



ulm university universität  
**ulm**

**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 26. August 2017

© 2017 Fabian Fischbach, Luis Beucamp 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-L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub>

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation und Problemstellung . . . . .	1
1.2 Zielsetzung . . . . .	2
1.3 Struktur der Arbeit . . . . .	2
<b>2 Grundlagen</b>	<b>5</b>
2.1 Grundlagen der Bewertungen und Wartezeitenberechnung . . . . .	5
2.2 Mobile Plattform Android . . . . .	7
2.3 Frameworks . . . . .	9
<b>3 Anforderungsanalyse</b>	<b>11</b>
3.1 Funktionale Anforderungen . . . . .	11
3.2 Nichtfunktionale Anforderungen . . . . .	13
<b>4 Konzept, Entwurf und Implementierung</b>	<b>15</b>
4.1 Implementierungsdetails . . . . .	15
4.2 Besonderheiten . . . . .	22
4.2.1 Wartezeit berechung und -darstellung . . . . .	22
4.2.2 Client-Server-Kommunikation mit Azure . . . . .	22
4.2.3 User-Verwaltung mit Firebase . . . . .	22
4.2.4 Parkverwaltung . . . . .	22
4.3 Architektur . . . . .	22
4.3.1 Datenmodell und -speicherung . . . . .	22
4.3.2 Klassen- und Activity-Architektur . . . . .	22
4.3.3 Arbeitsaufteilung . . . . .	22
4.4 Schwierigkeiten während der Implementierung . . . . .	23
<b>5 Anforderungsabgleich</b>	<b>25</b>
5.1 Funktionale Anforderungen . . . . .	25
5.2 Nicht Funktionale Anforderungen . . . . .	25

*Inhaltsverzeichnis*

<b>6 Ausblick</b>	<b>27</b>
<b>7 Fazit</b>	<b>29</b>

# 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 uns für eine Freizeitpark-App. Als zentrale Platform für Freizeitparkliebhaber sollten Parks mit Standorten, zugehörige Attraktionen mit Wartezeiten und Bewertungen verfügbar sein.

Eine große Anzahl potentieller Nutzer von Freizeitpark-Apps dürfte vorhanden sein. So hat zum Beispiel alleine der Europapark, Deutschlands größter Freizeitpark, jährlich 5,5 Millionen Besucher, Tendenz steigend<sup>1</sup>. Zusammen mit allen anderen Freizeitparks in Deutschland oder sogar weltweit kommt schnell eine beachtliche Summe von Freizeitparkbesuchern und damit potentiellen Nutzern zusammen, die wir mit unserer App ansprechen wollen. Umso erstaunlicher ist es, dass bisher nicht wirklich Apps oder Dienste für Freizeitparkbesucher zur Verfügung stehen.

Wenn man einmal die aktuelle Marktlage betrachtet fällt auf, dass es nur Apps für einzelne bestimmte große Parks, wie zum Beispiel Disneyworld, eigens entwickelte Apps

---

<sup>1</sup>Quelle:<http://presse.europapark.com/de/presse/nachricht/datum/2017/07/21/europa-park-ist-tourismusmagnet-im-schwarzwald/>

## *1 Einleitung*

oder von Fans entwickelte Apps 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. Wir konnten keine App für Android finden, welche im Ansatz der unserer Vorstellung ähnelte.

## **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, deren Attraktionen 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

### *1.3 Struktur der Arbeit*

Architektur gezeigt und wir erläutern Schwierigkeiten, die wir während der Implementierungsphase hatten. Abschließend gibt es einen Anforderungsabgleich und einen Ausblick auf die Zukunft des Projektes.

