Aufgabenverwaltung besser machen!

Qualitätssicherungsdokument

Gruppe 21:	Pascal Fleckenstein <pascal.fleckenstein@stud.tu.darmstadt.de> Ksenia Panarina <ksenia.panarina@stud.tu-darmstadt.de> Frederik Röper <frederik.roeper@stud.tu-darmstadt.de> Florian Sens <florian.sens@stud.tu-darmstadt.de> Alexander Ziesing <alexandersteffen.ziesing@stud.tu-darmstadt.de>
Teamleiter:	Florian Fähnrich <florian.faehnrich@stud.tu-darmstadt.de>
Auftraggeber:	Dr. Wolfgang Heenes <heenes@informatik.tu-darmstadt.de> Fachschaft Informatik <bp@d120.de> Fachbereich Informatik
Abgabedatum:	10.02.2019

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelor-Praktikum WiSe 2018/2019
Fachbereich Informatik

Inhaltsverzeichnis

1 Einleitung

Das Ziel unseres Projektes „Aufgabenverwaltung besser machen!" ist die Erstellung einer neuen Verwaltungssoftware für die Fachschaft Informatik zur Verwaltung, Planung und Organisation ihrer Aufgaben und zur Koordination verschiedener Projekte. Diese Verwaltungssoftware soll zum Abschluss des Praktikums in eine Website integriert werden und direkt von der Fachschaft zur Aufgabenverwaltung eingesetzt werden können. Dabei werden wir uns bezüglich der grundlegenden Funktionalität an der Vorgängersoftware „Trac" orientieren: Wir werden ein Ticketsystem zur Aufgabenverwaltung entwickeln, das auch Optionen zur Mitgliederverwaltung bietet und dieses über eine Website erreichbar machen. Dazu werden wir uns auf zwei für den Auftraggeber entscheidende Aspekte fokussieren: Wir werden eine benutzerfreundlichere Website bieten, welche Interaktionen mit der Software angenehmer gestaltet, sowohl für neue, als auch für erfahrene Nutzer. Dafür ist uns wichtig, dass die Website mit wenigen Klicks zu navigieren ist. Weiterhin werden wir das Produkt von Grund auf neu entwickeln, wobei wir von dem Auftraggeber vorgeschlagene Tools verwenden, um die Wartbarkeit nach Ende des Projektes zu garantieren und einen langfristigen Einsatz unserer Website zu ermöglichen. Eingesetzt wird unsere Website dabei für Projekte unterschiedlicher Größe, von der Orientierungswoche mit vielen kleinen Unteraufgaben, bis hin zu wöchentlichen Terminen und Aufgaben. Dazu wird die Website das angesprochene Ticketsystem bieten. Hier können einzelne Aufgaben als Ticket angelegt und verwaltet werden. Zur Verwaltung erhält jedes Ticket einen Status und eine eigene Historie, in welcher Änderungen an dem Ticket eingesehen werden können. Weiterhin besteht die Möglichkeit Tickets zu kommentieren, eine Deadline festzulegen oder wiederkehrende Tickets zu erstellen, welche sich in einem festgelegten Intervall neu öffnen. Ebenso wird die Website Optionen zur Mitgliederverwaltung bieten. So können Tickets an einzelne Mitglieder oder Gruppen von mehreren Mitgliedern zugewiesen werden, wobei die Gruppen auf der Website selbst erstellt und verwaltet werden können. Nutzer werden über solche Zuweisungen oder andere für sie relevante Aktionen, wie zum Beispiel das Kommentieren eines ihnen zugewiesenen Tickets, durch eine E-Mail informiert. Zudem soll die Website in die bereits bestehende Infrastruktur der Fachschaft eingebunden werden. Dies ermöglicht es Nutzern mittels der bereits in der Nutzerdatenbank der Fachschaft hinterlegten Daten zu verifizieren und erschafft eine Anbindung an das Mailsystem der Fachschaft.

2 Qualitätsziele

Im Rahmen des Bachelorpraktikums fokussieren wir uns auf die zwei Qualitätsziele Benutzbarkeit und Wartbarkeit. Beide sind aufgrund des häufig wechselnden Personals in der Fachschaft wichtig, um Übergänge zwischen den für die Wartung verantwortlichen Mitgliedern zu vereinfachen und neuen Mitgliedern eine einfache Einarbeitung zu ermöglichen.

