Bachelorarbeit am Institut für Informatik der Freien Universität Berlin,

Arbeitsgruppe Software Engineering

Entwicklung einer Android App zum Erfassen und Teilen von Erfolgen im Hinblick auf eine Erweiterung zu einer App Familie

Marco Gester
Matrikelnummer: 4470405
m.gester@gmx.net

Erstgutachter: Prof. Dr. Lutz Prechelt
Zweitgutachter: Prof. Dr. Jochen Schiller

Berlin, den 04.08.2015

Zusammenfassung

Diese Bachelorarbeit beschreibt den Entwicklungsprozess einer Android App zur Erfassung und zum Teilen von Achievements (Erfolgen).

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Berlin, den 04.08.2015

Marco Gester

Inhaltsverzeichnis

1 Einleitung

Die hier vorliegende Bachelorarbeit beschreibt den Entwicklungsprozess einer App.

Die App soll es ermöglichen den Besuch von bestimmten Orten als Achievement (Erfolg) zu speichern und diesen auch mit anderen Personen via Facebook, Twittter und Instagram (als die wichtigsten Vertreter von sozialen Netzwerken)zu teilen .

Die Gespräche die zu der Entwicklung der App Idee führten, drehten sich um den noch relativ neuen Trend zur Gamifizierung des Lebens. Hierbei geht es darum, Prinzipien aus dem Spieldesign und Spielmechaniken auf das reale Leben zu übertragen.

Das in dieser App aufgegriffene Konzept von Achievements wird in Spielen dazu genutzt Motivation beim Spieler zu erzeugen. Dadurch können auch eigentlich langweilige Tätigkeiten Spielspass erzeugen. Eine weitere Dimension der Achievements in Spielen ist das Vorzeigen seiner gesammelten Erfolge vor anderen Spielern. Dies ist eine Mittel in einem Spiel Ansehen zu gewinnen und den Ehrgeiz der Spielteilnehmer zu fördern. Dieser Aspekt soll mit dieser App und einer Erweiterung auf andere Domänen aufgegriffen werden.

Im folgenden zweiten Kapitel gehe ich kurz auf meinen Erfahrungshintergrund im Programmieren ein, um eine Einordnung der erzielten Ergebnisse am Ende dieser Bachelorarbeit zu ermöglichen.

Das Kapitel 3 setzt sich mit der Erkundung der Domäne auseinander. Es wurden hierzu Gespräche mit Testpersonen geführt, um die Erwartungen von potentiellen Benutzern dieser App zu erfassen und um Ideen für eine Erweiterung der Domäne zu sammeln.

Kapitel 4 beschreibt den Prozess des Programmierens der App, die diversen Probleme, die gelöst werden mussten und diskutiert einige Designentscheidungen, die während der Entwicklung getroffen wurden.

Im Kapitel 5 wird eine Evaluation der erzielten Ergebnisse versucht.

Das Fazit in Kapitel 6 beschreibt meine persönlichen Eindrücke und Lehren, die ich aus den Arbeiten an dieser Bachelorarbeit ziehen konnte.

Vom Leser dieser Bachelorarbeit werden grundlegende Kenntnisse im objektorientierten Programmieren erwartet. Viele Begriffe aus diesem Feld werden im Rahmen dieser Arbeit nicht weiter erläutert.

Der Code zu dieser Arbeit ist auf der beigefügten CD oder im Internet (Serverkomponente: `https://github.com/Marcoge/gesterachievementserver`, App: `https://github.com/Marcoge/gesterachievementapp`) zu finden. Eine Anleitung zur Inbetriebnahme der Komponenten liegt jeweils mit der README.md bei.

2 Erfahrungshintergrund

Bevor ich mit den Arbeiten an dieser Bachelorarbeit begann, hatte ich relativ wenig Erfahrung im Programmieren und keine Erfahrung in der Umsetzung von kompletten Projekten.

Mein Kenntnisstand bewegte sich auf einem Level, der es mir ermöglichte die meisten kleinen Programmieraufgaben, die auf den verschiedenen Übungszetteln während des Studiums zu finden waren, zu lösen, ging aber wenig darüber hinaus.

Der größte Teil der in der Universität zu bearbeitenden Aufgaben wurde in Java programmiert. Hinzu kamen kurze Exkursionen in Haskell, GO, Assembler und C. Meine Kenntnisse in diesen Sprachen sind allerdings im besten Falle als rudimentär zu bezeichnen. Selbst die relativ einfachen und wenig komplexen Aufgaben auf den Übungszetteln kosteten mich viel Zeit.

Ich realisierte in meiner Studienzeit zwei eigene kleine Hobbyprojekte in C# . Das erste Projekt sollte ein Programm werden, dass die optimalen Züge in einem Scrabblespiel berechnet. Das Programm wurde nie fertig, da immer wieder wichtigere Arbeiten im Studium erledigt werden mussten.

Das zweite Projekt war ein kleines Hilfsprogramm, das es mir erlaubte das Volumen des Datendurchsatzes beliebiger Netzwerkschnittstellen graphisch darzustellen.

Weiterhin war ich an der Entwicklung eines Email Clients mit nativ integrierter Ende zu Ende Verschlüsselung beteiligt. Mein Beitrag zu diesem Projekt beschränkte sich auf Bugfixes (vor allem in den integrierten Parsern) und kleinere Implementierungen im Rahmen der Benuntzeroberfläche (z.B. Popups, die über neue Emails informieren).

Während meines Berufspraktikums konnte ich ein wenig Erfahrung im Bereich Webdevelopment sammeln. Ohne Vorwissen arbeitete ich mich in den sechs Wochen hauptsächlich in das Google Web Toolkit[1] ein. Um wirklich an Projekten der Firma mitzuarbeiten war die Einarbeitungszeit leider zu kurz, da ich neben der Beschäftigung mit GWT noch diverse andere Aufgaben hatte. GWT zusammen mit WPF (Windows Presentation Foundation, das UI Framework von C#) vermittelte mir vorab schon ein gewisses Verständnis wie Benutzeroberflächen programmiert werden. Das Konzept das Android nutzt ist dem von C# 's WPF sehr ähnlich.

3 Erkunden der Domäne

Zur Erkundung der Domäne sind zwei klar von einander getrennte Teilbereiche zu betrachten:

Zum einen wurden Gespräche mit potentiellen Nutzern der App geführt, in denen die Erwartungen der Personen an eine App zum Thema Reiseachievements ermittelt und Anregungen für eine Erweiterung der App Familie auf weitere Domänen gesammelt werden sollten.

Zum anderen wurden generelle Überlegungen zum Aufbau des Projekts angestellt und eine Vorauswahl der zu nutzenden Technologien getroffen. Auf dieser Grundlage erfolgten weitere Vorbereitungen und Nachforschungen.

3.1 Der menschliche Faktor

Die Auswahl der Gesprächspartner wurde bewusst heterogen gestaltet. Es wäre naheliegend gewesen nur junge, technik-affine Menschen zu befragen. Diese Gruppe erscheint für das Erreichen der Zielsetzung dieser Gespräche gut geeignet. Allerdings wird damit ein großer Teil des Spektrums der potentiellen User der App nicht abgedeckt. Diese App zielt auch auf Menschen, die vielleicht keinen besonderen technischen Hintergrund haben und ihr Smartphone nicht intensiv nutzen. Reisen ist nicht auf junge, technisch versierte Menschen beschränkt. Insbesondere soll auch die Generation 50+ angesprochen werden.

Es wurden drei Gesprächspartner ausgewählt, die der ersten Gruppe zuzuordnen sind. 4 sind der zweiten Gruppe zuzuordnen. Durch eine starke Einbindung der weniger technisch versierten Gruppe sollte ein besseres Verständnis erlangt werden, was Menschen von solch einer App erwarten. Erkundet wurde wie groß das Verständnis für derartige Konzepte bei Personen ist, die bis jetzt wenig mit den in der App vertretenen Konzepten (Achievements, etwas „einloggen“) in Berührung gekommen sind.

Die Gespräche mit beiden Gruppen liefen nach dem selben, vorab entwickelten, Schema ab:

Eröffnet wurde das Gespräch mit folgender kurzer Einleitung:

„Stell Dir vor, Du bist auf Reisen und willst gerne dokumentieren wo du schon überall warst und was Du schon gesehen hast. Die App, die dafür benutzt wird, soll die Möglichkeit bieten, diese Erfolge (Achievements) mit Freunden zu teilen und einen einfachen Überblick bieten, wer wo schon ge-

wesen ist. “

Die Einführung ist bewusst kurz und allgemein gehalten, da das Gespräch der Ideenfindung und Erkundung der Erwartungen der Personen dienen sollte. Eine zu ausführliche Einleitung mit zu vielen Nennungen von geplanten Funktionen hätte die Vorstellungen der Gesprächspartner zu sehr beeinflusst.

