Traildevils App

Bachelorarbeit
Frühjahrssemester 2016

Abteilung Informatik
Hochschule für Technik Rapperswil

Autoren
Mirco Strässle, Robert Vogt

Betreuer
Prof. Dr. Markus Stolze

Projektpartner
Frontline Media GmbH

Experte
Thomas Kälin

Gegenleser
Prof. Stefan F. Keller

Abstract

Die Frontline Media GmbH betreibt die Mountainbike Community traildevils.ch. In ihrem Auftrag untersucht die vorliegende Arbeit die Notwendigkeit und Vorteile einer mobilen App, wenn bereits eine mobilfähige Webseite vorhanden ist. In einem Prototyp soll gezeigt werden, dass eine dedizierte App als Ergänzung zur Traildevils-Webseite nicht nur theoretische, sondern auch praktische Vorteile mit sich bringt.

Webtechnologien werden auch in mobilen Browsern immer umfangreicher und bieten laufend mehr Möglichkeiten. Native Apps sind jedoch häufig reaktionsschneller. Auch die Möglichkeiten der Offlinehaltung von Daten sowie die Nutzererreichbarkeit durch Push Benachrichtigungen sind denen eines Browsers klar überlegen. Diese Vorteile sind als Teil eines Kriterienkatalogs für eine Evaluation von Cross-Plattform Technologien verwendet worden.

In der Evaluation zeigte sich Facebooks React Native als geeignete Technologie für diese App. Daher wurde auf Basis von React Native ein Prototyp der Traildevils App umgesetzt, der die essentiellen Szenarien und Use-Cases der Plattform abbildet. Die Cross-Plattformfähigkeit ist für iOS und Android bereits im Prototyp gewährleistet. Als Architektur wird mit Redux eine Art von Flux verwendet. Flux gibt den Datenfluss in eine Richtung vor und stellt so eine interessante Alternative zu den bekannten MV*-Architekturen dar. Diese Architektur erlaubt es, eine solide Basis für zukünftige Weiterentwicklungen bereit zu stellen. Konzeptionell wird gezeigt, dass das Zusammengehörigkeitsgefühl der Community durch Push Benachrichtigungen gestärkt und die Aktivität erhöht werden kann. Im Prototyp wird gezeigt, dass mit einem flüssigen UI und einer klaren Nutzerführung die Nutzeraktivität gegenüber einer mobilen Webseite erhöht werden kann. Die Reduzierung der Informationen auf ein Minimum führt dazu, dass sich Nutzer gerade unterwegs einen sehr schnellen Überblick verschaffen können. Der Prototyp und die erarbeiteten Resultate sind zum Schluss in einem Usability Test validiert worden.

Management Summary

Ausgangslage

Die Firma Frontline Media GmbH unterhält die Webseite traildevils.ch und entwickelt diese weiter. Traildevils steht für die grösste Online-Community von Bikern in der Schweiz und ist dort auch das meistbesuchte Mountainbike-Portal. Biker tauschen sich täglich über den Zustand von Trails, Meinungen zu neuen Mountainbikes und über den Bikesport im Allgemeinen aus.

Traildevils ist aktuell nur über einen Webbrowser nutzbar. Entwickelt wurde die Website primär für Desktop Computer, inzwischen sind aber einzelne Teile der Seite optimiert für mobile Endgeräte. Durch Statistiken und Tools wie Google Analytics ist aber erkennbar, dass die Nutzer primär über Desktop Computer zugreifen.

Um die mobile Nutzung von Traildevils zu stärken, sucht die Frontline Media GmbH nach Möglichkeiten, das mobile Erlebnis für die Nutzer zu verbessern. Aus diesem Grund sollen die Notwendigkeit und die Vorteile einer mobilen App analysiert und in einer prototypischen Umsetzung demonstriert werden.

Vorgehen

Zur Durchführung des Projektes wurden agile Methoden der Softwareentwicklung angewendet. Initialisiert wurde das Projekt mit einem Ideen Brainstorming um zusammen mit dem Auftraggeber die allgemeine Stossrichtung zu definieren. In wöchentlichen Meetings wurden die weiteren Aufgaben priorisiert und in planbare Tasks aufgeteilt.

