Studiengang Medieninformatik

Hochschule
Kaiserslautern
University of
Applied Sciences

Studienprojekt

Projektbericht

Copsi - eine Kollaborationssoftware

WS 2018/19

vorgelegt von

Miller Caroline

Schuler Sebastian

Spandl Philipp

18. Februar 2019

Betreuung

Prof. Dr. Dieter Wallach

GLIEDERUNG

1 Historie

Name	Kapitel	Datum	Versionsnr.	Beschreibung
Caroline Miller	0	06.01.2019	1.0	Erstellung und Gliederung
Caroline Miller	0	09.01.2019	1.1	Umstrukturierung der Gliederung
Caroline Miller	2	11.02.2019	1.2	Beschreibung Kapitel 2
Caroline Miller Philipp Spandl	4	13.02.2019	1.3	Beschreibung Kapitel 4
Sebastian Schuler Philipp Spandl	6	13.02.2019	1.4	Beschreibung Kapitel 6
Caroline Miller Philipp Spandl Sebastian Schuler	2, 3, 4, 5	14.02.2019	1.4	Beschreibung Kapitel 2, 3, 4, 5
Sebastian Schuler, Philipp Spandl	6, 7	15.02.2019	1.4	Beschreibung Kapitel 6, 7
Sebastian Schuler	11	15.02.2019	1.4	Links und Beschreibung Quellen

2 Einleitung

2.1 Projektbeschreibung

Copsi ist eine Desktop-Applikation, die zur Kommunikation und zum Austausch von Dokumenten zwischen Professoren, Assistenten und den Studierenden der Hochschule Kaiserslautern dient. Der aktuelle Austausch von Informationen, Neuigkeiten oder relevanten Daten zu jeweiligen Modulen sieht vor, dass die Studierenden auf verschiedene Plattformen wie das OLAT-System, Google Drive, Studierenden E-Mails oder auch auf Instant-Messaging-Dienste wie WhatsApp zurückgreifen müssen, um sich halbwegs einen Überblick verschaffen zu können. Informationen werden schnell übersehen und die direkte Kommunikation mit Kommilitonen ist oftmals nicht möglich, da hierfür die Studierenden gezielt im Campusboard nach jeweiligen Kommilitonen im Personenverzeichnis suchen müssen, vorausgesetzt es besteht überhaupt Kontakt.

Aufgrund dieser Unzufriedenheit der verschiedenen Dienste, die im Studien-Alltag genutzt werden müssen entstand die Idee eine Software zu kreieren, die die Kommunikation sowie den Datenaustausch vereinfachen soll. Dies soll über einen zentralen Server in Verbindung mit einer Datenbank erfolgen.

2.2 Projektziel

Das Ziel dieses Studienprojektes ist ein Minimum Viable Product (MVP), das die Struktur von Kursen, Channels und Nachrichten aus einer Datenbank lesen und darstellen kann. Zusätzliche Funktionen wären zum Beispiel das Hoch- und Herunterladen von Dateien und User Rechte.

2.3 Projektumfeld

Das Projekt wurde von Prof. Dr. Dieter Wallach als Dozent der Hochschule Kaiserslautern in Auftrag gegeben. Unser Projektteam bestehend aus Caroline Miller, Sebastian Schuler und Philipp Spandl hat als Studentisches Projektteam der Hochschule Kaiserslautern diesen Auftrag angenommen. Somit dient Prof. Dr. Dieter Wallach als direkter Ansprechpartner des Projektteams. Planung und Realisierung des Projekts entstehen bei den Mitgliedern des Teams privat.

Interessengruppen sind in diesem Projekt Dozenten und Studierende der Hochschule Kaiserslautern.

2.4 Vorgehensweise

Nachdem im zweiten Kapitel die Problemstellung und Zielsetzung dargelegt wurden, werden im dritten bis sechsten Kapitel die Realisierung des Projekts inklusive Planung, Analyse, Entwurf und Implementierung aufgeführt.

Zu Beginn wird die Planung vorgenommen, in der das aktuelle Problem von Kommunikationsplattformen, genutzt von Lehrenden und Studierenden, erläutert wird. Anschließend werden die nächsten Phasen definiert, sowie Meilensteine festgelegt. In der Analysephase wird eine Ist-Analyse an der bestehenden OLAT-Plattform und ihren Arbeitsprozessen und Problemen erstellt. Basierend auf der Problemanalyse wird die Soll-Analyse durch Proto-Personas und Interviews mit Stakeholdern ermittelt. daraufhin können Szenarien erstellt werden woraus sich die Anforderungen ableiten lassen. Diese werden in funktionale und nicht-funktionale Anforderungen an das System unterschieden und in einer Priorisierungsmatrix dargestellt.

In der Entwurfsphase werden die ersten Designentwürfe in Skizzen bzw. Scribbles erfasst und durch Wireframes digital visualisiert. Mit den aus Wireframes erstellten Prototypen werden Usability Tests mit Stakeholdern durchgeführt, evaluiert und der bestehende Prototyp verfeinert. Anschließend wird das vorläufige Design festgelegt.

In der Implementierungsphase werden erste Implementierungsansätze realisiert. Mit Hilfe von Klassendiagrammen, verschiedenen Frameworks und Packages, wird nach und nach die Funktionalität erweitert um den Anforderungen aus den vorhergehenden Phasen gerecht zu werden.

In der Implementierungsphase wurde an zwei Stellen gleichzeitig gearbeitet: Der Client Oberfläche (Caroline) und der Client & Server Funktionalität (Sebastian, Philipp). Zuerst wurde jedoch eine Datenbank nach dem DB Modell erstellt und ein Script, dass die Datenbank mit Sample Daten zum Testen füllt.

Am Ende soll das gesamte Projekt noch einmal kritisch betrachtet werden und ein Fazit aufgestellt werden.

3 Projektplanung

3.1 Problem Statement

Es gibt keine einheitliche Plattform, die die Studierenden und Lehrenden nutzen können um zu kommunizieren und Dateien bereitzustellen. Mit diesem Projekt wird es möglich sein, sein Studium auf nur einer Plattform übersichtlich zu organisieren. Außerdem ist die bestehende Plattform schwerfällig und überladen.

3.2 Zeitplanung

15.10.18:
Erhalt des Projekts durch den Auftraggeber. Anfängliche Planung des Projekts.

29.10.18 - 07.11.18:

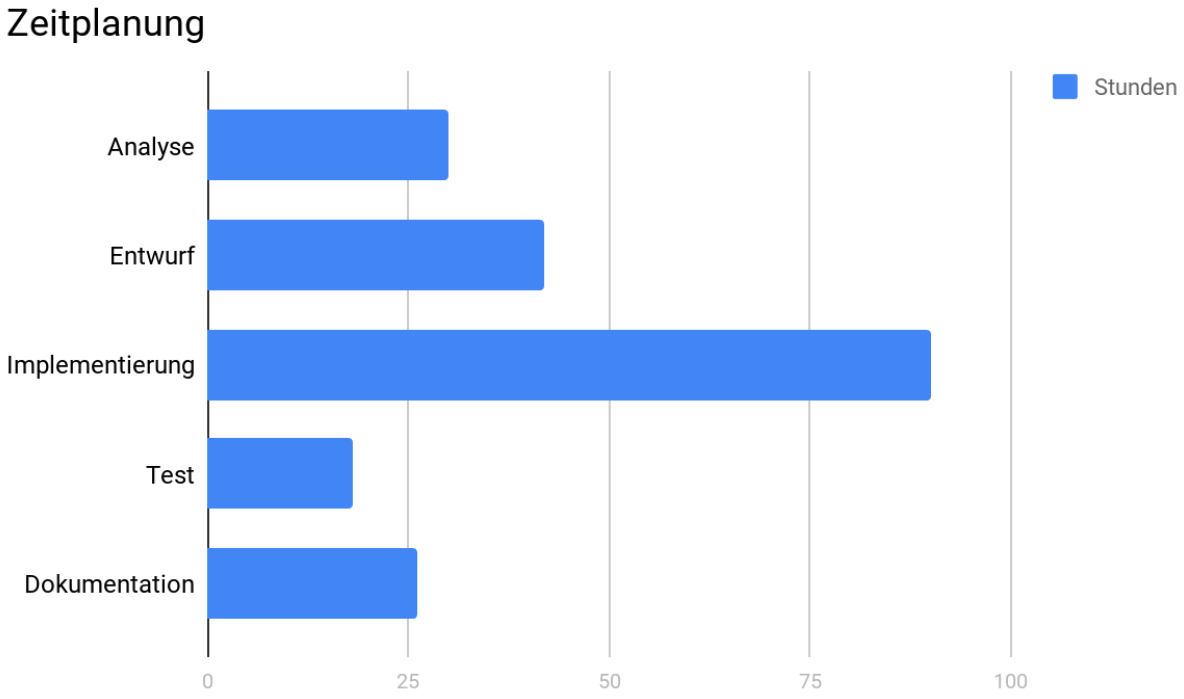
Analysephase: Proto Personas, Interviews
anfängliche Entwurfsphase: Scribbles
Meilenstein: Erste Implementierung einer simplen Chat Funktion.

08.11.18:
Treffen mit Auftraggeber, Demonstrierung der simplen Chat Funktion.

09.11.18 - 28.11.18:
Entwurfsphase: Wireframes, Prototyp, Usability Tests, Design
Meilenstein: Erster Ansatz eines funktionalen Chat Clients mit Server.