Für den weiteren Gesprächsverlauf gab es drei Fragen, die in jeder der Unterhaltungen als Leitfaden genutzt wurden:

- „Was für Features erwartest du von solch einer App?“
- „Welche Erfahrungen hast Du auf Reisen gemacht, die einen interessantes Achievement ergeben würden?“
- „Was für Domänen für Apps könntest Du Dir noch vorstellen, in denen das Konzept von Achievements umzusetzen wäre?“

Der individuelle Gesprächsverlauf differierte sehr stark. Jede Person benötigte unterschiedlich umfangreiche Erklärungen zu den Konzepten und dem generellen Funktionsumfang, der in dieser App umgesetzt werden sollte. Die meisten Erläuterungen waren zum Begriff des Achievements (Erfolges) nötig. Selbst Personen mit sehr guten technischen Kenntnissen waren nicht immer mit diesem Konzept aus dem Spieldesign vertraut.

Zuerst sollen die Gespräche mit Personen aus der ersten, jungen und technisch versierten Gruppe betrachtet werden:

Hier waren weniger Erklärungen nötig als in der Gruppe der älteren Versuchspersonen. Oft wurde die Fragestellung auf Anhieb verstanden und die Antworten waren überwiegend konstruktiv und sehr hilfreich.

Die besten Ideen für den Funktionsumfang und Aufbau der App oder verwandter Apps kamen aus dieser Gruppe.

Neben den obligatorischen Sehenswürdigkeiten wurde vorgeschlagen auch Insider Spots (z.B. gute Restaurants, versteckt gelegene Aussichtsmöglichkeiten etc.) mit in die Liste der Achievements aufzunehmen. Die Vorschläge für diese Insider Spots sollten bevorzugt von den Nutzern der App kommen. Die Aufnahme der Achievements in die Datenbank könnte über eine Feedback Funktion in der App realisiert werden.

Von vielen Personen wurde die Funktion gefordert Achievements „privat“schalten zu können. D.h. es sollte die Möglichkeit geschaffen werden, dass in den

Funktionen der App die Achievements von anderen Usern anzeigen, die privat markierten Achivements nicht angezeigt werden.

Ein weiterer Vorschlag war es Bestenlisten mit in die App aufzunehmen. Diese Bestenlisten könnten zum Beispiel die Gesamtzahl der Achivements wiederspiegeln oder die Anzahl der in einer Stadt oder einem Zeitraum erreichten Achievements.

Weiterhin wurde vorgeschlagen Gruppenachievements mit in die App zu integrieren. Dies würde bedeuten, dass einige Achievements nur zu loggen wären, wenn sich eine gewisse Anzahl von Freunden (mit aktivierter App) zum Zeitpunkt des Logversuches in einem bestimmten Umkreis aufhalten.

Es gab auch Ideen die keine Aufnahme in die App fanden.

Zum Beispiel wurde vorgeschlagen, die an einem bestimmten Ort verbrachte Zeit mit als Achievement aufzunehmen. Für die in dieser Bachelorarbeit programmierte App wurde dieser Vorschlag nicht berücksichtigt, da die in diesem Vorschlag enthaltene Zeitkomponente der Idee entgegensteht einmalig besuchte Orte als Achievements zu loggen. Diese Idee eignet sich allerdings hervorragend dazu eine andere App zu programmieren, mit der man zum Beispiel die verbrachte Zeit in der Universitätsbibliothek loggen kann, um mit seinen Kommilitonen eine Art Lernwettstreit führen kann.

Neben der Idee für einen Studienwettkampf gab es weitere Vorschläge für andere Domänen in denen Apps entwickelt werden könnten. Beispielsweise wurde die Idee formuliert Achievements für gelesene Bücher zu vergeben. Die Verifikation zum Erlangen der Achivements könnte über das einscannen des ISBN Codes auf der Buchrückseite oder bei Ebooks über ein Photo des Covers gewährleistet werden.

Nah verwandt hiermit war die Idee Achievements für angeschaute Filme zu vergeben. Hier besteht allerdings das Problem, wie das Anschauen eines Filmes verifiziert werden kann. Bei Kinobesuchen geht dies über ein Photo des Tickets (Probleme die Bilderkennung mit sich bringt sollen an dieser Stelle außen nicht behandelt werden). Wenn diese Filme im Fernsehen oder über Streaminganbieter geschaut werden, ist die Verifikation ungleich schwieriger.

Ein sehr ergiebiger Teilbereich für Achievement Apps sind Hobbies. Viele Bereiche die im Laufe der Gespräche genannt wurden lassen sich unter diesem Bereich zusammenfassen. Die Möglichkeiten die sich hier bieten sind nahezu unerschöpflich. Für die genaue Einschätzung wie gut sich einzelne Hobbies für eine Umsetzung in eine Achievement App eignen müssten gesondert Gespräche mit Personen geführt werden, die das im Fokus stehende

Hobby betreiben.

Die Gespräche mit den Personen aus der Gruppe der älteren Personen ergaben folgendes:

Mit dieser Gruppe war die Kommunikation wesentlich schwieriger und die Ergebnisse schlechter als mit der ersten Gruppe. Es waren mehr Erklärungen zu Begriffen notwendig. Es dauerte mitunter lange, bis das geplante Konzept den betreffenden Personen verständlich gemacht werden konnte. Bei einigen Gesprächspartnern entstand der Eindruck, dass bis Gesprächsende nicht verstanden worden war, was die App tun sollte und was das Ziel der Fragestellung war.

Diese Kommunikationsprobleme sind meiner Meinung nach der Tatsache geschuldet, dass einige der älteren Personen keine bzw. erst seit kurzem Smartphones besitzen. Daher haben Sie keine ausreichende Vorstellung über die Anwendungsszenarien dieser Technologie und der damit verbundenen Konzepte. Wenn ein grundlegendes Verständnis fehlt, was Apps sind und was sie in der Lage sind mit dem Trägermedium Smartphone zu leisten, dann ist auch eine qualifizierte Aussage zum Leistungsumfang oder weiteren Anwendungsgebieten schwierig bis unmöglich.

Viele der Vorschläge oder Ideen die aus dieser Gruppe kamen waren daher auch nicht für die Arbeit an dieser App nutzbar. Entweder waren die geforderten Funktionen schon in der Grundfunktionalität eines Smartphones enthalten („Soll mir den Weg zu den Achievements anzeigen“) oder lagen weit außerhalb des geplanten Funktionsumfanges der App („Die App soll mir anzeigen ob eine Sehenswürdigkeit behindertengerecht ist und wo die behindertengerechten Zugänge liegen“).

Trotz dieser Probleme kamen einige Themen zur Sprache, die bis zu diesem Zeitpunkt in der Planung nicht berücksichtigt worden waren. Es wurde zum Beispiel der einfache Zugriff auf und die einfache, komplette Löschung der gespeicherte Daten gefordert.

Ein Feature, das von dieser Gruppe vorgeschlagen wurde, dass auch Aufnahme in den Funktionsumfang der App gefunden hat, war die Möglichkeit Freunde zu Achievements einzuladen. Dies kann sowohl als eine Art Vorschlag eine bestimmte Sehenswürdigkeit zu besuchen genutzt werden, als auch als kleiner Wettkampf wer es zuerst schafft das Achievement zu loggen.

Zur Domänenerkundung ist es sinnvoll die Gesprächspartner so zu wählen, dass diese ein gutes bis sehr gutes Verständnis der Domäne haben. Das hier vorliegende Themengebiet Reiseachievements ist in dieser Form neu. Daher

sollten die Gesprächspartner beide zu Grunde liegende Konzepte kennen oder einen Erfahrungshintergrund haben, der er es ihnen ermöglicht sich schnell in das gestellte Problem hineinzudenken.

Ein rudimentärer Hintergrund in Computerspielen ist hier wünschenswert. Zwar kann das Konzept von Achievements auf einer abstrakten Ebene erklärt werden, allerdings gehen dabei viele Konnotationen die durch die Einbindung der Achievements in das größere Ganze des Spieles unterbewusst vermittelt werden verloren.