Ausgewählte Ideen daraus sind anschliessend in der Form von User Journeys konkretisiert worden. Aufgrund der daraus gewonnenen Erkenntnisse, wurde eine für das Vorhaben geeignete Cross-Plattform Technologie evaluiert sowie ein UI & UX Konzept für die App erarbeitet. Dieses Konzept wurde mit Hilfe eines Papierprototyps in einem Experteninterview verifiziert.

Im Anschluss daran wurde die App iterativ in 2-wöchigen Sprints implementiert. Abschliessend wurde der erarbeitete Prototyp in einem Usability Test und die Code Basis in einem Code Review mit unbeteiligten Personen geprüft.

Technologie

Der App Prototyp wurde mit React Native für iOS und Android umgesetzt. Es handelt sich dabei um eine Technologie, die von Facebook entwickelt wurde, um die Vorteile der nativen und der Web Entwicklung zu verbinden. Code für React Native wird dabei in JavaScript, JSX und einem Subset von CSS geschrieben.

Die Architektur besteht aus dem UI, welches nach dem Atomic Design Prinzip organisiert wurde und einem Domain-Layer, welcher nach dem Domain-driven Design Prinzip aufgebaut ist. Der Datenfluss zwischen diesen Layern ist nach dem Flux Prinzip gestaltet und mit Hilfe der Redux Bibliothek implementiert.

Das zentrale Element der App ist eine Kartenansicht, die von Mapbox zur Verfügung gestellt wird. Mapbox verwendet für das Kartenmaterial teilweise Open Street Map Daten. Die React Native Komponente für Mapbox basiert auf den nativen Mapbox SDKs für iOS und Android.

Ergebnisse

Mit dem App Prototypen konnte eine Basis für die zukünftige Entwicklung von Traildevils im mobilen Bereich gelegt werde. Dabei konnte auch gezeigt werden, dass eine App ergänzend zur mobile Webseite Mehrwerte bieten kann.

Funktionell deckt der Prototyp die wichtigsten User Journeys rund um die Objekte Trails, Shops und Destinationen ab. Sie werden alle auf einer Karte angezeigt, können über eine Suchfunktion gefunden werden und Detailinformationen sind für den Nutzer ersichtlich. Für Trails ist zudem die Möglichkeit gegeben, dass Nutzer eigene Bewertungen und Zustandsmeldungen abgeben können. Bei Shops sind Bewertungen ebenfalls vorhanden.

Im Usability Test wurde festgestellt, dass auch das Ziel zur Steigerung der Nutzeraktivität mit der App erreicht werden dürfte. Es ist für Nutzer sehr einfach und schnell möglich

Trails und Shops zu finden und diese mit Inhalten wie Zustandsmeldungen oder Bewertungen anzureichern.

Offline Funktionalität für die Karten und Traildevils Daten wurde im Prototypen nicht implementiert, jedoch wurde dafür eine Basis gelegt. Der Suchverlauf wird bereits jetzt offline gespeichert. Weitere Traildevils Daten werden aktuell im Arbeitsspeicher als Cache abgelegt, was zu einer vollen Offlinespeicherung ausgebaut werden kann. Auch die Mapbox Implementierung bietet die Möglichkeit für die zukünftige Offlinespeicherung von Kartenabschnitten.

Ausblick

Nebst dem umgesetzten Prototyp wurden auch verschiedene Konzepte zu personalisierten und geobasierten Inhalten erarbeitet, die als Grundlage für zukünftige Entwicklungen dienen können.

Es wurde gezeigt, dass es für Push Benachrichtigungen sowie offline verfügbare Daten viele Anwendungen gibt. Ebenfalls könnte mit der angedachten Strava-Integration die Attraktivität weiter gesteigert werden.

Im Verlaufe der Entwicklung mussten einige Features aufgrund mangelnder technischer Unterstützung aufgeschoben werden. Dazu gehören vor allem Features in Bezug auf Mapbox wie Clustering, offline Karten sowie eingezeichnete GPS-Tracks. Die Wichtigkeit dieser Features wurde im Usability Test bestätigt und sie sollten, sobald die technischen Hindernisse beseitigt sind, umgesetzt werden. Aktuelle Informationen deuten darauf hin, dass hier Fortschritte innerhalb eines Jahres zu erwarten sind.

