April 2016

IPA-Bericht

Samuel Keller

**Inhaltsverzeichnis**

[1 Obligatorischer Teil 3](#_Toc447699416)

[1.1 Ausgangslage 3](#_Toc447699417)

[1.2 Detaillierte Aufgabenstellung 3](#_Toc447699418)

[1.2.1 Login 3](#_Toc447699419)

[1.2.2 Benutzerverwaltung 3](#_Toc447699420)

[1.2.3 Die Verwaltung von Tickets 4](#_Toc447699421)

[1.2.4 Ticket Workflow 4](#_Toc447699422)

[1.2.5 Dokumentation 4](#_Toc447699423)

[1.2.6 Design 5](#_Toc447699424)

[1.2.7 Testing 5](#_Toc447699425)

[1.2.8 Versionsverwaltung 5](#_Toc447699426)

[1.2.9 Abgrenzungen 5](#_Toc447699427)

[1.2.10 Zeitplan 5](#_Toc447699428)

[1.2.10.1 Initialisierung (Montag, 04.04.2016) 5](#_Toc447699429)

[1.2.10.2 Analyse (Dienstag, 05.04.2016) 5](#_Toc447699430)

[1.2.10.3 Entwurf (Mittwoch, 06.04.2016) 5](#_Toc447699431)

[1.2.10.4 Implementierung (Mittwoch, 06.04.2016 - Dienstag, 12.04.2016) 6](#_Toc447699432)

[1.2.10.5 Testing (Montag, 11.04.2016 - Dienstag, 13.04.2016) 6](#_Toc447699433)

[1.2.10.6 Abschlussphase (Montag, 13.04.2016 - Dienstag, 14.04.2016) 6](#_Toc447699434)

[1.2.10.7 Buffer (Freitag, 15.04.2016) 6](#_Toc447699435)

[1.2.11 Mittel und Methoden 6](#_Toc447699436)

[1.2.12 Vorkenntnisse 6](#_Toc447699437)

[1.2.13 Vorarbeiten 6](#_Toc447699438)

[1.2.14 Neue Lerninhalte 6](#_Toc447699439)

[1.3 Projektorganisation 7](#_Toc447699440)

[1.3.1 Projektmanagementmethode 7](#_Toc447699441)

[1.3.1.1 Initialisierungsphase 7](#_Toc447699442)

[1.3.1.2 Analysephase 7](#_Toc447699443)

[1.3.1.3 Entwurfsphase 7](#_Toc447699444)

[1.3.1.4 Implementierungsphase 7](#_Toc447699445)

[1.3.1.5 Testingphase 7](#_Toc447699446)

[1.3.1.6 Abschlussphase 7](#_Toc447699447)

[1.3.2 Zeitplan 8](#_Toc447699448)

[1.4 Arbeitsjournal 9](#_Toc447699449)

[1.4.1 Montag, 4. April 2016 9](#_Toc447699450)

[1.4.2 Dienstag, 5. April 2016 10](#_Toc447699451)

[1.4.3 Mittwoch, 6. April 2016 11](#_Toc447699452)

[2 Projektdokumentation 12](#_Toc447699453)

[2.1 Versionierung 12](#_Toc447699454)

[2.2 Backup 12](#_Toc447699455)

[2.3 Commits 12](#_Toc447699456)

[3 Quellenverzeichnis 16](#_Toc447699457)

[3.1 Webseiten **Fehler! Textmarke nicht definiert.**](#_Toc447699458)

[3.1.1 Wikipedia **Fehler! Textmarke nicht definiert.**](#_Toc447699459)

[3.1.1.1 Versionierung 16](#_Toc447699460)

[3.1.1.2 Risikoanalyse 16](#_Toc447699461)

Formate:

* Laufzeit (Liberierungs- und Rückzahlungsdatum)
  + Allfällige periodische Verzinsung

# Obligatorischer Teil

## Ausgangslage

Die Abteilung Web Service ist für verschiedenen Webapplikationen der Schweizerischen Bankiervereinigung verantwortlich. Sie ist zuständig für die Weiterentwicklung, die Aufrechterhaltung und teilweise Bewirtschaftung der Webapplikationen. Die Aufträge für die Bewirtschaftung der Webseiten werden via E-Mail oder in wenigen Fällen via Telefon an die Webabteilung mitgeteilt. Dies hat zur Folge, dass vielmals der Überblick über die Aufträge nicht mehr gewährleistet ist, und somit die Fehleranfälligkeit erhöht. Die Koordination und Konsolidierung der Aufträge durch ein zentrales webbasierter Ticketingsystem würde für die Benutzer sowie für die Webabteilung eine grosse Entlastung bringen.

