Technology
Arts Sciences
TH Köln

Entwicklungsprojekt Interaktive Systeme
Wintersemester 2014 / 2015

Team 33:
Christian Poplawski, 11088931
Timo Mämecke, 11088767

Projektbegründung

Betreut von:
Prof. Dr. Kristian Fischer
Prof. Dr. Gerhard Hartmann
Sheree Saßmannshausen
Ngoc-Ahn Dang

Inhaltsverzeichnis

1. MS1

1.1. Projektexposés

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS1/Expos%C3%A9%20Studiendiskussion.pdf

Zu Beginn wurden Exposés zu zwei Projektideen verfasst.

1. Eine Anwendung, mit der Videos, die über den Tag vom Nutzer aufgenommen wurden, automatisiert als komplettes, semiprofessionelles Video geschnitten werden und als soziale Komponente mit Familie, Freunden und Bekannten teilbar ist.
2. Eine Plattform, mit der einem Nutzer Links zu Hypermedia-Inhalten gesendet werden können, um diese mit einer zentralen Plattform geordnet sammeln zu können.

Bei beiden Ideen mangelte es sowohl an echter Problematik als auch an Verteiltheit der Anwendungslogik. Daher wurde eine weitere Projektidee formuliert, welche nicht zum finalen Projekt wurde, allerdings die Grundzüge zur letztendlichen Projektidee schon beinhaltete. Es sollte eine Studieninformationsplattform für Studieninteressierte geben, da dies aus eigener Erfahrung vor dem Studium ein Problem dargestellt hat. Nach Vereinheitlichung und Umstrukturierung der Idee wurde festgehalten:

> **Problematik:** Als Studieninteressierter suche ich nach relevanten Informationen und Erfahrungen von Studenten, und nicht nach generellen Daten.
> **Lösungsidee:** Eine Plattform als Messageboard zur Informationsbeschaffung und Kommunikationsweg zwischen Student und Studieninteressent.

Ziel der **Verteiltheit** ist vorallem die Verifikation von Studenten, um diese als authentische Informationsgeber kenntlich machen zu können. In der späteren Umsetzung sollen verschiedene Anwendungskomponenten miteinander agieren, um dies zu erreichen. Technisch ist eine In-App-Verifikation mit Hyperlink geplant, wobei auch noch andere Möglichkeiten in Betracht gezogen werden können.
Weiterhin wäre die Suche bzw. das Erkunden von Studiengängen nach Geo-Position als verteilte Anwendungslogik denkbar. Es soll keine verteilte Anwendungslogik erzwungen werden.

2. MS2

2.1. Umfrage

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Umfrage%20Studiengangsfindung.pdf

Um die Problematik zu unterstützen und bessere Ziele zu finden, wurde eine Umfrage geführt. Die Umfrage hatte die Thematik der Studiengangsfindung mit Schwerpunkt auf Schwierigkeiten und Zufriedenheit. Bei der Umfrage haben 113 Personen teilgenommen.

Die genauen Ergebnisse der Umfrage sind in MS2 zu finden.

Ziel der Frage “Ich bin… (Student, auf der Suche, Ehemaliger, …)” war es einen Überblick über die Teilnehmer zu bekommen, um die nachfolgenden Fragen besser interpretieren zu können.

18,6% der Teilnehmer haben schon einen Studiengang begonnen und abgebrochen. Die meisten informieren sich über die Studiengänge im Internet. Weniger als die Hälfte davon hat sich mit Studenten unterhalten. Bei der Frage “Wie hättest du dich außerdem gerne informiert?” war **das Gespräch mit Studenten** die zweitbeliebteste Antwort, neben dem Besuch von Hochschulen. Es lässt sich verknüpfen, dass sich viele im Internet informieren, sich nicht mit anderen Studenten unterhalten haben, es aber gerne würden.

67,3% der Teilnehmer haben einen Studiengang an verschiedenen Hochschulen verglichen.

Über die Hälfte aller Teilnehmer (55,4%) haben bei der Zufriedenheit mit der Studiengangsinformation auf einer Skala von 1 (sehr unzufrieden) bis 10 (sehr zufrieden) einen Wert von 5 oder weniger vergeben.

Als Resultat der Umfrage lässt sich sagen, dass die im Exposé beschriebene Problematik verifiziert wurde und auf großes Interesse stößt.

2.2. Zielhierarchie

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Zielhierarchie.pdf

Im Hinblick auf die Projektidee hat sich als strategisches Ziel nur ein einzelnes, jedoch sehr wichtiges strategisches Ziel ergeben. Auf diesem Ziel wurde eine hierarchische

Struktur der taktischen und operativen Ziele gebildet, welche alle wichtigen Faktoren der Anwendung abbilden.

Dabei wurde SMART angewendet[1], um klare und erfüllbare Ziele zu formulieren. Der terministische Aspekt wurde außen vor gelassen, er ist für die Zielhierarchie in diesem Projekt nicht notwendig.

2.3. Marktrecherche

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Marktrecherche.pdf

Insgesamt wurden 8 mögliche Konkurrenzprodukte genauer betrachtet, davon befinden sich 6 im direkten Kontext der Domäne und 2 mit vergleichbaren Funktionalitäten, die sich allerdings außerhalb der Domäne befinden. Festgestellt wurde, dass die Hauptfunktionalität zur Informationsbeschaffung, Fragestellung, Interaktion und Diskussion auf keiner der untersuchten Plattformen möglich war.

Alle betrachteten Plattformen waren Internetseiten. Die Recherche nach Anwendungen, wie z.B. mobilen Anwendungen, blieb erfolglos. Hier wurden nur sehr wenig weitere gefunden, welche außerdem keine anderen – wenn sogar weniger – Funktionalitäten besitzen.

Eine noch umfangreichere Marktrecherche wurde nicht mehr durchgeführt. Sollte bei der Recherche nach ähnlichen Anwendungen so viel Zeit verstreichen, dann wird die Nutzerbasis ebenfalls nicht nach weiteren, anderen Plattformen suchen.

2.4. Domänenrecherche

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Dom%C3%A4nenrecherche.pdf

Zur Domänenrecherche wurde die Domäne der *Studieninformation* unterteilt in generische Informationen und einen persönlicheren Aspekt auf Erfahrungsbasis von Studierenden und eigenen Fragen. Die generischen Informationen sind nicht nur im Überfluss vorhanden, sondern auch nicht das ausschlaggebendste Element für einen Studieninteressierten. Die Anwendung fokussiert sich letztendlich auf jenen zweiten Teil dieser Domäne.

[1] University of Virginia, “Writing S.M.A.R.T. Goals”, abgerufen am 5. November 2015 http://www.hr.virginia.edu/uploads/documents/media/Writing_SMART_Goals.pdf

Vorgänge und Fragestellungen wurden auf eigener, persönlicher Erfahrung formuliert, was noch durch Belege gefestigt werden muss. Hierfür könnte sich auch eine Umfrage unter Studieninteressierten und Studenten eignen.

2.5. Alleinstellungsmerkmale

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Alleinstellungsmerkmale.pdf

Aus den zusammengefassten Ergebnissen von Zielhierarchie, Marktrecherche und Domänenrecherche konnten Alleinstellungsmerkmale gegenüber anderer Produkte formuliert werden. Dafür wurden die Vor- und Nachteile der Marktrecherche mit den Problemen aus der Domänenrecherche kombiniert und zuletzt mit den Zielen der Zielhierarchie verglichen.
Die wichtigsten Merkmale hierbei ist die Strukturierung als Board-System und vorallem die Verifikation der Nutzer als Studenten.

