Realisierungsbericht

|  |  |
| --- | --- |
| **Status** | Abgeschlossen |
| **Projektname** | RentAFlat |
| **Projektleiter** | Victor Hargrave |
| **Auftraggeber** | Andreas Scheidegger |
| **Autoren** | Lars Glatthard, Luca Berger, Victor Hargrave, Felix Neidhart |
| **Verteiler** | Felix Neidhart |

**Änderungskontrolle, Prüfung, Genehmigung**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Datum | Beschreibung, Bemerkung | Name oder Rolle |
| 0.1 | 05.05.15 | Initialversion | Felix Neidhart |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Definitionen und Abkürzungen**

|  |  |
| --- | --- |
| Begriff / Abkürzung | Bedeutung |
|  |  |

**Referenzen**

|  |  |
| --- | --- |
| Referenz | Titel, Quelle |
| [1] |  |
| [2] |  |
| [3] |  |

**Inhaltsverzeichnis**

[1 Zusammenfassung (Felix) 3](#_Toc418584721)

[2 Technische Detailspezifikation (Vic, Luca) 3](#_Toc418584722)

[2.1 Systemdesign 3](#_Toc418584723)

[2.1.1 Struktur 3](#_Toc418584724)

[2.1.2 Dynamik 3](#_Toc418584725)

[2.2 Schnittstellendefinitionen (Luca) 4](#_Toc418584726)

[2.3 Datenmodell (Lars) 4](#_Toc418584727)

[3 Systemdokumentation (Felix) 4](#_Toc418584728)

[3.1 Anwendungshandbuch (Felix) 4](#_Toc418584730)

[3.1.1 Übersicht 4](#_Toc418584731)

[3.1.2 Funktionen und Detailbeschreibung 4](#_Toc418584732)

[3.1.3 Fehlerbehandlung 4](#_Toc418584733)

[3.2 Integrations- und Installationshandbuch (Weglassen, Erklärung Felix) 5](#_Toc418584734)

[3.3 Betriebshandbuch (Weglassen, Erklärung Felix) 5](#_Toc418584735)

[4 Systemtest (Vic, Lars) 5](#_Toc418584736)

[4.1 Unit-Test 5](#_Toc418584738)

[4.2 Systemtest 5](#_Toc418584739)

[4.2.1 Testfälle 5](#_Toc418584740)

[*Testfall 1 „Neuen Benutzer registrieren und anmelden“* 6](#_Toc418584741)

[*Testfall 2 „Bestehenden Benutzer registrieren“* 6](#_Toc418584742)

[5 Testprotokoll (Vic, Lars) 7](#_Toc418584743)

[*5.1* *Systemtest 1* 7](#_Toc418584744)

[*5.1.1* *Testfall 1 „Neuen Benutzer registrieren und anmelden“* 7](#_Toc418584745)

[*5.1.2* *Testfall 2 „Bestehenden Benutzer registrieren“* 7](#_Toc418584746)

[6 Weiterführung der Projektplanung (Felix) 7](#_Toc418584747)

[6.1 Abgleich von Planung und tatsächlichem Verlauf der Phase Konzept 7](#_Toc418584748)

[6.2 Aktualisierung der Risikosituation 7](#_Toc418584749)

[6.3 Planung der nächsten Phase 7](#_Toc418584750)

[Anhang A: Quellcode 8](#_Toc418584751)

[Anhang B: Testcode Unit-Tests 8](#_Toc418584752)

**Abbildungsverzeichnis**

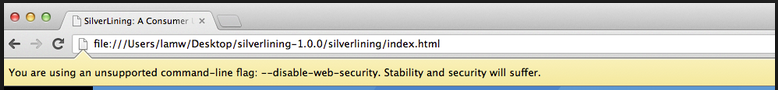
# Zusammenfassung (Felix)

Geben Sie hier eine kurze Zusammenfassung des Inhalts dieses Dokumentes. Wozu dient das Dokument (Zweck) und welche Informationen enthält es?

