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

Projektexposés

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.

Zielhierarchie

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.

Marktrecherche

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.

Domänenrecherche

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.

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.

---

[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

Alleinstellungsmerkmale

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.

Architekturdiagramm und -begründung

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, zusätzlich ein normales Anwendungsdiagramm zu erstellen. In diesem Diagramm befinden sich außerdem die Datenübertragungen und -formate.

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 auch mit AtomPub, allerdings sind Datenstrukturen in XML sehr groß und für mobile Zwecke eher ungeeignet. JSON ist im Vergleich zu XML schneller und leichter serialisierbar und deserialisierbar.

Die Idee zur einheitlichen Datenstruktur in genesteten Objekten kam auf, um bei der späteren Entwicklung besonders einfache und einheitliche Funktionalitäten zur Darstellung der Datensätze zu haben. Dabei wurde sich an der Struktur der Reddit API orientiert.

WBA-Modellierung

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. Eine Peer-to-Peer Anwendungsverteilung ist durch die Art der Anwendungsdaten gänzlich ungeeignet.

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.

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.

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 konnten auch die Funktionsweisen der Anwendungskomponenten besser skizziert werden.

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

- **/studiengang** zur Auflistung und Ausgabe von Studiengängen
- **/thread** für Beiträge und Kommentare
- **/user** für Benutzerverwaltung
- **/verify** zur Verifikation des Studenten-Status
- **/tld** 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.

Datenstrukturen

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 /studiengang) 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.