29.11.18:
Treffen mit Auftraggeber, aktuellen Stand der Anwendung vorzeigen

ab 29.11.18 - 01.01.19:
Implementierungsphase, Programmieren des Clients und des Servers
Meilenstein: Fertigstellung des Clients und des Servers mit Datenbank.


Zeitplanung
Stunden
Analyse
Entwurf
Implementierung
Test
Dokumentation
0
25
50
75
100

4 Analysephase

Zur Planung des Projektes müssen relevanten Informationen und Funktionen über die aktuell genutzte Lernplattform gesammelt und analysiert werden, ob diese weiter relevant für dieses Projekt sind oder nicht.

Dazu werden in der Soll-Analyse zwei Proto-Personas erstellt und Interviews mit Stakeholdern geführt. Aus diesen Vorgängen lassen sich die Szenarien definieren und zuletzt können die Anforderungen abgeleitet werden, welche in funktional und nicht funktional unterteilt werden.

4.1 Ist-Analyse

Die OLAT-Lernplattform ist eine Web-Applikation, die bis auf wenige Ausnahmen von den Professoren der Hochschule Kaiserslautern genutzt wird um ihre Kurse zu organisieren. Es ist unter anderem möglich einen neuen Kurs anzulegen, Kurse im Verzeichnis zu suchen, Studierende einzuladen, Dateien hoch- und runterzuladen, sogar die Möglichkeit für einen Kurs-Chat und Diskussionsforum wird gegeben.

Die angebotenen Funktionen von OLAT werden nicht von den Studierenden angenommen, da die Nutzung von OLAT zu kompliziert ist für den täglichen Gebrauch. Durch ein missglücktes Interface fällt die Orientierung schwer, beispielsweise ist es nicht auf dem ersten Blick erkennbar, dass das System überhaupt eine Such-Funktion besitzt. Es gibt keine allgemeine Startseite, bei dem der Nutzer sich sicher sein kann, dass er an einen Orientierungspunkt zurück gelangt - zumindest ist die “Startseite” nicht als solche gekennzeichnet.

Es wirkt schwerfällig und überladen mit Funktionen die scheinbar nicht nötig bzw relevant sind, wie zum Beispiel dem eigenen Profil. Außerdem kann die aktuelle Sitzung ablaufen, was dazu führt dass der Benutzer abgemeldet wird - ohne Warnung oder Benachrichtigung.

4.2 Soll-Analyse

Aus Proto Personas und Interviews lassen sich die Zielgruppen definieren und die Ergebnisse für die Anforderungen ableiten.

4.2.1 Proto Persona

Proto Personas werden aus Annahmen heraus erstellt, die sich im weiteren Verlauf durch die Interviews mit Stakeholdern bestätigen.

Jan PROFESSOR

Prof. Dr. (Informatik), 40 Jahre alt
viel Erfahrung mit Applikationen, hohe Ansprüche bei der Benutzung

Kontext:
> Oft in der Uni
> Häufig bei Meetings und Treffen

Ziele:
> Austausch mit Studierenden
> Aktuellen (Vorlesungs-)Stand weitergeben
> Arbeitsmaterialien für Studierende bereitstellen
> feste Termine verkünden

Aufgaben:
> Vorlesungszeit planen
> hält Veranstaltungen
> betreut Projekte
> eigene Forschungsprojekte
> Sprechstunden

Frustpunkte:
> Kommunikation mit Studierenden nur mit E-Mail möglich
> nutzte bereits OLAT, sammelte schlechte Erfahrungen, weshalb er nur noch mit E-Mail kommuniziert

Sven STUDENT

Informatik-Student im 4. Semester, 22 Jahre alt
etwas Erfahrung mit Applikationen, ungeduldig bei der Benutzung

Kontext:
> besucht nicht regelmäßig die Veranstaltungen
> arbeitet häufig von Zuhause alles nach
> lernt Zuhause

Ziele:
> Studium in Regelzeit schaffen
> guten Schnitt
> mit anderen Studierenden austauschen (bei Fragen, bei denen er sich nicht direkt an den

Professor wenden möchte)
> Lerngruppen finden

Aufgaben:
> Arbeitsmaterialien beschaffen
> Bearbeitete Arbeitsmaterialien hochladen

Frust Punkte:
> findet sich intuitiv schlecht auf OLAT zurecht
> genervt, dass Dozenten über unterschiedliche Plattformen kommunizieren, da er mehrere Accounts auf Neuigkeiten überprüfen muss

4.2.2 Interviews

Die Interview mit den Stakeholdern wurden benötigt um Ziele klar zu definieren. Es war wichtig die Schwächen und Fehler des Konkurrenz Produktes herauszufinden um diese zu verbessern.

Interviews mit fünf Stakeholdern (Professoren und Studierende) bestätigten die Annahmen, die durch die Proto Personas erstellt wurden. Hierbei gab es bei Vieren sehr viele Beschwerden über OLAT, die größte Gemeinsame wäre die Unübersichtlichkeit. Dazu kommt noch wie sehr es mit Funktionen überladen ist und dass es an manchen Stellen umständlicher ist als es sein müsste, beispielsweise Dateien hoch- und runterladen - ein Doppelklick führt direkt dazu die Datei runterzuladen, statt erstmal eine Vorschau zu erhalten.

4.3 Anforderungen

Die herauszufilternde Zielgruppe ist für diese Anwendung recht offensichtlich. Die Anwendung soll gezielt für Studierende und Dozenten entwickelt werden. Hierbei stellt sich dann die Frage, welche Wünsche die Zielgruppe haben könnte um mit dem System zu interagieren.

Folgende Szenarien wären plausibel:

Ein Student möchte dem Dozent eine Frage zur Übung stellen, ist sich jedoch nicht sicher, ob die Frage eventuell “dumm” sein könnte. Es ist ihm peinlich und er unterlässt die Frage.

Ein Dozent möchte seinen Studierenden Dateien zukommen lassen. Das OLAT System ist ihm jedoch zu kompliziert und er lädt sie an falscher Stelle hoch.

Ein Dozent ist krank und möchte seinen Studierenden noch am gleichen Tag benachrichtigen, dass die Vorlesung ausfällt, jedoch bietet das OLAT System dafür keine Lösung und Studenten schauen zu spät in ihr Email Postfach.

4.3.1 Funktional

Das System soll das Managen von Kursen durch Professoren und Studierenden ermöglichen.

Das System soll Kurse welche im Stundenplan im Campusboard eingetragen sind automatisch für die Studierenden übernehmen.

Das System soll den Studierenden ermöglichen Arbeitsgruppen zu bilden.

Das System soll ermöglichen private Nachrichten zu verschicken.

Das System soll für einen Kurs einen Gesamt-Chat bereit stellen.

Das System soll ermöglichen Nachrichten anonym zu verschicken.

Das System soll Professoren ermöglichen Dateien zu einem Kurs zur Verfügung zu stellen und Dateien herunterzuladen.

Das System soll Studenten ermöglichen Dateien (zum beispiel) Lösungen hochzuladen und öffentliche Dateien herunterzuladen.

Das System soll ein Rechtesystem für das Einsehen von Channels, Dateien etc. ermöglichen.

Das System soll die Rechte der Professoren über die Rechte der Studierenden stellen.

Das System soll eine Liste der beigetretenen Kurse bereitstellen.

Das System soll eine Suche nach Kursen ermöglichen.

Das System soll eine Suche innerhalb des Kurses ermöglichen.

Das System soll es den Professoren ermöglichen zu einem Kurs Channels anzulegen.

Das System soll ermöglichen den eigenen Account zu personalisieren (Prof. Dr. Zimmermann).

Das System sollte für Professoren Templates für Kurse bereitstellen und Kurse aus alten System übernehmen.

Das System sollte intuitiv bedienbar sein.

Das System soll eine Login-Funktion bieten, welche den Campusboard Account nutzt.

Das System soll eine Option bieten das Profil minimal zu bearbeiten.

Das System soll eine Drag-and-Drop Funktion bieten.

Das System soll ermöglichen Benachrichtigungen von Kursen, Chats etc. stumm zu schalten.

Das System soll ermöglichen Leute über einen Einladungslink zum Server einzuladen.

Das System soll eine Vorschau von herunterladbaren Dateien ermöglichen.

4.3.2 Nicht funktional

Das System soll nach 3 gescheiterten Anmelde Versuchen den Account für 5 Minuten sperren.

Das System soll mindestens 4 Wochen ohne Unterbrechung laufen können.

Das System soll Nachrichten innerhalb von maximal 100 ms an den Server geschickt haben.

Das System sollte minimale Last auf die CPU haben.

Das Versenden von Nachrichten sollte SSL-verschlüsselt erfolgen.

Das System sollte nicht länger als 100 ms für den Wechsel zwischen Servern-Channeln benötigen.

Das Design des Systems sollte konsistent sein und sich am CI der HS Kaiserslautern orientieren.