**Hinweis:**

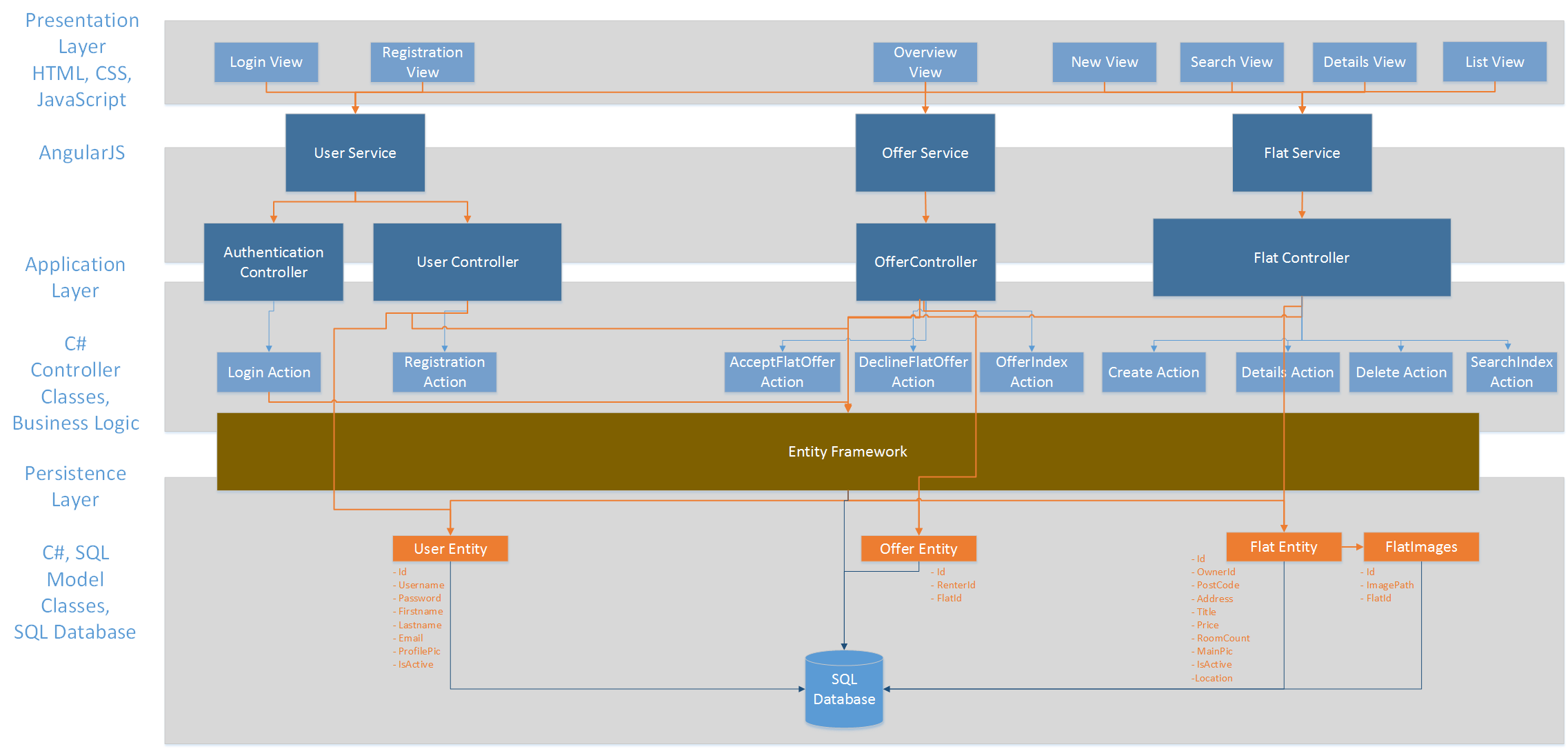
Unglücklicherweise erhalten wir von allen Browsern die Message „Same-origin Policy wurde verletzt.“  
Dies geschieht, weil der Server auf einem anderen Port läuft als der Client.  
Die Folge davon ist, dass der Browser die Requests zum Server nicht abschickt, sondern blockiert.

