered by RequestResponse and I was like get itemId and then I saw 'oh you can actually get the user'.

Q *Ahh yeah.*

S Then I was like I don't need to actually do this and I can get the user directly from there.

Q *Ah fair. So you found it by the auto completion basically?*

S Yes, which is how I find most things actually. Knowing what is available and what is not and just look for what is available.

Q *Which makes sense. Okay, so you haven't used the tool too much, that's fine, what was the reason for not using it?*

S Well the reason is that I knew that just looking for [redacted] would lead me to it. And I already knew I had to go into the view and because the database or well the codebase is actually well-structured and most things are structured in the same way — once you have seen one you've seen basically all structures — and I had to into a view so I knew that the JSON would basically generate the other thing and I knew to then actually — then to make something out it'd need to add it here. As it works the same way for any other model. So I didn't need the tool because I knew I'd say where I had to go, what I had to do.

Q *Okay, cool. Then I would —*

S I had a couple of feedback on the tool.

Q *Oh yeah sure!*

S But very small ones. One is that sometimes I get this even though by changing it at some point you get it back. Now it's not.

Q *This one is consistent. Because there's a regular expression on these paths and for instance older models have a different path, so they won't show up there.*

S Yes, but sometimes it happens that it does or doesn't.. —

Q *When you click too fast?*

S Yes, —

Q *Yeah, yeah, but now you see that wasn't toggled anymore.*

S And then one thing that I think for view, this issue never change. While by expanding this they go up and down. So I think this should be somewhere else. Because I think it's annoying that it goes up and down. Maybe somewhere it doesn't go like blublublub. [up and down]. Because I thought it would change like the first time I was like, then this probably changes because it goes up and down, but it doesn't.

Q *Ooh, yeah.*

S You see like, now it's hidden and a bit weird, but that's just the user interface.

Q *Yeah, the thing is, I had them before above this, above developers even, but it could become a really long list.*

S I agree with you. That's why I don't think it should be up but it should be in a way that maybe it doesn't — that it's understandable that — maybe like down here, or maybe you can toggle it.

Q *Yeah or maybe in tabs —*

S But that's personal.

Q *I hadn't thought about the expectation that like when you select a filter that that changes as well. But yeah, it makes sense because it's below it.*

S Yeah I mean, obviously when you think about it yeah it makes sense. But if someone you know, someone is new as —

Q *Or maybe put a box around the files part already, then it's like, separate.*

S I thought about it but I like the fact that you can see the whole thing, so that you don't have to scroll back —

Q *Yeah but if you have like a small border around it, enclosing it —*

S Oh yeah, yeah, could be, but something like it.

Q *Yes, okay, I will consider that, thank you.*

-

Q *Okay, let's see.*

S Right. So first of all, this was really nice to understand how basically all models were used together. Because well as far as I know there's not a way to see for example this is not specific or the RequestResponse is not specific to [redacted]. But it's nice to see everything together so you know that this is basically taken from other things. So that was really useful. Obviously I had to also use the codebase to actually understand what things did. So I was thinking whether it might be possible to, almost integrate, maybe not directly if you click you go somewhere, but like some sort of preview of files. But that could be very complicated. But you know —

Q *What do you envision by such a preview?*

S You know how —when you have a method here, let's see [opening file]. Well anyways, when you go for, you press like F1, yes you get something like this —

Q *Oh what. How did you do that?*

S F1?

Q *Fn + F1?*

S Well it depends on your settings..—

Q *Yeah yeah yeah. O-kay. Today I Learned.*

S You didn't know this? It's really nice actually. So it could be interesting something like that. I mean it might be difficult because sometimes we have like huge models. But I don't know, but maybe something with JSON, maybe the list, I don't know. But something could be nice so that you don't have to switch —

Q *What information would be most useful here for you then?*

S So for example let's say, maybe a list of the main fields or like, you might not need all of this, but you might need okay like, relations, main fields and subfields. Something like that, some snapshot. Or even just the documentation. Something. That could be nice. So yeah, that was useful. Right so, yeah, I mean, so the first thing I'd say, we definitely need the RequestResponse model which it seems like it is the basis for any kind of this kind of things so it seems in the little handy thing that you did. I assumed that everything like iDeal [redacted] goes through that. And then we obviously would make an AliPayTransactionModel, in the same way that we have [redacted]TransactionModel or IdealIssuerTransactionModel. And that basically seems like it holds all the information of the transaction, which is kinda what we did before with that. So you can just add the username or in this case you just have the UUID at the beginning. And then we use the common QR Request model. And you just need to basically add a new element to it, so you wouldn't need to do a new thing, just add the AliPay thing. And with obviously the view model together. And then at that point when the backend sets the user in the transaction it connects the user model and the monetary account model and insert the info I guess in the transaction model. Right? And then you would also have I guess, I'm not sure when the mutation is actually created, but I would say after the user confirms the payment.

Q *Yeah.*

S Right. So the Mutation model basically goes and feeds that after the user is has confirmed it. And this 'polling the backend' I guess it's what is done via the Message. I'm not sure. I'm not sure whether this Message is send afterwards or whether this is what, that I wasn't really — because I was looking at the Messages, but it doesn't really say much. As in message data, it was a bit hard to understand.

Q *Yeah, I see. These Messages are used for the communication protocols, iDeal uses XML for instance, and that XML is parsed into the Message object, which is then passed along in our backend.*

S Okay, but is this for this, or is this to then communicate after the transaction.

Q *No, this polling, basically every five seconds the website can make a request to our API —*

S So it's a request, okay, all right. Fair enough.

Q *It would be a GET request to our endpoint and then it would send back the current status of the model basically.*

S Okay, okay, just an API call, cool.

Q *So would you know then how to implement that now?*

S How to implement this one?

Q *The polling endpoint basically.*

S Well I wouldn't know necessarily. I know the idea of a like REST API. You send a JSON and you get a Response but — as a JSON as well. Like HTTP or HTML. And then you get a JSON back right? Something like that. I wouldn't know how to implement it properly but, an idea. Yes, so okay, so for for example that was not easy to understand by looking at this if you don't know how it actually works. Because I guess, this is not part of a model. Oke, yes, so I guess, you'd have to — only the first reference that I will see to an API call is here, but then again that is me not knowing that. But I guess if someone doesn't know that this is done through an API call it's not easy to understand from here. That would be my feedback. But it's not something negative, yeah. Everything else I basically managed to understand just from here.

Q *Yeah. Have you checked the controllers part, some of the files?*

S I did. But I didn't look at all of them. I kinda looked at the names and so trying to understand what the flow was. So this is how I got the QR but I couldn't find the —

Q *There are some that don't exist anymore, this one is one of them, so this one keeps loading forever. Have you worked with workflows in our backend?*

S Yeah yeah. I have.

Q *So basically there is a mapping when you make — Are you familiar with HTTP calls? POST, GET, etc.?*

S More or less.

Q *When you make a POST call it will in our backend automatically be mapped to a Workflow... Create workflow. When you make a GET request it will automatically map to the Workflow... Read. So that's the mapping you see here.*

S Okay, I didn't know that.

Q *Yeah, that's a detail then that you need to know.*

S That makes sense.

Q *And Update for PUT requests. So for instance the polling here, that would happen through the Read workflow.*

S Right, okay, cool. I had no idea. Yes so I guess then at that point when the user confirms the payment the Mutation is created. Then the combinations [?] in the model are updated and then — here is when we have to get basically some sort of confirmation which I guess it's stored in the certificate which is — No?

Q *The certificate is used to have a secure channel to* [redacted]*. Like we get a certificate — Do you know like how private/public key stuff works?*

S Yeah, more or less.

Q *So we exchange keys, and we use a public key of the counterparty —*

S Yes, exactly, encryption, okay okay.

Q *And the certificate is just an encapsulation of the public key, so we know for sure it's their public key, and we store that.*

S Makes sense.

Q *Now we can verify that it is actually* [redacted] *talking to us, and if it's not we can ignore the entire request.*

S Right, right, that makes sense. I had something on that. Yes, so I didn't find anything on the certificate in the controllers. And I don't know whether that's because it doesn't use a workflow or — because the only mention of that if I'm correct was in the Models, yeah, I mean the Library. But it was a bit difficult to understand how the certificate was actually used. It's decently easy to understand this flow apart from the API stuff. Yeah, so that was my sort of interpretation.

Q *Okay, cool, for the last part you mentioned the certificates but basically they just poll again and the status is in the JSON response and when they see that it is accepted they know like 'okay the payment has been accepted'. It is processed. So they can use simply that.*

S So what are the main messages.

Q *The messages — we are not allowed to pass around strings in the backend so we get a raw XML or something.*

S Aaaah, and then we just parse it? Exactly. I thought it was something more complex.

Q *So depending on the protocol for AliPay you would need —*

S Ah right, right, right.

Q *That's basically the thing.*

S All right.

Q *Then that was your story, then I'll get my question list. Did you like the assignment?*

S Yeah, it was, — well I liked the second one. This [part 3] was I think a bit vague. In the sense that it was hard to understand what you wanted. I think it was useful to get to know [redacted] but I think it will be very easy if you knew, but a bit hard if you had no idea. So maybe, maybe some more guidance on what you actually want to understand. So for example, right, so here you say 'what model is used to do this?' Or 'what models are updated at each step?' Something a bit more structured probably would make me dig deeper into — like now it was very vague so I was like I'm just gonna see, but I didn't know whether what I had was enough because — well maybe if you ask questions — or maybe as two parts. So first you get a sort of general thing, and then afterwards you can give an extra sheet, so they don't look at it before, but asking a bit targeted questions. So like 'okay, at that moment at that point what model was used', or 'what could be used to make the poll'. Then at that point I'd be like 'okay, you know, if I already saw the messages were this then the okay, the only other way was an API call', so something like that.

Q *Yeah. I see. I don't want to steer too much. Because I want to see whatever you can find out by using the tool. So yeah, that's a balance indeed.*

S I think that really depends really on the experience of the person and knowledge of well both the coding but also the banking stuff. Right, so.

Q *Yeah, makes sense. You have used the Find By Label feature right? In the tool.*

S I have yes. I know it's a slice [?] but I did use it to do this.

Q *For what goal did you use it? What was the intention?*

S To find the basically the code for [redacted] for this part 3.

Q *And did this answer your expectations basically? Or do you say like there should be more information or less information?*

S The only thing where I was in doubt was whether — like I assumed it was Payments, but I wasn't 100% sure that is wasn't Deposits. The way I knew it wasn't Deposits was because you weren't in the developers. But if you had been there I may have thought that is was a combination of the two maybe. Because maybe I would have thought like, maybe okay I do the payments but then a deposit happens at some point during that. So I would have — And I tried to find iDeal, but you say iDeal issuer and there was an iDeal issuer so I was like 'I don't know which one of the three it is'. So I then looked for [..?] but I didn't really use it. Also because, yeah there were no files matching the filter.

Q *Yeah, iDeal is pretty old. So, we started using the CIT tags in commit messages which I use as a basis for the indexing. So, back then it was quite messed up basically. So yeah. That's causing me more troubles sometimes like, when there are typo's in the messages then it can put files for a feature while it shouldn't belong there basically. And that's the shitty part of my approach. The Find By Developer feature, have you used that one?*

S I did not use it apart from the part 1, where you told me to use it. But no, because I knew what I was looking for, so, yeah I didn't need it at this point.

Q *Do you think that it could be useful in certain cases or do you say that it doesn't have any added value?*

S I don't think I'd use it just by — like this. But it could be an advanced search where you can search a combination of things.

Q *So a Label and a Developer.*

S Yes, a Label, — let's say, well now there are not that many, but let's say you do something a bit more, — like, bug is very vague right? So bug you get a billion things, — but I guess you can also..—

Q *Yes that jumps into the other filter.*

S But not staying on bug?

Q *Exactly.*

S So yeah, I would do something a combination of things.

Q *And what would you use that for, that combination?*

S Well let's say I'm looking for a bug that I remember vaguely as I worked on, but I do remember who worked on it and then that is what I would use. But I don't think I would ever use developer, because I mean, [a senior developer] has done a billion things, and like yeah, at this point yeah so — when you click on [this developer] and then you click on the feature then it should show only the feature and those worked on by [this developer].

Q *Makes sense.*

S And also here it should be, rather than just by, like you could — maybe if you do just one box that if you even select one you search only for that, but you should have the option to select multiple to narrow it down. And that's how it'd be useful.

Q *And what the benefit in terms of information that you get when you have this combined search?*

S Well I mean you would find things quicker, that's the benefit I guess, that's what you use search for.

Q *Yeah okay, yeah.*

S Yeah otherwise for me it is impossible to say now 'yeah I found the bug, I don't really remember what the bug is but I remember I worked on it with [a senior developer], let me look for it'. But this is [the developer] and everything [that developer] has ever worked on but I also can't — how —

Q *What are you looking for?*

S I don't know. I was looking at bunq.me — ah yeah, okay okay, that's fine. Right, yeah, I wanted to see whether I could do basically narrow down these just by — that's what I was trying to do. Especially when you start having you know, in two years this will be double the amount of stuff. It becomes — yeah.

Q *Yeah I see, okay. cool. The Find By CIT? I assume it's basically the same story?*

S Yeah, same story. But I do like the format quite a bit. I really like this sort of interface. I didn't use it, on the other hand — let's try the one that you gave me — yeah this is nice. I like this one, I don't know what I'm copying here but —

Q *It should be the reference for GitLab. But I don't even know whether I finished that button and the right button doesn't work there's a bug in it, I found out during the first test.*

S But this is nice I think. Yeah, that's nice. This could stay by itself because I mean it's a very specific sort of URL. Right, so it wouldn't need to be combined. So it could be a nice feature yeah, that's good.

Q *What fo you think it could be used for? When would it be useful?*

S Well I mean if you know — if you are on an issue or you know the issue by heart because you have done it so many times, then you could go directly there because that would be quicker than looking by Label or by developer. So you know you just go here — it would be an alternative rather than something more maybe.

Q *And do you think that people would have a different intention when using the Feature by CIT compared to Feature By Label?*

S Well Feature by CIT is one very specific issue right, so the Label one is just the general — maybe also to look for all the different issues that have worked while this is clearly one specific issue. So that narrows it down to things that have been used for that specific issue. So that might be, so I feel like this may be more useful for Product or — while this could be — well no, both, I guess both for Product obviously. But I guess this is to track a specific problem or a specific implementation rather than a general project or a feature for example. That's how I would use it differently.

Q *Okay, cool in general, the benefits of the tool if any, other than what you have already mentioned?*

S Well, I like that it gives you a general overview of what you see, what I think it would be very nice is if — because I think one of the issues could be making for example developers actually use this, because they know the backend so well that they would just — which is what I did for example, to do this, I knew I worked [..?] I didn't need to do this. So this could be very nice for them for training purposes and I think if structured in a certain way it could be a good way to get for example non-developers, for some product owners, to understand a bit more of the code. But then in that case it would need to be simplified even more, in a sense that maybe that you like — maybe that they don't need see everything here. But yes, I guess the benefit would eventually be more noticeable in people who don't necessarily work first-hand on the backend. But for example, for me which I work on it but I'm not that big of an expert then that would definitely be useful. So that's why I'm saying training or beginners or people that don't need to work — so for example they data scientist this would be really nice. Because you know, we all try to understand how models are created and what they refer to, but then I had to learn PHP but [a data scientist] doesn't really know how to use it. So that could be a nice way to understand the backend without knowing the code. Or without having to install PhpStorm and stuff.

Q *Okay, makes sense. Things that should be different according to you that you haven't mentioned yet?*

S Well, I think I've mentioned — the logo.

Q *Yeah, you're not the first one to mention that haha.*

S Well I've been telling you everything that I could think of. Yeah I mean I told you like structure kind of things but, maybe a bit more visual differentiation between different things, so like color coding.

Q *Color coding the entries in the file list basically?*

S Yeah, so PHP, JSON, or even like Workflow, View, Model. That would definitely help. Yeah the hovering thing could be nice. Like, if not done too invasive. But yeah, if I think of something else I will let you know. But I've told you everything I could think of.

Q *Cool. Any other general feedback?*

S I couldn't find myself.

Q *Could be. Because it considers only commits on the develop branch so if your work hasn't been merged yet —*

S A lot of my work has been merged.

Q *When?*

S Since March.

Q *Interesting.*

S Yes.

Q *It could be that the references to CIT were messed up then, basically. I should check the raw database basically because –*

S But I definitely have like — I mean like I definitely had loads of things.

Q *I'll check it out actually.*

S And I definitely had to follow the CIT technical and number because I shot at right at the beginning if I was — when I wasn't doing it. So yeah.

Q *Yeah I know, for instance some people put like a typo in the commit message for the tag and then it would also —*

S Yeah, right, no no, I've definitely done a few typo's in my commits but I've committed multiple things so I should see something.

Q *Okay, I'll check it out. Interesting. Like I indexed three weeks back by now so it could be that if it was merged meanwhile.*

S No it was definitely —

Q *If it was sooner than —*

S Like my first merge was in April.

Q *Yeah then it should have been in here. Weird. Anyway. What is your impression of the bunq backend when you look at it?*

S It's very scary at the beginning. My issue, I thought that PHP was very scary but then I realised that PHP is just normal code. It's just that the backend is big, no really explained to me what you explained here [assignment paper]. So the whole Model, Controller, View and workflows are really nice once you understand them. But before you understand them they're just like a cute JSON and it doesn't make any sense. I

think the learning curve is quite steep. But then it gets way easier as it goes. Obviously the sort of notation, the style is good to have it strict. But it's a bit frustrating that it is very strict, but then stuff that was merged 6 months ago was completely different requirements. So you never now what is right and what is not. But I think it's very good. If you see our bunq analytics repository.. you'll laugh. Yes, so I think, for example, if we had in place — because we wanted to organise a sort of crash-course of the database for coders / non-developers. That never happened. But that'd be — we have a Kibana course obviously, but that's not meant to teach you how to do everything, but just to give you an idea. So that'll basically remove a lot of fear from the database and let people understand way better. At least how it works. Yes, my first few merges were like 'yeah okay I changed the model but I had no idea that it changes the controllers as well'. So I merged something and nothing changed. And I was like 'why?'. And then I had to wait for the next deploy to actually have it happen because no-one told me that the controllers as well needed to be changed. Yes, so that's — but yeah, overall it's, in the end it's very nice because it's modular and that definitely helps with a lot of things.

Q *Okay, cool. Do you think it's feasible to implement or improve features using the tool that I've created, and then without the help of a senior developer? Or like partially without? What's your view on that?*

S I don't necessarily think that implementation itself would benefit from it but maybe the planning phase. And understanding things that are already there. At least for me the problems of implementations are more, — like if I have to implement something then I actually need to look at the code for how it was done. So in the planning phase, obviously all of this I like at least the general structure, I did not really have to look at the code, I just looked at your tool. But then when you're actually go and implement things then you need to look at the code. So yeah, definitely maybe in planning, definitely helps, especially when you're sort of copying — You know, all my stuff that I've done is a lot of taking what's already there and modifying it or amending it or adding things. And, so to get an overview of what has already been done and how things sort of work together that would be really nice. But then once that I move to code then I might not use it again that much. So it'd be nice to have in the beginning. But then it also needs to be integrated well in things that we use daily.

