|  |
| --- |
|  |
| Probe IPA |
| Individuelle Praktische Arbeit M223 |
|  |
| **Olivia Pawlowitz** |
| **01.11.2017** |

|  |
| --- |
| [Geben Sie hier das Exposee für das Dokument ein. Das Exposee ist meist eine Kurzbeschreibung des Dokumentinhalts. Geben Sie hier das Exposee für das Dokument ein. Das Exposee ist meist eine Kurzbeschreibung des Dokumentinhalts.] |

Inhalt

[1 Vorwort 3](#_Toc497394594)

[1.1 Risiken 3](#_Toc497394595)

[1.1.2 Externe Einflüsse 3](#_Toc497394596)

[1.1.3 Falsche Zeiteinschätzung 3](#_Toc497394597)

[1.1.4 Datenverlust 3](#_Toc497394598)

[2 Aufgabenstellung 4](#_Toc497394599)

[3 Deklarationen 7](#_Toc497394600)

[3.1 Vorkenntnisse 7](#_Toc497394601)

[Fachliche Vorkenntnisse 7](#_Toc497394602)

[3.1.1 Architektur 7](#_Toc497394603)

[3 Projektmethode IPERKA 8](#_Toc497394604)

[3.1 Begründung der Wahl 8](#_Toc497394605)

[4 Zeitmanagement 9](#_Toc497394606)

[4.1 Struktur 9](#_Toc497394607)

[4.1.1 Meilensteine 9](#_Toc497394608)

[Journal 10](#_Toc497394609)

[6 Management Summary 12](#_Toc497394610)

[6.1 Projektübersicht 12](#_Toc497394611)

[6.2 Lehrbetrieb 12](#_Toc497394612)

[6.3 Involvierte Personen 12](#_Toc497394613)

[7 Entwicklungsumgebung 13](#_Toc497394614)

[7.1 Rechner 13](#_Toc497394615)

[8 Planung 14](#_Toc497394616)

[8.1 Anforderungen 14](#_Toc497394617)

[8.1.1 Backend 14](#_Toc497394618)

[8.1.2 Frontend 14](#_Toc497394619)

[8.2.1 Versionsverwaltung 14](#_Toc497394620)

[8.2.2 Subversion (SVN) 14](#_Toc497394621)

[Use Case 15](#_Toc497394622)

[Use Case: Login 16](#_Toc497394623)

[Use Case: Register 16](#_Toc497394624)

[Use Case: Create Ticket 16](#_Toc497394625)

[Use Case: View Ticket 16](#_Toc497394626)

[Use Case: Delete Ticket 16](#_Toc497394627)

[8.3 Benutzeroberfläche 17](#_Toc497394628)

[8.4 Datenbank 17](#_Toc497394629)

[9 Entscheidung 18](#_Toc497394630)

[9.1 Versionsverwaltung 18](#_Toc497394631)

[9.2 Datenbank 18](#_Toc497394632)

[9.3 Programmiersprache 18](#_Toc497394633)

[10 Realisierung 19](#_Toc497394634)

[10.1 Umsetzung 19](#_Toc497394635)

[Für die Entwicklung der Login -Applikation wird gemäss der Entscheidung aus der durchgeführten Nutzwertanalyse das CSS-Framework Bootstrap verwendet. 19](#_Toc497394636)

[10.1.1 Projektstruktur 19](#_Toc497394637)

[Testphasen 20](#_Toc497394638)

[Fehlerklassifizierungen 20](#_Toc497394639)

[Testfälle 20](#_Toc497394640)

# 1 Vorwort

Anhand dieser Probe IPA-Dokumentation wird die gesamte, individuelle Praxisarbeit festgehalten und dokumentiert. Alle projektrelevanten Informationen sowie der eigentliche Projektverlauf sind somit klar ersichtlich. Diese Dokumentation wurde im November 2017 von Olivia Pawlowitz im Rahmen des Moduls 223 zur Vorbereitung auf den Lehrabschluss als Informatiker, Schwerpunkt Applikationsentwicklung, verfasst.

# 1.1 Risiken

## 1.1.2 Externe Einflüsse

Die Projektarbeit ist teilweise abhängig von externen Einflüssen, welche die planmässige Durchführung durcheinander bringen könnten. Zu den externen Einflüssen gehören beispielsweise Server, Netzwerk, Internet Zugang, usw.

