HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

ins
INSTITUTE FOR
NETWORKED SOLUTIONS

Ticketing System for Developing Countries

Projektstrukturplan

Hochschule für Technik Rapperswil

Herbssemester 2019
10. Januar 2020

Autor:	Luca Gubler, Alessandro Bonomo
Betreuer:	Prof. Frank Koch
Projektpartner:	INS Institute for Networked Solutions
Arbeitsperiode:	16.09.2019 - 10.01.2020
Arbeitsumfang:	360 Stunden, 12 ECTS pro Student

Inhaltsverzeichnis

1 Inhalt

1.1 Zweck

Dieses Dokument beschreibt die Planung der Bachelorarbeit, in welchem ein Ticketing System für Entwicklungsländer, welches bereits als frühere Bachelorarbeit realisiert wurde, mit zusätzlicher Funktionalität erweitert wird.

1.2 Gültigkeitsbereich

Dieses Dokument ist während der gesamten Laufzeit der Bachelorarbeit gültig. Die Änderungsgeschichte kann in Github nachverfolgt werden.

1.3 Referenzen

Dieses Dokument wurde mit dem Wissen erstellt, welches in den Modulen Software Engineering 1 & 2, Microsoft Technologies sowie in gewissen Grenzen in Cloud Infrastructure und Cloud Solutions vermittelt wird.

2 Projektübersicht

Es soll das bereits bestehende Ticketing System um zusätzliche Funktionalität erweitert werden. Ziel des Ticketing Systems ist es, Menschen an Orten mit limitiertem Zugang zu modernen Hilfsmitteln zu unterstützen. Dabei können zum Beispiel die Menschen mit Experten einer Hilfsorganisation in Kontakt treten, ohne dass diese physisch vor Ort sein müssen. Somit können sehr viele Menschen mit vergleichsweise niedrigem Zeitaufwand angesprochen werden

2.1 Zweck und Ziel

Das Ticketing System wurde bereits in einer früheren Bachelorarbeit umgesetzt. Ziel dieser Bachelorarbeit ist es, das bereits existierende System um zusätzliche Funktionalität zu erweitern.

Folgende Ziele wurden für diese Arbeit gesetzt:

- Der erfolgreiche Abschluss der Bachelorarbeit und das Erhalten der 12 ECTS.
- Die erlernte Theorie der Module ”Software Engineering 1 & 2” in die Praxis umsetzen.
- Den Umgang mit neuen Technologien kennen lernen.

2.2 Lieferumfang

Folgende Dokumente werden am Ende der Bachelorarbeit abgeliefert:

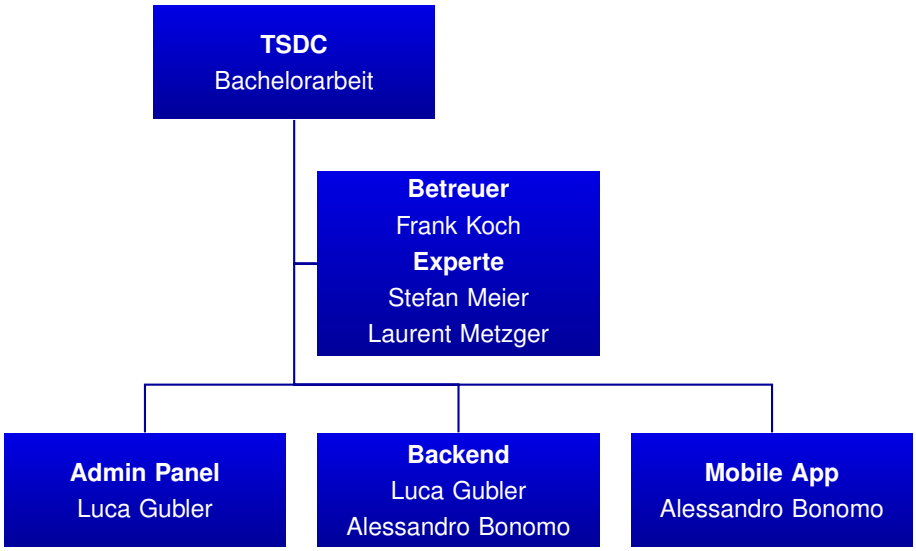
- Abstract
- Aufgabenstellung
- Einverständniserklärung
- Erklärung zur Urheberschaft
- Passwörter
- Persönliche Berichte
- Protokolle
- Source Code