Bei vielen Sachverhalten die durch die Gespräche geklärt oder in einem anderen Licht betrachten werden sollten, war es bei den älteren Gesprächspartnern notwendig zuerst ein Verständnis der Fragestellug zu vermitteln. Bei den jüngeren Gesprächspartnern genügte oft eine kurze Beschreibung des Problems und es konnte über Lösungen, Ansätze oder Sichtweisen gesprochen werden, da die zu Grunde liegenden Begriffe und Zusammenhänge bekannt waren. Bei den älteren Gesprächspartnern ging es zumeist nicht über ein Verstehen des Problems hinaus. Wer ein Problem nicht kennt oder noch nicht hatte, der hat sich im Allgemeinen auch noch keine oder nur sehr wenige, Gedanken über eine Lösung, oder um die verschiedenen Dimensionen gemacht die hinter diesem Problem stehen.

3.2 Technik

Zu Beginn der technischen Umsetzung der App standen generelle Überlegungen zum Aufbau der App, welche Komponenten programmiert werden müssen und welche Technologien dafür genutzt werden sollen.

Für den geplanten Funktionsumfang der App ist es notwendig zusätzlich zur Implementierung auf einem Smartphone eine Infrastruktur mit Server und angeschlossener Datenbank zu schaffen. Dafür gibt es mehrere Gründe:

Die Liste von Achievements soll dynamisch erweiterbar sein, damit auch Vorschläge von Usern mit in den Kanon der Achievements aufgenommen werden können. Auch ist zu erwarten, dass der Umfang der verfügbaren Achievements zum Zeitpunkt einer Veröffentlichung noch nicht dem endgültigen Stand entsprechen wird. Eine Möglichkeit Achievements hinzuzufügen ohne die App auf dem Smartphone zu aktualisieren ist hier sinnvoll.

Für die geplanten sozialen Funktionen (Freunde zu Achievements einladen, Gruppenachievements) ist eine zentrale Autorität notwendig, die die Freundschaftsbeziehungen der einzelnen User verwaltet.

Auch im Hinblick auf die Erweiterbarkeit auf andere Domänen ist eine weitergehende Infrastruktur mit einem Server und einer Datenbank sehr hilfreich.

Die Technologien, die zum Einsatz kamen wurden wie folgt gewählt:

- App (Android[2]/Java)
- Server (Apache Tomcat[3])
- Datenbank (PostgreSQL[4])

Für die Auswahl der Technologien, die genutzt wurden um diese Teilbereiche zu implementieren, waren die folgende Kriterien ausschlaggebend:

Die Wahl der mobilen Plattform zur Umsetzung der App fiel auf Android, weil dies die Plattform mit dem mit Abstand größten Marktanteil ist[5]. Weiterhin wird Java als Programmiersprache genutzt. Ich habe mit Java durch meine Erfahrungen im Praktikum und in der Universität den größten Erfahrungsschatz. Ein weiterer wichtiger Faktor war die Tatsache, dass mir der Zugang zu einem Mac fehlt und ich somit keine Entwicklungsmöglichkeit für iOS habe. Eine Entwicklung für Windows Phone kam auf Grund der geringen Verbreitung nicht in Frage.

Die Entscheidung für Apache Tomcat als Server ist zum überwiegenden Teil der Tatsache geschuldet, dass dies eine Servertechnologie ist mit der ich während meines Studium und meines Praktikums schon Erfahrung sammeln konnte. Die grundlegenden Prinzipien, wie der Server funktioniert und wie eigene Software darauf installiert werden kann musste nicht neu erlernt werden. Weiterhin ist Apache Tomcat eine der führenden Webserver Technologien[6].

Die Gründe für PostgreSQL als Datenbanktechnologie waren noch einfacher. PostgreSQL ist die einzige Datenbanksystemtechnologie mit der ich bis jetzt in Berührung gekommen bin.

Bei allen Komponenten spielte auch die gute Verfügbarkeit durch Open Source Lizenzen eine Rolle. Hier ist mit den wenigsten Problemen zu rechnen ist, sollte der Code im Rahmen der Bachelorarbeit öffentlich zur Verfügung gestellt werden müssen.

Technische Details oder Spezifikationen zu den einzelnen Technologien waren kein Kriterium. Es fehlte zu diesem Zeitpunkt das Wissen und die Erfahrung um beurteilen zu können, welche technischen Spezifikationen im Laufe des Projekts wichtig werden würden oder ob bestimmte Kennzahlen überhaupt einen Einfluss auf den Ausgang haben würden.

Auch Vergleiche von Leistungsdaten (z.B. Performance von MySQL vs PostgreSQL) waren an dieser Stelle nicht sinnvoll, da noch nicht bekannt war, wo Performancebetrachtungen Sinn machen würden oder ob die getesteten Bereiche genutzt werden würden.

Viele Komponenten und Frameworks die benötigt wurden um die Funktionalität der App zu implementieren sind an dieser Stelle noch nicht aufgeführt. Zu diesem Zeitpunkt des Projekts war noch nicht bekannt, dass diese Komponenten notwendig sein würden.

Insgesamt war die Herangehensweise an dieses Projekt iterativ. Am Anfang standen sehr generelle Überlegungen wie sie oben beschrieben wurden. Nachdem Rahmenbedingungen festgelegt waren (Wahl der Technologien), konnten weitere Teilaspekte geplant werden. Oft kam es hierbei zu unvorhergesehen Nebeneffekten oder es tauchten Problemkomplexe auf, die bis zu diesem Zeitpunkt nicht beachtet worden sind oder deren Existenz erst dann evident wurde. Dies hatte teilweise zur Folge, dass Teile des Projekts stark geändert oder sogar komplett neu überdacht werden mussten. Genaueres dazu im Kapitel 4 „Das Bauen einer App“
.

4 Das Bauen einer App

Am Beginn der Programmierarbeiten an diesem Projekt stand die Entscheidung möglichst viele Probleme mit Hilfe von Frameworks zu lösen. Dies hat mehrere Gründe. Viele Funktionen die zur Umsetzung der App nötig waren, wurden an anderer Stelle schon einmal benötigt und folglich umgesetzt. Auf diesen Erfahrungsschatz von Programmierern zurückzugreifen erschien sinnvoll. Durch den Einsatz von Frameworks wird viel Arbeit gespart, da Dinge die zur Umsetzung von Funktionalität notwendig sind, sich immer wiederholen (z.B. Http Connection Handling von Jersey serverseitig) und durch Frameworks so weit es möglich ist verborgen werden. Dies spart sehr viel sogenannten Boilerplate-Code[7].

Im Laufe des Projekts erwies sich diese Entscheidung wiederholt als richtig. Viele Probleme, die manche Aufgaben noch zeitaufwendiger gemacht hätten, als sie ohnehin schon waren, fielen durch die Nutzung von Frameworks weg (Beispiele dazu später in diesem Kapitel).

Leider erfordern einige Frameworks eine aufwendige Einarbeitung in ihre Funktionsweise und die zu Grunde liegenden Logik. In Hinsicht auf die geplante Erweiterung des Projekts auf eine App Familie lohnt sich diese Einarbeitung. Wäre nur die Reiseachivement App die Grundlage, hätte sich der Einsatz von manchen Frameworks sicherlich nicht gelohnt.

Auch haben alle Frameworks einen ursprünglichen Zweck für den sie entwickelt wurden. Dies kommt in der Benutzung teilweise sehr deutlich zum Tragen. Wenn Funktionalität umgesetzt werden soll, die in dieser Form in einem Framework nicht vorgesehen ist, so ist dies mitunter ebenso aufwendig, wie diese Funktionialität eigenständig, außerhalb der Rahmenbedingungen des Frameworks zu programmieren.

4.1 Android SDK

Der urspüngliche Plan zur Herangehensweise an die Umsetzung des Androidteils des Projektes sah wie folgt aus:

Die Benutzerführung sollte mit einzelnen Bildschirmskizzen geplant werden und auf diese Weise einen Überblick über die Zuordnung von Funktionen zu Teilen der graphischen Oberfläche (Bestandteile der einzelnen Screens) und die Anordnung von funktionalen Elementen innerhalb der Screens geben.

Es wurde schnell klar, dass eine angemessene Planung nicht möglich war, ohne vorher verstanden zu haben, welches Konzept bei Android hinter den einzelnen Screens steht, wie diese Zusammenarbeiten und welche Designprin-

zipien im Hintergrund verwendet werden.

