Alexander Hildebrandt, Elias Baalmann

# Praktikumsbericht

Praktikum Agile Softwareentwicklung

25.09.2016

# Inhaltsverzeichnis

# 1 Einleitung

Das Praktikum „Agile Softwareentwicklung“ ist ein Programmierpraktikum welches als 3-wöchige Blockveranstaltung während der vorlesungsfreien Zeit, in unserem Fall des 4. Semesters stattfindet. *„Ziel des Praktikums ist, die für Softwareentwicklung in der Praxis notwendigen technischen, organisatorischen und sozialen Fertigkeiten zu erwerben“*.

Zu programmieren war das rundenbasierte Politik-Persiflage Browsergame *Fraud & Bribery* zu dem ein Ausgangssystem vorgelegt wurde.

Da praktische Programmiererfahrung im Studium sonst hauptsächlich über die Module Softwareentwicklung 1 & 2 erworben wurde, waren die meisten Teilnehmer hauptsächlich mit dem Programmieren als Pair und in der Programmiersprache Java vertraut. Um die nötigen Projektmanagements- und Webentwicklungstechnologien kennen zu lernen wurden die Teilnehmer zu Beginn des Praktikums in kleine Expertengruppen eingeteilt, die jeweils zu einer Technologie einen Vortrag erarbeiteten um diesen den anderen Teilnehmern vorzustellen, um allen einen groben Überblick zu geben und gleichzeitig mindestens eine Expertengruppe für jede Technologie zu haben, die während der Entwicklung für Fragen zur Verfügung stehen sollte. So wurde die erste Woche des Praktikums für das Vorbereiten und Vortragen der Vorträge, das Einrichten der Entwicklerrechner und eine kurze Einarbeitung in das Ausgangssystem genutzt.

Zur Implementation wurden zwei Teams gebildet, aus denen täglich jeweils zwei Teilnehmer die Rolle als Teamleiter einnahmen. Diese hatten die Aufgabe mit dem Kunden die zwei täglichen Meetings abzuhalten, auszudiskutieren welche Features in welcher Zeit fertiggestellt werden könnten und diese dann in Tasks einzuteilen, welche dann auf die ebenfalls täglich wechselnden Programmierpärchen aufgeteilt wurden (siehe Projektmanagement). So wurde insgesamt acht Tage an dem Projekt gearbeitet mit dem Ziel das Praktikum am letzten Praktikumstag mit einer Abschlusspräsentation des Spiels zu beenden.

# 2 Projektmanagement

## 2.1 Trello

Um die Organisation der Teams zu vereinfachen wurden drei Trello [0] Boards eingerichtet. Ein allgemeines Storyboard, in dem die mit dem Kunden besprochenen Features eingetragen wurden, um einen Überblick darüber zu haben was in jeder Iteration beendet bzw. nicht beendet wurde. Und zusätzlich dazu pro Team ein Board, in das die aktuellen Tasks durch die Teamleiter eingetragen wurden. So konnte gewährleistet werden, dass nie mehrere Programmierpärchen am selben Task arbeiteten. Vor der Abarbeitung eines Tasks wurde die benötigte Zeit geschätzt und in das Board eingetragen. Wurde der Task beendet so konnte die tatsächlich benötigte Zeit damit abgeglichen werden.

## 2.2 Git

Die Versionierung des Praktikums wurde durch Git [1] realisiert. Zu diesem Zweck wurde unser Projekt in vier Repositories aufgeteilt. Diese waren, abgesehen von dem „Common“ Repository, voneinander unabhängig, wodurch wir das System in einzelnen Teilen erweiterten und testen konnten. Dadurch war es uns auch möglich, Konflikte beim Programmieren auf kleine Gruppen zu beschränken. Allerdings hat diese Aufteilung später im Praktikum für eine Komplikation gesorgt, als wir eine seitenübergreifende Navigationsleiste einbauen wollten. Die Logik dieser Funktion wurde in einem unabhängigen Repository angelegt, weshalb es in den anderen Repositories nun zu einer Abhängigkeit kam, falls man mit der Navigationsleiste arbeiten wollte.

Für einen sicheren Umgang mit der Software hielten mehrere Gruppen Vorträge, um unser Team mit den Grundfunktionen bekannt zu machen. Zusätzlich hatten viele Teammitglieder auch in anderen Projekten bereits Erfahrung mit Git gesammelt, wodurch in der Regel immer jemand zu finden war, falls es doch zu Komplikationen kam. Außerdem half EGit [2] bei der übersichtlichen Darstellung von Git in unserer Entwicklungsumgebung, besonders beim Zusammenführen von Veränderungen an der gleichen Datei. Des Weiteren haben wir mit Feature-Branches gearbeitet, damit eine unfertige Version einer Funktion keine andere Gruppe im gleichen Bereich des Projektes behindern kann. Durch

all diese Vorkehrungen ist es uns gelungen, immer eine stabile Version des Projektes zu besitzen und Konflikte kleinteilig ohne nennenswerte Probleme abzuarbeiten.