## Detaillierte Aufgabenstellung

Im Rahmen der Prüfungsarbeit IPA soll ein webbasiertes Ticketingsystem für die Webabteilung entwickelt werden. Die Arbeit ist auf die Grundfunktionen der Applikation und die Usability ausgerichtet.

Das Ticketingsystem besteht aus drei Komponenten:

### Login

Die Implementierung eines einfachen und benutzerfreundlichen Log-ins für den Benutzer.

### Benutzerverwaltung

Es existieren zwei Benutzergruppen: Benutzer und Administratoren.

Benutzer können neue Tickets erstellen und ihre persönlichen Informationen, bestehend aus E-Mail, Telefonnummer und Passwort, bearbeiten. Administratoren können zusätzlich zu den Benutzern folgende Aktivitäten vornehmen: Jedem Ticket einen Administrator zuweisen, alle Tickets bearbeiten, den Status des Tickets verändern, neue Nutzer erstellen und deren persönlichen Informationen bearbeiten.

Die persönlichen Informationen sind über ein Formular erreichbar.

### Die Verwaltung von Tickets

Nach dem Einloggen werden die Benutzer auf eine Übersichtsseite weitergeleitet. Auf dieser Seite sind alle aktiven Tickets publiziert. Folgende Kriterien eines Tickets sind ersichtlich: das Erfassungsdatum, den Status des Tickets, eine allfällige Deadline, den Tickettitel, die Person, welche das Ticket erstellt hat sowie die Person, die das Ticket bearbeitet. Die Ticketübersicht kann nach dem Status des Tickets, der Kategorie, dem Ersteller und dem Administrator gefiltert werden.

Tickets können in einer Detailansicht geöffnet werden. In dieser Ansicht ist die genaue Beschreibung des Problems ersichtlich. Die Administratoren können in dieser Ansicht die Ticketinformationen verändern.

### Ticket Workflow

Jeder Benutzer hat die Berechtigung, ein neues Ticket zu erfassen. Das Ticket kann mittels eines Formulars erfasst werden. Im Formular können die Ticketinformationen (bestehend aus dem Tickettitel, der Kategorie, der Ticketbeschreibung und einer allfälligen Deadline) angegeben werden. Zudem kann er eine Datei z.B. ein Printscreen hochladen und anschliessend das Ticket zur Bearbeitung abschicken. Nach dem Abschicken des Tickets erhalten die Administratoren eine E-Mail Benachrichtigung. Ein Administrator kann dem Ticket einen Administrator zuweisen, welcher sich um das Ticket kümmert. Nach der Zuweisung erhält der Erfasser des Tickets eine E-Mail mit der Meldung, dem Ticket sei ein Administrator zugewiesen worden. Der Erfasser des Tickets erhält ebenfalls eine E-Mail, wenn der Status des Tickets verändert wird. Mit dem Status „Archiv“ wird das Ticket aus der Standardansicht, die alle offenen Tickets anzeigt, ausgeblendet.

### Dokumentation

Die IPA-Dokumentation enthält die Ausgangslage des Projektes, den Ist-Zustand sowie den Sollzustand, die Analyse des Projektes, das Vorgehen in der Entwicklung sowie das Testing der Applikation. Zudem geht die Dokumentation darauf ein, mit welchen zusätzlichen Optionen die Applikation nach der Arbeit erweitert werden könnte.

Es wird ein Benutzerhandbuch erstellt, das dem Benutzer als Unterstützung dient. Dieses kann der Benutzer auf der Ticketing Website herunterladen.