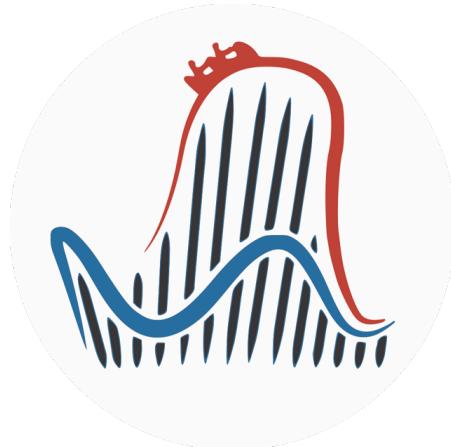


Abb. 1.1: Coaster2go Logo



# 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.

## 2 Grundlagen

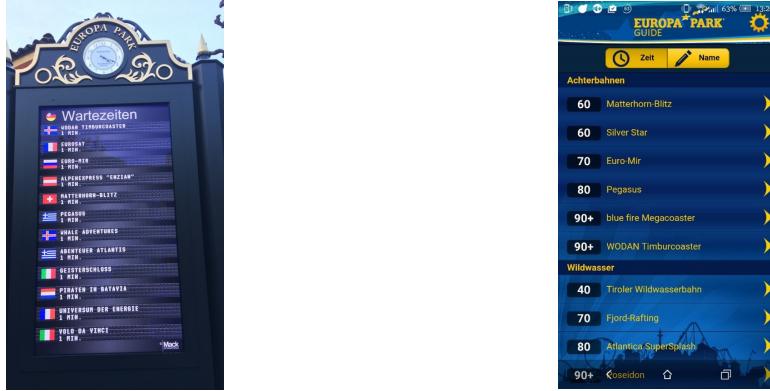


Abb. 2.2: Wartezeitübersicht des Europaparks als Monitor (links) und als App (rechts),  
TODO Bildquelle

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 eingetragenen 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

## 2.2 Mobile Plattform Android

- 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 möglich 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.

## 2 Grundlagen

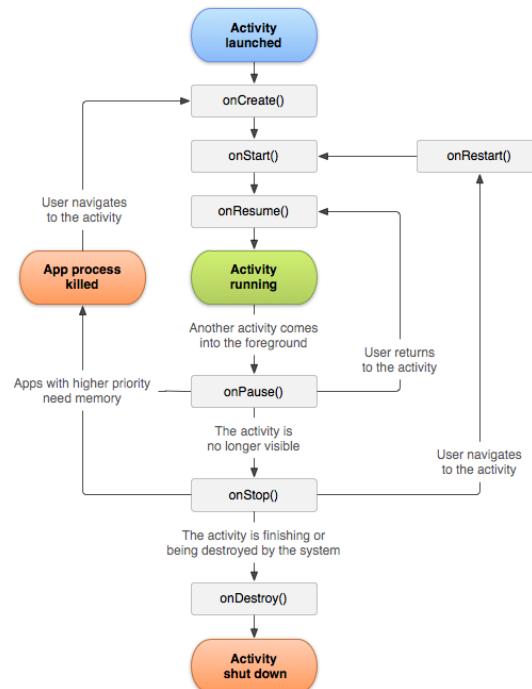


Abb. 2.3: Android Zustandsmodell, TODO Bildquelle

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 einmal 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

<b>ID:</b>	<b>FA1</b>
TITEL:	Startbildschirm
BES:	Nach dem Start der Anwendung sieht der Benutzer einen Startbildschirm, auf dem er sich anmelden bzw. registrieren kann, oder ohne Anmeldung zur Parkübersicht gelangt.
<b>ID:</b>	<b>FA2</b>
TITEL:	Parkübersicht
BES:	Diese Ansicht zeigt eine Liste aller Parks, die auf verschiedene Weisen sortiert werden kann. Außerdem gibt es eine getrennte Liste, in der sich die favorisierten Parks des Nutzers befinden. Sofern die Position des Smartphones bekannt ist, wird die Entfernung neben den Parks angezeigt und die Liste kann nach Entfernung sortiert werden.
<b>ID:</b>	<b>FA3</b>
TITEL:	Park Detailansicht
BES:	Für jeden Park gibt es eine Detailansicht mit näheren Informationen. Hier befinden sich Links/Buttons zu den Bewertungen, der Kartenansicht, sowie zur Attraktionsübersicht des Parks. Außerdem kann der Park zu den Favoriten hinzugefügt werden.