2.1 Benutzbarkeit

Die Benutzbarkeit ist unser wichtigstes Qualitätsziel. Unser Projekt ist entstanden, da unser Auftraggeber eine neue Verwaltungssoftware benötigt, was vor allem durch die mangelnde Benutzbarkeit der Vorgängersoftware notwendig ist. Daher wollen wir eine Verwaltungssoftware entwickeln, die übersichtlich und einfach zu navigieren ist. So wollen wir den zukünftigen Nutzern, den Fachschaftsmitgliedern, eine bessere Übersicht über die anstehenden Aufgaben geben, sodass diese koordinierter abgearbeitet werden können. Ebenso ist es uns wichtig, dass die Interaktion mit der Software ohne Vorkenntnisse bedienbar ist. Aufgrund der häufig wechselnden Mitglieder der Fachschaft ist es von hoher Bedeutung, dass die Website keine lange Einarbeitung erfordert.

2.1.1 Maßnahme 1: Nutzerstudie

Unsere erste und zugleich wichtigste Maßnahme zur Sicherstellung der Benutzbarkeit ist die Durchführung einer Nutzerstudie. Diese werden wir in Kooperation mit unserem Auftraggeber, der Fachschaft Informatik, durchführen. Dabei erhalten wir direktes Feedback von den zukünftigen Nutzern unseres Produktes und können frühzeitig auf Probleme bei der Benutzbarkeit reagieren. Unsere Teilnehmer sind mit dem Vorgänger unseres Produktes vertraut und können unser Produkt direkt mit diesem vergleichen und uns wertvolle Erkenntnisse dazu liefern. So können wir bei Bedarf zusätzlich erwünschte Features direkt in unser Produkt integrieren, um die Benutzbarkeit zu verbessern. Wir setzen uns für die Studie zwei Ziele: Wir möchten die Schwachstellen unserer Website in Hinsicht auf die Benutzbarkeit identifizieren und potentielle Lösungsansätze erarbeiten. Weiterhin möchten wir einen Katalog an Features erstellen, die entweder direkt von den Nutzern gefordert wurden, oder zur Behebung der identifizierten Schwachstellen beitragen können. Außerdem werden wir unsere zweite Maßnahme, das Nutzerhandbuch(siehe:2.1.2 Ma0nahme 2), zur Benutzbarkeit in die Nutzerstudie einbinden und dadurch validieren. Das Handbuch stellt während der Studie das einzige Hilfsmittel dar, auf das die Tester Zugriff haben. Ebenso werden die Tester während der Studie gebeten sich einmal in das Handbuch einzulesen, auch wenn sie es zum Lösen der Aufgaben nicht benötigt haben sollten.

Prozess: Nutzerstudie
Für eine sinnvolle Durchführung der Studie wird bis Mitte Januar eine Alpha-Version der Software implementiert, die alle grundlegenden Funktionalitäten beinhaltet. Die Umsetzung wird von Frederik sichergestellt. Ebenso soll ein Server bereitstehen, auf den alle Teilnehmer der Studie zugreifen können, um die Funktionalität der Website bei parallelen Zugriffen zu testen. Für