### Design

Die Webapplikation ist übersichtlich und benutzerfreundlich gestaltet. Sie ist programmatisch aufgebaut und hat ein schlichtes Design. Jedes Formular wird validiert.

### Testing

Getestet werden die Grundfunktionen der Applikation mittels mehreren Testfällen für Tickets, Benutzer und Administratoren. Ebenfalls werden alle Formulare mit Grenzwerten überprüft. Ein Testfall besteht aus einer Nummer mit einem Titel, einer Aktion und aus dem erwarteten Resultat. Die Testfälle werden während der Analysephase festgelegt.

Die Testfälle werden übersichtlich in einer Tabelle dargestellt. In dieser wird der Testfall mit Nummer und Titel angegeben. Zusätzlich sind das Datum des Tests, die Testperson sowie das Resultat festgehalten. Das Resultat hat die Optionen: „Wie erwartet“, „Nicht implementiert“ sowie „Funktioniert nicht“. „Wie erwartet“ wird nur vergeben, wenn der Testfall in den drei Browser Mozilla Firefox, Google Chrome und Microsoft Internet Explorer erfolgreich war. Die Testfälle werden ebenfalls separat in Testprotokollen festgehalten.

### Versionsverwaltung

Die Versionsverwaltung wird mit einem git GUI-Client erstellt.

### Abgrenzungen

* Installation und Wartung von Hard- und Software ist nicht Teil der Prüfung.
* Verwendung des Frameworks Bootstrap.
* Die Applikation ist so entwickelt, dass es auf den gängigen Browsern auf einem Desktop funktioniert, jedoch nicht auf Tablets und Smartphones.

### Zeitplan

Der Zeitplan und die beschriebenen Arbeitsschritte können sich noch ändern. An der Dokumentation wird jeden Tag gearbeitet.

#### Initialisierung (Montag, 04.04.2016)

Grundgerüst der Dokumentation, erste Texte erstellen, GIT einrichten

#### Analyse (Dienstag, 05.04.2016)

Analyse der Webseite, Erstellung der Testfälle

#### Entwurf (Mittwoch, 06.04.2016)

Erstellung des Webseitenlayouts.

#### Implementierung (Mittwoch, 06.04.2016 - Dienstag, 12.04.2016)

Implementierung der drei Komponenten Log-in, Benutzerverwaltung und Ticketverwaltung.

#### Testing (Montag, 11.04.2016 - Dienstag, 13.04.2016)

Testen der Applikation und Erstellung des Testberichtes

#### Abschlussphase (Montag, 13.04.2016 - Dienstag, 14.04.2016)

Abschluss der IPA-Dokumentation und des Benutzerhandbuches.

#### Buffer (Freitag, 15.04.2016)

Ein Buffertag für nötige Ergänzungen in der Dokumentation und der Programmierung.

### Mittel und Methoden

* PHP
* CSS
* HTML
* MySQL
* Git
* Eclipse

### Vorkenntnisse

* HTML & CSS: 2 Jahre
* PHP: 2 Monate
* MySQL: Basiswissen

### Vorarbeiten

* Es werden im Voraus die Dokumentvorlagen für die Dokumentation der Prüfung vorbereitet, damit ab dem ersten Tag bereits dokumentiert werden kann.
* Erstellung einer statischen Website mit Referenzlayout.
* Die Datenbank wird im Voraus geplant und erstellt.

### Neue Lerninhalte

* PHP Log-in mit Usermanagement
* PHP in Verbindung mit MySQL
* PHP E-Mail Versand

## Projektorganisation

### Projektmanagementmethode

Der Auftrag ist mit knapp zwei Wochen als kleines Projekt anzusehen. Das Wasserfall-Modell ist ideal für diese Art von Projekt. Die klare Einteilung in Phasen und komplexe Strukturen macht das Modell übersichtlich. Das IPERKA-Modell wäre eine Alternative gewesen. Dieses wurde in der Schule auch als Alternative beworben. Mangels Kenntnisse und Sicherheit in dieser Methode wurde dann schliesslich das Wasserfall-Modell gewählt.

