Praktikumsbericht

von

Janik Teune

Matrikelnummer: 233966

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Das Schreiben von Programmen hat mich schon lange interessiert, doch während der ersten Semester konnte ich an der Universität nur wenig praktische Erfahrung sammeln. Die Module *Einführung in die Informatik* und *Algorithmen und Datenstrukturen* konnten grundlegende Konzepte des Programmierens anhand der Programmiersprache *Java* vermitteln. Die dort vorgestellten Probleme und Übungsaufgaben konnten allerdings in wenigen Zeilen Code gelöst werden. Das Arbeiten an einem größeren Projekt und in einem Team habe ich im Studium nur wenig kennengelernt.
Außerdem wurden eher grundlegende Strukturen der Programmiersprache *Java* vermittelt. Moderne und neue Ideen habe ich kaum kennengelernt.

Von meinem Praktikum bei *valantic FSA* erhoffte ich mir moderne Konzepte beim Programmieren kennenzulernen, sowie die Arbeit an einem großen Projekt zu erfahren. Das Unternehmen habe ich durch das Sponsoring des Programmierwettbewerbs kennengelernt, welcher im Rahmen des Moduls *Algorithmen und Datenstrukturen* stattfand. Das freundliche Auftreten der Mitarbeitenden sowie die Empfehlung eines Kommilitonen überzeugten mich schließlich, mich bei *valantic FSA* zu bewerben. Nach einem erfolgreichen Bewerbungsgespräch konnte ich am 10.01.2022 meinen ersten Arbeitstag und meine sechsmonatige Probezeit in der Abteilung *iQbonds* beginnen.

1.2 Tätigkeitsübersicht

Zu meinen Aufgaben gehört:

- Spezifikation des Verhaltens neuer Anforderungen
- Implementation dieser Spezifikationen
- Fixen von Bugs
- Code Reviews
- Migration alter Projekte
    - Von *SVN* zu *Git*
    - Von *Ant* zu *Gradle*
    - Von älteren *Java* Versionen zu *Java 21*

2 Unternehmensprofil

Die *valantic GmbH* ist ein Unternehmen, welches Beratungsdienstleistungen und Software in verschiedenen Branchen anbietet. Auf der Webseite sind unter anderem die Baubranche, Chemische und Pharmazeutische Industrie, das Finanzwesen, die Konsumgüterindustrie und die Kulturwirtschaft vermerkt. (1; 2)

valantic Financial Services Automation (kurz *valantic FSA*) ist ein Privatunternehmen, das

zur *valantic GmbH* gehört und Finanzdienstleistungen anbietet. Es beschäftigt über 300 Mitarbeitende in 9 Büros in Europa und Nordamerika. Das Büro in Magdeburg befindet sich auf dem Werder in der Mittelstraße 10. Der Hauptsitz des Unternehmens befindet sich in Frankfurt am Main. (3; 4)

valantic FSA beschäftigt sich mit der Automatisierung von Prozessen in der Finanzindustrie. Die angebotene Software richtet sich an Banken. Das Ziel der Firma ist es, die Wertetröme ihrer Kunden zu digitalisieren und weiterzuentwickeln. Damit will *valantic FSA* die Effektivität und Agilität verbessern sowie das Verständnis dafür steigern. (4; 5)

3 Aufgaben

Ich arbeite in der Abteilung an verschiedenen Projekten, selten helfe ich auch noch in anderen Abteilungen aus.

Einen großen Teil meiner Arbeitszeit verbringe ich am *Diagnosetool*, welches vor allem von der Test-Abteilung verwendet wird. Die Mitarbeitenden können das Tool nutzen, um verschiedene Performance-Counter aus Log-Dateien auszulesen und sich diese tabellarisch und graphisch anzeigen zu lassen.

Häufig setze ich auch neue Anforderungen der Kunden um, zum Beispiel im *iQbonds-Client*. Während ich als einziger Entwickler am *Diagnosetool* arbeite, handelt es sich beim *iQbonds-Client* um ein großes Projekt, an dem seit vielen Jahren von vielen verschiedenen Mitarbeitenden gearbeitet wird. Um die Arbeit als Team an diesem Projekt zu ermöglichen, wird das gewünschte Verhalten eindeutig spezifiziert. Die Spezifikation wird vor Beginn der Implementation abgeschlossen und kann dann aus verschiedenen Perspektiven verifiziert werden. Unser Dokumentations-Team überprüft das Dokument auf sprachlicher Ebene, ein weiterer Entwickler verifiziert die technischen Details und das Projektmanagement überprüft, ob das spezifizierte mit dem gewünschten Verhalten übereinstimmt. Während implementiert wird, kann jemand aus der Test-Abteilung anhand der fertigen Spezifikation diverse Testfälle ableiten, um das Verhalten des Programms zu verifizieren. Durch diese Vorgehensweise werden viele Bugs früh erkannt, Verhalten ist eindeutig dokumentiert und die Arbeit im Team läuft organisiert ab.
Nach dem Abschluss einer Implementation oder nach dem Fixen eines Bugs werden die Änderungen in *Gitlab* reviewt. Das wird meist von meinem Ansprechpartner oder einem anderen Werksstudenten gemacht. Auch ich reviewe regelmäßig den Code von anderen Mitarbeitenden.