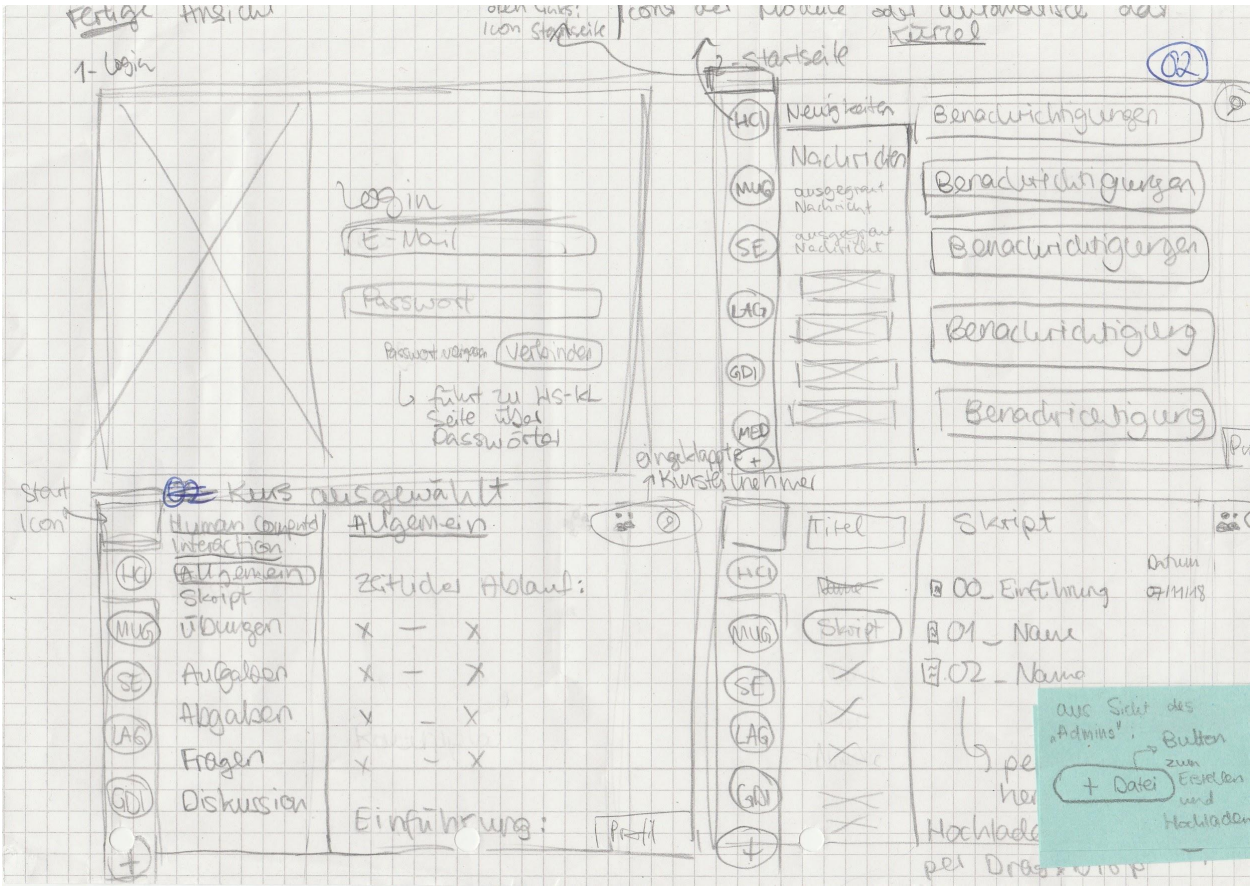
5 Entwurfsphase

In der Entwurfsphase wird die Architektur und das vorläufige Design des Clients festgelegt, basierend auf den folgenden Methoden der Entwurfsphase. Der Auftraggeber ließ einem hierbei eine freie Gestaltungsmöglichkeit, mit der Bedingung, sich möglichst an der Corporate Identity der Hochschule Kaiserslautern zu orientieren, hierbei wurde sich vor allem an den Farben der einzelnen Studienbereiche orientiert, beispielsweise Blau für Informatik.

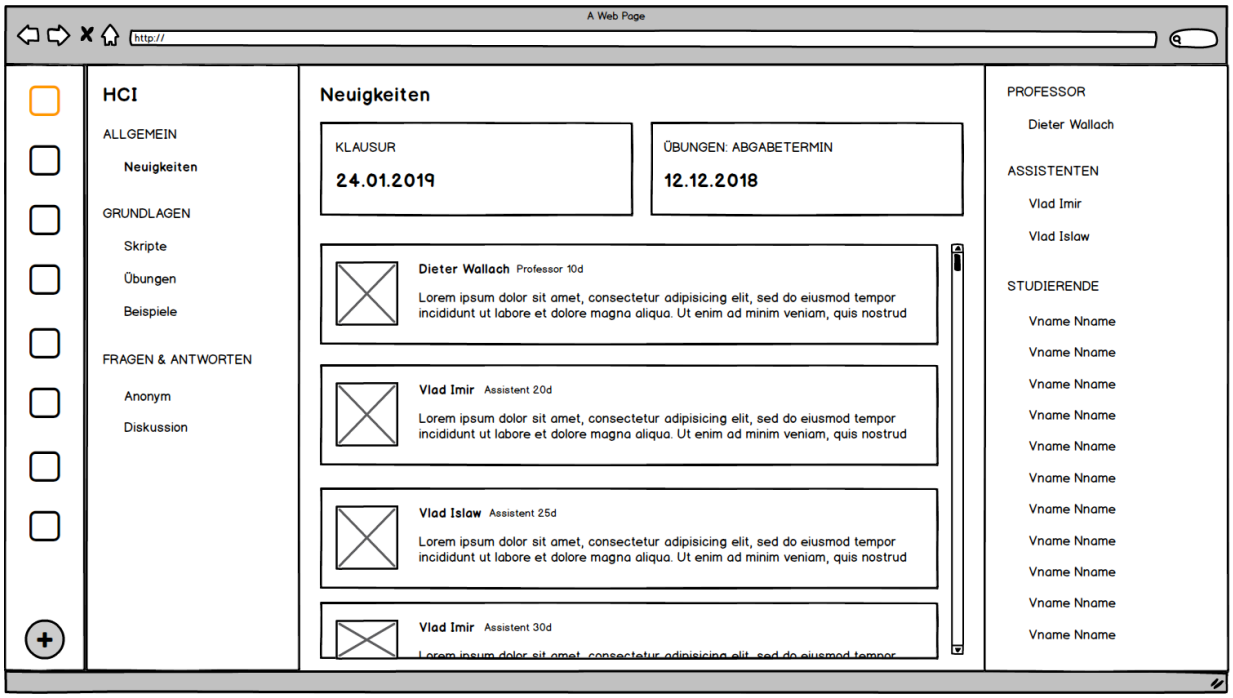
5.1 Scribbles und Mockups

Um sich einen Überblick zu verschaffen, wie die einzelnen Seiten des Clients aussehen und welche Funktionen angeboten werden sollen um dem Nutzer einen möglichst einfachen und intuitiven Umgang mit dem System zu gewährleisten, wurden grobe Zeichnungen mit Bleistift angefertigt. Diese sind einfach zu korrigieren und mit Post-Its ist es möglich, ein Szenario abzulaufen und Unklarheiten im Vorfeld zu beseitigen oder erst gar nicht zu erschaffen.

Mithilfe der Scribbles war es möglich die Grundstruktur festzulegen und sich darauf zu einigen, welche wiederkehrenden UI-Elemente verwendet werden.



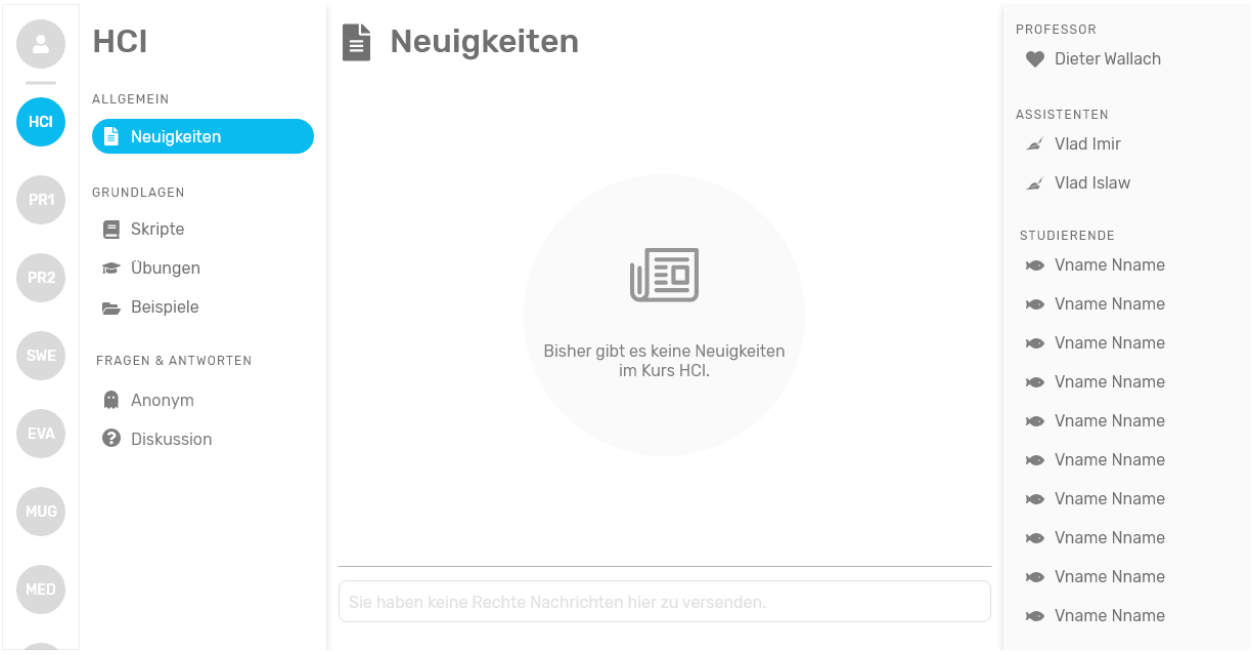
Die Scribbles konnten daraufhin grob mithilfe Balsamiq als Mockups visualisiert werden.