Q *Such as?*

S For one of the problems that I see is that is for example with dashboards. I make a dashboard for compliance. And it's a complete new one and I have to make sure that they save it somewhere and they look at it because once you add a new tool, you need to get people to use it. And I have the same like oh there's this cool website that I could use, but then I'm so used to do it the other way and then I forget to check it, and then I never check it again and then I forget it and then finally I'm like aah could have used it, but — So you know, it will need to be sort of, not enforced, fed into people. I don't know how.

Q *Yeah, they should be drawn to the tool.*

S Exactly.

Q *Well, that's also why I asked like 'what information are you missing' because then I can even make it more useful.*

S Yeah yeah yeah.

Q *Okay, yeah, thanks. Then one final question. What are according to you the most important sources of information when learning a big existing codebase / framework?*

S Like for example a feature or something? What do you mean by —

Q *Well, say you're a new developer and you're dropped into like a huge framework, where would you turn to for your information basically?*

S Which is kinda what happened to me haha.

Q *But not necessarily bunq, like you can reflect on bunq as well —*

S Right, well, I mean I remember I was very confused at the beginning and all I could do was basically look at the code and trying to understand. But again, for example, if I had known how you know, workflows worked, that would already have been easier because you read the workflow and you understand all the steps. So I really liked for example the visualisation of the workflows that you made. Because those are really really clear. That's what I'm saying, so these would be really useful for when you're thrown into something new and you've no idea. So, and that's why I also think it would be good to integrate some code within the actual tool. Because at some point you still need to dive into the code. And, that could be nice to integrate somehow. Yeah I mean, obviously the code is always the single source of truth. So, I had to ask a lot to other people, like what things were. Then again, I don't really know how for new backenders how does it work? I had no real person who was following me at the beginning so I kinda chose random people or like — and then you happened to —

Q *Say you jump onto a new project, where you have all possible information sources that you can think of, which ones would you like best basically?*

S What kind of information sources? Can you name some examples?

Q *Any kind like wiki, persons to ask questions, reading the code, —*

S Okay, even if the wiki is well done, which is very rare, it can be very useful. Because it can have step by step — so yeah, the first thing I'd like is something that explains to you in words rather than code. So something like we've gone through the [redacted] together. Something like thing, you know, is definitely necessary. And I also think that's one of the problems here is that, is hard for expert people to understand what people don't know. And they might so familiar with concepts that they're just like

'yeah of course the iDeal transaction works that way'. So they might skip over some fundamental parts of the explanation. That's why something like a wiki or an explanation in instead of words that's just there, doesn't change, it could be nice. And that basically would make looking at the code way easier because you know what you're looking for. You know what things needed — you know what is implemented so you know when you find a piece of code you kinda have a wild guess of what it might do. So, that's why this could be cool because it could be a mix of the two. Oh yeah, so the problem — like obviously someone who explains it to you would be ideal but again I found that the problem is that people always assume that you know things. Not entirely but it's hard to know what a person knows. And often people might be to scared to say that they don't know things, or like someone already explained to them but they didn't really understand so they don't ask again and then.. — or like me for example, I don't really care about payments, so whenever people ask me I'm just like, okay "I'm gonna try to understand what I need to make the thing" but like I'm still a bit iffy on like iDeal on how it actually works because yeah I mean, I know enough to do what I have to do but — yeah.

Q *Okay, do you have any questions for me?*

S Ehm, no, but well done. It's a really good idea and I hope it's gonna be implemented well.

Q *I hope so too.*

C.16 Subject 16

Questioner (Q) *Okay what have you done?*

Subject (S) Okay, so based on what I know already and then I was also taken a look at your implementation, at least if what I think is a representation of what you worked on. I will have to setup a TransactionModel for AliPay, like the one we have for iDeal [redacted]. And an API to be used in the webshop which basically needs to be made of a controller that creates Transactions, gets and updates the status of a Transaction in there. Of course a view, so something that the webshop can communicate with in order to make this calls. A daemon that will then process the transaction using also a workflow or a few workflows from the controller. This API will communicate of course with the webshop you also need a frontend for that. Another component that the frontend will use is a controller for creating the QR code. Some sort of QR Code Request, that returns then a QR code model. There is technically no need to create many more things, because as far as I've seen you just extend it or add a type or I don't know what you did. Anyway, then again to be able to show everything in the webshop you need labels of some kind. Something to show some specific information that are only for AliPay. And yes,.. did I mention the daemon already?

Q *The daemon? Yes.*

S Okay. Well we need tests. Otherwise it doesn't work. But then there are the last two parts, so when the transaction is created and then the user scans the QR code we should be able to show to provide to the user a Request and creating a Request for that. I mean AliPayPayment Transaction would request some changes so the creation of the Request. And then user approves or not, so when this is approved then transaction enters a specific state and the daemon processes. And then what happens when the daemon processes, the daemon probably if AliPay is not something we came up with —

Q *It's something — a name I just*

S Ah, okay, so if it is an external system then we need to communicate with that external system. And therefor have possible an API or maybe simply — no possibly an API. Like some system that communicates with AliPay that handles message exchange, authentication or handshakes between the two systems in order to make sure that only stuff that needs to pass is passed. And probably also some — now that I think about it — maybe there's something to integrate with the transaction monitoring. And risk related things. But I think that could be basically it. I might probably forget something but, —

Q *No, it sounds fairly complete. So yeah definitely. Then I'll just — I have a question list, things I want to ask. So, you used the Find By Label feature I see?*

S First of all I actually I went from your page — so first of all I looked by developer, thereafter I asked you whether you building [redacted], so I could've gone also for [a senior developer] but a lot of stuff was done. So looking for your name for sure could have given a more isolated case to work with. Also, this is something I have worked a lot with, from the frontend side. So I know quite well how it works. So that probably helped. And yeah, then from your profile, from your page, I clicked on — I didn't know whether it would be [redacted] —

Q *Ah yeah, I see.*

S And, but [redacted] was not it. So then I clicked on the other one and I got here.

Q *Okay very cool. Is the information on this page helping you in understanding how it was build, do you miss information or what are your thoughts when you see this page?*

S So if a see this, this has been very helpful. But because I knew already what it — kinda what I was looking for. So I could've — it's something that I've already done other times, but only looking at maybe a specific merge request. And then you have to open others and see 'okay, so this was then here, this was then here, this was then here'. I don't know if I would have had the same — if this information would have been as useful as it has been right now, if I had no idea what's going on.

Q *So, that's then having an understanding of the structure of the code or something different?*

S Yeah, structure of the code for sure, and also know how we do things a little bit. If we create, for example, a new type of issuer then only these things are needed because the others are probably there. Or the fact that here you — it was enough to create this one —

Q *Yes.*

S Because then probably this can be used — can be calling this, I don't know. Those are kinda details. So he we do things, how we trigger sub workflows, how we usually extends a model and whether it is by adding a type or by actually creating an extended model.

Q *Yeah I see. Do you have an suggestion on how to make this more clear from the interface?*

S Is it possible to see which code was —?

Q *There's no code because this is based on the git history so the code itself is basically fetched — like for the diagrams, I fetch the latest version of the code from GitLab.*

S [click non-existing workflow, mumbles something]

Q *Or it doesn't exist anymore. That's also possible. [checking internet connection]*

S [debugging diagrams not loading..]

Q *Well, anyway for the graphs it pulls the latest version of the raw files basically from GitLab. But other than that I don't have any source code in the system right now.*

S But you could have, I mean, you could pull the commit. Right?

Q *Yeah, I could.*

S Okay. If you're using the git history then you can probably pull a specific version.

Q *Yes, I have the file path, I have the commit. So I can show the source code, that's no problem.*

S Also, maybe something to understand. What you created and what you changed.

Q *Yes.*

S Because then it's more clear to — it can become more clear to see what you actually, the components you need. Because here I only talked about the new components and I didn't write anything about changing the existing ones. So maybe that could already be a —

Q *So, indicate like here what needs to be extended versus what needs to be created, yeah okay.*

S That will make a better plan. Because if you have two files and maybe one file is 1500 lines and you need to add 1, and the other one is 800 and you have to write all of them, that's different.

Q *Yeah, for sure.*

S So you have an idea of what you need to do.

Q *Okay, makes sense. Any other feedback on this screen? Things you would have liked to see, or ..?*

S Yeah I don't know if it would be actually useful but I don't know who wrote what.

Q *Ah, the authors? Of the changes.*

S Yeah, I don't know if it would be actually useful. Like this was already quite nice, as said. Because I knew already what I was looking for.

Q *And what would you use like that information for? Seeing the authors of a change.*

S Okay maybe not. Forget about that, yeah. Again, definitely not a — not useful. Yeah no, I right now don't have any other feedback on this screen.

Q *Okay, cool. The Find By Developer you used as well you mentioned. Why did you choose to use the developer versus the searching by label at first?*

S Well, mainly because I asked you whether you build that. And you said yes.

Q *Do you always go by developer first, or what's your typical approach when you search for information?*

S Well, it still depends on what I'm doing. For example, the other times that I've been looking for on GitLab for something like this is when I had to — a few times I had to extend the generator. In that case I needed to know to find that one merge request that was extending the generator with a new type. So I just looked around, this but also this merge request, and see 'does it work?', so probably if I had something like this I wouldn't be looking by developer. I probably would be looking by feature.

Q *Okay I see.*

S Not even by label, but actually by feature. Just get that one issue that integrated that one. And then see inside pieces that are probably related to that. Of course if I know a feature is about [redacted], and actually what I want to do is completely different. I want to add a new type for the database and that's it, I know that if I look at the list of files changed, that I will not have to do something with [redacted] but it will be probably in the very generic files such as the generator, BunqType and things like that.

Q *Yeah. Okay. So for whats situations would you use the Find By Developer? Because now you used it because you asked me like 'did you build it', but what else can you think of? If anything.*

S Yeah maybe if I already can know what I'm looking for. So, not really to explore. But — Or actually, ah, actually the first time I used it, the first thing I did was actually looking for my name, because I wanted to see what I worked on. So only for a historical perspective. And, for fun basically. Either I know what I'm looking for or for for fun.

Q *Okay, the Find By CIT have you used that one? The last option.*

S Only in the —

Q *In the exploration —*

S Yeah, only in the exploration and — yeah, that something that came — I would say, if I had known which issue you worked I would have used that for sure. Then I would have preferred that to that Find By Label.

Q *And what's the reason for preferring it?*

S Because issues in that case are very well self-contained. Of course they might not contain bug fixes, but if I want to build the feature, I find it more likely — I think it's more likely that I will find what I need — the information I need in the issue, rather than finding it with the label.

Q *Okay, makes sense.*

S Especially if you start having PARP or some new product.

Q *Yeah, exactly.*

S [interrupt by developer under supervision of Parrello]

Q *Okay, so we were talking about this page.*

S You asked me about CIT, and I told you that I think that it might contain more information if the project is big then Find By Label. In the long run I works best.

Q *Yes, okay. Why do you think it works best, Find By Label?*

S Because then you start having also the buggies and you start having this and that. And all kinds of edits, so on one side it gets more complete but now that I think about it on the other side it also gets more cluttered. Dirtier.

Q *Okay, perfect. Are there any benefits to the tool that you haven't mentioned yet?*

S Well I haven't seen the UML visualisation, —

Q *Maybe it works now.. [checks GitLab connection] basically it shows the visualisation of the workflow, it shows all the steps, with like arrows like how you get there. And if there are branches like if-conditions then it shows with an arrow with the condition on it how you flow, so it's a flowchart basically.*

S And that's nice. It's the reason why we used to have them back in the days.

Q *Yeah, in the flow tool?*

S Yeah. Exactly. Sometimes it can get quite annoying to figure out what goes where. Especially when you take a look at old workflows that for some weird reason are not even ordered the right way. So it's pretty annoying and having some tool that helps you see what is going where like actually considering sub flows and anything else, would be cool.

Q *Well that's partially what was in there. Do you see added value for like new developers when using this tool or?*

S Visualisation yes. I think the Find By Label, could be good. Yeah, Find By Label could be good, with some additions. And — maybe it could be nice what I was saying about edited files and new files.

Q *Ah yeah, the file status.*

S That could be an option because in fact if I just want to see which files have references to a specific feature I mean I don't care about knowing whether files are edited or new. I just want to know the extent — how wide a feature is. So Monetary Accounts probably pffr —

Q *Yeah that explodes all over the place.*

S Exactly. So that could also be a nice thing. To see what is what. How important things are — how intertwined features are, right now at the moment. Also, maybe could be possible, could be interesting also to plan some refactoring. If a feature is too much around — it's all over the place and you want to make it more independent, this could be maybe a tool to figure out what. But it requires — the fact that it is dependent on Labels could be a little bit risky. Not risky, but hard to maintain. But again, that's the only thing that is basically possible.

Q *Okay, cool, are there things that should be different in the tool? Or that you say like this is not necessary, this is not helping me?*

S Maybe have it a little bit bigger.

Q *Like wider?*

S Yes, wider. Because this is just an instruction block that once you read, you're done I believe. So the exploration maybe might be a bit, take more space than what it does right now. So you can actually see trees and maybe you can visualise a workflow while you're still browsing the rest of the stuff. Because you have space anyway and okay this is a 15", on my 13" it would be what, like this? So you still have a lot of space to use.

Q *A lot of space yeah.*

S And then it goes down. Explore, issues, so yeah probably some visual changes could make it nicer to use.

Q *And in terms of information?*

S Sorry, what you mean?

Q *Well in terms of information are there things that should be different? Like things that should be added — well you mentioned the authors and the file statuses, anything you haven't mentioned yet?*

S No, I don't think so. There's nothing that I can think of right now.

Q *Okay, cool. Any general feedback that you haven't mentioned yet?*

S No.

Q *Okay, when looking at the bunq backend, what is your impression?*

S It's big. It's really big. And the bigger it gets, the harder it is to keep it in a decent state, in a good state. And I don't know how sustainable this is.

Q *Why do you think that?*

S Because, right now we have three people reviewing at the end, three people are merging and so they can't review everything, and as a consequence code — I think code is not as good as it used to be. We can train — we are training more developers to do that, but training more people means also that it's harder to keep consistency. So either, — I don't know if there's a solution for that — apart from automatic review.

Q *Yeah, but that's also never perfect.*

S No, but at least it has a framework and that's it.

Q *Yeah for sure.*

S So having something like that. At least maybe with warnings for naming for example. Warnings so things that then a reviewer could have and say 'yes, this is fine, ignore', 'ah no, this is actually a good point, comment'.

Q *Yeah, so that you have some focus points.*

S Yeah, exactly.

Q *Yeah, that makes sense. Do you think it's feasible to implement or improve features using this tool without the help of a senior? Now we ask seniors for help often times, but do you think this tool would be able to replace that either completely or partially?*

S Partially yes, at least for the exploration part because one of the things that happen a lot of time is that someone just says 'I need to implement this, how do I do it?'. And the answer is always, or most of the times, 'yeah there is something similar, look for this' and then 'where do I find it?', and this could be a tool that makes the helps them being proactive in this. When new system, new — like the more we write the more chances there are that something similar is existing already. So something like that could be nice.

Q *Okay, then a more generic question. What are, according to you, the most important sources of information when you learn a new, big, code base or framework?*

S So, outside of bunq?

Q *Yeah, or both, like..*

S Usually documentation of the framework for the setup. And maybe the first things I try. And then as soon as I am working on real problems it's either again the documentation if well done. If not it's Google and Google and Google and Google. And sometimes maybe even check systems for example doing some JavaScript, it can be useful to look at websites that are implementing that.

Q *Aah, going by examples basically.*

S Yes, exactly. "Is this done? Yeah okay, good." —

Q *— Just steal this and this and it works for me.*

S Yeah, precisely.

Q *Okay, cool. Those were my questions do you have any questions for me?*

S No, I don't think so.

Q *Okay, thank you very much then.*

C.17 Subject 17

Questioner (Q) *Okay, cool, wat heb je gedaan, gevonden om het implementeren?*

Subject (S) Wil je dat ik part 3 vertel of part 1?

Q *Eh, part 2. Eerst de implementatie opdracht, wat je hebt gedaan, hoe je er bent gekomen. Dat soort zaken.*

S Eh ja, dus ik heb gewoon gezocht op [redacted] inderdaad, geklikt op de views daarvan, view naam gekopieerd, geplakt in het zoekvak van PhpStorm super makkelijk, ik hoef bijna niet na te denken. En, hoe heet het, gewoon de view definitie aangepast. En geen idee of dit goed is maar voor de vorm.

Q *Yes. Even kijken hoor, in RequestInquiry createdBy, oke, hoe wist je dat je RequestInquiry kon gebruiken?*

S Ik heb de auto-complete gewoon gebruikt op het TransactionModel.

Q *Ah right. Met de auto-complete gewoon proberen en dan —*

S Ja, trial and error. Auto-complete is heel goed.

Q *Ja absoluut, dat is allemaal goed geannotate en alles. Dat is het voordeel van dat object oriented.. — Okay, cool. Heb je verder nog de tool gebruikt voor deze opdracht of heb je echt alleen de view en daarvandaan verder?*

S Nee, dat was eigenlijk alle informatie die ik nodig had uit de tool, de naam van de view.

Q *Had je—*

S En een beetje om een overzicht te krijgen van wat er nog meer in zit. Maar ik kan vrij gericht kijken.

Q *Had je verder nog feedback aan de hand van deze opdracht?*

S Nee, dat is bij de volgende pas eigenlijk. Geen zin om tests te schrijven nu —

Q *Nee ja, dat hoeft ook niet.*

S Alleen dat de tool af en toe een beetje buggy lijkt misschien. Omdat ie soms wel en soms niet files matching terug geeft. Of is dat mijn verbeelding?

Q *Ah er zit blijkbaar met die bar, als je te snel klikt dan is geen een van die dingen meer getoggeld for some reason. Ik weet niet echt waarom —*

S Precies, dan komt ie in een invalid state terecht.

Q *Precies. En soms heb je, — ik gebruik zeg maar regular expressions op die folder names van* `_model`, `_controller`, *whatever. En vooral in de oude structuur toen was dat nog niet, en dat matched ie niet meer. Dus dat zou kunnen. In principe bij all zou 'ie wel altijd alles moeten laten zien, maar als dan vervolgens zo'n subview niks laat zien dan is het hoogstwaarschijnlijk dat.*

S Ja. Verder vind ik het heel fijn dat ik hier gewoon een dropdown en autocomplete — dat is echt super behulpzaam. Als developer kun je gewoon lekker zien wat er allemaal in zit. Completion time ook? [?]

Q *Ja dat zijn labels die ze vroeger gebruikt hebben. Ik moet nog even kijken of zeg maar hoe ik dat verder — [interrupted by colleagues].*

S Dus part 3?

Q *Had je die al af?*

S Daar was ik wel al mee bezig.

Q *Oh oke, ja, dan zou ik zeggen, maak die gewoon lekker af.*

S Ben ik wel de juiste doelgroep eigenlijk hiervoor omdat ik de backend al ken?

Q *Ja, maar ik probeer nu eigenlijk te vergelijken met mensen die geen ervaring met mensen die junior en senior zelfs of die er ook nog informatie uit kunnen halen die nuttig is. Vond ik wel een goed advies van m'n begeleider, die zegt dan kan je mooi zien 'hebben zij er ook nog wat aan', of 'als je er nog iets aan toevoegt hebben ze er dan wat aan'.*

S Ja, slim.