3.1 Aufgabe – Speichern von Log und Konfigurationsdateien

Im *iQbonds-Client* hatte der Kunde die Möglichkeit, die Konfiguration seiner Client-Instanz in ein ZIP-Archiv zu schreiben und dieses zu speichern. Damit kann zum Beispiel

der Zustand dieser Instanz besser nachvollzogen und das Nachstellen eines Bugs erleichtert werden.
Diese Funktion habe ich erweitert. Mit der *java-swing-API* habe ich einen Dialog gebaut, mit dem der Kunde mithilfe von Kontrollkästchen auswählen kann, welche Dateien in ein ZIP-Archiv hinzugefügt werden.
Die Anforderung habe ich zunächst spezifiziert, sodass die Test-Abteilung diverse Blackbox-Tests vorbereiten und das spezifizierte Verhalten vom Projektmanagement bestätigt werden kann, und danach implementiert.
Der Kunde hat nun die Optionen

- Die Konfigurationsdateien des Clients zu speichern
- Die Log-Dateien des Clients zu speichern
- Ältere archivierte Log-Dateien zu speichern
- Die Konfigurationsdateien des *iQbonds-Servers* anzufragen und diese ebenfalls zu speichern

Das Speichern der Dateien des Clients war einfacher, da alle Daten zur Erstellung der zu speichernden Dateien bereits vorliegen. Zur Speicherung der Serverkonfiguration muss allerdings eine Nachricht an den *iQbonds-Server* gesendet werden. Danach muss auf die Antwortnachricht des Servers gewartet, diese Nachricht ausgewertet und transformiert werden und schließlich die Serverkonfigurationsdatei im XML-Format erstellt und zum ZIP-Archiv hinzugefügt werden.
Zusätzlich habe ich im Zuge dieser Anforderung den *FileChooser-Dialog* der *java-swing-API* erweitert. Die Erweiterung validiert den eingegebenen Dateinamen und kann eine Auswahl an Fehlermeldungen anzeigen. Sollte der Dateiname invalide sein, so wird der Bestätigungsknopf deaktiviert. (6)

3.2 Aufgabe – Der *Rangeslider*

Wenn das Diagnosetool Log-Dateien importiert, kann man einen Start- und Endzeitpunkt für die Auswertung der Daten auswählen. Für diese Auswahl habe ich einen *Rangeslider* implementiert. Der *Rangeslider* funktioniert genauso wie der *JSlider* aus der *java-swing-API*. Der Unterschied ist, dass es 2 Knöpfe auf dem Slider gibt. Die Knöpfe können nicht aneinander vorbeigezogen werden. So kann der linke Knopf zur Auswahl der Startzeit und der rechte Knopf zur Auswahl der Endzeit verwendet werden. Die Herausforderung bei dieser Aufgabe lag dabei, dass sowohl die Logik als auch die Darstellung der Knöpfe implementiert werden musste. (7)

3.3 Aufgabe – *Git*-Migration der *User Guides*

Das Dokumentationsteam hat für jedes Produkt zusätzlich ein eigenes Projekt *User Guide*. In diesen Projekten arbeitet das Dokumentationsteam mit einem externen Tool, um den Kunden in verschiedenen Varianten (PDF, HTML5, …) einen umfassenden Guide zu dem entsprechenden Produkt in Deutsch und Englisch zur Verfügung zu stellen.
Zur Versionsverwaltung der *User Guide*-Projekte wurde *Subversion* (kurz: *SVN*) verwendet.
Meine Aufgabe war es, jedes einzelne dieser Projekte von *SVN* zu *Git* zu migrieren. Dafür

wurde mir ein Tool *svntogit* zur Verfügung gestellt, welches mir viel Arbeit abnehmen konnte.
Der grobe Ablauf für die Migration eines einzelnen Projektes ist der folgende:

1. *svntogit* ausführen
    a. Den alten Projektpfad angeben
    b. Den neuen Projektpfad angeben
    c. Die Ausführung dauert mehrere Stunden
    d. Das Ergebnis ist ein *Git*-Repository mit dem Namen *repo*
    Alle Commits, Branches und Tags aus dem alten *SVN*-Repository wurden übernommen
2. Das neue *Git*-Repository muss nun dem Projekt entsprechend umbenannt werden
3. Die *.gitignore* und *.gitattributes* Dateien müssen erstellt werden
4. Das Projekt muss auf *Gitlab* angelegt werden
5. Das lokale Repository muss auf *Gitlab* gepushed werden
6. Die Pipeline für das Projekt muss aufgesetzt werden
7. Auf dem alten *SVN*-Repository muss markiert werden, dass dieses Projekt migriert wurde

Da es sehr viele *User Guide* Projekte gibt, habe ich die Projekte nicht alle händisch migriert. Stattdessen habe ich ein *Batch*-Skript geschrieben, was anhand einiger Parameter die Schritte der Reihe nach für mich durchführt.
Bei der Aufgabe habe ich mich mit *Windows-Batch*-Skripten, *Subversions* und *Gitlabs* Kommandozeileninterface und mit der *Git*-Pipeline auseinandergesetzt. Außerdem habe ich die Arbeitsstruktur des Dokumentationsteams kennengelernt.
Schließlich habe ich das Dokumentationsteam im Umgang mit *Git* und *Gitlab* geschult.

3.4 Aufgabe – *AppInvoker Simulator*

Eine komplexe Anforderung, die ich für unsere Test-Abteilung umgesetzt habe, ist der *AppInvoker Simulator*. Der *AppInvoker* ist eine Komponente, welche definierte Befehle, die vom Kunden im *iQbonds-Client* konfiguriert werden können, an die firmenexternen Systeme *Bloomberg* oder *Reuters Eikon* schicken kann. Der Kunde kann jeden konfigurierten Befehl in einem Menü im Client auswählen. Entsprechend wird der Befehl über den *AppInvoker* entweder an *Bloomberg* oder an *Eikon* gesendet.
Wenn ein Testentwickler das Verhalten unserer Software auf eine Antwort von *Bloomberg* oder *Eikon* testen möchte, war es nötig, eine Verbindung dorthin herzustellen, und auf die Antwortnachricht zu warten. Mit dem Simulator soll den Terstern Zeit und Arbeit erspart werden.
Dafür verbindet sich der *AppInvoker Simulator* über eine separate Netzwerkverbindung zum *iQbonds-Server* und meldet sich als *AppInvoker* an. In der graphischen Nutzeroberfläche des Simulators, die ich ebenfalls implementiert habe, kann ausgewählt werden, ob eine Verbindung zu *Eikon* oder zu *Bloomberg* simuliert werden soll. Der Server denkt nun, es gäbe über den *AppInvoker* eine Verbindung zu einem der beiden Systeme und es kann ein Nachrichtenaustausch zwischen Simulator und *iQbonds-Server* stattfinden.

Alle Nachrichten, die an *Eikon* oder *Bloomberg* gesendet werden sollen, werden im *AppInvoker Simulator* in einer Liste angezeigt. Ein Tester kann für jedes Element dieser Liste einzeln eine Antwort auswählen, die zurück an den Server geschickt werden soll. Alle beim Simulator ein- und ausgehenden Nachrichten werden geloggt und können in einem extra Fenster angezeigt werden.
So kann auf einfache Art und Weise das Verhalten getestet werden, ohne von firmenexternen Systemen abhängig zu sein.
Zur Umsetzung dieser Aufgabe habe ich mich zunächst mit dem Nachrichtenprotokoll auseinandergesetzt, welches zum Austausch zwischen *AppInvoker* und *iQbonds-Server* definiert ist. Dann habe ich das Verhalten des Simulators spezifiziert. Als nächstes habe ich eine mögliche Nutzeroberfläche für den Simulator entworfen und die entsprechenden Elemente ebenfalls zur Spezifikation hinzugefügt. Nachdem ich die verschiedenen Reviews der Spezifikation ausgewertet hatte, habe ich den Simulator implementiert. Schließlich wurde meine Implementation von der Test-Abteilung getestet und das Verhalten mit meiner Spezifikation abgeglichen.