Erklärung der Eigenständigkeit

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde.
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.
- dass wir keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt haben.

Rapperswil-Jona, 17. Juni 2016

Robert Vogt

Mirco Strässle

Aufgabenstellung

Aufgabenstellung Bachelorarbeit Abteilung I, FS 2016 Mirco Straessle, Robert Vogt

Traildevils App

1. Betreuer & Praxispartner

Betreuer dieser Arbeit ist
Prof. Dr. Markus Stolze mstolze@hsr.ch

Co-Referent für diese Arbeit ist
Stefan Keller

Externer Experte für diese Arbeit ist
Thomas Kälin

Praxispartner für dieser Arbeit ist
Mischa Trecco
Frontline Media GmbH
Rosenbergstrasse 9
8630 Rüti

2. Ausgangslage

Ein zentrales Produkt der Frontline Media GmbH ist der Traildevils-Website, eine Online Community Site für Mountain-Bikers. Der Traildevils-Website ist schon heute mit dem Webbrowser von Mobilgeräten aus zugreifbar, aber der Auftraggeber meint, dass sich mit einer App die Attraktivität des Angebots an Mountain-Biker noch verbessern lässt und damit die Attraktivität der Community gesteigert werden kann.

3. Ziele der Arbeit

In Absprache mit dem Praxispartner sollen Anforderungen für eine Cross-Platform App erhoben werden welche für Nutzer Mehrwert gegenüber dem aktuell auch mobil zugreifbaren Website aufweist. Die Machbarkeit einer solchen App und wichtige Eigenschaften wie Flüssigkeit der Bedienung sollen anhand einer prototypischen Umsetzung demonstriert werden. Die Aufgabe beinhaltet auch die Unterstützung des Praxispartners bei der Wahl einer angemessenen Umsetzungstechnologie.

4. Dokumentation

Für den Praxispartner ist eine angemessene Dokumentation der Arbeitsresultate zu verfassen. Umfang und Form dieser Dokumentation ist im Rahmen der Anforderungsanalyse festzulegen. Folgende Elemente sollten enthalten sein: Dokumentation zur Einrichtung der Entwicklungsumgebung, Kompilierung und Testen der App sowie Instruktion zum Publizieren der App, so dass Übernahme und Weiterentwicklung des Projekts einfach möglich ist.

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die Dokumentation ist vollständig auf CD/DVD in einem Exemplar abzugeben (Exemplar für das Sekretariat Informatik) Sowie ein Download-Link für Prof. Stolze und weitere Exemplare nach Absprache mit dem Co-Referenten und dem Experten

Zudem ist eine kurze Projektresultatdokumentation im öffentlichen Wiki von Prof. M. Stolze zu erstellen.

5. Weitere Regeln und Termine

Im Weiteren gelten die allgemeinen Regeln zu Bachelor und Studienarbeiten
„Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik" (HSR Intranet)
https://www.hsr.ch/Ablaeufe-und-Regelungen-Studie.7479.0.html)

Der Terminplan ist hier ersichtlich (HSR Intranet)
https://www.hsr.ch/Termine-Bachelor-und-Studiena.5142.0.html

6. Rechte

Die resultierende Software und Dokumentation darf von den Studenten, vom Auftraggeber und der HSR genutzt und erweitert werden. Eine Veräusserung an Dritte erfordert die Zustimmung des Praxispartners. Der Source-Code darf ohne die Einwilligung der Studierenden und des Praxispartners nicht veröffentlicht werden. Eine Veröffentlichung in Ausschnitten (z.B. Blogposts und öffentliche Dokumentation der Bachelorarbeit) durch die Studierenden ist erlaubt. Es wird durch alle Parteien sicher gestellt, im Source-Code und im Impressum der Anwendung die originale Urheberschaft durch die HSR Studenten weiterhin sichtbar bleibt.
Der Bericht der Bachelorarbeit (ohne geheime Anhänge) wird von der HSR im E-Prints Respository der HSR (eprints.hsr.ch) elektronisch veröffentlich. Titel, Abstract und Praxispartner der Arbeit dürfen von der HSR und Studierenden schon während der Arbeit kommuniziert werden.

