SW Engineering Fulda Spring 2019

Study Home

Master Team Project 2019 Team 1

Member	E-Mail
Michael Iglhaut	Michael.Iglhaut@informatik.hs-fulda.de
Moritz Mrosek	Moritz.Mrosek@informatik.hs-fulda.de
Manuel Schmitt	Manuel.Schmitt@informatik.hs-fulda.de
Simon Leister	Simon.Leister@informatik.hs-fulda.de
Ramón Wilhelm	Ramon.Wilhelm@informatik.hs-fulda.de

Milestone 1 finished on 15.05.2019

1. Executive Summary:

Neben dem Studium umzuziehen, kann echten Stress bedeuten. Um Studierenden das Umziehen zu erleichtern, möchten wir mit STUDY HOME eine Plattform erschaffen, die auf die besonderen Bedürfnisse dieser Zielgruppe eingeht. Speziell das Vermitteln von WGs und freiwerdenden Studentenwohnungen soll im Vordergrund stehen und somit eine Erleichterung für den wohnungssuchenden Studierenden darstellen. Mit STUDY HOME sollen sich Studierende einfach vernetzen können, um so ein reibungsloseres Umzugserlebnis zu erfahren.

Wohnungssuche -auch in begrenzten Umkreisen- wird mit STUDY HOME für Studierende besonders einfach.
Studierende sowie Vermietende und Makler haben die Möglichkeit, Wohnungssuch- bzw. Wohnungsanzeigen aufzugeben und untereinander zu kommunizieren.
Benutzer von STUDY HOME können Annoncen aufgeben, um WG-Mitbewohner anzuwerben oder ihr freiwerdendes Studentenzimmer anzubieten; mit entsprechender Referenz auf die jeweilige Wohnung.
Video-Boxen der einzelnen Zimmer und mögliche Übertragung von Live-Bildern sollen die Wohnungssuche noch intuitiver machen.

Unser Team besteht aus Ramón Wilhelm (Bachelor Digitale Medien), Moritz Mrosek (Bachelor Digitale Medien), Manuel Schmitt (Bachelor Digitale Medien), Simon Leister (Bachelor Angewandte Informatik mit Vertiefung Embedded Systems) und Michael Iglhaut (Bachelor Angewandte Informatik mit Vertiefung Wirtschaftsinformatik). Alle Abschlüsse wurden an der Hochschule Fulda absolviert.

2. Personae and main Use Cases:

Amber Steinbarth
Ist 21 Jahre alt, studiert Economics an der SFSU, wohnt in Oak Park und zu ihren Hobbys zählen Jogging, Zumba, Kochen und Schwimmen.
Ihr Motto: Weniger ist mehr!

Ziele:
Amber muss sich für ihr Studium nach einer Wohnung in San Francisco umsehen.

Vorerfahrung:
Amber nutzt entweder ihr Smartphone, Tablet, um nach Kochrezepten zu suchen und für sonstige Aktivitäten. Sie ist eher wenig im Internet unterwegs. Sie hat vorher noch nie nach einer Wohnung im Internet gesucht.

Abneigungen:
Sie meidet textlastige und Content überladene Seiten. Sie ist außerdem bei Privatkäufen im Internet sehr skeptisch, nachdem sie beinahe Opfer eines Betrugs auf Craigslist wurde. Deswegen benötigt sie immer klare Auskunft und Kontaktdaten zum Angebot.

Rupert „Rapz“ Pantz
Ist 26 Jahre alt, wohnt in Anaheim und arbeitet als Full-Stack Entwickler. Zu deinen Hobbys zählen Cafés, Musikfestivals, Videospiele, Filme und Theater.
Sein Motto: Keep things as simple as possible.

Ziele:
Rupert ist auf der Suche nach einem Portal, wo er ohne Probleme sein Zimmer an Studenten vermieten kann.

Vorerfahrung:
Durch das Portal „Apartments.com“ hat seine neue Wohnung in Anaheim gefunden. Mit dem Portal ist er jedoch unzufrieden.

Abneigungen:
Er mag keine langen Wartezeiten auf Webseiten. Seiten müssen bei ihm gut designend sein.

Wendy Wells
Sie ist 23 Jahre alt, arbeitet neben dem Studium als Schriftstellerin und wohnt zurzeit in Seattle. Zu ihren Hobbys zählen Literatur, Lesen und Schreiben.
Ihr Motto: „An Zerstreuung lässt es uns die Welt nicht fehlen. Wenn ich lese, will ich mich sammeln.“ - J.W. Goethe

Ziele:
Wendy sucht für ihr Studium eine günstige Wohnung in San Francisco bei einem seriösen Anbieter.