### 3 Anforderungsanalyse

<b>ID:</b>	<b>FA4</b>
<b>TITEL:</b>	Bewertungsseite (Park / Attraktionen)
<b>BES:</b>	Es gibt eine Ansicht der neuesten Bewertungen (nach Datum sortiert). Eine Bewertung hat jeweils Verfasser, Datum, Stern-Anzahl, und Kommentar. Falls der Benutzer eingeloggt ist, sieht er einen Button zum Hinzufügen einer Bewertung.
<b>ID:</b>	<b>FA5</b>
<b>TITEL:</b>	Bewertung verfassen
<b>BES:</b>	Sofern eingeloggt, kann der Benutzer in einem separaten Dialog eine Bewertung verfassen. Hat er zuvor schon eine Bewertung zu diesem Park / dieser Attraktion verfasst, kann er diese bearbeiten. Der Administrator kann den Kommentar ausblenden.
<b>ID:</b>	<b>FA6</b>
<b>TITEL:</b>	Attraktionsübersicht
<b>BES:</b>	Diese Ansicht zeigt eine Liste aller Attraktionen, die auf verschiedene Weisen sortiert und gefiltert werden kann. Außerdem gibt es eine getrennte Liste, in der sich die favorisierten Attraktionen des Nutzers befinden. Neben jeder Attraktion wird, sofern verfügbar, die jeweilige „aktuelle“ Wartezeit angezeigt. (Sofern die Position des Smartphones bekannt ist, wird die Entfernung neben den Parks angezeigt und die Liste kann nach Entfernung sortiert werden.)
<b>ID:</b>	<b>FA7</b>
<b>TITEL:</b>	Attraktion Detailansicht
<b>BES:</b>	Für jede Attraktion gibt es eine Detailansicht mit näheren Informationen. Hier befinden sich Links/Buttons zu den Bewertungen, der Kartenansicht, sowie zu den Wartezeiten. Auch kann die Attraktion zu den Favoriten hinzugefügt werden. Zudem gibt es verschiedene Statistiken zu den Wartezeiten (siehe FA8). Ist der Nutzer eingeloggt und befindet sich in der Nähe des Parks (GPS muss aktiviert sein), so kann er eine Wartezeit eintragen. Dies kann er nur ein Mal pro Stunde machen.

### 3.2 Nichtfunktionale Anforderungen

<b>ID:</b>	<b>FA8</b>
TITEL:	Wartezeit-Statistiken
BES:	<p>Es gibt vier verschiedene Statistiken über die Wartezeit bei einer Attraktion:</p> <ul style="list-style-type: none"> <li>• „Aktuell“: Zeigt die aktuelle Wartezeit an</li> <li>• „Heute“: Zeigt den Durchschnitt für den aktuellen Tag an</li> <li>• „Gesamt“: Zeigt den bisherigen Gesamtdurchschnitt an</li> <li>• Ein Säulendiagramm, welches den stündlichen bisherigen Durchschnitt anzeigt</li> </ul>
<b>ID:</b>	<b>FA9</b>
TITEL:	Wartezeiten-Verlauf
BES:	Es gibt eine Listenansicht der bisher eingetragenen Wartezeiten einer Attraktion. Ein Wartezeiten-Eintrag enthält Verfasser, Datum, Uhrzeit, und Wartezeit (in Minuten).
<b>ID:</b>	<b>FA10</b>
TITEL:	Admin-Funktionen
BES:	<p>Ein Administrator hat folgende Zusatzfunktionen:</p> <ul style="list-style-type: none"> <li>• Parks / Attraktionen hinzufügen und erstellte bearbeiten</li> <li>• Kommentare ausblenden</li> </ul>

## 3.2 Nichtfunktionale Anforderungen

<b>ID:</b>	<b>NFA1</b>
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.