Als Workaround, kann man im Chrome diese Einstellung verhindern:

1. Navigieren sie im cmd zu folgendem Pfad:  
   „C:\Program Files (x86)\Google\Chrome\Application“
2. Geben sie im cmd folgenden Befehl ein:  
   „chrome.exe --disable-web-security“
3. Warten sie bis der Browser folgende Nachricht anzeigt:  
   
4. Navigieren sie anschliessend in **diesem Tab** auf die Seite RentAFlat!

# Technische Detailspezifikation (Vic, Luca)

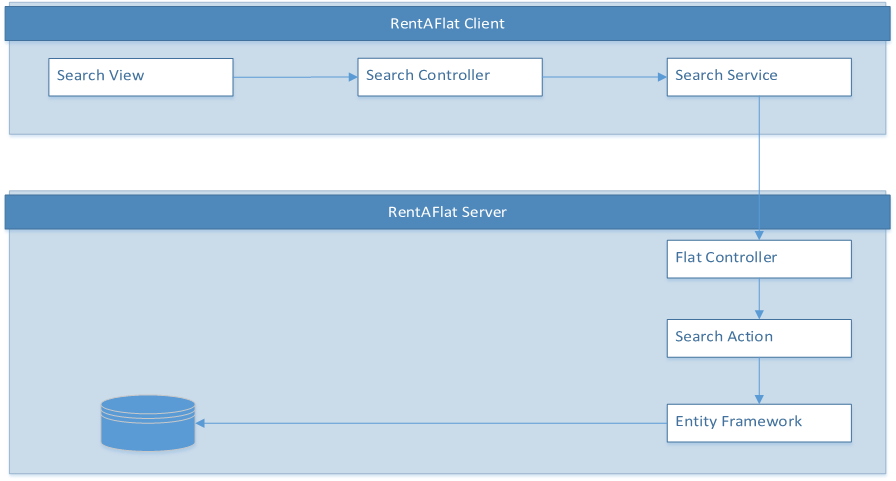
## Systemdesign



### Struktur

Der Benutzer öffnet den Browser und navigiert zur Website „RentAFlat“. Die Seite besteht aus HTML, CSS und AngularJS (JavaScript Framework). Wenn der Benutzer ein Request zum Server senden möchte, kann er dies über ein Suchfeld machen. (Z. B. das Suchen nach einer Wohnung). Von der „Search“-View wird eine Anfrage zum Controller geschickt, der dies danach zum Service weiterleitet. Dieser Service kümmert sich danach um die Verbindung zum Server und nimmt die gegebenen Parameter entgegen.   
Sofern der Service alle Parameter beisammen hat, sendet er ein http-Request zum Server. Dieser kümmert sich dann um die Authentifizierung, sofern diese benötigt ist und schickt anschliessend die Daten im JSON Format an den AngularJS Service zurück.

### Dynamik



## Schnittstellendefinitionen (Luca)

In unserem Projekt haben wir zwei Schnittstellen. Das User Interface, wo dem Benutzer die Möglichkeit gegeben wird, mit dem Tool zu interagieren. Hierbei gibt es keine Regeln.

Die zweite Schnittstelle ist ein von uns programmierter Webservice, der die Requests vom Client entgegen nimmt und dann eine Antwort zurücksendet. Wenn der Service nicht erreichbar ist, wird dem Benutzer eine Meldung auf dem GUI angezeigt.

Die Meldungen und die Daten sind immer im JSON Format und werden als Parameter im http-Request, resp. In der http-Response mitgeschickt. Falls die Daten vom Client nicht korrekt kommen, wird der Webservice nicht aufgerufen und es wird eine Meldung im GUI angezeigt.

## Datenmodell (Lars)

Wenn Sie in Ihrem System eine Datenbank haben, fügen Sie hier eine ERD ein. Beschreiben Sie die einzelnen Tabellen wo nötig kurz. Aus dem ERD müssen die Primary- und Foreign-Keys, sowie die Multiplizitäten hervorgehen

Wenn Sie objektorientiert arbeiten und ein Mapping-Framework (z.B. JPA/Hibernate oder ADO.Net Entity-Framework) verwenden, können Sie auch ein Klassendiagramm mit den Entitätsklassen einfügen. Auch hier beschreiben Sie die einzelnen Klassen, wo nötig, kurz.

# Systemdokumentation (Felix)

Hermes 5 definiert unter anderem das Anwendungshandbuch, die Integrations- und Installationsanleitung, sowie das Betriebshandbuch als Resultate der Realisierungsphase. Hier werden diese Ergebnisse unter dem Oberbegriff Systemdokumentation zusammengefasst.