Vorerfahrung:
Vor ihrem Studium ist sie mehrmals umgezogen und besuchte deswegen verschiedene Wohnungsseiten im Internet.

Abneigungen:
Sie möchte die Angebotssuche so unkompliziert wie möglich haben.
Bei ihren Recherchen möchte sie keine „Romane lesen“, sie möchte ihre Informationen kompakt und trotzdem ausführlich.

Use Case	Wohnung suchen
Kurze Beschreibung	Wenn ein Student auf der Suche nach einer Wohnung ist, kann er durch die Anpassung eines Filters Wohnungen im Bereich der gewünschten Hochschule oder Universität suchen. Filteroptionen können zum Beispiel Entfernung zur Hochschule/Universität, Größe der Wohnung, Preis der Wohnung oder Anzahl Zimmer sein.
Akteure	Student/in (Mieter)
Auslöser	Student/in sucht eine Wohnung in der Nähe der jeweiligen Universität/Hochschule.
Vorbedingung	Auf der Suche nach einer Wohnung.
Standardablauf	Student betritt die Webseite. Student klickt auf den Button „Wohnung suchen“. Student setzt Filteroptionen. Student schaut sich die angezeigten Ergebnisse an.

Use Case	Wohnung anbieten
Kurze Beschreibung	Wenn ein Vermieter neue Mieter für eine Wohnung sucht, kann er durch Angabe von Wohnungsdaten diese zum Vermieten anbieten. Wohnungsdaten sind zum Beispiel Bilder, Lage oder Größe.
Akteure	Vermieter
Auslöser	Eine Wohnung des Vermieters ist frei geworden und es wird ein Nachmieter gesucht.
Vorbedingung	Freie Wohnung. Nachmieter gesucht.
Standardablauf	Vermieter loggt sich mit seinem Account ein. Vermieter klickt auf den Button „Wohnung anbieten“. Vermieter gibt die Wohnungsdaten ein. Vermieter gibt Anzeige auf.

Use Case	Konto anmelden
Kurze Beschreibung	Im Kundenkonto können User ihre persönlichen Daten pflegen und favorisierte Wohnungen speichern. Des Weiteren können Studenten Kontakt zum Vermieter aufnehmen, um einen Besichtigungstermin auszumachen oder Fragen zu klären. Vermieter können Wohnungen anbieten und die Fragen der Studenten beantworten.
Akteure	Vermieter und Studenten (Mieter)
Auslöser	Speichern einer favorisierten Wohnung. Anschauen der favorisierten Wohnungen. Kontakt aufnehmen zu einem Vermieter oder Studenten.
Vorbedingung	Ein Konto ist vorhanden.
Standardablauf	Student: Gibt Hochschul- oder Uni-Mail an. Vermieter: Gibt private Emailadresse an. User authentifiziert sich mit einem Passwort.

Use Case	Vermieter anschreiben
Kurze Beschreibung	Wenn ein Student Interesse an einer Wohnung hat, kann er über den Messenger Kontakt zum Vermieter aufnehmen. Dort kann er Fragen bezüglich der Wohnung oder eines möglichen Termins stellen.
Akteure	Student/in (Mieter)
Auslöser	Interesse an einer Wohnung.
Vorbedingung	Konto angemeldet.
Standardablauf	Student meldet sich an. Student wählt aus seinen Favoriten oder aus der Suche eine Wohnung aus. Student klickt auf den Button „Vermieter kontaktieren“. Student schreibt Text.

Use Case	Studentenwohnung anbieten
Kurze Beschreibung	Studenten haben die Möglichkeit Anzeigen zu erstellen, um freiwerdende Plätze in ihrer WG zu bewerben oder freiwerdende Studentenwohnungen anzubieten.
Akteure	Student/in (Mieter)
Auslöser	Ein/e Student/in möchte aus einer WG oder Studentenwohnung ausziehen und muss einen Nachmieter für den Platz finden.
Vorbedingung	Freier Platz. Nachmieter gesucht.
Standardablauf	Student/in betritt die Webseite. Student/in klickt auf den Button “Studentenwohnung anbieten”. Student/in gibt die Wohnungsdaten ein. Student/in gibt die Anzeige auf.

3. List of main data items and entities:

Anzeige: Enthält Informationen zu der Wohnung. Informationen beinhalten Daten zur Wohnung, Bilder/Videos und Daten zum Vermieter.

Vermieter: Ein Vermieter ist eine Person, die eine freie Wohnung auf der Seite anbietet.

Mieter: Ein Mieter ist eine Person, die eine Wohnung in der nähe der jeweiligen Hochschule/Universität sucht.