Q *Normaal staat er ook 2 uur voor, maar voor iemand die er al in gewerkt heeft ben je in een uur klaar of zo. Ook wat je zegt, je wist dat je de view nodig had en je was klaar basically.*

-

Q *Okay, cool.*

S Oke, ik heb de stapjes een beetje opgeschreven. Eerst goed gekeken naar de andere payment methods, iDeal [redacted]. En daarna eerst bedacht wat de routes zullen moeten zijn, wat we exposen precies om dit te doen. Dus we hebben nodig, die initiation een POST, een QR code op vragen en de status van de transactie kunnen opvragen, of 'ie al completed is of niet vanuit de app. Dat doet de webshop. En vanuit het perspectief van de app hebben we dan ook nog een paar routes nodig. Diezelfde transactie status en een manier om de transactie te updaten. Hoe heet het, dus die tool is superhandig om even de naming conventie te achterhalen. Dat we bijvoorbeeld [redacted], `core_[redacted]_transaction` hebben. Dan moet dit waarschijnlijk iets vergelijkbaars zijn. `alipay_issuer_transaction_view` enzovoort. Dus, dat even kunnen ontdekken is fijn. En bijvoorbeeld met de controllers kon ik ook een verband leggen waar ik anders niet had gelegd waarschijnlijk. Namelijk dat we een TokenQrRequest controller hebben. Die kan ik wel gebruiken voor het genereren van een QR code dat was fijn — of de `core_qr_code`. Ja hier is TokenQrRequestIdealCreate, [redacted], gewoon eentje toevoegen voor AliPay. Dus die discoverability is wel fijn. Volgende stap, views, naming conventie dus achterhaald uit de tool, ja. En toen bleek dat die `token_qr_request_view` al bestaat en dat daar al een iDeal [redacted] view in zaten en dat ik hem daar gewoon weer bij kan schuiven. Models, ook gewoon eerst gekeken in de tool. [redacted] blijkt dan een [redacted]MessageModel te hebben, dat is een implementatie detail van [redacted].

En misschien heb je dan voor AliPay ook zoiets nodig, hangt er vanaf van hun implementatie.

Q *Ja precies.*

S Dus dat eigenlijk. Is dat wat je wilt of wil je het uitgebreider hebben dan dit?

Q *Ja nee, — ik zie dat je ook nog wat hebt staan over de controllers? Had je daarvan gevonden welke je nodig had allemaal?*

S Ja, ik denk daar vond ik dus die core QR token controller in. Deze TokenQrRequestIdealCreate enzovoort.

Q *Yes, exactly.*

S Daar zou ik dan misschien niet onmiddellijk, die zou ik niet onmiddellijk hebben gevonden in PhpStorm.

Q *Nee precies, want hoe zou je er normaal naar hebben gezocht denk je?*

S Als je op Qr zoekt krijg je natuurlijk van alles en misschien zou je dan geneigd zijn om bijvoorbeeld in `qr_code_lib` te kijken van hebben we hier een library-functie om dat te genereren. Maar, je moet het natuurlijk netjes vanuit een workflow doen, dat leer je ook met ervaring. Dus dan ga je zoeken naar WorkflowQr, QrCodeGenerate, en dan zie je misschien ooit dat er iets is waar we er een miljoen van doen.

Q *Ja, exactly.*

S Die fuzzy finding in PhpStorm is wel nice natuurlijk. Dat ik gewoon WorkflowQr en dat 't er ook tussen kan zitten, dat ik het nog vind.

Q *Maar je zal iets meer moeten zoeken dan.*

S Ja.

Q *Okay, cool. Dan heb ik nog wat vragen voor je. Vond je de assignment leuk?*

S Ja.

Q *Okay, gelukkig. Je hebt de Find By Label zo te zien gebruikt. Voor welk doel heb je die gebruikt en wat vond je er van, van de informatie die je hier krijgt bijvoorbeeld?*

S De Find By Label?

Q *Ja zeg maar wat je nu hier — die bovenste search option.*

S Ja, want dan heb je het expliciet gelabeld als Feature: [redacted]. Hoe heet het, het is wel fijn dat er veel informatie is, dat je bij iDeal Incoming en Outgoing dat hier ook Top-Up bij staat bijvoorbeeld. Eh ja, het is allemaal heel duidelijk gewoon. En heel fijn om zo een beetje te kunnen ontdekken wat er allemaal in de backend zit. Dat had echt ontzettend geholpen denk ik toen ik net begonnen was.

Q *In welk opzicht had het je geholpen?*

S Denk beter die verbanden te zien van 'oh, dit gebruikt dat', hoe het allemaal in elkaar knoopt. Verbanden die je misschien niet meteen legt als je naar de code kijkt.

Q *Je ziet net iets meer de verborgen —*

S Dat is het hele idee natuurlijk, dat je verbanden legt met —

Q *En qua information die je aan de rechterkant aan je getoond wordt? Kom je daar nog iets tekort of dat je zegt 'dat is niet nodig'?*

S Soms staat er voor mijn gevoel meer tussen dan je eigenlijk zou willen. Of heel veel common dingen staan er hier tussen. Waarom staan die er dan tussen bijvoorbeeld? Omdat die ook aangeraakt zijn tijdens het bouwen van de [redacted]?

Q *Die zijn dan een keer aangeraakt met een commit die verwijst naar een issue dat weer met deze label zit.*

S Ja, dat kan ook op zich heel nuttig zijn. Maar misschien een beetje uitleggen van waarom is dit eigenlijk, dat je door kan klikken op die commit misschien. Ik denk gewoon even hardop.

Q *Jaja, het kan natuurlijk meerdere commits zijn, dus dat je dan kan zien in welke commits het bestand allemaal gewijzigd is en wat er gewijzigd is maar —*

S Dat zou helemaal vet zijn. Om een soort inline difference te kunnen zien, voor deze feature waren deze twee regels geschreven in a certain time [?].

Q *Ja precies.Op zich moet het —doable zijn. Want ik heb de commit hashes dus dan moet ik met de GitLab API de diffs van een commit eruit trekken, filteren op een bestand,.. moet te doen zijn.*

S Ja. Dit boxje hier heb ik helemaal niet gelezen.

Q *Het informatieboxje?*

S Het zou evengoed een advertentie kunnen zijn, ik heb er niet eens naar gekeken eigenlijk.

Q *AdBlocker blocked hem niet. Al zit er ook geen AdBlocker op. Maar, ja.*

S Dat was het wel zo'n beetje wat ik had te zeggen.

Q *Okay, cool. De Find By Developer, had je die nog gebruikt?*

S Nee, alleen voor part 1 omdat het daar gevraagd werd om het te doen.

Q *Ja, precies. En wat is de reden dat je hem niet hebt kunnen of niet hebt willen gebruiken?*

S Ik had het niet echt nodig voor m'n gevoel. Dat zou ik dan meer uit persoonlijke interesse bekijken van 'oh wat heeft die persoon allemaal geschreven'. Af en toe vind ik het leuk om een beetje door te lezen, maar ik zie niet direct wat het nut zou zijn bij het bouwen van een feature.

Q *Oke, helder. De Feature By CIT?*

S Heb ik ook niet gebruikt. Bij het bouwen van een nieuwe feature hoef je sowieso niet echt te kijken naar bestaande bugs bijvoorbeeld.

Q *Maar zou je er wel een use case voor kunnen bedenken in dit geval of zeg je ook van 'ja, nee ik zie eigenlijk niet het voordeel er van in'?*

S Ja, ik denk als ik nog even nader moet onderzoeken van 'deze ticket, wat is daar allemaal door — wat is daar allemaal mee veranderd/gemoeid?'

Q *En wat zou daar het voordeel van kunnen zijn als je dat kan zien denk je?*

S Een use-case die ik kan bedenken is als ik een ticket overneem van iemand anders of zo. Of iets af moet maken van iemand anders.

Q *Ah ja. Ja. Dat je kan zien wat 'ie allemaal al heeft gedaan.*

S Ja.

Q *Ja. Oke. Useful. Zijn er voordelen aan het gebruik van de tool, en if so, welke?*

S Voordelen, ja, gewoon dat je een overzicht krijgt — dat je kan ontdekken van deze feature, dit hoort er allemaal bij. Dat is, de killer feature denk ik toch?

Q *Naja, dat vraag ik aan jou.*

S Eh ja, het is heel fijn om gewoon te kunnen kijken hoe is dit al eerder gebouwd, want je moet gewoon de conventies hergebruiken, en die kan ik er een beetje uit aflezen van 'oh zo wordt dit genoemd', 'zo wordt dit ingedeeld'. Kan ook weer een valstrik zijn, dat je gewoon maar terug valt op wat er al is en niet zelf meer bedenkt van 'kan ik dit beter structureren'.

Q *Ah ja, fair, je ziet de wel de oude structuren en niet — ja. Dat is inderdaad wel een pitfall. Je kopieert dat eigenlijk de fouten die al eerder zijn gemaakt. Fair.*

S Echter was ik daar in mijn tijd bij bunq ook een beetje huiverig voor om dingen anders te doen of zo, je wilt eigenlijk passen in het stramien zodat er niet over gezeurd wordt.

Q *Precies.*

S Zijn er ook nog dingen die anders hadden gemoeten in de tool? Extra of,.. ? Informatie die je mist bijvoorbeeld?

Q *Ja, dit is heel erg gericht op de files. En ik zie hier niet echt de — bijvoorbeeld de CIT issues in terugkomen. Oh hier, die zijn hieronder. Ja precies. Die waren me niet echt opgevallen eerlijk gezegd, want het staat helemaal onderaan. Dat is eigenlijk wel handig om door te kunnen klikken.*

S Ja je kan in principe nu zo naar GitLab en zo.

Q *Ja, dus hier heb ik de issue. En dan kan ik hier ergens de merge request vinden waarschijnlijk.*

S Ja. Is het nuttig om direct de merge requests bij een issue ook in de tool te hebben denk je?

Q *Zeker. Ja. Dan kun je echt gewoon de merge request lezen en daarvan leren. Dit is hoe het toen geschreven was de eerste keer. En dit was de feedback van [a senior developer] erop ofzo. Mocht je dat willen.*

S Oke. Fair. Anderzijds nog feedback op de tool die je nog niet hebt genoemd?

Q *Nee, niet echt. Ziet er goed uit en ik vind het een fijne tool.*

S Dat is altijd mooie feedback. Dan iets generiekere vragen. Wat is jou indruk van de bunq backend, als je dat zo voor je ziet, wat denk je dan?

Q *Een grote dumbster fire. Nee, hoe heet het, ik vind het toch een beetje erg tied coupling. Ik werk nu op een project met allemaal micro services wat allemaal in aparte repositories zit, dus dat dwingt je om alles helemaal gescheiden te houden. Hier heb je het een soort van logisch ingedeeld op feature niveau. Maar overal zitten snoertjes tussen eigenlijk. Dat is het ding een beetje.*

S Het is een grote spaghetti bal eigenlijk?

Q *Ja. Niet helemaal, maar wel een beetje.*

S En als je daarmee zou moeten werken, als je dat de eerste keer zo ziet, je eerste dag en je zou er mee moeten werken. Zou je dan denken van ‘zo dat gaan we eens even doen’ of hoe zou je je daarbij voelen denk je? Of naja, hoe voelde je je erbij, want je hebt ermee gewerkt?

Q *Het voelt soms wel als een soort oerwoud, van ‘oh hier zit ook weer een ding dat ik niet wist wat belangrijk is om te weten’ en ‘oh hier ook weer en hier ook weer’.*

S Het voelt soms wel als een soort oerwoud, van ‘oh hier zit ook weer een ding dat ik niet wist wat belangrijk is om te weten’ en ‘oh hier ook weer en hier ook weer’.

Q *Ja.*

S Maar op een gegeven moment ken je het natuurlijk wel.

Q *Ja, al doende leert men. Denk je dat het mogelijk is om features te implementeren of te verbeteren met deze tool zonder de hulp van een senior developer of deels vervangen?*

S Dat betwijfel ik. Ik denk dat het geen vervanging kan zijn voor goeie mentoring.

Q *Oke, en wat is de reden daarvan?*

S Je kan de tool niet gewoon een vraag stellen — of de tool kan niet alles beantwoorden want het is maar een tool.

Q *Ja, precies.*

S Uiteindelijk zul je toch mensen nodig hebben om vragen aan te stellen althans — of je moet een genie zijn die alles begrijpt dat ie leest de eerste keer.

Q *Fair point.*

S Dat ben ik niet in ieder geval.

Q *Wat voor dingen zou je dan nog aan een senior willen vragen? Als je deze tool hebt? Wat voor dingen zouden dat kunnen zijn?*

S Beetje een inkoppertje, maar 'is dit de juiste manier hoe ik het nu ga bouwen'.

Q *Oke, fair, maar dat zeggen de seniors ook niet altijd. Die zeggen vaak vak 'zoek het maar uit of het de goede manier is'.*

S Ja dan kom je in de review fase wel achter als je nog meer tijd hebt verspild met het bouwen van het verkeerde ding.

Q *En dan zit er dadelijk dode code in de backend die nooit gebruikt wordt.*

S Ja, en als ik eerst wil vragen 'is deze opzet goed' en ik krijg te horen 'zoek het maar uit' en dan bouw ik het, en dan wordt ik afgerekend 'dit is niet goed', ja —

Q *"Waarom dan?".*

S Dat had allemaal niet nodig geweest maar ja.

Q *Precies. Wat zijn volgens jou de meest belangrijke informatiebronnen als je een groot nieuw framework of een groot nieuwe codebase gaat leren waar je aan moet werken?*

S Als 'ie er is, documentatie.

Q *En dan bedoel je documentatie in de code of meer als een wiki?*

S Losse documentatie. Wiki, of gewoon een documentatie site van bijvoorbeeld een Symphony of een Laravel is super handig. Hoe heet het, ja in het project waar ik nu werk hebben we gewoon een hele repository alleen maar met MarkDown documentatie files. Allemaal georganiseerd van 'begin hier' en lees dit allemaal ook even. Dat is

chill. Gewoon de code lezen op zich. Het is zo veel in de backend, dat je gewoon niet weet waar je moet beginnen in het begin. Dus ja. Daar helpt de tool wel mee, dan kan je wat meer gericht lezen. En, de uitleg die je hier hebt, die helpt al enorm. Dat kan ook gewoon op de bunq wiki denk ik.

Q *Ja dat is ook wel het idee dat ik het daar nu ga zetten. Ik heb van meer mensen gehoord van 'als ik dit had gehad aan het begin dan..', dus ik denk dat ik ook wel gewoon die plaatjes, gewoon — het visualiseert heel erg. En nu staat dat nergens. Okay, cool. Nog andere bronnen die voor jou belangrijk zouden kunnen zijn? Van informatie?*

S Nee, ik denk dat dit gewoon heel erg helpt om een paar features door te nemen in het begin, in je eerste week van 'oke, deze feature waar bestaat het allemaal uit' en dan cherry-picken hieruit en de code lezen bijvoorbeeld. Dat is wel super chill.

Q *Dus jij zou zeggen van in de eerste week, dan kan je iemand deze tool geven en zeggen van 'kijk eens naar een feature hoe het in elkaar zit, dat je een beetje kan zien, structuur en wat er nodig is' en dan je eerste taak. —*

S Ja, deze hele opdracht is in principe een soort backend training. Want alles is helemaal gedefinieerd wat je moet doen van 'oh je gaat een nieuwe payment method designen, hoe ga je dat doen?'. En hier is alles wel van te voren al bekend natuurlijk maar dat helpt enorm om er even in te komen.

Q *Het is eigenlijk een soort oefen exercise op die manier. Ja. Dat is wel — zo kan ik het misschien zelfs wel voorstellen nog van 'kan ik die niet als standaard exercise voor de backenders ..' — Ja, oke. Fair. Dat waren mijn vragen, ik weet niet of jij nog vragen hebt of feedback die je kwijt wilt of whatever.*

S Nee, alleen goed werk, goed bezig.

Q *Dankjewel.*

C.18 Subject 18

Questioner (Q) *Oke, wat heb je geprobeerd?*

Subject (S) Oke, ik heb eerst hier dat ding opgezocht '[redacted]', en opgezocht — hier even naar gekeken. Vervolgens hier een hint gezien dat je naar de views moet kijken, dat had ik eerst niet door en volgens mij de eerste keer dat ik op 'views' klikte gaf ie ook een leeg iets, maar goed.

Q *Ja, de button blijft af en toe hangen.*

S En dan zag ik hier volgens mij naar de transaction gegaan, ja die. Hier even gekeken hoe de structuur eruit ziet en mijn eigen ding toegevoegd. Vervolgens, naja naar de Response gegaan. Daar even gegeven hoe die werkte en zag ik dat hier alle fields geset werden. Dus eerst geprobeerd een field te setten met Hello World en dan vervolgens

echt heel veel dingen in TransactionModel geprobeerd om te vinden hoe ik bij het User ID kom maar dat werkte helaas allemaal niet.

Q *Oke. En hoe had je gevonden, want ik zie bijvoorbeeld nu dat je RequestInquiry — hoe ben je daar gekomen om te proberen?*

S Oh dat is gewoon letterlijk auto complete vanaf TransactionModel, gewoon auto complete en alles afgaan om te kijken of je op de een of andere manier bij een gebruiker aan komt. En ik heb ook nog wel hier gekeken. Of hier nog — of ik hier nog iets aan had —

Q *Ja.*

S Maar misschien dat ik net iets te ver uit het systeem zit om helemaal te begrijpen wat hier allemaal gebeurd. Dus dat ik niet echt de link kan leggen tussen — want ik zie hier verder niks met User staan, bij alle drie niet volgens mij. Dus het was — het is dan nogal lastig om de link te leggen tussen — oh 'hier kan ik de gebruiker van de transaction vinden'.

Q *Ja, precies. Dus je hebt — heb je nog bestanden bekeken die je hier genoemd stonden of heb je eigenlijk met name —*

S Ik heb via auto complete ben ik bestanden in gegaan. Dus ik denk dat ik wel in een — volgens mij had ik er nog een paar van de Messages ingegaan maar dat was nog toen ik die JSON bestand aan het zoeken was. En ja ik heb nog wel al deze dingen zitten openen, maar daar — misschien dat ik gewoon net er overheen kijk ofzo.

Q *Ja, nee oke. Dan, je zit wel al de goede kant op met de RequestInquiry, —*

S Ah toch wel.

Q *Ja, ja, je hebt RequestInquiry en Response, — even deze terug zetten naar wat je had — oh sterker nog, getLabelUser, daarvandaan kan je geloof ik —*

S Ja, dat, ik zou toch zweren dat ik dat net geprobeerd heb.

Q *Ik weet niet wat ie dan nu voor error geeft, maar.. —*

S Ja en dan geeft ie dus dit. En dat gaf ie dus elke keer als ik zeg maar een veld had dat ik daar —

Q *Oooh, heb ik dan het testen misschien niet goed uitgelegd, dat zou kunnen hoor, aah, de testRead geeft een Exception.*

S Ah.