2.6. Methodischer Rahmen

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/MethodischerRahmen.pdf

Abwägung der nutzbaren Methodiken
Als mögliche Methodiken für dieses Projekt wurden *Usage Centered Design*, *Discount Usability Engineering* und *User Centered Design* in Erwägung gezogen.

Es wurde recht schnell deutlich, dass Usage Centered Design für uns keine Zufriedenstellende Lösung sein würde. Zum einen stellte sich die Recherche nach Material zu dieser Vorgehensweise als schwierig heraus, als einzige verlässliche Quelle ist das Buch von Constantine und Lockwood[2] zu nennen. Da keines der Teammitglieder bisher Erfahrungen mit Usage Centered Design gesammelt hat würde die Verwendung dieser Methodik einen langwierigen Lernprozess mit sich bringen, der wiederum den Projektablauf verzögern wurde.
Nun ist ein langer Lernprozess keine legitime Begründung dafür, eine Methodik nicht zu benutzen. Hier ist daher außerdem anzumerken, dass sich der Vorgehensmodell des Usage Centered Design primär mit Tasks und Usage Patterns befasst, unsere Anwendung allerdings keine außergewöhnlichen Tasks aufweist.

---

2 https://books.google.de/books?id=Y0ic4kK9E7cC&lpg=PT9&ots=xkdDPFd7i6&dq=usage%20centered%20design&lr&pg=PP1#v=onepage&q&f=false

Als Beispiel sei hier das Kommunikationssystem genannt. Es gibt bereits etablierte Arten von Kommunikationssystemen (Chats, Boardsysteme, …), wovon eines mit einer Begründeten Entscheidung für unsere Anwendung gewählt werden muss. Jedoch besteht keine Notwendigkeit, den Task der Kommunikation näher zu analysieren.

Discount Usability Engineering ist aufgrund der kleinen Teamgröße eine reizvolle Methodik. Da wir uns außerdem in der Domäne auskennen (wir sind Studenten und vor nicht allzu langer Zeit waren wir auch Studieninteressierte) ist die Wahl dieser Methodik nicht sehr abwegig.
Jedoch ändert sich die Technik im Bereich Mobile sehr schnell und im schlechtesten Fall liegen bis zu 5 Jahre zwischen den momentanen Studieninteressierten und uns, wir können uns also nicht darauf verlassen, dass Erfahrungen aus unserer Zeit noch immer Bestand haben.

Alle oben angesprochenen Punkt brachten uns zu der Entscheidung, ein Benutzerzentriertes Vorgehensmodell zu wählen. Wir können nicht sicher sein, wie sich die Benutzer verhalten, User Research ist also unabdingbar. Diese Annahme wir dadurch verstärkt, dass Studieninteressierte und Studenten breit gefächerte Hintergründe haben, die einzige Gemeinsamkeit ist ein gewisser Bildungsstand.

Konkrete Techniken
Zur Benutzermodellierung werden zunächst User Profiles angelegt, welche eine gewisse Kategorisierung in der breiten Masse von potenziellen Benutzern schaffen soll. Da von uns durchgeführte Interviews und Umfragen nur eine begrenzte Reichweite haben werden außerdem offizielle Statistiken genutzt, um die User Profiles so möglichst repräsentativ zu gestalten.
Das verwenden von Personae ermöglicht uns ein autonomeres und somit effizienteres Arbeiten, da diese zu jeder Zeit verfügbar sind und für das Arbeiten mit ihnen keine lange Wartezeit in Kauf genommen werden muss.

Das Arbeiten mit Use Case Briefs erlaubt es uns, zunächst einen Überblick über alle Funktionen der Anwendung zu gewinnen. Anhand dieser Use Case Briefs und den aus der Benutzermodellierung gewonnenen Ergebnissen können dann User Goal Use Cases entwickelt werden, auf dessen Umsetzung wir uns vorrangig konzentrieren.

Als Technik zur Evaluation wird das Think Aloud angewendet. Diese Technik benötigt keine spezielle Infrastruktur oder andere Artefakte (wie z.B. Eye-Tracking oder vorgefertigte Fragebögen), sie ist unabhängig von der Art des Prototypen und in einem Team von 2 Leuten durchführbar (wenn auch nicht optimal).

Bei Prototypen wird darauf geachtet, dass diese wann immer es möglich ist am tatsächlichen Gerät evaluiert werden, da die Evaluation eines Smartphone-Interfaces an einem Computerbildschirm oder auf Papier verfälscht werden kann. (Der Benutzer kann die Entfernung vom Geräte zu seinen Augen variieren, ein tatsächliches Gerät wir in der Hand gehalten, es kann gedreht werden, …)

2.7. Kommunikationsmodell

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Kommunikationsmodell.pdf

Die Definition des Kommunikationsmodells erforderte einige Abwägungen. Durch das von uns identifizierte Problem, dass es Studieninteressierten schwer fällt, eine Kommunikation mit bereits Studienreden eines bestimmten Studienganges oder dessen Absolventen aufzubauen, war bereits ein abstrakter Rahmen gesetzt. Studieninteressierten musste eine einfache Kommunikation mit Studenten und Absolventen möglich sein. Weiterhin musste diese Kommunikation Thematisch abgegrenzt und einem bestimmten Studiengang an einer bestimmten Hochschule zugeordnet sein. Für die konkrete Umsetzung des Kommunikationsmodells gab es einige Ansätze.

Eine der ersten Überlegungen war ein *privates Nachrichtensystem* mit jeweils zwei Teilnehmern. Obwohl dieses für Studieninteressierte die Möglichkeit bieten würde, einen persönlichen Mentor zu haben, hätte dieses Modell einige Chancen verschwendet und das Projekt wäre zu nicht viel mehr als einem Instant-Messenger mit Hochschule-Thematisierung geworden. Außerdem wären Studenten aufgrund der fehlenden öffentlich zugänglichen Informationen immer wieder mit den gleichen Fragen konfrontiert, was vermutlich die Langzeitmotivation zur Nutzung der Plattform und damit zu einem Versagen des Systems führen würde.

Auf den Nachteilen des privaten Nachrichtensystems aufbauend entstand die Überlegung eines *Abonnement-Modells,* bei dem die Benutzer Studiengänge Abonnieren müssten, um Zugang zu den Konversationen innerhalb des Studienganges zu bekommen. Innerhalb des Studienganges würden hier Konversationen für alle Mitglieder einsehbar sein.
Nach einer kritischen Reflektion wurde aber auch dieses Modell nicht als optimal angesehen. Das Abonnieren wäre eine künstliche Hürde für die Benutzer, da diese mit keinerlei Anforderungen einher geht (Verifizierung, Monetarisierung etc.), sie würde lediglich dazu führen, dass auch rein konsumierende Nutzer ein Benutzerkonto besitzen müssten, um ihnen ihr abonnierten Studiengänge zuordnen zu können.

Diese Restriktionen wurden verworfen und daraus ergab sich ein komplett öffentlich einsehbare Kommunikationsmodell. Dieses bietet die Möglichkeit, unterschiedliche Perspektiven zu einem bestimmten Thema in einer Konversation zu vereinen. Falsche Informationen und andere Denkansätze sind so einfacher zu vermitteln. Jede Konversation beginnt hier mit einem startenden Gedanken oder einer Frage.