Die Installation von Android Studio war sehr einfach und funktionierte ohne Probleme. Android Studio benutzt standardmäßig Gradle[8] als Build Manager. Im Interesse der einfachen Bereitstellung des Codes im Rahmen der Abgabe dieser Bachelorarbeit und der vereinfachten Handhabung von diversen Abhängigkeiten von zu importierenden externen Bibliotheken wurde in diesem Projekt von Gradle Gebrauch gemacht. Die Handhabung und das Importieren von Abhängigkeiten stellte sich mit Gradle unkompliizierter und komfortabler als beim Konkurrenten Maven[9] dar, das als Build Manager für die serverseitigen Elemente benutzt wurde.

Mit der Inbetriebnahme von Android Studio und des Durcharbeitens von Tutorials ergab sich ein erstes Verständnis wie mit der Planung der Screens fortgefahren werden konnte. Android definiert die einzelnen Screens als Activities deren Lebenszyklus sich wie folgt darstellt:

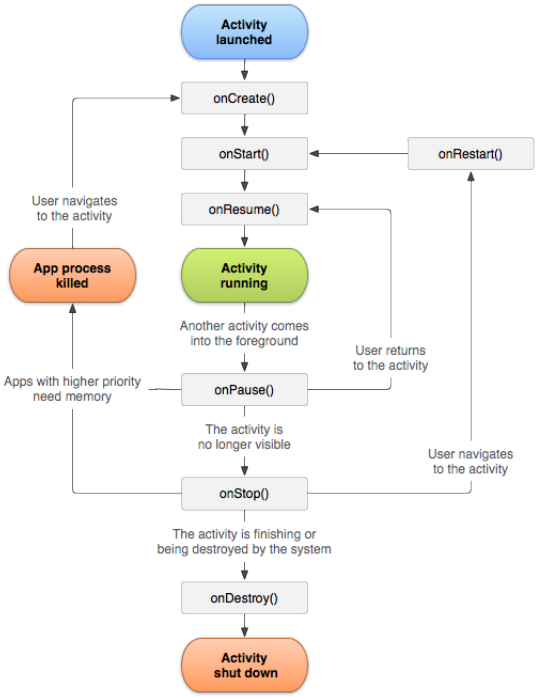

Abbildung 1: Lebenszyklus einer Activity

Hierbei sind die Methoden onCreate() und onStart() für den weiteren Verlauf besonders interessant. In der onCreate() Methode wird der gesamte Initialisierungsvorgang der Elemente der Benutzeroberfläche (z.B. Views) und das Binden von Daten an diese Elemente druchgeführt.

Die onStart() Methode wird aufgerufen, wenn die Activity für den Benutzer sichtbar wird.

Die Funktionalität der einzelnen Oberflächenelemente (wie Klicks auf Buttons) werden über ein Eventsystem umgesetzt.

Das Layout der Activities wird in einer XML Datei festgelegt. Hier werden mit Hilfe von XML die einzelnen Elemente der Oberfläche definiert und angeordnet. Dieses System ist sehr nahe mit dem von C# genutzten WPF[10](Windows Presentation Foundation) System verwandt, mit dem ich während meiner Hobbyprojekte schon Erfahrung sammeln konnte.

Insgesamt war der Aufbau der Oberfläche simpel. Auch die Verknüpfung mit dem dahinter liegenden Java Code stellte kein Probleme dar. View Elemente (Listen, statische und dynamische Textboxen) können nach einer Bindung an ein Java Objekt mit Hilfe von Methoden auf diesem Objekt manipuliert werden (bei Textviews kann beispielsweise der angezeigte Text via setText(String) geändert werden). Die von Buttons ausgelösten Events werden mit Hilfe von Eventhandlern im Java Code abgefangen und lösen entsprechende Aktionen aus.

Aus diesen grundlegenden Bausteinen resultierten die folgenden Activities (Screens):

- Home Bildschirm
- Log Achievement
- Logged Achievements

4.1.1 Home Bildschirm

Diese Activity ist der Ausgangspunkt für registrierte Nutzer der App.

Der Nutzer wird in der obersten Textzeile darüber in Kenntnis gesetzt, wie viele Achivements schon gelogged wurden (X ist hier Platzhalter, da die Funktionalität noch nicht implementiert ist). Ein Klick auf diesen Text führt zur Logged Achievement Activity.

Der Button Log Achievement führt zur Log Achievement Activity, deren Funktionsumfang weiter unten beschrieben ist. Dieser Button soll in der finalen Version dieser App der prominenteste Teil dieses Bildschirms werden. Die hier gezeigte Version ist nur eine vorläufige Version.

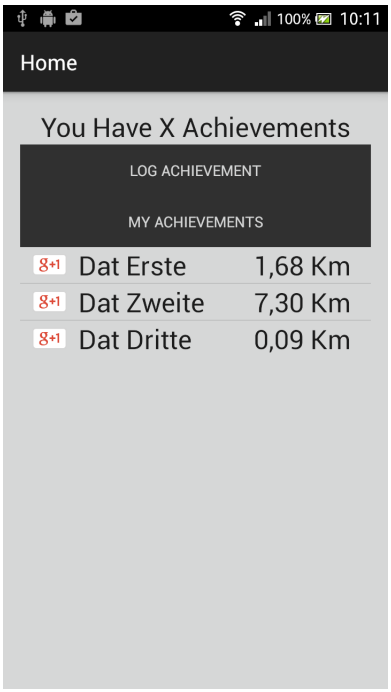

Abbildung 2: Screenshot des Home Bildschirms

Der zweite Button My Achievements dient Debugzwecken und wird in dieser Form nicht in der fertigen App Version enthalten sein. Der Button führt derzeit zum Bildschirm Logged Achievements.

Darunter befindet sich eine Liste, die in der Nähe verfügbare Achievements anzeigt. Es werden Name, zugeordnetes Bild und Entfernung zum eigenen Standort für alle in der Liste vorhandenen Achievements angezeigt. Die zugeordneten Bilder erfüllen außer der visuellen Alleinstellung der einzelnen Achievements keinen weiteren Zweck.

Die Liste soll in der fertigen Version nach Entfernung sortiert sein. Dies ist noch nicht implementiert.

Ein Klick auf die Listenelemente soll jeweils ein Popup erscheinen lassen, in dem eine nähere Beschreibung zum Achievement angezeigt wird. Diese Funktion ist in dieser Version noch nicht implementiert.

Um das Anzeigeverhalten der Liste soweit zu ändern, dass das gewünschte Ergebnis erzielt wird, waren die ersten Anpassungen der Standardelemente des Android SDKs notwendig. Für die Liste wurde ein ListView Element genutzt. Dieses wird über einen ListViewAdapter mit Informationen versorgt,

wie die Elemente anzuzeigen sind. Der ListViewAdapter wird seinerseits mit einer Sammlung von Objekten versorgt, aus der dann die Anzeigeelemente für die Listview entstehen.

Um das Anzeigeergebnis wie oben in Abbildung 2 zu erreichen musste ein eigener ListViewAdapter geschrieben werden, der aus dem vom Server gelieferten Array mit Achievement Objekten die notwendigen Informationen zog (Der Aufbau eines Achievement Objektes wird im Abschnitt 4.2 Server näher beschrieben).

Um die Informationen vom Server zu erhalten, die in der Liste stehen, muss eine Anfrage mit den aktuellen Positionsdaten (Longitude und Latitude) versendet werden. Die Anfrage wurde mit Hilfe des Volley Frameworks[11] realisiert. Die Verwendung dieses Frameworks zur Abwicklung von aus- und eingehenden Verbindungen wird in der offiziellen Dokumentation von Android empfohlen. Volley kapselt sehr viele Dinge, um die sich der Entwickler bei http Verbindungen kümmern muss (u.a. Verbindungsauf und -abbau, Parsen der Anfrage und der Antwort).

Gelöst wird dies von Volley über eine Request Queue (Anfragewarteschlange) an die Request Objekte verschickt und dann abgearbeitet werden. In diesen Request Objekten wird angegeben, wohin die Anfrage gehen soll (inkl. Parameter) und welche Art von Antwort erwartet wird. Im Falle der Anfrage für die Liste wird als Antwort ein Gson Array (Gson[12] ist die JSON Implementierung von Google) erwartet, in dem Achievement Objekte codiert sind. Dieses wird dann in ein Achivement Array umgewandelt und dem Listviewadapter zur Verfügung gestellt.