Q *Oh en dit makes sense dat deze null is, en daarom schiet ie er dus doorheen. En dit klopt, zeg maar je gaat nu via de RequestInquiry, die wordt pas geset op het moment dat je hem scanned.*

S Ah

Q *En op dit moment is ie nog niet gescand zeg maar. Omdat de Payment is aangemaakt —*

S Dus ik had gewoon het antwoord goed.

Q *Je zat inderdaad goed.*

S Ja, want een van die dingen die had ik dus al eerder, maar dat werkte — zeg maar ik zat dat object in te kijken en dat werkte dus niet. Dus ik had echt iets van 'what the f***' dit zou het toch echt moeten zijn. En ik had nog een paar andere dingen geprobeerd die ik nu ook niet meer weet. Die allemaal wel iets met Users te maken hadden, maar daar kwam dus elke keer gewoon een leeg object uit of bijna leeg object uit. Dus ik dacht — ik was zelfs nog helemaal die Exceptie in gaan steppen en toen zag ik nog iets met 'heeft geen read access', dus dacht ik 'oh blijkbaar kun je het wel getten, maar heb je geen access of zo' dus toen..

Q *Ja, nee oke, maakt niet uit, je bent — naja je zit al bij LabelUser dus dat is nice. Cool. Dan zou ik voorstellen dat je naar de derde gaat. Daar hoef je niet meer te implementeren. Ik — wil je met pen/papier werken of wil je —*

S Nee, ik schrijf het wel even op.

Q *Oke, dat is prima. En het gaat er dan met name om dat je mij dan zo duidelijk mogelijk kan uitleggen straks wat er allemaal voor stappen nodig zijn om het te implementeren. Het hoeft niet op method level, maar in ieder geval de klassen bijvoorbeeld die je bijvoorbeeld nodig hebt, de entiteiten in de backend —*

S Het zijn er heel veel — vrij veel volgens mij.

Q *Het zijn er aardig wat, ik ben gewoon benieuwd hoe ver je komt. En dan, naja, in principe geef ik daar meestal een uurtje voor dus — maar als je zegt 'ik ben er klaar mee' mag je het zeggen.*

-

Q *Vertel, wat denk je dat er nodig is?*

S Een model, een view en een controller.

Q *Fair point.*

S Specifiek de TransactionController, Model en View die je ook in [redacted] ziet. Als je het zo uitspreekt. Oh en dan nog een QrCodeController die de — valideert dat je daadwerkelijk de persoon bent die — als je 'm scant dat je dan — dat de token niet opnieuw gebruikt is en dat er daadwerkelijk een account aan verbonden zit, of iets met genoeg geld of zo. Ik wist niet precies wat daar gebeurde, maar — De controller die kan Create, Read en ProcessUpdate en misschien een Status, volgens de website maar die zit niet meer in de code.

Q *Ah ja klopt, dat had ik je niet gezegd — als file inmiddels verwijderd zijn of renamed zijn dan toont ie dat niet. Daarom krijg je ook de graph niet meer. Klopt.*

S Naja ik had hier al gezien dat er geen Status meer in stond dus. Dus die heb ik hier even met vraagteken bijgezet. Nou de Controller definieert die dus. Dan heb je het Model, daar zit de representatie van de data in, die dus — dat is dus de huidige status van de TransactionalData. En als laatste heb je de view waar een Create Request naar toe kan sturen om te maken en een update Request elke 5 seconden om te kijken of er een update is.

Q *Yep. — That's it*

S Volgens mij is dat het.

Q *Awesome. Had je hier ook de relaties nog in verwerkt tussen de RequestInquiry en de RequestResponse? Had je daar nog naar gekeken?*

S Oh nee, dat niet.

Q *Had je die wel gevonden? Dat daar een relatie in zat bij* [redacted]

S Nouja, daar hadden we het net al een beetje over dat de Inquiry is volgens mij het Request en de Response is de naja de Response daarop.

Q *Ja, er zit in de IssuerTransactie zit echt een expliciet relatie zeg maar van de bijbehorende Inquiry en de Response.*

S Ja hier, je ziet het wel — ja ik had het daarnet wel al gemerkt inderdaad — dat daar iets mee was.

Q *Oke, cool. Klinkt compleet. Nice. Dan heb ik wat vragen eerst voor je, wat vond je van de assignment?*

S Ja, ik vond het wel leuk. Maar wel een beetje, omdat ik natuurlijk helemaal niks met de code heb gemaakt is het wel even zoiets van 'oh shit dit is allemaal data die je heel snel moet kunnen verwerken'. En aangezien je waarschijnlijk wilt horen of de website daarbij hielp, —

Q *Ehr, dat is wel —*

S Hij hielp wel, dat wel. Het was wel echt handig dat je hier kan zoeken van — weet je, als je met opdracht 2 had gezegd van 'oja, maak dit, enne succes', dan had ik echt geen idee. Maar hier is t van 'oh hier is de feature waarmee je kan kijken hoe het allemaal een beetje werkt'. Maar aan de andere kant is het ook wel zo van, ja ik kan wel heel fijn op een workflow gaan klikken en dan zie ik allemaal dingen, maar, — en dan merk ik ook, heel vaak zie ik CreateTokenQrRequest, maar die zit bij allemaal. En dan heb ik zoiets van 'oke, doet ie dan daadwerkelijk ook iets?' Of 'waar is dat dan voor nodig?'. Dus in dat opzicht is misschien de workflow net iets minder handig, maar het was wel handig om precies het goede bestand te kunnen zoeken.

Q *Ja. Dus je hebt het met name gebruikt om de goede bestanden te vinden in de code?*

S Ja, en dan vervolgens in de code ben ik gaan Cmd-klikken om dieper in te gaan om te kijken hoe alles een beetje werkt.

Q *Dus, om de tool — een verbeterpunt zou kunnen zijn dat je dat klikken niet meer hier zou hoeven te doen, maar dat je basically in de tool kan doorklikken dan.*

S Ja, of misschien zelfs wel dat als je hier overheen hovered dat je dan een beschrijving van — ik weet niet precies wat dit nou is, of dat nou ook ergens in de code gedefinieerd is of niet. Maar als dit bijvoorbeeld een daadwerkelijke methode is in de code dat je dan er overheen hovered en dat je dan een korte uitleg krijgt. Ik merkte ook dat de code niet heel erg gedocumenteerd is behalve —

Q *We mogen geen documentatie gebruiken.*

S Mag je geen documentatie — naja dat maakt het wel lastig. Maar voor een ander bedrijf dan misschien, dat wel documentatie gebruikt zou het heel fijn zijn als je hier overheen hovered en dat er dan staat 'gets Transaction Model doet dit dit en dit' zoiets. Zodat ik daadwerkelijk snap wat er gebeurd. Naja, want weet je, als je zoiets klikt — dit is wel heel logisch van 'als er geen update nodig is, dan ga je gelijk naar het eind'. Maar als er wel een update nodig is, dan gebeurd er weer een hele hoop aan stuff die lijkt alsof elke Request heeft een paar dingen die altijd gebeuren. En dan is er naja bij UpdateStatus is volgens mij alleen deze relevant want de rest is allemaal fluff die er omheen moet zitten om het te laten werken. Het enige blok wat dan echt nodig is, is UpdateStatus. Dat zou dus misschien ook nog een verbeterpunt zijn dat je het kan inklappen —

Q *Deze twee bijvoorbeeld StartTransaction en CommitTransaction dat is puur om een database transaction te starten. Dat je altijd een isolated atomic operation doet op je database —*

S Ja, dan zou je het misschien ook nog kunnen color-coden om de fluff aan te geven om te geven 'hey het gebeurd wel, maar is niet belangrijk en deze methodes dat is echt uniek aan deze request, of deze workflow'.

Q *Nou moet ik zeggen, die Assert enzo zijn op zich wel nodig omdat je hier op zich een soort data validatie doet dat je niet zomaar een workflow — een workflow is eigenlijk gewoon een operatie uitvoeren, maar netjes in een stappenschema zeg maar gestopt.*

S Ja, nee, dat snap ik ook wel, maar als je bijvoorbeeld probeert te vinden hoe iets werkt, dan is die assert misschien wat minder belangrijk.

Q *Ja, fair.*

S Want nu zocht ik gewoon naar een manier om bijvoorbeeld bij de User te komen, dus ik had hier eigenlijk gehoopt dat ik hier ergens had kunnen vinden bijvoorbeeld — naja wat we ook met hover zeiden dat je bijvoorbeeld kan zien wat voor data er dan daadwerkelijk in die UpdateStatus zit. Zoiets. Dat gaat wel heel ver dan.

Q *Ja, je had als het goed is, ik denk dat het deze is DetermineSenderAndReceiver, dat hierin op een gegeven moment User —*

S Ah ja hier zo.

Q *Ja. Daar zit het dan wel ergens in verstopt. Maar ik ben met je eens — dat was ook wel deels de assignment, van 'hoe diep kan je komen/hoever kom je?'. Maarja, je had het heel ver weggestopt had je het — naja je hadden RequestInquiry gevonden uiteindelijk.*

S Ja omdat ik natuurlijk wel vaker aan het debuggen ben via de IDE was dat ook wel — was dat ook eigenlijk wel mijn eerste idee om gewoon zo snel mogelijk het bestand te zoeken en dan alles via de IDE te doen. Maar ik heb uiteindelijk ook nog wel de website een paar keer gebruikt.

Q *Ja. En had dat dan nog meerwaarde om die website nog te gebruiken of had je zoiets van 'naja ik deed het nu voor het experiment maar ik had het net zo goed, net zo snel in de editor kunnen doen'?*

S Nou zeker in het begin om het bestand te vinden dat had ik niet graag via de editor willen doen. Dat was wel echt meerwaarde en dan later ook bij de derde assignment om — naja gewoon een heel kort overview van de model view en controller te hebben was het ook wel heel erg chill. Omdat je hier natuurlijk heb je wel drie packages helemaal uitgeklapt om alles te bekijken. Maar zo is het wel chill om te kunnen zien — naja omdat je natuurlijk die transaction view hebt, en dan heb je hier transaction model — ja hier. Dus dan kun je goed zien wat er allemaal bij elkaar hoort en wat je een beetje kan wegstrepen omdat het bij opdracht 3 niet meer relevant was omdat het een extensie erbovenop is of wat dan ook. En ook daadwerkelijk dat je dan kan zien dat de QR code ook nog belangrijk op de een of andere manier is. Dat is ook wel handig.

Q *Ja. Ja, het is de dezelfde flow he wat ik je liet zien met hoe je die QR code krijgt die je moet kunnen scannen die moet wel gegenerate worden. Dus ja dat is belangrijk —*

S Ja, dus dat had wel meerwaarde in ieder geval —

Q *Ja. Oke. Fair. Je hebt de Find By Label heb je gebruikt —*

S Ja.

Q *Heb je daar — heb je 'm gebruikt, en if so, why or why not?*

S Omdat het me vertelt werd om te gebruiken.

Q *Maar heb je hem alleen daarom gebruikt?*

S Ik zat er nog wel aan te denken om hem te gebruiken, bij opdracht 2 om iets van een user te vinden. Maar toen dacht ik van 1, ik denk veel te moeilijk en 2, ik weet niet waar ik op had moeten zoeken. Want ik had wel feature — iets van Feature User of zo —

Q *Je kan gewoon 'User' typen.*

S Oh, ik had gewoon 'User' kunnen typen. Maar dan dacht ik van maar dan krijg ik misschien een generiek User object en niet specifiek wat ik nu nodig heb, dus dat dat lijkt me te ingewikkeld om dat te gebruiken.

Q *Ja, oke. Waren er hier nog informatie dingen die je mistte of die overbodig waren of?*

S Ehm, ja hebben we het wel al een beetje over gehad denk ik. Dus dat de — Nee, naja zoals ik al zei dat de workflow kan misschien nog cooler zijn maar is sowieso wel een beetje handig. En verder ben ik vooral nieuwsgierig gewoon waarom dit allemaal miscellaneous is.

Q *Er zit een Regular Expression op, op de path van een file, en daarop trek ik ze uit elkaar en als ik niet een match heb dan wordt het allemaal daarin getost —*

S Ja, naja, dat — misschien dat dan nog uitgebreider maar weet je dat is ook zo. Dat dit al gesplitst is is al best wel cool.

Q *Het was eerst gewoon echt een lange tree zonder filtering.*

S Oh ja, dit, dit is — dat is het enige waar ik — ik had zelf ik had zoiets van — want ik klikte hier dus op en ik had zoiets van 'is mijn internet kapot, het is een oude laptop, werkt ie nog? [?], oke mijn internet is niet kapot'..

Q *Ja ik vergeet het steeds te zeggen, het is echt annoying. Maar wat er gebeurd, ik ga zeg maar over de git history heen en dan kijk ik dus steeds naar file changes, dus welke bestanden zijn aangeraakt, maar ik kwam er later ook achter inderdaad dit. Maar om dan in te bouwen nog dat ie de file status gaat bijhouden met de removal/update/weet ik veel wat. Dan moet je dus ook bij gaan houden van op dit punt werd ie toegevoegd, toen verwijderd, toen werd een bestand met dezelfde naam toegevoegd en dan moet je dus date filtering — toen had ik zoiets van ik laat het wel staan. Ik vind het wel goed. Het meeste blijft toch staan.*

S Oh enige nog, als je een heel groot ding hebt, dan is het echt kut om hem te sluiten, omdat je helemaal op de rand zo..—

Q *Nee, geen kruisje.. ja fair.*

S Dat is ook iets heel kleins.En je kan ook nog gecentreerd doen en allemaal styling.

Q *Je kan Esc doen. De Find By Developer heb je die nog kunnen gebruiken?*

S Alleen voor opdracht 1. Voor opdracht 2 bedenk ik me net dat ik misschien jou naam in had kunnen typen omdat ik zag dat jij dit allemaal had gedaan, dat ik dan misschien wat had gevonden. Maar aan de andere kant ik zat al helemaal in het [redacted] ding, dus ik had zoiets van naja weet je, heeft dat nog wel meerwaarde dan voor mij om nu iets te zoeken wat ik nodig heb?

Q *Ja, kan je je iets indenken waar het wel nuttig zou kunnen zijn? Of zeg je van, ik zou dat niet snel gebruiken?*

S Als een persoon iemand moet benaderen om een feature die niet werkt. Nee ja, in mijn geval denk ik niet. Behalve als ik, — wat je nog misschien zou kunnen hebben is dat je hier ook nog de auteur van het ding hebt. En dat je dan kan zien van 'oh deze persoon heeft die die en die gemaakt, misschien nog even kijken of hij nog wat anders heeft gemaakt wat er in de buurt is'. En dat zou wel —maar dat — in deze opdracht had ik het niet nodig.

Q *Oke. Zelfde vraag voor de Feature By CIT.*

S Zelfde antwoord. Zo ongeveer. Ik heb het gebruikt. Maar hier zie je weer niet — ja je ziet hier wel de issues inderdaad, en daar had ik ook nog wel even naar gekeken. En, ik zie net dat hier misschien had kunnen gebruiken om het goede bestand te kunnen vinden maar het was toch heel makkelijk om via de features het goede bestand te vinden met de hint ook — als hier geen view had gestaan dan had ik misschien inderdaad naar die issues gaan kijken om te kijken welk bestand er mee te maken had. Maar omdat hier dus al view al hint staat was het niet meer nodig. Want toen wist ik gelijk 'oh hier, kijk naar views, oh bij transaction, de Response en het zijn nog maar twee bestanden, ah top'. Je hebt dan nog maar een definition over, dus dan wist ik gelijk waar het was.

Q *Ja. Was de link tussen definitions en de generated classes een beetje duidelijk voor jou?*

S Ja. Dat, ja behalve die ene bug was het gewoon; je maakt dit en een hele hoop bestanden worden vanzelf aangepast. Ik zag ook gelijk dat als ik hier op generate drukte dan ging ik naar de goede — volgens mij is het deze, deze? — ja hier werd ie dan gelijk als const toegevoegd.

Q *Ja, klopt.*

S Ik wist niet helemaal natuurlijk als description en minified en documentation enzo maar —

Q *Nee, ja, ik had bij sommigen dat mensen voor minified gingen zoeken in alle andere waardes van minified; nou die lijkt er het meest op die copy paste ik wel. Terwijl het eigenlijk unique values moeten zijn.*

S Ja, dat dacht ik ook al want ik zag dat ze allemaal gewoon een afkorting van de andere was.

Q *Ja, precies. Gewoon om de JSON klein te kunnen maken mocht iemand dat willen. Volgens mij is het überhaupt deprecated maar, — Voordelen van de tool, if any, die je nog niet hebt genoemd?*

S Ik denk dat ik ze allemaal heb genoemd, het is wel fijn om in zo'n grote codebase om heel snel te kunnen zien van dit werkt zo, en dat werkt zo, in plaats van dat je door een paar duizend packages moet gaan scrollen om de goede te vinden. Want als ik hier had

moeten zoeken, ja dan had ik waarschijnlijk in de IDE helemaal naar boven gegaan in bunq en dan rechtermuisknop en dan Find In Path en dan had het waarschijnlijk heel lang geduurd en hier is het wel fijn met autocomplete ook dat je gewoon [redacted] typt 'oh hier heb ik al alles wat ik nodig heb'.

Q *Ja, oke, zijn er nog dingen die anders moeten die je nog niet hebt genoemd? Die je mist, dat je zegt 'dit zou een nuttige toevoeging zijn voor mij'?*

S Nee, ik heb alles al genoemd. Workflow beter en misschien hier dat je hier overheen hovered of hiernaast nog de auteurs zet, of dat je issues kan zien waar het aan gelinkt is ooit of zoiets.

Q *Dat zijn deze right?*

S Ja, oke, maar niet dat is niet specifiek per file. Dus je kan hier niet per file, want volgens mij staat hier bij elk bestand staat wel wie de auteur is.

Q *Ja de originele auteur.*

S Ja, wat nog echt heel cool zou zijn, maar dat is volgens mij onmogelijk, is als je hier op het bestand klikt en dat dan je IDE vanzelf je IDE het goede bestand opent.

Q *Ja dat zou inderdaad heel cool zijn, ik weet inderdaad niet of dat mogelijk is.*

S Ik denk niet dat het mogelijk is maar nu was het wel zo van ik had zoiets van 'ah cool, ik heb het goede bestand gevonden scroll scroll scroll scroll, oke'

Q *Kende je* ⇧ + ⇧ *in deze editors?*

S Nah.

Q *En dan kan je gewoon letterlijk de eerste letters, oh je weet al —*

S Ja, maar dat is te fancy voor me.

Q *Ik vind het echt een gouden feature.*

S Ja, maar als iemand daar niet van weet dan — en alsnog het zou gewoon echt heel cool zijn zo van 'klik' en dat dan vanzelf IntelliJ opent.

Q *Ik weet niet eens of het kan eigenlijk.*

S Ik denk niet dat het kan.

Q *Misschien met een plugin in je browser. Want je moet local filesystem iets draaien.*

S Dan moet IntelliJ het ook nog supporten.

Q *IntelliJ support het. Want je kan zeg maar, als je een terminal erbij pakt, als ik nu zeg pstorm, dan bam. Dat kan.*

S Oh.