### *3 Anforderungsanalyse*

<b>ID:</b>	<b>NFA2</b>
<b>TITEL:</b>	Reaktionszeit
<b>BES:</b>	Die Reaktionszeit der Anwendung soll zu jeder Zeit maximal 1,5 Sekunden betragen
<b>ID:</b>	<b>NFA3</b>
<b>TITEL:</b>	Benutzbarkeit
<b>BES:</b>	Die Anwendung soll intuitiv bedienbar sein. D.h., die Buttons und andere Auswahlmöglichkeiten sollen eine ausreichende Größe haben und wie erwartet reagieren.
<b>ID:</b>	<b>NFA4</b>
<b>TITEL:</b>	Offline-Modus
<b>BES:</b>	Die Anwendung soll alle Anfragen zwischenspeichern, um auch Offline Zugriff auf die wichtigen Informationen zu gewährleisten (Bilder werden von der Bild-Bibliothek gecached).

# 4

## Konzept, Entwurf und Implementierung

### 4.1 Implementierungsdetails

Wir erklären in diesem Kapitel die einzelnen Funktionen unserer Anwendung ausführlicher. Dies wird unterstützt durch Screenshots und den zugehörigen Mockups, die zu Beginn des Projekts erstellt wurden, um dem Kunden seine Vorstellungen visuell präsentieren zu können.

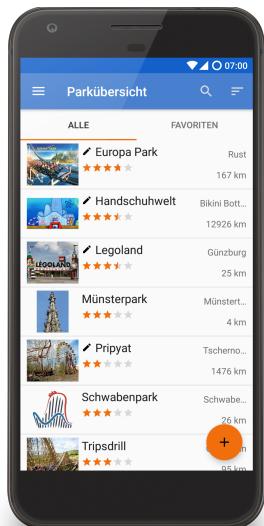


Abb. 4.1: Parkübersicht



Abb. 4.2: Mockup Parkübersicht

Nach dem Öffnen der Anwendung befindet man sich auf der Startseite unserer App. Diese ist auch zugleich die Parkübersicht, welche in Abbildung 4.1 zu sehen ist. Die Parkübersicht ist eine Liste aller Parks, die man durchsuchen und nach Alphabet, Bewertung oder Entfernung sortieren kann. Außerdem ist die Liste in zwei Tabs unterteilt.

#### 4 Konzept, Entwurf und Implementierung

Zum Einen das Tab 'Alle', in dem, wie der Name schon sagt, alle Parks angezeigt werden, und als zweites noch das Tab 'Favoriten' in dem alle Parks angezeigt werden, die der Nutzer sich als Favorit markiert hat.

TODO Bild von der grünen Menüleiste in der man sich einloggen kann

Ist ein Nutzer eingeloggt kann er Park und Attraktionen erstellen und bearbeiten, Wartezeiten eintragen und Bewertungen abgeben. Möchte man sich nicht einloggen kann man die App trotzdem als Informationsquelle nutzen, denn man kann alle Infos wie Wartezeiten und Bewertungen sehen, aber eben nicht selber eintragen.

Falls man als Nutzer, über den orangenen Plus-Button unten rechts in der Parkübersicht, einen Park erstellt hat, wird neben dem Namen des Parks das Piktogramm eines Stifts angezeigt, was bedeutet, dass man selber Admin des Parks ist und diesen und seine Attraktionen bearbeiten kann.

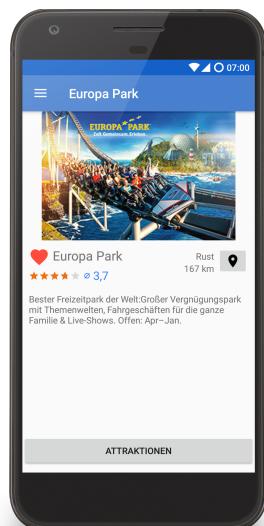


Abb. 4.3: Parkansicht



Abb. 4.4: Mockup Parkansicht

Wählt man nun, durch das Tippen auf einen Park, diesen aus, so kommt man als nächstes auf die Parkdetail Seite, welche in Abbildung 4.3 zu sehen ist.

