ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Mobile Application Development

Ausarbeitung zur App "Coaster2Go" an der Universität Ulm

Vorgelegt von:
Fabian Fischbach, Luis Beaucamp und Tim Stenzel

Gutachter:
Marc Schickler

Betreuer:
Marc Schickler

2017

Fassung 11. August 2017

© 2017 Fabian Fischbach, Luis Beaucamp und Tim Stenzel

This work is licensed under the Creative Commons. Attribution-NonCommercial-ShareAlike 3.0 License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-sa/3.0/de/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.

Satz: PDF-LaTeX 2$_\varepsilon$

Inhaltsverzeichnis

1

Einleitung

In dieser Dokumentation wird die Entwicklung einer App im Rahmen des Anwendungsfaches Mobile Application Development an der Universität Ulm und die dadurch erstellte App in Form einer Freizeitpark-App vorgestellt.

1.1 Motivation und Problemstellung

Die Problemstellung war für dieses Projekt wesentlich offener gegeben, da wir eine beliebige App für unsere präferierte mobile Platform entwickeln sollten. Nachdem wir einige Ideen gesammelt hatten, uns Gedanken zur Implementierung gemacht haben und die Vor- und Nachteile abgewägt hatten, entschieden wir und für eine Freizeitpark-App. Als zentrale Platform für Friezeitparkliebhaber sollten Parks mit Standorten, zugehörige Attraktionen mit Wartezeiten und Bewertungen verfügbar sein.

Wenn man einmal die aktuelle Marktlage betrachtet fällt auf, dass es nur Apps für einzelne bestimmte große Parks, wie z. B. Disneyworld, gibt. Bei genauerer Betrachtung zeigen diese Apps aber Mängel auf. Sie bieten kaum oder nur wenig Funktionen, sind teilweise recht veraltet und nicht benutzerfreundlich designt.

Eine kleine Nebeninformation: Mit täglich ca. 15000 Besuchern ist der Europapark der meistbesuchte Freizeitpark Deutschlands.

Man sieht also, dass wenn viele Freizeitparks in der App verfügbar sind, mehr als genug potentielle Nutzer der App vorhanden sind, die wir mit unserer App ansprechen möchten.

1.2 Zielsetzung

Da also auf dem Markt eine Nachfrage nach einer Freizeitpark-App entsteht, wollen wir diese ausnutzen um im Laufe dieses Projekts eine eigene Anwendung zu entwickeln. Diese wollen wir auf Basis von Android und Java implementieren. Dabei geht es uns primär darum, den Umgang mit den neuen Techniken (beispielsweise GPS) zu erlernen und unsere bereits vorhandenen Kenntnisse zu erweitern.

Inhaltlich möchten wir eine Anwendung entwickeln in der es möglich sein sollte aus einer Liste von Freizeitparks, die Attraktionen eines bestimmten anzusehen. Inklusive zu den Attraktionen bieten wir die zugehörigen Wartezeiten, Statistiken und Bewertungen. Ebenfalls sollte es die App dem Benutzer ermöglichen selbst Wartezeiten einzutragen und Parks bzw. die Attraktionen zu bewerten. Außerdem sollen in einem Admin-Bereich die Funktionen zur Erstellung und Bearbeitung eigener Parks und Attraktionen vorhanden sein. Des Weiteren stehen auch die Benutzerfreundlichkeit, das Design und die Kompatibilität mit möglichst vielen Android-Smartphones im Vordergrund, sodass ein breites Spektrum an Nutzern angesprochen wird.

1.3 Struktur der Arbeit

In dieser Dokumentation werden zuerst einmal die Grundlagen der Bewertungen und Wartezeitenberechnung erklärt, sowie Android vorgestellt und unsere verwendeten Frameworks präsentiert. Den Hauptteil bildet die Implementierung, welche unsere App im Allgemeinen und mit ihren Besonderheiten vorstellt. Außerdem wird darin unsere Architektur gezeigt und wir erleutern Schwierigkeiten, die wir während der Implementierungsphase hatten. Abschließend gibt es einen Anforderungsabgleich und einen Ausblick auf die Zukunft des Projektes.