7. Beurteilung

Eine erfolgreiche Bachelorarbeit zählt 12 ECTS-Punkte pro Studierenden. Für 1 ECTS Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert. Entsprechend sollten ca. 350h Arbeit für die Bachelorarbeit aufgewendet werden. Dies entspricht ungefähr 25h pro Woche (auf 14 Wochen) und damit ca. 3 Tage Arbeit pro Woche pro Student.

Für die Beurteilung ist der HSR-Betreuer verantwortlich. Die Bewertung der Arbeit erfolgt entsprechend der verteilten Kriterienliste.

Die definitive Aufgabenstellung wurde am 5.4.2016 beschlossen.

Rapperswil, 5.4.2016
Prof. Dr. Markus Stolze, Institut für Software, Hochschule für Technik Rapperswil

Inhaltsverzeichnis

1 Ausgangslage

1.1 Vision

Mit über 28'000 Mitgliedern und über einer Million Seitenaufrufe pro Monat ist traildevils.ch das meistbesuchte Mountainbike-Portal der Schweiz [1]. Biker tauschen sich täglich über den Zustand von Trails, Meinungen zu neuen Mountainbikes und über den Bikesport im Allgemeinen aus. Das Portal wird von der Frontline Media GmbH betrieben und laufend weiterentwickelt.

Die Vision der Traildevils App ist die Nutzeraktivität auf der Plattform zu erhöhen. Das verfügbare Angebot soll allerdings nicht eins-zu-eins abgebildet oder ersetzt, sondern durch zusätzliche Funktionalität ergänzt werden. Damit eine möglichst breite Zielgruppe bedient werden kann, ist eine Veröffentlichung auf den führenden Plattformen iOS und Android sehr wichtig. Das soll mit kleinem Aufwand möglich sein.

Traildevils.ch ist aktuell nur über einen Webbrowser bedienbar. Entwickelt wurde die Website primär für Desktop Computer, inzwischen sind aber einzelne Teile der Seite für mobile Endgeräte optimiert. Durch Statistiken und Tools wie Google Analytics ist aber erkennbar, dass die Nutzer primär über Desktop Computer zugreifen. Die mobile Plattform soll gefördert werden, sodass Inhalte bereits unterwegs erfasst respektive bewertet werden können und sich so die Nutzeraktivität erhöht.

Verstärkt sollen einem Nutzer personalisierte und geobezogene Daten nahegebracht werden. Zum Beispiel können gezielt Daten und Dienstleistungen von Partnerangeboten präsentiert werden, wenn er sich bereits für deren Inhalte interessiert. So kann ebenfalls die Interaktion zwischen Nutzer und Partner (meist Tourismus-Destinationen) erhöht werden.

Dem Nutzer wird der Mehrwert geboten, sich in einer übersichtlichen Art und Weise einen Überblick über die nahegelegenen Mountainbike-Angebote und Aktivitäten zu verschaffen. Der Nutzer kann auf einer Kartenansicht sehen, welche Trails, Touren, Events und Mountainbike-Shops in seiner direkten Umgebung sind.

Im Gegenzug kann Traildevils aufgrund der Lokalisierung gezielt den Nutzer zur Aktivität auffordern. Ein Nutzer kann dazu aufgefordert werde einen Trail zu bewerten, welchen er gerade eben befahren hat. Eine stärkere Einbindung und die Steigerung der Interaktion des Nutzers bedeuten auch, dass theoretisch das Nutzerprofil geschärft werden kann (welche Trails gefallen ihm, welche Destinationen besucht er oft, etc.) und somit können auch Inhalte personalisiert werden.

Für ein möglichst optimales Angebot auf der Plattform arbeitet Traildevils intensiv mit diversen Mountainbike- respektive Tourismusdestinationen zusammen. Schaut sich ein Nutzer gerade die Angebote in der Region Lenzerheide an, kann er dazu aufgefordert werden, eine Umfrage der Lenzerheide auszufüllen oder die Lenzerheide als Destination zu bewerten. Für die Destination ist eine gute Bewertung der eigenen Trails und Angebote auf einer prominenten Bike-Plattform wie Traildevils natürlich wünschenswert.

1.2 Ziele

In der Praxis stellt sich oft die Frage, ob eine mobile App, eine mobile-fähige Website oder eine Kombination entwickelt werden soll. In dieser Arbeit werden die Vor- und Nachteile von mobilen Apps gegenüber Webseiten untersucht und dokumentiert.

