|  |
| --- |
| Siemens Schweiz AG |
| Probe IPA |
| Individuelle Praktische Arbeit M223 |

|  |
| --- |
| Olivia Pawlowitz  01.11.2017 |

Inhalt

[1 Vorwort 3](#_Toc497898692)

[1.1 Risiken 3](#_Toc497898693)

[1.1.2 Externe Einflüsse 3](#_Toc497898694)

[1.1.3 Falsche Zeiteinschätzung 3](#_Toc497898695)

[1.1.4 Datenverlust 3](#_Toc497898696)

[1.2 Projektorganisation 4](#_Toc497898697)

[1.2.1 Lehrbetrieb und Durchführungsort 4](#_Toc497898698)

[1.2.2 Kandidat 4](#_Toc497898699)

[1.2.3 Fachvorgesetzter 4](#_Toc497898700)

[2 Aufgabenstellung 5](#_Toc497898701)

[3 Deklarationen 8](#_Toc497898702)

[3.1 Vorkenntnisse 8](#_Toc497898703)

[Fachliche Vorkenntnisse 8](#_Toc497898704)

[3.1.1 Architektur 8](#_Toc497898705)

[4 Projektmethode IPERKA 9](#_Toc497898706)

[3.1 Begründung der Wahl von IPERKA 9](#_Toc497898707)

[5 Zeitmanagement 10](#_Toc497898708)

[5.1 Struktur 10](#_Toc497898709)

[5.1.1 Meilensteine 11](#_Toc497898710)

[Journal 12](#_Toc497898711)

[Journal 13](#_Toc497898712)

[Journal 14](#_Toc497898713)

[Journal 14](#_Toc497898714)

[Journal 15](#_Toc497898715)

[7 Management Summary 16](#_Toc497898716)

[7.1 Projektübersicht 16](#_Toc497898717)

[8 Entwicklungsumgebung 17](#_Toc497898718)

[8.1 Rechner 17](#_Toc497898719)

[9 Planung 18](#_Toc497898720)

[9.1 Anforderungen 18](#_Toc497898721)

[9.1.1 Backend 18](#_Toc497898722)

[9.1.2 Frontend 18](#_Toc497898723)

[9.2.1 Versionsverwaltung 18](#_Toc497898724)

[9.2.2 Subversion (SVN) 18](#_Toc497898725)

[Use Case 19](#_Toc497898726)

[9.3 Benutzeroberfläche 25](#_Toc497898727)

[9.3.1 Backend 25](#_Toc497898728)

[9.4 Datenbank 26](#_Toc497898729)

[9.5 Tabellendefinitionen 27](#_Toc497898730)

[9.5.1 users 27](#_Toc497898731)

[9.5.2 tickets 28](#_Toc497898732)

[10 Entscheidung 29](#_Toc497898733)

[10.1 Versionsverwaltung 29](#_Toc497898734)

[10.2 Datenbank 29](#_Toc497898735)

[10.3 Programmiersprache 29](#_Toc497898736)

[11 Realisierung 30](#_Toc497898737)

[11.1 Umsetzung 30](#_Toc497898738)

[11.1.1 Benutzeroberfläche 30](#_Toc497898739)

[12 Kontrolle 31](#_Toc497898740)

[12.1 Testphasen 31](#_Toc497898741)

[12.1.1 Fehlerklassifizierungen 31](#_Toc497898742)

[12.1.2 Testfälle 31](#_Toc497898743)

[13 Auswertung 37](#_Toc497898744)

[13.1 Möglichkeiten zur Weiterentwicklung 37](#_Toc497898745)

[13.1.1 Wie ich die Arbeit empfunden habe 37](#_Toc497898746)

[13.1.2 Was ich das nächste Mal verbessern könnte 37](#_Toc497898747)

[14 Glossar 37](#_Toc497898748)

[15 Anhang 38](#_Toc497898749)

# 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.

## 1.2 Projektorganisation

## 1.2.1 Lehrbetrieb und Durchführungsort

Firma Lehrbetriebsverbund Siemens Schweiz AG

Adresse Freilagerstrasse 40, 8047 Zürich

## 1.2.2 Kandidat

Name Frau Olivia Pawlowitz Email olivia@pawlowitz.com

Tel. Mobil 079 551 69 50

## 1.2.3 Fachvorgesetzter

Name Angelo Aloise Email angelo.aloise@siemens.com

# 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 |

# 4 Projektmethode IPERKA

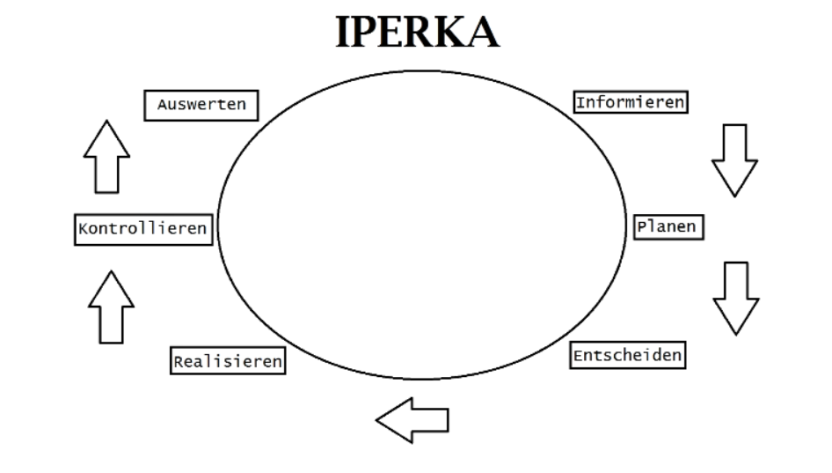


Abbildung : IPERKA Arbeitsablauf

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

## 3.1 Begründung der Wahl von IPERKA

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 zahlreiche Projektmanagement-Methoden, die vielleiht geeigneter für die Entwicklung einer Applikation wären, doch für so ein kleines Projekt, wie die Probe IPA, ist IPERKA die einfachste und übersichtlichste Methode. Zudem haben andere Methoden den Nachteil, dass sie zu viel Zeit für die Planung beanspruchen, was für die Probe IPA nicht sehr passend wäre.

# 5 Zeitmanagement

## 5.1 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.

**Begründung Wahl 2-Stunden-Blöcke**

Ich habe mich für 2-Stunden-Blöcke entschieden, da die meisten Aufgaben so viel Zeit in Anspruch nehmen.1-Stunden-Blöcke wären für ich viel zu unübersichtlich und ich würde meine SOLL Zeit ziemlich unterschätzen, sodass der IST Plan zu sehr abweichen würde.

Hier kommt der Zeitplan hinein.

## 5.1.1 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 | Entscheidungen werden gefällt bezüglich Versionierungsverwaltung, Umsetzung | 02.11.2017 |
| **Realisieren** Klassen dokumentieren | Frontend und Backend | 03.11.2017 |
| **Kontrollieren** Testfälle, Testphasen | Applikation wird getestet und Testfälle sowie Testphasen werden dokumentiert | 08.11.2017 |
| **Auswerten**  Weiterentwicklung, Verbesserungsvorschläge | Was ist gut gelaufen, was ist schlecht gelaufen, was kann verbessert werden? | 10.11.2017 |

# 6 Arbeitsjournale

## 6.1 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.

## 6.2 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

Heute konnte ich pünktlich um 8.15 mit meiner Arbeit beginnen. Als erstes habe ich meinen Zeitplan angeschaut, um nachzusehen, welche Aufgaben ich für den heutigen Tag definiert habe.

Von 8.15-10.00 Uhr habe ich mich an die Ist-Soll Analyse gesetzt. Ich bin gut vorangekommen so dass ich später dann mit dem ERM fortfahren konnte. Bis zur Mittagspause war mein Aufgabenbereich also gut gedeckt. Ich bin schneller vorangekommen als gedacht und konnte die fehlende Zeit von gestern wieder aufholen. Den Nachmittag habe ich damit verbracht mein Testkonzept zu schreiben, ich habe mir alle Testphasen notiert und überlegt, was alles zu testen ist. Mein Testkonzept ist nun vorbereitet und ich kann am Schluss nur noch meine Testdaten einfügen.

Um 13 Uhr hatte ich mein erstes Expertengespräch und ich habe meinem Experten Remo den Zeitplan und mein Arbeitsjournal von gestern gezeigt. Ausserdem habe ich meinen aktuellen Standpunkt erläutert.

Nachdem alle mit dem Expertengespräch durch waren, haben wir einen Input erhalten, wie wir das erste Expertengespräch verbessern könnten. Das fand ich sehr gut und hilfreich. Es muss auf sehr viele kleine Details geachtet werden, damit das Gespräch erfolgreich verläuft.

Morgen werde ich mit der Realisierung beginnen.

## 6.3 Tagesablauf Freitag, 03.11.2017 (Tag 3)

|  |  |  |
| --- | --- | --- |
| **Zeit** | **Erledigte Arbeiten** | **Erfüllt** |
| 8.15 – 9.00 | Frontend planen | Ja |
| 10.00 – 11.30 | ERM Tabellen beschreiben | Ja |
| 12.30 – 13.30 | Klassen erstellen für DB | Ja |
| 14.00 – 16.30 | Frontend implementieren | Ja |

### Journal

Heute habe ich pünktlich um 8.15 mit meiner Arbeit begonnen. Die ersten Stunden habe ich damit verbracht, die Umsetzung des Frontend zu planen und meine Tabellen zu beschreiben, allerdings habe ich diese noch nicht so ausführlich beschrieben wie ich es eigentlich wollte, das werde ich gegen Schluss noch nachholen. Ich habe simple Tabellen erstellen, anhand von diesen Tabellen konnte ich nochmals die Datentypen und die kurze Beschreibung aufzeigen. Nach Beschreiben der Tabellen habe ich sie auch direkt erstellt.

Ich erstellte das File dbconfig.php und konfigurierte die Datenbankverbindung.

Gegen Nachmittag habe ich die Datenbankverbindung hergestellt und angefangen das Frontend zu realisieren. Mit der Implementation bin ich heute schon ziemlich weit gekommen, ich habe mir mein Layout vorgezeichnet, so wie ich es haben möchte und versucht nach zu bauen. Zuerst hatte ich Mühe, weil ich ein paar Flüchtigkeitsfehler gemacht habe und etwas unkonzentriert war. Jedoch habe ich nach der Pause wieder Fuss gefasst und kam gut voran. Ich bin schon sehr zufrieden mit meinem Frontend, jedoch muss ich nächsten Mittwoch zügig arbeiten, denn sonst wird es knapp.

## 6.4 Tagesablauf Mittwoch, 08.11.2017 (Tag 4)

|  |  |  |
| --- | --- | --- |
| **Zeit** | **Erledigte Arbeiten** | **Erfüllt** |
| 8.15 – 9.00 | Frontend implementieren | Ja |
| 09.00 – 11.30 | Login implementieren | Ja |
| 12.30 – 14.30 | Backend implementieren | Ja |
| 15.40 – 16.30 | Klassen dokumentieren | Ja |

### Journal

Heute konnte ich pünktlich um 8.15 Uhr starten. Ich habe am Freitag angefangen das Frontend zu implementieren. Um ca. 9.00 Uhr hatte ich das Frontend soweit fertig. Später dann habe ich begonnen das Login zu implementieren, anfangs hatte ich ein paar Schwierigkeiten mit der Datenbank Verbindung aber schlussendlich hat sich herausgestellt, das waren nur Flüchtigkeitsfehler.

## 6.5 Tagesablauf Freitag, 10.11.2017 (Tag 5)

|  |  |  |
| --- | --- | --- |
| **Zeit** | **Erledigte Arbeiten** | **Erfüllt** |
| 8.15 – 9.00 | Frontend planen | Ja |
| 10.00 – 11.30 | ERM Tabellen beschreiben | Ja |
| 12.30 – 13.30 | Klassen erstellen für DB | Ja |
| 14.00 – 16.30 | Frontend implementieren | Ja |

### Journal

# 8 Entwicklungsumgebung

## 8.1 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 |

**8.1.1 Lokaler Entwicklungsserver**

Als lokalen Entwicklungsserver verwende ich XAMPP.

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

**8.1.2 Entwicklungswerkzeug**

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

# 9 Planung

## 9.1 Anforderungen

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

### 9.1.1 Backend

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

### 9.1.2 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 |  |

**9.2 Umsetzung**

### 9.2.1 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.

### 9.2.2 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.

**9.2.3 Git**

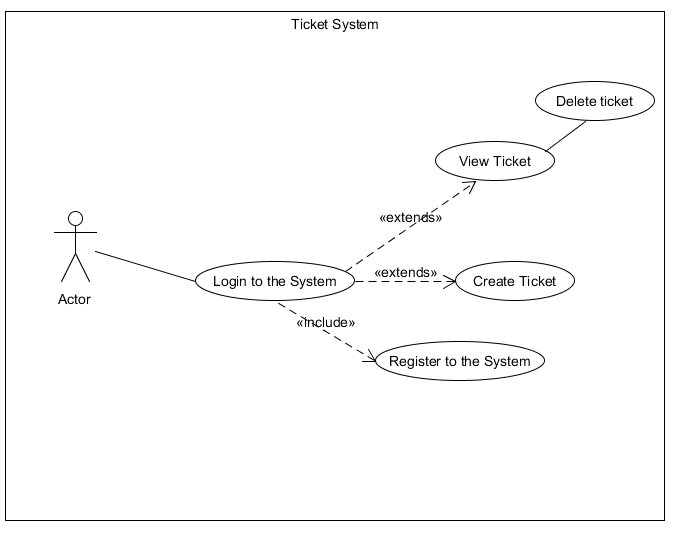
Git hat viele Ähnlichkeiten zu Subversion, doch es unterscheidet sich gänzlich bei der Funktionsweise. Wie auch unter Subversion arbeitet man hier mit einem sogenannten Repository. Es gibt jedoch keine zentrale Stelle, in welcher alle Änderungen gespeichert werden. Jeder Benutzer besitzt eine lokale 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.

**9.2.4 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 |  |
| Use Case | User anmelden |
| Beschreibung | Der User möchte sich einloggen |
| Ziel / Ergebnis | Am Ende sollte der User die Möglichkeit haben, Zugriff auf das Ticket System zu haben. |
| Kategorie | primär (notwendig u. Häufig verwendet) |
| Vorbedingungen | Die einzige Vorbedingung wäre, dass der User überhaupt die Möglichkeit hat sich zu registrieren |
| Nachbedingungen | Erfolg: Wenn der User erfolgreich seinen Benutzernamen und sein Passwort eingegeben hat.  Fehlschlag: Der Benutzer kann sich nicht einloggen.(Benutzername oder Passwort falsch) |
| Invarianten | Die Funktion kann für alle definierten Personen erstellt werden. |
| Akteure | Der User, der sich einloggen möchte ist der Akteur |
| Auslösendes Ereignis | UseCase wird initiiert, wenn das Programm gestartet wird und der User sich erfolgreich eingeloggt hat. |
| Ablaufbeschreibung | **Beschreibung des Standardfalls**  1. User startet das Programm  2. User klickt in das Textfeld“Username“  3. User klickt in das Textfeld „Passwort“  4. User gibt Userame und Passwort ein  5. User klickt auf „login“  6. User wird weiteregeleitet nach erfolgreichem Login  **Alternativen des Standardfalls**  6.1 Anstelle von „Login“ klickt der User auf „Registrieren“ falls dieser sich noch nicht registriert hat. |

|  |  |
| --- | --- |
| Use Case |  |
| Use Case | User registrieren |
| Beschreibung | Der User möchte sich registrieren |
| Ziel / Ergebnis | Am Ende sollte der User die Möglichkeit haben, sich in das System einzuloggen |
| Kategorie | primär (notwendig u. Häufig verwendet) |
| Vorbedingungen | Die einzige Vorbedingung wäre, dass der User überhaupt die Möglichkeit hat das Fenster „Registrieren“ zu öffnen |
| Nachbedingungen | Erfolg: Wenn der User erfolgreich seinen Benutzernamen, sein Passwort, seinen Namen, und seine Email Adresse eingegeben hat.  Fehlschlag: Der Benutzer kann sich nicht einloggen, da seine Daten nicht stimmen. Er hat zum Beispiel eine falsche Emailadresse eingegeben oder der Benutzer ist schon vorhanden. |
| Invarianten | Die Funktion kann für alle definierten Personen erstellt werden. |
| Akteure | Der User, der sich registrieren möchte ist der Akteur |
| Auslösendes Ereignis | UseCase wird initiiert, wenn das Programm gestartet wird und der User sich erfolgreich registriert hat. |
| Ablaufbeschreibung | **Beschreibung des Standardfalls**  1. User startet das Programm  2. User klickt in das Textfeld“Name“  3. User klickt in das Textfeld „Username“  User klickt in das Textfeld „Email“  User klickt in das Textfeld „Passwort“  4. User gibt seine Daten ein  5. User klickt auf „register“ |

|  |  |
| --- | --- |
| Use Case |  |
| Use Case | Create Ticket |
| Beschreibung | Der User möchte ein neues Ticket erstellen |
| Ziel / Ergebnis | Am Ende sollte der User die Möglichkeit haben, ein neues Ticket zu erstellen |
| Kategorie | primär (notwendig u. Häufig verwendet) |
| Vorbedingungen | Die einzige Vorbedingung wäre, dass der User überhaupt die Möglichkeit hat sich einzuloggen um auf das Ticket System zugreifen zu können |
| Nachbedingungen | Erfolg: Der User kann ein neues Ticket erstellen, indem er das Formular ausfüllt und auf speichern klickt.  Fehlschlag: Der Benutzer kann das Ticket nicht speichern. |
| Invarianten | Die Funktion kann für alle definierten Personen erstellt werden. |
| Akteure | Der User, der ein neues Ticket erstellen möchte ist der Akteur |
| Auslösendes Ereignis | UseCase wird initiiert, wenn der User erfolgreich ein neues Ticket erstellt hat. |
| Ablaufbeschreibung | **Beschreibung des Standardfalls**  1. User startet das Programm  2. User klickt in das Textfeld“Ticket Name“  3. User klickt in das Textfeld „Category“  User klickt in das Textfeld „Departement“  User klickt in das Textfeld „Priority“  User klickt in das Textfeld „Description“  4. User gibt seine Daten ein  5. User klickt auf „speichern“  6. Erstelltes Ticket wird angezeigt |

|  |  |
| --- | --- |
| Use Case |  |
| Use Case | View Ticket |
| Beschreibung | Der User möchte alle Tickets ansehen |
| Ziel / Ergebnis | Am Ende sollte der User die Möglichkeit haben alle Tickets anzusehen |
| Kategorie | primär (notwendig u. Häufig verwendet) |
| Vorbedingungen | Die einzige Vorbedingung wäre, dass der User überhaupt die Möglichkeit hat sich einzuloggen um auf das Ticket System zugreifen zu können |
| Nachbedingungen | Erfolg: Der User kann auf das System zugreifen und sieht die Tabelle mit allen erstellten Ticktes.  Fehlschlag: Der Benutzer kann nicht auf das System zugreifen und sieht die Tabelle nicht. |
| Invarianten | Die Funktion kann für alle definierten Personen erstellt werden. |
| Akteure | Der User, der die Tickets ansehen möchte ist der Akteur |
| Auslösendes Ereignis | UseCase wird initiiert, wenn der User erfolgreich ein neues ein Ticket aufrufen kann. |
| Ablaufbeschreibung | **Beschreibung des Standardfalls**  1. User startet das Programm  2. User loggt sich ein  3. User kann Tabelle mit Tickets sehen  4. Erstelltes Ticket wird angezeigt |

|  |  |
| --- | --- |
| Use Case |  |
| Use Case | Delete Ticket |
| Beschreibung | Der User möchte ein Ticket löschen |
| Ziel / Ergebnis | Am Ende sollte der User die Möglichkeit haben ein Ticket zu löschen |
| Kategorie | primär (notwendig u. Häufig verwendet) |
| Vorbedingungen | Die einzige Vorbedingung wäre, dass der User überhaupt die Möglichkeit hat Tickets zu erstellen um diese später auch wieder zu löschen. |
| Nachbedingungen | Erfolg: Der User kann mit dem Button „löschen“ ein Ticket löschen.  Fehlschlag: Der Benutzer kann kein Ticket erstellen. Somit kann er auch keine Tickets löschen. |
| Invarianten | Die Funktion kann für alle definierten Personen erstellt werden. |
| Akteure | Der User, der das Ticket löschen möchte ist der Akteur |
| Auslösendes Ereignis | UseCase wird initiiert, wenn der User erfolgreich ein Ticket löschen kann |
| Ablaufbeschreibung | **Beschreibung des Standardfalls**  1. User startet das Programm  2. User loggt sich ein  3. User kann Tabelle mit Tickets sehen  4. User klickt auf „löschen“  5. Ticket wird entfernt |

## 9.3 Benutzeroberfläche

Die Applikation ist in zwei Bereiche aufgeteilt, das Frontend und das Backend. Für beides existieren keine Richtlinien und ich kann frei wählen.

### 9.3.1 Backend

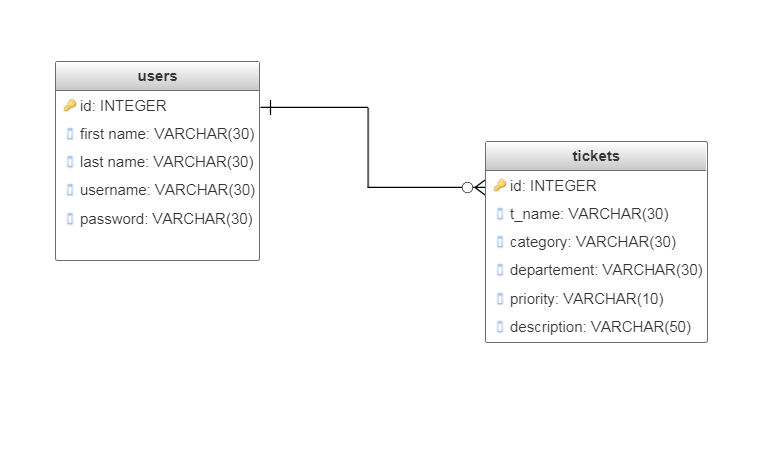
Da ich bis jetzt die meisten Backend mit Bootstrap realisiert habe und ich dies vom Design her sehr ansprechend finde, verwende ich auch bei diesem Projekt das CSS Framework.

Bootstrap Version: v3.3.7

<http://getbootstrap.com/>

## 9.4 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.



## 9.5 Tabellendefinitionen

### 9.5.1 users

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Spaltenname | Typ | PK | NN | Beschreibung |
| id | INT(10) | X | X | Primärschlüssel der Tabelle |
| First name | Varchar(30) |  | X | Vorname des Benutzers |
| Last name | Varchar(30) |  | x | Nachname des Benutzers |
| Username | Varchar(30) |  | X | Benutzername |
| Password | Varchar(30) |  | X | Passwort des Benutzers |

### 9.5.2 tickets

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Spaltenname | Typ | PK | NN | Beschreibung |
| id | INT(10) | X | X | Primärschlüssel der Tabelle |
| t\_name | Varchar(30) |  | X | Name des Tickets |
| category | Varchar(30) |  | X | In welche Kategorie: Bug Report etc. |
| departement | Varchar(30) |  | X | Zu welcher Abteilung der Bug Report gehört |
| priority | Varchar(10) |  |  | Wie hoch die Wichtigkeit des Tickets ist |
| description | Varchar(50) |  | X | Beschreibung des Tickets |

# 10 Entscheidung

### 10.1 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.

### 10.2 Datenbank

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

### 10.3 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.

# 11 Realisierung

## 11.1 Umsetzung

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

### 11.1.1 Benutzeroberfläche

# 12 Kontrolle

## 12.1 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.

## 12.1.1 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 |

## 12.1.2 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 | 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“ |

|  |  |
| --- | --- |
| Testfall 09 | Ticket Description mit mehr 50+ Zeichen befüllen |
| Ausgangszustand | Formular wird angezeigt um neues Ticket zu eröffnen |
| Beschreibung | Der User gibt möchte eine Ticket Beschreibung eingeben |
| Aktion | User gibt 50+ Zeichen in das Textfeld ein. |
| Erwarteter Ergebniszustand | Das Ticket kann nicht gespeichert werden, da nur 50 Zeichen in der Beschreibung erlaubt sind. |

|  |  |
| --- | --- |
| Testfall 10 | Registrierung mit falschen Daten/Sonderzeichen |
| 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 Sonderzeichen ein |
| Erwarteter Ergebniszustand | Fehlermeldung „Bitte geben Sie einen korrekten Benutzernamen, Passwort etc. ein.“ |

|  |  |
| --- | --- |
| Testfall 11 | Login mit falschen Daten/Sonderzeichen |
| Ausgangszustand | Login Formular wird angezeigt |
| Beschreibung | Ein User möchte sich in das Ticket System einloggen. Mit Hilfe des Login Formulars gibt er seine Daten an. |
| Aktion | User gibt Sonderzeichen ein |
| Erwarteter Ergebniszustand | Fehlermeldung „Dieser Username bzw. dieses Passwort ist nicht vorhanden.“ |

**12.2 Testprotokoll**

|  |  |
| --- | --- |
| Name der Testperson | Olivia Pawlowitz |
| Datum der Durchführung | 10.11.2017 |
|  |  |

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

|  |  |
| --- | --- |
| Testfall 02 | Registrierung mit korrekten Daten |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 03 | Ticket erstellen |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 04 | Ticket speichern |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 05 | Ticket löschen |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 06 | Ticket bearbeitem |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 07 | Login mit falschen Daten |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 08 | Registrierung mit falschen Daten |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 09 | Ticket Description mit mehr 50+ Zeichen befüllen |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 10 | Registrierung mit falschen Daten/Sonderzeichen |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

|  |  |
| --- | --- |
| Testfall 11 | Login mit falschen Daten/Sonderzeichen |
| Ausgangszustand |  |
| Beschreibung |  |
| Aktion |  |
| Erwarteter Ergebniszustand |  |
| Ergebnis / Abweichung |  |

**12.3 Testbericht**

**12.4 Testergebnis**

# 13 Auswertung

## 13.1 Möglichkeiten zur Weiterentwicklung

## 13.1.1 Wie ich die Arbeit empfunden habe

## 13.1.2 Was ich das nächste Mal verbessern könnte

# 14 Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
|  |  |
|  |  |
|  |  |

# 15 Anhang

## Management Summary

## Web Summary