2.3 Annahmen und Einschränkungen

Pro Teammitglied wird mit einem Arbeitsaufwand von 24 Stunden pro Woche gerechnet. Falls jedoch Probleme auftreten oder der Arbeitsaufwand falsch eingeschätzt wurde, kann sich der Arbeitsaufwand auf bis zu 40 Stunden pro Woche erhöhen. Falls es jedoch zu Verzögerungen kommt, wird der Arbeitsumfang in Absprache mit den Betreuern dementsprechend angepasst.

3 Projektorganisation

3.1 Organisationsstruktur



Im Bild wurde die Organisationsstruktur dargestellt. Diese Organigramm wurde jedoch nur zum aufzeigen der groben Zuständigkeiten erstellt. Bei Fragen oder Problemen wird sich das Team gegenseitig unterstützen.

3.2 Externe Schnittstellen

Bei dieser Bachelorarbeit übernimmt Professor Frank Koch die Rolle des Betreuers. Stefan Meier, welcher bereits bei der früheren Bachelorarbeit die Rolle als externer Co-Examinators übernahm, wird auch bei dieser Bachelorarbeit diese Aufgabe übernehmen. Zusätzlich wird Professor Laurent Metzger diese Bachelorarbeit als interner Co-Examinator betreuen.

4 Management Abläufe

4.1 Zeitaufwand

Die Bachelorarbeit begann in der Woche vom 16. September 2019 und dauert insgesamt 17 Wochen. Für das Erreichen der 12 ECTS ist geplant, dass jedes Teammitglied 360 Stunden arbeitet. Daraus resultiert eine durchschnittliche Arbeitszeit von knapp 24 Stunden pro Woche.

Projektstart	**16.09.2019**
Projektdauer	17 Wochen
Arbeitsstunden pro Person	24h pro Woche, Total 360h
Arbeitsstunden Total	720h
Projektende	10.01.2020

Für die Bachelorarbeit stehen total 720 Stunden zur Verfügung. Mit dem definierten Projektumfang wird diese Arbeitszeit voraussichtlich vollständig ausgenutzt. Sollte der Umfang jedoch früher als erwartet abgeschlossen werden können, kann das Projekt um weitere Funktionalitäten erweitert werden.

4.2 Zeitplanung

4.2.1 Phasen / Sprints

Als Projektmanagement Methode wurde SCRUM gewählt. Das gesamte Projekt wird in die vier Phasen Inception, Elaboration, Construction und Transition eingeteilt. Pro Phase gibt es wiederum einzelne Sprints. Die Sprints starten jeden zweiten Donnerstag und beginnen um 13 Uhr. Zudem wurden einzelne Meilensteine definiert, welche auf der untenstehenden Tabelle entnommen werden können.

4.2.2 Meilensteine

Datum	**Meilenstein**
06.10.2019	M1: Projektplan erstellt
13.10.2019	M2: Prototyp installiert
17.10.2019	M3: Zwischenpräsentation
20.10.2019	M4: Domainanalyse und Anforderungsspezifikation erstellt
27.10.2019	M5: End of Elaboration
20.12.2019	M6: End of Construction
10.01.2020	M7: Abgabe der Dokumentation

4.2.3 Iterationsplanung

Zu Beginn jedes Sprints setzt sich das Team zusammen, um den nächsten Sprint zu planen. Dabei wird jeweils besprochen, welche Aufgaben des vergangenen Sprints nicht

vollständig abgeschlossen werden konnten. Die nicht abgeschlossenen Arbeiten werden mit neu definierten Aufgaben in den neuen Sprint übernommen und jeweils zeitlich abgeschätzt und priorisiert. Da sich im Team nur 2 Mitglieder befinden, wird darauf verzichtet, die Arbeitspakete unter den Teammitgliedern untereinander zuzuweisen. Die gesamte Planung und Verwaltung der Aufgaben wird in Jira erledigt.