Das Wasserfall-Modell wurde in die folgenden Phasen unterteilt:

#### Initialisierungsphase

In dieser Phase werden die Vorbereitungen für das anschliessende Projekt getroffen. Beispielsweise wird die Grundstruktur der Dokumentation und ein Zeitplan erstellt, sowie eine Back-up-Lösung angelegt

#### Analysephase

Die Ausgangssituation wird genauer analysiert und mit Text und Grafiken beschrieben. Mit einem Soll / Ist Vergleich und einer Risikoanalyse werden mögliche Gefahren identifiziert, um diese in den weiteren Phasen umgehen zu können und die Planung und Umsetzung zu erleichtern.

#### Entwurfsphase

In der Entwurfsphase werden vorgehende Diagramme und Konzepte erweitert. Ebenso wird ein GUI-Entwurf erstellt. Aufgrund diesem wird dann die Implementation geplant und das Benutzerhandbuch geschrieben.

#### Implementierungsphase

In der Implementierungsphase wird das Projekt, welches in den vorigen Phasen geplant wurde, umgesetzt.

#### Testingphase

In dieser Phase wird die Qualitätssicherung sichergestellt. Aufgrund eines Testkonzepts wird ein Test-Szenario erstellt. Auf diesem basieren Testfälle, mit denen man die Software auf ihre Funktionen testet und auswertet. Diese werden in Test-Protokollen festgehalten.

#### Abschlussphase

In der Abschlussphase werden die letzten Arbeiten an dem Projekt beendet. Die Dokumentation wird auf Orthographie und Grammatik geprüft, und die letzten Dokumente werden geschrieben und der Kriterienkatalog überprüft. Am Schluss wird ein ausführlicher Rückblick auf das Projekt verfasst.

### Zeitplan

## Arbeitsjournal

### Montag, 4. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsplatz einrichten * Lokales Repository erstellen * Repository auf GitHub erstellen * Dokumentstruktur erstellen * Dokumentation des obligatorischen Teil anfangen * Versionierung und Back-up-Lösung des Repositorys definieren und dokumentieren * Zeitplan erstellen |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Für das USB-Stick-Back-up habe ich in der Mittagspause einen neuen USB-Stick gekauft. |
| Geplante Arbeiten nach Arbeitsschluss | Texte zu Hause korrigieren lassen. |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | Ich habe Detlef Nünninghoff nach einer Vorlage für die Dokumentation mit dem SwissBanking Design gefragt und gezeigt bekommen, wo ich dieses finde. |
| Soll / Ist Zeitvergleich | Alles läuft nach Zeitplan. Jedoch habe ich gewisse Schritte in einer anderen Reihenfolge bearbeitet als im Zeitplan definiert. Das liegt nicht daran, dass es so unbedingt mehr Sinn ergibt, sondern daran, dass ich gerne zwischen den einzelnen Dokumenten und Texten wechsle. So versteife ich mich auch nicht auf ein Dokument und/oder Textteil und kann laufend Dokumente ergänzen. |
| Sonstiges | Dokumentationstexte werden separat geschrieben und später zusammengetragen. |
| Arbeitszeit | 08:30 Uhr – 10:15 Uhr  Kurze Pause  10:20 Uhr – 12:30 Uhr  Mittagspause  13:30 Uhr – 17:00 Uhr  Kurze Pause  17:10 Uhr – 18:00 Uhr |
| Visum |  |

### Dienstag, 5. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Versionierungstext in die Dokumentation übernehmen * Arbeitsjournal (Tag 1) in die Dokumentation übernehmen * Risikoanalyse dokumentieren * „Wieso Bootstrap“ dokumentieren * Soll / Ist-Use-Case erstellen * Implementierungsvorgang (Planung) dokumentieren |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | Kurzes Brainstorming zur Nutzerhandbuch-Struktur |
| Ungeplante Arbeiten | keine |
| Geplante Arbeiten nach Arbeitsschluss | Texte zu Hause korrigieren lassen. |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | Nach der Erstellung der Soll / Ist-Grafiken habe ich diese meinem Vater und Alexandra Arni (FV) gezeigt. Die Modellstruktur blieb so wie geplant, die Darstellung änderte sich. |
| Soll / Ist Zeitvergleich | Ich bin im Moment im Zeitplan. Das Benutzerhandbuch habe ich bereits mit einer Inhalts-Struktur angefangen und über das Testkonzept bin ich mir bereits am Gedanken machen. |
| Sonstiges | Ich habe die Ausgangslage mit dem Soll / Ist-Vergleich verbunden, da ein Text zur Ausgangslage schon vorhanden ist. Das Test-Konzept wurde zum Testing verschoben. |
| Arbeitszeit | 08:30 Uhr – 10:00 Uhr  Kurze Pause  10:10 Uhr – 12:30 Uhr  Mittagspause  13:30 Uhr – 15:00 Uhr  Kurze Pause  15:10 Uhr – 18:00 Uhr |
| Visum |  |