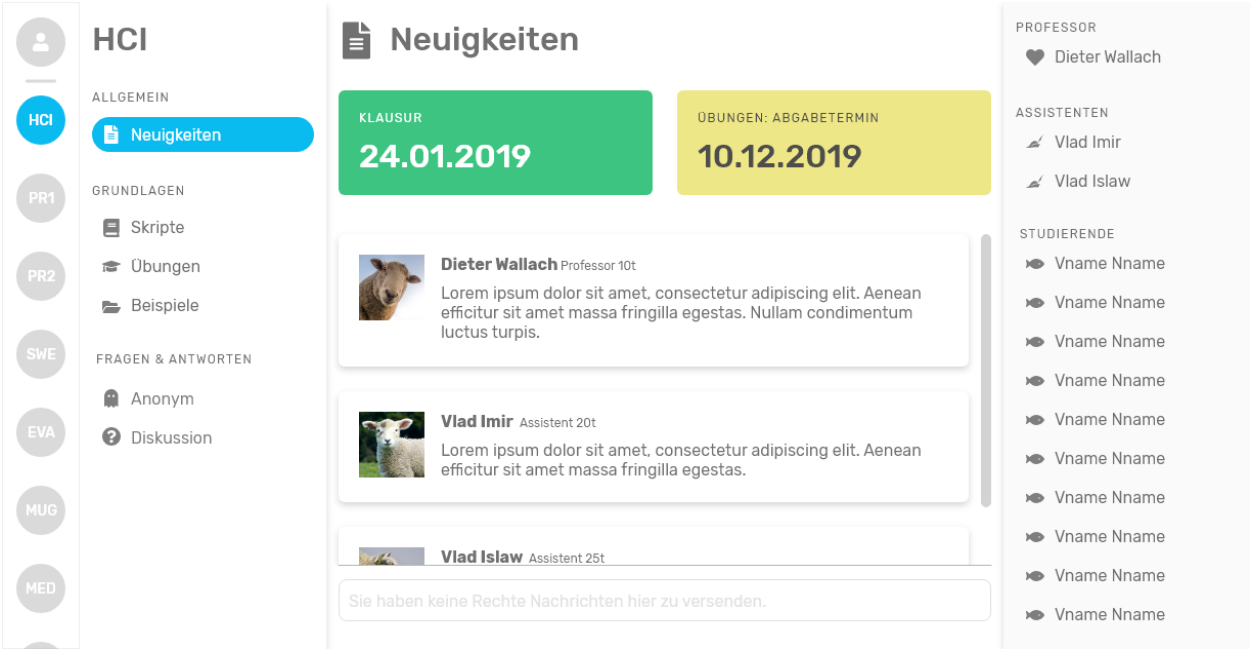
Im Anhang unter 10.1 und 102. sind weitere Screens und Mockups zu sehen.

5.2 Wireframes und Prototyp

Mit Adobe XD wurden die Wireframes angefertigt, welche anschließend durch Interaktionsmöglichkeiten verbunden wurden um als Prototyp zu dienen. Von jedem Wireframe gab es zwei Versionen: einen “leeren” und einen ausgefüllten Screen, d.h. wie der Screen aussehen würde, wenn Professoren und Assistenten regelmäßig Neuigkeiten verfassen und Dateien hochladen und wie Screens aussehen könnten, wenn Teilnehmer miteinander chatten.



Ein leerer Screen aus der Sicht eines Studenten, anschließend ein Screen, wie er mitten im Verlauf des Semesters aussehen könnte:



Im Anhang sind die restlichen Wireframes und eine Übersicht des Prototypes zu finden.

5.3 Usability Tests

Mithilfe formativer Usability Tests wurde herausgefunden wo Optimierungspotenzial am Design der Wireframes herrscht, nach jedem Durchlauf wurden die Wireframes verändert. Hierbei wurden fünf Probanden befragt, diese bekamen zu Beginn ein kurzes Briefing worum es bei der Anwendung geht und die Bitte, ihre Gedanken möglichst laut zu jedem Punkt, der ihnen auffällt, wiederzugeben. Jeder Proband durchlief zwei Tests, ein Prototyp war hierbei nicht mit Nachrichten oder Dateien gefüllt, der nächste gab den Anschein, man hätte die Anwendung über das Semester hinweg ausführlich gepflegt und auf den aktuellsten Stand gehalten. Dabei bekamen sie zu jedem Screen des Prototyps erst die Möglichkeit all ihre Gedanken zu allen Punkten, die ihnen auffielen, zu äußern, anschließend erhielten sie einfache Anweisungen, was sie mit der Anwendung tun sollen. Hierbei wurden im zweiten Durchlauf die Szenarien aus Kapitel 4.3 durchlaufen. Zuletzt hatten sie die Möglichkeit nochmal ihr Feedback abzugeben, Kritikpunkte anzusprechen und Vorschläge zu bringen.

5.4 Vorläufiges Design

Der Login-Screen zeigt im Hintergrund einen Farbverlauf von zwei CI-Farben der Hochschule, blau für Informatik und grün für BWL - bisher war es noch nicht möglich alle Farben der Studienbereiche in einen harmonischen Farbverlauf zu bringen. In der Mitte gibt es ein einfaches Login Formular, jeder Nutzer meldet sich hierzu mit seiner Hochschul-E-Mail und dem dazugehörigen Passwort an. Sollte derjenige das Passwort vergessen haben, gibt es eine Verlinkung zur Hochschule Kaiserslautern und einen Artikel zu den Passwörtern und Hochschul-Accounts.

Farben der Hochschule:

- #69B22F grün
- #006C55 dunkelgrün
- #237D87 teal
- #0CB0D7 blau

Das vorläufige Design der Anwendung ist in erster Linie in weiß mit dunkelgrauer Schrift gehalten, die Schriftart ist Rubik. Es ist eine sans-serif Schriftart, welche durch leicht-abgerundete Ecken keinen harten Kontrast wirft und einfach zu lesen ist.

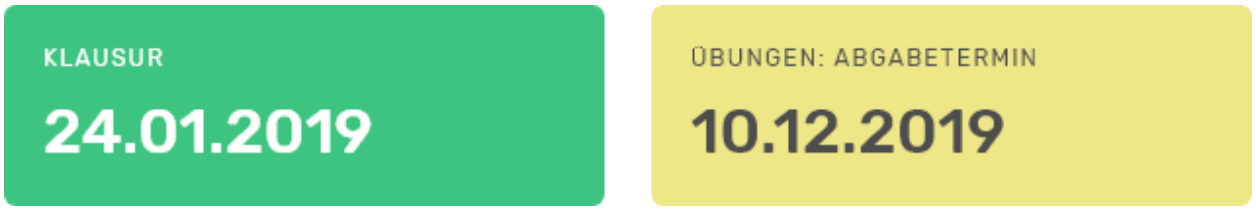
Die Struktur sieht vor, möglichst alle wichtigen Informationen immer im Blick zu haben, das bedeutet, dass die Informationen, in welchem Modul und in welchem Channel sich der Nutzer gerade aufhält sowie die Teilnehmer des Moduls, immer angezeigt werden.

Das Design besteht aus vier Spalten, die erste Spalte von links zeigt alle vorhandenen Module an, die nächste Spalte bezieht sich auf das ausgewählte Modul, welches in der Akzentfarbe Blau

hervorgehoben wird. Die Akzentfarbe Blau entspricht hierbei dem CI-Blau der Fakultät Informatik und Mikrosystemtechnik der Hochschule Kaiserslautern.

In der zweiten Spalte gibt es eine Auflistung von sogenannten Channels, der aktive Channel wird hierbei ebenfalls in der Akzentfarbe hervorgehoben. Die jeweiligen Channels werden zusätzlich mit einem assoziierbaren Icon dargestellt, so hat beispielsweise der Channels “Neuigkeiten” eine Papierseite als Icon, welches man mit einer Seite von einer Zeitung für Neuigkeiten verbinden könnte und der Channels “Anonym” einen Geist:

Die dritte Spalte zeigt den Haupt-Content an, hier gibt es vorerst drei Arten: Neuigkeiten, Tabellen und Chats. Alle Arten haben gemeinsam, dass der jeweilige Titel im Kopf angezeigt wird, auf selber Höhe gibt es noch ein Text-Element, welches die aktuellen Semesterdaten anzeigt. Neuigkeiten und Tabellen haben eine weitere Gemeinsamkeit, hier werden direkt unter dem Titel zwei Karten mit je einem wichtigen Datum angezeigt. Die linke Karte zeigt den Klausurtermin an, die rechte Spalte kann ein beliebiges wichtiges Datum anzeigen, beispielsweise die Deadline für eine Abgabe:

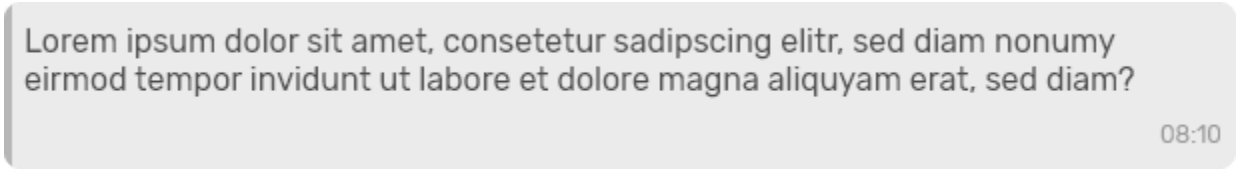


Der Neuigkeiten-Content besteht hauptsächlich aus weißen “Karten”, angelehnt an die Karten aus Material Design, die in der linken Seite ein Bild vom Verfasser der Neuigkeit anzeigen, vorläufig wurden Bilder von Schafe und Lämmer als Platzhalter verwendet, diese haben keine tiefere Bedeutung. In der rechten Seite der Karte wird der Name des Verfassers, die Rolle, eine Zeitangabe und der Text angezeigt:

Im unteren Teil dieser Spalte befindet sich die Texteingabe. Durch eine normale Eingabe und der Enter-Taste können Nachrichten verschickt werden.