Die Volley Request Queue wurde als Singleton implementiert. Ein Singleton Muster stellt sicher, dass von einer Klasse immer nur ein Objekt existiert. Im Falle der Request Queue wurde diese Vorgehensweise aus mehreren Gründen gewählt. Zum einen wird die Queue erst initialisiert, wenn sie auch wirklich benötigt wird, zum anderen werden durch die einmalige Initialisierung Ressourcen gespart, da die Volley Request Queue ein recht schwergewichtiges Objekt ist. Weiterhin wird durch das Einsetzen des Singleton Musters garantiert, dass es keine Zugriffskonflikte auf die Kommunikationsmodule des Smartphones gibt.

Realisiert wird ein Singleton Muster, in dem der Konstruktor der Klasse „private“gesetzt wird und das Objekt der Klasse nur als Rückgabe einer öffentlichen Methode zugegriffen werden kann. So wird sichergestellt, dass immer nur ein Objekt gleichzeitig existiert.

Die zur Bestimmung des Standortes des Handys notwendige Nutzung der

Google Play API[13]führte zu einigen Problemen:

Es traten Probleme mit dem Zusammenspiel des Google Play Api Clienten und der Initialisierung der Komponenten der Oberfläche auf. Für die Anfrage an den Server, der die Objekte liefern sollte, die für das Füllen der Liste notwendig sind, muss die Position des Handys bekannt sein. Das Location Objekt, das vom Google Client geliefert wird, in dem diese Position gespeichert wird, ist erst verfügbar, nachdem der Google Client erfolgreich initialisiert wurde und die Abfrage an das GPS Modul erfolgreich ist. In der Programmlogik bedeutet dies, das die Initialisierung des Google Clients in der onCreate() Methode der Activity angestossen wird. Danach laufen Initialisierung des Google Clients und der weitere Ablauf der onCreate() Methode parallel. Der Google Client ruft nach erfolgreicher Initialisierung seine onConnect() Methode auf. Erst ab diesem Zeitpunkt kann das Location Objekt verwendet werden. Dies führte dazu, dass die gesamte Initialisierung der Oberflächen Elemente nicht in der eigentlich dafür vorgesehenen onCreate() Methode der Activity stattfinden konnte, sondern in die onConnect() Methode des Google Clients ausgelagert werden musste.

Diese Umleitung der Programmlogik hätte durch die Verwendung eines Singleton Patterns für den Google Play API Clients vermieden werden können. Die Initialisierung des API Clients beim Starten der App, hätte die Initialisierung des Location Objekts zu einem Zeitpunkt geschehen lassen, zu dem es noch nicht unmittelbar notwendig gewesen wäre. Der Client hätte Zeit gehabt seine Initialisierung abzuschließen, die Position des Handys auszulesen und im Location Objekt zur Verfügung zu stellen bevor die Position in den Anfragen an den Server notwendig benötigt wird. Leider fehlte die Zeit um dies umzusetzen.

Ein weiteres Problem, das mit der Einbindung des Google Play API Clients auftauchte war, dass der mit Android Studio ausgelieferte Emulator die App nicht mehr emulieren konnte. Bis zur Einbindung des Google Play API Clients funktionierte die Ausführung der App auf dem Emulator problemlos. Als der API Client in Betrieb genommen war und die Position abfragen sollte, gab der Emulator nur noch die Meldung aus die Google Play Services upzudaten. Trotz intensiver Bemühungen und Versuche, war dieses Problem nicht in den Griff zu bekommen. Alle Lösungsansätze die vom Hersteller oder in diversen einschlägigen Foren verfügbar waren (u.a. manuelles Upaten der Play Services auf dem Emulator, spoofen der GPS Koordinaten mit den eingebauten Funktionen), führten nicht zum Erfolg. Auch der Umstieg auf einen anderen Emulator (Genymotion[14]) führte nicht zu dem gewünschten Ergebnis.

Letztendlich musste ein Android Handy (Android Version 4.1.2) zum Testen

der App verwendet werden. Hier war die implementierte Funktionalität ohne weitere Änderungen am Code sofort verfügbar und funktionierte so wie beschrieben und gedacht.

Den Funktionsumfang und die Robustheit der verfügbaren Android Emulatoren habe ich als sehr unzureichend wahrgenommen. Zwar laufen die grundlegenden Funktion (Oberflächen Elemente korrekt anzeigen, per Knopfdruck Activities wechseln) problemlos, allerdings gibt es bei weitergehender Nutzung der normalen Smartphonefunktionen Probleme. Trotz intensiver Bemühungen war eine Nutzung von GPS Daten nicht möglich.

4.1.2 Log Achievement

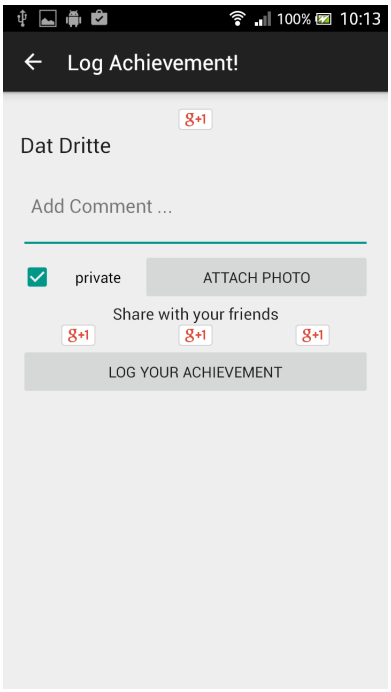

Abbildung 3: Screenshot des Log Achievement Bildschirms

Im oberen Teil des Bildschirms wird ein Achievement angezeigt, das in der Nähe (der Umkreis in dem ein Achivement gelogged werden kann beträgt ca. 100 m) ist, inklusive Bild und Beschreibung. Sollte sich in der Nähe kein Achivement befinden, so wird ein Platzhaltertext angezeigt: „No Achievements Near By“.

Darunter befindet sich ein editierbares Textfeld mit dem ein eigener Kommentar erstellt werden kann.
Des Weiteren besteht die Möglichkeit das Achievement per Checkbox privat

zu machen und ein Photo hinzuzufügen.

Ein Achievement als privat zu kennzeichnen bedeutet, dass dieses nicht angezeigt wird, wenn andere Personen über die in der App verfügbaren Mittel (Liste von Achievements von Freunden) auf die Achievements der betreffenden Person zugreifen.

Im unteren Teil gibt es drei Buttons mit denen das Achievement in sozialen Netzwerken geteilt werden kann (Facebook, Twitter, Instagram). Allerdings ist diese Funktionalität auf Grund von Zeitmangel nicht implementiert.

Im unteren Teil befindet sich ein Knopf mit dem das Achievement mit Kommentar, Photo und Einstellungen geloggt werden kann, d.h. das loggedAchievement Objekt wird zum Server geschickt und dort in die Datenbank geschrieben.

Das Versenden das Objektes geschieht wieder über die RequestQueue von Volley. Vor dem Versenden wird das Objekt noch mit Hilfe von Gson in einen JSON String umgewandelt.

Die in dieser Activity aufgetauchten Probleme sind die gleichen, die schon im vorhergehenden Abschnitt besprochen wurden. Die einzige Veränderung im Funktionsprofil dieser Activity ist das Versenden eines kompletten Objektes anstatt nur einzelner Parameter. Durch die vom Volley Framework und Gson zur Verfügung gestellten Funktionalität traten hier keine neuen Probleme auf.

4.1.3 Logged Achievement

Die obere Liste zeigt die schon erreichten Achievements an. Es werden Bild, Name und vergangene Zeit zum Logzeitpunkt angezeigt. Die Liste kann nach Name und Logzeitpunkt sortiert werden. Ein Klick auf die Achievements soll ein Popup erscheinen lassen. In diesem Popup sollen der Name, der Logzeitpunkt, das zugehörige Achievementbild, der selber verfasste Kommentar, das eventuell angefügte Bild und der Privatstatus des Achievements angezeigt werden. Es soll zusätzlich die Möglichkeit bestehen, den Privatstatus zu ändern. Dieses Popup und die Sortierung sind auf Grund von Zeitmangel nicht implementiert.