4.3 Besprechungen

Die Teammitglieder arbeiten an mindestens zwei Tagen pro Woche zusammen im Bachelorarbeits Zimmer. So können Fragen schnell geklärt werden und es kann sich gegenseitig geholfen werden. Im Normalfall findet jeden Donnerstag ein Meeting mit dem Betreuer statt, in dem der aktuelle Stand vorgestellt, Probleme besprochen und das weitere Vorgehen besprochen wird.

4.4 Versionskontrolle

Um den Source Code der gesamten Applikation zu verwalten, kommt git zum Einsatz. So kann sichergestellt werden, dass die Qualität des Codes zu jeder Zeit gewährleistet ist. Damit die Teammitglieder immer auf dem neusten Stand sind, wurde beschlossen, dass Merge-Requests unumgänglich sind. Mit dieser Massnahme wird erzwungen, dass jede geschriebene Zeile Code, die in den Master-Branch gemerged werden soll, von mindestens einer weiteren Person zur Kenntniss genommen und überdacht wird.

4.5 Projektverwaltung

Als Projektverwaltungstool wird Jira verwendet. Man hat sich für dieses Tool entschieden, da es die gewünschte Funktionalität mit sich bringt und trotzdem sehr schlank und übersichtlich ist. Um den Stand der einzelnen Arbeitspakete möglichst genau darzustellen, wurde ein Workflow definiert, welcher jedes Arbeitspaket durchlaufen muss. Wie im Bild oben ersichtlich ist, muss jedes Arbeitspaket folgende Status durchlaufen: Open, In Progress, Review und Done. Ein Arbeitspaket kann jeweils nur ein Status vor und/oder zurück verschoben werden. Dies soll verhindern, dass ein Paket direkt vom Status Open in den Status Done verschoben wird. Sobald ein Arbeitspaket den Status Done erreicht hat, kann dies nicht mehr rückgängig gemacht werden.

4.6 Code Styleguide

Bei der Vorgänger Arbeit wurde StyleCop eingesetzt. Dabei handelt es sich um ein Tool, welches bestimmte Style und Konsistenz Regeln erzwingt. Dieses Tool kann direkt in Visual Studio verwendet werden, ohne das Änderungen am Code vorgenommen werden müssen.

4.7 Performance Tests

Um sicherzustellen, dass die Performance der Applikation durch Anpassungen nicht verschlechtert wird, wird beim Beginn der Arbeit ein Performance Test durchgeführt. Diese Performance Tests werden regelmässig durchgeführt, so dass allfällige Verschlechterungen schnell bemerkt und behoben werden können. Die Vorgänger haben bereits eine Applikation geschrieben, welche die Performance testet. Diese Applikation wird auch bei dieser Arbeit wieder verwendet.

4.8 Continuous Integration / Continuous Development

Beim CI/CD Ansatz wird der Code regelmässig in den Master Branch gemerged. Bei diesem Schritt werden die Änderungen automatisch validiert und getestet. Somit kann eine ”integration hell” vermieden werden, da man nie zu weit vom Master Branch abweicht.
Um diesen Ansatz umsetzen zu können, kommt Jenkins zum Einsatz.