Favoriten: In den Favoriten kann der Mieter Wohnungen speichern, die in interessieren, um schneller auf diese zugreifen zu können.

Chat: Der Chat dient dem Mieter dazu, Kontakt zum Vermieter aufzunehmen, um zum Beispiel Fragen zur Wohnung zu klären.

Suchfilter: Mit dem Suchfilter kann der Mieter Wohnungen nach gewissen Kriterien Filtern. Kriterien können zum Beispiel Größe der Wohnung, Standort der Wohnung oder Preis der Wohnung sein.

Konto (Vermieter): Das Konto eines Vermieters enthält Informationen über den Vermieter und bietet ihm die Möglichkeit, Wohnungen einzustellen.

Konto (Mieter): Das Konto eines Mieters enthält Informationen über den Mieter und bietet ihm die Möglichkeit, Wohnungen in seinen Favoriten zu speichern, sowie einen Suchfilter zu speichern.

4. Initial list of functional requirements:

1. Der Kunde muss einen Account in seiner jeweiligen Rolle (Mieter, Vermieter) erstellen können.

2. Der Kunde muss sich in seinen persönlichen Account einloggen können.

3. Der Kunde muss seinen Account bearbeiten können (z.B. Passwort ändern).

4. Kunden (Mieter mit Vermieter) müssen untereinander über einen Chat schreiben können.

5. Kunden (Mieter) müssen Wohnungen zu ihrer Favoritenliste hinzufügen können.

6. Kunden (Vermieter) müssen eine Übersicht über ihre angebotenen Wohnungen haben.

7. Kunden (Vermieter) müssen Angebote für Wohnungen erstellen können.

8. Kunden (Mieter) müssen nach bestimmten Suchkriterien Wohnungen suchen können (z.B. Umkreis).

9. Kunde (Mieter) muss Wohnungen nach bestimmten Kriterien Sortieren können (z.B. Entfernung).

10. Kunde (Mieter) muss eine Übersicht über die wichtigsten Informationen einer Wohnung sehen.

11. Kunde (Mieter) muss seine Suchen speichern können.

12. Kunden (Vermieter) müssen Bilder oder Videos, die die Wohnung zeigen zu einem Angebot hinzufügen können.

13. Kunden (Mieter) müssen sich die Bilder oder Videos eines Angebots ansehen können.

14. Kunden (Vermieter) müssen einsehen können, ob sie im Chat angeschrieben wurden.

15. Kunden (Vermieter) müssen ein erstelltes Angebot wieder löschen können.

16. Kunden (Vermieter und Mieter) müssen die Möglichkeit haben, ihr Konto wieder kündigen zu können.

17. Kunden (Vermieter) müssen ihr Angebot überarbeiten können.

18. Kunden (Vermieter und Mieter) müssen Favoriten aus ihrer Liste wieder entfernen können.

19. Kunden (Vermieter) müssen sehen können, wie oft ihr Angebot als Favorit markiert wurde.

20. Kunden (Mieter) müssen bei nicht vermieteten Wohnungen die Möglichkeit haben, via Live Stream die Wohnung sehen zu können.

21. Kunden (Mieter) müssen neben dem Chat noch weitere Möglichkeiten haben Kontakt mit dem Vermieter aufnehmen zu können (z.B. Telefon).

22. Kunden (Vermieter) benötigen klare Regeln, bevor sie ein Angebot reinsetzen können (z.B. die Pflicht eine E-Mail-Adresse anzugeben).

5. List of non-functional requirements:

1. Die Anwendung wird nur unter der Verwendung von Tools und Servern entwickelt, getestet und bereitgestellt, die vom Technischen Direktor genehmigt und in M0 festgelegt wurden.

2. Die Anwendung soll für Desktop und Laptop Browser optimiert sein. Sie wird in wenigstens zwei verbreiteten Browsern korrekt angezeigt.

3. Bestimmte Funktionen der Anwendungen müssen auf mobilen Endgeräten korrekt angezeigt werden.

4. Daten sollen in der vom Team gewählten Datenbank auf dem Server des Teams gespeichert werden.

5. Zu keinem Zeitpunkt sollen mehr als 50 Benutzer gleichzeitig auf die Anwendung zugreifen.

6. Die Privatsphäre der Benutzer wird geschützt und alle Richtlinien zum Schutz dieser werden den Benutzern verständlich mitgeteilt.