Die untere Liste zeigt alle verfügbaren Achievements an. Hierbei ist die Anzahl der Achievements die vom Server geladen werden auf die 100 nächsten (basierend auf dem eigenen Standpunkt) beschränkt. Diese Liste kann nach Name und Entfernung des Achievement sortiert werden. Ein Klick auf ein Achievement soll ein Popup erscheinen lassen. In diesem Popup sollen Na-

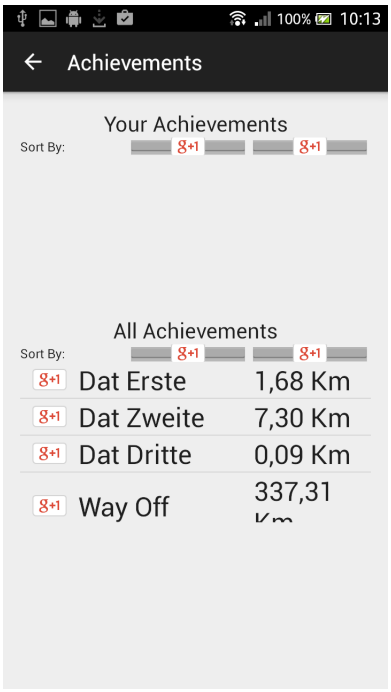

Abbildung 4: Screenshot des Logged Achievement Bildschirms

me, zugehöriges Achievementbild, Beschreibung und Entfernung des Achievements angezeigt werden. Wie bei der oberen Liste sind die Sortierfunktion und das Popoup aus Zeitmangel nicht umgesetzt worden.

In der aktuellen Version der App ist die obere Liste nicht funktionsfähig. Es gibt Verdachtsmomente warum dieser Fehler auftaucht. Aus Zeitmangel konnte dieser aber nicht behoben werden.

Der mit dem Volley Framework verbundene Gson Parser, erkennt die vom Server gesendete Antwort nicht als Json Array bestehend aus loggedAchievement Objekten. Ein Debuggen der Anwendung ergab, dass die Antwort vom Server ein korrektes Json Array liefert. Ich vermute, dass die Formatierung des in dem Array enthaltenden java.lang.Date Feldes (Log Zeitpunkt) für den Fehler verantwortlich ist. Mit diesem Objekt gab es schon auf Serverseite Probleme (dazu mehr im Abschnitt 4.2 dieser Arbeit).

4.1.4 Geplante aber nicht umgesetzte Activities

Die folgenden Activities sind auf Grund von Zeitmangel nicht umgesetzt worden:

- Registrationsdialog
- Friends
- Optionen/Konfiguration

Der Registrationsdialog soll der erste Bildschirm sein, den ein neuer User beim ersten Starten der App sieht. Hier soll eine einfache Registrierung als neuer User oder ein Registrieren mit einem Sozial Media Account (Facebook, Twitter oder Instagram) erfolgen. Dies ist notwendig um den Benutzer eindeutig zu identifizieren und in Erfahrung zu bringen wer die Freunde sind und ob diese eventuell schon diese App nutzen.

Die Friends Activity soll anzeigen welche Freunde aus Sozialen Netzwerken auch diese App nutzen und welche Achievements diese schon errungen haben. Es gibt noch keine genaue Planung dieser Activity, da andere Aspekte und Probleme zu viel Zeit in Anspruch genommen haben.

In der Optionen/Konfiguration Activity soll die Möglichkeit bestehen die Accounts der Sozialen Netzwerke zu verwalten (hinzufügen und löschen) und den kompletten Account mit allen zugehörigen Daten zu löschen.

4.2 Server

Das Einrichten einer Apache Tomacat Instanz und das Einbinden dieser in die Eclipse Entwicklungsumgebung zum einfachen Testen der Serverkomponente ging problemlos von statten und funktionierte auf Anhieb. Insbesondere die einfache und komfortable Einbindung in Eclipse ist hier hervorzuheben.

Die Kommunikation zwischen App und Server wurde serverseitig durch eine REST Schnittstelle realisiert. Die Entscheidung für eine REST Schnittstelle ist durch die Leistungsfähigkeit der existierenden Frameworks begründet. Das zu diesem Zweck gewählte Jersey[15] stellt alle Funktionen zur Verfügung die zur Umsetzung der geplanten Funktionalität nötig sind.

Für jede Anfrage steht eine eigene URL zur Verfügung, und es können beliebige neue URLs hinzugefügt werden. Das komplette Handling der Verbindungen wird durch das Framework übernommen. Auch das Parsen der eingehenden Parameter und der ausgehenden Antworten ist in den Funktionen von Jersey enthalten. Der auf Serverseiten erforderliche Code ist sehr gering im Umfang. Hinter jeder definierten Anfrage URL steht eine Methode, die als Eingabeparameter die Query Parameter (oder ein Json Objekt) geliefert bekommt und als Rückgabewert (in den hier implementierten Anwendungsszenarien) ein Json Objekt liefert.

So gut die Realisierung des Servers mit Jersey funktionierte so viele Probleme gab es allerdings mit der Anbindung der Datenbank.

Der gesamte Zugriff (sowohl Speichern als auch Abrufen von Daten) auf die Datenbank wurde mit einem Object-Relation-Mapper durchgeführt. Das hierfür eingesetzte Framework ist Hibernate [16].

Die Gründe einen OR Mapper für den Datenbankzugriff zu verwenden sind vielvältig. Der OR Mapper ist dafür zuständig Tabellen der Datenbank direkt mit Java Objekten zu verknüpfen. Dabei wird das Mapping der einzelnen Felder der Objekte auf die Spalten einer Tabelle der Datenbank automatisiert. Dies muss bei Hibernate über Annotationen direkt in der Klasse oder über eine gesonderte Konfigurationsdaten festgelegt werden. In der Theorie werden dadurch sehr viel Code und Mühen gespart. Es muss kein Handling der Verbindungen mit der Datenabank erfolgen und die Zuordnung der Spaten zu den Feldern der Java Objekts geschieht an einem zentralen Ort und nicht bei jeder einzelnen Abfrage.

Leider gab es mit dieser Komponente in der Praxis sehr viele Probleme. Diese waren so gravierend, dass die Architektur der Datenbank angepasst werden musste. Der ursprüngliche Plan war mit Hibernate nicht umzusetzen.

Zu Beginn des Projekts war es geplant die Datenbank wie folgt zu gestalten:

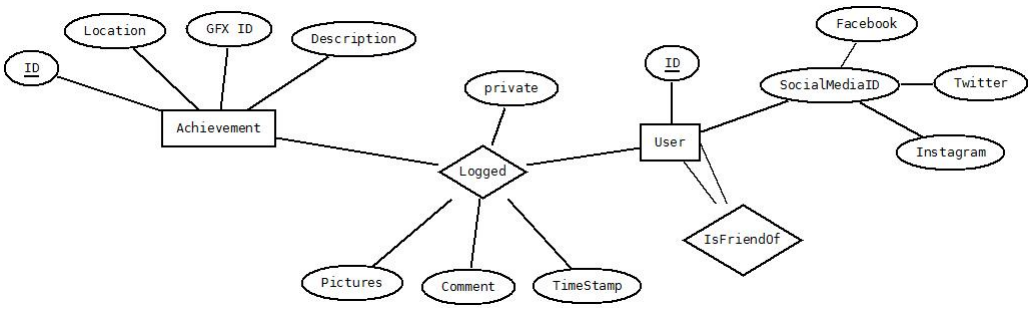

Abbildung 5: Erstes Design der Datenbank

Die Beziehungen zwischen den Entitäten haben folgende Kardinalitäten:

- Achievements m:n User
- User m:n isFriendOf

Auf Grund der Probleme die mit der Handhabung von Hibernate bestanden, musste das Datenbank Design dahin gehend verändert werden, dass alle Beziehungen die in dem oben stehenden Diagramm in der Datenbank nicht wie

üblich über Fremdschlüssel abgebildet wurden, sondern keinerlei Verknüpfungen zwischen den Tabellen bestanden.

Ein Überblick über den endgültigen Aufbau der Datenbank ist in Sektion 4.3 zu finden.

Wie schon angedeutet ist die Struktur der Datenbank hauptsächlich den Problemen mit Hibernate geschuldet. Hibernate bildet die Felder einer Klasse direkt auf Spalten in der Datenbank ab. Eine Beziehung erfordert in beiden an dieser Relation beteiligten Klassen ein weiteres Feld in Form eines Sets. In diesem Set werden die Fremdschlüsselbeziehungen verwaltet. Bei einer Verwendung der Klassen und Objekte im Kontext des Servers funktionierte dieses System nach einigen Anlaufschwierigkeiten ohne Probleme. Nun werden, wenn ein User ein Achievement loggt Objekte vom Typ loggedAchievement in der App erstellt und über den Umweg eines Json Objektes an den Server übertragen. Trotz vieler Versuche war es mir nicht möglich das Feld mit dem Set zur Darstellung der Fremdschlüsselbeziehungen in der App korrekt zu erzeugen.