Das schreiben von Erfahrungsberichten über den Studiengang (was verifizierten Studenten vorbehalten ist) wurde in das Modell implementiert, um auch rein lesenden Benutzern einen Mehrwert schaffen zu können. Diese Art der Beiträge führt auch zum Konsens über einen Studiengang, der Studieninteressierten gleich zu Beginn einer Suche zeigt ob es sich lohnt, sich weiter mit dem Studiengang an der bestimmten Hochschule zu befassen.

2.8. Risiken

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Risiken.pdf

Da die beiden Projektteilnehmer seit 5 Semestern häufig zusammen Projekte bearbeiten war das finden von typischen persönlichen Problemen (auch solchen, die speziell in dieser Konstellation auftreten) eher einfach. Die beiden Teammitglieder kennen die Stärken und Schwächen des anderen bereits recht und und auch das sozio-technische System, das sich währen der Arbeit zwischen den beiden aufspannt ist ausreichend bekannt.
Risiken wie schwerwiegende Missverständnisse oder Streitigkeiten, die den Projektablauf bedrohen konnten daher mit einem gewissen Grad an Sicherheit ausgeschlossen werden, wenngleich diese Risiken nie komplett eliminiert werden können.
Auch ließen sich einige bekannte Risiken, die zu einer hohen Wahrscheinlichkeit auftreten würden leichter identifizieren und Mittel und Methoden zu deren Minimierung finden.
Weiterhin bemühten wir uns, auch Ereignisse in die Risiken mit aufzunehmen, auf die wir keinen Einfluss nehmen können.

Da der Abgabezeitpunkt für das Projekt fest gesetzt ist und sich nicht verschieben lässt war eine Verlängerung der Laufzeit des Projektes keine Option für die Reaktion auf das tatsächliche Eintreten eines Risikos. Auch eine Verlängerung der Täglichen Arbeitszeit wäre hier keine valide Lösung, da Studien zeigen[3], dass die Produktivität hier längerfristig gesehen abnimmt.
Die einzig effektive Möglichkeit stellt die Verkleinerung des Projektumfangs dar. Im Notfall müssten geplante Funktionalitäten gestrichen werden, was für eine verkürzte Projektdauer zu sorgen.

Da die meisten von uns identifizierten technischen Risiken sich in Proof of Concepts überprüfen lassen wurden diese in den Risiken nur kurz aufgelistet und in der Spezifikation der PoCs genauer definiert.

[3] http://alifeofproductivity.com/number-of-hours-work-a-week-to-be-the-most-productive-35/#

2.9. Proof of Concepts

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/SpezifikationderPoCs.pdf

Wie bereits erwähnt ergeben sie die Proof of Concepts aus den technischen Risiken. Hier wurden jedoch nur Risiken einbezogen, die sich in einem klar definierten zeitlichen Rahmen überprüfen ließen und für die eindeutige Fail- und Exit-Kriterien definiert werden konnten.
So konnte beispielsweise das Risiko “Probleme bei der Clientseitigen Programmiersprache” nicht als Proof of Concept abgebildet werden. Weder war hier klar, wie lange es dauern würde zu einem Ergebnis zu kommen, noch gibt es einen klaren Zustand ab dem man sicher sagen kann, dass es zu keinen Problemen mit der Programmiersprache kommen wird.

Bei der Spezifikation der PoCs wurde außerdem darauf geachtet, dass nur tatsächlich ungewisse Sachverhalte einbezogen werden, um möglichst effizient zu arbeiten.
Hier wäre zum Beispiel die Überprüfung der Übertragung von Dateien im JSON-Format zwischen den Sprachen Node.js und Java zu nennen. Da hiervon die Kommunikation zwischen Client und Service abhängt ist diese Funktionalität kritisch für das Projekt. Aufgrund von persönlichen Erfahrungen und unterstützt von einer kurzen Recherche wurde jedoch klar, dass weder das Senden von JSON-Formaten von Node.js Applikationen, noch das empfangen von JSON-Daten in Java-Applikationen ein Problem darstellt. Beide Sprachen unterstützen das Format nativ.

2.10. Architekturdiagramm und -begründung

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Architekturdiagramm.pdf
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS2/Architekturbegr%C3%BCndung.pdf

Als Architekturdiagramm wurde zuerst ein eigener, nicht-standardisierte Diagrammtyp verwendet, mit Elementen aus Sequenz-, Ablauf- und Verteiltheitsdiagrammen. Dies hat uns einen Überblick über die Funktionen, den Ablauf der Funktionalitäten und der Verteiltheit gegeben. Nach der Besprechung des Diagramms wurde uns geraten, Anwendungsdiagramm zu erstellen, was die Komponenten als Blackbox betrachtet.

Die verteilte Anwendung ist eine typische Client-Server-Architektur. Eine Peer-to-Peer Anwendungsverteilung ist durch die Art der Anwendungsdaten gänzlich ungeeignet.

Die Datenübertragung verläuft – wie üblich für solche Anwendungstypen – über HTTP. Als Datenformat wurde JSON gewählt. Dies besitzt zwar kein Schema, ist allerdings sehr datensparsam und in heutigen Anwendungen sehr häufig zu finden. XML mit einem passenden Schema wurde auch in Betracht gezogen. Dabei hätte man sogar abonnierbare Feeds erstellen können, möglicherweise mit AtomPub, allerdings sind Datenstrukturen in XML sehr groß und für mobile Zwecke ungeeigneter als JSON. JSON ist im Vergleich zu XML auch einfacher serialisierbar und deserialisierbar.

Die REST Schnittstelle soll in Richardsons Maturity Model Level 3 umgesetzt werden. Für einen Dienstnutzer wird es damit einfacher Links zu Ressourcen aufzurufen und mögliche Operationen einzusehen.

3. MS3

3.1. Dokumentation der PoCs

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/tree/master/MS3/PoC/Dokumentation

Für die Dokumentation wurden gezielt diejenigen PoCs gewählt, deren Versagen das Umsetzen der Anwendung so nicht möglich gemacht hätte und zu einem erheblichen Verlust an Anwendungslogik geführt hätte.

App-Links
Diese waren notwendig, um eine Verifizierung am Client ohne zusätzliches Web-Interface umsetzen zu können. Ziel hierbei war es, einen Link aus einer Mail am Client zu öffnen und die URL zur Weiterverarbeitung an die Anwendung weiter zu reichen.
Im Fall des Nicht-Erfüllens hätte hier ein Web-Interface für die Verifizierung entwickelt werden müssen, was sowohl zusätzliche Zeit in Anspruch genommen, als auch einen Teil der Anwendungslogik eliminiert hätte.

Service Datenstruktur
Die Datenstruktur für Beiträge ist am Service ist bewusst so aufgebaut, dass dem Client ein nicht unerhebliches Maß an Anwendungslogik zu teil wird, bevor dieser zur Präsentationslogik übergehen kann.

Mailverifizierung
Für die Verifizierung von Studenten wurde eine Liste mit TLDs aller Deutschen Hochschulen benötigt. Tatsächlich existierte diese so nicht und wir mussten eine eigene Lösung entwickelt, die regelmäßig die TLDs von einer verlässlichen Quelle erhält. Hier wurde zusätzliche und nicht geplante Anwendungslogik implementiert, allerdings liegt diese beim Service.