Q *Daar zijn kant en klaar scripts voor. Dit komt uit de editor.*

S Oh, nou dan misschien dat het dan wel kan inderdaad. Maar je kan volgens mij via JavaScript gewoon de command line aanroepen.

Q *Weet ik niet, dat lijkt me best wel een security flaw.*

S Ja, maar JavaScript is een grote security flaw. Nouja, dat zou dan nog wel de coolste feature zijn, maar dat is wel echt een maybe possibly should have.

Q *Someday. Cool. Heb je in general nog feedback op de tool, iets dat je kwijt wilt?*

S Nee, ik vond het cool om hem te gebruiken. En ook heel snel, dat is ook wel cool dat je hier echt — overduidelijk is dit allemaal een keer gegenerate en is ie dat niet on the fly aan het generaten.

Q *Nee. Nouja, het is een graaf database zeg maar. En die queried ie wel ad hoc, maar de database is wel van te voren —*

S Nee, ja, precies. Maar het is wel cool dat ie gewoon heel snel is.

Q *Je moet voor de gein eens 'App: bunq' openen. Dan doet ie er iets langer over. Dat is zo'n generiek label zeg maar. Dan duurt het iets langer zeg maar.*

S Oh, dat valt ook nog wel mee.

Q *Maar hij trekt er een hoop informatie uit. Het is basically de hele backend wat je nu terug krijgt.*

S Wat nog wel mee aan views dan eigenlijk.

Q *Ja, views die matchen. Als ze niet exact de regular expression volgen dan — maar dan is het op zich weer een voordeel dat zo'n strikte naam policy heb want dan kan je gewoon met een regular expression zeggen 'dit is een view' en anders niet.*

S Ja, nee dat klopt.

Q *Wat is je indruk van de backend als je hem zo ziet?*

S [redacted]. Het is al allemaal snake case, dus allemaal met underscores, en maak daar gewoon allemaal losse packages van. Maak gewoon een package admin, een package core, billing en dan ga je gewoon steeds dieper. Want alles — je kan wel zeggen oh nee we vinden subpackages niet leuk, fair enough, maar als je dan vervolgens naar — vervolgens heeft elke package alsnog wel subpackages. Dus dat is dan geen argument meer dat je geen subpackages wil gebruiken. Want ik zie oprecht niet de meerwaarde er van om dit — m'n punt is, alles is al gescheiden. Er zijn al

— er is al een hele duidelijke structuur van 'ah hier zie je in bunq core [redacted] transaction en dan model view controller'. Dus waarom dat niet allemaal in losse subpackages zit dat ga ik niet begrijpen zelfs als er een reden voor gegeven wordt.

Q *Ik heb het idee dat het van de oude structuur komt. Vroeger was de core repository alle submappen waren elk een nieuwe git repository. En dat hebben ze op een gegeven moment gemerged in een grote repository. Maar vroeger als je dus een view, een controller en een model aanmaakte moest je dus drie merge requests aanmaken voor elk van de repositories basically.*

S Dat is ook gewoon niet sustainable volgens mij.

Q *Daarom hebben ze het nu ook gemerged.*

S En vervolgens zie ik ook dat bestanden geen documentatie hebben dat slaat ook nergens op. Ik bedoel — ja, als —

Q *Self-documenting..*

S Ja, dat zal best. Kijk, dat code, methods zichzelf moeten uitleggen dat snap ik. Maar ik vind wel dat elke klasse een korte beschrijving moet hebben. Want je kan t iets zoals [redacted]TransactionViewPostProcessor noemen, maar dan weet ik nog steeds niet wat er in die klasse gebeurd. Behalve dat ie een [redacted]TransactionView post-processed. En dan misschien geen documentatie maar weet je, genereer dan hiervoor documentatie — oh, trouwens, dat is misschien wel nog een feature die je kunt toevoegen. Als je hier naar Feature [redacted] gaat en daarop zoekt dat je dan hier een kleine summary hebt van wat een feature doet en waar die voor gemaakt is.

Q *Ah, zo. Ja.*

S Dus dat je — je hebt hier de labels, developers en files dat ja daarboven of ertussen summary met [redacted] handelt alles wat met [redacted] te maken heeft het doet dit dit en dit, het heeft met deze klassen te maken en is gebaseerd op iDeal. Ik zeg maar wat. Maar zoiets, weet je wel.

Q *Ja.*

S Want dan heb je in ieder geval — want ik me echt voorstellen als je hier net begint dat je dan echt iets hebt van 'what the f***' gebeurd hier. Dat je gewoon niet begrijpt van wat — de structuur nouja dat kan je nog wel begrijpen, zeker met de website [tool], maar dat je daadwerkelijk moet begrijpen hoe het systeem in elkaar zit dat is echt — dat lijkt me echt heel lastig. En je zal wel niet met alles te maken hebben dus je hoeft ook veel niet te weten —

Q *Maar de pest is wel dat als je iets aanraakt dat met iets anders in aanraking komt ver weg dan snap je dat verband vaak niet.*

S Ja, precies. Wat jij ook net zei, met die link tussen de Inquiry — die had ik wel al gezien maar het staat niet expliciet ergens in de code daadwerkelijk. Je kan wel zien dat de ene afhangt van de ander maar er staat nergens dat zonder deze gaat code aan de andere kant stuk.

Q *Ja, dependencies.. Ja, fair. Denk je dat het mogelijk is met deze tool om features te implementeren danwel verbeteren maar dan zonder de hulp van een senior basically? Of met minder hulp van de senior.*

S Als het makkelijke features zijn wel.

Q *Waarom denk je dat?*

S Naja, wat wil je — de laatste opdracht die we hebben gedaan, of die ik heb gedaan, dat was gewoon bijna letterlijk copy-paste. Sterker nog, ik denk dat ik dit binnen een twee uur bijna had kunnen implementeren ook. Omdat het gewoon allemaal copy-paste en je veranderd wat namen en dan test je of het werkt. Maar als het echt iets is van 'maak betalen van de ene telefoon naar de andere telefoon via NFC' ik weet niet of bunq dat nu doet, maar waarschijnlijk niet.

Q *Nee.*

S Maar als je tegen een junior developer die inderdaad 2 maanden er zit zegt van 'ga dit maar maken' dan lijkt me dat heel lastig. Omdat je dan niks hebt om — het lijkt me dan ook niet dat de website heel handig is want je kan dan misschien wel randdingen die je nodig hebt, geld opvragen en geld versturen, maar het midden, het meest belangrijke daar kan je dan helemaal geen houvast op vinden. Dus het lijkt me vooral dat je de structuur begrijpt en dat je dingen kan verbeteren en misschien wat dingen aan de rand van die je nu al hebt. Maar als je echt een heel nieuwe feature gaat maken lijkt het me dat deze niet heel handig voor je is.

Q *Want dan zou je —*

S Want je kan dus wel — ja, als je het zeg maar als eilanden ziet, dus als je nu zo bunq hebt en je probeert nu een nieuw eiland aan te maken, dan kan je dus wel wat rand delen kan je hiermee vinden. Maar dan het echte nieuwe dat bestaat dan natuurlijk nog niet. Dus dat zit ook niet in het systeem dus je kan er ook niet naar zoeken. Dus dan is het systeem nutteloos, maar dat is ook logisch.

Q *Zou je het systeem denk je kunnen gebruiken om dan een beeld te vormen van hoe alles is opgezet en hoe je dan ook jou project zou moeten opzetten?*

S De structuur wel qua model, view, controller maar — en de subpackages en al die shit, maar ik denk dat je ook misschien nog wel wat andere dingen mist — ja, je kan dus wel de opzet zien, maar je kan dan hier in ieder geval niet de code zelf in zien. Dat kan je dan via je IDE zien, en ja, in dat opzicht wel. Dus je kan wel structuur zien maar niet — maar ik weet niet of je ook via, voor elk feature weer een andere soort structuur hebt.

Q *Nee. De structuur die is overal hetzelfde.*

S Ja, oke. Maar bijvoorbeeld weet je, als je een nieuwe webcall moet implementeren dan is het model, view, controller. Als het een nieuwe — weet ik veel, een nieuwe currency moet implementeren dan moet dat toch weer anders denk ik.

Q *Ja, ja, fair.*

S Maar dat kan je ook weer opzoeken neem ik aan. Even als voorbeeld. Maar dan — ja, dus je kan het wel opzoeken maar nieuwe feature gaat toch wel echt iets te ver denk ik.

Q *Ja, oke. Duidelijk. Wat generiekere vraag, wat zijn volgens jou de meest belangrijke informatiebronnen als je een bestaande codebase, grote bestaande codebase of framework moet leren? —*

S StackOverflow. Als ik een nieuwe codebase moet leren, ehm, sowieso iets van git, als er issues zijn is het heel fijn en anders ja als ik dan even zeg hoe ik dan — dan loop ik gewoon een beetje door het project heen, kijk ik wat — als ik iets specifieks nodig heb dan zoek ik naar het bestand en dan kijken wat ik verder nodig heb. Als het heel ver gaat dan open ik het in de IDE en dan ga ik met ⌘+Click ga ik helemaal het bestand — alle bestanden doorheen lopen om te kijken of ik het nodig heb.

Q *Doe je dat nog met een bepaalde structuur, dat er doorheen lopen?*

S Meestal heel gericht, want als ik bijvoorbeeld een nieuw framework gebruik — bijvoorbeeld op m'n werk, naja dat is als voorbeeld daar moest ik hun systeem gebruiken om in te loggen voor een tijdje en dat ging toen kapot maar ik wou niet steeds aan m'n collega vragen om hulp dus ben helemaal dat bestand in — dat hele project ingegaan. Dat was niet zo groot als dit overigens maar. Dat hele project ingegaan om te kijken wat ik precies nodig had. En dan was — ik wist iets met Authentication dus ik zocht naar Authentication en dan ging ik er steeds dieper in totdat vond wat er fout was. Maar dan ook met testen uitvoeren en `var_dump`s en al die shit. Dus zo, weet je, ja ik vind het wel heel fijn om in een IDE steeds door te klikken naar nieuwe bestanden en dan op die manier een beetje de relaties te vinden. En naja issues op GitHub of git wat dan ook, is altijd, dat helpt.

Q *Wat voor nuttige informatie haal jij dan voor jezelf uit git, de issues?*

S Nou meestal ga ik — nouja dat is meestal als ik een probleem heb met een framework. Daar kan je, gewoon omdat het iets nog specifieker is dan StackOverflow, dus iemand kan tegen je zeggen van 'oh ja ik wil weten hoe ik een decorator op iets moet gooien, kan dat?' En dat dan de developer daar een kleine tutorial in geeft weet je. Dan leer je ook weer heel veel over het framework. Dus het is meer gewoon de antwoorden van de developer die zijn meestal heel — of mensen die het veel gebruiken — die zijn meestal heel uitgebreid en daar leer je weer meer van het framework over. En of ze open of gesloten zijn dat maakt niet uit.

Q *En in het geval van bunq, wat zouden dan jou ideale informatiebronnen zijn? Hoe zou je dat het liefst leren?*

S Ik denk dat ik dan eerst misschien heel globaal schema zou willen van hoe alle systemen met elkaar werken. Want ik zie, je hebt het core systeem blijkbaar en je hebt het admin systeem — ik neem aan dat die allemaal met elkaar verbonden zijn op een hele ingewikkelde manier. Dus dan misschien eerst een heel globaal systeem en dan afhankelijk van waar ik dan zou gaan werken een gedetailleerd schema van dat en dan misschien nog met die site of anders een IDE daar een beetje over leren en ik neem aan dat bunq ook wel iets van een document heeft waar ze [?] en code standaarden hebben gedocumenteerd.

Q *Mja, de coding style guides hebben we wel.*

S Nee, oke, maar zo leer je het. En misschien dan gewoon een beetje dingen gaan proberen. Zo van voeg een gebruikersnaam toe aan een transaction.

Q *Ja. Denk je dat het een nuttige oefening is voor nieuwe recruits binnen bunq om kennis te maken met het framework?*

S Ja, ik zou er wel nog wat meer bij doen, maar het is wel een goeie start in ieder geval.

Q *Wat voor dingen zou je er nog aan toevoegen?*

S Nah hier is het natuurlijk 'voeg iets aan een view toe, of aan een request'. Ik zou er gewoon nog iets bij doen van 'maak een makkelijke request die Hello World — weet je wel, desnoods Hello World toevoegt'. Maar dan weet je in ieder geval hoe zo'n Request — want nu zag ik dat er een request was en ik zag hoe die gemaakt was en hoe ik hem moest updaten maar als je nu tegen mij zou zeggen 'oh maak een nieuwe request' dan zou ik daar echt best wel lang voor nodig hebben. Da's natuurlijk altijd zo, updaten is altijd makkelijker dan het maken, maar dat lijkt me wel een goede oefening alsnog — want dan leer je — nu wist ik dat er model view en controller was maar de model en de controller heb ik eigenlijk niet nodig gehad. Dus eigenlijk alleen de view. En ik heb de view geüpdatet en een regel code toegevoegd maar dan alsnog mis je dan misschien toch nog die stap om het je echt eigen te maken om daadwerkelijk nog helemaal zelf iets toe te voegen.

Q *Ja. Oke. Duidelijk. Dat waren mijn vragen. Heb jij nog vragen aan mij?*

S Nee, die heb ik niet.

C.19 Subject 19

Questioner (Q) *Oke, kan je me vertellen wat je hebt gedaan?*

Subject (S) Ik heb uiteindelijk de — vooral de PreProcessor en in de PostProcessor dingen aangepast. Ik was begonnen bij de PostProcessor. In het ResponseObjectBody functie. En, even kijken hoor, daar heb ik dit aangepast —

Q *Ja.*

S Die ik uit de UserBase haal, ik heb hem gerunned, het geeft geen errors maar of dat betekent dat ik ook alles uit de goede plek haal dat weet ik dan niet precies. Die had in elk geval een displayName. En hij had ook nog een nickName enzo maar ik nam aan dat name of the user, displayName, dat dat oke is. En anders is dat zo aan te passen I guess, als je een andere specifiek wil uit die —

Q *Ja, displayName is in principe de publieke zichtbare naam dus dat is prima.*

S Oke, ja. En daar heb ik deze functie toen voor aangemaakt. Aan de hand van eigenlijk vooral de description denk ik. Even kijken hoor, dit is in —

Q [redacted]*Transaction —*

S Base. Ja. Ja, daar heb ik dus een beetje — even kijken hoor, ik weet niet of dit vanuit die description gehaald is. Ik heb alleen de get trouwens gedaan niet de set en de unset, maar die zou je natuurlijk op dezelfde manier kunnen doen. Ik heb alleen het idee dat het voor deze specifieke lijn aan dingen niet direct nodig was. Ja, — toen heb ik deze nog aangemaakt, dat is een protected variabele. Verder niks meer mee gedaan.

Q *Nee ja, dat is prima, dus je hebt in de Base class heb je een field toegevoegd, getter toegevoegd en die gebruik je dan.*

S Ja.

Q *Even kijken, en dit is in* [redacted]*TransactionBase. Yes.*

S Ja. En volgens mij is dat — even kijken hoor wat ik gedaan had hiervoor hoor — deze bestond al in UserBase dacht ik, oh nou dat is dan een fout —

Q *Ja, dat komt omdat UserBase niet geïmport is. Dus als je even hier op UserBase gaat staan en dan* ⌥ + ⏎ *of* Alt + ⏎ *dan kan je zeggen 'import' — eh, should not be called statically — Ja, oke er zit nog wel ergens anders een mismatch in maar dat maakt verder niet uit.*

S Ja. En daarna ben ik in ieder geval naar de PreProcessor gegaan. Even kijken of die hier open staat. Dan zal ik die even opnieuw moeten openen. Nee, volgens mij is blauw wat er waarschijnlijk veranderd is.

Q *Ja. Blauw is changed files.*

S Even kijken hoor, toen was ik hier begonnen, ook weer de userDisplayName veld aangemaakt in TransactionCreateBase.

Q *Ja.*

S Naja, die zo genoemd. En even kijken hoor, en deze die parseDisplayName aangemaakt. Ja, die gebruikt dan weer die field die je net aangemaakt hebt. En dat is een beetje wat ik gedaan heb volgens mij.

Q *Oke, cool. Hoe had je die UserBase gevonden? Waar je die getDisplayName van probeerde te gebruiken?*

S Ik was eerst hier gaan scrollen. Naar `core_user`, maar daar zag ik een hele hoop dingen staan, vervolgens ging ik hierin terug lezen — ik had ook even naar dat dit verhaal gekeken —

Q *Ja.*

S En toen zag ik hierzo staan "lastly there's a folder `core_user_model/code`' which contains the actual UserModel class". En toen kwam ik er dus achter van 'oke ik moet in die klasse kijken'. Dacht ik in ieder geval. Ik weet niet of dat de goeie is nog steeds maar het zijn super veel dingen die hierzo staan.

Q *User is een vrij centrale spil in de backend natuurlijk.*

S Ja, ja want ik zat eerst te kijken of er al andere velden in de [redacted]Transaction stonden die iets met Users te maken had, maar dat was nog helemaal niet het geval volgens mij. Eh — Waar is ie gebleven? -

Q *Wat was het?* `core_user_model` *nog iets, wacht even, 'L', 'M', hier. UserModel.*

S Ja, toen heb ik eerst hier gekeken, UserModel. Beetje gekeken wat voor dingen erin stonden. En toen zag ik ergens een displayName staan. Hierzo zag ik een displayName staan.

Q *Aah, ja.*

S Vandaar dat ik die — wat is het —

Q *Dit is method getDisplayName — heel eventjes kijken hoor waar dit voor gebruikt wordt normaal. Voor een hoop. Ah, voor de ACL methods oke. Dus zo heb je die gevonden. Oke, creatief. Ik had het nog niet gezien op deze manier.*

S Ja, ik weet niet of ie werkt dus want ik vind het niet heel duidelijk wat hier nou uiteindelijk uit moet komen.

Q *Nee ja, hij zal nu niet direct werken zoals je hem hebt aangeroepen maar je komt ook wel een aardig eind ermee denk ik. Hoe was je op het idee gekomen om het UserModel überhaupt te gebruiken? Want je bent gaan zoeken naar* `core_user_model`*, maar ..?*

S Ja, ik was op zoek naar natuurlijk die username zien te vinden ergens, en ik zag het in die [redacted] dingen niet staan en ik moet hem ergens vandaan halen. Dan mis ik alleen nog een beetje de link tussen welke User ik nou ophaal.

Q *Ja.*

S Maar ja. Want volgens mij gaat dit niet werken zo. Maar naja — dat was eigenlijk een beetje. Ik heb nog wel in die database gekeken maar daar stond ook geen veld ofzo voor. In de [redacted] —

Q *Wat had je in de database bekeken?*

S Ja, die [redacted]Transaction die die gebruikt.

Q *En wat had je hier wel gevonden dan —?*

S Ja, verder had ik hier niet meer gekeken.

Q *Oke.*

S Alleen even deze velden gekeken.

Q *En hoe was je bij deze PreProcessor PostProcessor uitgekomen?*

S Nou je moest natuurlijk in die existing endpoint, de view, moest je iets aanpassen. Even kijken hoor, dus ik ging volgens mij dus in die view kijken en daar zag ik die Pre en Post processor en even kijken hoor en met dit verhaal [assignment] dat dit de uiteindelijke response genereert. Dus vandaar. De workflow is volgens mij geen — dat is meer hoe dat werkt in de code en dat had ook niet een eigen code klasse ofzo. Volgens mij.