Im Tabellen-Content können Dokumente hoch- und runtergeladen werden. Hier ist eine klassische Tabelle zu sehen. Name, Größe und Datum stehen im Header, darunter gibt es dann die Auflistung der einzelnen Dateien. Zu jeder Datei werden einige Details wie Name, Größe, Datum und rechts davon ein Download Button angezeigt. Im unteren Bereich ist ein Button in der Akzentfarbe, mit dem die Dateien hochgeladen werden können.

Der Chat-Content ist vorerst nur anonym verfügbar, hier werden die Nachrichten in farblichen “Boxen” angezeigt, es gibt keine Anzeige von Namen, nur eine Zeitangabe:

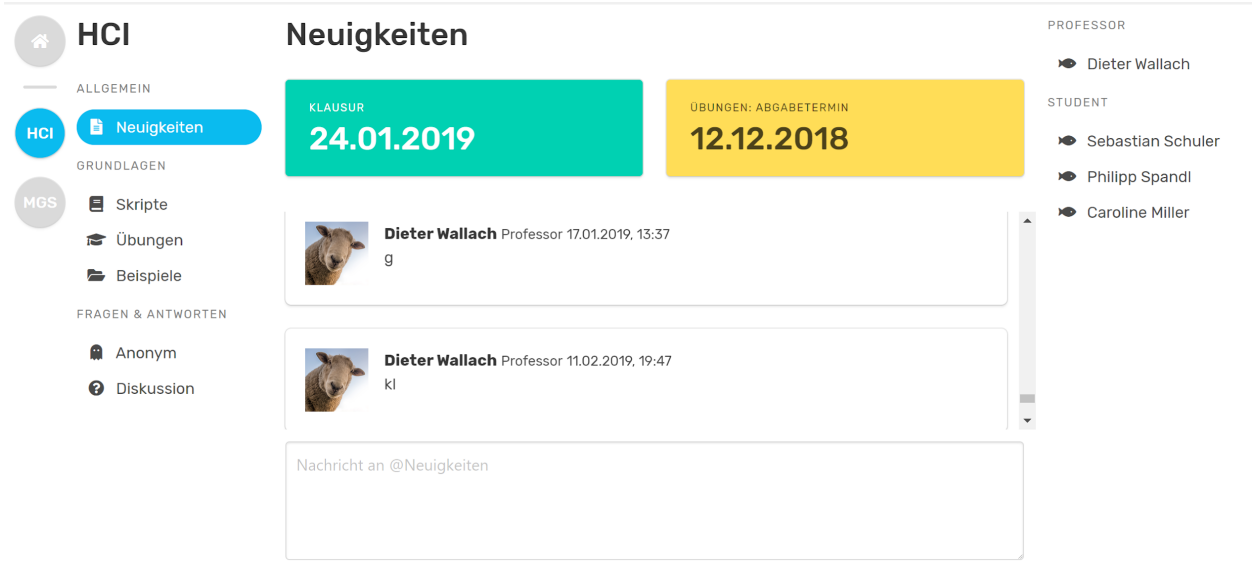


Wie im Neuigkeiten-Content gibt es auch hier im unteren Bereich den Textbereich um seine Nachrichten zu verfassen.

Die vierte Spalte zeigt die Mitgliederliste des Moduls an. Beginnend mit dem Professor, gefolgt von den Assistenten und anschließend die Studierenden. Ein Hover-Button suggeriert, dass es möglich sein kann, diesen Personen eine private Nachricht zu verschicken.

Umgesetzt wurde das Design mit HTML und CSS, durch die einfache Struktur und den UI-Elementen wie den Karten oder den Boxen wurde das Bulma Framework benutzt. Es ist ein simples CSS Framework, welches bei dem Layout und vielen UI-Elementen seinen Einsatz findet und trotzdem die Möglichkeit gibt, jedes einzelne Element individuell zu kreieren und zu gestalten.

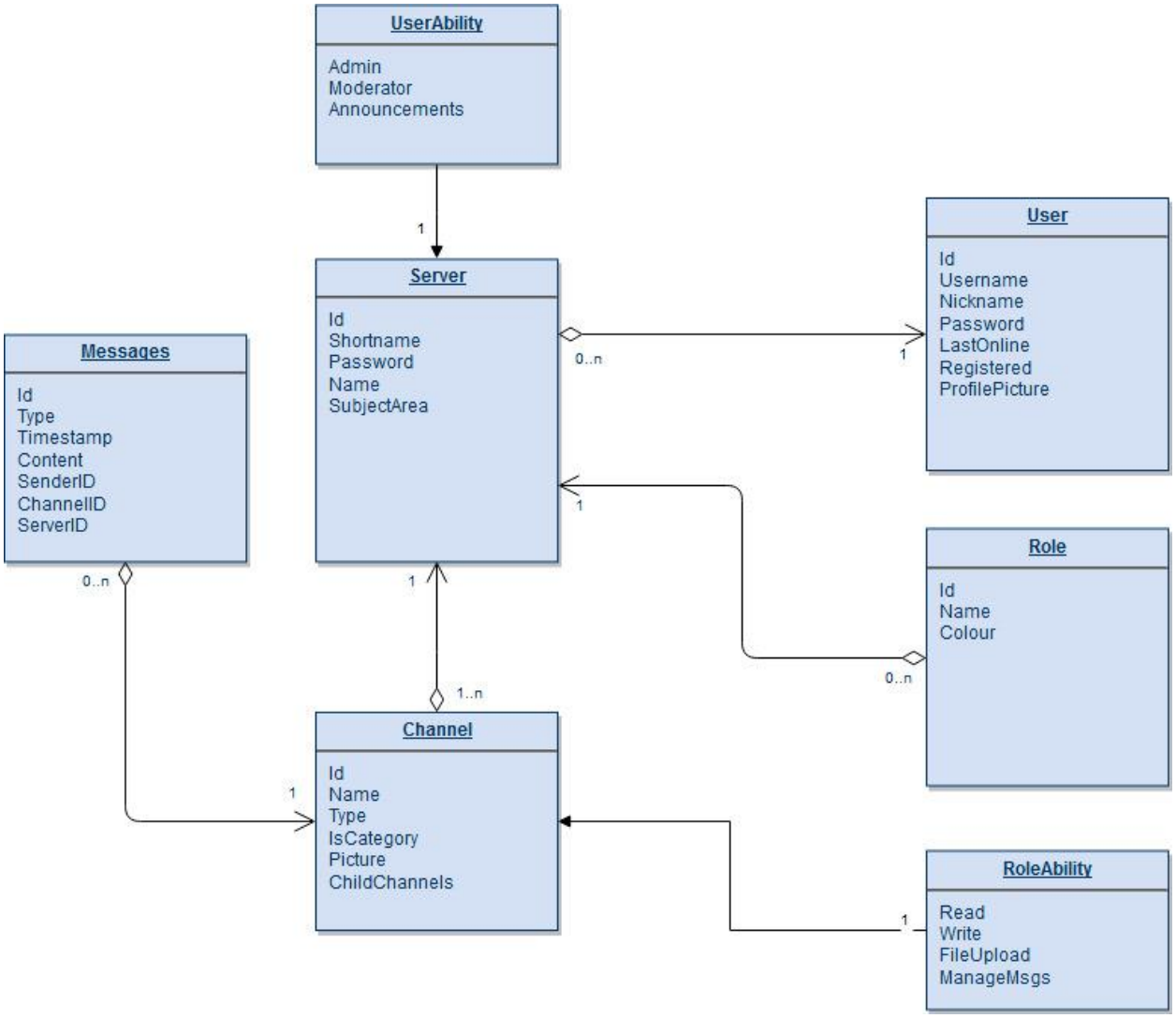
Im folgenden sind zwei Screenshots aus der vorläufig fertigen Anwendung zu sehen.