7. Die verwendete Sprache sei Englisch.

8. Die Anwendung soll einfach benutzbar und intuitiv sein.

9. Google Analytics wird hinzugefügt.

10. Keine E-Mail-Clients sind erlaubt.

11. Bezahlfunktionen sollen weder implementiert noch simuliert werden.

12. Grundlegende bewährte Praktiken zur Seitensicherheit werden angewendet.

13. Bevor die Seite öffentlich zugänglich gemacht wird, muss der Inhalt dieser vom Administrator genehmigt werden. Zum Inhalt gehören die aufgelisteten Apartments und Bilder.

14. Moderne SE Prozesse und Praktiken werden, wie in der Veranstaltung spezifiziert angewendet.

15. Die Webseite zeigt deutlich erkennbar auf allen Seiten den Text “Fulda Software Engineering Project, Spring 2019. For Demonstration Only”, oben auf der Seite, um eine Verwechselung mit einer echten Anwendung zu vermeiden.

6. Competitive analysis:

Seiten:
https://www.zillow.com/san-francisco-ca/apartments/
https://offcampushousing.usfca.edu/
https://www.student.com/us/san-francisco/p/kapi-residences-park-merced
https://sanfranciscostudenthousing.com/

	Zillow.com	Student.com	Sanfrancisco studenthousing	Offcampushousing.edu	Unsere Seite
Liveübertragung	Nein	Nein	Nein	Nein	Ja
Konto	Ja	Ja	Ja	Ja	Ja
Favoriten	Ja	Ja	Ja	Ja	Ja
Direkter Chat	Nein	Ja	Nein	Nein	Ja
Suchfilter	Nein	Nein	Nein	Ja	Ja
Responsive	Ja	Nein	Ja	Ja	Ja
Google Maps	Ja	Ja	Ja	Ja	Ja
Karussell /Slide Show	Nein	Ja	Ja	Ja	Ja
Tour Guide	Nein	Nein	Ja	Nein	Ja

Die gefundenen Seiten bieten alle verschiedene Funktionen an. Auf jeder Seite können die Wohnung in eine Favoritenliste gespeichert werden. Um diese Favoriten speichern zu können kann man sich jeweils einen Account erstellen. Unsere Seite soll sich von den meisten anderen abheben, indem nicht nur eine Nachricht an den Vermieter gesendet werden kann, sondern es einen richtigen Chat gibt. In diesem können dann mehrere Nachrichten ausgetauscht werden. Des Weiteren soll die Wohnung nicht nur über Bilder einsehbar sein. Auf unserer Webseite sollen Videos oder sogar Liveübertragungen möglich sein. Diese Funktion wird lediglich von der Seite Sanfranciscostudenthousing angeboten. Insgesamt soll unsere Seite eine Kombination aus den Vorteilen aller anderen Webseiten sein und kompakte Informationen liefern. Zusätzlich soll sie den Kunden optisch ansprechen und mit wenig Text auskommen.

7. High-level system architecture and technologies used:

Hardware/Software – Core (IaaS+PaaS)

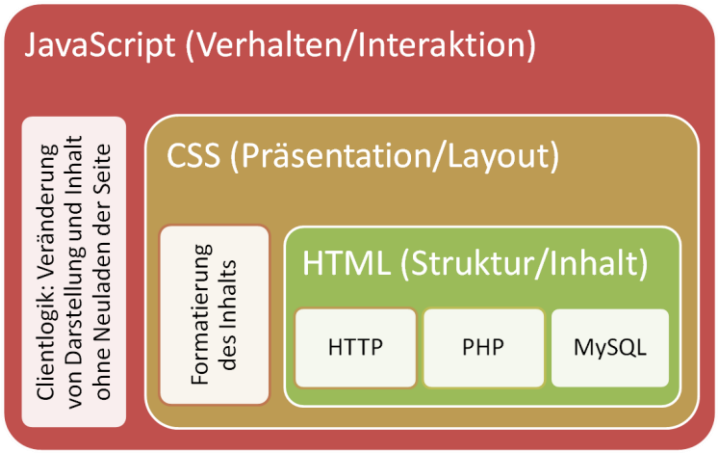
- Betriebsystem: Centos 7 (RHEL Fork)
- 10GB HDD / Up to 1,5GB RAM dynamically
- Failover mit VirtIO auf Replikationpartner mit verteilt-replizierten Block Devices
- Generischer Servername: pizarro.uberspace.de für Nutzer hsftp
- Webservername: https://hsftp.uber.space

Websoftware-Stack (PaaS)

- Apache 2.4 mit nGinx als HTTPS (Let's Encrypt) und HTTP/2 Frontend (Möglichkeit zur Aufschaltung von anderen Anwendungen; z. B.: /mynodejsapp wird auf Port xyz durchgereicht, anstatt per Standard an den „inneren“ Apache Webserver)
- PHP 7.3 with FastCGI / Apache mod_proxy_fcgid und PHP-FPM (FastProcessManager)
- MariaDB 10.3 (MySQL Fork, Abspaltung nach Oracles MySQL Kauf)