die Einrichtung ist Alexander verantwortlich. Für die Studie werden zudem von Florian Aufgaben ausgearbeitet, die von den Teilnehmern abgearbeitet werden sollen. Zudem wird jedem Teilnehmer der Studie ein Fragebogen zur späteren Auswertung ausgehändigt, dabei hat jeder Fragebogen eine eigene ID, um die Auswertung zu vereinfachen. Beim Fragebogen ist es uns wichtig Fragen zu stellen, die kein Bias erzeugen. Dafür orientieren wir uns stark an der System Usability Scale, kurz SUS[1], die 1986 von der Digital Equipment Corporation veröffentlicht wurde. Die SUS besteht aus einer Likert-Skala und einem zugehörigen Auswertungsschema. In einer Likert-Skala kann der Teilnehmer anhand von fünf Möglichkeiten seine Zustimmung zu der jeweiligen Aussage angeben. Diese Möglichkeiten reichen von 5, „ich stimme voll zu" bis zu 1, „ich stimme überhaupt nicht zu". Pascal und Florian werden die SUS als Basis unseres eigenen Fragebogens verwenden und weitere, an unsere beigelegten Aufgaben angepasste Punkte hinzufügen. Zudem werden sie mehrere Textfelder für Rückmeldung zur Website anfügen. In diesen Textfeldern sind die Nutzer angehalten schlechte Bewertungen in der Skala zu begründen, damit diese später als Referenz für Verbesserungen dienen können. Damit sind alle Vorbereitungen für die Studie abgeschlossen.
Die Studie wird in Kooperation mit dem Auftraggeber organisiert und in Anschluss an die Fachschaftssitzungen am 23.01 durchgeführt und vom gesamten Team begleitet. Dementsprechend werden an der Studie sowie an der Kontrollstudie jeweils etwa 15 Teilnehmer beteiligt sein. Zur Einleitung der Studie wird den Teilnehmern das Aufgabenblatt und der Fragebogen ausgehändigt. Während die Teilnehmer die Aufgaben bearbeiten, werden wir als Team nur beobachten und protokollieren, aber keine Hilfestellungen geben. Den Nutzern stehen also nur das Handbuch und die integrierten Tooltips als Hilfe zur Verfügung. Nach Abschluss der Aufgaben werden die Nutzer gebeten den Fragebogen auszufüllen. Daraufhin können die Nutzer die Website frei erkunden, uns Fragen stellen und direktes Feedback geben. Das Feedback wird von Alexander protokolliert, um es später zusammen mit den Fragebögen auszuwerten. Nach der Durchführung der Studie wird Pascal eine initiale Auswertung der Fragebögen und der Protokolle durchführen. Dafür wird eine Tabelle angelegt, in der für jede der Fragen eingetragen wird, wie viele Teilnehmer den jeweiligen Wert von Eins bis Fünf angekreuzt haben. Zusätzlich wird hier der normierte Durchschnitt angegeben. Normiert bedeutet hier, dass die negativ formulierten Fragen wie „Ich fand das System unnötig komplex." angepasst werden, damit eine einheitliche Auswertung möglich ist. Hierbei werden auch sämtliche Bewertungen die einer Eins entsprechen, also sehr schlecht bewertet wurden, samt Fragebogen ID und Nummer der Frage notiert. Zusätzlich wird der SUS Score kalkuliert. Den SUS Score nutzen wir zur groben Orientierung, nicht aber als festen Maßstab. Dabei betrachten wir alles unterhalb eines SUS Scores von 70 als Indikator, dass Verbesserungen stattfinden müssen. Weiterhin wird er die Kommentarfelder des Fragebogens, sowie die vom Team angefertigten Protokolle durchlesen und dabei zwei Listen anlegen: Eine Liste von Schwachstellen, die erwähnt wurden und eine Liste von Features, die gewünscht wurden. Dabei wird jeweils notiert, von welchem Fragebogen oder von welchem Protokoll der jeweilige Eintrag stammt. Zum Teamtreffen am 29.01. werden wir Lösungsansätze für die in der Studie gefundenen Schwachstellen formulieren. Dafür ordnen wir die vorher aufgelisteten Features den Schwachstellen zu, wenn wir die Möglichkeit sehen, dass das jeweilige Feature eine oder mehrere dieser Schwachstellen beheben könnte. Zudem führen wir für jede gefundene Schwachstelle ein zweiminütiges Brainstorming im Team durch, um eigene Lösungsansätze hinzuzufügen. Ergebnis des Teamtreffens ist somit ein Protokoll, in dem