3.2. Benutzermodelle

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/tree/master/MS3/Benutzermodellierung

Für die Benutzermodellierung wurde in zwei Schritten vorgegangen. Zunächst wurden User Profiles geschrieben, auf denen aufbauend dann Personae erstellt wurden.

In die User Profiles flossen einerseits uns persönlich bekannte Daten ein, die sich aus unserem Umfeld ergaben, aber auch bundesweit erhobene statistische Daten. Das

einbeziehen von statistischen Daten ermöglichte es uns, bei den User Profiles einen Querschnitt aus ganz Deutschland zu erstellen, sodass die User Profiles nicht nur regional Gültigkeit besitzen.
Dies ist insofern relevant, dass sich beispielsweise schon die Schulsysteme, die zur Allgemeinen Hochschulreife führen von Bundesland zu Bundesland unterscheiden.

Beim erstellen der Personae wurde darauf geachtet, dass diese eine möglichst geringe Ähnlichkeit zu uns im echten Leben bekannten Personen aufweisen, sowohl durch offensichtliche Attribute wie das Foto, als auch durch abstraktere beschreibungen von Charakter und Leben.
Weiterhin wurden für Studeininteressierte mehr Personae erstellt, da diese unter den Stakeholdern unsere Hauptzielgruppe abbilden und wie durch die User Profiles ersichtlich sehr verschiedene Eigenschaften aufweisen können.

3.3. Benutzungsmodelle

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/tree/master/MS3/Benutzungsmodellierung

Auch für das erstellen der Benutzungmodell wurde in zwei Schritten vorgegangen. Zunächst wurden Use Case Briefs und anschließend User Goal Use Cases erstellt.

Die Use Case Briefs erlaubten es uns, einen relativ kompletten Überblick über alle Use Cases, die in der Anwendung eine Rolle spielen zu gewinnen währen der Aufwand hierfür vergleichsweise moderat war. Außerdem war die Liste der Use Case Briefs für die Benutzung in den frühen Stadien des Projekts deutlich übersichtlicher, da diese die Use Case recht stark abstrahieren und nur wenig auf die Details in den Abläufen eingehen.

Für die späteren Phasen das Projekts und die Entwicklung eines Prototypen waren die Use Case Briefs natürlich nicht detailliert genug, daher wurden basierend auf ihnen User Goal Use Cases entwickelt. Die Entscheidung für User Goal Use cases rührt dabei vor allem von deren narrativen Charakter her. Diese sollten einen Ausgleich zu den fehlenden Szenarien schaffen und können als hybrides Artefakt angesehen werden.
Sie sind sowohl für die Aufgabenbereiche der Mensch-Compurter-Interaktion nützlich, als auch als Leitfaden für die programmatische Entwicklung von Funktionen.
Einige der User Goal Use Cases konnten mehr als einen Use Case Brief abbilden, beispielsweise durch alternative Szenarien.

Obwohl es Ziel der User Goal Use Cases war, alle Use Case Briefs abzubilden und dies auch weitesgehend erreicht wurde, wurden nicht alle User Goal Use Cases auch für die Entwicklung des Prototypen verwendet. Hier war von Anfang an klar, dass die Zeit nicht zum umsetzen aller Szenarios reichen würde, daher lag der Fokus auf den Use Cases, die für das erstellen eines MVP notwendig waren.

3.4. Anforderungen

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS3/Anforderungen.pdf

Für die Anforderungen wurden zum ersten Teil *"Bedingungen für den Nutzer zur Zielfindung"* die operativen Ziele der Zielhierarchie betrachtet und in passende Bedingungen gruppiert bzw. abgeleitet.
Kritisch betrachtet hätte im Artefakt die *Zielfindung* stärker beachtet werden sollen. Es stellt sich die Frage, ob eine *Funktionalität zum Einloggen und zum Registrieren* wirklich eine Zielfindung darstellt.

Ein tatsächlicher Vertrag, der von der Anwendung erfüllt werden muss, sind die Geschäftsbedingungen[4] des Google Play Stores. Jene müssen beachtet werden, wenn die Applikation tatsächlich über den Play Store veröffentlich wird. Dies ist jedoch innerhalb dieses Projekts ein fiktives Szenario und wird nicht als Ziel gesetzt.

Das BSI hat für Webanwendungen Best Practices[5] veröffentlicht, an die sich Entwickler halten sollten. Im Artefakt wurde darauf zwar eingegangen, letztendlich wurde in der Anwendung die Sicherheit des Nutzerkontos nicht weiter berücksichtigt, da diese für die Bewertung des Projekts keine Rolle spielt. Um nicht mit jedem Aufruf Klartext-Passwörter zu übermitteln, wurde ein Token zur Authentifizierung eingebaut.

Barrierefreiheit ist nicht zwingend eine Fähigkeit, die erfüllt werden muss – sie sollte jedoch erfüllt werden. Für barrierefreiheie Anwendungen gibt es Richtlinien von Android[6] und auch eine ISO Norm 16071[7], welche die ISO 9241 um Barrierefreiheit erweitert. Dies geht mit dem dritten Punkt *"Mögliche Einschränkungen des Nutzers"* einher.

Einschränkungen des Nutzers können körperliche Einschränkungen oder Einschränkungen des Gerätes sein, das benutzt wird. Internetzugang und Softwareversion gehören zu den Einschränkungen des Gerätes. Diese Einschränkungen können von Nutzer zu Nutzer variieren und sogar bei einem Nutzer variieren, wenn sich beispielsweise die mobile Internetverbindung ändert.
Einschränkungen des Nutzers können bei unserer Applikation eine Farbenblindheit/ Sehschwäche oder auch Sehbehinderung sein. Jene werden durch die Richtlinien von Android größtenteils abgedeckt.

---

[4] https://play.google.com/intl/ALL_de/about/developer-content-policy.html
[5] https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Studien/WebSec/WebSec_pdf.pdf?__blob=publicationFile
[6] http://developer.android.com/training/accessibility/accessible-app.html
[7] http://www.iso.org/iso/catalogue_detail.htm?csnumber=30858

4. MS4

4.1. Datenstrukturen

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS4/Datenstrukturen.pdf

Passend zu MongoDB und mongoose wurden die Datenstrukturen gleich als mongoose-Schema angefertigt. Hier wurden während der Erarbeitung die Collections *Hochschule*, *Studiengang*, *Beitrag*, *Benutzer* und *Verifikation* herausgearbeitet. Dies geht fast mit den Ressourcen aus der WBA-Modellierung einher. Um die Datensätze besser abstrahieren und strukturieren zu können, wurde die Hochschule als eigener Datentyp vom Studiengang getrennt. Bei der Suche nach einem Studiengang (GET /courses) werden *Hochschule* und *Studiengang* zusammengeführt. Gut erkennbar wurde hier auch die Häufigkeit der Referenzierung innerhalb der Collections, für die die *populate*-Funktion aus mongoose zwingend benötigt wird. Erkennbar ist auch, dass die Collections hierarchisch aufeinander referenzieren, was einer Baumstruktur entspricht.

Die Beiträge wurden nicht in *Beitrag* und *Kommentar* unterteilt, um die benötigte rekursive Referenzierung ermöglichen zu können. Ein Kommentar unterscheidet sich folglich von einem Beitrag in der Hinsicht, dass nicht alle Felder ausgefüllt werden.

4.2. WBA-Modellierung

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS4/WBAModellierung.pdf

Mit Hilfe des Architekturdiagramms konnte die WBA-Modellierung genauer erfolgen. Wie schon in der Architektur deutlich wird handelt es sich um ein Client-Server Anwendungssystem.