Oben auf der Seite ist immer ein Bild des Parks zu sehen. Darunter befinden sich das Herz als Symbol für 'Favorit' das sich aktivieren und deaktivieren lässt, der Parkname,

#### 4.1 Implementierungsdetails

der Ort und falls GPS aktiviert ist auch die aktuelle Entfernung zum Park. Der Standort des Parks lässt sich außerdem durch den Button auf der rechten Seite auf GoogleMaps anzeigen.

Es befindet sich des Weiteren auch noch eine Anzeige der durchschnittlichen Bewertung (Skala von 0 bis 5) auf dieser Seite. Der Durchschnitt wird einmal als Sterne dargestellt und einmal als Dezimalzahl mit einer Nachkommastelle. Durch das Tippen auf die Sterne oder die Zahl lässt sich die Seite der Bewertungen öffnen, was aber weiter unten gezeigt wird.

Zusätzlich sind auch noch allgemeine Infos zum Park gegeben, wie z. B. die Öffnungszeiten.

Durch den am unteren Bildrand gelegenen Button 'Attraktionen' kommt man dann weiter zu den Attraktionen des Parks, die wieder, wie auch schon die Parks, als Liste gegeben sind.

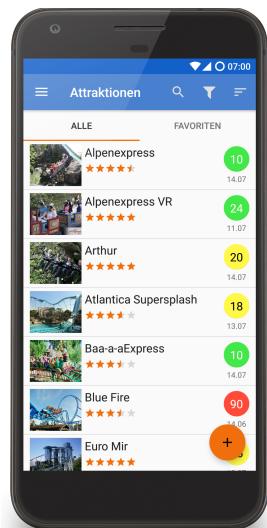


Abb. 4.5: Attraktionsübersicht



Abb. 4.6: Mockup Attraktionsübersicht

Die Seite aller Attraktionen eines Parks, die sog. Attraktionsübersicht, die in Abbildung 4.5 zu sehen ist, ist sehr ähnlich aufgebaut wie die Parkübersicht. Es gibt ebenfalls die Möglichkeiten zur Ansicht aller Attraktionen oder nur der Favoriten und zur Sortierung und Suche, aber hinzu kommt noch die Möglichkeit zum Filtern der Liste nach Kategorien der Attraktionen, wie z. B. Achterbahn oder Essensstand. Die einzelnen Elemente der

#### 4 Konzept, Entwurf und Implementierung

Liste bestehen aus Bild, Name und Bewertung und dazuhin noch der Wartezeit in Minuten und wann diese eingetragen wurde. Die Farben der Wartezeiten bedeuten einfach dass man entweder nur kurz anstehen muss (grün), durchschnittlich lang (gelb) oder dass momentan die Wartezeit sehr hoch ist (rot). Später wird aber noch genauer auf die Wartezeiten eingegangen und auch der Algorithmus dahinter erklärt.



Abb. 4.7: Attraktionsdetail

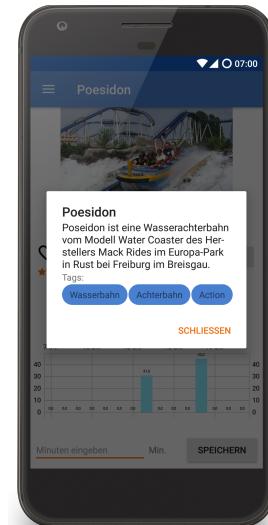


Abb. 4.8: Attraktions-Info

Wählt man nun eine Attraktion aus der Liste aus kommt man zur Detailansicht dieser Attraktion, wie in Abbildung 4.7 zu sehen.