## Anwendungshandbuch (Felix)

Das Anwendungshandbuch richtet sich an den Anwender oder Benutzer des Systems. Es „enthält alle Informationen, die der Anwender eines Produkts / Systems braucht, um es ordnungsgemäss zu bedienen und im Fall von Problemen richtig reagieren zu können“ (siehe HERMES 5 Handbuch Seite 127). Dazu gehören Übersicht, Funktionen, Detailbeschreibungen zur Anwendung und Fehlerbehandlung.

Denken Sie daran, dass die Anwender in der Regel keine IT-Fachpersonen sind. Entsprechend muss das Anwendungshandbuch in einer für das Zielpublikum verständlichen Sprache geschrieben sein.

### Übersicht

Hier folgt ein kurzer Überblick des Systems für die Anwender. Beschreiben Sie den Zweck und die Ziele des Systems.

### Funktionen und Detailbeschreibung

Beschreiben Sie hier die einzelnen Funktionen des Systems soweit, dass der Anwender diese erfolgreich anwenden kann. Stellen Sie sich also die Frage: „Was muss dem Anwender gesagt werden, damit er das System erfolgreich benutzen kann?“

Wenn Sie mit Anwendungsfällen gearbeitet haben, können Sie diese als Basis verwenden. Arbeiten Sie mit Screenshots. Denken Sie auch daran, dass es unter Umständen verschiedene Anwender gibt, wie registrierte Benutzer, Administratoren, etc.

### Fehlerbehandlung

Listen Sie hier mögliche Fehler, mit denen der Anwender konfrontierte werden kann auf und beschreiben Sie, was mögliche Gründe sein können und wie beim Auftraten dieser Fehler zu verfahren ist.

## Integrations- und Installationshandbuch (Weglassen, Erklärung Felix)

Fügen Sie hier eine Installationsanleitung für Ihr System ein. Je nach Art des Systems, muss eine unabhängige Fachperson oder gar der Anwender selber in der Lage sein, mit dieser Anleitung Ihr System erfolgreich zu installieren. Beschreiben Sie die einzelnen Schritte, allfällige Konfigurationsdaten und Tests zur Überprüfung, ob die Installation erfolgreich war.

## Betriebshandbuch (Weglassen, Erklärung Felix)

Wird der Betrieb des Systems nicht durch den Anwender selbst wahrgenommen, so braucht es ein Betriebshandbuch, welches „alle Informationen, die der Betreiber benötigt, um das

System ordnungsgemäss betreiben und im Fall von Problemen richtig reagieren zu können“ enthält (siehe HERMES 5 Handbuch Seite 128). Dazu gehören im Minimum Angaben über den Start / Neustart des Systems, Überwachung des Systems (z.B. Logs), Sicherung des Systems (z.B. Backup) sowie zu ergreifende Massnahmen bei Fehlern

# Systemtest (Vic, Lars)

Im Modul 226 haben Sie die verschiedenen Teststufen und Vorgehensweisen zum Testen kennengelernt. Zur Repetition: Das Ziel beim Testen ist es, Fehler zu finden. Es geht nicht darum zu zeigen, dass das System keine Fehler hat.

Getestet wird auf verschiedenen Ebenen vom Einzeltest bis zum Abnahmetest (Akzeptanztest beim Kunden). Wir beschränken uns auf die beiden Ebenen Einzeltest (Unit-Test) und Systemtest/Abnahmetest.

Tests müssen reproduzierbar sein. Die Ausgangslage, der Ablauf und die erwarteten Resultate eines jeden Testfalls müssen detailliert beschrieben sein. Dies ist der Zweck der Testspezifikation.



## Unit-Test

Auf Unit Tests haben wir verzichtet, da wir unter grossem Zeitdruck standen und das Produkt im Vordergrund steht.

## Systemtest

Auf System Tests haben wir verzichtet, da wir unter grossem Zeitdruck standen und das Produkt im Vordergrund steht.

### Testfälle

Nachfolgend beschreiben Sie die einzelnen Testfälle für den Systemtest Ihres Systems. Als Ausgangspunkt nehmen Sie die Ziele aus der Voranalyse und die Anwendungsfälle aus Ihrem Konzept. Jeder Anwendungsfall führt jetzt zu mindestens einem separaten Testfall.