Abb. 1.1: Coaster2go Logo

2

Grundlagen

2.1 Grundlagen der Bewertungen und Wartezeitenberechnung

Da Freizeitparks mit sehr großen Besucherströmen zu tun haben, entstehen schnell lange Schlangen von Menschen an den verschiedensten Stellen, wie Attraktionen oder Essensausgaben, im Park. Die Zeit, die ein Besucher an einer Attraktion mit Warten verbringt, bezeichnet man als Wartezeit der Attraktion. Diese kann vom einstelligen bis hin zu einem dreistelligen Minutenbereich andauern. Größere Freizeitparks sind mittlerweile sehr gut darin, Besucherzahlen abzuschätzen und daraus aktuelle oder durchschnittliche Wartezeiten anzugeben. Diese werden meist am Eingang einer Attraktion angegeben.

Abb. 2.1: Wartezeitanzeige mit geringer (links) und hoher Wartezeit (rechts), TODO Bildquelle

Der Nachteil daran ist, dass die Wartezeiten nur direkt beim Betreten der Attraktion sichtbar sind. Es wäre für den Besucher besser, diese auch andernorts und vergleichend betrachten zu können, um besser voraus planen zu können. Auch dafür haben einige Parks eine Lösung gefunden. So gibt es zum Beispiel Monitore mit allen Wartezeiten im Park oder eine parkeigene App mit allen aktuellen Wartezeiten.

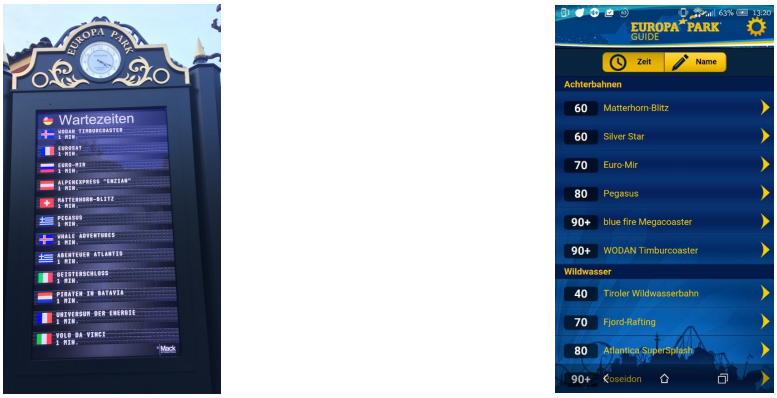

Abb. 2.2: Wartezeitübersicht des Europaparks als Monitor (links) und als App (rechts), TODO Bildquelle

Da diese parkeigenen Apps, aus verschiedenen Gründen, nur innerhalb des Parks funktionieren, bleibt auch hier das Problem, dass die aktuellen Wartezeiten nur innerhalb des Parks verfügbar sind und man nicht von außerhalb seinen Freizeitparkbesuch planen kann. Es kommt hinzu, dass man meistens nur eine aktuelle Wartezeit zur Verfügung hat, gerne aber weitere Informationen wie die Durchschnitte bestimmter Tage oder Uhrzeiten hätte. Außerdem gibt es bisher kein direktes Feedback der Nutzer über ihre tatsächlich verbrachte Wartezeit.

Um all diese Probleme zu behben gibt es in unserer App verschiedene Metriken zum Vergleichen der Wartezeiten verschiedener Attraktionen. Diese werden nicht von den Freizeitparks zur Verfügung gestellt, sondern von den Nutzern selbst eingetragen, um somit aus der Gesamtheit aller Nutzerdaten verschiedene Werte berechnen zu können. Speziell unterscheidet unsere App zwischen folgenden fünf Arten von Wartezeiten einer Attraktion:

- Aktuelle Wartezeit: der Durchschnitt der neusten eingetragen Wartezeiten einer Attraktion, denkbar wäre hier auch das Einbinden von Livedaten aus vorhandenen APIs bestimmter Freizeitparks

- Tagesdurchschnitt: der Durchschnitt aller eingetragenen Wartezeiten des aktuellen Tages einer Attraktion

- Gesamtdurchschnitt: der Durchschnitt aller eingetragenen Wartezeiten einer Attraktion
- Durchschnitt nach Uhrzeit: der Durchschnitt aller eingetragenen Wartezeiten einer Attraktion eines bestimmten Zeitraums
- Wartezeitenliste: das Anzeigen aller eingetragenen Wartezeiten, welche nach Datum oder Dauer sortiert werden können

Da nicht nur die Wartezeit sondern auch die Qualität einer Attraktion ausschlaggebend ist, gibt es in unserer App zusätzlich ein Bewertungssystem für die einzelnen Attraktionen und auch für den Park insgesamt. Dieses ist wie jedes klassische Bewertungssystem durch das "5-Sterne-Bewertungssystem"realisiert. Dabei entsprechen 5 Sterne einer hervorragenden und 0 Sterne einer mangelhaften Bewertung.