## 1.1.3 Falsche Zeiteinschätzung

Meistens ist es so, dass der Zeitplan nicht auf das Detail genau eingehalten werden kann. Damit das Projekt trotzdem planungsgemäss erledigt werden kann, wurden Zeitreserven in die Planung mit einbezogen.

## 1.1.4 Datenverlust

Die Daten werden regelmässig gesichert, um den Verlust von relevanten Projektdaten vorzubeugen. Es besteht jedoch ein geringes Risiko, dass Daten während der Projektarbeit verloren gehen könnten.

**1.1.5 Krankheit/Unfall**

Die Projektarbeit kann aus gesundheitlichen Gründen jederzeit unterbrochen werden. Der Endtermin des Projekts verschiebt sich entsprechend der verlorenen Zeit.

# 2 Aufgabenstellung

**Projekt**

Projekttitel: Erstellen einer Multi User Applikation

Ticketing System

Im Auftrag von : Remo Steinmann Modul 223  
Auftragnehmer: Olivia Pawlowitz  
Starttermin: 01. November 2017  
Geplante Projektzeit: 5 Arbeitstage(40h) davon ca. 3 Tage Entwicklung(24h)

Erstellen einer Multi User Applikation, beinhaltend Frontend, Backend, Anbindung an relationaler Datenbank.

Wieso ein Ticket-System?

Es gibt verschiedene Varianten, Kundenanfragen in einem Unternehmen zu bearbeiten. Ticket-Systeme grenzen sich von anderen Methoden ab, sie speichern einzelne Anfragen und bieten dem User eine geniale Übersicht über alle Anfragen. Die Tickets können kategorisiert und unzählige Male abgerufen werden.

Programmiersprache : PHP, JavaScript  
IDE: Eclipse  
DB: MySQL  
Framework: Bootstrap  
Library: JQuery

**Soll Zustand**

Der User kann sich registrieren und sich mit den Anmeldedaten einloggen. Es wird ein neues Fenster geöffnet und der Benutzer sieht die Hauptseite. Dort besteht für ihn die Möglichkeit ein neues Ticket zu erstellen oder alle Tickets anzusehen. Der Benutzer kann ein Ticket ansehen und löschen.

Das Projekt wird mit PHP und JavaScript objektorientiert realisiert. Die Daten werden in einer MySQL Datenbank gespeichert. Es werden 5 Tabellen angelegt.

-Kategorie   
-Abteilung   
-Priorität   
-Tickets   
-User

Beim Login bzw. registrieren werde ich, zum übertragen der Daten, Ajax verwenden. Das Passwort wird jeweils verschlüsselt in die DB eingetragen.

Nachdem der User sich angemeldet bzw. registriert hat, wird dieser auf die Hauptseite weitergeleitet. Dort werden ihm zwei Möglichkeiten angeboten. Entweder er erstellt ein Ticket, er sieht sich alle Tickets an oder er loggt sich wieder aus.

**Fallbeispiel 1: Ticket erstellen**

Der User möchte ein neues Ticket erstellen. Er hat nun die Möglichkeit sein Ticket zu erstellen und muss den jeweiligen Bereich für das Problem auswählen, die Wichtigkeit und die Abteilung in der er sich befindet. Der User gibt dem Ticket ein Titel und fügt ausserdem eine Beschreibung hinzu. Nun kann das Ticket veröffentlicht werden.

**Spezifikationen**

Es dürfen eine maximale Anzahl Wörter in der Beschreibung enthalten sein. Diese Anzahl reduziert sich auf 50 Wörter. Ein Upload von Bildern sprich Screenshots ist möglich und auch erwünscht. Die Anzahl der Bilder, welche hochgeladen werden können ist unbegrenzt.

**Fallbeispiel 2: Ticket ansehen**

Der Benutzer möchte nun alle Tickets ansehen. Es erscheint eine Tabelle mit allen erstellten Tickets. In dieser Tabelle sieht der User, wer Ticket xyz erstellt hat, welche Wichtigkeit es hat, zu welcher Abteilung es gehört und den Titel. Er kann die Tickets löschen und bearbeiten.