Die Programmierung des Servers soll in Node.js erfolgen. Zum einen wurde dies schon in WBA2 benutzt und stellt daher eine gewohnte Arbeitsumgebung dar, zum anderen besteht auch auf privater Basis viel Kenntnis in JavaScript und Node.js, was sowohl eine stabile als auch schnelle Programmierung ermöglicht. Weiterhin gibt es auf privater Basis viele und gute Erfahrungen mit MongoDB als Datenbanksystem. Dies bietet im Vergleich zu dem in WBA2 benutzten Redis bessere Werkzeuge zum Abfragen und Verarbeiten von Datensätzen. Mongoose ist ein sehr beliebtes Rahmenwerk (object modeling tool) für MongoDB. Wie im Artefakt erwähnt wird vorallem die populate-Funktionalität von

mongoose benötigt um – ähnlich wie in SQL mit JOINs – ein Dokument in ein anderes Dokument anhand einer Referenz-ID zu überführen.

Natürlich wäre dies alles auch händisch mit Redis möglich, stellt aber einen wesentlichen, vermeidbaren Mehraufwand da. Apache CouchDB wäre eine weitere mögliche Datenbankalternative, die allerdings noch nicht benutzt wurde. Eine Einarbeitung in CouchDB zur Abwägung zwischen jenem und MongoDB wäre in einem großen Projekt angebracht, jedoch in der gegebenen Projektzeit nicht möglich.

HATEOAS für die Umsetzung nach Richardsons Maturity Model Level 3 wird mittels Reference-Links, wie auch bei XML, erreicht. JSON besitzt keine genormten Schemas wie XML. Es gibt einige Ansätze zu JSON-Schemas, die sich noch nicht voll etabliert haben. Für unsere Umsetzung mittels link-Key wurde JSON-HAL[8] als Vorbild genommen. Einige andere Ansätze für ref-links in JSON wurden nicht weiter verfolgt. Meist besitzen sie Arrays mit Link-Attributen, die jedoch eigener Meinung nach zu Umständlich zum Parsen sind. Eine solche Alternative könnte so aussehen:

```
{ …,
  “links” : [
    { ref: “self”,
      href: “http://service.tld/ressource/12345” },
    { ref: “list”,
      href: “http://service.tld/ressource” },
    { ref: “next”,
      href: “http://service.tld/ressource?page=1” }
  ]
}
```

Eine solche Alternative verfolgt zwar den Ansatz eines eindeutigeren Schemas, ist allerdings bei der Implementation durch die Verwendung des Arrays nur durch Schleifen komplett auslesbar.
Die von uns gewählte Struktur bietet die Möglichkeit allein durch den Selektor *link.self* schon die URL zur Ressource zu bekommen.

Als weiteres, externes Mittel wurde Google Cloud Messaging (GCM) zur asynchronen Kommunikation als Versand von Notifications gewählt. Durch die kostenlose Natur von GCM und einfache API bietet sich GCM bestens an, vor allem da es mit der clientseitigen Anwendung in Android die gängigste Plattform ist. Parse.com bietet zwar auch eine Schnittstelle zum Versand von Notifications an, jedoch ist Parse.com mit einer vollständigen Datenbankstruktur ausgestattet, die gar nicht benötigt wird.

Das *Light Ping* verfahren ist im Gegensatz zum *Fat Ping* datensparsamer.

[8] http://blog.stateless.co/post/13296666138/json-linking-with-hal

Durch die Vorgabe der clientseitigen Programmierung in Java wurde sich für eine Android-App entschieden, was auch in der Domäne eine passende Anwendungsplattform ist. Nachteil ist, dass damit nur Nutzer mit Android-Smartphones erreicht werden und andere Systeme wie iOS oder Windows Mobile vernachlässigt werden.

Nach der Bestimmung der Ressourcen als gängige REST-Tabelle[9] konnten auch die Funktionsweisen der Anwendungskomponenten besser skizziert werden.

Als Ressourcen wurden ausgearbeitet und mit den jeweiligen HTTP-Methoden versehen:

- **/courses** zur Auflistung und Ausgabe von Studiengängen
- **/entries** für Beiträge und Kommentare
- **/users** für Benutzerverwaltung
- **/verification** zur Verifikation des Studenten-Status
- **/tlds** zur Abfrage der Hochschul-TLDs

Übertragene Daten werden immer als application/json formatiert. Denkbar wäre hierfür eine Content-Negotiation um Daten in für Systeme geeigneteren Formaten zu übertragen. Aus Zeitgründen wurde die Content-Negotiation außen vor gelassen, da sie für die Fertigstellung des Projektes nicht benötigt wird. Node.js kann selbstverständlich mit dem JSON-Datenformat umgehen. Android bietet über org.json auch sehr gute Tools zur Verarbeitung von JSON-Datenformaten an.

4.3. Prototypen UI

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS4/PrototypeUI.pdf

Entgegen der Planung wurde die Prototypen UI nicht in UXPin, sondern in der Entwicklungsumgebung selbst angefertigt. Diese Vorgehensweise bringt zwar einige Nachteile mit sich, aber auch einige erhebliche Vorteile. Die Entwickelten Prototypen sind jedoch non-funktional, da hier noch mit Änderungen in späteren Iterationen zu rechnen war.

Zum Einen sind in der Entwicklungsumgebung bereits die Design-Libraries für das finale Produkt vorhanden, somit würde uns viel Arbeit auf einer übergreifenden Ebene abgenommen. Elemente wie eine für die mobile Benutzung angemessene Höhe von Buttons oder angemessene Größen von Abständen konnten mit wenig Problemen übernommen werden.
Zum Anderen erlaubte uns dieses Vorgehen ein Testen direkt am Gerät in einer nicht emulierten Umgebung. Es ist sogar möglich, die Prototype UI am Geräte des Benutzers zu installieren um so möglichst unverfälschte Ergebnisse in der Evaluierung zu erhalten.

[9] Webber, “REST in practice”, Chapter 4, Building CRUD Services, Tabelle 4-1

Außerdem rechneten wir mit einer Zeitersparnis im späteren Projektverlauf, da die UI bereits im richtigen Medium verfügbar wäre und nur noch mit Funktionalität gekoppelt werden müsste.

Die Prototyen UI umfasste außerdem nicht den kompletten Umfang der Features der finalen Anwendung, sondern nur den Teil, den wir für die finale Umsetzung als realistisch einschätzten. Diese Entscheidung sollte uns Zeit sparen, sowohl in der Entwicklung als auch in der Evaluation.
Die Zeitersparnis in der Evaluation ist allerdings ein zweischneidiges Schwert. Zwar würden die nicht benötigten Elemente, da sie nicht vorhanden sind, die Probanden nicht verwirren und somit unverfälschtere Ergebnisse liefern, allerdings müsste für die fehlenden Features zu einem späteren Zeitpunkt eine erneute Evaluation stattfinden. Zudem müssen diese Funktionalitäten dann in ein bereits bestehendes System eingebunden werden, was die Möglichkeiten und Freiheiten in der Implementierung etwas einschränkt.