Das Ziel ist die vollständige Abdeckung der Funktionalität Ihres Systems durch die Testfälle. Es wird deshalb nötig sein auch weitere Testfälle zu beschreiben.

Spezifizieren Sie jeden Testfall separat mit einer Tabelle.

Testfall 4 „Wohnung erfassen“

|  |  |  |
| --- | --- | --- |
| **Beschreibung** | Ein eingeloggter Benutzer erfasst eine neue Wohnung im System. | |
| **Abgedeckte Anwendungsfälle** | Als Benutzer möchte ich eine Wohnung ausschreiben können, um die Wohnung zu vermieten. | |
| **Ausgangssituation** | Der Benutzer ist eingeloggt | |
| **Vorbereitungsschritte** | Der Benutzer hat sich eingeloggt und ist auf der Flat „My Flats“ Page | |
| **Testschritte** | | **Erwartetes Resultat** |
| Der Benutzer klickt auf dem Button „Wohnung erfassen“ | | Der Benutzer wird auf er /Flat/Create Page weitergeleitet |
|  | | Das Wohnungerfassen Formular wird angezeigt. |
| Der Benutzer gibt folgende Daten ein :  Post Code: 1111  Address: Bernstrasse 59  Title: Cheap Wohnung  Price: 400$ / Night  Room Count: 3.5  Main Pic: none  Location: Bern | |  |
| Den Button „Speichern“ wählen | | Der Benutzer wird auf der „My Flats“ Page weitergeleitet und sieht seine neu erfasste Wohnung |
| … | | … |
| … | | In der Datenbank ist in der Tabelle Flats ein neuer Record mit dem Namen Cheap Wohnung vorhanden. Überprüfung mit SQL Abfrage in VS. |

Testfall 5 „Wohnung löschen“

|  |  |  |
| --- | --- | --- |
| **Beschreibung** | Ein eingeloggter Benutzer löscht eine von seiner Wohnungen im System. | |
| **Abgedeckte Anwendungsfälle** | Als Benutzer möchte ich meine Wohnungsinserate löschen können, damit sie nicht mehr im System gefunden werden. | |
| **Ausgangssituation** | Der Benutzer ist eingeloggt  Testfall 4 wurde ausgeführt | |
| **Vorbereitungsschritte** | Der Benutzer hat sich eingeloggt und ist auf der Flat „My Flats“ Page | |
| **Testschritte** | | **Erwartetes Resultat** |
| Der Benutzer klickt auf dem Link „Löschen“ beim Eintrag mit dem Namen „Cheap Wohnung“ | | Das System fragt ob der Benutzer den Eintrag wirklich löschen will |
| Der Benutzer klickt auf ja | | Die „My Flats“ Page wird neu geladen und den Eintrag „Cheap Wohnung“ wird nicht mehr gefunden |
| … | | In der Datenbank ist in der Tabelle Flats kein Record mehr mit dem Namen „Cheap Wohnung“ vorhanden. |

Testfall 6 „Mich für eine Wohnung bewerben“

|  |  |  |
| --- | --- | --- |
| **Beschreibung** | Ein eingeloggter Benutzer möchte ich mich für eine Wohnung bewerben | |
| **Abgedeckte Anwendungsfälle** | Als Benutzer möchte ich mich für eine Wohnung bewerben, damit ich die Wohnung mieten kann. | |
| **Ausgangssituation** | Der Benutzer ist eingeloggt | |
| **Vorbereitungsschritte** | Der Benutzer hat sich eingeloggt und ist auf der Flat „Details“ Page von einer Wohnung | |
| **Testschritte** | | **Erwartetes Resultat** |
| Der Benutzer klickt auf dem Button „mich bewerben“ | | Ein kleines Offer Formular öffnet sich |
| Der Benutzer gibt folgende Daten ein :  From : 02.11.2015  Until : 04.11.2015 | |  |
| Der Benutzer klickt auf auf dem Button „Speichern“ | | Die Page lädt sich neu. |
|  | | Ein Offer würde am Vermieter der Wohnung geschickt |
| … | | In der Datenbank ist in der Tabelle Flats ein neuer Offer für die Wohnung vorhanden. Überprüfung mit SQL Abfrage in VS. |

# Testprotokoll (Vic, Lars)

Das Testprotokoll gibt darüber Auskunft, wer, wann, was getestet hat und welches die Befunde des betreffenden Testdurchlaufes waren. Führen Sie die einzelnen Testdurchläufe in separaten Unterkapiteln auf.