[1] https://hell.meiert.org/core/pdf/sus.pdf

wir die festgestellten Schwachstellen, mehrere Lösungsansätze für die Schwachstellen und eine Liste von Features die zur Website hinzugefügt werden könnten, festhalten. Dieses werden wir im nächsten Treffen mit dem Auftraggeber am 31.01. besprechen, um festzulegen, welche der gesammelten Lösungsansätze und Features umgesetzt werden sollen. Daraufhin wird Florian zu allen ausgewählten Maßnahmen User Stories anlegen. Anfang März werden wir eine Kontrollstudie durchführen, um zu prüfen, ob seit der ersten Studie eine Verbesserung feststellbar ist. Zur Kontrollstudie setzen wir voraus, dass alle aus der ersten Studie gezogenen und am 31.01. mit dem Auftraggeber festgelegten Maßnahmen umgesetzt wurden, dies wird von Frederik geprüft. Ablaufen wird die Kontrollstudie identisch zur ersten Nutzerstudie. Wir werden den Fragebogen allerdings um ein paar Kommentarfelder erweitern, welche auf den Vergleich zur ersten Studie und seit der ersten Studie hinzugefügte Features abzielen wird. Florian wird dafür die Notizen zum vorherigen Fragebogen auswerten und Änderungs- und Erweiterungsvorschläge sammeln. Diese werden in einem Teamtreffen, mindestens sieben Tage vor Durchführung der Kontrollstudie, besprochen. Dabei wird festgelegt, welche von den vorgeschlagenen Erweiterungen in den Fragebogen aufgenommen werden soll. Florian wird diese Erweiterungen in den Fragebogen integrieren. Nach der Durchführung der Studien haben wir ein klares Bild, was unsere Nutzer von der Website erwarten und wie wir diese Erwartungen erfüllen können, um die Benutzbarkeit zur Zufriedenheit aller zukünftigen Nutzer sicherzustellen.

2.1.2 Maßnahme 2: Handbuch mit Website Integration

Wir werden ein begleitendes Handbuch zu unserer Website bereitstellen. Ein Handbuch wird den Nutzern helfen sich mit der Website vertraut zu machen, um so die Benutzbarkeit zu verbessern. Weiterhin hilft es bei Fehlermeldungen und bietet dazu einen Leitfaden, wie auf Fehlermeldungen reagiert werden muss. In dem Handbuch wird für alle verfügbaren Ansichten erklärt werden, welche Funktionen dort zur Verfügung stehen, wie der Nutzer auf diese zugreifen kann und welche Reaktion der Nutzer von der Website erwarten kann. Zusätzlich werden im Handbuch alle möglichen von uns implementierten Fehlermeldungen aufgelistet und mögliche Ursachen erklärt, sowie Lösungen vorgeschlagen. Das Handbuch wird als separates Dokument bereitgestellt und in die Website integriert. So wird auf jeder Ansicht der zugehörige Abschnitt im Handbuch als Help Button integriert, um so sofortigen Zugriff auf den relevanten Abschnitt zu ermöglichen.

Prozess: Handbuch
Pascal verfasst die erste Version des Handbuches bis zum 22.01. , dabei beschreibt er kurz die verfügbaren Funktionen und wie auf diese zugegriffen werden kann. Um alle implementierten Funktionen abzudecken orientiert er sich an den abgenommenen User Stories und den zugehörigen Beschreibungen. Das Handbuch wird daraufhin in die Nutzerstudie am 23.01. integriert, so können wir das Handbuch erproben und die erste Fassung im Einsatz testen. Zur Validierung unseres Handbuchs wurde der Fragebogen um ein Kommentarfeld für spezifische Rückmeldung zum Handbuch erweitert. Nach Durchführung der Studie wird Florian die Rückmeldung zum Handbuch auswerten. Dafür werden die in den Fragebogen gesammelten Kritikpunkte und gewünschten Inhalte zusammengetragen und im nächsten wöchentlichen Teamtreffen besprochen, wobei wir entscheiden, welche Inhalte wir in das Handbuch aufnehmen. Wichtig für die Entscheidung ist, ob der hinzuzufügende Inhalt bereits im Handbuch selbst, den Tooltips oder der Website für den Nutzer verfügbar ist. Wenn dem so ist wird ein Hinweis in

das Handbuch aufgenommen, wo die Information zu finden ist. Wenn nicht, wird der jeweilige Inhalt in das Handbuch aufgenommen. Alle notwendigen Erweiterungen werden von Florian in das Handbuch eingefügt. So erhalten wir ein umfangreiches Handbuch, welches alle bisher aufgekommen Fragen zur Website beantworten kann.