5. MS5

5.1 Funktionale Prototypen

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/tree/master/MS5/funktionalePrototypen

Bei den funktionalen Prototypen handelt es sich nicht um eigens angefertigte Artefakte, sondern um Ist-Stände der Implementation zu einem bestimmten Zeitpunkt. Die Entwicklung wurde so geplant, dass in den Prototypen die wichtigsten Funktionen enthalten und benutzbar sind.
Diese Vorgehen sorgte für eine effizientere Entwicklung und ermöglichte es uns außerdem, direkt auf Basis der Prototypen weiter zu arbeiten.

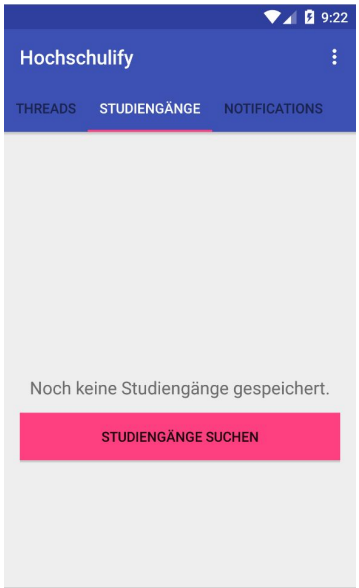
5.2. Evaluationsergebnisse UI

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS5/EvaluationsergebnisseUI.pdf

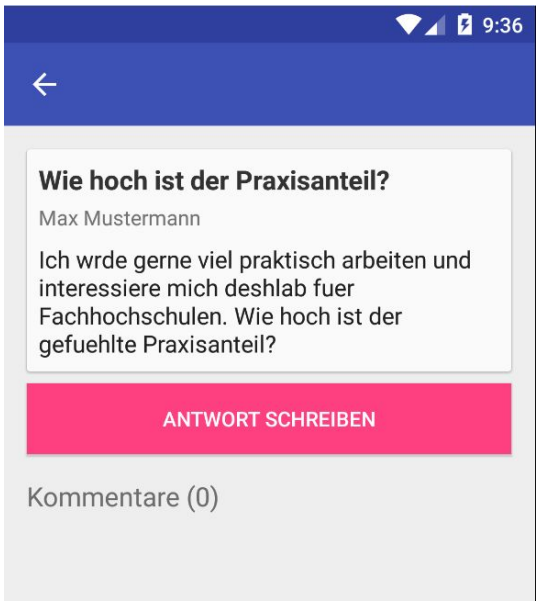
Kritische Anmerkungen zum Vorgehen
Zur Evaluation müssen eine Punkte hervorgehoben werden, die Einfluss auf die Evaluation in ihrer Qualität und Vollständigkeit nahem.
Zum einen gehörten zur Gruppe der Tester ausschließlich Studenten. Damit konnte die Evaluation nicht repräsentativ für die gesamte Zielgruppe sein. Allerdings wurde darauf geachtet, auch Studenten aus dem ersten Semester zur Evaluation heran zu ziehen, da diese noch vor kurzem zur Zielgruppe der Studieninteressierten gehörten.
Zum anderen waren die Tester Informatikstudenten. Da sich diese unter Umständen besser mit Software auskennen als Studenten anderer Themengebiete könnte dies Einfluss auf die Testergebnisse genommen haben.
Außerdem zu erwähnen ist, dass alle Tester auch privat Nutzer des Betriebssystems Android sind. Da das Interface jedoch gezielt für Android entworfen wurde ist diese Tatsache zu vernachlässigen. Zwar würden sich Nutzer der Plattform iOS unter Umständen schlechter zurecht finden, allerdings würden diese auch nie mit dem Android-Interface in Kontakt kommen, sondern mit einem angepassten iOS-Interface.

Änderungen aufgrund der Evaluationsergebnisse
Basierend auf den Ergebnissen der Evaluation konnte eine neue Iteration des User Interfaces erstellt werden. Zu diesem Punkt ließ sich jedoch bereits absehen ließ, dass nicht alle in der Prototype UI gezeigten am Ende auch funktional umgesetzt werden könnten. Wir beschränkten uns daher bei der Iteration auf die Elemente, die in der Implementation auch umgesetzt wurden.

Im Reiter “Studiengänge” werden dem Nutzer nun sowohl eine Nachricht, die ihn über den Sinn dieser Ansicht informiert (“noch keine Studiengänge gespeichert”) als auch ein Button, der ihn sofort zur Suche bringt angezeigt. Der Nutzer lernt damit von Anfang an, welchen Zweck die Ansicht hat und über welchen Weg er sie mit Inhalten befüllen kann.

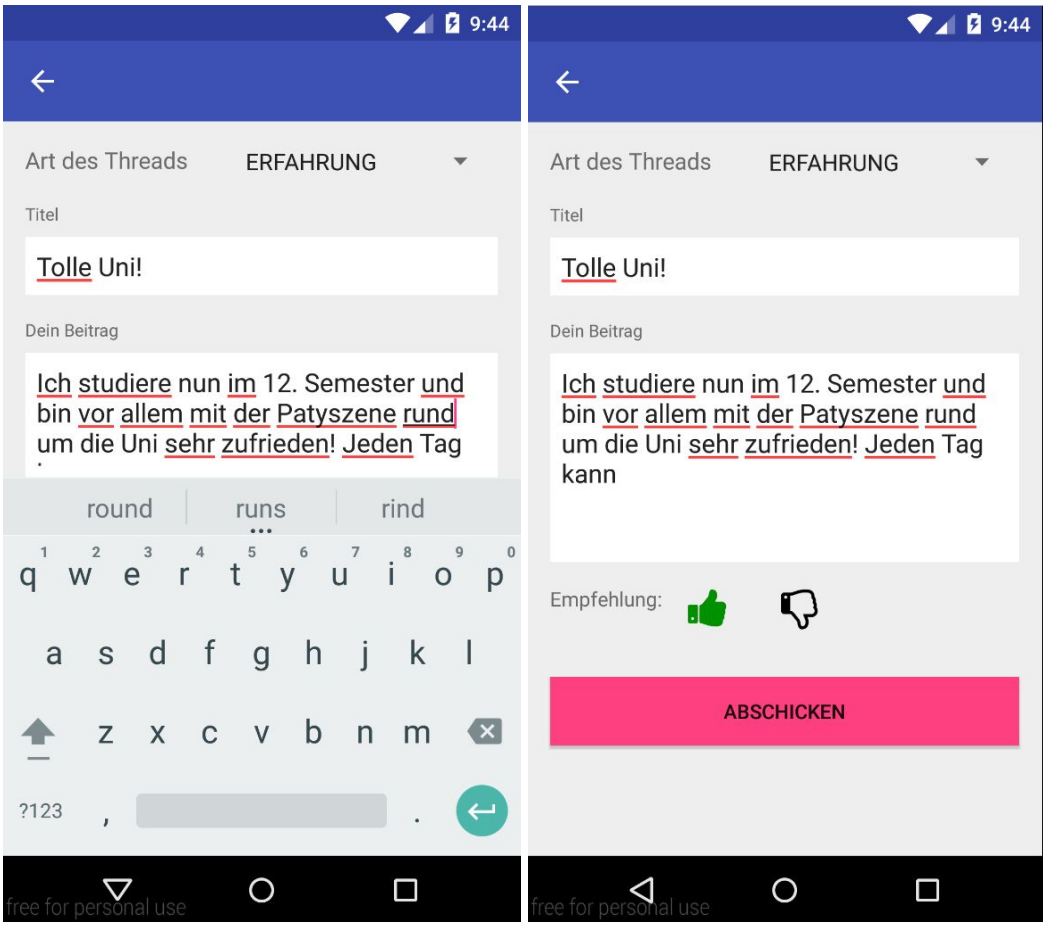


Die Funktion zum Antworten auf Beiträge wurde in Form eines Buttons im Stil der Applikation unter dem jeweiligen Beitrag prominenter Platziert.
Da dieser Button der einzige in der Ansicht ist wird dieser deutlich schneller erkannt und seine Funktion identifiziert.