### Mittwoch, 6. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 2) in die Dokumentation übernehmen * Use-Case dokumentieren * Bootstraptext in die Dokumentation übernehmen * Risikoanalyse in die Dokumentation übernehmen * Zwei ERD’s erstellen * Webseitenentwurf erstellen, auf dem bei der Implementation aufgebaut werden kann * Benutzerhandbuch mit Text erweitern |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Ich habe eine zusätzliche Use-Case Grafik erstellt, die den Kommunikationsweg besser zeigt. |
| Geplante Arbeiten nach Arbeitsschluss | Texte zu Hause korrigieren lassen. |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | keine |
| Soll / Ist Zeitvergleich | Laut Zeitplan ist alles in Ordnung. |
| Sonstiges | Da ich die ERD’s mit dem Tool „MySQL Workbench“ erstellt habe, habe ich etwas länger für die Grafiken benötigt, als wenn ich diese in Visio erstellt hätte. Der Webseitenentwurf gefällt mir sehr gut und ich freue mich auf die Implementierung. Das Benutzerhandbuch ist auch auf einem guten Weg. |
| Arbeitszeit | 08:30 Uhr – 10:15 Uhr  Kurze Pause  10:20 Uhr – 12:40 Uhr  Mittagspause  13:40 Uhr – 18:00 Uhr |
| Visum |  |

# Projektdokumentation

## Versionierung

Für die Versionierung des Git-Repositorys wurde ein klassisches System mit Hauptversionsnummer, Nebenversionsnummer, Revisionsnummer und Buildnummer mit ergänzender Information zum Entwicklungsstadium der Software gewählt. Die Versionsnummern beziehen sich auf die Software und nicht auf die Dokumentation. Diese enthält eine eigene Versionsnummer unabhängig von dem des Repositorys. Das Repository startet nur mit Revisions- und Buildnummer, da am Anfang nur die Dokumentation existiert. Mit der Implementation und Umsetzung der Software wird diese hinzugefügt. Während dem Entwickeln erhält die Versionsnummer den Zusatz „b“ am Schluss, um auf den Beta-Status hinzuweisen. Die finale Version wird mit einem „rc“ für den Veröffentlichungskandidat gekennzeichnet. Für das Arbeiten mit dem Repository wird die Software „SourceTree“ verwendet.

Die Versionsnummer des Repositorys setzt sich folgendermassen zusammen:

0.0.0-0000x

│ │ │ │ └─── Stadium

│ │ │ └────── Buildnummer

│ │ └───────── Revisionsnummer

│ └─────────── Nebenversionsnummer

└───────────── Hauptversionsnummer

Abbildung: Quelle Wikipedia

## Backup

Durch die Versionierung mit Git wird das Zurückspringen in der Entwicklung oder das Wiederherstellen einer vorherigen Version vereinfacht. Um die Sicherheit noch zu verstärken ist das Repository auf GitHub erreichbar. Das öffentliche Repository kann jederzeit wieder lokal hergestellt werden. Zusätzlich sind die Releases in GitHub übersichtlich dargestellt. Ebenso werden alle Dateien zur Sicherheit auf einem USB-Stick gespeichert. Das Backup wird mindestens einmal am Ende des Tages erstellt.