2.1.3 Maßnahme 3: Front End Design

Ein gutes und strukturiertes Design des Front Ends führt zu einer besseren Benutzbarkeit, da der Nutzer so kaum zusätzliche Hilfestellung benötigt, um die Website zu navigieren. Um dies umzusetzen werden wir uns auf das Framework Bootstrap stützen, da es momentan eines der am weitesten verbreiteten Frameworks für das Front End Design darstellt. So erwarten wir, dass den Nutzern die meisten Designelemente bereits bekannt sind und die Interaktion mit der Website durch einen Wiedererkennungseffekt intuitiv verständlich ist. Weiterhin werden wir möglichst wenig Java Script einsetzen, um Konflikte mit Ad- und Scriptblockern zu vermeiden und eine problemfreie Interaktion mit der Website zu garantieren. Im Rahmen dieses Projektes werden wir die aktuellste Bootstrap Version, Bootstrap 4, einsetzen.
Des Weiteren ist die Navigation auf der Website nach dem „Zwei Klick Prinzip" aufgebaut. Von jedem Punkt auf der Website sollte man alle Bereiche auf der Website innerhalb von zwei Klicks erreichen können. Die Wahl von aussagekräftigen Symbolen, sowie Überschriften wurde möglichst schlicht gewählt und wird durch die Nutzerstudie evaluiert. Der Aufbau der Website unterteilt sich in drei Ebenen. Erstens die Übersicht, bei der dem Nutzer alle wichtigen Informationen angezeigt werden; Zweitens die Interaktion, durch die der Nutzer mit Elementen interagieren kann; Drittens die Bearbeitung, wobei der Nutzer Änderungen und Einstellungen vornehmen kann. Durch dieses Ebenenmodell wird versichert, dass der Nutzer explizit wählt, wie er mit der Website interagieren will, da jede Interaktion nur in der korrespondierenden Ebene möglich ist.

Prozess: Front End Design
Zur Umsetzung werden wir uns beim Design der Website auf die von Bootstrap bereitgestellten Möglichkeiten stützen. Das bedeutet vor allem, dass wir das grundlegende Design des Frontends ausschließlich mit Bootstrap und ohne Java Script implementieren werden. Sollte ein gewünschtes Design oder eine notwendige Funktionalität nicht innerhalb dieser Maßstäbe umgesetzt werden können, informiert der verantwortliche Entwickler den Rest des Teams darüber. Im nächsten wöchentlichen Teamtreffen wird das Problem besprochen und nach einem Lösungsansatz gesucht. Dabei wird zuerst geklärt, welche Ansätze der Entwickler bisher verfolgt hat. Daraufhin führt das Team ein Brainstorming durch, um noch nicht versuchte Lösungsansätze oder alternative Implementierungsmöglichkeiten zu sammeln. Diese werden im Team besprochen und ein sinnvoller Ansatz wird ausgewählt, der bis zum Teamtreffen in der folgenden Woche verfolgt wird. Der Entwickler ändert sich hierbei nicht. Sollte das gesamte Team keinen konkreten Lösungsansatz finden wird abgewägt, wie wichtig die Funktionalität ist. Alle explizit vom Auftraggeber geforderten und von dem gesamten Team als essenziell betrachteten Funktionalitäten werden wir auch, falls nicht anders möglich, außerhalb unserer Maßstäbe implementieren. Wenn es sich um keinen solchen Fall handelt wird die Entwicklung dieser vernachlässigbaren Funktionalität eingestellt, um unsere gesetzten Maßstäbe zu wahren. Die Umsetzung des „Zwei Klick Prinzips" wird von Alexander kontrolliert. Dafür überprüft er nach jeder Iteration, welche Änderungen an den Ansichten in den abgeschlossenen User Stories stattgefunden haben und ob das Prinzip gewahrt ist. Falls das Prinzip verletzt wurde

kontaktiert Alexander den verantwortlichen Entwickler und sucht zusammen mit diesem nach einer Lösung. Dabei prüfen sie, ob die Funktionalität anders in eine der vorherigen Ansichten integriert werden kann. Falls dies nicht möglich ist wird eine eigene Übersicht angelegt, um diese Funktionalität zu implementieren. Durch die Durchführung dieses Prozesses erhalten wir eine Website, die den gesetzten Maßstäben genügt und somit einen schnellen Zugriff auf die gewünschten Funktionen und ein einsteigerfreundliches Design bietet. Dies ermöglicht den Nutzern eine einfache und effiziente Benutzung der Website.