Q *Ja, je hebt de controller map, dus je hebt* `core_[redacted]_transaction_controller`*, daar zitten de workflows in.*

S Ja. Dus misschien dat ik — even kijken hoor, had ik dat gechecked — want daar heb ik uiteindelijk niks aangepast. Dat leek meer om de Transaction zelf te gaan, het proces. En iets minder over de inhoud. Dus daarom heb ik toen niks aangepast.

Q *Maar deze heb je gevonden door te gaan scrollen eigenlijk?*

S Nouja [..?] wist ik dus de endpoint name [redacted]-transaction, toen ben ik gewoon gaan scrollen naar de [redacted] [redacted] Transaction gedeelte.

Q *Oke, cool. Duidelijk. Dan de volgende opdracht is design ding meer. Dan zou ik je ook willen vragen probeer de tool te gebruiken. Want je bent nu gaan scrollen, dat is op zich prima, maar dan ben ik straks ook wel benieuwd van waarom je dat hebt gedaan en niet dacht van 'oh misschien is de tool nuttig', want —*

S Deze tool?

Q *Ja.*

S Oh ja. Ik had het idee dat het vooral de branch tracking en de issue tracking was en niet perse de voor de code zelf zeg maar. Dus daarom eigenlijk.

Q *Ja oke, dus welke informatie mistte je voor je gevoel dan?*

S Dit gaat alleen maar over de feature namen en zo en de labels die daar aangegeven zijn. En ja, dat was voor mijn gevoel nu in ieder geval niet echt — ja, ik bedoel ik ben natuurlijk wel bezig met een issue I guess, maar dit was helemaal niet gelinkt aan een CIT naam of wat dan ook.

Q *Ja, oke. Ik zie wat je bedoelt. En als je probeert te zoeken hierzo by Label op deze [redacted], dan ben ik wel benieuwd als je die informatie ziet of je daar wat mee kan.*

S Kan sowieso nog — toen ik bij Feature By CIT zocht heb je een knopje 'visit in new tab', deze —

Q *Ah, ja daar zit een bug in.*

S Die werkte niet.

Q *Dat was ook de eerste deelnemer die zei ook 'hey, kijk er zit een bug in' en toen zakte ik al door de grond van 'oh shit, zal je net zien..'.*

S Ja, ik zag hem ook in ieder geval. Maar je wist er dus van. Even kijken, was dit nou de goede.

Q *Je zit nu in [redacted]Transaction of wat is het, hoe heet ie, [redacted] ja. Oh nu is er geen eentje — dat is ook een kleine bug trouwens dat soms de toggle uitvalt, vraag me niet waarom. Kijk want hier zie je bijvoorbeeld ook die processors weer terugkomen.*

S Ja.

Q *Nu je dit ziet, heb je dan zoiets van 'oh dit had handig geweest' of had je zoiets gehad van 'ik had alsnog gaan scrollen in de editor en dat vind ik fijner'?*

S Ik had zo wel preciezer iets gevonden denk ik, maar uiteindelijk denk ik dat het weinig uitmaakt. Alhoewel ik misschien hier ook wat meer bij deze had gekeken eigenlijk. Gewoon even kijken wat er in stond dan enzo. Dat heb ik nu natuurlijk niet gedaan. En die RequestResponse view heb ik ook niet bekeken dus dat zou ik dan ook wel even gedaan hebben. Dus wat dat betreft zou je wel wat preciezer wat meer houvast hebben zeg maar.

Q *Ja.*

S Ja, ik deed het — ja die Features dat had natuurlijk het keyword moeten zijn.

Q *Ja, daar had je het mee kunnen vinden inderdaad.*

S Maar op zich hebben die `core` zo, op zich is dat wel een logische naam die dat heeft. Dus op die manier kan je hem al goed vinden.

Q *Ja, dat blijkt want je hebt ze gevonden dus dat —*

S Ja, alleen is het wel inderdaad is het op deze manier heb je wel iets meer houvast denk ik.

Q *Ja, want in principe zijn dit alle bestanden die zijn aangeraakt om deze feature te kunnen implementeren. Dus die zijn er op de een of andere manier ooit bij betrokken geweest. Zij het door een foutje of zij het gewoon echt bewust.*

S Dat gebeurt ook wel eens dat ie —?

Q *Naja, als jij bijvoorbeeld — zeg maar, die links waren gemaakt met tags in de commit message, right? Als je daar een typefout in maakt of zo dan worden dingen dus nu onterecht geassocieerd met een feature. Dus dat is —*

S Ja oke.

Q *Nee oke, maar dan zou ik voorstellen dat je doorgaat met het volgende deel. Dan nu probeert de tool te gebruiken ook een beetje. Ja en zeg je van 'dit helpt mij gewoon echt niet ik doe het liever in de codebase' dat is ook prima. Maar dan ben ik erg benieuwd natuurlijk waarom.*

S Ja oke. Dan ga ik even kijken wat deze opdracht is.

-

Q *Oke, wat heb je gevonden?*

S Ik heb in eerste instantie naar de [redacted] feature gekeken. Om te kijken —

Q [redacted] *ja.*

S Om te kijken wat er geïmplementeerd moet worden, want — iDeal staat niet uitgelegd hier maar [redacted] lijkt wel heel erg overeen te komen met de AliPay transactions. Want later heb ik nog geprobeerd de feature van iDeal te vinden maar dat was een stuk vager, want incoming bestaat niet. Die geeft in elk geval een No Data Found error. En er is geen payments of zo voor iDeal. Ik vond dat een beetje vaag. Dus ik heb het vooral op de [redacted] feature gebaseerd. In principe moet het gewoon aan het MVC moet het voldoen. Dus je hebt models nodig. En dan moet je gewoon de AliPay versies van maken van dit I guess.

Q *Yes, en welke models heb je nodig dan denk je?*

S Je hebt natuurlijk die QR ding heb je nodig. Omdat je die ook gebruikt. En je hebt dus daar dat QrRequestModel nodig, de TransactionModel is gewoon de model van natuurlijk van de [redacted] in dit geval maar daar zal je dan een AliPay versie van maken.

Q *Ja.*

S Response model heb je nodig, maar dat is het algemene RequestResponseModel. Dus zal je in die RequestResponseModel zal je daar waarschijnlijk een functie moeten aanpassen om daar een AliPay versie van te maken. MessageModel was me niet helemaal duidelijk wat dat deed. Ik heb net even in de code gekeken maar dat werd niet veel duidelijker. Dus ik weet niet zeker of dat iets van [redacted] alleen is of dat je dat ook moet implementeren.

Q *Dat hoort bij het [redacted] protocol. Dus ja, dat kan je hier negeren inderdaad.*

S Ja oke. Ja en hetzelfde geldt voor het Certificate model die zal vast wel iets van certificaat nodig zijn om dat, hoe zeg je dat, te verifiëren. Maar hoe dat dan precies werkt bij die AliPay dat is me niet helemaal duidelijk maar ik denk dat daar ook een versie van moet zijn. Dan heb je dus View's. Je zal hier waarschijnlijk een reference moeten maken in de BunqMeRequestViewPostProcessor naar een AliPay instance. Je hebt een RequestResponseView, zelfde verhaal. En dan moet ja natuurlijk de AliPayIssuerTransactionView moet je dan ook weer maken. Zelfde geldt voor de QR, daar moet je dan een — ze staan hier als aparte files per payment methode dus daar moet je dan ook een nieuwe file voor aanmaken. Ja, een beetje hetzelfde idee voor de controller. Ben ik ook op die manier langs gegaan. En een beetje gekeken naar hoe dat voor [redacted] gedaan werd en gekeken of daar dan zo'n zelfde iets voor gemaakt moest worden. Ja, dus daar moet je een soortgelijk iets doen. — Ho, dat wilde ik niet, ik weet niet waar ik op klikte maar.

Q *Je klikte op de link. Had je dat in de eerste assignment gezien dat je daar op kon klikken en dat je dan een visualisatie kreeg van de controllers?*

S Nee. Nouja, ik kwam er per ongeluk op maar kwam er niet achter waardoor dat kwam. Ik heb er ook niet echt verder naar gezocht. Maar ja nee, het was me niet duidelijk dan in principe. — Nu ben ik wel benieuwd.. — Dit is hoe ie er doorheen gaat?

Q *Yes, exactly. Nou was dit een vrij straight-forward conditional jumps en dan ga je dus wel zien van hoe ga je er echt doorheen. Denk dat de update, oh hier, —*

S Oh ja en dan ook de conditions waarop hij naar een bepaald iets gaat, dat is wel handig ja. Ja, en verder de libraries — veel kan je gewoon hergebruiken daarvan denk ik. Behalve dat je dan waarschijnlijk voor de AliPay bijvoorbeeld hier die [redacted], `core_[redacted]_transaction_lib` daar zal je waarschijnlijk een AliPay versie van moeten maken denk ik. Afhankelijk van wat je precies nodig hebt uiteindelijk. En het certificate zelfde verhaal. Verder zul je tests moeten maken. Ja beetje zelfde verhaal. En wat stond er bij miscellaneous ook al weer, die Message dan daar hebben we het natuurlijk al over gehad dan. Tests — je zou denken dat ie deze bijvoorbeeld ook — deze erboven — dat ie die in het tabje tests zou zetten.

Q *Ah de TestScenarioApiCallParser, ja —*

S Die staat nu bij miscellaneous gegroepeerd —

Q *Wat ik doe zeg maar voor die filters is een regular expression en wat je dus ook hier ziet is dat je hierzo een subfolder test hebt, en daar zit de regular expression op. En dat zit hier niet in omdat ie in een API lib folder zit. Dus vandaar dat ie nu — het is inderdaad related to tests, maar de regular expression pakt hem niet mee. Dus dat zul je wel met net wat meer bestandjes zien dat je denkt van 'huh maar dat hoort gewoon daar' maar dat —*

S Ja want het is allemaal library dus het zou bij de libs kunnen horen. Even kijken hoor. Ja, die web ben ik vergeten uit te checken maar ik neem aan dat dat de weginterface is eromheen. Dus dat daar zal je ook een AliPay variant van moeten maken.

Q *Ja. Correct.*

S Message was dan specifiek. Want dat is eigenlijk hetgeen wat ik dan nog wel mis hieraan, is verder enige uitleg over wat het inhoudt. Want de code wordt daardoor ook — is er verder vaak ook niet duidelijker in. Want jullie houden je wel aan het — oh hoe heet die stijl ook al weer, met die comments er boven met wat de parameters zijn enzo. Maar er staat niet meer in dan naam van de parameter is dit zeg maar. Het legt meestal niet veel meer uit. Dat is soms — weet je wel, als je programmeert super duidelijk van 'oh dat is dit, dit is dit', maar bijvoorbeeld wat ik dan vroeg in het begin bij — waar staat het — counterPartyName, wat een counterParty is, was voor mij niet helemaal duidelijk bijvoorbeeld. Dat wordt dan verder ook niet echt duidelijk hierin. Maar ja dat is een beetje — als je natuurlijk zelf programmeur bent dan — bij bunq — dan kan je dat gewoon vragen natuurlijk als je op zoiets vastloopt.

Q *Dat is het idee ja.*

S Kan dat niet?

Q *Normaal — hier staat teveel documentatie trouwens met de huidige style dingen die we gebruiken. Huidige stijl is nul documentatie.*

S Oh echt.

Q *Dan staat er letterlijk alleen maar parameter type naam en return type. En that's it. De rest moet er allemaal afgestript worden.*

S Het moet eraf ook?

Q *Ja, als we — zeg maar als ik deze klasse zou tegenkomen en ik moet hem aanpassen dan ben ik gelijk verplicht om hier die bovenste regel weg te halen, deze documentatie moet weg, dat moet weg, en dan voor alle methodes.*

S Waarom?

Q *Consistency.*

S Maar je maakt het jezelf juist onduidelijker daarmee. Je hebt minder — je gaat informatie weg strippen.

Q *Ja.*

S Voor iets dat eigenlijk geen informatie geeft.

Q *Gebruik jij de docs normaal om informatie uit te halen of..?*

S De — deze bedoel je?

Q *Ja. Gebruik je dat normaal?*

S Ja. Dat gebruik ik wel eigenlijk. Maar vooral voor vreemde code I guess. Ja ik vind, — een zo'n zinnetje kan toch best al wat uitleggen. Nou is deze 'canCreate' waarschijnlijk wel een vrij duidelijke functie, maar weet je wel, bijvoorbeeld die MessageModel, ik had nog even in de code gekeken of ik daar wijzer uit kon worden maar dat legt ook helemaal niks uit.

Q *Nee.*

S Dus dan moet je weten hoe dat [redacted] in elkaar zit en dat dat een Message ding gebruikt voordat je weet wat je daarmee moet. Mja, dat is even een dingetje. Ik weet ook niet of dat dan verder hierin moet of niet, maar dat zou nog wel nice zijn als dat ergens past. Misschien dat je — ik zag wel dat ie die PARP ofzo, dat, die hadden iets meer — dan kwam ik op een documentatie website uit.

Q *Ah, in GitLab ja.*

S Ja, GitLab, dat was het. Daar was wel heel veel meer duidelijk. Zeg maar — ik heb het niet echt gelezen maar heel veel meer informatie in ieder geval. En dat zijn dan natuurlijk de hele grote issues. Dus dat is eigenlijk wat ik gedaan heb. Ik weet niet of er verder nog.. —

Q *Nee dat is prima. Dan heb ik nog een ander lijstje met vragen die ik wilde stellen. Vond je de assignment een beetje leuk? Of, wat vond je ervan moet ik vragen?*

S Ik vond hem wel goed opgezet. Die eerste is dan een beetje om die Feature Explorer duidelijk te maken. Dat was voor mij niet helemaal duidelijk dat ik dat bij part twee moest gebruiken.

Q *Nee, ja, dat heb ik je even subtiel duidelijk gemaakt.*

S Het is alleen wel even een bak code waar iemand doorheen moet. Maar op zich voor jou onderzoek lijkt me dat wel nodig. Dat je een beetje ermee probeert —

Q *Ja, dat wilde ik mensen wel laten proberen inderdaad. Niet alleen maar suf zitten kijken, maar dat je ook wat kan doen dat leek me ook wel leuk. Dat is wel lastig inderdaad.*

S Ja, want ik kan me nu ook wel voorstellen dat iedereen wat anders gedaan heeft bij part 2 twee bijvoorbeeld.

Q *Ja.*

S Ik weet ook niet in hoeverre het klopt wat ik nu gedaan heb, maar ik weet ook niet, het is niet heel belangrijk voor jou of zo uiteindelijk voor je onderzoek, of wel?

Q *Nee het gaat meer om het te kunnen vinden van informatie, Dat is belangrijk.*

S Ja. Dus dat.

Q *Oké. Dan wilde ik eigenlijk even feature for feature er langs gaan. De Find By Label heb je gebruikt nu. Wat vond je daarvan? Waar heb je die voor gebruikt? Dingen die missen?*

S Die vond ik best wel goed. Het was me alleen niet helemaal duidelijk toen ik bijvoorbeeld op ideal zocht en toen kreeg ik in het dropdown menu de feature iDeal incoming top-up aangeraden. Die bleek hij niet te kunnen vinden. De informatie erbij. Het was me niet helemaal duidelijk hoe hij dan überhaupt aan het label komt.

Q *De labels komen vanuit GitLab. Zeg maar wat je aan het begin zag, de uitleg, daar staat van de commits die zijn met een tag gerelateerd aan een issue en van dat issue trekt hij de labels af.*

S Oké, dus wat jij commit daar pakt hij de labels van?

Q *Zo zit de link naar GitLab erin. Maar ja soms zal je het zien dat een verkeerd label er ophangt, een label die niet meer bestaat, nou ja noem het maar. En dan krijg je dus dit soort verhalen.*

S Oké, de stad wordt dus niet ge-update verder?

Q *Nee, hij doet op dit moment — is het een indexer die runt een keer, indexed alles tot nu zeg maar en daarom is deze nu ook al drie weken oud of zo. En hij update niet live zeg maar. Dat zou weer een stap verder zijn. Dus daarom heb je soms dat een label geen/ weinig/ slechte resultaten geven. Als de links niet goed zijn in de code, ik bedoel, een paar jaar geleden deden ze dat nog niet in de commit messages. Dus dat zal hij ook niet kunnen vinden. En ideal is vrij aan het begin gebouwd.*

S Ja, precies. Dus daarom kan je daar vrij weinig over vinden.

Q *Ja, exactly.*

S Verder vond ik het wel Nice werken. De search functie werkt best wel goed volgens mij. Voor wat er dan instaat.

Q *En wat vond je van informatie die je terugkreeg? Waren er dingen overbodig? Waren er dingen die misten? Wat je begon natuurlijk met scrollen in de code, Dus zou er iets kunnen zijn dat ik nu toevoeg dat je dat niet meer nodig hebt? Het scrollen.*

S Nou wat ik dus net al zei, misschien wat meer uitleg over de inhoud van een functie. Meer globale informatie eromheen. Dus bijvoorbeeld die message protocol bij [redacted], waar dat dan voor bedoeld is. Zo een kleine regel kan dat dan al duidelijker maken.

Verder vind ik de developers heel mooi hier boven staan. Dat is wel echt duidelijk. En die result for had wat mij betreft wat groter gemogen, Dat je nu, bij elke [?] Aan het kijken bent — het valt een beetje weg in de related labels.

Q *Ja. En afgezien van dat je het mooi vindt, die developers die daar zo staan, denk je dat het ook nog nut heeft?*

S Ja ik denk het wel. Dan kan je gelijk zo van, oh je bent hiermee bezig geweest, dan kan ik je daar vragen over stellen of zo. Dat kan je makkelijk terugvinden. Zijn dat er echt maar zes geweest? Die hiermee bezig zijn geweest?

Q *Het is heel vaak dat het zelfs maar een developer is die er aan werkt, en dan vertrekt hij en dan zijn er bugs en dan pakt iemand anders het op. Dus dat kan zijn waarom het nu, je zit nu in slice, dus dat zijn er iets meer want dat is gewoon een losse app. Maar, ja, je zal vaak zien dat het er niet veel zijn.*

S Ja, oké. Even kijken hoor. Ja dit is ook wel mooi overzicht, Die issues. Ik weet niet of die net zoals bij [redacted] er ook bij horen? Waar is dat eigenlijk? Waarschijnlijk daar. Waarschijnlijk heb ik gewoon niet verder naar beneden gekeken. Ja, het zijn er hier maar twee zo te zien.

Q *Ja.*

S Wat is dit? Oh ja. Dus nee, dat vond ik nog best wel goed eigenlijk.

Q *Oké, had je de find by Developer nog gebruikt?*

S Nee, ik heb er even mee gespeeld maar dat is vooral denk ik als je Meer op de persoon wil zoeken en kijken wat die gedaan heeft. Dan is dat handig.