Die Dokumentation von Hibernate und Beiträge in diversen Foren führten zu keiner Lösung dieses Problems. Auch eigene Nachforschungen waren durch die starke Kapselung der Datenbankabfragen nicht erfolgreich. Hibernate nutzt eine eigene, an der SQL Syntax orientierte Abfragesprache. Diese wird dann als Abfrage über das Framework an die Datenbank geschickt. Was dort im Hintergrund geschieht war leider auch durch intensives Debuggen nicht in Erfahrung zu bringen.

Es wurden also alle Fremdschlüsselbeziehungen aus der Datenbank entfernt und die Abhängigkeiten ausschliesslich über die Programmlogik sichergestellt. Dies ist zwar keine gute Lösung, aber die einzige, die mit den engen Zeitbeschränkungen umzusetzen war. Hätte mehr Zeit zur Einarbeitung mit Hibernate und der Umsetzung der Relationen in diesem Framework zur Verfügung gestanden, wäre es möglich gewesen eine bessere Lösung als die aktuelle zu finden.

Die Nutzung des Hibernate Frameworks war eine Entscheidung bei der ich mir in der Nachbetrachtung nicht sicher bin, ob sie richtig war. In der Theorie bringt Hibernate viele Vorteile mit sich und erleichtert die Handhabung von Datenbanken, insbesondere bei größeren Projekten, ungemein. Allerdings hatte ich mit der Handhabung von Hibernate so große Probleme, dass nicht nur das Design der Datenbank verändert werden musste, sondern auch fast zwei Wochen der Bearbeitungszeit ausschließlich für Problembehebungen im Zusammenhang mit Hibernate aufgewendet werden mussten. Wäre eine längere Einarbeitungszeit praktikabel gewesen und wäre das Projekt

größer im Umfang so bestände kein Zweifel an der Sinnhaftigkeit des Einsatzes von Hibernate. Unter den gegebenen Umständen und im Rückblick muss der Einsatz von Hibernate aber als Fehler gewertet werden.

4.3 Datenbank

Die Datenbank besteht aus den oben genannten Gründen aus drei vom ursprünglichen Design abweichenden Tabellen mit folgenden Spalten:

Tabelle achievement: id, name, location_ e, location_ n, gfx_ id, description

Diese Tabelle speichert die einzelnen Achievements. Die location Spalten bilden zusammen die GPS Postion des Achievements ab. gfx_ id ist eine interne Repräsentation des zugeordneten Bildes.

Tabelle logged_ achievement: id, useremail, achievement_ id, comment, pic, time_ stamp, private

Diese Tabelle speichert die geloggten Achievements. useremail und achievement_ id dienen hier als Zuordnung welcher User welches Achievement geloggt hat. Diese Spalten sind keine Fremdschlüssel in der Datenbank. Sie werden nur von der Programmlogik der App verwaltet.

Tabelle users: id, email, facebook, twitter, instagram

Diese Tabelle speichert die Benutzer der App und ihre Accounts bei Sozialen Netzwerken.

Die isFriendsOf Relation wurde noch nicht in die Datenbank aufgenommen. Sie würde in einer weiteren Tabellen realisiert werden in der Emails von Benutzern gepaart werden um die Freundschaftsbeziehung zwischen diesen darzustellen.

Die gravierendsten Probleme mit der Datenbank sind, wie schon im vorhergehenden Abschnitt geschildert, der Verwendung von Hibernate zuzuschreiben. Allerdings gab es auch andere Probleme die nur mittelbar mit Hibernate zu tun hatten.

Es war nicht möglich ein Objekt vom Typ java.lang.Date zuverlässig in das PostgreSQL eigene Date Format zu übersetzen. Die Spalte time_ stamp speichert als Datentyp Text, der eins zu eins aus dem Json Objekt übernommen wird. Das von Gson in der App übersetzte Feld vom Typen java.lang.Date wird auf dem Server vor dem Speichern in der Datenbank also nicht mehr

zu einem anderen Datentyp als String geparsed. Ich vermute, dass hier die Probleme mit der Erkennung des loggedAchievement Json Arrays liegen, die ich in Abschnitt 4.1.3 angesprochen hatte. Zwar sollte beim Speichern des Datums als Text keine Information verloren gehen, da dieses ja auch als Text im Json Array übertragen wird, allerdings ist dies die beste Erklärung die bis zum Zeitpunkt der Abgabe gefunden werden konnte.

5 Evaluation

5.1 Rückmeldung von Usern

Es wurden zur Evaluation der App Tests mit verschiedenen Personen durchgeführt. Dabei wurde die Funktionsweise der App demonstriert und den Nutzern Zeit gegeben die App selber auszuprobieren. Danach wurden die Testpersonen nach Verbesserungsvorschlägen und allgemeinem Feedback befragt. Leider waren die Ergebnisse nicht aussagekräftig.

Es konnten auf Grund der begrenzten Bearbeitungszeit und den gravierenden Problemen bei der Umsetzung des geplanten Funktionsumfanges (siehe vorherige Kapitel) nur eine halbfertige Version der App getestet werden.

Viele geplante Funktionen funktionieren noch nicht richtig oder sind gar nicht vorhanden. Zum Beispiel ist der gesamte Funktionsumfang im Zusammenhang mit den Sozialen Netzen noch nicht integriert. Weiterhin ist die Benutzeroberfläche noch in einem sehr frühen Stadium. Alle Elemente sind nur grob angeordnet und viele Dinge die später zur Benutzererfahrung beitragen sollen (u.a. durch Farbschema und Form hervorgehobene Buttons, Icons zur Repräsentation der Achievements, eigenes Farbschema für die Gesamtanmutung der App) sind nicht integriert oder nur durch Platzhalter angedeutet.

Diese Umstände machten es schwer eine aussagekräftige Rückmeldung der Testpersonen zu erhalten. Die Reaktionen waren durchgehend positiv, allerdings bezogen sich die Verbesserungsvorschläge ausschließlich auf Dinge die noch nicht in ihrer endgültigen Form vorhanden waren.

Eine interessante Erkenntnis, die erst in diesen Tests auftauchte war, dass die Funktionalität dieser App in leicht abgewandelter Form schon in Facebook integriert ist. Man kann dort in Orte „einchecken“, d.h. anderen Nutzern mitteilen, dass man gerade an einem bestimmten Ort ist oder diesen besucht hat. Diese Funktion macht diese App allerdings nicht überflüssig, da in der App die Möglichkeit geschaffen wird, sich einen Überblick über die schon besuchten Orte einer befreundeten Person zu verschaffen. Dies ist mit Facebook alleine nicht möglich.

Aussagekräftige Ergebnisse von Userbefragungen können erst erwartet werden, wenn ein voll funktionsfähiges Produkt zur Verfügung steht oder gezielt nur bestimmte voll umgesetzte Teile der Funktionalität getestet werden. Die Tests mit dieser App wurden in einem Stadium der Entwicklung durchgeführt in der Tests mit Endbenutzern nicht sinnvoll sind. Dies ist dem engen Zeitrahmen in dem die Entwicklungsarbeit stattfand geschuldet und wird

nicht zur Nachahmung empfohlen.

Die Iterationsschritte die im Expose zu dieser Bachelorarbeit angekündigt wurden, konnten leider nicht durchgeführt werden, da nicht einmal der erste Prototyp mit vollem Funktionsumfang fertiggestellt werden konnte.

5.2 Technik

Die Entscheidung in der Entwicklung der App nach Möglichkeit alle Teilaufgaben mit Hilfe von Frameworks zu realisieren ist wichtiger Bestandteile der Methodik dieser Bachelorarbeit.

Die Erfahrungen mit den Frameworks zeigen, dass nach einer gewissen Anlaufphase die zu erledigenden Aufgaben teilweise sehr vereinfacht werden. Dieser Eindruck basiert allerdings nur auf der Abschätzung der Aufgaben die durch das Framework übernommen werden.

Leider waren die anfänglichen Schwierigkeiten im Umgang mit den Frameworks teilweise erheblich. Auch die Handhabung von manchen Frameworks erschien sehr umständlich. Für eine abschließende Beurteilung fehlt mir die Erfahrung. Um sicher beurteilen zu können um welchen Faktor der Einsatz von Frameworks die Arbeiten an diesem Projekt vereinfacht hat, muss ein ähnliches Projekt ohne den Einsatz von Frameworks gegenüber gestellt werden. Dies würde den Rahmen dieser Bachelorarbeit bei weitem übersteigen.