2.2 Wartbarkeit

Die Wartbarkeit hat für den Auftraggeber und somit auch für uns einen hohen Stellenwert. Unser Produkt soll direkt im Anschluss an das Bachelorpraktikum in der Fachschaft als Verwaltungssoftware eingesetzt und für mehrere Jahre verwendet werden. Um einen langfristigen und problemfreien Einsatz der Software zu ermöglichen soll unsere Software gut und ohne große Vorkenntnisse wartbar sein. In einer Fachschaft findet generell ein konstanter Personalwechsel statt, weswegen die Verantwortlichen der Wartungsarbeiten sich voraussichtlich häufig ändern. Daher ist es uns wichtig, dass die Wartung unseres Produktes ohne lange Einarbeitung möglich ist und mögliche auftretende Fehler leicht zu finden sind.

2.2.1 Maßnahme 1: automatisierte Tests

Das Testen ist für die Softwarewartbarkeit eine bedeutende Maßnahme. Dabei wird die Qualität der erstellten oder geänderten Software mit unterschiedlichen Verfahren, Vorgehensmodellen, Testarten und Teststufen vor der Übergabe zur tatsächlichen Nutzung überprüft. Wir werden automatisierte Tests bereitstellen, die bei jeder Aktualisierung des Produktes ausgeführt werden. Damit verfolgen wir zwei Ziele: Einerseits werden wir damit die Grundfunktionalität der Website sicherstellen. Die Grundfunktionalität umfasst hier die Authentifikation mittels LDAP, das Lesen, Erstellen, Editieren und Löschen von Gruppen und Tickets und eine fehlerfreie Kommunikation mit der Serverdatenbank. Andererseits werden wir Tests für alle Funktionalitäten bereitstellen, die wir zum Abschluss des Bachelorpraktikums implementiert haben. Das bedeutet, dass für jede implementierte Funktionalität eigene Tests angelegt werden, die mögliche Interaktionen mit dem Code testen. Für jede einzelne Methode werden wir mehrere Tests bereitstellen, welche das Verhalten der Methode bei möglichen Inputs prüfen. Hierbei werden auch alle Inputs berücksichtigt, die bereits im Codereview gesammelt wurden.

Prozess: automatisierte Tests
Alle Tests werden initial von Ksenia implementiert. Sobald eine User Story abgeschlossen ist, wird Ksenia darüber in Kenntnis gesetzt und sie beginnt mit der Entwicklung der Tests, wobei sie sich nach festgelegten Punkten richtet. Zu jeder Funktionalität werden die Unittests implementiert um alle grundlegenden Komponenten des Programms (Views, URLs, Models) durchzutesten. Dabei wird berücksichtigt, dass alle im Model definierten Funktionen durch die Tests aufgerufen werden und die gelieferten Ergebnisse korrekt sind. Insbesondere die Prüfung der Randfälle, der unerwarteten Datentypen und das Verhalten des Programms bei einem fehlerhaften Input. In diesen Fällen werden spezifische Tests implementiert um sicherzustellen, dass eine entsprechende Fehlermeldung zurückgeliefert wird. An dieser Stelle soll der Tester noch einmal prüfen, ob die Fehlermeldungen ausreichen, um auf die eigentliche Fehlerquelle hingewiesen zu werden. Außerdem werden Integrationstests mithilfe von Selenium entwickelt. Dies ermöglicht die Benutzerinteraktion der Webanwendung im Browser zu simulieren und somit mögliche Probleme bei der Authentifikation, Grundfunktionalität und dem Interagieren unterschiedlicher Komponenten des Programms aufzudecken. Die Tests werden in den Buildordner integriert und mittels Pytest ausgeführt. Beim Einbinden in die Build-Umgebung gilt das Prinzip des Fail-Fast-Ansatzes. Unit-Tests werden als Erstes angegangen, schlägt ein Test fehl, werden alle folgenden in der Hierarchie nicht mehr ausgeführt. Zudem wird nach jeder Iteration die Co-