Q *Zou je een situatie kunnen bedenken maar dat dat werkelijk zou kunnen gebruiken? Of zeg je van "geen idee eigenlijk".*

S Ja misschien als je bijvoorbeeld aan het overleggen bent met iemand en hij zegt "nou daar heb ik iets van aangepast" dat je dan makkelijk eventjes via die persoon kom kijken want het zijn minder features dat je daar dus snel kan kijken maar je krijgt niet te zien wat iemand precies gemaakt heeft. Als je hierop klikt ga je natuurlijk weer naar een feature. Dan zie je niet wat die developer precies gedaan heeft. Dus wat dat betreft vind ik het een beetje beperkt denk ik.

Q *Dus je zou dan van een developer — je zou willen kunnen zien wat een developer heeft gedaan dan?*

S Ja, bijvoorbeeld. Ik weet niet of dat haalbaar is hoor. Want nu geef je alleen de features aan die je gebruikt hebt en dat is — zou je vrij weinig gebruiken denk ik. Behalve als ik toevallig wil weten welke feature jij gewerkt hebt maar ik zou niet zo goed weten wanneer je dat zou gebruiken als developer.

Q *Oké.*

S Dat.

Q *De Find By CIT, heb je die nog gebruikt?*

S Nee, maar dat had ik waarschijnlijk bij die tweede kunnen gebruiken. Alleen want er staat geen nummer bij natuurlijk.

Q *Je kan natuurlijk wel — als je zoekt op* [redacted] *dan kan je die issues doorklikken, Die onderaan stonden.*

S Ja, precies. Ja, heb ik nu niet gebruikt. Ja dat is handig denk ik als je bezig bent met, Ja hele specifieke issues oplossen of als er vragen bij een bepaalde issue zijn, dan is dat handig denk ik. Maar in relatie met deze opdrachten heb ik ze niet echt gebruikt.

Q *Oké. If any, wat zijn de voordelen van het gebruik van deze tool? Volgens jou?*

S Los van het tracken van issues?

Q *Gewoon in General. Gewoon, alles.*

S Want ik weet niet, die CIT, waar stond dat ook alweer voor—

Q *Central issue tracking.*

S Dat ja. De Central issue tracking dat was wel handig om te hebben, Dat lijkt me super nuttig. En ik denk dat deze tool ervoor zorgde dat dat een stuk inzichtelijker wordt, wat er allemaal aan de hand is en hoe dat zit. Dit begreep ik bijvoorbeeld niet, de commit messages en hoe die dan Erin zitten, maar dat is natuurlijk weer — pakt natuurlijk het label eruit en dan zoek je.

Q *Ja, ik had de commit messages er eerst wel in staan maar dat weet bij heel veel features een hele lange lijst die vond ik niet veel toevoegen. Zou jij dat wel nuttig vinden als de commit messages er ook onder staan, Desnoods in een apart tabje?*

S Misschien wel. Zeker als je met een bepaalde issue langer bezig bent. Wat je zegt al dat vaak maar een developer bezig is geweest met een bepaalde feature en als er dan commentaar op is — of er moet iets aangepast worden dan kan je wel een beetje de historie terugzien. En als je dan een nieuw persoon bent dan kun je al zien van dit is wel toegevoegd. Dus wat dat betreft lijkt het me wel handig, Maar als jij — ja je moet dan in de praktijk kijken of dat dan ook echt handig is. Dat is dan weer een tweede natuurlijk. Het klinkt wel handig.

Q *Zou je het zelf gebruiken denk je? Ik bedoel, doe je nu in een project soms de historie bekijken, Informatie uithalen of..?*

S Ja, ik gebruik het wel eens. Op GitHub bijvoorbeeld, dat is dan ook meestal waar ik mijn project in doe, dan kijk ik wel in de historie naar om te kijken — ja, Wat er veranderd is en wanneer, dat doe ik wel eens ja.

Q *Oké. Zijn er nog dingen die anders hadden gemoeten in de tool? And if so, why?*

S De error's er uit? Anders hadden gemoeten?

Q *Ja, zeg maar, mis je informatie dat je zegt "oh, dit had nog wel geholpen om het te begrijpen". Je had het al genoemd, Die omschrijvingen. Zijn er nog andere dingen dat je zegt "dit had het echt nog duidelijke gemaakt of inzichten gegeven of dit kun je echt weghalen of noem het maar".*

S Nou ja dit kan natuurlijk best wel een lange lijst worden. Dit zijn de files die aangepast zijn, maar ik zou niet echt weten hoe je dat beter kan weergeven. Dus wat dat betreft, niet specifiek commentaar. Dus nee, ik zal even niks weten, eigenlijk.

Q *Oké. In General nog feedback die je nog niet heb genoemd?*

S Nee, ik denk dat ik alles genoemd heb.

Q *Cool. Wat is je indruk van de bunq bekend als je die zo ziet?*

S Vooral weinig documentatie dan. En een beetje, de naamgeving vond ik wel duidelijk, Over het algemeen.

Q *Bedoel je dan de naamgeving van de mappen?*

S Ja, maar ook wel van de functies Enzo, Dat is allemaal wel vrij goed vind ik. Vooral de documentatie die ik vaag vind. En ik heb natuurlijk maar een klein stukje ervan gezien. Het ziet er niet uit alsof het een zootje is of zo. Ja, hij moet er alleen wel wat van afweten qua achtergrondinformatie. Maar dat is dan meer documentatie dus, denk ik. Ja en verder heb ik maar zo'n klein stukje ervan gezien dat ik niet kan zeggen of het goed in elkaar zit of zo.

Q *Nee, oke. Maar als je het zo ziet, wat is dan je eerste indruk? Dan zeg je dus van — tja de naamgeving noemde je —*

S Ja, en dit vind ik een beetje onoverzichtelijk, Maar —

Q *De hoeveelheid mappen?*

S Ja, het zijn zoveel mappen. Maar, ja voor een groot project is dat misschien ook wel logisch. Ja bijna alles heet core. Ik weet niet of dat — waarom dat precies gekozen is.

Q *Het meeste is voor de core businesses, en we hebben ook een hoop tooling. Dat hangt er dan meer omheen. Om core draaiende te houden.*

S Ja, tools en Mastercard nog een stukje en hier staan nog een beetje extra dingen. Ja, dus wat dat betreft zou je misschien zelfs nog een subtabje met core of zo, ja ik weet niet, dan krijg je in core nog steeds bijna alles. Dus ik weet niet of dat echt veel toevoegt.

Q *Nee ja, er zijn nog wat alternatieven voor maar.. Denk je dat deze tool, Als je een feature moet implementeren of verbeteren, ja dus danig kan helpen dat je senior niet meer tot minder nodig hebt? Normaal kun je een senior vragen stellen en zo natuurlijk.*

S Misschien minder? Maar ik denk dat je altijd wel iemand nodig hebt, Maar dat is een beetje — lijkt me altijd wel handig hoe dan ook waar je ook werkt. Dat er altijd iemand is, of je manager, of iemand — ja zeker als je zo'n grote doorloop hebt aan developers, denk ik dat een senior wel super nodig is. Al was het maar omdat je — het kost tijd om overzicht te vergaren. En zo'n senior kan ervoor zorgen dat het overzicht vergaren sneller gaat. Als je dat allemaal zelf moet gaan ontdekken dan gaat dat een stuk langzamer.

Q *En op wat voor manier zou een senior daarbij kunnen helpen? Wat denk jij dat het beste werkt?*

S Meer voor vragen stellen denk ik, Op het moment dat je echt ergens niet uitkomt of misschien de eerste pointers van "je kan hier en hier kijken" en "het zal wel ongeveer hier aan gerelateerd zijn" en "kijk daar even", een beetje op die manier.

Q *Wat voor soort vragen zou je dan nog willen stellen aan een senior?*

S Vooral overzichtsvragen. Dus van "ik ben nu aan deze en deze feature bezig, het lijkt hiermee te maken te hebben, maar ik snap nog niet helemaal hoe dit dan geconnect is aan dit" dat soort vragen.

Q *Ja. Denk je dat dat soort vragen met Een tool wel, Ja,— ja, dus jij denkt dat een tool dat nooit helemaal weg zou kunnen nemen, om dat volledig overzicht te kunnen krijgen? Dat dat niet met een tool mogelijk is?*

S Ik denk dat het voordeel van een senior is dat het een mens is zeg maar, Waar je gewoon heel — waar je in gesproken taal vragen aan kan stellen. Terwijl een tool is altijd heel — je moet heel specifiek keywords, vragen stellen zeg maar. En ik denk dat, Wat dat betreft, een senior onvervangbaar is dan een tool. Ik denk wel dat een tool dus wel vragen kan verminderen. En dat dit al wel een goed overzicht geeft — geeft dit je wel al een hoop houvast dat je voor veel dingen misschien geen vragen hoeft te stellen. Dus in zoverre kan het wel helpen.

Q *Oké. Dan een iets generiekere vraag. Wat zijn volgens jou de meest belangrijke informatiebronnen als je een nieuwe grote code base Moet leren, of een groot framework?*

S Belangrijke bronnen van informatie? Dingen als documentatie denk ik — even denken hoor. Ik heb nog niet zoveel grote projecten hoeven doorspitten.

Q *Maar waar zou je dan als eerste naar kijken, denk je?*

S Ja het hangt er een beetje van af wat er beschikbaar is eigenlijk. Ik zou eerst voor documentatie zoeken, daar een beetje doorlezen en het hangt ook een beetje af van wat je moet doen. Na de documentatie zou ik ook een beetje door de code gaan bladeren

denk ik. Ja, en eventueel exploren wat dit soort dingen zouden kunnen doen. Dit soort extra tools. Maar dat hangt heel erg vanaf van zo'n project van wat er al bestaat bij zijn project.

Q *In een ideale wereld, Stel alle mogelijkheden zijn beschikbaar, wat zou je dan kiezen?*

S Qua wat bedoel je?

Q *Waar je informatie vandaan zou kunnen halen. Het is een magisch project waar echt elk mogelijke informatiebron beschikbaar is.*

S Ja, vooral geschreven documentatie zoals bijvoorbeeld zo'n formuliertje die dingen uitlegt.

Q *Maar dan zeg maar zo geschreven of documentatie in de code zelf?*

S In eerste instantie — kijk, voor het overzicht dan — ja niet geschreven dan, maar online — zo'n geschreven documentatie is denk ik voor het overzicht handig en dan daarna in de code zou ook heel veel helpen als je daar gewoon goed opschrijft wat daar — wat een functie doet, En waar een klasse voor is dan zouden dat al ideale dingen zijn denk ik. En qua features, Ja, dat is ook een beetje organisatorisch gezien denk ik, zo'n Central Issue Tracking, dat is handig gewoon ook om bij te houden wat je — wat er nog moet veranderen, moet verbeteren, Waar iemand aan aan het werk is, op die manier is dat wel handig. Wat dat betreft trouwens, als je bij Search By Developers zoekt kan je niet zien wat iemand als laatste — geen historie of zo. Wat iemand als laatste heeft aangepast of zo.

Q *Ah de issues van een developer basically.*

S Ja waar iemand op dit moment aan aan het werk is of zo, Dat kan je nu niet zien.

Q *Ah ja, fair.*

S Dat zou misschien nog wel handig zijn. Van "oude persoon heeft eerst aan — weet ik veel, bij jou even kijken aan bank accounts gewerkt en daarna aan login en daarna aan attachments of zo".

Q *Ja.*

S Dat kan ook handig zijn. Maar ik weet ook niet — het is misschien ook weer privacy die informatie die je dan weggeeft.

Q *Ja op zich is dit intern. Dus, het is allemaal public info in GitLab dus dat is op zich het probleem niet. Het is alleen dat ik dan met timestamps moet gaan kloten enzo. Dat had ik nu ook al bewust niet gedaan. Het is nu ook dat als je een bestand verwijderd bijvoorbeeld, dat wordt niet meegenomen in die tool dus soms als je zo'n grafiekje probeert open te klikken, in de controllers bijvoorbeeld, dan blijft hij eindeloos laden. Omdat het bestand dan niet meer bestaat bijvoorbeeld. Want anders dan moet je dus filters gaan maken van "tussen deze en deze range" en die moet gaan kijken van als*

het bestand verwijderd is en later weer aangemaakt enzovoort dan krijg je allemaal rare shit in je database en dat maakt de queries ook 100 keer lastiger. Dus dat ligt buiten de scope van mijn thesis.

S Maar dat zou dus uiteindelijk ook wel nuttig kunnen zijn denk ik.

Q *Ja.*

S Was dat het? Ik weet niet of jij nog vragen had?

Q *Ik had verder geen vragen meer. Heb jij nog vragen?*

S Nee. Ik hoop dat je er wat aan gehad hebt.

Q *Absoluut. Altijd. Dan, hartstikke bedankt.*

C.20 Subject 20

This interview is performed differently due to time constraints. Furthermore this participant did not do the assignment, but rather was asked for feedback on the tool. The participant was too familiar with the bunq backend. The participant has been involved in the backend since the first lines that were written for it.

Questioner (Q) *Oké, cool. Dus ja, dit is dus de interface te zien krijgen. Dus ik zou zeggen, probeer even wat dingen, het zou voor zich moeten spreken.*

Subject (S) Het logo is een beetje platgedrukt.

Q *Ja, I know. Er stonden eerst meerdere menu items in. Maar die zijn er uitgehaald.*

S Ja, precies. Als ik wil weten hoe ik een payment moet doen, gewoon een normale payment — search.. — dan zie ik wie er allemaal iets mee gedaan hebben. En dit zijn alle dingen waarvanuit dit aangeroepen wordt of..

Q *Zeg maar, ik gebruik de git log om te kijken welke bestanden zijn er aangepast. In je commit message zit altijd een reference naar een issue —*

S Ja.

Q *En vanaf die issue heb ik weer een reference naar wat voor feature label eraan hangt in GitLab. Dus op die manier — ik ga vanaf dat Feature Payments naar de issue, dan alle commits die daarnaartoe referencen en dan naar alle files zeg maar.*

S Oke. Maar hoe vindt je dat issue?

Q *Dat issue gaat vanuit de commit message.*

S Oke. Dus je zoekt eigenlijk naar commit messages waar Payment in staat, begrijp ik dat goed?

Q *Stel dat je CIT/bugs#123 dat dat een label heeft Feature Payments, dan —*

S Oh dit zijn de GitLab labels?

Q *Ja. Correct.*

S Ja, oke.

Q *Omdat die in principe redelijk de features afkaderen.*

S Ja, precies. Dus als ik nu SWIFT Payments doe, iets overzichtelijker ding —

Q *En iets ouder zo te zien.*

S Valt op zich wel mee.

Q *Er is een moment geweest dat jullie die tags in de commit messages hebben geïntroduceerd. En alles daarvoor dat is wat rotter vindbaar.*

S Oh dat weet ik niet bij deze. Het klopt wel, alleen er zit wat — oeh hey, dit is grappig. [visualisation]

Q *En dit wordt dynamisch gegenereerd aan de hand van de versie die op dit moment in develop staat.*

S Ja, want dit is in feite een visualisatie van de workflow definition?

Q *Ja, correct.*

S Ja, precies. Ja dat is wel cool. We hebben ooit iets gehad met — dat we die — of waren we ooit mee begonnen —

Q *Flow?*

S Dat we een tool hadden om al die workflow definition files visueel te editten. Dat was altijd enorm veel gedoe om goed te krijgen. En dan was het wel of weer niet goed te lezen [some mumling]. Ja hij pakt dus niet helemaal precies wat er in die directories, of repo's vroeger, zit.

Q *Hoe bedoel je hij pakt niet alles?*

S Nee, hier staan bijvoorbeeld nog meer dingen in die code directory.

Q *Ah zo. Ja. De commits die dan door mijn tool zijn gevonden, alleen dit bestand is in die commits aangepast. En er is natuurlijk op een gegeven moment een backend import geweest dat het allemaal in deze repo is geschoven dus misschien dat het daarvoor nog is geweest. De allereerste commit was echt een huge commit —*

S Ja precies. Billing — [looking around in tool]. Deze heeft ook nog echt oude oude billing erin staan, wat allang deprecated is maar waar nog wel code van is.

Q *Ah ja, bestanden die inmiddels verwijderd zijn daar heb ik niet naar gekeken omdat ik dan best wel complexe queries moest gaan schrijven van 'dit bestand bestaat in deze timeframe' en —*

S Ja, precies. Er zijn ook nog wel bestanden die nog wel bestaan maar gewoon echt niet meer gebruikt worden maar —

Q Bestanden die verschoven zijn dat zit er op zich ook nog in.

S En dit zijn de issues die er bij —

Q *Ja.*

S Want, even kijken hoor, als ik de andere kant op doe, als ik op basis van feature, maar dat pakt ook niet alles natuurlijk.

Q *Nee, dit zou dan alle features moeten teruggeven waaraan je hebt gewerkt.*

S Hoe recent is dit trouwens allemaal wat hier staat? Of is het realtime?

Q *Nee dit is niet realtime. Het wordt van te voren geindexed. Dit is nu een maand, anderhalve maand geleden uit m'n hoofd. Nee langer, ik heb in juni/juli m'n experimenten gedaan, rond die tijd heb ik geindexed, iets ervoor.*

S Ja.

Q *Dus OAuth enzo zal er allemaal niet instaan.*

S Ja precies, en dit is met Trophies, dus dat is niet —

Q *Deze is iets compacter.*

S Deze was ook iets beter tagged. Dus dat klopt ook wel. De workflow PHP en de workflow JSON die zijn identiek dan toch?

Q *Ja ja. Ik heb geen link gemaakt tussen — om te laten zien van 'dit is dezelfde'.*

S En zo in GitLab, dat vindt je makkelijk.. maar op zich zo'n overzichtje van wat erin zit is wel grappig.

Q *Als je geen issue nummer weet om in te vullen dan kun je hier naar beneden scrollen daar staan de issue references — als je deze hovered en dan dat groene knopje —*

S Dan doe je hetzelfde als die.

Q *Yes, exactly. Dan hoef je niet moeilijk te zoeken.*

S Ja, precies. Dus we hebben dat issue en wat zat er allemaal onder.. dat zijn deze files die zijn aangeraakt. Dus dan kun je eigenlijk iets sneller dan in GitLab zelf, dat is niet altijd even snel — deze mensen hebben eraan gewerkt, ik, — Oke.

Q *Ja, dan heb je in principe alle zoekmogelijkheden gehad. Als je dan de — met name de informatie die je te zien krijgt, dan ben ik erg benieuwd wat je daarvan vindt. Of het voor jou zelf ook nuttig zou kunnen zijn bijvoorbeeld of dat je zegt van 'dit soort dingetjes moet je echt nog toevoegen want dat geeft heel veel waarde of . . . ' ja noem het maar.*

S Ik had hier even gemist dat die gefilterd had op type labels.

Q *Ja, dit is ook een beetje terug naar het verleden omdat er af en toe een label was dat een feature was maar net een andere naamgeving had, en ik doe nu letterlijk filteren op die "Feature: ".*

S En als je kijkt naar [opens big label] —

Q *Dat is waarschijnlijk flink wat wat ie nu terug gaat geven. — Het valt nog mee qua..*

S Ja, precies. Het is wel grappig want dan heb je dus inderdaad bunq Update 7 en dan wat voor features daaronder staan — het is wel veel, die zijn niet allemaal related

Q *Ja ik heb die related features dat kijkt zeg maar hoeveel overlap er zit tussen de bestanden, ik weet van welke features welke bestanden er nu bij horen en dan de overlap daartussen bepaald hoe hoog ze daar geranked worden.*

S Wat altijd een leuke is, is Priority Showstopper —

Q *Er zijn een hoop mensen met showstoppers zo te zien.*

S Ja, ja als ik nu — want ik heb nu Related Labels, kan ik een beetje verder filteren als het ware? Dat ik zeg ik wil Showstoppers uit Billing?