In der Ticketview ist es dem User möglich zu sehen, wann ein Ticket erstellt wurde: Datum & Uhrzeit.

**Fallbeispiel 3: User abmelden**

Der User möchte sich abmelden und klickt auf den Button „Logout“.

**Anforderungen an die Applikation**

-Folgende Anforderungen müssen erfüllt werden:

-Multi-User Applikation

-Relationale Datenbank

-Objektorientiert Programmieren

-Transaktionen

-Mehrere Clients müssen gleichzeitig auf den gleichen Datenbestand zugreifen

**Zusätzliche Kriterien**

-194 Plausibilierung der Benutzer-Eingaben

-166 Coding Style – lesbarer Code

-163 Design Doku

-123 Kommentar im Code

-164 Fehlerbehandlung

-130 Vollständiges ERM bzw. Datenmodell

-125 Gliederung des Programms

**Warum dieses Projekt zum Modul Multiuser-Applikationen passt:**

Alle Anforderungen, welche Anfangs Modul gestellt wurden, können erfüllt werden.

**Herausforderung: PHP objektorientiert**

Da ich bis zu diesem Zeitpunkt erst einmal PHP Objektorientiert programmiert habe, wird dies eine Hürde sein, in kürzester Zeit die Aufgabenstellung so genau wie möglich umzusetzen.

# 3 Deklarationen

## 3.1 Vorkenntnisse

In meiner bald 4-jährigen Lehre als Applikationsentwickler konnte ich mir die in diesem Abschnitt beschriebenen Vorkenntnisse erlernen, welche für das Projekt relevant sind.   
Projektrelevante Vorkenntnisse werden stichwortartig festgehalten:

### Fachliche Vorkenntnisse

|  |  |  |
| --- | --- | --- |
| Thema | Kenntnisse | Bemerkungen |
| HTML | sehr gute Kenntnisse | Module absolviert und Selbststudium |
| CSS | sehr gute Kenntnisse | Selbststudium |
| JavaScript | gute Kenntnisse | Selbststudium |
| SQL | sehr gute Kenntnisse | Erfahrung aus Praxis, Modul |
| PHP | gute Kenntnisse | Selbststudium |
| Java | Gute Kenntnisse | 1 Jahr Erfahrung in der IT-Abteilung |

## 3.1.1 Architektur

Die Arbeitsumgebung ist im Meeting Raum „Merkur“ im Gebäude der Berufsbilder der Siemens Schweiz.

# 3 Projektmethode IPERKA

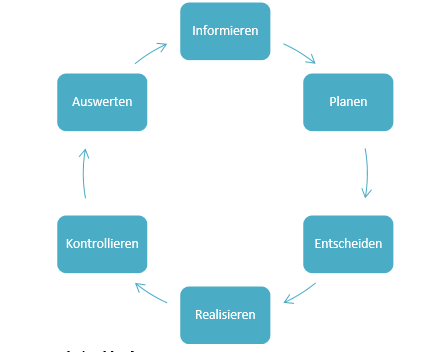
****

Abbildung : IPERKA Arbeitsablauf

In der „Abbildung 1: IPERKA Arbeitsablauf“ sind die einzelnen Projektphasen zu sehen, welche man mit IPERKA unterteilt. Das Ziel von IPERKA ist es, eine klare Struktur zu schaffen nach der gearbeitet wird. IPERKA enthält dabei alle relevanten Phasen eines Projekts.

## 3.1 Begründung der Wahl

Die Projektmanagement-Methode IPERKA haben wir schon am ersten Tag als Informatiker im Basislehrgang kennengelernt und stets bei jedem Projekt eingesetzt. Es bietet meiner Meinung nach einige Vorteile. Das Projekt lässt sich so in übersichtliche Teilschritte gliedern und es ist deshalb immer ersichtlich, in welchem Teilschritt man sich befindet.

Es gibt sicherlich weitere Projektmanagement-Methoden, die geeigneter für die Entwicklung einer Applikation wären, doch für so ein kleines Projekt, wie die IPA, ist IPERKA die einfachste und übersichtlichste Methode, um ein Projekt dieser Art durchzuführen. Zudem haben andere Methoden den Nachteil, dass sie zu viel Zeit für die Planung beanspruchen, was für die IPA nicht sehr günstig wäre.