# 3 Beispielfeatures

## 3.1 Regionsnamengenerator

Eines der ersten Features das implementiert werden sollte war die zufällige Generierung und Vergabe von Regionsnamen. Die Idee war Regionsnamen durch verbinden eines Anfangsteilwortes und eines Endteilwortes zu generieren, wobei alle Anfangsteilworte gut mit allen Endteilworten verknüpfbar sein sollten. Schnell stellte sich heraus, dass sowohl reale als auch fiktive Staats-/Welt-/Regionsnamen einen guten Anhaltspunkt gaben, so klingen z.B. alle Kombinationen von Belg/Hol/As mit ien/land/erbaidschan nach sinnvollen Regionsnamen (Belgien, Belgland, Belgerbaidschan, Holien, Holland, Holerbaidschan, Asien, Asland, Aserbaidschan). Nach diesem Schema wurden ca. 20 Anfangs- und Endteilworte festgelegt. Die ermöglicht das Generieren von $20^2 = 400$ eindeutigen Regionsnamen. Sollten mehr Regionen erzeugt werden wird ein eindeutiger Standardstring verwendet (eine Kombination aus Land und einer Zahl) sodass jede Region einen eindeutigen Namen hat. Diese Funktionalität konnte vollständig in Java programmiert werden es ergaben sich also kaum Schwierigkeiten, da auch das Testen von Java- Klassen in Softwareentwicklung 2 sehr detailliert behandelt wurde.

## 3.2 Rededuell

Eine der aufwendigsten Funktionen, die wir im Praktikum realisieren sollten, war das Rededuell. Hierbei ging es um ein rundenbasiertes Kampfsystem zwischen zwei Spielern. Nachdem man durch eine Aussage des Gegenspielers angegriffen wurde, muss man die passende defensive Aussage entgegnen, um die Runde zu gewinnen. Danach hat man selbst die Möglichkeit zum Angriff. Der erste Spieler, der eine bestimmte Anzahl von Runden gewinnt, hat das Duell gewonnen.

Die erste große Herausforderung war zum einen die Darstellung der Duell-Historie, welche die bisher gewählten Aussagen beinhalten sollte, und zum anderen das Hinzufügen von neuen Aussagen zu dieser Historie. Zu diesem Zweck haben wir einen Duell-Controller geschrieben, der die benötigten Informationen für die Duell-Historie aus unserer Datenbank ausliest und an die richtige HTML-Seite weitergibt. Hier kam es nun zu ersten

Schwierigkeiten, da wir Thymeleaf [3] benutzen wollten, um die übergebenen Informationen der Duell-Historie zu verarbeiten und anzuzeigen. Allerdings war es uns nicht möglich, die korrekte Syntax für dieses Problem aus anderen Stellen des Projektes, an denen Thymeleaf benutzt wurde, abzuleiten, sodass wir die Hilfe der Thymeleaf-Experten unserer Gruppe brauchten. Dadurch ergaben sich natürlich sowohl für uns als auch für die Experten Verzögerungen in den jeweiligen Zuständigkeitsbereichen.

Nachdem die Arbeit an der Duell-Historie abgeschlossen wurde, mussten wir als nächstes dafür sorgen, dass sich die Duell-Seite nach der Wahl einer Aussage auch bei dem Gegenspieler automatisch aktualisiert. Um dies zu realisieren, haben wir durch WebSockets [4] für eine bidirektionale Verbindung zwischen Client und Server gesorgt. Diese Verbindung wurde dann von uns durch das STOMP [5] Sub-Protokoll erweitert, um ein Abonnement-System zu erlauben. Dank diesem System konnten wir festgelegen, dass die Teilnehmer eines Duells automatisch beim Betreten der Duell-Seite einen bestimmten Pfad abonnieren. Sobald ein Spieler nun eine Aussage trifft, sendet er über WebSockets gleichzeitig eine Nachricht an den Server. Nach Erhalt dieser Nachricht sendet der Server nun eine Nachricht an alle Abonnenten des Pfades und setzt sie damit über eine neue Duell-Aussage in Kenntnis. Allerdings gab es Probleme mit unserem NGINX [6] Web-Server, da WebSockets nicht gut mit dessen Proxys zusammenarbeiten können. Da wir uns vorher nie mit NGINX auseinandersetzen mussten, hat uns dieser Schritt einige Zeit gekostet. Das eigentliche Problem mit den WebSockets war für uns jedoch die Arbeit mit JavaScript, da der Großteil des benötigten Codes in dieser Sprache geschrieben werden musste und wir beide noch keine Erfahrung mit ihr hatten. Deshalb mussten wir abermals die Hilfe eines Experten beanspruchen. Zusätzlich sorgte die aktuelle Implementation der Abonnements dafür, dass jedes Duell alle anderen Duell-Seiten aktualisieren konnte, da es keine individuellen Pfade für jedes einzelne Duell gab. Allerdings mussten wir wegen Zeitdruck zuerst das Duell funktionsfähig machen, bevor die Abonnements ausgebessert werden konnten.