Q *Eh, nee. Je gaat nu echt zoeken op die Feature Billing, om te kijken wat daar dan in zit. Gecombineerde filters dat heb ik er niet in zitten.*

S Nee, want dat kan dus wel leuk zijn inderdaad ook voor dingen dat je achteraf nog makkelijk kan checken van nou 'ik wil de showstoppers van billing door developer'.

Q *Ja, precies. Dat je hem steeds dieper filtert eigenlijk.*

S Ja.

Q *En waar zo jij dat voor gebruiken?*

S Naja, vooral gewoon showstoppers voor billings kan je echt zien — dat zou je ook kunnen gebruiken wie hoeveel problemen, issues, heeft — hierin gehad of gefixed. En wat dat betreft ook te zien van hoe goed is je code nou eigenlijk.

Q *Ja. En vanuit het perspectief van een nieuwe ontwikkelaar?*

S Ja, dat is meer vanuit het perspectief van een echt senior die kijkt hoe iemand het doet.

Q *Ja, dus dan zou het een soort van observing zijn van —*

S Ja, ja.

Q *Oh, en omdat het dan showstoppers zijn zie je dan gelijk waar de knelpunten zijn geweest bij die nieuwe —*

S Ja, want in principe moet je natuurlijk nooit een showstopper eruit — want showstoppers die zijn echt bij Triage testing komen die naar voren, en die hadden de ontwikkelaars in principe ook zelf kunnen vinden.

Q *Ja, maar dat zou dan een feedback moment kunnen zijn voor nieuwe —*

S Ja, dat is meer dan feedback.

Q *Ja.*

S Het valt — met die labels valt en staat het wel met hoe goed die labels bijgehouden worden.

Q *Ja. En het is in de commit messages natuurlijk ook, als je daarin een typefout maakt in die message dan wordt ie ook al niet meer goed gekoppeld aan de rest. Maar denk je dat voor een nieuwe developer zo'n overzicht van features dat dat nuttig kan zijn voor hen om te leren te werken in de backend zeg maar?*

S Nee, het is wel een stuk makkelijker om hier gewoon een beetje doorheen te klikken om gewoon naar features te kijken, wat eronder valt, dan in GitLab zelf. Waar je al snel te maken krijgt met verschillende Merge Requests waar je in moet duiken, dat is hier dan al samengevoegd. Want ik heb nu dus inderdaad gewoon een issue geopend en daar staan gewoon al die files die eronder vielen. Dit is wel een goede, die deposit of safe keeping fee, want die heeft best wel wat files aangeraakt. Maar dan staat meteen vrij duidelijk van 'oh ja dit komt er allemaal bij kijken' en inderdaad DepositDailyCalculate die heeft ook een mooi — [visualisation]

Q *Ja, zeker bij complexere workflows komt die structuur ..—*

S Dat scheelt wel echt.

Q *Het is op zich een hele simpele syntax. Ik definieer alleen maar de nodes, want de states zijn allemaal nodes, en ik definieer 'er is een transition van die node naar die node'. En het renderen doet ie allemaal zelf. En ook op een manier dat het nog leesbaar wordt, dat ie niet alles squeesh —*

S Wat ook nog wel een leuke zou zijn hier — want dit zijn gewoon alle files die aangeraakt zijn toch?

Q *Ja. Is dat je meer ziet van welke ook nieuw zijn toegevoegd onder een issue. En wat is de toegevoegde waarde daarvan voor jou?*

S Toegevoegde waarde in dit geval bijvoorbeeld is dus dat je makkelijk onderscheid kan maken tussen wat hoort — wat echt nieuwe geïntroduceerd is onder een feature, zoals WorkflowDepositDailyCalculate, terwijl een TimebombLib, dat is een vrij generieke en dus niet perse relevant, dus dat is meer noise binnen dit overzicht.

Q *Dus dan zou je het meer kunnen gebruiken om echt de feature unieke files eruit te kunnen halen van de common files die extended zijn om —*

S Ja. Want zo'n TimebombLib die wordt door heel veel verschillende issues en features aangeraakt. Dus dat geeft geen concrete informatie over de feature of over het issue. En ook van, welke zijn — waar zaten echt de changes, in plaats van de details omdat je een functie hebt gerenamed ofzo.

Q *Ja.*

S Maar wat hier wel inderdaad duidelijk is gewoon issue van deze daemon en deze workflow en dat model, —

Q *Denk je dat dit een systeem is dat jij als senior ook zou kunnen gebruiken? Als in dat het van toegevoegde waarde is of voegt het —*

S Ik denk dat het voor mij niet direct veel toevoegt. Omdat — ja gewoon de onderlingen verbanden tussen controllers, models en views dat weet ik gewoon. Dat zit er zo goed in inmiddels. En dan kom ik er vaak ook wel met gewoon kijken vanuit PhpStorm zelf. Wat ik ook — wat voor structuur voor nieuwe mensen nog wel interessant kan zijn is dat je — je hebt natuurlijk hier billing deposit controller en `tools_billing_deposit_daemon` en `billing_deposit_model`, maar die staan hier gewoon een beetje door elkaar heen met ook andere dingen er tussen, —

Q *Ja, dat is interesting to be honest. Want het zou gewoon alphabetical moeten staan.*

S Want echt halverwege staat nog een AdminPaymentRequestView. Nou, dit is niet alfabetisch.

Q *Nee, dat zie ik nu ook. Bij mijn weten was hij gewoon alfabetisch gesorteerd maar misschien dat Neo4J iets raars teruggeeft in dit geval. I don't know. Dat zou ik moeten bekijken. Normaal zou ie alfabetisch zijn.*

S Ja, oke. Maar wat wel — oh wacht je hebt hier wel iets met Models, Views en Controllers staan.

Q *Ja. Dit doet een —*

S Nee dat wilde ik een beetje van — omdat het gewoon een lijst was, zag je eigenlijk de structuur van hoe models, views, controllers en daemons — die staat hier ook niet tussen [filter bar] — hoe dat zicht verhoudt ten opzichte van elkaar. Want deze heeft toevallig wel een view, maar er zijn ook wel eens changes dan heb je een model en een controller en een daemon maar geen view.

Q *Ja, de daemons had ik niet aan gedacht voor een los tab. Maar omdat we in principe een heel sterk MVC model hebben, die stonden er dan in elk geval bij, en toen ben ik in die lange lijst gaan kijken van wat heb je nog meer als grote entiteiten. Ja, de Libraries zag je vrij veel natuurlijk en tests. Maar zeker voor oude code zal dit nog niet altijd even vlekkeloos werken omdat het gewoon een regular expression is op de path. Dat werkt nu met de codebase echt supergoed omdat ie zo consistent is natuurlijk, maar in het verleden — Je zal af en toe — Miscellaneous is de categorie waar alles in wordt gezet als het niet filteren in een van die eerste dan, dan komt het daar terecht, maar daar zie je dus af en toe ook models terugkomen die nog een oude benaming hadden ofzo.*

S Wat misschien nog wel een leuke kan zijn, — dat is meer als algemene explorer als het ware denk ik — qua code, met PhpStorm kun je met die search functie — hier kun je dus inderdaad — dit is billing deposit, elke directory die echt essentieel is voor de feature heeft `billing_deposit` in de naam en de dingen die geen billing deposit hebben dat is meer zijdelings gerelateerd. De core van de feature is echt `billing_deposit_model`, `billing_deposit_view`, `billing_deposit_controller` en `tools_billing_deposit_daemon`. En natuurlijk `billing_deposit_lib`, ja precies, die zit er ook tussen. En op dat niveau — ik weet even niet hoe je dat visueel zou kunnen maken, maar dan zie je echt hoe per categorie — ja, ik weet anders even niet hoe ik het moet noemen — waar zijn daemons, models, views, controllers, libs, tests, want dan komt eigenlijk ook meteen naar voren wat de hoofdfeatures zijn. In dit geval de billing deposit is de kern en die komt in al die filters voor. Model, view, controller en tests. Terwijl een `core_user_company_view` die staat alleen in de tests.

Q *Ja. Dus daar is heel toevallig iets aangepast —*

S Ja, en dat dan waarschijnlijk ergens in een test file een method is renamed of zoiets dergelijks.

Q *Ja, precies.*

S Als je dus de changes voor dit issue wilt begrijpen, de structuur die eronder ligt, dan is alles met billing deposit hetgeen waar het om draait.

Q *Ja. Nou heb ik iedereen die het experiment heeft gedaan die heb ik een assignment gegeven van 'joh, kijk naar [redacted]', dat heb ik dan gebouwd dus ik wist dat de structuur goed was. En probeer nu AliPay op te zetten, en dat werkt dan op dezelfde manier als [redacted]. En aan de hand van deze tool heb ik ze dat op laten zetten. Denk je dat dat een leuke exploration assignment zou zijn zeg maar voor nieuwe developers?*

S Bij [redacted] zei je?

Q [redacted], *sorry.*

S [Looking up feature in tool]

Q *Dus ik heb ze gezegd, gebruik deze feature als basis, dat je kan zien hoe zo'n structuur in elkaar zit, en probeer dan een soortgelijke structuur op te zetten.*

S Oke, dus even kijken. Ik heb nu dan inderdaad die feature, ik heb gewoon doorgekregen 'kijk naar die feature'. Prima. En dan hebben we dus die grote lijst met al die — dat is veel. Oke dat is me te veel. Dus ik kijk even naar Models, dat is wel interessant. Welke heb ik hier allemaal nodig.. — Certificate, Message, Transaction en RequestResponse en ook een TokenQr. Wat hier dan inderdaad handig zou zijn in zo'n situatie als je dit echt moet gaan bouwen, is welke zijn nieuw toegevoegd onder die feature en welke bestonden al. Want dan zie je 'hey, `core_[redacted]_transaction_model` was nieuw. `core_[redacted]_message_model` was nieuw, certificate model was nieuw'. Maar RequestResponse die bestond al. Dus als je een nieuwe feature hebt dan zal je RequestResponse waarschijnlijk wel aan moeten passen.

Q *Ja.*

S En die heeft geen definition hier staan, dus die — dat is waarschijnlijk alleen ACL geweest.

Q *Ja, in dit geval —*

S In dit geval dan he.

Q *— er moest een aparte case bijkomen voor de [redacted] transaction, er zit een transaction aan gekoppeld en dat kon of iDeal [redacted] zijn.*

S Ja, want je doet op basis van de files die in de commit messages stonden he?

Q *Ja.*

S Ja, dus daar staat ook wel of ie toegevoegd of gewijzigd was. Dat is makkelijker parsen als wanneer je echt moet gaan kijken wat er gewijzigd is.

Q *Ja, de diffs heb ik echt helemaal niks mee gedaan.*

S Nee, dat —

Q *Ik heb zeg maar de — het indexen gebeurt aan de hand van een git log commando, en dan kan je die `-name-status` tag aan meegeven en dan komt er onderaan een lijstje met die files.*

S Diffs zou veel te veel moeite zijn.

Q *Ja, die kan je ook gewoon uit GitLab trekken op zich nog, stel dat je die wilt toevoegen. "Show the diffs", dan kun je die dynamisch uit GitLab trekken. Wat dat betreft heeft GitLab best wel een sterke API.*

S Ja.

Q *Maar denk je dat — stel je laat een nieuwe developer hiernaar kijken om de backend te verkennen, zou die dan een goed beeld hebben gekregen van de backend nadat ie dat heeft gedaan? Van de high-level concepten en hoe het samenhangt? Of komt er nog wel meer bij kijken?*

S Ja, er moet wel uitgelegd worden hoe een view hier een controller hier gebruikt. Je moet wel even er vanuit gaan dat iemand niet helemaal is met MVC dingen natuurlijk.

Q *Ik moet zeggen, ik heb ze er wel een klein A4'tje bijgegeven met wat basis uitleg waarin ook wordt uitgelegd van 'joh, onze generator generate een deel van de classes' en hoe de flows vanaf de Router naar de pre-processors, workflows, post-processors en dan komt er een response uit rollen. Dat stukje is daar zeg maar uitgelegd.*

S Ja.

Q *En daarnaast hadden ze deze tool en dan naja, probeer maar of je eruit komt, of je begrijpt hoe het zit.*

S Ja, precies.

Q *Dus dat eerste stukje van die samenhang dat is dus wel een soort van uitgelegd.*

S Ja, maar dat is dus inderdaad aparte uitleg dan.. — Ik vind die visualisatie echt grappig. Ja, en dan zou je dus bijvoorbeeld kunnen voortborduren om ook bij view definitions nog een soort visualisatie te krijgen van hey, dit is een endpoint met zo'n soort body — of met deze velden. Want dat is nu wel, op het moment dat je inhoud zou willen bekijken dan moet je hier helemaal uit gaan.

Q *Ja, dan moet je wisselen — PhpStorm staat er ook [on the laptop]. Bij mijn weten kan ik niet vanuit JavaScript PhpStorm opeens naar boven ploppen.*

S Dat zou echt ideaal zijn als dat.

Q *Ja, het zou ook best wel een security flaw zijn als JavaScript commands op je terminal zou gaan uitvoeren of somehow —*

S Even kijken. Want ik zie daaronder ook issues staan, er staan er twee. Maar staan er altijd twee of is dat ook afhankelijk van of er ook in een issue bijvoorbeeld een model is aangepast.

Q *Nee de — dat heb ik inderdaad ook als feedback gekregen — deze balk [filters] is onafhankelijk van wat daar staat [issues]. Het zijn echt losse secties zeg maar.*

S Ja.

Q *En deze die stond eerst hier tussen de developers en de files in, maar bij sommige features was die lijst met issues zo lang dat je filelist helemaal niet meer zag zeg maar.*

S Ja, oke. Dus dat is inderdaad niet helemaal duidelijk dat ie niet — dat zou eigenlijk net als hier, een extra streepje.

Q *Ja, of dat het een tabje —*

S Ja, of zoiets.

Q *Maar wel inderdaad dat er een duidelijke scheiding is, maar nee dit zijn gewoon alle issues die het label hebben waarop je nu hebt gezocht.*

S Maar dat zou ook nog wel een leuke kunnen zijn dat als wel een filter hebt dan zou je apart issues — we hebben dus een feature [redacted], maar dan — dat is eigenlijk ook een beetje wat je gegroepeerd kan hebben wat ik al eerder zei, iets meer op elkaar stapelen — dat je het issue pakt voor de admin implementatie en de issue voor gewoon de mobile API implementatie. En dan ziet wat er in welke views voor of welke files en welke views voor admin waren aangeraakt en welke juist niet. Maar dan dat je eigenlijk — ja dat kan met de issues, maar eigenlijk is dat gewoon combineren van labels.

Q *Ja, dus echt die search options uitgebreider, —*

S Ja precies, je kan wel gewoon zeggen een issue pakken. Hmm, ik heb nu één issue gepakt en er staan hier al drie anderen er onder. Hoe werkt dat?

Q *Het kan zijn dat deze issues dan weer zeg maar terugkoppeling hebben — zeg maar. —*

S Zijn dit weer issues referenced in dit issue?

Q *Issues die hier staan zijn ook de issues die in de issue body op GitLab een reference hebben. Soms zie je van 'dit gaat verder op issue incremental' en die worden hier ook getoond. De files zijn wel alleen expliciet die via commit gekoppeld zijn zeg maar. En dat was hier met name omdat je — het wordt niet meer veel gebruikt maar die PARP issues die staan vaak — die hebben geen directe commits eraan hangen. Vandaar dat ik ben gaan kijken via de issue body van 'dit is de PARP van issue..'.*

S Ja.

Q *Dat was in eerste instantie mijn idee dat de PARPs een beetje als de feature identifier zouden gelden. Maar dat werkte — naja, er waren niet genoeg PARPs meer eigenlijk.*

S Nee, precies.

Q *Toen ben ik dus op de labels verder gegaan.*

S Kan je niet anders — want — dat kan ook nog wel handig zijn, dat je juist de andere kant op zoekt. Dat je begint bij een file, en dan dus ziet wat voor issues of labels er allemaal aan gekoppeld waren.

Q *Ja, en waar zou je dat voor willen gebruiken denk je?*

S Nou, dat ik gewoon kan zeggen van 'joh, kijk naar dit bestand, hier zit de core van de logica'. En dat iemand dus inderdaad dat bestand erbij pakt in PhpStorm maar daarnaast ook van daaruit kan kijken van wat is dan gerelateerd—

Q *Oh ja. Zo ja.*

S Je hebt dus inderdaad de workflow [redacted]TransactionCreate. Je kan dat ook wel door dan in PhpStorm dan de git history te pakken, de issue te zoeken en dat dan hier in te vullen.

Q *Ja, dat is net wat omslachtiger. Ja, fair. En dat is nu geen search entrypoint. Oke, fair. Had je anderzijds nog feedback dat je zegt van —*

S Niet meer. [Small discussion about contribution of senior developers to total backend]

Q *Hij exclude trouwens de merge commits. Het is dus niet dat bij elk stukje code een senior staat per se.*

S Even kijken, hij pakt het op basis van de commits in eerste instantie toch, en dan van daaruit de issues zoeken en dan labels?

Q *Ja.*

S Maar dat zou je dan potentieel ook om kunnen bouwen dat ie eigenlijk commits en issues pakt waar geen feature aan gelinked is. Niet voor mensen om de structuur te begrijpen, maar meer omdat ik hier een aantal van die features pak en denk 'joh, daar zit veel meer changes zitten er onder dan hier zichtbaar zijn' omdat het gewoon niet goed gelabeled is in GitLab. Ik had net een mooie, de Business-2-Business SDD, dat is een "Feature: B2B Direct Debits" en die heeft vier changed files. Nou, ik weet zeker dat dat veel meer was dan dat. Ja, en dat suggereert eigenlijk dat er gewoon er op het originele issue geen feature label stond.

Q *Ja. Dus dan zou je via de history van een file kunnen proberen te traceren van welk issue hier dan nog meer bij betrokken is geweest. Dat dat de mother issue is geweest of zo?*

S Dat, maar ook gewoon dat je — ergens is dan een commit waar een B2B SDD methode is toegevoegd, en daar begon het dan allemaal mee maar dat kan je nu — zo'n feature kan je hier niet goed terugvinden.

Q *Ja.*

S Ja en dat misschien, wat ik ook al eerder zei, andersom zoeken op basis van een file. Want dit was een issue met — hier staat een issue met dus inderdaad gewoon een duidelijke bugfix [title of issue] dus er is dan een Workflow die relevant is geweest en dan de git history daarvan, zou interessant kunnen zijn. Goed dan kan je dus, dat is met alles eigenlijk, je kan alles ook wel in GitLab vinden maar dit is wel wat fijner om doorheen te klikken, om een beetje te browsen. Dat scheelt wel.

Q *Dat was je feedback?*

S Dat was het.