Webapplikations-Softwarearchitektur (SaaS)

Jede Webapplikation wird von mindestens einem Webserver (sowie weitere Backend-Server [Datenbank], z. B. MySQL) ausgeliefert. Dabei wird das zustandslose HTTP/S Protokoll verwendet. Die Ressourcen jeder Schicht werden immer über den Webserver an den Client/Webbrowser verteilt – unterschiedlichen Ursprungs sowie Zweck sind jedoch die Daten in den unterschiedlichen Layer:



1. HTML ist ein MarkUp, eine Grundstruktur, des Dokuments. Sie kann sowohl statisch im Dateisystem des Webservers hinterlegt sein, als auch dynamisch durch einen HyperText-Preprocessor wie PHP generiert werden. Auf Client-Seite wird sie im Browser interpretiert und dargestellt und ist zunächst statisch.

2. CSS formatiert den erhaltenen MarkUp und kann für unterschiedliche Medien und Zielgeräte (Screen, Print ...) die passende Darstellung finden, sofern entsprechende Darstellungsregeln hinterlegt worden sind. Für das Projekt soll eine nutzerfreundliche Darstellung für unterschiedliche Bildschirm-Endgeräte (Desktop, Smartphone, Tablet) bereitgestellt werden. Diese werden über CSS3 MediaQueries realisiert werden. CSS ist in der Regel als statische Datei am Webserver hinterlegt und wird vom Browser interpretiert.

3. JavaScript (ECMA-262) bzw. deren Nachfolger (TypeScript, WebAssembly…) ermöglichen die clientseitige Dynamik einer Webapplikation, die bis jetzt nur dem webserverseitigen Präprozessor für HTML vorbehalten war. Durch JavaScript sind erst die heute typische bekannten ProgressiveWebApps (PWA‘s, MicrosoftOffice 365, ElsterOnline...) möglich geworden. Durch die Komplexität bzw. die unterschiedlichen Entwicklungs- & Implementierungsstände in den unterschiedlichen Browsern (Internet Explorer, Edge, Firefox, Chrome, Safari, Opera...) bzw. deren RenderEngines (Trident, Edge, Quantum, 2x WebKit, Presto bzw. Webkit), wurde Frameworks geschaffen, die diese Problematik gegenüber dem Programmierer abstrahieren. JS wird in der Regel als statische Datei am Webserver hinterlegt und durch den Browser bzw. deren JS Engine (Spider, V8…) interpretiert und in einer (hoffentlich undurchdringbaren) Sandbox ausgeführt.

Bzgl. der dritten Schicht einer Webapplikation existiert die größte Vielfalt, sodass die Festlegung auf ein Framework noch nicht endgültig vollzogen worden ist; in die engere Wahl fielen AngularJS, ReactJS und Bootstrap.
Für eine gute SEO (Suchmaschinen-Optimierung) Strategie empfiehlt es sich immer die einzelnen Schichten strikt zu trennen und nur so wenig Berührungspunkte (Include, Script-Tags…) wie möglich zu haben, um das Signal-Rausch-Verhältnis (Entropie von Struktur & Inhalt VS z.B. Inline-CSS / JS) einer Webseite gering zu halten.

Auto-Deployment Prozess über GitHub auf Webserver
Im Team-Repository auf GitHub ist ein WebHook eingerichtet, der ein PHP Skript auf dem Ziel-Server aufruft und eine Synchronisation des GitHub und dem Webserver - Repository auf durchführt. Anschließend wird die Master-Branch in das Webapplication-Document-Root geschrieben und ist öffentlich einseh- und ausprobierbar.

8. Team:

Member	Roles
Ramón Wilhelm	Datenbank, Back-End
Moritz Mrosek	Front-End-Lead
Manuel Schmitt	Back-End
Simon Leister	Back-End-Lead, GitHub-Master
Michael Iglhaut	Team-Lead, Dokument-Lead, Datenbank

9. Checklist:

Item	Answer
•Team found a time slot to meet outside of the class	DONE
•GitHub master chosen	DONE
•Team decided and agreed together on using the listed SW tools and deployment server	DONE
•Team ready and able to use the chosen back- and front-end frameworks and those who need to learn are working on learning and practicing	ON TRACK
•Team lead ensured that all team members read the final M1 and agree/understand it before submission	DONE
•GitHub organized as discussed in class (e.g. master branch, development branch, folder for milestone documents etc.)	ON TRACK