An der Ansicht zum verfassen eines Beitrages wurde trotz des versteckten Buttons zum abschicken nur wenig verändert. Das Icon zum Absenden in der Toolbar wurde entfernt,

das dieses nur selten genutzt wurde und seine Funktionalität durch das Icon nicht explizit klar wurde.
Der Button ist weiterhin von dem Software-Keyboard verdeckt, dieser Umstand ist bewusst so gewählt. Es gab hier einige Lösungsansätze, so war eine Überlegung den Button immer oberhalb des Keyboard zu platzieren. Dadurch würde allerdings der Bereich, in dem der vom Benutzer geschriebene Text angezeigt wird noch weiter schrumpfen. Da die intendierte Textlänge auf der Plattform 300 Zeichen übersteigt, wäre das verfassen von Texten dadurch sehr unergonomisch.
Eine weiterer Ansatz ist, die Enter-Taste auf dem Keyboard durch eine Sende-Taste zu ersetzen. Auch wenn dies technisch ohne weiteres möglich wäre würde dieser Ansatz dem Benutzer eine entscheidende Möglichkeit zur Textformatierung nehmen.
Die aktuelle Lösung ist also nicht perfekt, sie ist jedoch die beste Lösung, die in der gegebenen Zeit und mit dem gegebenen Ressourcen gefunden werden konnte.



Weiterhin wurden einige Indikatoren hinzugefügt. Es ist nun ersichtlich, welche Nutzer verifizierte Studenten sind, ob bewertende Beiträge positiv oder negativ gegenüber dem Studiengang sind und welche Anteil an Studenten mit einem Studiengang zufrieden ist.

5.3. Kommunikationsziele und Konzept für Poster

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS5/Kommunikationsziele.pdf

Die Projektpräsentation soll die Projektproblematik und -lösung zeigen, außerdem sollen auch technische Details behandelt werden. Somit sollen Informatiker und Geldgeber angesprochen werden, bzw. ein fiktives Publikum von *fachkenntlich potentiellen Geldgebern*.

Das Poster wurde in verschiedene thematische Bereiche aufgeteilt. Die Bereiche beziehen sich aufeinander und führen von der Projektidee bis hin zur technischen Umsetzung.

Beginnend mit der Problematik der Studiengangsfindung (oben links) zeigen die Umfrageergebnissen (oben rechts), dass es sich beim benannten Problem um ein echtes Problem handelt und dass ein Markt für das Projekt vorhanden ist.
Darunter wird das Projekt anhand von den aussagekräftigsten Screenshots der App vorgestellt. Dazu gehören die Erfahrungsberichte mit Anzeige von verifizierten Studenten und die Suche von Studiengängen mit Übersicht über den Grad der Empfehlung. Jeweils darunter werden die beiden Komponenten der Anwendung mit Ablaufdiagrammen technisch erklärt.
Dies zeigt ebenfalls, dass es bereits einen funktionierenden Prototypen gibt, was bei Geldgebern einen positiven Eindruck hinterlässt.

Mit einem Fokus auf Technik werden darunter die Ressourcen aufgelistet. Diese sind ebenfalls räumlich den beiden Komponenten zugeordnet, werden aber auch durch den farblichen Verlauf zusammengeführt, da die verschiedenen Ressourcen nicht komplett unabhängig voneinander sind.

Zuletzt wird genannt, dass der Service eine REST-Schnittstelle programmiert im Maturity Model Level 3 von Richardson implementiert hat, und welche Technologien für dafür verwendet wurden. Die Nutzung von Node.js, mongoDB, express.js und Google Cloud Messaging veranschaulicht die modernen Technologien der Implementierung.

6. MS6

6.1. Prozessassessment

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS6/Prozessassessment.pdf

Zum Prozessassessment wurden in einer ruhigen Runde die vorigen Wochen und Monate zeitlich besprochen. Sowohl Positives als auch Negatives wurde angesprochen und im Artefakt notiert.

Die Führung des Projektplans war für beide Teammitglieder keine erfreuliche Arbeit. Es schien so, als würde zu viel Zeit verwendet werden um das Projekt zu planen, als es tatsächlich durchzuführen. Für den Projektplan wurden die Artefakte (mit potentiellen Unteraufgaben) nur in chronologischer Reihenfolge geplant. Die Überarbeitung eines Artefakts wurde zu der vorigen Dauer addiert. Eine genaue chronologische Übersicht ist deshalb nicht erkennbar. Im Nachhinein wurde bemerkt, dass man diese Überarbeitungen nochmals separat hätte auflisten müssen.

Mittels Trello wurde der Status aller Artefakte beobachtet, diskutiert und mit Notizen und ToDo's versehen. Zeiten für Artefakte, die sich über mehrere Tage gestreckt haben, wurden dort notiert und summiert in den Projektplan eingetragen. Das hat zwar dazu geführt, dass Zeiten genauer waren, allerdings sind dadurch zwei entkoppelte Sammlungen von Zeiten entstanden. Vor allem bei der Projektbegründung geht dadurch nicht hervor, wie viel Zeit die Begründung zu einem Artefakt in Anspruch genommen hat.

Positiv war, dass zu Beginn bei den Risiken auch zeitliche Risiken ausgearbeitet wurden. Somit konnte im Projektplan die geringe verfügbare Zeit über die Weihnachtsfeiertage und -ferien berücksichtigt werden.

Durch die Programmierkenntnisse beider Teammitglieder konnte die Dauer der Implementation gut abgeschätzt werden. Es wurde sogar zu viel Zeit eingeschätzt, wodurch zeitliche Rückstände noch besser aufgeholt werden konnten.

Generell hätten Zeiten lieber leicht überschätzt werden sollen, als zu versuchen Zeiten so genau es geht zu schätzen.

6.2. Fazit

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/blob/master/MS6/Fazit.pdf

Beim Fazit ist aufgefallen, dass einige Funktionen, die man umsetzenwollte, nicht in den Zielen verschriftlicht wurden. Dabei handelt es sich um Features, die nicht geplant waren und erst im Laufe des Projekts, insbesondere während der Implementation, dazu kamen. Da in den Zielerreichungsgraden jedoch die wichtigsten Ziele vorhanden waren, wurden die Wichtigsten umgesetzt. Nicht erreichte Ziele wurden trotzdem mindestens in Grundzügen umgesetzt. Sie wurden also nur nicht komplett erreicht. Gänzlich nicht erreichte Ziele waren zur Projektproblematik sekundär. Möglicherweise hätten diese Zeile nicht in den Zielerreichungsgraden formuliert werden sollen.
Die Darstellung der geschachtelten Kommentare wurde nicht erreicht, da dies komplexer als ursprünglich gedacht war. Hieraus hätte man im Vorfeld ein Proof of Concept machen müssen.

6.3. Implementation

Artefakt:
https://github.com/Plsr/EISWS1516MaemeckePoplawski/tree/master/MS6

Für die Implementation konnten viele schon vorhandenen Ergebnisse genutzt werden. Da die Prototypen UI direkt in Android Studio umgesetzt wurde, wurde der Client auf dieser direkt aufgebaut. Auch die Proof of Concepts zur Datenstruktur, zum Öffnen der Verifikationslinks und zur Auflistung der Hochschul-TLDs konnten teilweise vollständig übernommen werden und in die Implementation eingefügt werden.