Der letzte große Schritt für das Rededuell nach der Darstellung und Aktualisierung war nun die Logik selbst. Ein anderes Pair hatte bereits eine Service-Klasse geschrieben, die von uns nur noch in den Duell-Controller eingebunden werden musste. Nachdem wir diese Klasse ausreichend studiert haben, gab es keine nennenswerten Probleme bei der Einbindung und das Rededuell war nun funktionsfähig. Zu guter Letzt wollten wir aufge-

schobene Aufgaben am Duell abarbeiten, aber die Logik wurde erst am letzten Programmiertag fertig gestellt, weshalb nur noch Zeit blieb, um einen „Duell aufgeben“-Button zu implementieren.

# 4 Fazit

Die zu entwickelnden Funktionen im Praktikum variierten realitätsgemäß in Umfang und Komplexität. Wie an den zwei Beispielen im funktionalen Teil dieses Berichts zu sehen ist, fingen unsere Programmieraufgaben in der Regel klein an und beschränkten sich auf wenige Frameworks. Mit der Zeit wurden die Anforderungen der Kunden immer anspruchsvoller, weshalb einzelne Funktionen von immer mehr Frameworks abhängig waren und mehr Zeit gekostet haben. Diese Entwicklung sorgte auch dafür, dass spätere Funktionen von mehreren Pairs bearbeitet werden mussten, wodurch es an den letzten Programmiertagen immer öfter zu Bottlenecks gekommen ist.

Eines der größten Probleme im Praktikum war der Zeitdruck. Da es nur acht Programmiertage gab und die Kunden fast jeden Tag neue Funktionen verlangt haben, durfte nicht viel Zeit verschwendet werden. Leider gab es außerhalb unserer Kontrolle viele Aspekte, die für Zeitverlust gesorgt haben.

Zum einen waren die Vorträge am Anfang des Praktikums zu lang. Sie mussten vor dem ersten Programmiertag gehalten werden, da jedes Thema schon früh relevant werden konnte. Das hatte zur Folge, dass die Vorträge hintereinander in großen Blöcken präsentiert wurden, weswegen eine angemessene Konzentration für die Zuhörer bereits nach wenigen Vorträgen erschwert wurde. Man hätte viel Zeit sparen können, wenn wir keine Vorträge, sondern nur die Handouts der Themen hätten erarbeiten sollen. Dadurch würden sich die Erzeuger der Handouts immer noch genug Wissen über ihr Thema aneignen, um als Experte für das Praktikum fungieren zu können und man hätte zusätzlich ein paar Programmiertage mehr gewonnen.

Zum anderen waren wir trotz der Vorträge nicht ausreichend über die Frameworks des Praktikums informiert, wodurch die bereits aufwendigen Funktionen ebenfalls Einarbeitungszeit erzwungen haben. Die Einarbeitungszeit hätte allerdings verringert werden können, wenn es zu komplexeren Frameworks bereits mehr Beispiele im vorgefertigten Code gegeben hätte. Dies hätte dafür gesorgt, dass man keine Zeit beim Suchen eines Beispiels verschwendet und dass das Beispiel, mit dem man arbeitet, bereits im gleichen Kontext wie die zu entwickelnde Funktion liegt.

Mehr Beispiele im Code hätten außerdem das Potential, die Vorträge am Anfang des Praktikums zu verbessern. Die Vortragenden könnten ihre Themen dadurch anhand des

Projektes anschaulich erklären und die Zuhörer würden gleichzeitig einen Überblick des vorgefertigten Codes erhalten. Die nach den Vorträgen geplante Einarbeitung in das Projekt würde dank diesem Überblick auch vereinfacht werden, wodurch abermals Zeit gespart werden könnte.

Ein weiterer zeitaufwendiger Aspekt des Praktikums war das tägliche Wechseln der Projektmanager. Es ist wahrscheinlich eine durchaus bereichernde Erfahrung, einmal Projektmanager einer Softwareentwicklergruppe zu sein. Allerdings bietet ein Tag bei weitem nicht genug Zeit, um tatsächlich ein Gefühl für die Aufgaben zu entwickeln und es gibt sicherlich viele Teilnehmer, die keine Aspiration dafür haben, jemals als Projektmanager zu arbeiten. Man hätte auch hier Zeit sparen können, indem man einzelnen Teilnehmern die Möglichkeit gibt, mehrere Tage als Projektmanager zu fungieren und dafür anderen Teilnehmern diese Aufgabe erspart.

Alles in allem war das Praktikum eine durchaus akkurate Simulation von realistischer Projektarbeit. Trotz vielen Entscheidungen, die unsere potentielle Arbeitszeit verringert haben, waren die acht Programmiertage ein bereichernder Überblick in viele Aspekte moderner Programmierung.

# Anhang

## Literaturverzeichnis

[0] Trello: https://trello.com/

[1] Git: https://git-scm.com/

[2] EGit: http://www.eclipse.org/egit/

[3] Thymeleaf: http://www.thymeleaf.org/

[4] WebSockets: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API

[5] STOMP: https://stomp.github.io/

[6] NGINX: https://www.nginx.com/resources/wiki/