## Commits

Die Commits haben jeweils einen Titel. Diesem folgen die Änderungen im Repository auf separaten Linien. Für jede Meldung wird eine separate Linie beansprucht. Diese wird nicht mit einem Satzzeichen beendet. Die Meldungen sind im Imperativ geschrieben und zeigen die grössten Änderungen auf. Falls Fehler entfernt wurden, werden diese unter den Änderungen aufgeführt.

## Soll/Ist vergleich

## Risikoanalyse

### Datenverlust

Datenverlust ist ein Grosses problem während der Arbeit. Der verlust von Daten kann hauptsächlich durch Software Probleme, wie beispielsweise Absturz des Texteditors, oder durch unachtsamkeit entstehen. Ohne Entsprechende Schutzmnassnahmen kann der Verlust von Daten fatale folgen haben unt unter umständen sogar das projekt zum scheitern verurteilen.

#### Wahrscheinlichkeit

Die warscheinlichkeit eines Datenverlustes ist sehr gering. Falls jedoch einer passiert

#### Folgen

Die Folgen eines Datenverlustes sind bedingt durch den Stand des verlusttes und des Standes des Backups. Bei eiem grösseren verlust kann dies zum scheitern des Projektes führen. Falls die Auswirkungen nicht so fatal sind, wirk es sich vorallem auf die Zeitplanung aus, da man zuerst den Stand von vorhin wieder erreichen muss.

#### Gegenmassnahmen

Zur gegenmasnahme existiert ein Backup auf einem USB-Stick, ein lokales Git-backup sowie ein äffentliches auf GitHub . Alle Backups sind immer auf dem selben Stand. Es kann jederzeit auf ein altes zurückgegriffen werden. Falls der erstfall von einem haarddiskfehler passiert, kann an einem anderen Arbeitsplatz weiter gearbeitet werden.

#### Schadensminimierung

Um den Schaden zu minimieren wird mindestens einmal am Tag ein komplettes Backup gemacht. Bei grösseren Änderungen an Dokumentation oder Programmcode wird ebenfalls ein neues Backup erstellt, damit der Datenverlust möglichst gering gehalten werden kann.

### Zeitprobleme

Bei einer Arbeit von knapp 10 Arbeitstagen und eriner knapp bemessenenZeit sind Zeitprobleme eine der grössten Probleme, die während der Arbeit auftreten können. Falls unvorhergesehene Probleme oder zusätzliche Arbeiten aufkommen, führt das zu einer veränderung des Zeitplanes und somit zur Zeitverzögerung des Projekts.

#### Wahrscheinlichkeit

Die Warscheinlichkeit für Zeitprobleme sind sher hoch. Es goibt viele Fehler die bei der zeitplanung vor allem als leie gemacht werden können. So können für verschiedene Punkte zu viel Zeit gebraucht werden oder allgemein die bemessene Zeit falsch bemessen worden sein.

#### Folgen

Zeittprobleme können von nicht korrigieren eines Textes bis zur nicht fertigstellung des Projektes führen. Das verpassen des Abgebetermins ist dabei auch ein Problem das auftreten könnte. Jedoch würde ich lieber ein nicht fertiges Produkt abgeben als diesen zu verpassen.

#### Gegenmassnahmen

Um genügend zeit für alles zu haben, wurde ein Buffertag eingeplant. An diesem können letzte anstehende Arbeiten erledigt werden, falls diese existieren.

#### Schadensminimierung

Falls schon vor dem Buffertag bemerkt wird, das bestimmte arbeiten mehr zeit benötigen, kann noch etwas dagegen getan werden. Wie zum beispiel den Fokus auf etwas anderes setzen oder im Notfall mehr gearbeitet werden. Falls erst am Buffertag Probleme auftreten, kann das fatale folgen haben, da der Abgabetermin am selben tgag ist. Es könnte sein, dass diese probleme dann ignoriert werden müssten, um diesen zu verpassen und automatischen Notenabzug zu erhalten.

### Internetbrowser-Probleme

Die geplante Applikation wird für die meistgebrauchten Browser entwickelt. Jeder dieser Browser hat seine eigenheiten und kann zum Beispiel in der Darstellung des geplanten GUI variieren. Diese manuell anzupassen kann viel Zeit kosten.