de Coverage getestet, um die Lücken in der Testabdeckung zu finden. Unser Ziel ist es hierbei, eine Coverage von mindestens 90% beizubehalten. Im Falle einer durch die Tests aufgedeckten geringeren Coverage werden die Ausgaben von Pytest-Coverage überprüft und anhand dieser zusätzlichen Tests entwickelt, die möglichst jede Zeile im Quellcode aufrufen. Sobald die Implementierung abgeschlossen wurde oder die Tests während der Testentwicklung fehlgeschlagen haben, informiert Ksenia den verantwortlichen Entwickler. Nach dem erfolgreichen Abschluss des Testens überprüft dieser Entwickler, ob die Tests alle relevanten Methodenaufrufe abdecken. Falls die Tests fehlschlagen, wird überprüft, ob der zu testende Code oder die Tests selbst einen Fehler beinhalten. Falls die Tests korrekt sind und ein Fehler im Code vorliegt, wird eine User Story erstellt, in welcher der Fehler beschrieben und ein Verweis auf den anschlagenden Test angehängt wird. Diese User Story wird dem ursprünglichen Entwickler des fehlerhaften Codeausschnittes zugewiesen. In dem Buildordner auf GitHub wird Travis CI installiert, um somit bei jeder durch Git ausgelösten Änderung des Ordners automatisch die Tests auszuführen und dadurch sicherzustellen, ob die Codeänderungen das Verhalten der Anwendung beeinflusst haben.

2.2.2 Maßnahme 2: Codereview

Durch den Codereview möchten wir verständlich geschriebenen und ausführlich dokumentierten Code erhalten, der für in Python und Django erfahrene Entwickler einfach nachzuvollziehen ist. Einfach nachvollziehbarer Code ist eine Grundlage der guten Wartbarkeit, da es erheblich einfacher ist Fehler im Code zu finden und zu beheben, den man leicht versteht. Die Codereviews werden nach Abschluss einer Iteration für alle abgeschlossenen User Stories durchgeführt. Dabei wird jede einzelne User Story einem Codereview unterzogen, indem eine Checkliste abgearbeitet wird. Das Design der Checkliste orientiert sich am Beispiel einer Codereview Checkliste der University of Washington[2]. Dabei haben wir die zu prüfenden Punkte an unser Projekt angepasst. Unsere Checkliste, sowie die Beschreibung der einzelnen Punkte, befindet sich im Anhang.

Prozess: Codereview
Der Codereview ist durch das wöchentliche Teamtreffen organisiert, dabei werden alle seit dem letzten Treffen abgeschlossenen User Stories besprochen und zur Kontrolle an einen anderen Entwickler weitergegeben, der nicht an der Bearbeitung beteiligt war. Dabei wird der Reihe nach ein Teammitglied ausgewählt, um die Kontrolle durchzuführen. So wird auch die Durchführbarkeit der Kontrollen sichergestellt. Der Kontrolleur schaut nun im Git nach, welche Stellen im Code bei der Umsetzung der ihm zugeteilten User Story bearbeitet wurden und kontrolliert, ob die bearbeiteten Codeausschnitte den Anforderungen der Checkliste entsprechen. Dafür beginnt er bei den letzten im Git dokumentierten Änderungen. Falls er feststellt, dass ein Ausschnitt nicht den Anforderungen entspricht, notiert er in der Checkliste die jeweilige Datei und Zeile. Während der Kontrolle prüft der Kontrolleur, ob ihm potentielle Fehlerquellen auffallen, die noch nicht durch die Checkliste abgedeckt sind. Sollten ihm solche auffallen fügt er diese als Frage zu der Checkliste hinzu. Sobald er alle bearbeiteten Abschnitte kontrolliert hat informiert er den Entwickler der User Story über das Ergebnis. Sollten Mängel vorliegen, muss der Entwickler diese bis zum nächsten Teamtreffen ausbessern. Falls sich der Entwickler und Kontrolleur bei einem Punkt uneinig sind, wird ein drittes Gruppenmitglied einbezogen, um

[2] https://courses.cs.washington.edu/courses/cse403/12wi/sections/12wi_code_review_checklist.pdf

das Anliegen zu klären. Um sicherzustellen, dass der Entwickler dafür ausreichend Zeit hat, muss der Kontrolleur die Rückmeldung innerhalb von drei Tagen nach Beginn des Codereviews abgeschlossen haben. Nach dem nächsten Treffen wird die User Story dann von einem neuen Entwickler kontrolliert. Durch die Durchführung dieses Prozesses erhalten wir verständlichen Code, der unserer Prüfung standgehalten hat und eine gute Grundlage zur Wartbarkeit unserer Software bietet.