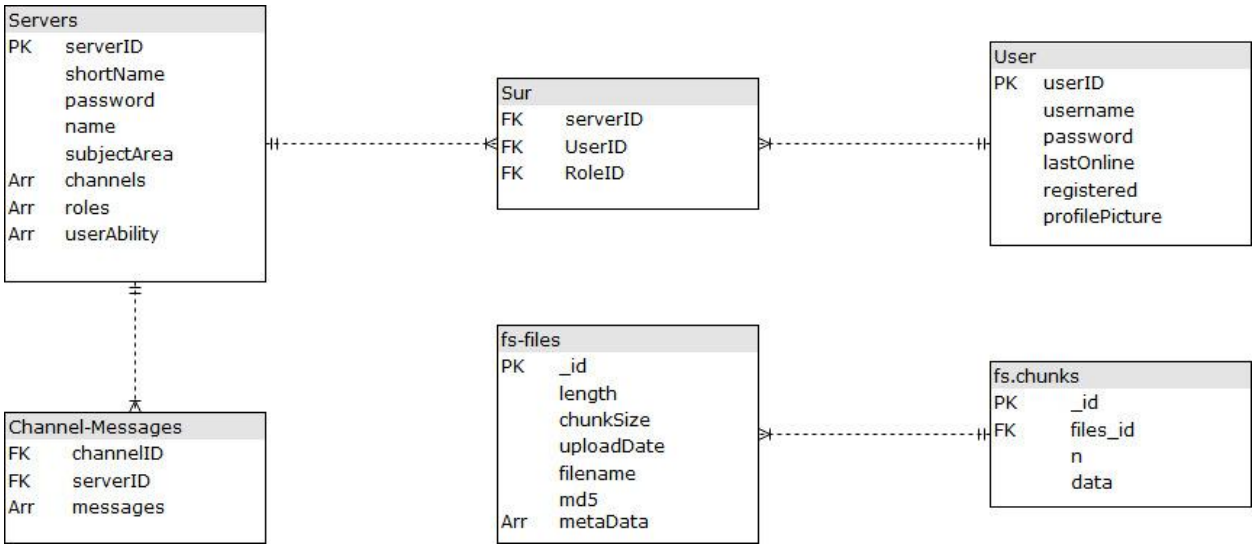

HCI
ALLGEMEIN
Neuigkeiten
GRUNDLAGEN
Skripte
Übungen
Beispiele
FRAGEN & ANTWORTEN
Anonym
Diskussion
Neuigkeiten
KLAUSUR
24.01.2019
ÜBUNGEN: ABGABETERMIN
12.12.2018
Dieter Wallach Professor 17.01.2019, 13:37
g
Dieter Wallach Professor 11.02.2019, 19:47
kl
Nachricht an @Neuigkeiten
PROFESSOR
Dieter Wallach
STUDENT
Sebastian Schuler
Philipp Spandl
Caroline Miller

6 Implementierungsphase

Zuerst wurden die Klassenobjekte User, Server (bzw Kurs für den User), Channel, Rolle und Nachricht erstellt. Dazu noch die benötigten Kombinations Klassen, die nur IDs beinhalten und zur Verbindung von n:n beziehung dienen.
Die extra Klassen UserAbility und RoleAbility beinhalten Arrays für jede Funktion wie z.B. Lesen und Schreiben in einem spezifischen Channel. In die RoleAbility Arrays werden Rollen IDs benutzt, und in UserAbility die jeweiligen User IDs (Bsp. ID des Professors könnte zu Admin gesetzt werden, Assistent Moderator, etc). Dies ist allerdings nur die Struktur für die Applikation, auf der Datenbank wird eine Andere, effektivere, Struktur verwendet.



Das daraus erstellte Datenbank Modell sieht etwas anders aus um den Zugriff vom Server zu erleichtern. Die Datenbank enthält nur zwei richtige Tabellen, zwei Kombinations Tabellen und zwei Datei Tabellen. Sie wurde realisiert mit MongoDB, das wir aufgrund der hohen Kompatibilität mit Node.js gewählt haben.



Im Gegensatz zu klassischen Datenbanksystemen unterstützt MongoDB Objekte, wie sie in Javascript existieren. Das bedeutet Arrays sind erlaubt (oben gekennzeichnet mit Arr). Die Tabellen Server und User sind die wichtigsten und werden als erstes von der DB geladen. Sur (kurz für Server-User-Rolle) gibt an welcher User in welchem Server ist und Channel-Messages ist eine Kollektion aller Nachrichten eines bestimmten Channels in einem bestimmten Server.
Die zwei extra Tabellen fs.files und fs.chunks sind zum speichern von Dateien da, files enthält die metadaten und chunks die Datei aufgesplittet in chunks deren Größe man in den Einstellungen der DB ändern kann.

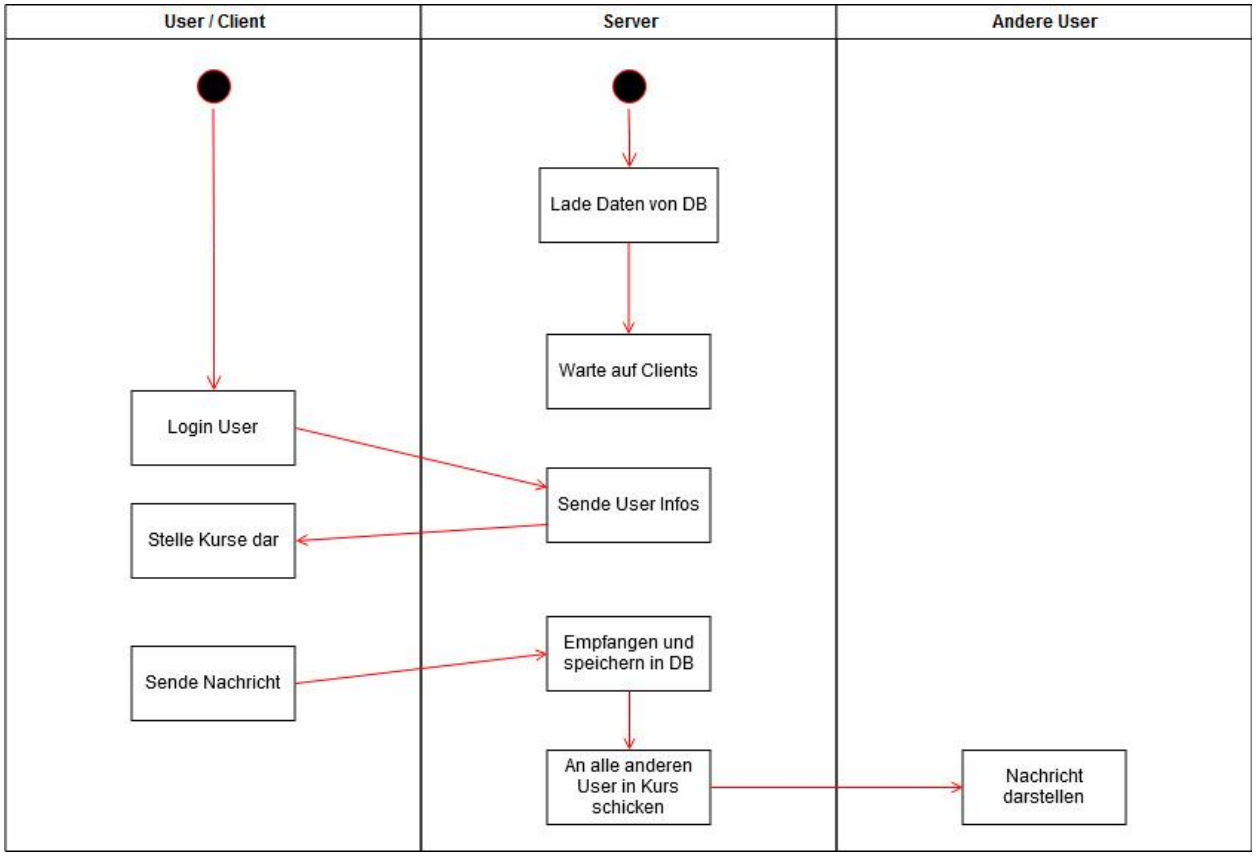
6.1 Server

Der Server basiert auf Node.JS und Express. Er verbindet sich mit der Datenbank und lädt alle verfügbaren Objekte damit sie bereit sind sobald ein User sich verbindet. Sollte ein Objekt verändert oder neu erstellt werden, muss dies nur an die DB gesendet werden und kann lokal auf dem Server übernommen werden. Sensitive Daten wie Passwörter werden mit Hilfe von BCrypt.js verschlüsselt, auf der DB abgelegt und erst vom Server wieder entschlüsselt. Ein großteil der Programm Logik befindet sich im Server wie z.B. Nachrichten Verteilung, Rechte Abfrage, Dateien aus DB laden / speichern und vieles mehr. Dadurch bleibt der Client unbelastet und muss sich nur um die Darstellung und die Kommunikation zum Server kümmern.
Nach Login eines neuen Clients werden diesem alle relevanten Informationen in einem Paket geschickt. Dazu gehören die Server / Kursliste des Users und dessen persönliche Daten (Passwörter werden von Objekten vor Senden entfernt).

Ausserdem verwaltet der Server die Server, User, Sur und Message Listen als Variablen. Dazu parallel werden ‘Änderungen auch immer in der DB gespeichert.

6.2 Client

Der Client basiert auf Electron, das eine stabile und schnelle Grundlage für eine Desktop Applikation bietet. Die eigentliche Benutzeroberfläche wurde mit Hilfe von Bulma erstellt und angepasst. Der Austausch von Client und Server ist Eventbasiert, das heißt es wird nur etwas gesendet wenn zB eine Nachricht gesendet oder empfangen wurde.



7 Fazit

Das Ziel dieser Anwendung war es einen MVP einer Kollaboration Anwendung zwischen Studenten und Professoren zu entwickeln. Dieser sollte als Minimum eine Chat Funktion bieten und die Möglichkeit Dateien hoch- und herunterzuladen.

Um dieses Ziel zu erreichen wurden als Erstes in der Analysephase, bereits existierende Plattformen, sowie Zielgruppen analysiert und entsprechende Anforderungen und Szenarien für diese Anwendung aufgestellt.

Anschließend wurden in der Entwurfsphase Methoden angewendet, um die grundlegende Architektur und das Design der Anwendung festzulegen, welches später in der Implementierungsphase als grundlegende Hilfestellung dienen soll.