#### Wahrscheinlichkeit

Browserprobleme werden vor allem bbei der Darstellung des GUI auftreten können. Die Warscheinlichkeit hält sich meiner erfahrungen etwa bei 60%.

#### Folgen

Bei Browserproblemen wird es zu Zeitverzögerungen kommen, welche wiederum zu fatalen folgen führen können.

#### Gegenmassnahmen

Um den Problem entgegenzukommen wird das Frame Work Bootstrap benutzt. Dieses hat hat bereits viele probleme der unterschiedlichen <browsert gelöst. Somit kann der Fokus auf etwas anderes gelegt werden und das Styling der Webseite muss sich nicht viel gekümmertt werden

#### Schadensminimierung

Falls dennoch Probleme auftreten muss der Fehler ausgewertet werden. Falls der Feher nur optisch ist und nicht die Funtionalität der Applikation winschränkt, kann dieser fehler in der Bufferzeit oder einer nchsten Version der Applikation bearbeitet werden. Falls das Problem mdie Funktionalität einschränk, muss sofort gehandelt werden und das Problem manuell mit Hilfe von >Dritter und Internet gelöst werden.

## Wieso Bootstrap

### Was ist Bootstrag

Bootstrap ist das meist genutzte front-end Framwork, welches für Responive mobile-first Webseiten genutzt wird. Viele Webseiten heutzutage benutzen dieses Framework. Die Meinungen von personen zum nuzen dieses frameworks unterscheiden sich von person zu person sehr stark. Viele Persoenen lieben es, währdem es andere hassen. Die Benutzung ist also bei Entwicklern umstritten.

Um die Entscheidung Bootstrap zu enutzen besser aufzuzeigen, sind hhier die grösten Vor und nachteile von bootstrap:

### Vorteile

* Erleichtert und beschleunigt die Entwicklung
* Erlaubt schnelles Prototyping
* Funktioniert in allen modernen Browsern
* Unterstützt JavaScript und jQuery
* Arbeitet mit eigener Gitter-Struktur
* Mobile First
* Löst viele CSS Probleme
* Es wird nicht viel eigenes CSS benötigt
* Open Source

### Nachteile

* Durch das vordefinierte CSS des Frameworks, sehen viele Webseiten gleich au.
* Bootstrap legt den schwerpunkt nicht auf das Design
* Es ist schwer sein eigenes Design daraus zu erstellen
* Die Bootstrap Logik ist überladen
* Bootstrap Updates sind mühsam und oft wird die Version mit welcher die bseite erstellt wurde die finale Version für die Seite.

## Wieso Bootstrap bewnutz wird

Bootstrap bietet mit dem bereits vorhandenen CSS und dem grid System eine vorhandene Strucktur an welcher man nicht viel arbeiten muss. Da Bootstrap sehr gut Dokumentiert ist, ist es ein leichtes, die Webseite so auszusehen zu lassen wie man möchte, ohne viel eigenes CSS zu schreiben. Das spart bei der entwicklung sehr viel arbeit und vorallem Zeit. Das Bootstrap design ist so weit verbreitet, das das aussehen nicht negativ auftritt. Bootsrap erlaubt schnelle entscheidungen.

Design

Testing

# Abschlussbericht

## Rückblick

## Lessons learned & Was ist gut gelaufen

## Was schief gelaufen ist

## Mit welchen Funktionen kann das Projekt sinnvoll erweitert werden?

## Resultat

## Schlusswort

# Quellenverzeichnis

## Versionierung

<https://de.wikipedia.org/wiki/Versionsnummer>

* Darstellung der Versionierung übernommen und erweitert.
* Als Referenz für den Text benutzt

## Risikoanalyse

https://de.wikipedia.org/wiki/Risikoanalyse

* Als Referenz für den Text benutzt

## Bootstrap

<http://stackoverflow.com/questions/14227000/pros-cons-of-switching-to-twitter-bootstrap>

<http://www.hyperarts.com/blog/twitter-bootstrap-sucks-and-its-awesome/>

<http://www.htmlcenter.com/blog/the-bootstrap-framework-controversy-should-you-use-it-or-not/>

* Vor- und Nachteile übersetzt und übernommen