A Anhang

Codereviews Checklist

Review Information:
Prüfer:
Entwicklers:
Geprüfte Dateien:
Geprüfte User Story:

- ☐ **Coding Stil**
  - ☐ Verständlich
  - ☐ korrekte Einrückung
  - ☐ keine überlangen Codezeilen
  - ☐ Methoden haben keine überflüssigen Parameter
  - ☐ keine unnötigen Variable
  - ☐ Benennung der Variablen, Methoden und Funktionen
    - ☐ passende Namenswahl
    - ☐ nur Underscores (kein Camelcase etc.)
  - ☐ Abstand zwischen Methoden
- ☐ **Kommentare**
  - ☐ keine unnötigen Kommentare (veraltet, redundant)
  - ☐ Abhängigkeiten dokumentiert
    - ☐ veränderte externe Parameter
  - ☐ keine überlangen Kommentarzeilen
  - ☐ kein auskommentierter Code
  - ☐ ausreichend Detailliert
    - ☐ Methoden Parameter
    - ☐ Methoden Rückgabe
    - ☐ Variablen
    - ☐ Schleifen
- ☐ **Fehlermeldungen**
  - ☐ verständliche Fehlermeldungen
  - ☐ Randfälle sind abgedeckt (null,0,negativ)

Notizen:

Beschreibung zur Checklist

Coding Stil:
Verständlich: Der Code muss für den Prüfer nachvollziehbar sein. korrekte Einrückung: Der Code muss die festgelegte Einrückung vorweisen. Das bedeutet, dass jede Scope durch ein zusätzliches Tab von der vorherigen Scope getrennt ist.
keine überlangen Codezeilen: Eine einzelne Codezeile enthält maximal 79 Chars.
Methoden haben keine überflüssigen Parameter: Alle Parameter sind sinnvoll im Code eingebunden.
keine unnötigen Variablen: Alle initialisierten Variablen sind notwendig für die Funktion der zugehörigen Methode.
Benennung: Die Namen für alle Variablen, Methoden und Funktionen sind passend zu Ihrer jeweiligen Aufgabe gewählt.
nur Underscores: Zur Trennung innerhalb einer Benennung werden Underscores verwendet.
Abstand zwischen Methoden: Alle Methoden sind durch einen Absatz vom Kommentar der folgenden Methode abgetrennt.

Kommentare:
keine unnötigen Kommentare: Alle Kommentare weisen nur die für die jeweilige Methode, Variable oder Funktion notwendigen Informationen auf. Es wird keine Beschreibung zu anderen Klassen wiederholt und Kommentare zu obsoleten Funktionalitäten und Abhängigkeiten wurden entfernt.
Abhängigkeiten dokumentiert: Alle Abhängigkeiten zu anderen Klassen sind dokumentiert. Insbesondere ist in den „View" Klassen dokumentiert, mit welcher Front End Datei und den dortigen Parametern sie verknüpft sind.
veränderte externe Parameter: Alle veränderten Parameter die nicht zur Rückgabe gehören werden beschrieben.
keine überlangen Kommentarzeilen: Eine einzelne Kommentarzeile enthält maximal 79 Chars.
kein auskommentierter Code: Es befindet sich kein auskommentierter Code mehr in den bearbeiteten Klassen.
ausreichend Detailliert:
Alle Methoden Parameter, Rückgabewerte und Variablen sind beschrieben, inklusive erwarteten Datentyp und was der jeweilige Parameter repräsentiert.
Für alle Schleifen ist beschrieben, worüber diese Iterieren und was in der Schleife geschieht.

Fehlermeldungen:
Alle Fehler geben eine Fehlermeldung zurück. Die Fehlermeldung ist verständlich und passt zu dem Fehler, durch den sie aufgerufen wird.
Alle üblichen Randfälle wie 0, Null und negative Werte werden behandelt.