Die Technologie für die App des Praxispartners soll Cross-Plattform-fähig sein. Dabei gibt es verschiedene Ansätze, welche ebenfalls untersucht und miteinander verglichen werden sollen.

Cross-Plattform Apps bringen nicht nur technische respektive technologische Herausforderungen mit sich. So sind auch ein gutes User Interface Design oder User Experience Patterns nicht zu unterschätzen. Verschiedene Ansätze sollen untersucht und bewertet werden.

Diese beiden Fragen werden nicht nur theoretisch, sondern auch anhand eines Prototypen untersucht. Als primäre Zielplattform für die Umsetzung dieses Prototypen gilt iOS. Die Umsetzung für Android ist ein sekundäres Ziel. Es soll allerdings gezeigt werden, dass diese zusätzliche Plattform mit geringem zusätzlichen Aufwand unterstützt werden kann.

1.3 Konkurrenzlösungen

Da Traildevils zurzeit nur auf dem Schweizer Markt aktiv ist, ist es schwierig direkte Konkurrenten zu eruieren. Analysiert wurden die Lösungen von drei Konkurrenten. Die erste App, Graubünden Mountainbike, ist eine Schweizer Lösung mit anderen Zielen als Traildevils. Die beiden anderen Lösungen, Pinkbike und Trailforks, sind ausländische Apps mit ähnlichen Zielen, haben dafür nur wenige Daten für die Schweiz.

1.3.1 Graubünden Mountainbike

Die Graubünden Mountainbike App stammt von Graubünden Tourismus, wurde aber von Outdooractive GmbH & Co. KG entwickelt. Outdooractive stellt Angebote für Outdoor-Aktivitäten auf der eigenen Plattform oder via Schnittstelle anderen Anbietern zur Verfügung. Verfügbar ist die App für die Plattformen iOS und Android. Die Schwestern-App Graubünden Wandern stellt ähnliche Inhalte bereit und ist ebenfalls auf beiden Plattformen verfügbar.

Features

Touren	Touren sind in einer Listenansicht dargestellt, die sich nach Entfernung, Qualität und Strecke sortieren lässt. Einzelne Touren lassen sich offline speichern.
Karte	Die Karte kann in den Gelände-, Satelliten- und Kartenansicht von Google Maps betrachtet werden. Zusätzlich steht Kartenmaterial von Outdooractive, SwissTopo und OpenStreetMap zur Verfügung. Eine »near me«-Funcktion hat die App nicht.
Tourenplaner	Der Tourenplaner erlaubt das Setzen von Wegpunkten und berechnet dann eine mögliche Strecke.
Kategorien	Kategorien sind zusätzliche Daten wie Freizeitangebote, Unterkünfte oder Sehenswürdigkeiten. Diese werden ebenfalls in einer Listenansicht dargestellt.

Höhenmeter-Sammler	Dieses Feature soll die App von der Konkurrenz abheben. Per Klick werden die zurückgelegten Höhenmeter (positiv und negativ) getrackt und übermittelt. Mit der Verwendung des Features (und einer gewissen Anzahl zurückgelegten Höhenmetern) nimmt man automatisch an einem Wettbewerb teil.
Wetter	Mit Hilfe einer WebView wird ein Wetterbericht eingebunden. Die eingebundene Seite wird von Graubünden Tourismus betrieben, die Wetterdaten stammen von SRF Meteo.

Tabelle 1: Feature Übersicht Graubünden Mountainbike

Fazit

Im Vergleich zur Konkurrenz sind die Daten der Graubünden Mountainbike App sehr hochwertig. Genau hier liegt aber auch die Limitierung der App: sie ist auf den Raum Graubünden beschränkt. Aktivitäten ausserhalb des Handlungsbereichs von Graubünden Tourismus sind nicht verfügbar.

Die App ist schlicht gehalten. Das Design hält sich weder an die Standards der jeweiligen Plattform, noch wurde die Designwelt von Graubünden Tourismus wirklich übernommen. Dabei entsteht eine sonderbare Mischung von nativen und individuellen UI Elementen.

Die App positioniert sich vor allem im Bereich der Tourenplanung, wird also vor dem effektiven Mountainbike-Ausflug verwendet. Dabei liegt der Fokus auf längeren Mountainbike-Routen und lassen einzelne Trails und Trailabschnitte ausser Acht.