Als funktionaler Prototyp wurden zuvor die wichtigsten Funktionalitäten umgesetzt. Darauf aufbauend konnte die Implementation vervollständigt werden.

Durch eigene Erfahrung bei der Programmierung mit express.js wurde die Struktur des Services relativ schnell solide aufgesetzt. Für den Service wurden neue Elemente aus der JavaScript Syntax von ES6 verwendet, welche über Babel transpiliert wird, um kompatibel mit der aktuellen Node.js Version zu bleiben. ES6 besitzt eine klare Syntax zum importieren und exportieren von Funktionen. Außerdem konnte durch die Verwendung von ES6 diese moderne Syntax erlernt werden, damit sie auch in zukünftigen Projekten verwendet werden kann.
Der in Node.js Version 5 verfügbare `Promise` wurde oft genutzt, um den schrittweisen Anwendungsverlauf innerhalb der einzelnen Routen-Funktionen (bei uns *Controller* genannt) besser kontrollieren zu können. Die typische Callback-Struktur in JavaScript, die wie ein seitliches Dreieck aufgebaut ist[10], konnte mittels Promise minimiert werden, was

[10] http://callbackhell.com/

den Ablauf des Programms besser lesbar macht. Auch werden Promises von mongoose unterstützt[11].
Innerhalb des Services wurde das Versenden von GCM-Notifications asynchron programmiert. Beim Request wird folglich nicht gewartet, dass die Benachrichtigung über GCM verschickt wurde, sondern der Response erfolgt direkt sobald es möglich ist, und die GCM-Nachricht wird zeitgleich und abgekoppelt vom Response verschickt.

Um bei authentifizierte Zugriffen keine Klartext-Passwörter senden zu müssen, wurde eine `/auth` Route hinzugefügt, die außerhalb des REST-Scopes ist. Beim Übertragen von E-Mail und Passwort an diese Route, wird ein Token generiert, mit dem sich der User über die HTTP Header `X-Auth-User` (als UserId) und `X-Auth-Token` bei einem Request authentifizieren kann. Dies entspricht einer sehr simplen und naiven Umsetzung von standardmäßigen OAuth(2.0) Funktionalitäten.
Dieser Token führt momentan noch dazu, dass man sich nur auf einem Gerät gleichzeitig einloggen kann.

Auf dem Client wird nach dem Login der Token zusammen mit der UserId in den Shared Preferences gespeichert, um authentifizeirte Requests ohne weitere Eingabe eines Passworts durchführen zu können.
Mit dieser Vorgehensweise sollte man die Anwendung natürlich nicht veröffentlichen.

Auf dem Client müssten noch Lade-Indikatoren für asynchrone Funktionen wie HTTP Requests angezeigt werden, um dem Benutzer auch in MCI-Hinsicht besseres Feedback geben zu können. Dies wurde nicht eingebunden, da es keine Funktionalität des Clients zeigt.

Bei der Implementation des Clients wurde versucht auf Ergebnisse der Benutzungsmodellierung einzugehen. Beispielsweise war das bookmarken von Studiengängen ursprünglich nicht geplant und wurde aus MCI-Sicht trotzdem implementiert.
Nicht funktionsfähig ist hingegen die Paginierung der Beiträge in einem Studiengang am Client. Momentan werden die ersten 10 Einträge aufgelisetet. Der Service

Die Umkreissuche von Studiengängen konnte auf dem Client erfolgreich implementiert werden und wird ausgeblendet, sobald über das Gerät keine Geoposition verfügbar ist. Damit werden dem Benutzer keine Funktionalitäten suggeriert, die für ihn nicht nutzbar sind. Bei der Umkreissuche werden Ergebnisse nach Entfernung sortiert aufgelistet. Es ist sogar möglich nach bestimmten Studiengängen im Umkreis zu suchen. Denkbar wäre es zusätzlich die Entfernung zur Hochschule anzeigen zu lassen.
Die Umkreissuche ist ein eigenes Feature des Clients, das der Service so nicht anbietet. Der Service gibt nur die Geokoordinaten der Hochschule zu einem Studiengang aus. Es wäre sogar möglich, statt den Geokoordinaten die Adresse der Hochschule zu speichern. Der Client könnte mittels Google APIs die Geokoordinaten dieser Adresse selbst herausfinden. Android bietet dafür eine native Geocoder Klasse an[12].

[11] http://mongoosejs.com/docs/promises.html
[12] http://developer.android.com/reference/android/location/Geocoder.html

Weitere Ziele wie die Anzeige der Empfehlung zu einem Beitrag und der hervorgehobenen Anzeige von verifizierten Mitgliedern konnten erreicht werden. Wie im Fazit schon angesprochen, wurde die geschachtelte Anzeige der Kommentare nicht erreicht. Dafür müssten verschiedene Implementierungsmöglichkeiten ausprobiert und abgewägt werden.

Kritische Reflektion über die Vorgehensweise

Der nachfolgende Abschnitt soll eine kurze, kritische Reflektion zur allgemeinen Vorgehensweise darstellen. Hier sollen alle die Dinge Erwähnt werden, die nicht zum Fazit (weil sie nicht direkt etwas mit dem Zielerreichungsgrad zu tun haben) oder in das Prozessassessment passen.

MCI-Spezifischer Teil
Die Entwicklung der Prototype UI in Editor-Umgebung ist im Nachhinein eher kritisch zu betrachten. Zwar trafen alle erwarteten Vorteile auch ein, jedoch stellte diese Herangehensweise keine große Zeitersparnis dar.
Wie erwartet konnte auf der Prototype UI aufbauen auch ein funktionaler Prototyp entwickelt werden, jedoch nahm die Entwicklung und Änderung mehr Zeit in Anspruch, als dies bei einem Grafikprogramm oder Prototyping-Tool der Fall gewesen wäre.
Abschließend bleibt hier die Frage zu beantworten, ob dem Projekt mit diesem Ansatz genau so gut geholfen ist wie mit einer möglichen weiteren Evaluation und Iteration, die durch eine kürzere Entwicklungszeit für die Prototypen unter Umständen möglich gewesen wäre.

Weiterhin war es zeitweise kompliziert, das Benutzerzentrierte Vorgehensmodell zu befolgen. Dies liegt darin begründet, dass die Zeitlich mögliche Zahl der Iterationen recht begrenzt war. Um für das User Interface trotzdem zufriedenstellender Ergebnisse zu erzielen musste für die Prototypentwicklung und Evaluation teilweise auf Methoden des Discount Usability Engineering zurück gegriffen werden

Implementation
In der Retrospektive wäre es ratsam gewesen, zu Beginn der Implementation Coding Guidelines zu erstellen. Zwar kam es zu keinem Zeitpunkt des Projektes zu Schwierigkeiten bei der Zusammenführung von Code, jedoch hätten Guidelines für ein einheitlicheres Bild uns leichter verständlichem Quellcode geführt.

Außerdem wäre es strukturell angenehmer gewesen, zusammen an Features zu arbeiten und diese fertig zu stellen, anstatt wie öfter Features “nebeneinander her” zu Programmieren. Hier wäre unter Umständen eine erhöhte Produktivität möglich gewesen.