Systemtest 1

*Getestete Version: 0.5, Iteration 1*

*Tester: Victor Hargrave*

*Datum, Zeit: 10.5.2015, 14.00 – 15.30*

Testfall 4 „Wohnung erfassen“

|  |  |  |
| --- | --- | --- |
| ***Testschritt*** | ***Erfüllt*** | ***Bemerkung*** |
| *1* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
| *2* | 🗷 | *-* |
| *3* | 🗷 | *-* |
|  |  | *Ganzer Testfall nicht erfüllt.* |

Testfall 5 „Wohnung löschen“

|  |  |  |
| --- | --- | --- |
| ***Testschritt*** | ***Erfüllt*** | ***Bemerkung*** |
| *1* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
| *2* | 🗷 | *-* |
|  |  | *Ganzer Testfall nicht erfüllt.* |

Testfall 6 „Mich für eine Wohnung bewerben“

|  |  |  |
| --- | --- | --- |
| ***Testschritt*** | ***Erfüllt*** | ***Bemerkung*** |
| *1* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
| *2* | 🗷 | *-* |
| *3* | 🗷 | *-* |
|  |  | *Ganzer Testfall nicht erfüllt.* |

|  |  |  |
| --- | --- | --- |
| ***Testschritt*** | ***Erfüllt*** | ***Bemerkung*** |
| *1* | 🗹 |  |
| *2* | 🗹 |  |
| *3* | 🗹 |  |
| *4* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
| *5* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
| *6* | 🗷 | *Die Pages sind bis jetzt gar nicht erstellt worden.* |
|  |  | *Ganzer Testfall nicht erfüllt.* |

# Weiterführung der Projektplanung (Felix)

## Abgleich von Planung und tatsächlichem Verlauf der Phase Konzept

Soll/Ist-Vergleich bezüglich der Zeitplanung für die Phase Realisierung.

Welche Risiken sind während der Phase Realisierung Realität geworden? Wie wurde diese Situation gemeistert? Welche Konsequenzen ergeben sich daraus für das gesamte Projekt?

## Aktualisierung der Risikosituation

Haben sich an der Risikosituation Änderungen (Verschärfung oder Entschärfung, neue Risiken) ergeben? Dann beschreiben Sie hier die Risikosituation neu. Führen Sie wiederum konkrete Massnahmen auf, wie Sie den Risiken in der nächsten Phase begegnen wollen.

## Planung der nächsten Phase

Aktualisieren und verfeinern Sie hier Ihren Projektplan, vor allem in Bezug auf die nächste Phase.

# Anhang A: Quellcode

Fügen Sie zuerst eine **Übersicht mit allen Dateien** des Codes ein. Zum Code gehören auch HTML, CSS, JS, JSP, SQL, etc.

Anschliessend folgt der Code.

**Unbedingt beachten**:

* Formatierung und Einrückung
* Neue Seite bei neuer Datei

# Anhang B: Testcode Unit-Tests

Da wir keine Unit Test gemacht haben, gibt es keinen Code dazu.