2.2 Mobile Plattform Android

Android ist ein mobiles Betriebssystem, also für Smartphones und Tablets, das von Google entwickelt wurde und auf Linux basiert. Die App-Entwicklung ist geprägt durch einzelne Aktivitäten (ein angezeigter Screen ist eine Aktivität), die miteinander kommunizieren und in ihrer 'Lebenszeit' ein vorgegebenes Zustandmodell 2.3 durchlaufen. Wir beschränken uns in Bezug auf die Plattform Android auf diese kurze Einführung, da die Plattform dank seiner weltweit hohen Verbreitung als bekannt angesehen werden dürfte.

Dieses Zustandmodell ist auch anfangs einer der Nachteile von Android, da es nicht so leicht zu verstehen ist und uns auch ein paar Probleme bereitet hat. Nachdem wir uns aber im Laufe der App-Entwicklung immer mehr mit Android vertraut gemacht haben, war auch das Modell kein Problem mehr, sondern eher ein Vorteil, da es sehr logisch und durchdacht ist. Eine weitere Schwierigkeit, die während der Entwicklung aufgetreten ist, ist, dass es so viele verschiedene Android-Versionen und Geräte gibt. Da wir unsere App für so viele Versionen wie möglichen entwickeln wollten, kamen deshalb auch mal das ein oder andere Problem auf, wie z. B. dass manche Libraries oder Frameworks erst ab bestimmten Versionen verfügbar sind oder die vielen verschiedenen Geräte alle unterschiedlichen Seiten- und Größenverhältnisse haben.

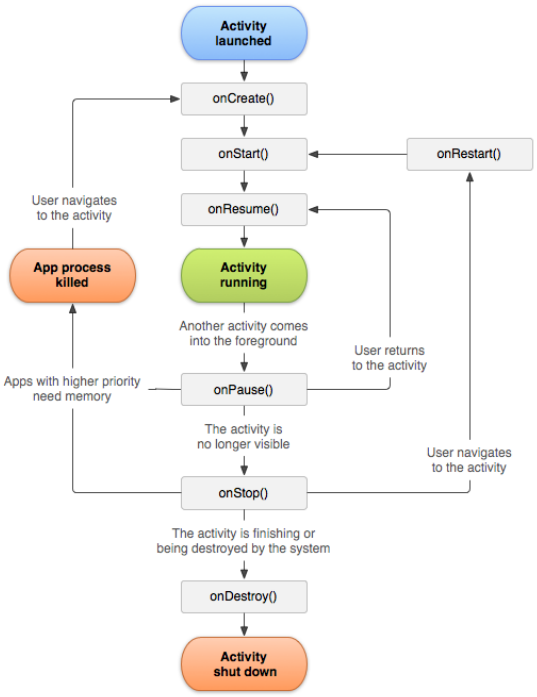

Abb. 2.3: Android Zustandsmodell, TODO Bildquelle

Die Vorteile von Android überwiegen aber auf jeden Fall, vor allem, wenn man sich damit tiefer beschäftigt und eingearbeitet hat. Einer der größten Vorteile ist die sehr gute Dokumentation von Android durch die das Einlesen in die Möglichkeiten und Funktionen recht leicht ist. Auch die weltweite Verbreitung und Beliebtheit von Android ist hier ein Vorteil, da es sehr viele Entwickler gibt und so jedes Problem schon eimal aufgetreten ist und daher auch zu den allermeisten auch Lösungen oder Workarounds bekannt sind. Außerdem wird Android stetig Weiterentwickelt, weshalb immer mehr möglich ist und der Umgang mit bestimmten Funktionen, wie z. B. der Zugriff auf Gerätefunktionen wie Kamera oder GPS immer leichter wird. Weitere Vorteile für uns sind die Vertrautheit mit Java und die Einfachheit von Android Studio, welches wir für die Entwicklung unserer App benutzt haben.