In der Implementierungsphase wurde die letztendliche Anwendung mit verschiedenen Mitteln implementiert und zum Schluss getestet.

An erster Stelle steht hierbei die Zeitplanung, statt eines ausgereiften und übersichtlichen Zeitplans wurde sich lediglich an einzelne Termine gehalten und danach festgelegt was bis zum nächsten Termin ungefähr fertig sein sollte. Dies ähnelt zwar etwas dem Agilen Vorgehen, jedoch wurde dies nicht im vornherein festgelegt.

Die Analysephase hätte etwas ausführlicher gestaltet werden können, es wurden zwar Anforderungen aus Analysen aufgestellt, jedoch wurde meist alles etwas schwammig formuliert. Es hätte außerdem mehr mit der Zielgruppe kommuniziert werden können.

Die Entwurfsphase welche eigentlich als grundlegende Stütze für die Implementierung dienen soll, wurde etwas spät fertig gestellt, was zur Folge hatte, dass die Implementierung zum Teil bereits angefangen wurde, bevor der eigentliche Entwurf fertig war. Außerdem hätte auch in dieser Phase etwas mehr mit den Zielgruppen kommuniziert werden sollen.

In der Implementierungsphase war oftmals nicht ganz klar wer an was arbeiten soll, was zur Folge hatte, dass zum Teil an der gleichen Stelle gearbeitet wurde und der Fortschritt dadurch verlangsamt wurde. Außerdem gab es Anfangs Unklarheiten zu den genutzten Frameworks, Programmiersprache etc., welche durch bessere interne Kommunikation hätte gelöst werden können.

Schlussendlich wurde das Ziel eines MVP leider nicht erreicht, da es möglich ist Dateien nur hochzuladen, aber nicht runterzuladen.

7.1 Ausblick

Zukünftig sollte die Anwendung an der gesamten Hochschule auf einem zentralen Server laufen, dabei sollten Professoren die Möglichkeit haben Ihre Kurse dynamisch zu erstellen um so Studenten verschiedenste Dokumente bereit zu stellen. Es soll außerdem möglich sein Lerngruppen zu bilden und es sollte möglich sein sich in diesen Gruppen zu organisieren.

Außerdem sollte das System noch Quality of Life Verbesserungen erhalten, wie zum Beispiel eine Drag and Drop Funktion für Dateien, Vorlagen für die Erstellung von Kursen und allgemeine Verbesserung der Nutzerfreundlichkeit.

Für den Login sollte künftig der campusboard-Account bzw. der Hochschul-Account nutzbar sein.

8 Anteil an Projektdurchführung

Laut dem Modulhandbuch stehen dem Studienprojekt ca. 210 Stunden Arbeitsaufwand zu, die bei diesem Projekt erreicht und bei einem ungefähren tatsächlichen Arbeitsaufwand von 270 Stunden überschritten wurden.

Caroline Miller:
Design, Projektmanagement, Darstellungs-Script
Arbeitsaufwand ca. 80 Stunden

Sebastian Schuler:
Client-Server Kommunikation, Server-Implementierung, Design
Arbeitsaufwand ca. 110 Stunden

Philipp Spandl:
Server-Implementierung, Design, Projektmanagement
Arbeitsaufwand ca. 80 Stunden

9 Ehrenwörtliche Erklärung

Hiermit erklären wir,

Caroline Miller,

Sebastian Schuler,

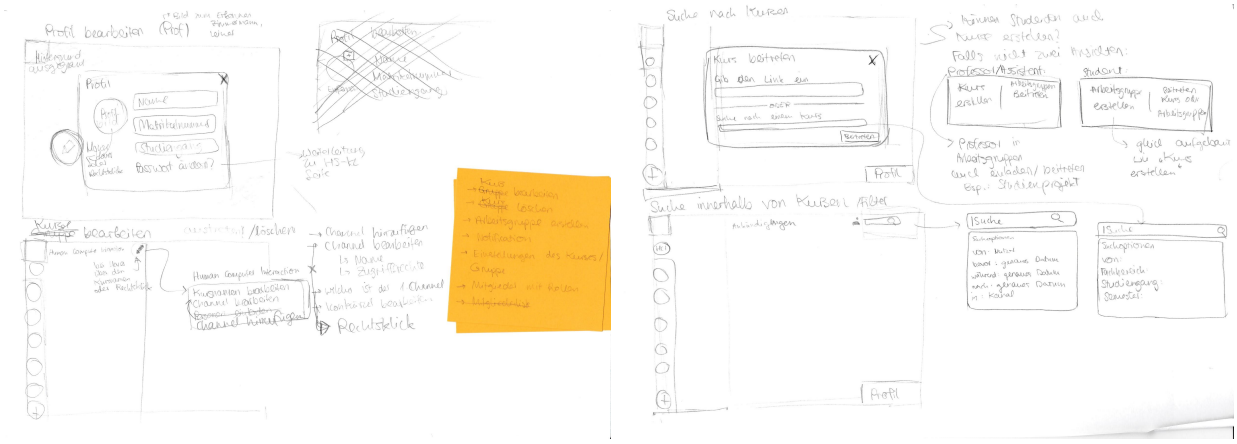
Philipp Spandl,

ehrenwörtlich, dass wir die vorliegende Projektarbeit zum Studienprojekt selbstständig und ohne fremde Hilfe angefertigt haben und keine anderen als in der Abhandlung angegebenen Hilfen benutzt haben;

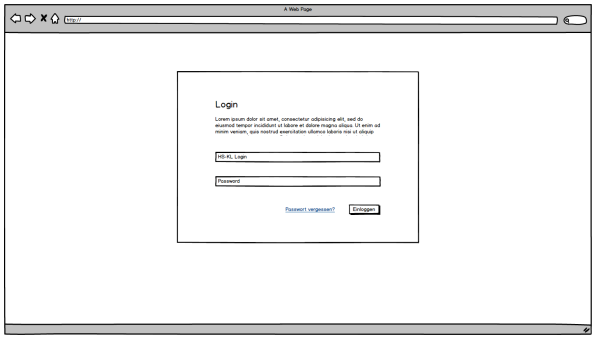
Wir sind uns bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