# Arbeitsjournale

|  |  |  |  |
| --- | --- | --- | --- |
| **Arbeitsjournal Felix Neidhart**  **Datum: 27.01.2015** | | | |
| **Tatsächlicher Zeitbedarf**  **Personen Lektionen (PL)** | **Geplanter Zeitbedarf** | **Beschreibung der Arbeit** | **Bemerkungen, Probleme, genutzte Hilfestellungen** |
| 16PL | 16PL | Projektinitialisierungsauftrag fertig stellen. Die Projektidee wird ausformuliert und im Projektinitialisierungsauftrag schriftlich festgehalten. | Erste Lektion im Projekt.  Probleme entstanden keine.  Fragen sind bei diversen Pflichtpunkten in der Vorlage aufgetaucht. Diese konnten jedoch von der Lehrperson geklärt werden. |
| **Reflexion des Arbeitstages**  Der Projektinitialisierungsauftrag konnte mit den zur Verfügung stehenden Ressourcen fertig gestellt werden. | | | |
| **Pendenzen für den nächsten Arbeitstag**  Beginn Initialisierungsphase:   * Studie * Projektplan | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Arbeitsjournal Luca Berger**  **Datum: 28. April 2015** | | | |
| **Tatsächlicher Zeitbedarf** | **Geplanter Zeitbedarf** | **Beschreibung der Arbeit** | **Bemerkungen, Probleme, genutzte Hilfestellungen** |
| 1/4 PL | 1/2 PL | Route Config im Client implementiert. | Die Applikation rendert je nach URL ein anderes Template in eine „Div-Box“ |
| 1 PL | 1 PL | Login View und Controller in AngularJS programmiert |  |
| 1/2 PL | 1 PL | Registration View und Controller entwickelt |  |
| 1/8 PL | 1/2 PL | Mit Victor die Server Web API Parameter definiert. | Server brauchte gewisse Paramater um ein Objekt zu erstellen, diese wurden kurz definiert |
| 2 PL | 1 PL | User Service im Client implementiert und die Verbindung zur Server Web API erstellt. | Same Origin Policy wurde verletzt, konnte nicht gefixt werden. |
| 1/8 PL | - | Weiteres Vorgehen im Projekt-Team besprochen. |  |
| **Reflexion des Arbeitstages**  In dieser Lektion bin ich persönlich eigentlich sehr gut vorwärts gekommen. Ich habe zwei Views und die entsprechenden Controller dafür erstellt. Zudem haben wir die erste Verbindung zum Server geschafft. Allerdings gab es bei dem Senden des http-Requests, es wurde nämlich die Same Origin Policy verletzt, da der Server und der Client auf einem anderen Port laufen. Mit den Ansichten selber bin ich sehr zufrieden, da sie optisch schön sind und auch funktionieren.  Am Schluss des Tages haben wir im Team besprochen, was wir an diesem Tag erreicht haben und wie wir weiter vorgehen wollen. | | | |
| **Pendenzen für den nächsten Arbeitstag**  Am nächsten Arbeitstag, sollten wir endlich etwas mit den Wohnungen machen. Wir haben uns entschieden die „Overview“ Ansicht zu realisieren, da dies wie eine Startseite für die Wohnungen ist. Victor wird sich um die Server Seite kümmern, Lars wird die Datenbank nochmals updaten und Felix wird sich die „ListFlats“ Ansicht vorknüpfen. Wir sollten das Problem mit der Same Origin Policy noch einmal anschauen, denn als Zwischenlösung haben wir etwas gefunden. Man kann bei Chrome die Security Einstellungen deaktivieren und danach geht es, dies kann allerdings nicht die definitive Lösung sein. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Arbeitsjournal Victor Hargrave**  **Datum:05.05.2015** | | | |
| **Tatsächlicher Zeitbedarf** | **Geplanter Zeitbedarf** | **Beschreibung der Arbeit** | **Bemerkungen, Probleme, genutzte Hilfestellungen** |
| 1.45h | 30m | Login und Registrierung zum Laufen gebracht | Der Gibbix war leider sehr langsam. Ich konnte kaum troubleshooten und normalle Operationen die meistens eine Sekunde dauern gingen eine Minute. Ich fand es sehr frustrierend, aber irgendwie konnte ich die Aufgabe erledigen (die Gibbix Leistung konnte ich nicht mehr erhöhen) |
| 30m | 30m | Planung für Realisierungsbericht,  Aufgaben verteilt | Herr Scheidegger kam kurz vorbei und sagte uns, dass er uns nur nach die Doku bewerten konnte. Deswegen haben wir Fokus auf die Realisierungsbericht gesetzt |
| 1.15h | 45m | Ich und Luca fingen machten unserer Teil von der Doku | Hier war der Gibbix wieder sehr unperformant und wir kopierten den Doku Teil einfach auf dem Desktop von der Host Maschine. |
| **Reflexion des Arbeitstages**  Der Gibbix war viel langsamer als sonst. Ich habe ein wenig geforscht und konnte das Problem am Wochenende finden.  Wir benutzen GIT und durch unsere viele Commits hatten wir kein Speicherplatz mehr auf dem Disk. Das führt immer zu Problemen | | | |
| **Pendenzen für den nächsten Arbeitstag**  Ich muss einfach am Wochenende noch ein wenig mehr am der Dokumentation arbeiten, damit wir sie rechtzeitig abgeben können. | | | |