5 Risikomanagement

5.1 Risiken

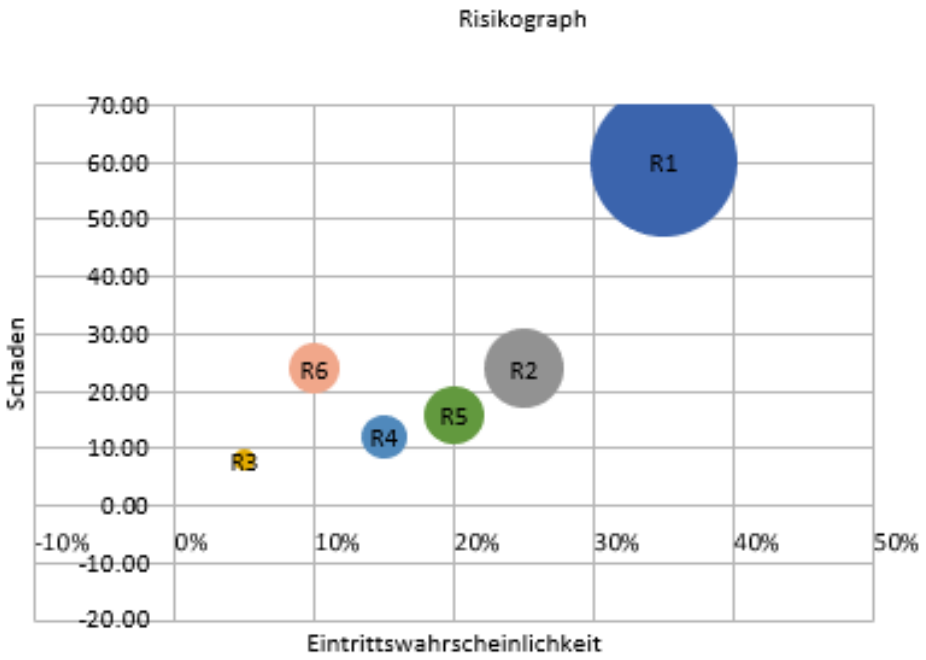


Abbildung 1: Ursprüngliche Risikoanalyse

Die detailierte Risikoanalyse kann dem Dokument ”TechnischeRisiken.xlsx” entnommen werden.
Wie man der Grafik entnehmen kann, ist das Risiko R1 - ”Probleme beim Aufsetzen des Prototypen” das grösste Risiko. Zum einen ist der Prototyp sehr umfangreich und hat mehrere Komponenten wie das Admin Panel oder die Serverseitigen Applikationen, respektive IIS. Bei diesen Komponenten könnte zum einen fehlendes Know-How zu einem Problem werden. Des weiteren ist der Prototyp mittlerweile 4 Jahre alt. Es könnte durchaus sein, dass API Abfragen nicht mehr funktionieren oder Komponenten veraltet sind. Diese müssten zuerst korrigiert werden, bevor der Prototyp überhaupt läuft.

5.2 Umgang mit Risiken

Risiken lassen sich in einem grösseren Projekt leider nicht vermeiden. Allfällige Risiken werden jeweils zu Beginn eines Sprints im Team angesprochen. Falls es als sinnvoll erachtet wird, werden auch gleich Massnahmen ergriffen um das Risiko einzudämmen.
Für die wahrscheinlichen Risiken wurden Massnahmen definiert, um den auftretetenden Schaden unter Kontrolle zu bringen. Sollten während des Projektes neue Risiken

hervortreten, werden diese in die Risikoanalyse aufgenommen und bewertet.

5.3 Aktualisierte Risikoanalyse

Nach der Aufnahme der Risikon wurde versucht, die am höchsten gewichteten Risiken zu minimieren. Wie man der Grafik entnehmen kann, konnten alle Risiken ausser dem Risiko R1, verringert werden.

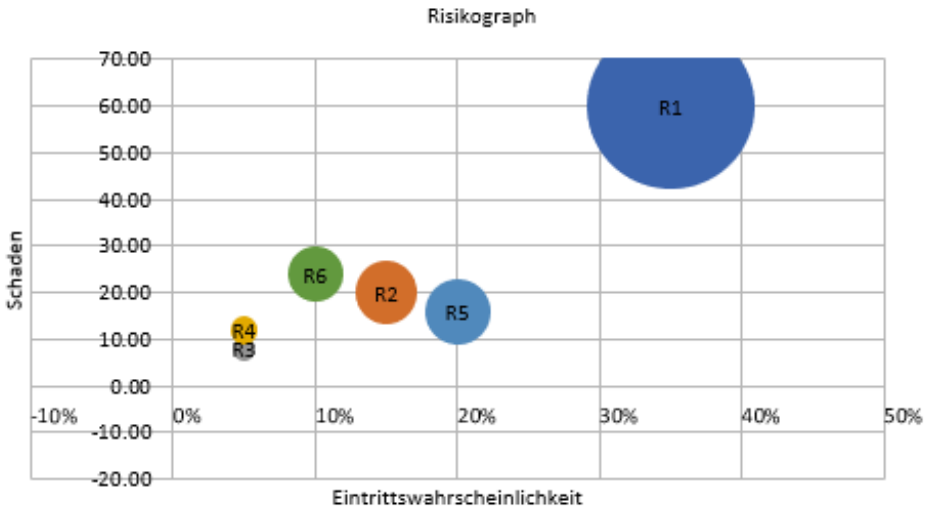


Abbildung 2: Aktualisierte Risikoanalyse

Details können dem Dokument ”TechnischeRisiken.xlsx” entnommen werden.