2.3 Frameworks

Frameworks sind eine Art Gerüst oder Rahmen, die eingesetzt werden um das Programmieren zu vereinfachen und die geschriebenen Zeilen zu verringern.
Wir haben in unserer App die folgenden aufgelisteten Frameworks als Hilfen genutzt. Alle sind öffentlich und über den jeweiligen Namen auffindbar.

- Microsoft Azure Mobile App Service: Server mit persistenter Speicherung in SQLite-Datenbank (näheres dazu in Kapitel 4.2.2)
- Google Firebase: einfache Nutzerverwaltung (näheres dazu in Kapitel 4.2.3)
- Cloudinary: kostenloses Hosten und Auslesen von Bildern via HTTP-Request
- Picasso: erlaubt einfaches Handling (z. B. Größentransformationen) von Bildern in oftmals einer Zeile
- Glide: ermöglicht eine bessere Darstellung der animierten Lade-Animation
- MPAndroidCharts: ermöglicht die Erstellung von Diagrammen (in unserem Fall Säulendiagramme für die Statistiken)
- MaterialChipsInput: Darstellung der Unterkategorien als Chips

3

Anforderungsanalyse

In diesem Kapitel werden die ursprünglich definierten funktionalen und nichtfunktionalen Anforderungen tabellarisch dargestellt. Jede der Anforderungen hat einen eindeutigen Identifikator (ID), einen Titel (TITEL), und eine Beschreibung (BES).

3.1 Funktionale Anforderungen

ID:	**FA1**
TITEL:	Startmenü
BES:	Nach dem Start der Anwendung sieht der Benutzer ein Startmenü mit den Einträgen „Spiel starten“, „Einstellungen“, „Spielregeln“, „Deckübersicht“ („Rangliste“), („Infos“)

ID:	**FA2**
TITEL:	Einstellungen
BES:	Es gibt eine Möglichkeit, die Spieleinstellungen zu ändern. Diese umfassen Schwierigkeitsgrad, Soundeffekte (an/aus) und Spielmodus (rundenbasiert/-zeitbasiert/Kartenbasiert)

ID:	**FA3**
TITEL:	Spielregeln
BES:	Es gibt eine Möglichkeit, die Spielregeln anzuzeigen.

ID:	**FA4**
TITEL:	Infoseite
BES:	Es gibt eine Möglichkeit, eine Infoseite (mit Copyright- und weiteren Informationen) anzuzeigen

ID:	**FA5**
TITEL:	Deckübersicht
BES:	Der Benutzer hat die Möglichkeit eine Liste mit allen verfügbaren Decks anzuzeigen. Beim Auswählen eines Decks kann der Benutzer durch die Karten des Decks scrollen. Dabei sieht er bei jeder Karte das Bild und die Attribute. Außerdem kann der Benutzer auf der Deckübersichts-Seite neue Decks hinzufügen. Diese Decks dienen als „Erweiterung“ und der Benutzer kann sie sich herunterladen.

ID:	**FA6**
TITEL:	Spiel starten
BES:	Der Benutzer hat vom Startmenü aus die Möglichkeit, das Spiel zu starten. Dafür öffnet sich ein Dialog, auf dem der Benutzer das Deck auswählt und festlegt, ob er gegen einen anderen Spieler oder gegen einen Computer spielen will. Danach werden die Karten auf die beiden Spieler verteilt und es wird zufällig bestimmt, welcher Spieler beginnt.

ID:	**FA7**
TITEL:	Spielablauf
BES:	Während des Spiels sieht der Spieler pro Runde nur seine „oberste Karte im Stapel“. In einer Runde werden die Werte des erstgewählten Attributs verglichen. Nach Auswahl eines Attributs vom ersten Spieler werden die zu vergleichenden Werte beider Spieler angezeigt und das gewinnende Attribut markiert. Bei einem Gleichstand werden die Karten beider Spieler unter den jeweils eigenen Stapel gelegt.