Auf dieser Seite kann man sich, gleich wie bei Parks, die Attraktion als Favorit hinzufügen, den Standort anzeigen lassen und die Bewertungen ansehen. Neu hinzu kommt, dass man sich noch weitere Informationen über einen 'i'-Button anzeigen lassen kann, wo auch unter Anderem zu sehen ist, in welche der Kategorien die Attraktion gehört (siehe Abbildung 4.8).

Des Weiteren werden hier noch mehr Wartezeiten angezeigt, nämlich links der Durchschnitt der letzten drei Eintragungen, in der Mitte der Tagesdurchschnitt und auf der rechten Seite der Gesamtdurchschnitt. Darunter wird eine andere Wartezeitstatistik angezeigt. Hier sind auf der x-Achse die Stunden dargestellt und auf der y-Achse die Minuten. Die Säulen des Diagramms sind dann also die durchschnittliche Wartezeit in

## 4.1 Implementierungsdetails



Abb. 4.9: Mockup Attraktionsdetail

der angegebenen Stunde, die aus allen bisher eingetragenen Wartezeiten berechnet wird.

Ganz unten wird, falls der Nutzer eingeloggt ist, noch ein Feld zum selbst Wartezeiten eintragen angezeigt. Das Eintragen ist aber nur möglich, falls sich der Nutzer zusätzlich noch im Park befindet (Standort muss < 2km vom Park entfernt sein) und nicht schon innerhalb der letzten Stunde für diese Attraktion eine Wartezeit eingetragen hat.

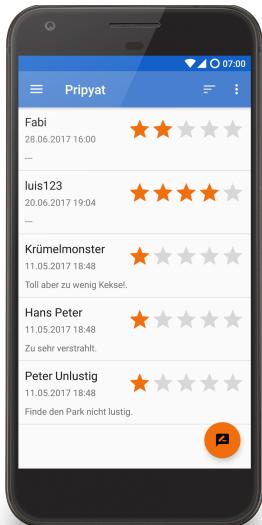


Abb. 4.10: Bewertungen



Abb. 4.11: Mockup Bewertungen

#### 4 Konzept, Entwurf und Implementierung

Will man nun die Bewertungen einer Attraktion bzw. eines Parks sehen oder diese auch selbst bewerten, kommt man auf die in Abbildung 4.10 zu sehende Activity.

Ganz oben wird noch einmal der Durchschnitt aller Bewertungen angezeigt. Darunter sind alle bisherigen Bewertungen mit Nutzernamen, Datum und Kommentar zu sehen.

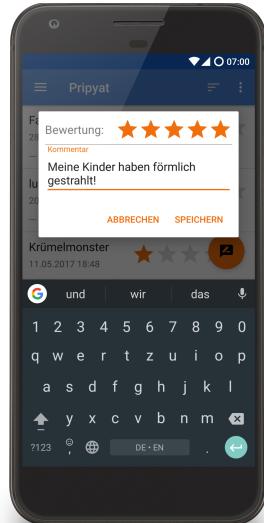


Abb. 4.12: Bewertung verfasssen



Abb. 4.13: Mockup Bewertung verfassen

Um selbst eine Bewertung abzugeben tippt man auf den orangenen Plus-Button unten Rechts und es öffnet sich ein Eingabedialog, wie in Abbildung 4.12 zu sehen. Falls man zu diesem Park bzw. dieser Attraktion eine Bewertung abgegeben hat, so wird diese hier angezeigt und man kann sie bearbeiten, aber es wird keine neue erstellt. Hat man bisher noch nicht bewertet, vergibt man jetzt 0 bis 5 Sterne, schreibt ein Kommentar und speichert dann mit 'OK'.