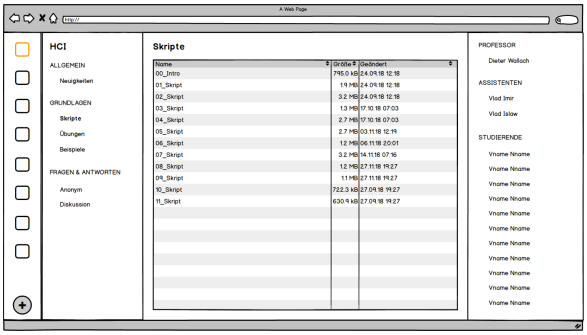
10 Anhang

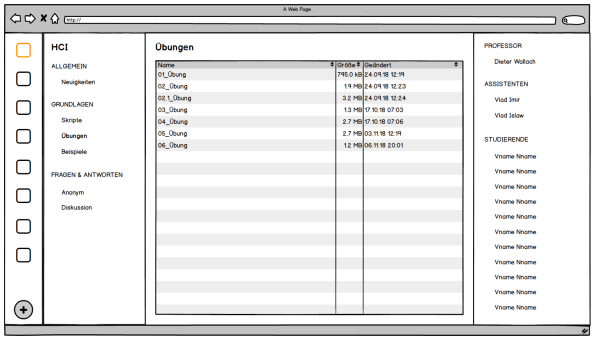
10.1 Scribbles

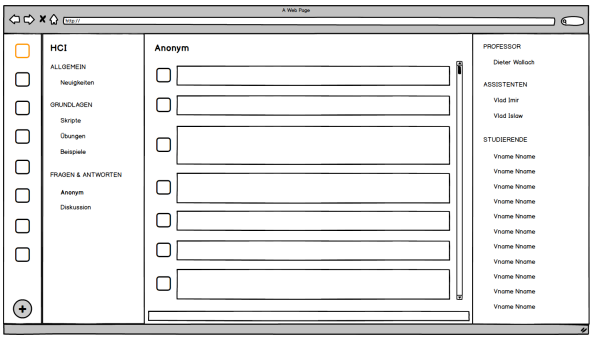


10.2 Mockups

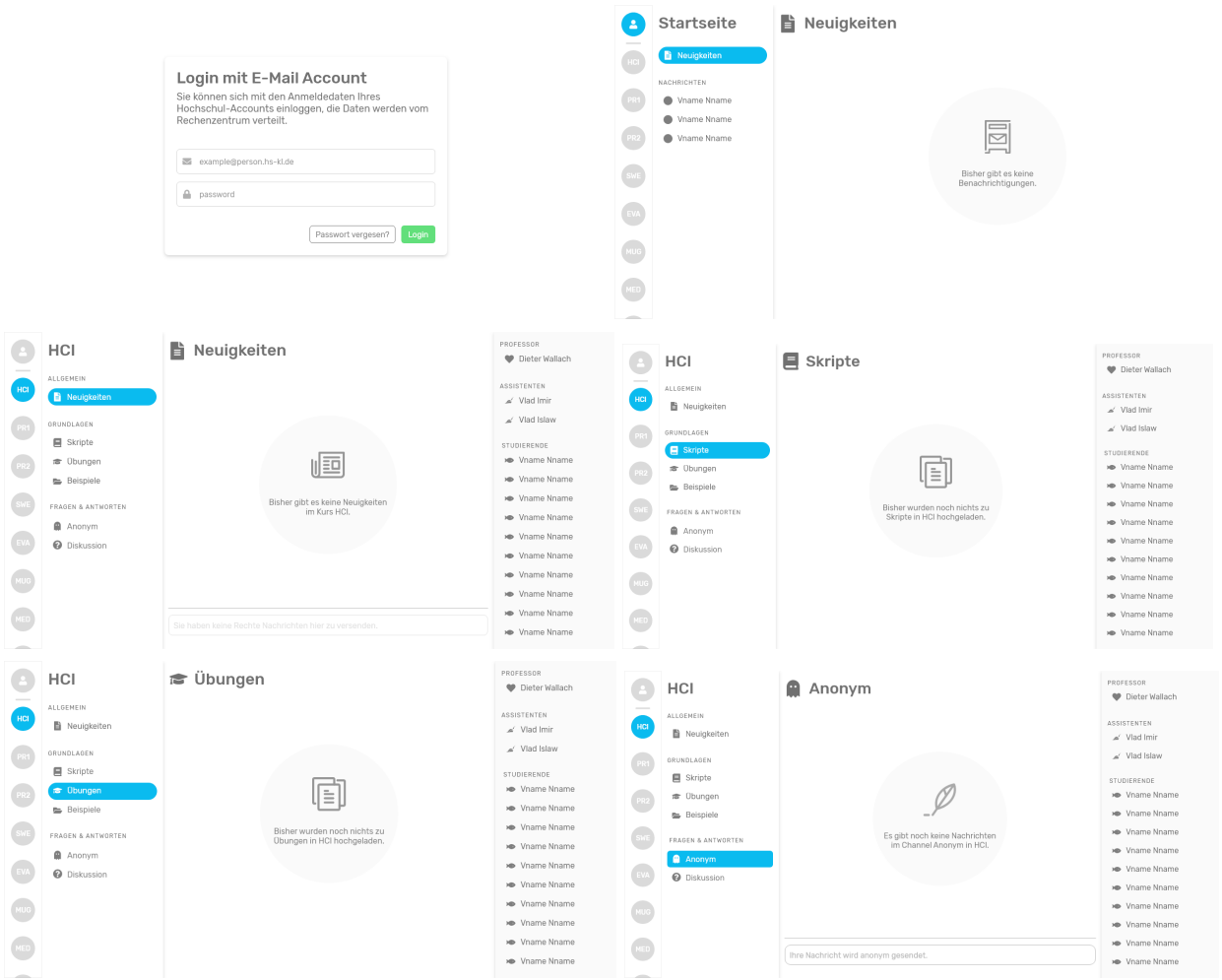


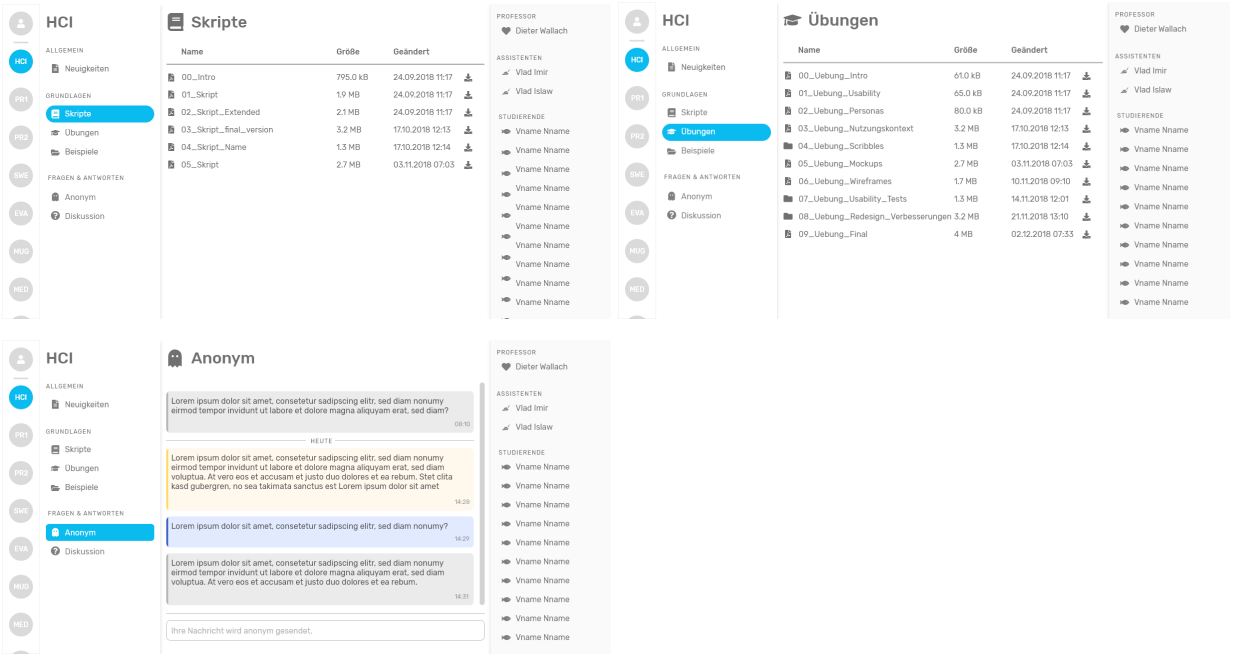




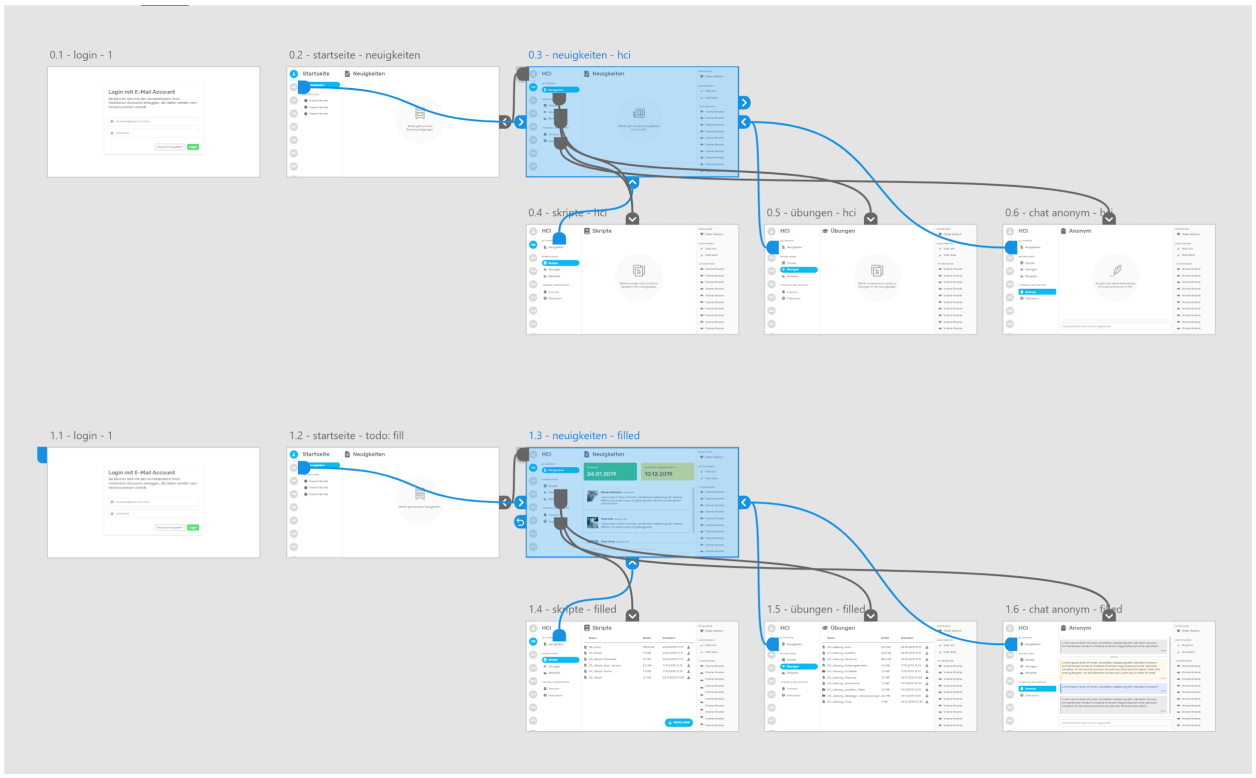


10.3 Wireframes





10.4 Prototyp



11 Quellenverzeichnis

Client Framework: Electron
https://electronjs.org/

Datenbank: MongoDB
https://www.mongodb.com/

Server: Expressjs
http://expressjs.com/

Server Nachrichten Austausch: Socket.io
https://socket.io/

DB Verschlüsselung: bcrypt
https://www.npmjs.com/package/bcrypt

UI CSS Framework: Bulma
https://bulma.io/

Unique IDs für Objekte:
https://github.com/dylang/shortid

Datum / Zeit Formatierung: momentjs
http://momentjs.com/

Icons: Font Awesome
https://fontawesome.com/

Toni Steimle, Dieter Wallach. Collaborative UX Design. Proto-Persona
Verfügbar: http://www.collaborative-uxdesign.com/scoping/protopersonas

Pexels. CC0-License
https://www.pexels.com/photo/focus-photo-of-brown-sheep-under-blue-sky-227691/
https://www.pexels.com/photo/white-sheep-during-daytime-59863/
https://www.pexels.com/photo/agriculture-animal-animal-photography-bleat-459215/