1.3.2 Pinkbike

Pinkbike ist mit über 700'000 Nutzern die grösste Mountainbike Community im Internet. Im Zentrum stehen dabei User-generierte Inhalte und der Austausch zwischen den Bikern. Pinkbike ist exklusiv über den Browser bedienbar und bietet keine mobile App.

Features

Media Feed	In einem Foto- und Videofeed können von Usern Fotos gepostet, bewertet und kommentiert werden. Dabei werden die am besten bewerteten Medien als „Photo of the day" ausgezeichnet (Motivation).
Marktplatz	Im Marktplatz kann gebrauchte Ausrüstung und Bikes ge- und verkauft werden.
Blogs & News	Es gibt einen Newsbereich, in welchem laufend Blogposts, Kritiken, Interviews und Pressemeldungen veröffentlicht werden.
Friendship-System	Wie in den meisten Communities gibt es bei Pinkbike ein Friendship-System. Dabei kann man „Freundschaften" mit anderen Ridern eingehen, um in Kontakt zu bleiben. Dazu steht auch eine Art Mailsystem zur Verfügung.
Forum	Für den Austausch zu allgemeinen Mountainbike-Themen werden diverse Foren zur Verfügung gestellt.

Tabelle 2: Feature Übersicht Pinbike

Fazit

Der Fokus von Pinkbike liegt ganz klar auf der Community. Diese teilt sehr aktiv multimediale Inhalte (Fotos und Videos). Auch der Marktplatz hat tagesaktuelle Angebote und scheint oft genutzt zu werden.

Pinkbike.com ist nur in Englisch verfügbar, wodurch die Plattform für den Schweizer Markt nur begrenzt attraktiv ist. Das Design der Plattform wirkt veraltet, stärkt aber das Gemeinschaftsgefühl der Biker.

1.3.3 Trailforks

Trailforks ist eine Mountainbike Trail Datenbank und Management System für Mountainbiker, Trailbauer und Mountainbike-Vereine. Nutzer können Daten zur Datenbank beisteuern, diese werden aber von lokalen Vereinen kontrolliert, bewilligt und kuratiert, um eine möglichst hohe Datenqualität zu gewährleisten. Ergänzt werden die Traildaten auch durch User-generierte Inhalte von Pinkbike.

Features

Trails	Über eine Suche, eine Listen- oder eine Kartenansicht kann nach Trails gesucht werden. Zu den Trails gibt es qualitativ hochwertige, kuratierte Inhalte. Bewertungen und Zustandsmeldungen werden von den Pinkbike Benutzern bereitgestellt.
Touren	Touren funktionieren ähnlich wie Trails, sind aber länger und haben dadurch unterschiedliche Daten.
Berichte	Zu einem Trail oder einer Tour können Berichte verfasst werden. Diese Berichte können von kurzen Zustandsmeldungen bis hin zu Ausführlichen Streckenberichten reichen.

Tabelle 3: Feature Übersicht Trailforks

Fazit

Trailforks, komplementiert durch Daten von Pinkbike, ergeben eine Plattform, die Traildevils sehr ähnlich ist. Die Datenqualität ist sehr hoch, da sie akribisch von lokalen Vereinen geprüft wird. Die Zusammenarbeit zwischen Trailforks und diesen Vereinen ist sehr wichtig und funktioniert offensichtlich gut.

Zwar hat Trailforks viele Trails und Touren in der Schweiz, allerdings sind die Communitybeiträge wie z.B. Zustandsberichte eher selten zu finden. Das könnte unter anderem daran liegen, dass Trailforks nur auf Englisch verfügbar ist.

Trailforks ist via Webbrowser oder auch als mobile App auf iOS und Android nutzbar.

1.4 Einschränkungen

Nachdem die Anforderungen an diese Projektarbeit und das Umfeld der Traildevils Plattform analysiert worden sind, wird folgend auf die Einschränkungen der Projektarbeit eingegangen.