TODO wartezeiten text

TODO editor text

Screenshot Todos:

- Mapview

- Menu links

## 4.1 Implementierungsdetails

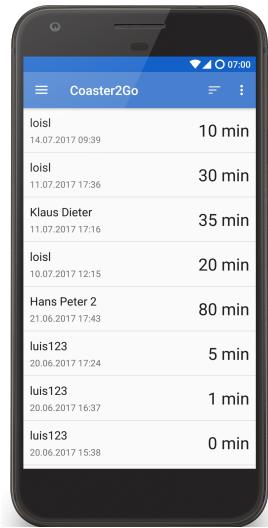


Abb. 4.14: Wartezeiten



Abb. 4.15: Mockup Wartezeiten



Abb. 4.16: Editor (Attraktion)

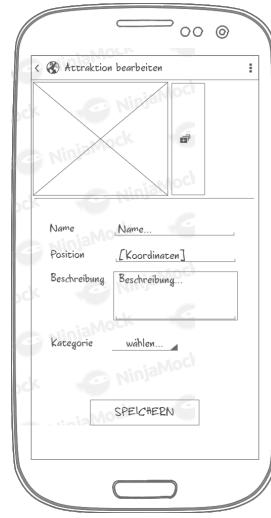


Abb. 4.17: Mockup Editor

## *4 Konzept, Entwurf und Implementierung*

### **4.2 Besonderheiten**

TODO

#### **4.2.1 Wartezeit berechnung 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*

### **4.4 Schwierigkeiten während der Implementierung**

TODO

- 1
- 2



# 5

## Anforderungsabgleich

### 5.1 Funktionale Anforderungen

TODO

### 5.2 Nicht Funktionale Anforderungen

TODO



# 6

## Ausblick

Wie im Kapitel 5 zu sehen ist, haben wir all unsere inhaltlichen Anforderungen an die App erfüllt. Deswegen haben wir uns in diesem Kapitel einige Gedanken gemacht, welche Funktionen für eine zukünftige Erweiterung der App denkbar wären. Diese könnten die bisherigen Funktionen sinnvoll ergänzen.

- Wartezeitenverifizierung: Bisher gibt es in unserer App keine Möglichkeit, falsch eingetragene Wartezeiten zu melden oder rückgängig zu machen. Zwar gleichen sich einzelne falsch eingetragene Wartezeiten dank der verwendeten Metriken zur Berechnung der Durchschnittswartezeiten schnell aus und haben keinen großen Einfluss. Trotzdem wäre es eine gute Ergänzung, wenn andere Nutzer die zuletzt eingetragenen Wartezeiten bestätigen oder ablehnen könnten, sodass die jeweilige Wartezeit an Gewichtung zu- oder abnimmt.
- Ansteh-Timer: Um die Wartezeit nicht jedes mal selbst messen oder ablesen zu müssen, wäre ein Ansteh-Timer eine gute Ergänzung für die Benutzer der App. Dazu wird beim Beginn des Anstehens ein Startknopf gedrückt, welcher den Timer startet. Kurz vor der Fahrt oder direkt nach der Fahrt kann dann der Stopknopf gedrückt werden. Dadurch wird die gewartete Zeit automatisch ermittelt und hochgehalten.
- Bewertungskategorisierung: Bisher gibt es bei unseren Bewertungen für die Parks und Attraktionen jeweils nur die Möglichkeit, allgemein zwischen null und fünf Sternen zu verteilen. Noch besser wäre, für jeden Park oder jede Attraktion bestimmte Kriterien der Kategorisierung vorzunehmen. So könnten zum Beispiel Achterbahnen nach der Action, Wasserbahnen nach ihrer Erfrischung, Familienbahnen nach

## *6 Ausblick*

der Tauglichkeit für Kinder oder Restaurants nach dem Geschmack bewertet werden. Dies bietet sich besonders bei Attraktionen an, da sie zum einen bereits in verschiedene Kategorien aufgeteilt sind. Zum anderen laufen die Benutzer so nicht in Gefahr, zum Beispiel eine Kinderbahn niedriger zu bewerten, weil sie nicht genügend Action für sie bereit hält.

- Tourplaner: Da die Attraktionen schon in einer Mapview angezeigt werden, wäre es eine sinnvolle Ergänzung, den Benutzern die Möglichkeit zu geben, sich selbst eine Tour zusammen zu stellen, welche sie dann auf der Karte angezeigt bekommen. Des Weiteren könnten auch bereits vorhandene Touren geladen werden oder Touren automatisch erstellt werden, zum Beispiel nach Kategorien oder nach Wartezeiten.