Um die Leistungsfähigkeit des Servers und der angeschlossenen Datenbank zu testen muss zuerst entschieden werden wo und wie der Server betrieben wird. Erst nach dieser Entscheidung können anhand der Rahmenbedingungen (Leistungsdaten der Maschine auf der die Serversoftware läuft, Daten zur Internetanbindung) Testprotokolle entwickelt werden.

Um die Leistungsfähigkeit der App zu bestimmen müssen Tests auf verschiedenen Smartphones durchgeführt werden. Sowohl Leistungsdaten der Telephone sind hier zu berücksichtigen, als auch die jeweils installierte Android Version. Neben Zeit und Ressourcenmangel war der Hauptgrund für das Nichtdurchführen dieser Test der Fakt, dass die App nicht fertig ist.

Da der vollständige Funktionsumfang noch nicht realisiert ist, ist davon auszugehen, dass sich der Ressourcenverbrauch der App noch signifikant verändern wird. Noch nicht einmal die Kompatibilität zu einer Android Api Version lässt sich zuverlässig testen, da mit der Umsetzung von neuer Funktionalität auch hier Veränderungen zu erwarten sind.

Bei der Testnutzung der App zu Entwicklungs und Debugzwecken fielen keine

Leistungseinbrüche auf. Allerdings fand die Testnutzung in einem geschlossenen Wlan mit sehr guter Verbindungsqualität und einer sehr kleinen Datenbank statt. Rückschlüsse auf die Leistung der App in einem realistischen Anwendungsbereich (Daten über GSM Modul, viele Nutzer, Datenbank mit vielen Achievements unter Anfragelast) lassen sich hieraus nicht ableiten.

Im Programmcode der App ist noch sehr viel Redundanz vorhanden. Die Benutzung der Google Play Api wird von jeder Activity selbst verwaltet. Es war geplant hier ein Singleton Pattern einzusetzen um Codezeilen und Ressourcen zu sparen, dies wurde aber aus Zeitmangel nicht mehr umgesetzt.

Eine Erweiterung der Domäne ist problemlos möglich. Zwar müssen für jede Domäne angepasste Apps programmiert werden, allerdings lassen sich die Datenbank und Server Komponente einfach erweitern.

Im Jersey Framework lassen sich sehr einfach neue URLs schaffen, die neue Abfragen bearbeiten. Die Datenbank ist auch sehr einfach erweiterbar. Zwar wäre es sinnvoll, eine neue Datenbank aufzusetzen um die logische und phsysische Trennung der Domänen zu gewährleisten. Aber eine Erweiterung der vorhandenen Struktur ist auch denkbar. Der Fakt, dass keine Abhängigkeiten zwischen den Tabellen in der Datenbank gibt ermöglicht dies.

6 Fazit

Die Realisierung der App, wie sie zum Ende des zwölf wöchigen Bearbeitungszeitraums vorliegt, entspricht nicht meinen Erwartungen. Jegliche Zeitplanungen die ich zu Beginn oder während des Projekts aufstellte, mussten im Laufe der Bearbeitung immer wieder angepasst werden und erwiesen sich letzendlich immer als völlig unzureichend.

Alle Teilschritte dauerten sehr viel länger als geplant. Dies ist verschiedenen Faktoren geschuldet. Durch meine Unerfahrenheit unterschätzte ich oft die Komplexität von Problemen und den Aufwand für die Implemenierung von Lösungen. Auch dauerte die Einarbeitung in die diversen Frameworks und die Inbetriebnahme (gemeint ist hier der Zeitraum von Installation bis mit der Umsetzung der beabsichtigten Funktionalität begonnen werden konnte) bei manchen Komponenten sehr lange. Der Zeitmangel wirkte sich negativ auf den Funktionsumfang der App und die Qualität der Umsetzung aus.

Auch viele der Designentscheidungen wurden mir durch die Nutzung von Frameworks regelrecht aufgezwungen. Zuweilen fühlte sich die Arbeit nicht wie ein Umsetzen von Ideen an, sondern ein Zurechtbiegen des Minimalziels, sodass es nicht mit den Beschränkungen des einen oder anderen Frameworks kollidierte. Trotz dieser gefühlten Einschränkungen durch die Frameworks bin ich mir sicher, dass ohne die Frameworks ein wesentlich schlechteres Ergebnis erzielt worden wäre.

Die zwölf Wochen Bearbeitungszeit waren in meinen Augen für das geplante Projekt, in Anbetracht meiner geringen Vorkenntnisse, viel zu kurz. In dieser Zeit alle Technologien fast von Grund auf neu zu erlernen, die nötig sind um eine komplett funktionsfähige und gut benutzbare App zu programmieren, auch wenn sie wie hier einen relativ geringen Funtionsumfang hat, ist nicht möglich.

Wenn während der Arbeiten unmittelbaren Zugriff auf erfahrene Programmierer möglich wäre, die Fragen schnell und unkompliziert beantworten können oder Probleme schon kennen, ist es durchaus möglich ein Projekt in diesem Umfang im vorgegebenen Zeitrahmen erfolgreich abzuschließen. So wurde zu viel Zeit mit Recherchearbeit (zu benutzende Technologien, generelle Lösungsansätze) und dem (teilweise sehr zeitraubenden) Suchen nach Fehlern verbraucht.

Sehr interessant war die Vielfältigkeit und Verschiedenheit der zu lösenden Probleme. Die gleichzeitige Entwicklung von App, Serverkomponente und Datenbank stellte mich vor teilweise sehr unterschiedliche Probleme. Das praktisch gleichzeitige Hineindenken in Probleme und das Erkunden der Lö-

sungen war nicht immer leicht. Die Systematik und Lösungsansätze die von den einzelnen Frameworks für die jeweiligen Problemkomplexe angeboten werden sind teilweise fundamental von einander verschieden.

Insgesamt war diese Bachelorarbeit ein sehr interessantes Projekt bei dem ich sehr viel gelernt habe. Auch wurden mir neue Perspektiven auf die Tätigkeit des Programmierens eröffnet, die mir sicherlich bei künftigen beruflichen Tätigkeiten sehr hilfreich sein werden. Gleiches gilt für die genutzten Arbeitstechniken.

Abbildungsverzeichnis

Literatur

[1] Kein Autor. (2015-08-02). Google Web Toolkit [Online], Adresse: `http://www.gwtproject.org/`.

[2] K. Autor. (2015-08-02). Android Studio [Online], Adresse: `https://developer.android.com/sdk/index.html`.

[3] The Apache Software Foundation. (2015-08-02). Apache tomcat [Online], Adresse: `http://tomcat.apache.org/`.

[4] The PostgreSQL Global Development Group. (2015-08-02). Postgresql [Online], Adresse: `http://www.postgresql.org/`.

[5] IDC Corporate USA. (2015-08-02). Smartphone OS Market Share, Q1 2015 [Online], Adresse: `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`.

[6] Netcraft Ltd. (2015-08-02). May 2015 web server survey [Online], Adresse: `http://news.netcraft.com/archives/2015/05/19/may-2015-web-server-survey.html`.

[7] Wikipedia. (2015-08-02). Boilerplate code [Online], Adresse: `https://en.wikipedia.org/wiki/Boilerplate_code`.

[8] Gradle Inc. (2015-08-02). Gradle [Online], Adresse: `https://gradle.org/`.

[9] The Apache Software Foundation. (2015-08-02). Apache maven project [Online], Adresse: `https://maven.apache.org/`.

[10] Wikipedia. (2015-08-02). Windows presentation foundation [Online], Adresse: `https://de.wikipedia.org/wiki/Windows_Presentation_Foundation`.

[11] Kein Autor. (2015-08-02). Volley [Online], Adresse: `http://developer.android.com/training/volley/index.html`.

[12] K. Autor. (2015-08-02). Gson [Online], Adresse: `https://github.com/google/gson`.

[13] Android Developers Blog. (2015-08-02). Google apis for android [Online], Adresse: `https://developers.google.com/android/guides/api-client`.

[14] Genymobile. (2015-08-02). Genymotion [Online], Adresse: `https://www.genymotion.com/`.

[15] Oracle Corporation. (2015-08-02). Jersey - restful web services in java [Online], Adresse: `https://jersey.java.net/`.

[16] redhat. (2015-08-02). Hibernate. everything data [Online], Adresse: `http://hibernate.org/`.