# 4 Zeitmanagement

## 4.1 Struktur

Es macht Sinn, den Zeitplan nach der IPERKA Methode zu strukturieren, da ich diese Projektmanagement-Methode gewählt habe. Der Zeitplan ist dabei in zwei verschiedene Bereiche gegliedert, nämlich in Hauptarbeitsschritte und wiederkehrende Arbeitsschritte. Es wird in Form eines GANTT-Diagramms dargestellt.

Hier kommt der Zeitplan hinein.

## 4.1.1 Meilensteine

|  |  |  |
| --- | --- | --- |
| Meilenstein | Beschreibung | Datum |
| **Informieren**  SOLL Zeitplan | Der Zeitplan für die SOLL Stunden ist fertig definiert. | 01.11.2017 |
| **Informieren** Use Cases | Use Cases sind geschrieben und in die Dokumentation implementiert | 01.11.2017 |
| **Planen**  ERM | Klassendiagramm erstellt | 02.11.2017 |
| **Entscheiden** Klassen erstellt für DB Login implementiert Frontend implementiert |  | 02.11.2017 |
| **Realisieren** Klassen dokumentieren |  | 03.11.2017 |
| **Kontrollieren** Klassen dokumentieren |  | 08.11.2017 |
| **Auswerten**  Klassen dokumentieren |  | 10.11.2017 |

**5 Arbeitsjournale**

**5.1 Tagesablauf Mittwoch, 01.11.2017 (Tag 1)**

|  |  |  |
| --- | --- | --- |
| **Zeit** | **Erledigte Arbeiten** | **Erfüllt** |
| 8.15 – 10.00 | Zeitplan | Ja |
| 12.30 – 13.00 | Grundgerüst Doku | Ja |
| 13.00 – 14.00 | Use Case dokumentieren | Ja |
| 14.00 – 14.30 | Meilensteine | Ja |
| 14.30 – 16.30 | ERM erstellen (anfangen) | Nein |

### Journal

Ich bin pünktlich um 8.15 in der Berufsbildung eingetroffen, leider konnte ich keinen geeigneten Laptop mitnehmen. Freundlicherweise habe ich einen von der Berufsbildung erhalten.

Als erstes habe ich begonnen den Zeitplan zu erstellen, die erste Version musste um 10.00 Uhr stehen. Da ich noch kein Office auf dem Laptop hatte, habe ich den Zeitplan mithilfe von Excel Online erstellt. Um ca. 9:00 Uhr bekam ich von Remo ein Office Paket, somit konnte ich meinen Zeitplan in Excel übernehmen.

Ich war schon ziemlich zufrieden mit meiner ersten Version des Zeitplans, jedoch sind mir noch ziemlich viele Sachen eingefallen. Noch vor dem Mittagessen habe ich mich an das Grundgerüst der Dokumentation gesetzt. Ich habe begonnen die verschiedenen Projektphasen von IPERAK zu implementieren.

Um Use Case zu erstellen habe ich Umlet verwendet. Das hat ganz gut geklappt jedoch müssen diese noch weiter ausgebaut werden in der Beschreibung. Dies werde ich morgen abschliessen. Ausserdem konnte ich noch Meilensteine definieren. Diese habe ich aus meinem Zeitplan entnommen. Leider hat es heute nicht mehr gereicht, um das ERM fertig zu stellen.

Ich hoffe ich kann dies morgen nachholen, ohne das mein Zeitplan sich verzögert.

**5.2 Tagesablauf Donnerstag, 02.11.2017 (Tag 2)**

|  |  |  |
| --- | --- | --- |
| **Zeit** | **Erledigte Arbeiten** | **Erfüllt** |
| 8.15 – 10.00 | Ist-Soll Analyse | Ja |
| 10.00 – 11.30 | ERM | Ja |
| 12.30-13.30 | Testkonzept | Ja |
| 14.00 – 16.30 | Datenbankkonfiguration dokumentieren | Ja |

### Journal

**5.3 Tagesablauf Freitag, 03.11.2017 (Tag 3)**

**5.4 Tagesablauf Mittwoch, 08.11.2017 (Tag 4)**

**5.5 Tagesablauf Freitag, 10.11.2017 (Tag 5)**

# 6 Management Summary

Die Experten bilden zusammen mit dem Fachausbildner und dem Kunden den Auftraggeber. Zusammen sind sie für die Formulierung der Aufgabenstellung und Bewertung der Projektarbeit als Ganzes zuständig.

6.1 Projektübersicht

Diese Dokumentation, die daraus entstandene Präsentation sowie das anschliessende Fachgespräch sind Bewertungsgrundlagen für meine praktische Lehrabschlussprüfung (Individuelle Praktische Arbeit, kurz IPA). Nach den Standards des Projekt Managements werden folgende Projektphasen dokumentiert:

* Aufgabenstellung
* Planung
* Analyse
* Realisierung
* Testphase
* Auswertung

Zusätzliche Informationen, welche in der nachfolgenden Dokumentation zu finden sind:

* Abbildungsverzeichnis
* Tabellenverzeichnis
* Quellenverzeichnis
* Protokolle
* Glossar
* Anhang

6.2 Lehrbetrieb

Siemens Schweiz AG   
Freilagerstrasse 40  
8047 Zürich

[Telefon](https://www.google.ch/search?site=async/lcl_akp&q=siemens+schweiz+ag+z%C3%BCrich+telefon&sa=X&ved=0ahUKEwiMu_bGgp3XAhWG2BoKHbaJB4AQ6BMIHzAF): 058 558 55 85

6.3 Involvierte Personen

Modulleiter: Remo Steinmann

# 7 Entwicklungsumgebung

## 7.1 Rechner

Die Web-Applikation wird an einem Standard LocalClient der Siemens Schweiz AG entwickelt.

Technische Daten

|  |  |
| --- | --- |
| Hersteller | Lenovo |
| Modell | L570 |
| CPU | Intel Core i5-7300U |
| Arbeitsspeicher | 8.00 GB |
| Betriebssystem | Windows 10 Pro |

**7.1.1 Lokaler Entwicklungsserver**

Als lokalen Entwicklungsserver verwende ich XAMPP.

|  |  |
| --- | --- |
|  | XAMPP 3.2.2 |
| Apache |  |
| PHP |  |
| MySQL |  |

**7.1.2 Entwicklungswerkzeug**

Als Entwicklungswerkzeug wird die integrierte Entwicklungsumgebung (IDE) Eclipse eingesetzt und bietet zahlreiche Funktionen für die Entwicklung von PHP-Applikationen.

# 8 Planung

## 8.1 Anforderungen

Die Anforderungen an die Applikation werden anhand von Kriterien definiert. Die MussKriterien müssen dabei zwingend erfüllt werden.

### 8.1.1 Backend

|  |  |  |
| --- | --- | --- |
| Kriterium | MUSS | KANN |
| Login | x |  |
| Logout |  | x |
| Registrieren | x |  |
| Ticket erstellen | x |  |
| Ticket bearbeiten | x |  |
| Ticket löschen | x |  |
| Bilder uploaden |  | x |

### 8.1.2 Frontend

Für das Backend wurden folgende Kriterien definiert:

|  |  |  |
| --- | --- | --- |
| Kriterium | MUSS | KANN |
| Login Fenster wird dargestellt | x |  |
| Ticket Tabelle wird abgebildet | x |  |
| Delete Funktion | x |  |
| Update Funktion | x |  |

**8.2 Umsetzung**

### 8.2.1 Versionsverwaltung

Die Wahl des Sicherungssystems in der Siemens Schweiz AG ist generell dem Entwickler überlassen, da es keine Richtlinien dafür gibt. In gewissen Abteilungen haben sich aber gewisse Standards etabliert, bei den einen ist es SVN und bei den anderen ist es Git. In diesem Abschnitt werden die Funktionalität und der Unterschied zwischen diesen beiden Varianten erläutert.

### 8.2.2 Subversion (SVN)

Subversion arbeitet mit einem sogenannten Repository, in welchem die Änderungen gespeichert werden und einer „Working Copy“, die den aktuellen Stand enthält, in welcher der Entwickler seine Änderungen durchführt. Es werden dabei, aber nur die Unterschiede zu bereits vorhandenen Ständen übertragen. Dabei ist die Verbindung zum Repository erforderlich um Aktionen durchzuführen. Die Versionierung erfolgt dabei in Form einer Revisionszählung. Bei jeder Änderung wird die Revision um eins hochgezählt.

**8.2.3 Git**

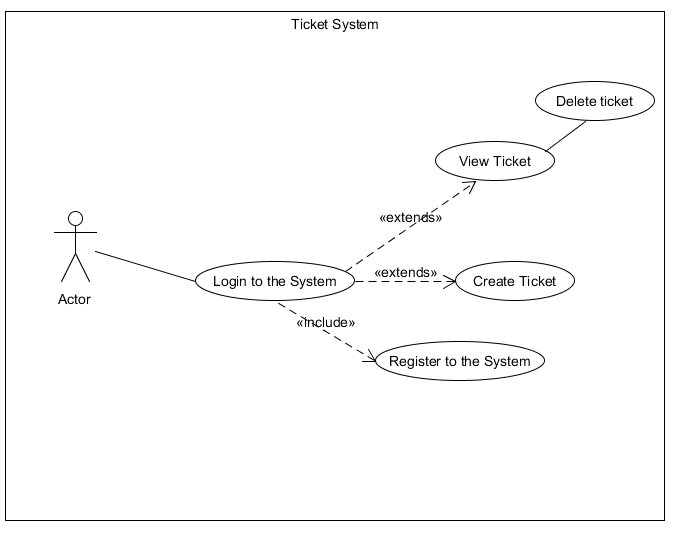
Git hat viele Ähnlichkeiten zu Subversion, doch es unterscheidet sich gänzlich bei der Funktionsweise. Wie auch unter Subversion arbeitet man hier mit einem sogenannten Repository. Es gibt jedoch keine zentrale Stelle, in welcher alle Änderungen gespeichert werden. Jeder Benutzer besitzt eine lokale Kopie des gesamten Repositorys. Die meisten Aktionen können so lokal und ohne Netzwerkzugriff ausgeführt werden. Bei Git ist das Repository gleichzeitig auch die „Working Copy“. Die Versionierung erfolgt dabei in Form einer SHA1-Checksumme, da Git bei einer Änderung das ganze Dateisystem abspeichert.

**8.2.4 Unterschiede**

Die wichtigsten Unterschiede im Überblick:

- Git ist schneller als Subversion   
-Subversion lässt sich einfacher bedienen   
-Unter Subversion ist die Verbindung zum Repository zwingend notwendig   
-Git speichert bei einer Änderung das ganze Dateisystem ab, Subversion hingegen nur die Änderungen

## Use Case



### Use Case: Login

Vorbedingung: Der User muss registriert sein

Der User kann sich anhand eines Login Fensters anmelden. Das Login Fenster besteht aus einem Feld für den Benutzernamen und einem Feld für das Passwort. Die Daten werden überprüft und nach erfolgreicher Übereinstimmung wird der User weitergeleitet.

### Use Case: Register

Um auf die Applikation Zugriff zu haben, muss sich der User registrieren. Dazu gibt er seine E-Mail Adresse und sein Passwort in das Fenster eintragen. Das Passwort muss bestätigt werden.

### Use Case: Create Ticket

Vorbedingung: Der User muss angemeldet sein.  
Der User kann ein neues Ticket eröffnen. Hier müssen folgende Felder ausgefüllt werden.

Titel; Abteilung; Wichtigkeit; Bereich; Beschreibung; Bilder;

Das Ticket kann gespeichert werden.

### Use Case: View Ticket

Der User kann alle Tickets mit Hilfe einer Tabelle ansehen, er sieht den Titel, die Wichtigkeit und die Ticketnummer. Ausserdem hat er hier die Möglichkeit die Tickets zu bearbeiten oder zu löschen.

### Use Case: Delete Ticket

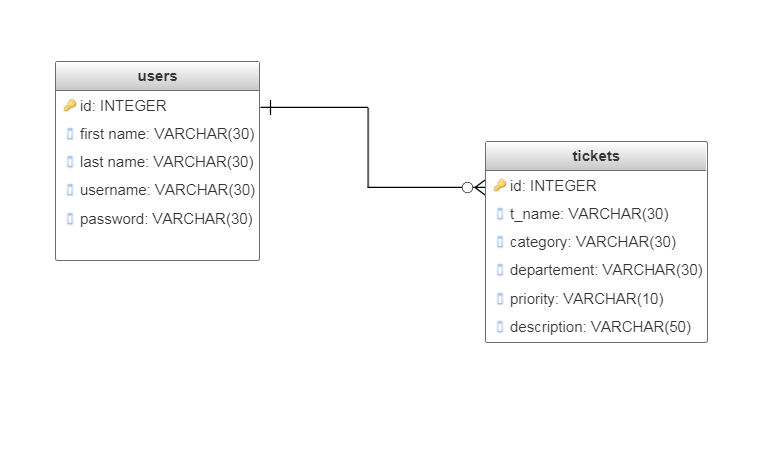
Der User kann ein Ticket aus der Ticket View ansehen und löschen.

## 8.3 Benutzeroberfläche

Die Applikation ist in zwei Teile aufgeteilt, in Frontend und Backend. Für beides gelten keine Richtlinien.

## 8.4 Datenbank

Das ERM dient dazu, Daten zu modellieren ohne technische Aspekte zu beachten. Das ERM stellt hier eine Tabellenstruktur dar, die für das Projekt in eine eigene Datenbank hinzugefügt wird.



# 9 Entscheidung

### 9.1 Versionsverwaltung

Für die Versionsverwaltung ist Git die Beste Wahl, doch da dies ein relativ kleines Projekt ist, macht es keinen grossen Unterschied, ob man Subversion oder Git einsetzt. Da ich mich nicht so gut mit Subversion auskenne habe ich mich dazu entschieden, weiterhin Git für die Versionsverwaltung zu verwenden.

### 9.2 Datenbank

Das Design der Datenbank und die Tabellendefinitionen wurden bereits in der Planung erstellt. Ich werde eine MySQL Datenbank verwenden.

### 9.3 Programmiersprache

Ich verwende die Programmiersprache PHP. Ich habe mich für diese Sprache entschieden, da ich schon vertiefte Grundkenntnisse erlangen konnte und täglich auch damit zu tun habe in meiner Abteilung. Ausserdem kann ich PHP Objekt orientiert programmieren, welches ein MUSS Kriterium ist für dieses Projekt.

# 10 Realisierung

## 10.1 Umsetzung

## Für die Entwicklung der Login -Applikation wird gemäss der Entscheidung aus der durchgeführten Nutzwertanalyse das CSS-Framework Bootstrap verwendet.

## 10.1.1 Projektstruktur

# Testphasen

Im folgenden Abschnitt werden die Testphasen und Methoden aufgezeigt, nach denen die Applikation getestet wird.

Was wird getestet?

Die normalen Anwendungen der Applikation werden in diesen Testphasen getestet. Die Testfälle richten sich nach den definierten Funktionen. Der Schwerpunkt wird dabei auf die Funktionalität der Muss Kriterien gelegt.

## Fehlerklassifizierungen

|  |  |
| --- | --- |
| Fehlerklasse | Beschreibung |
| OK | keine Abweichungen eingetreten |
| Kleine Abweichung | Die Funktionalität ist zwar gegeben, aber nicht vollständig |
| Mittlere Abweichung | Die Funktionalität kann nur eingeschränkt gebraucht werden |
| Grobe Abweichung | Die Funktionalität ist nicht gegeben oder Durchführung führt zu einem Absturz |

# Testfälle

|  |  |
| --- | --- |
| Testfall 01 | Einloggen |
| Ausgangszustand | Login Fenster wird angezeigt um Login Daten einzugeben |
| Beschreibung | Der User möchte sich einloggen, um auf das Ticket System zugreifen zu können. Mit Hilfe des Login Formulars kann der User seine Daten eingeben. |
| Aktion | Der User gibt seinen registrierten Usernamen und sein registriertes Passwort an. Der User klickt auf den Button „Login“ |
| Erwarteter Ergebniszustand | Die Daten werden in der DB überprüft und der User wird weitergeleitet auf index.php |