4 Arbeitsbedingungen

Nach einer kurzen Einführung im Büro musste ich aufgrund der Corona-Pandemie zunächst im Home-Office arbeiten. Trotzdem hatte ich meinen eigenen Arbeitsplatz im Büro. Nachdem sich die Lage entspannt hatte, wurde mir freigestellt, weiterhin im Home-Office zu arbeiten oder an meinen Arbeitsplatz im Büro zu wechseln. Bis auf wenige Ausnahmen habe ich meinen Arbeitsplatz im Büro verwendet. Viele Kollegen entschieden sich weiterhin von zuhause aus zu arbeiten.
Über die Wintersaison 2023/2024 wurde das Büro renoviert. Die Küche wurde neu gemacht und die Büros wurde neu aufgeteilt. Außerdem wurde eine Desk-sharing Policy eingeführt. Seitdem habe ich keinen eigenen Arbeitsplatz mehr. Stattdessen habe ich von der Firma einen Laptop zur Verfügung gestellt bekommen, den ich an jedem freien Arbeitsplatz an eine Docking Station anschließen kann. Für Kollegen, die täglich im Büro sind, haben sich im Laufe der Zeit wieder feste Plätze ergeben. Kollegen, die gelegentlich im Büro sind, können sich auf die verbliebenen freien Plätze verteilen.
Auch ich arbeite seit meinem Umzug im April 2024 nur selten im Büro. Für meine Arbeitsmaterialien habe ich ein Schließfach zur Verfügung gestellt bekommen. Ich muss also meinen Arbeitsplatz bei Arbeitsbeginn erst auf- und zum Feierabend wieder abbauen, wenn ich im Büro arbeiten möchte. Durch das Desk-Sharing müssen einige Mitarbeitenden ab und zu mehr Arbeitszeit aufbringen. Im Gegenzug konnte ein großer Teil der Bürofläche abgegeben werden, da durch das Desk-Sharing weniger Arbeitsplätze benötigt werden.

5 Fazit

Insgesamt schätze ich meine Werksstudentenstelle positiv ein. Der Anspruch und die Aufgaben sind abwechslungsreich und ich kann breitgefächert neue Erfahrungen bei der

Arbeit als Softwareentwickler sammeln.
Neue Erkenntnisse konnte ich nicht nur über die Programmiersprache *Java* erhalten, sondern auch über Build-Umgebungen wie *Ant* oder *Gradle*. Im Umgang mit *Git* habe ich gelernt, Pipelines in *Gitlab* aufzusetzen und mit komplexeren Projektstrukturen umzugehen. Dazu gehört zum Beispiel das Arbeiten an verschiedenen Versionssträngen und deren Synchronisation zum Beispiel durch *Cherry Picks*.
Gestört hat es mich, dass ich viel Arbeitszeit dazu aufwenden muss, alten und schlecht dokumentierten Code zu analysieren. Das kommt gerade im *Diagnosetool* häufig vor, da dieses Projekt fast nur von Werksstudenten betreut und dadurch von vielen unterschiedlichen Entwicklern mit teilweise wenig Erfahrung betreut wurde.
Abschließend denke ich, dass mir die Arbeit als Werksstudent bei *valantic FSA* ein gutes Bild davon gibt, wie ein Vollzeitjob als Softwareentwickler aussehen kann.

6 Literaturverzeichnis

1. valantic. ***valantic.*** **[Online] 29. 06 2024. https://www.valantic.com/en/.**
2. Wikipedia valantic. ***Wikipedia.*** **[Online] 29. 06 2024. https://de.wikipedia.org/wiki/Valantic.**
3. LinkedIn valantic FSA. ***LinkedIn.*** **[Online] 29. 06 2024. https://de.linkedin.com/company/valantic-fsa.**
4. About Us: valantic FSA. ***valantic FSA.*** **[Online] 01. 07 2024. https://www.valantic.com/fsa/about-us/.**
5. Wikipedia Finanztechnologie. ***Wikipedia.*** **[Online] 29. 06 2024. https://de.wikipedia.org/wiki/Finanztechnologie.**
6. Oracle Docs - JFileChooser. ***Oracle Java Platform.*** **[Online] 16. 11 2024. https://docs.oracle.com/javase/8/docs/api/javax/swing/JFileChooser.html.**
7. Oracle Docs - JSlider. ***Oracle Java Platform.*** **[Online] 16. 11 2024. https://docs.oracle.com/javase/8/docs/api/javax/swing/JSlider.html.**

Hiermit erkläre ich, dass ich diesen Praktikumsbericht selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als angegeben verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort:
Datum: Unterschrift: