April 2016, v1.6

IPA-Bericht

Samuel Keller

Änderungen:

1.6

Implementierungs-Text hinzugefügt.

Tagesjournal 8

Änderungen im Zeitplan

1.5

Tagesjournal 7 hinzugefügt

Schreibfehleränderungen in Journale

1.4.

Journal Tag 6 erweitert

Formatierungen der Journale verändert

Funktionen, die in einer Version 2.0 hinzugefügt werden könnten, dokumentiert.

User deaktivieren Feature eingebaut

**Inhaltsverzeichnis**

[1 Obligatorischer Teil 6](#_Toc448384091)

[1.1 Ausgangslage 6](#_Toc448384092)

[1.2 Detaillierte Aufgabenstellung 6](#_Toc448384093)

[1.2.1 Login 6](#_Toc448384094)

[1.2.2 Benutzerverwaltung 6](#_Toc448384095)

[1.2.3 Die Verwaltung von Tickets 6](#_Toc448384096)

[1.2.4 Ticket Workflow 7](#_Toc448384097)

[1.2.5 Dokumentation 7](#_Toc448384098)

[1.2.6 Design 7](#_Toc448384099)

[1.2.7 Testing 7](#_Toc448384100)

[1.2.8 Versionsverwaltung 8](#_Toc448384101)

[1.2.9 Abgrenzungen 8](#_Toc448384102)

[1.2.10 Zeitplan 8](#_Toc448384103)

[1.2.10.1 Initialisierung (Montag, 04.04.2016) 8](#_Toc448384104)

[1.2.10.2 Analyse (Dienstag, 05.04.2016) 8](#_Toc448384105)

[1.2.10.3 Entwurf (Mittwoch, 06.04.2016) 8](#_Toc448384106)

[1.2.10.4 Implementierung (Mittwoch, 06.04.2016 - Dienstag, 12.04.2016) 8](#_Toc448384107)

[1.2.10.5 Testing (Montag, 11.04.2016 - Dienstag, 13.04.2016) 8](#_Toc448384108)

[1.2.10.6 Abschlussphase (Montag, 13.04.2016 - Dienstag, 14.04.2016) 8](#_Toc448384109)

[1.2.10.7 Buffer (Freitag, 15.04.2016) 8](#_Toc448384110)

[1.2.11 Mittel und Methoden 9](#_Toc448384111)

[1.2.12 Vorkenntnisse 9](#_Toc448384112)

[1.2.13 Vorarbeiten 9](#_Toc448384113)

[1.2.14 Neue Lerninhalte 9](#_Toc448384114)

[1.3 Projektorganisation 9](#_Toc448384115)

[1.3.1 Projektmanagementmethode 9](#_Toc448384116)

[1.3.1.1 Initialisierungsphase 9](#_Toc448384117)

[1.3.1.2 Analysephase 10](#_Toc448384118)

[1.3.1.3 Entwurfsphase 10](#_Toc448384119)

[1.3.1.4 Implementierungsphase 10](#_Toc448384120)

[1.3.1.5 Testingphase 10](#_Toc448384121)

[1.3.1.6 Abschlussphase 10](#_Toc448384122)

[1.3.2 Zeitplan 11](#_Toc448384123)

[1.3.2.1 Vorgang 12](#_Toc448384124)

[1.4 Arbeitsjournal 13](#_Toc448384125)

[1.4.1 Montag, 4. April 2016 13](#_Toc448384126)

[1.4.2 Dienstag, 5. April 2016 14](#_Toc448384127)

[1.4.3 Mittwoch, 6. April 2016 15](#_Toc448384128)

[1.4.4 Donnerstag, 7. April 2016 16](#_Toc448384129)

[1.4.5 Freitag, 8. April 2016 17](#_Toc448384130)

[1.4.6 Montag, 11. April 201 18](#_Toc448384131)

[1.4.7 Dienstag, 12. April 2016 19](#_Toc448384132)

[1.4.8 Mittwoch, 13. April 2016 20](#_Toc448384133)

[1.4.9 Donnerstag, 14. April 2016 21](#_Toc448384134)

[1.4.10 Freitag, 15. April 2016 22](#_Toc448384135)

[2 Projektdokumentation 23](#_Toc448384136)

[2.1 Soll / Ist vergleich 23](#_Toc448384137)

[2.1.1 Ausgangssituation 23](#_Toc448384138)

[2.1.2 Endsituation 24](#_Toc448384139)

[2.2 Risikoanalyse 25](#_Toc448384140)

[2.2.1 Datenverlust 25](#_Toc448384141)

[2.2.1.1 Wahrscheinlichkeit 25](#_Toc448384142)

[2.2.1.2 Folgen 25](#_Toc448384143)

[2.2.1.3 Gegenmassnahmen 25](#_Toc448384144)

[2.2.1.4 Schadensminimierung 25](#_Toc448384145)

[2.2.2 Zeitprobleme 25](#_Toc448384146)

[2.2.2.1 Wahrscheinlichkeit 25](#_Toc448384147)

[2.2.2.2 Folgen 26](#_Toc448384148)

[2.2.2.3 Gegenmassnahmen 26](#_Toc448384149)

[2.2.2.4 Schadensminimierung 26](#_Toc448384150)

[2.2.3 Internetbrowser-Probleme 26](#_Toc448384151)

[2.2.3.1 Wahrscheinlichkeit 26](#_Toc448384152)

[2.2.3.2 Folgen 26](#_Toc448384153)

[2.2.3.3 Gegenmassnahmen 26](#_Toc448384154)

[2.2.3.4 Schadensminimierung 26](#_Toc448384155)

[2.3 Versionierung 27](#_Toc448384156)

[2.3.1 Backup 27](#_Toc448384157)

[2.3.2 Commits 27](#_Toc448384158)

[2.4 Wieso Bootstrap 28](#_Toc448384159)

[2.4.1 Was ist Bootstrag 28](#_Toc448384160)

[2.4.2 Vorteile 28](#_Toc448384161)

[2.4.3 Nachteile 28](#_Toc448384162)

[2.5 Wieso Bootstrap bewnutz wird 28](#_Toc448384163)

[3 Design 29](#_Toc448384164)

[3.1 Datenbank 29](#_Toc448384165)

[3.1.1 Vor der Erstellung der Applikation 29](#_Toc448384166)

[3.1.2 Nach der Erstellung der Applikation 30](#_Toc448384167)

[4 Implementierung 30](#_Toc448384168)

[4.1 Standard GUI 30](#_Toc448384169)

[4.1.1 header.inc.php 30](#_Toc448384170)

[4.1.2 navigation.inc.php 30](#_Toc448384171)

[4.1.3 footer.inc.php 30](#_Toc448384172)

[4.2 Autorisierung Seiten 31](#_Toc448384173)

[4.2.1 checkauth.inc.php 31](#_Toc448384174)

[4.2.2 checkrole.inc.php 31](#_Toc448384175)

[4.3 Loginseiten 31](#_Toc448384176)

[4.3.1 login.php 31](#_Toc448384177)

[4.3.2 logout.php 31](#_Toc448384178)

[4.4 Datenbank Verbindungsseiten 31](#_Toc448384179)

[4.4.1 dbsettings.inc.php 31](#_Toc448384180)

[4.4.2 dbconnection.inc.php 31](#_Toc448384181)

[4.4.3 pdosettings.inc.php 31](#_Toc448384182)

[4.4.4 pdoclass.inc.php 31](#_Toc448384183)

[4.5 Funktionsseiten 31](#_Toc448384184)

[4.5.1 functioncontroller.inc.php 31](#_Toc448384185)

[4.5.2 sendmail.inc.php 31](#_Toc448384186)

[4.6 Ticketverwaltungsseiten 31](#_Toc448384187)

[4.6.1 index.php 31](#_Toc448384188)

[4.6.2 ticketfilter.inc.php 31](#_Toc448384189)

[4.6.3 ticketliste.inc.php 31](#_Toc448384190)

[4.6.4 neuesticket.php 31](#_Toc448384191)

[4.6.5 ticket\_details.php 31](#_Toc448384192)

[4.7 Userverwaltung 31](#_Toc448384193)

[4.7.1 profil.php 31](#_Toc448384194)

[4.7.2 usermanagement.php 31](#_Toc448384195)

[4.7.3 neuernutzer.php 31](#_Toc448384196)

[4.7.4 benutzer\_details.php 31](#_Toc448384197)

[5 Testing 32](#_Toc448384198)

[6 Abschlussbericht 32](#_Toc448384199)

[6.1 Rückblick 32](#_Toc448384200)

[6.2 Lessons learned & Was ist gut gelaufen 32](#_Toc448384201)

[6.3 Was schief gelaufen ist 32](#_Toc448384202)

[6.4 Mit welchen Funktionen kann das Projekt sinnvoll erweitert werden? 32](#_Toc448384203)

[6.4.1 Tickets und Benutzer in der Detailübersicht 32](#_Toc448384204)

[6.4.2 Benutzerfilter und/oder Ansicht für deaktivierte Nutzer 32](#_Toc448384205)

[6.5 Resultat 32](#_Toc448384206)

[6.6 Schlusswort 32](#_Toc448384207)

[7 Quellenverzeichnis 32](#_Toc448384208)

[7.1 Versionierung 32](#_Toc448384209)

[7.2 Risikoanalyse 33](#_Toc448384210)

[7.3 Bootstrap 33](#_Toc448384211)

[8 Anhang 33](#_Toc448384212)

[8.1 Testprotokolle 33](#_Toc448384213)

[8.2 Quellcode 33](#_Toc448384214)

# 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 hätte ich auch nutzen können. Dieses wurde in der Schule auch als Alternative beworben. Mangels Kenntnisse und Sicherheit in dieser Methode wurde schliesslich aber 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.

#### Entwurfsphase

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

#### Implementierungsphase

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

#### Testingphase

In dieser Phase wird die Qualitäts der Applikation 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 und das Benutzerhandbuch geschrieben

### Zeitplan

#### Vorgang

Die Applikation ist in das Login, die Benutzerverwaltung und in die Ticketverwaltung mit E-Mail-Versand aufgeteilt. Da diese Applikationsteile teilweise aufeinander aufbauen, müssen diese in einer bestimmten Reihenfolge Implementiert werden.

Angefangen wird mit der rohen Struktur der Webseite in der Entwurfsphase. Auf dieser wird dann das Loginsystem aufgebaut. Nach der Erstellung des Logins wird die Nutzerverwaltung implementiert. Erst am Schluss wird das Erstellen und Bearbeiten von Tickets ermöglicht, da diese die Benutzerverwaltung und das Login voraussetzen.

Ein Grossteil der Dokumentation wird vor der Implementierung in der Analysephase und Entwurfsphase geschrieben. Die Dokumentation der Implementierung wird während der Implementierungsphase und in der Abschlussphase erstellt. Das Benutzerhandbuch wird nach der Erstellung der Applikation und nach dem Testing erstellt.

## Arbeitsjournal

### Montag, 4. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsplatz einrichten * Lokales Repository erstellen * Repository auf GitHub erstellen * Dokumentstruktur erstellen * Dokumentation des obligatorischen Teils 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 Implementierung aufgebaut werden kann * Benutzerhandbuch mit Text erweitern |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Ich habe zwei zusätzliche Use-Case Grafiken erstellt, die den Kommunikationsweg aufzeigen. |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| 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. Der Webseitenentwurf gefällt mir sehr gut und ich freue mich auf die Implementierung. Das Benutzerhandbuch hat eine genauere Gliederung sowie neue Texte erhalten. |
| Arbeitszeit | 08:30 Uhr – 10:15 Uhr  Kurze Pause  10:20 Uhr – 12:40 Uhr  Mittagspause  13:40 Uhr – 18:00 Uhr |
| Visum |  |

### Donnerstag, 7. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 3) in die Dokumentation übernehmen * Datenbank erstellen und teilweise anpassen * Datenbankanbindung erstellen * Login und Session implementieren * Navigation fertigstellen |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Ich habe gewisse Feldtypen in der Datenbank angepasst. Am Anfang hatte jedes Feld den Typ „Text“. Dieser Typ wird nach und nach bei mehreren Tabellen ersetzt, da „Text“ nicht für den Benutzernamen oder eine Telefonnummer geeignet ist. |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme | Die Implementierung des Logins hat mich mehr Zeit gekostet als erwartet, da ich noch nicht viel Erfahrung mit MySQLi habe. Ich habe zuvor immer mit PDO-Verbindungen gearbeitet. Da die IPA aber ausschliesslich Prozedural ist, konnte ich PDO leider nicht verwenden. Der Implementierungaufwand mit PDO wäre um einiges geringer gewesen. |
| Unterstützung Dritter | Ich habe die Datenbankanpassung mit Detlef Nünninghoff besprochen. Er hat ebenfalls empfohlen die Feldtypen zu ändern. |
| Soll / Ist Zeitvergleich | Für heute war die Implementation des Logins angedacht. Dieses wurde vollumfänglich implementiert. |
| Sonstiges | Ich bin froh, dass ich endlich bei der Implementierung angekommen bin. Dieser Teil des Projekts macht mir am meisten Spass. |
| Arbeitszeit | 08:30 Uhr – 13:10 Uhr  Mittagspause  13:40 Uhr – 16:00 Uhr  Kurze Pause  16:10 Uhr – 17:30 Uhr |
| Visum |  |

### Freitag, 8. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 4) in die Dokumentation übernehmen * Implementierung des Benutzerprofilos mit Validierung * Implementierung der Funktion „Neuer Nutzer erstellen“ * Implementierung der Benutzerliste |
| Nicht erreichte Ziele | Ich hatte nicht genug Zeit, um die Detailansicht der Benutzer zu erstellen. Diese muss am Montag und Dienstag zusammen mit den Ticketfunktionen umgesetzt werden, um wieder im Zeitplan zu liegen. |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Dokumentierung der eingefügten Funktionen per Hand. |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | keine |
| Soll / Ist Zeitvergleich | Wie bereits erwähnt, bin ich nicht mehr im Zeitplan, da ich nicht genug Zeit hatte, um alles an einem Tag umzusetzen. |
| Sonstiges | Da ich bemerkt habe, dass es mir zeitlich nicht für die Benutzerübersicht reichen wird, habe ich angefangen, Diagramme der Funktionen / Teilprogramme zu zeichnen. |
| Arbeitszeit | 08:30 Uhr – 12:40 Uhr  Mittagspause  13:40 Uhr – 15:20 Uhr  Kurze Pause  15:30 Uhr – 17:30 Uhr |
| Visum |  |

### Montag, 11. April 201

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 5) übernehmen * Projektvorgangtext in die Dokumentation übernehmen * Soll / Ist Vergleichstext dokumentieren * Use-Case-Grafiken dokumentieren * Leere Tagesjournale in die Dokumentation einfügen * Detailansicht für die Benutzer implementieren * Benutzer-Löschfunktion implementieren * Datenbank angepasst * Nutzer Löschfunktion umgebaut * PDO Datenbankverbindung einbauen * Ticketliste implementieren * Ticketfilter implementieren |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | Während dem Dokumentieren ist mir eine zusätzliche Funktion eingefallen, mit dem ich das Ticketsystem in einer zweiten Version erweitern könnte. Diese habe ich im Abschlussbericht eingefügt. Da ich jetzt PDO nutze, und mit dem mehr vertraut bin, geht die Implementation der Funktionen einiges schneller. |
| Ungeplante Arbeiten | Ich habe der Nutzertabelle ein Feld mit dem Typ Boolean hinzugefügt, welches anzeigt ob der Nutzer aktiv ist oder nicht. Daraufhin wurden ebenfalls die SQLi Scripts vom Login und der Benutzerübersicht angepasst. |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | keine |
| Soll / Ist Zeitvergleich | Ich bin etwas in Zeitdruck, da die Benutzerübersicht ebenfalls heute implementiert werden muss. Kurz vor dem Expertenbesuch war ich damit fertig und bin jetzt wieder im Zeitplan. |
| Sonstiges | Ich habe ein paar der von Hand erstellten Texte endlich in die Dokumentation übernommen und alle zukünftigen Journale in die Dokumentation eingefügt, um Formatier Probleme entgegen zu wirken. Die neue Wordversion verträgt sich scheinbar nicht gut mit der älteren.  Die Experten waren heute für etwa eine halbe Stunde da. Diese Zeit werde ich nachholen. |
| Arbeitszeit | 08:30 Uhr – 11:15 Uhr  Expertenbesuch  11:45 Uhr – 12:30 Uhr  Mittagspause  13:30 Uhr – 18:35 Uhr |
| Visum |  |

### Dienstag, 12. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 6) übernehmen * Detailansicht für Tickets implementieren * Datenbank Feldtypen ändern * Datenbankeinträge zu den Kategorien ändern * „Ticket erstellen“-Funktion implementieren. * Benutzer- und Profilvalidierung vereinen * Fileupload implementieren * E-Mail versenden implementieren * Ticket und File-Validierung implementiere |
| Nicht erreichte Ziele | Es hat mir nicht gereicht, das Projekt ausführlich zu dokumentieren. Dies muss in den nächsten Tagen erledigt werden. |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Diverse Änderungen am Navigationsdesign |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | keine |
| Soll / Ist Zeitvergleich | Mit der Programmierung bin ich soweit fertig. Die Dokumentation ist aber hinter dem Zeitplan, weshalb ich in den nächsten Tagen neben dem Testen wohl noch viel Dokumentieren muss. |
| Sonstiges | Ich bin froh, dass die Applikation endlich fertig ist. |
| Arbeitszeit | 08:30 Uhr 12:30 Uhr  Mittagspause  13:30 Uhr – 18:00 Uhr |
| Visum |  |

### Mittwoch, 13. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 7) in die Dokumentation übernehmen * Testfällen erstellen * Testen * Kommentare im Code hinzufügen * Dokumentation mit Implementierungsteil erweitern. |
| Nicht erreichte Ziele | Implementierungs-Dokumentation fertigstellen. |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | Kleinere Bug fixes am Programm vor dem Testen. Diese sind im GIT-Backup erwähnt. |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme | keine |
| Unterstützung Dritter | Jonas Arab und Rolf Keller testen die Applikation für mich. |
| Soll / Ist Zeitvergleich | Die Applikation ist fertig. Die Dokumentation ist zwar laut Zeitplan hinterher, aber sicher auf dem richtigen Weg. Ich werde nicht alles ganz ausführlich dokumentieren können. |
| Sonstiges | Die Testprotokolle sind noch nicht bei der Dokumentation dabei, da diese erst nach 18:00 fertiggestellt wurden. |
| Arbeitszeit | 08:30 Uhr 12:30 Uhr  Mittagspause  13:30 Uhr – 18:00 Uhr |
| Visum |  |

### Donnerstag, 14. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 8) in die Dokumentation übernehmen * Dokumentieren * Qualitätssicherung in die Dokumentation übernehmen |
| Nicht erreichte Ziele | Code komplett auskommentieren |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | kleine |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme |  |
| Unterstützung Dritter |  |
| Soll / Ist Zeitvergleich |  |
| Sonstiges |  |
| Arbeitszeit | 08:30 Uhr 12:30 Uhr  Mittagspause  13:30 Uhr – 18:00 Uhr |
| Visum |  |

### Freitag, 15. April 2016

|  |  |
| --- | --- |
| Erreichte Ziele | * Arbeitsjournal (Tag 9) in die Dokumentation übernehmen * Arbeitsjournal (Tag 10) in die Dokumentation übernehmen |
| Nicht erreichte Ziele | keine |
| Frühzeitig erreichte Ziele | keine |
| Ungeplante Arbeiten | kleine |
| Geplante Arbeiten nach Arbeitsschluss | keine |
| Aufgetretene Probleme |  |
| Unterstützung Dritter |  |
| Soll / Ist Zeitvergleich |  |
| Sonstiges |  |
| Arbeitszeit |  |
| Visum |  |

# Projektdokumentation

## Soll / Ist vergleich

Anhand mehrerer Diagrammen wird die aktuelle Situation aufgezeigt und der gewollten Endsituation gegenübergestellt.

### Ausgangssituation

Die Grafik zeigt wie die Web-Abteilung in Kommunikation mit den anderen Nutzerinnen und Nutzer steht.

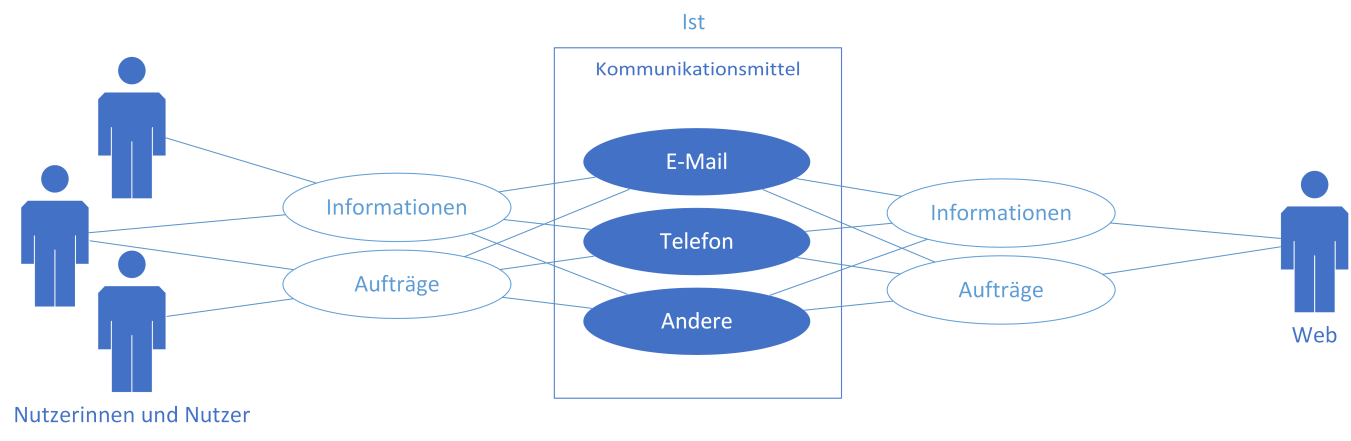


Abbildung: Aktueller Kommunikationsfluss der Nutzerinnen und Nutzer zum Web

Die Kommunikation zwischen den Nutzerinnen und Nutzer wird nicht ausschliesslich über ein Medium getätigt. Die Nutzerinnen und Nutzer kommunizieren hauptsächlich über E-Mail und Telefon mit dem Web. Das System hat den Nachteil, dass an arbeitsintensiven Tagen Informationen mit den Aufträgen vermischt werden. Da Informationen und Aufträge gleichzeitig auf allen Kommunikationsmitteln erhalten werden können, müssen diese beiden Komponenten vom Web wieder getrennt werden. Dabei kann sehr schnell die Übersicht verloren gehen.

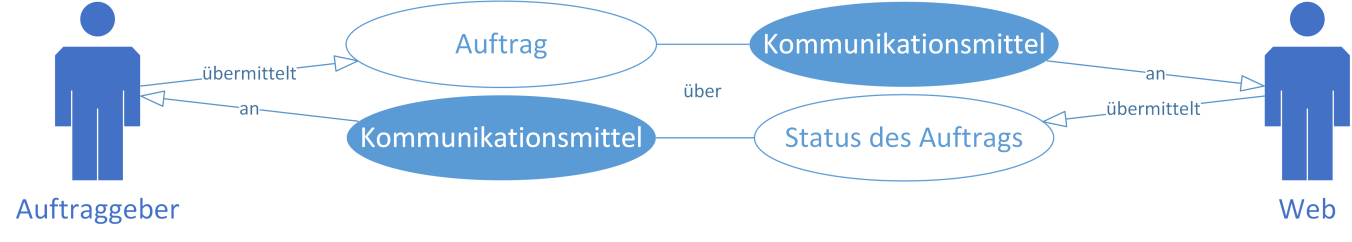


Abbildung: Kommunikation eines Auftrages

Durch den Austausch via Telefon oder E-Mail wissen immer nur die beteiligten über den Stand des Auftrages.

### Endsituation

Die folgende Grafik stellt die Situation dar, die mit der Einführung des Ticketsystems eintreten soll.

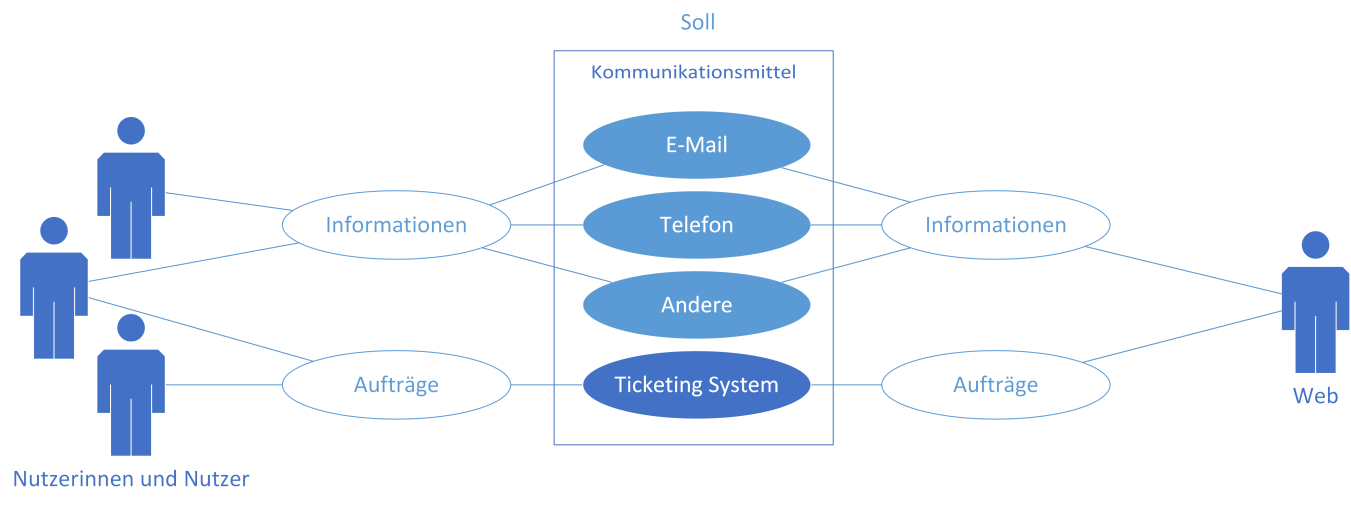


Abbildung: Aktueller Kommunikationsfluss der Nutzerinnen und Nutzer zum Web.

Wie vor der Einführung wird die Kommunikation nicht ausschliesslich über ein Medium getätigt. Die Nutzerinnen und Nutzer kommunizieren hauptsächlich über E-Mail und Telefon mit dem Web. Die Aufträge für das Web werden jedoch über das Ticketsystem kommuniziert. So werden an arbeitsintensiven Tagen die Informationen nicht mehr Aufträgen vermischt. Die Nutzerinnen und Nutzer ändern das Kommunikationsmittel für die Aufträge und erleichtern damit der Web-Abteilung die Übersicht.

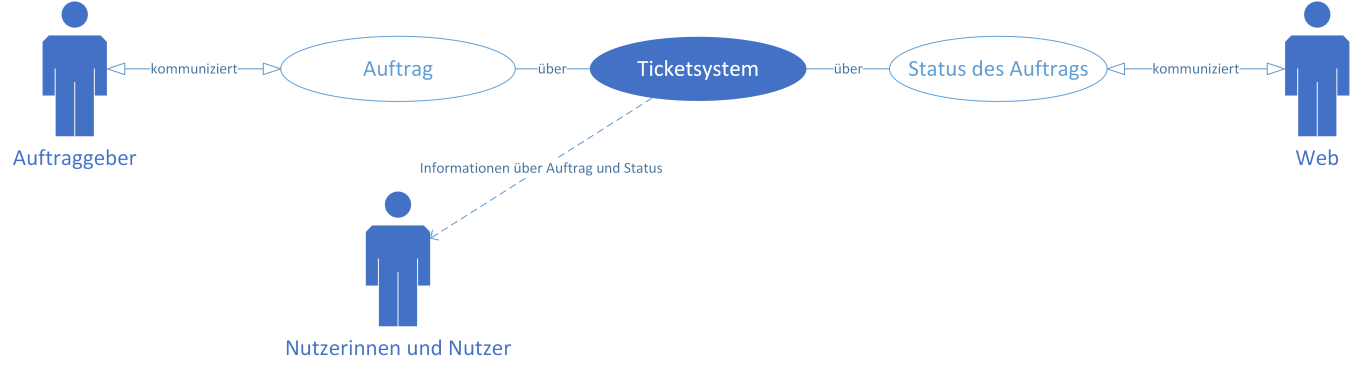


Abbildung: Kommunikation eines Auftrages nach der Einführung des Ticketsystems

Ein weiterer Vorteil des Ticketsystems ist, dass durch die Nutzung nicht nur der Auftraggeber der Stand des Auftrags weiss, sondern auch alle anderen Nutzer des Systems auf diese Informationen zugreifen können.

## Risikoanalyse

### Datenverlust

Datenverlust ist ein grosses Problem während der Arbeit. Der Verlust von Daten kann hauptsächlich durch Softwareprobleme, wie beispielsweise Absturz des Texteditors, oder durch Unachtsamkeit entstehen. Ohne entsprechende Schutzmassnahmen kann der Verlust von Daten fatale Folgen haben.

#### Wahrscheinlichkeit

Die Wahrscheinlichkeit eines Datenverlustes ist dank mehreren Back-ups sehr gering. Jedoch kann sich die Umschreibung eines Codeteiles auf die anderen Codeteile auswirken, ohne dass dies bemerkt wird. Wenn so grössere Fehler auftreten, muss auch auf eine jüngere Version zurückgesprungen werden, um das Rückverfolgen des Fehlers zu vereinfachen.

#### Folgen

Die Folgen eines Datenverlustes sind bedingt durch den Stand des Verlustes und des Standes des Backups. Bei einem grösseren Verlust kann dies zum Scheitern des Projektes führen. Falls die Auswirkungen nicht drastisch sind, wirkt es sich vor allem auf die Zeitplanung aus, da man zuerst den Stand von vor dem Datenverlust wieder erreichen muss.

#### Gegenmassnahmen

Es existiert ein USB-Stick-Back-up, ein lokales GIT-Back-up sowie ein öffentliches zugängliches Repository auf GitHub. Alle Back-ups sind immer auf demselben Stand und erhalten auf in GIT eine Versionierung. Durch das mehrfache Absichern der Dokumentation und der Applikation kann jederzeit auf ein altes Backup zurückgegriffen werden. Falls ein Harddiskfeher auftritt, kann ohne grössere Probleme an einem anderen Arbeitsplatz weiter gearbeitet werden. Um die Entwicklung zu vereinfachen, wird XAMPP benutzt.

#### Schadensminimierung

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

### Zeitprobleme

Bei einer Arbeit von knapp 10 Arbeitstagen und einer knapp bemessenen Zeit sind Zeitprobleme das grössten Problem, die während der Arbeit auftreten können. Falls unvorhergesehene Tätigkeiten oder Fehler aufkommen, führt das zu einer Veränderung des Zeitplanes und somit zur Zeitverzögerung des Projekts.

#### Wahrscheinlichkeit

Die Wahrscheinlichkeit für Zeitprobleme sind sehr hoch. Es gibt viele Fehler die bei der Zeitplanung gemacht werden können. So können für verschiedene Punkte zu viel Zeit gebraucht werden oder die bemessene Zeit für das Projekt falsch eingeteilt sein.

#### Folgen

Zeitprobleme können vom nicht korrigierten Text bis zur nicht Fertigstellung des Projektes führen. Das verpassen des Abgabetermins ist dabei auch ein Problem das auftreten kann.

#### Gegenmassnahmen

Um genügend Zeit für alles zu haben, wurde ein Zeitbuffer eingeplant. An diesem können letzte anstehende Arbeiten erledigt werden, falls diese existieren. Jedoch würde ich lieber ein nicht fertiges Produkt abgeben als den Abgabetermin zu verpassen.

#### 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 Notfallmehr arbeiten. Falls erst im Zeitbuffer Probleme auftreten, kann das fatale Folgen haben, da der Abgabetermin am selben Tag ist.

### 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 bei der Darstellung des GUI auftreten können. Die Wahrscheinlichkeit, dass etwas nicht wie gewünscht funktioniert oder aussieht, ist bei der Unterstützung von mehreren Browsern hoch.

#### Folgen

Bei Browserproblemen wird es zu Zeitverzögerungen kommen, da Browserspezifische Anpassungen implementiert werden müssen.

#### Gegenmassnahmen

Um den Problem entgegenzukommen wird das Framework Bootstrap benutzt. Dieses hat bereits viele Probleme der unterschiedlichen Browser gelöst. Somit kann der Fokus auf etwas anderes als das Styling der Webseite gelegt werden.

#### Schadensminimierung

Falls dennoch Probleme auftreten muss der Fehler ausgewertet werden. Falls der Fehler nur optisch ist und nicht die Funktionalität der Applikation einschränkt, kann dieser Fehler in der Bufferzeit oder einer nächsten Version der Applikation bearbeitet werden. Falls das Problem die Funktionalität einschränkt, muss schnell gehandelt werden und das Problem manuell mit Hilfe von Dritter und dem Internet gelöst werden.

## Versionierung

Für die Versionierung des GIT-Repositorys wurde ein klassisches System mit Hauptversionsnummer, Nebenversionsnummer, Revisionsnummer und Buildnummer mit ergänzender Informationen 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, erweitert mit dem Softwarestadium

### 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, können diese unter den Änderungen aufgeführt werden.

## Wieso Bootstrap

### Was ist Bootstrag

Bootstrap ist das meist genutzte Front-End Framework für Webentwickler, welches optimal für responsive, Mobile-first Webseiten genutzt werden kann. Viele Webseiten heutzutage benutzen dieses Framework. Die Meinungen zum Framework sind von Person zu Person sehr verschieden. Die Nutzung des Frameworks bringt viele Vor- und Nachteile mit sich, weshalb Entwickler komplett andere Meinungen dazu haben können

Um die Entscheidung Bootstrap zu benutzen besser aufzuzeigen, sind hier die grössten Vor- und Nachteile von Bootstrap:

### Vorteile

Bootstrap

* erleichtert und beschleunigt die Entwicklung
* erlaubt schnelles Prototyping.
* funktioniert in allen modernen Browsern
* unterstützt JavaScript und jQuery
* arbeitet mit eigener Gitter-Struktur
* ist Mobile-First
* löst viele CSS Probleme von Anfang an
* benötigt nicht viel custom CSS
* ist Open Source

### Nachteile

* Durch das vordefinierte CSS des Frameworks, sehen viele Webseiten durch kaum eigenes Styling gleich aus.
* Bootstrap legt den Schwerpunkt nicht auf das Design, sondern auf die Entwicklung.
* Es ist nicht einfach, der Webseite mit CSS ein anderes Design zu geben.
* Die Bootstrap Struktur bzw. Logik ist überladen.
* Bootstrap zu aktualisieren ist sehr mühsam, da sich immer sehr viel der Logik verändert. Oft wird die Version von Bootstrap, mit der die Applikation erstellt wurde die finale Bootstrap-Version.

## Wieso Bootstrap benutz wird

Bootstrap 3 bietet mit dem bereits vorhandenen CSS und dem Gitter-System eine vorhandene Struktur, an welcher man nicht viel arbeiten muss. Da Bootstrap sehr gut Dokumentiert ist, ist es ein leichtes, die Webseite innerhalb des Frameworks so auszusehen zu lassen wie man möchte, ohne viel eigenes CSS zu schreiben. Das spart bei der Entwicklung sehr viel Arbeit und vor allem Zeit. Das Bootstrap Webseitendesign wird mittlerweile von vielen Webseiten benutz. Deshalb sind viele Nutzerinnen und Nutzer bereits an die Darstellunsweise einer Bootstrap-Seite gewöhnt.

# Design

## Datenbank

### Vor der Implementierung

Vor der Implementierung wurde eine Datenbank erstellt, welche die Applikation für das Speichern von Tickets und Benutzer benötigt.

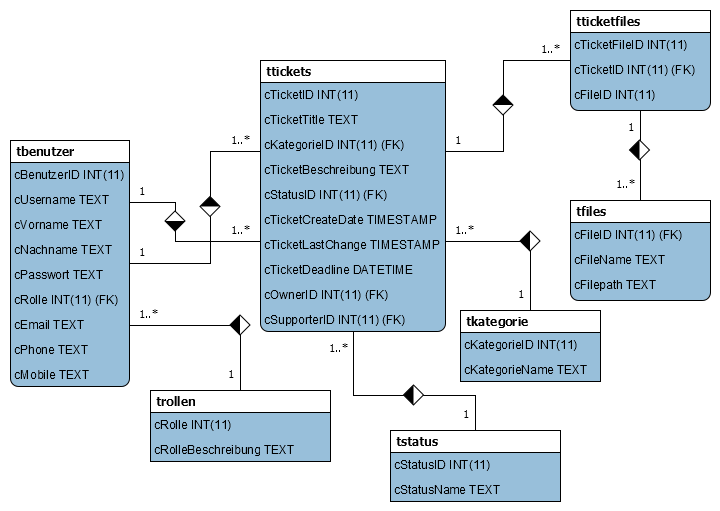


Abbildung: Datenbank vor der Erstellung der Applikation

Die Datenbank besteht aus sieben Tabellen. Die Tabelle „tbenutzer“ inklusive „trollen“ wird für die Identifikation der Nutzerinnen und Nutzer verwendet. Die Tabelle „ttickets“ enthält alle gespeicherten Tickets mit ihrem Status, welcher aus der Tabelle „tstatus“ genommen wird, der Kategorie, welche aus der Tabelle „tkategorien“ genommen wird, und die hochgeladen Dateien, welche über den Pfad in der Tabelle „tfiles“ referenziert werden. Diese gespeicherten Files sind durch die Verbindungstabelle „tticketfiles“ mit der „tticket“ Tabelle verbunden.

Um die Datenbank übersichtlich zu machen, wurde vorerst für jeden Eintrag in der Datenbank mit Text den Feldtyp „Text“ verwendet. Alle Zahlen haben den Feldtyp „int(11)“ erhalten. Das Datum des Tickets wir mittels dem Feldtyp „TIMESTAMP“ und „DATETIME“ gespeichert.

### Nach der Erstellung der Applikation

Die Datenbank hat sich während der Entwicklung der Applikation verändert. Die meisten „Text“-Feldtypen wurden zu „vachar“ umgeändert, da der„Text“-Feldtyp eigentlich für Text und nicht für einen Vorname oder Nachnamen angedacht ist. Zudem ist der Typ „varchar“ perfomanter.

Neben den Änderungen in den Feldtypen hat die Benutzertabelle ein weiteres Feld erhalten. Das Feld „cAktiv“ wurde hinzugefügt, um Benutzer ausblenden zu können. Dieses Feature wurde nach dem ersten Gespräch mit den Experten implementiert. Es wird benötigt, damit man erstellte Benutzer nicht löschen muss. Ein deaktivierter user kann sich nicht mehr einloggen.

Die veränderte Datenbank hat die folgende Struktur:

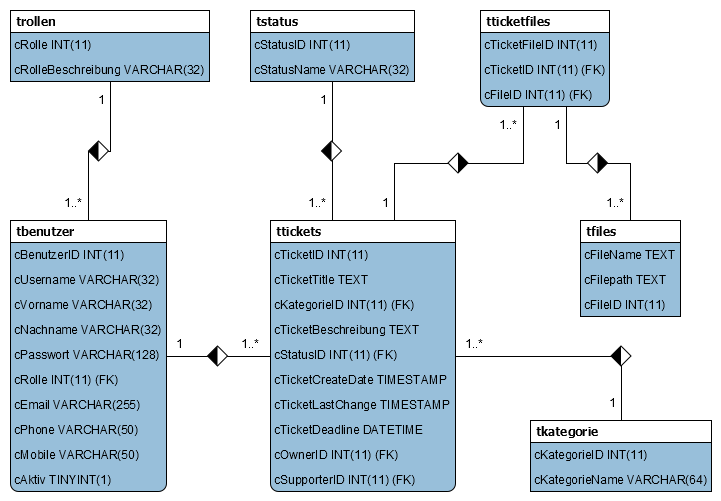


Abbildung: Neue Datenbankstruktur

Um die Darstellung zu vereinfachen, wurde die zweite Verbindung von der „ttickets“ Tabelle zu der „tbenutzer“ Tabelle ausgeblendet.

# Implementierung

## Standard GUI

Zuerst wurde das Grundgerüst der Webseite erstellt. Dieses besteht aus dem Header der Webseite, der Webseiten-Navigation und dem Footer.

### header.inc.php

Die Headerdatei startet die Html-Struktur mit dem Webseiten Icon sowie den links zu den CSS und JS Dateien, die für alle Seiten verfügbar sein müssen. Der Restliche Header wird durch die Funktion getHeader() angefügt. Diese determiniert aufgrund einer Variable, welche mit jeder Seite mitgegeben wird, welcher zusätzliche Header geladen werden müssen. Da nicht jede Seite jede Header benötigt, wurde dies so gelöst. Im Header wird das Bootstrap Framework geladen, sowohl als auch das Bootstrap 3 Plugin „Datepicker“ von Eonasdan.

### navigation.inc.php

Die Webseitennavigation gibt es in drei Varianten. Ein eingeloggter Benutzer erhält die Standardnavigation mit der Ticketübersicht, der Funktion „Ticket erfassen“, dem Benutzerhandbuch, dem Nutzerprofil und dem Log-Out. Ein eingeloggter Administrator erhält eine Navigation mit zusätzlichem Usermanager. Alle Seiten die etwas mit Login oder Berechtigungen zu tun haben, erhalten die dritte Navigation. Diese beinhaltet nur den Link zu dem Benutzerhandbuch. Die Navigation wird durch eine Variabel bestimmt, die in der PHP-Datei checkauth.inc.php festgelegt wird.

### footer.inc.php

Der Footer beinhaltet den Schluss der Html Struktur mit einer Copyright und Datumsangabe.

## Autorisierungs-Seiten

Die Ticketingapplikation beinhaltet zwei Authoritätschecks. Einen für den Zugang zum Ticketsystem selbst, sowohl einen für die Administratorenberechtigungen.

### checkauth.inc.php

Diese Prüfung wird auf jeder Ticketinginternen Seite durchgeführt. Es wird geprüft, ob der User, der auf die Ticketingapplikation zugreiffen möchte, eingeloggt ist. Falls dies nicht der Fall ist, wird diesem ein Hinweis auf das Einloggen angezeigt, die eingegebene URL kopiert und auf die Loginseite weitergeleitet. Die Kopierte URL wird zudem In die login URL kopiert, um nach dem erfolgreichen einloggen des Userers automatisch die vorher aufgerufene Seite anzuzeigen.

### checkrole.inc.php

Die checkrole.inc.php prüft die Session des eingeloggten Benutzers ab. Ist der User kein Administrator bzw. Supporter, wird Ihm der Zugriff auf die hinter dem Check versteckten Funktionen verweigert. Der Benutzer wird auf die zu niedrigen Berechtigungen aufmerksam gemacht, und zu der Ticketübersicht weitergeleitet.

## Loginseiten

Die Ticketapplikation verfügt über eine Log-In und eine Log-Out Seite.

### login.php

### logout.php

Die Logout-Seite ist für das Abmelden des eingeloggten Benutzers. Beim Aufruf der Seite wird die Funktion logOut() aufgerufen, die sich im functioncontroller.inc.php befindet.

logOut()

Die Logout Funktion beendet die aktive Session des Nutzers und löscht das vom Browser erstellte Cookie. Danach wird der Benutzer auf die Loginseite weitergeleitet.

## Datenbank Verbindungsseiten

Die Applikation enthält zwei verschiedene Datenanbindungen. Eine über MySQLi, und eine über PDO. Da das Projekt ursprünglich komplett Prozedural implementiert werden sollte, wurde der erste Teil in MySQLi umgesetzt. Nach dem ersten Gespräch mit den Experten wurde entschieden, für den teil der Applikation welche noch nicht implementiert ist, PDO zu verwenden, da dies die Implementation sehr stark erleichtert und beschleunigt.

### dbsettings.inc.php

Enthält die MySQLi Daternbankinformationen, die für die MySQLi Verbindung benötigt werden

### dbconnection.inc.php

Hier wird die MySQLi Datenbankverbindung hergestellt. Falls diese nicht zustanden kommen kann, wird eine gut sichtbare Fehlermeldung und die Login-Navigation angezeigt.

### pdosettings.inc.php

Enthält die PDO Daternbankinformationen, die für die PDo Verbindung benötigt werden

### pdoclass.inc.php

Die pdoclass.inc.php beinhaltet eine von John Richardson erstellte PDO-Klasse, die die Handhabung der PDO-Verbindung einiges erleichtert. Umgeschrieben wurde das Errorhandling, die Formatierung und Kommentierung wurde dem Styl der weiteren Codeteilen angepasst.

## Funktionsseiten

Um Funktionen einfacher im Code zu finden, wurden diese in separate PHP-Files verlagert. Es existieren zwei Funktionsfiles.

### functioncontroller.inc.php

Hier befinden sich fast alle Funktionen, die mehrmals in der Applikation verwendet werden. Es handeln sich dabei um Funktionen, die benötigt werden, um Daten aus der Datenbankn zu laden, oder Benutzer und Tickets zu Validieren. Eine Ausnahme bildet die E-Mail-Versandsfunktion.

### sendmail.inc.php

In dieser PHP Datei wird der E-Mail-versand abgewickelt. Dies wir über die Funktion sendMail() gemacht.

sendMail()

Die E-Mail-Funktion ist für den Versand der E-Mails in der gesamten Applikation verantwortlich. Der Funktion werden als Parameter die Ticket-Informationen sowie der Typ des E-Mails und den Empfänger mitgegeben. Jedes E-Mail erhält Standartinformationen, wie den Titel des E-Mails. Aufgrund des E-Mailtyp-Parameters wird dann die Art des E-Mails definiert. Diese bestimmt dann den weiteren Inhalt des Emails und schreibt diese in variablen. Das ganze E-Mail wird zum Schluss in einer Variabel zusammengeführt, und versendet.

Das E-Mail besteht dabei nur aus HTML-Tags und Variablen.

/\*\*

\* build the whole email from variables

\*/

$nachricht = '

<html>

'.$mailHead.'

<body>

'.$mailtTitel.'

'.$mailText.'

'.$mailLink.'

<table>

'.$mailTableTitle.'

'.$mailTableDefault.'

'.$mailTable.'

'.$mailTableAction.'

</table>

'.$mailFooter.'

</body>

</html>';

Ein E-Mail sieht nach dem Versenden so aus:

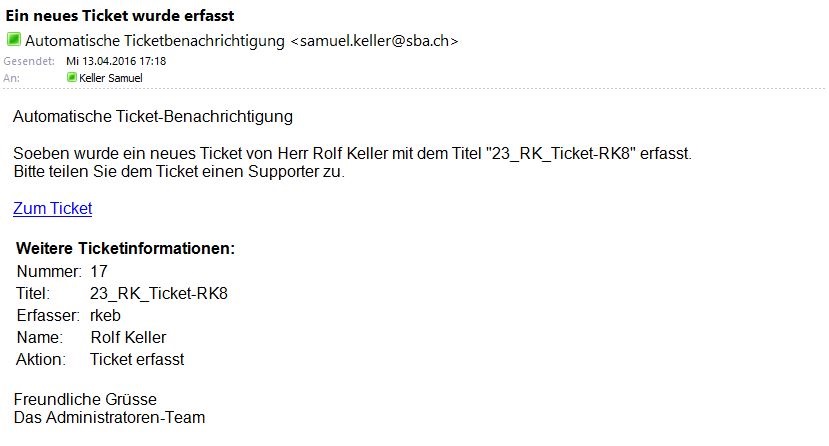


Abbildung: Ein E-Mail, das während der Qualitätssicherung versendet wurde.

## Ticketverwaltungsseiten

### index.php

### ticketfilter.inc.php

### ticketliste.inc.php

### neuesticket.php

### ticket\_details.php

## Userverwaltung

### profil.php

### usermanagement.php

### neuernutzer.php

### benutzer\_details.php

# Testing

# Abschlussbericht

## Rückblick

## Lessons learned & Was ist gut gelaufen

## Was schief gelaufen ist

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

Hier zeige ich nennenswerte Funktionen der Applikation auf, die ich gerne bei einer Version 2.0 hinzufügen würde

### Tickets und Benutzer in der Detailübersicht

Momentan werden die Benutzer per ID in der URL geladen. Diese möchte ich per Hash ansprechen. Dieser Hash wird beim Erstellen des Nutzers oder des Tickets generiert und ist zufallsbasiert. Diese zusätzliche Funktion hat Änderungen in der Datenbank und am Code zufolge. Die Datenbank erhält für jeweils den Benutzer und die Tickets ein weiteres Feld mit dem zufälligen Hash. Der Hash wird beim Erstellen vor dem speichern des Tickets / Nutzers in die Datenbank generiert. Nach dem generieren des Hashes wird gecheckt, ob dieser schon benutzt wird, falls nicht, wird das Ticket / der Benutzer mit diesem Hash abgespeichert. Der Hash wird für das aufrufen in der Detailansicht verwendet.

### Benutzerfilter und/oder Ansicht für deaktivierte Nutzer

Beim Deaktivieren eines Nutzers wird dieser nicht mehr in der Nutzerübersicht angezeigt und kann nur noch per Datenbank verändert werden. Mit einer zusätzlichen Ansicht und / oder Filterfunktion für die Benutzer wie diese bei den Tickets kann dies umgehen werden.

### Passwort ändern Funktion auf eine eigene Seite auslagern

Ich möchte die Passwort ändern funktion auf eine eigene Seite auslagern.

## 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

# Glossar

Administrator 31

CSS 31

Footer 31

Framework 28

Front-End 28

GIT 25

GitHub 25

GUI 10

Header 31

Html 31

IPERKA 10

JS 31

Log-Out 31

Mobile-first 28

PHP-Datei 31

Prototyping 28

Repository 25

responsive 28

Updaten 28

Wasserfall-Modell 10

XAMPP 25

# Anhang

## Testprotokolle

## Quellcode