ID:	**FA8**
TITEL:	Spielstand anzeigen
BES:	Während des Spiels wird dauerhaft der Spielstand (Anzahl Karten Spieler 1 : Anzahl Karten Spieler 2) angezeigt.

ID:	**FA9**
TITEL:	Spielzeit anzeigen
BES:	Beim zeitbasierten Modus wird neben dem Spielstand auch die verbleibende Rundenzeit und die verbleibende Gesamt-Spielzeit angezeigt.

ID:	**FA10**
TITEL:	Spielende
BES:	Das Spiel ist vorbei, wenn • [alle Modi] ein Spieler alle Karten hat (Dieser Spieler hat gewonnen) • [Zeitbasiert] die Zeit abgelaufen ist (Der Spieler mit den meisten Karten hat gewonnen) • [Rundenbasiert] die vorher ausgewählte Anzahl an Runden gespielt worden sind (Spieler mit den meisten Karten hat gewonnen)

ID:	**FA11**
TITEL:	Speicherung der Daten
BES:	Die Daten des Spiels (Decks, Karten und ihre Attribute) werden in einer Datenbank gespeichert. Dabei gilt für jedes Deck: • Die Anzahl an Karten soll eine gerade Zahl zwischen 16 und 64 sein. • Die Anzahl an Attributen soll eine Zahl zwischen 5 und 10 sein. • Für jedes Attribut wird spezifiziert, ob ein höherer Wert oder ein niedriger Wert gewinnt.

3.2 Nichtfunktionale Anforderungen

ID:	**NFA1**
TITEL:	Entwicklungssprache und -Umgebung
BES:	Die Anwendung wird in Java mit der Entwicklungsumgebung „Android Studio“ entwickelt und soll auf Geräten mit Betriebssystemen ab Android [4.1] funktionieren.

ID:	**NFA2**
TITEL:	Reaktionszeit
BES:	Die Reaktionszeit der Anwendung soll zu jeder Zeit maximal 1,5 Sekunden betragen

ID:	**NFA3**
TITEL:	Persistenter Spielzustand
BES:	Während eines Spiels soll die App minimiert werden können (z.B. durch den Home Button), ohne dass der Spielzustand verloren geht. D.h., ein Spiel soll bei erneutem Öffnen fortgesetzt werden können.

ID:	**NFA4**
TITEL:	Benutzbarkeit
BES:	Die Anwendung soll intuitiv bedienbar sein. D.h., die Buttons und andere Auswahlmöglichkeiten sollen eine ausreichende Größe haben und wie erwartet reagieren.

ID:	**NFA5**
TITEL:	Offline-Modus
BES:	Die Anwendung soll stets auch das Spielen ohne Internetanbindung ermöglichen.

4

Konzept, Entwurf und Implementierung

4.1 Implementierungsdetails

TODO

Abb. 4.1: Hauptmenü der App

Abb. 4.2: Mockup Hauptmenü

4.2 Besonderheiten

TODO

4.2.1 Wartezeit berechung und -darstellung

4.2.2 Client-Server-Kommunikation mit Azure

4.2.3 User-Verwaltung mit Firebase

4.2.4 Parkverwaltung

4.3 Architektur

4.3.1 Datenmodell und -speicherung

TODO

4.3.2 Klassen- und Activity-Architektur

TODO

4.3.3 Arbeitsaufteilung

Während der Implementation wurden die Aufgaben so aufgeteilt, dass jeder aus dem Team immer für eine bestimmte Aufgabe zuständig war und diese durch alle Schichten durch erledigen musste. Bei großen Aufgaben, wie dem Spielablauf, wurden die Aufgabe zudem in kleinere Schritte aufgeteilt. Da die App aus mehreren unabhängigen Bereichen besteht, konnten die Aufgaben gut eingeteilt werden, ohne dass es zu Behinderungen kommt. Wenn gerade keine Arbeit für eine Person angefallen war, beschäftigte diese sich mit Verbesserungen, Testen von Funktionen oder mit der Weiterentwicklung des Designs.

4.4 Schwierigkeiten während der Implementierung

TODO

- 1
- 2

5

Anforderungsabgleich

5.1 Funktionale Anforderungen

TODO

5.2 Nicht Funktionale Anforderungen

TODO

6

Zusammenfassung und Ausblick

TODO

Abbildungsverzeichnis