Zudem besteht zumindest theoretisch die Möglichkeit, Freizeitparks und ihre Daten direkt mit in die App einzubinden. Dadurch ergibt sich auch die Möglichkeit, wie sich die App finanzieren könnte. Freizeitparks, die eine Kooperation eingehen, könnten Vorteile oder Premiumaccounts erhalten. Mögliche Vorteile wären neben dem selbstständigen Verwalten der eigenen Parkdaten zum Beispiel das Einbinden ihrer eigenen Live-Wartezeiten aus bereits vorhandenen parkeigenen APIs, das Schalten von Werbung oder Aktionen des Freizeitparks oder auch das Hervorheben des Freizeitparks durch ein besonderes Design oder eine gut sichtbare Platzierung in der Übersichtsliste.

# 7

## Fazit

In diesem Kapitel soll ein kurzes Fazit zur App-Entwicklung und zum Anwendungsfach allgemein festgehalten werden.

Dadurch, dass wir als Team schon im Semester zuvor die Quartett-App zusammen entwickelt haben, hatten wir schon genügend Vorwissen, um in die Entwicklung unserer Freizeitpark-App einzusteigen. Trotzdem haben wir auch während der Entwicklung dieser App viel Neues gelernt was wir für die Zukunft und für die Entwicklung weiterer Apps gebrauchen können, wie zum Beispiel das Arbeiten mit Standorten und das Einbinden von vorhandenen Frameworks.

In unseren Augen ist uns die Coaster2Go App größtenteils gut gelungen und das Endergebnis ist genau so, wie wir es uns am Anfang des Semesters vorgestellt haben. Die App wäre theoretisch schon ausreichend für eine Veröffentlichung im App Store, da es bisher keine vergleichbare App im Google Play Store gibt. Natürlich gäbe es noch einige Stellen, die eine Anpassung benötigen, bevor es wirklich zu einer Veröffentlichung kommen kann. Zum Beispiel müsste die Speicherung der Daten auf einen anderen Server verlegt werden, da wir bisher mit unseren Studentenaccounts nur begrenzte Kapazitäten besitzen. Auch kleinere gestalttechnische oder funktionale Verbesserungen sind denkbar.

Auch was das Arbeiten im Team und die Aufteilung der Aufgaben angeht, konnten wir im Vergleich zum ersten Teil des Anwendungsfaches noch Neues dazu lernen. Die Teamarbeit hat immer gut funktioniert und es kam nie zu größeren Komplikationen. Wir werden es auf jeden Fall in Erwägung ziehen, möglicherweise im Masterstudium das Anwendungsfach fortzusetzen.



# Abbildungsverzeichnis

1.1	Coaster2go Logo . . . . .	3
2.1	Wartezeitanzeige im Park . . . . .	5
2.2	Wartezeitübersicht im Europapark . . . . .	6
2.3	Android Zustandsmodell . . . . .	8
4.1	Parkübersicht . . . . .	15
4.2	Mockup Parkübersicht . . . . .	15
4.3	Parkansicht . . . . .	16
4.4	Mockup Parkansicht . . . . .	16
4.5	Attraktionsübersicht . . . . .	17
4.6	Mockup Attraktionsübersicht . . . . .	17
4.7	Attraktionsdetail . . . . .	18
4.8	Attraktions-Info . . . . .	18
4.9	Mockup Attraktionsdetail . . . . .	19
4.10	Bewertungen . . . . .	19
4.11	Mockup Bewertungen . . . . .	19
4.12	Bewertung verfasssen . . . . .	20
4.13	Mockup Bewertung verfassen . . . . .	20
4.14	Wartezeiten . . . . .	21
4.15	Mockup Wartezeiten . . . . .	21
4.16	Editor (Attraktion) . . . . .	21
4.17	Mockup Editor . . . . .	21