|  |  |
| --- | --- |
| Testfall 02 | Registrierung mit korrekten Daten |
| Ausgangszustand | Registrierungsformular wird angezeigt |
| Beschreibung | Ein User möchte sich in das Ticket System einloggen und will sich nun registrieren. Mit Hilfe des Registrierungsformulars gibt er seine Daten an. |
| Aktion | Der User gibt seinen Usernamen und sein Passwort ein. Er wiederholt sein Passwort und klickt auf den Button „registrieren“. |
| Erwarteter Ergebniszustand | Die Daten werden nach Klick auf Button „registrieren“ in der DB gespeichert. |
| Testfall 03 | **Ticket erstellen** |
| Ausgangszustand | Formular wird angezeigt, um ein neues Ticket zu erstellen. Spalten; Titel Ticket, Category, Departement, Priority, Description |
| Beschreibung | Ein User möchte ein neues Ticket erstellen, dazu gibt er alle notwendigen Informationen in das Formular ein. |
| Aktion | Der User füllt die Textfelder mit Daten und klickt auf den Submit Button. |
| Erwarteter Ergebniszustand | Die Daten werden nach Klick auf Submit Button in der DB gespeichert. |

|  |  |
| --- | --- |
| Testfall 04 | Ticket speichern |
| Ausgangszustand | Der User gibt alle notwendigen Informationen in das Formular ein. |
| Beschreibung | Der User möchte sein neues Ticket speichern. |
| Aktion | Die Daten werden nach Klick auf den Submit Button in der DB gespeichert. Das Ticket wird in der View angezeigt. |
| Erwarteter Ergebniszustand | Die Daten werden in der DB gespeichert und das Ticket wird ausgegeben. |

|  |  |
| --- | --- |
| Testfall 05 | Ticket löschen |
| Ausgangszustand | Der User kann die Tabelle mit allen Tickets ansehen, es existiert ein Button „delete“ |
| Beschreibung | Ein Ticket soll gelöscht werden |
| Aktion | Der User löscht das Ticket seiner Wahl, in dem er auf den „delete“ Button klickt. |
| Erwarteter Ergebniszustand | Das ausgewählte Ticket wird gelöscht. |

|  |  |
| --- | --- |
| Testfall 06 | Ticket bearbeiten |
| Ausgangszustand | Der User kann die Tabelle mit allen Tickets ansehen, es existiert ein Button „update“ |
| Beschreibung | Ein Ticket soll bearbeitet werden |
| Aktion | Ein Ticket kann bearbeitet werden, wenn der User auf den Button „update“ klickt. |
| Erwarteter Ergebniszustand | Der User klickt auf den Button „update“ und es besteht die Möglichkeit das Ticket anhand des Formulars zu bearbeiten. Anschliessend wird das Ticket wieder gespeichert. |

|  |  |
| --- | --- |
| Testfall 07 | Login mit falschen Daten |
| Ausgangszustand | Loginformular wird angezeigt |
| Beschreibung | Der User möchte sich einloggen, um auf das Ticket System zugreifen zu können. Mit Hilfe des Login Formulars kann der User seine Daten eingeben. |
| Aktion | User gibt falschen Benutzernamen und falsches Passwort ein. |
| Erwarteter Ergebniszustand | Fehlermeldung „Username oder Passwort ist falsch“ |

|  |  |
| --- | --- |
| Testfall 08 | Login mit falschen Daten |
| Ausgangszustand | Loginformular wird angezeigt |
| Beschreibung | Der User möchte sich einloggen, um auf das Ticket System zugreifen zu können. Mit Hilfe des Login Formulars kann der User seine Daten eingeben. |
| Aktion | User gibt falschen Benutzernamen und falsches Passwort ein. |
| Erwarteter Ergebniszustand | Fehlermeldung „Username oder Passwort ist falsch“ |

|  |  |
| --- | --- |
| Testfall 09 | Registrierung mit falschen Daten |
| Ausgangszustand | Registrierungsformular wird angezeigt |
| Beschreibung | Ein User möchte sich in das Ticket System einloggen und will sich nun registrieren. Mit Hilfe des Registrierungsformulars gibt er seine Daten an. |
| Aktion | User gibt schon vorhandenen Benutzernamen ein. |
| Erwarteter Ergebniszustand | Fehlermeldung „Dieser Benutzername ist bereits vergeben“ |

**Testprotokoll**

|  |  |
| --- | --- |
| Name der Testperson | Olivia Pawlowitz |
| Datum der Durchführung |  |
|  |  |

|  |  |
| --- | --- |
| Testfall 01 |  |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

**Testbericht**

**Testergebnis**

**Auswertung**

**Glossar**