6 Infrastruktur

Die Infrastruktur setzt sich aus der Hard- und Software zusammen, welche für die Durchführung der Bachelorarbeit verwendet wird. So kann die Zusammenarbeit im Team erleichtert werden. Zudem kann die Code Qualität besser überprüft und verbessert werden.
Viele Tools sind bereits aus Modulen oder vergangenen Projektarbeiten bekannt, weshalb man sich schnell auf bestimmte Tools einigen konnte. Trotzdem gibt es noch einige Unklarheiten, da einige Tools von anderen abhängig sind. Erst im späteren Projektverlauf wird sich zeigen, ob diese Tools verwendet werden. Es kann aber auch sein, dass einige Tools, welche erwähnt werden, gar nicht im Projekt eingesetzt werden.

6.1 Dokumentation

Für das Erstellen der Dokumentation wird LaTeXverwendet. Dieses Tool erforderte zwar etwas Zeit, um sich damit vertraut zu machen. Das Team ist jedoch überzeugt, dass man mit LaTeXweniger Probleme hat, wenn man bereits eine gute Vorlage hat.
Die einzelnen Dokumentationen werden jeweils lokal und auf Github gespeichert. Da die Dokumentation auf Github gespeichert ist, kann auch zu jeder Zeit nachverfolgt werden, wann etwas geändert wurde.

6.2 Arbeitspakete Verwaltung

Für das Verwalten der Arbeitspakete entschied sich das Team, Jira zu verwenden. Für das Engineering Projekt und die Studienarbeit wurde ebenfalls Jira verwendet. So ist es mit Jira sehr einfach, Workflows zu definieren, neue Arbeitspakete zu erfassen und die Arbeitszeit auszuwerten. Zudem ist das Reporting mit Jira sehr gut und man sehr schnell Burndown Charts erstellen.

6.3 Konstruktion

Als Versionsverwaltung wird GIT verwendet, da es zum de facto Standart in der Software Entwicklung gehört. Der grosse Vorteil ist, dass man nicht ständig mit dem Server verbunden sein muss. Zudem konnte jedes Mitglied bereits Erfahrungen mit Github sammeln, was den Einstieg erleichtert.
Als Entwicklungsumgebung wird Visual Studio von Microsoft verwendet.

6.4 Testing

Sämtlicher Code muss getestet werden. Da die Team Mitglieder jedoch noch nicht viel Erfahrung in der C# / ASP.NET Entwicklung haben, muss dieser Punkt jedoch noch genauer evaluiert werden.

6.5 Infrastruktur

Da sich das Projekt aus mehreren Applikationen zusammen setzt, werden diese folglich auch an unterschiedlichen Orten eingesetzt. Das Admin Panel wird auf einem Windows Client ausgeführt und das Mobile App auf einem Android Handy. Die Server Applikation wird auf einem Windows Server deployed und als Datenbank kommt MS SQL Server zum Einsatz.
Das Ziel dieser Arbeit ist es, die bereits existierende Applikation um neue Funktionalitäten zu erweiteren. Schwerwiegende Änderungen an der Software Architektur sollen folglich vermieden werden.

7 Qualitäts- und Sicherheitsmassnahmen

Um sicherzustellen, dass die Qualität der Arbeit immer gewährleistet ist, wurden folgende Massnahmen getroffen:

Massnahmen	**Zeitram**	**Ziel**
Meeting	Wöchentlich	Hier wird der aktuelle Stand des Projektes besprochen und neue Entscheidungen werden gefällt
Besprechung der Meilensteine	7 Mal während der gesamten Projektdauer	Dient zur Kontrolle, ob das Team auf dem richtigen Weg ist
Code Reviews	Fortlaufend während der gesamten Projektdauer	Dient zur Erhöhung der Qualität des Codes
Verwendung aktueller Code Versionen	Fortlaufend während der gesamten Projektdauer	Bei den aktuellen Versionen wurden Bugs behoben, welche nicht mehr auftreten sollten

Abbildungsverzeichnis