1.4.1 API

Die App greift auf Daten von einer Schnittstelle (API) von Traildevils zu. Die Umsetzung dieser Schnittstelle ist nicht Bestandteil dieser Arbeit. Allerdings wird eine API Spezifikation erstellt. Auf Basis dieser werden die Schnittstellen umgesetzt. Damit dennoch ein Prototyp der App entwickelt werden kann, wird ein Mock der Schnittstelle verwendet So-mit können reibungslose Entwicklungs- und Testprozesse gewährleisten werden.

1.4.2 Funktionsumfang

Die Traildevils Webseite bietet viele unterschiedliche Nutzungsbereiche wie z.B. die Bike Map, den Marktplatz oder News an. Die zu entwickelnde App konzentriert sich ausschliesslich auf die Bike Map.

1.4.3 Technologie

Die von der App genutzte Technologie ist nicht festgelegt und muss zuerst evaluiert werden. Eine Kernanforderung an die Technologie ist Cross-Plattformunterstützung für iOS und Android. Der Praxispartner hat Know-How in den Webtechnologien ASP.NET, HTML, CSS und JavaScript, was bei der Wahl ebenfalls beachtet werden soll.

1.4.4 Plattform

Die primäre Zielplattform der App ist das iPhone (aktuelle iOS Version). Sekundär sind Android-betriebene Smartphones zu unterstützen. Die App soll unter Android zwar lauffähig sein, aber leichte Einschränkungen werden toleriert. Andere System oder Geräte wie z.B. Tablets werden nicht betrachtet.

2 Analyse

2.1 Domainanalyse

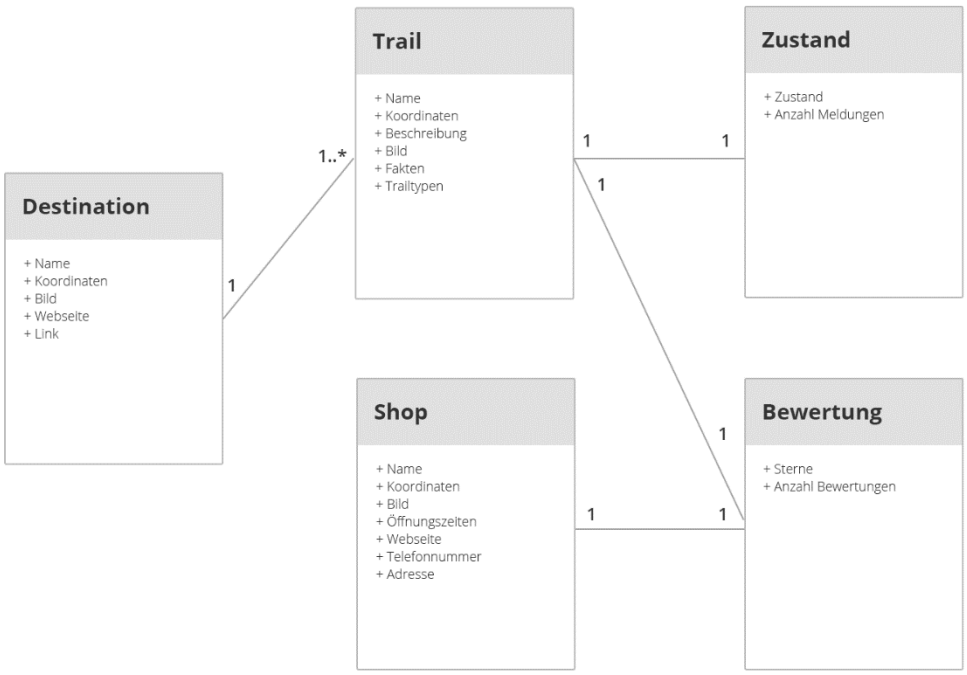

Abbildung 1: Domain Model der Traildevils App

Nachfolgend werden die wichtigsten Begriffe aus dem Domain Modell kurz erklärt.

Destination

Eine Destination bietet Informationen über Bikeregionen. So kann eine Region mit speziellen Bildern und mehr Informationen versehen werden. Ein Trail kann mit mehreren Trails verknüpft sein. Destinationen sind keine Nutzer-generierten Inhalte, sondern gehören zum Geschäftsmodell.

Trail

Ein Trail ist das Hauptelement der App. Sie repräsentieren Fahrtwege für Biker. Die Trails können von Benutzern erfasst und bewertet werden. Ein Trail hat einen Zustand, welcher die Fahrbarkeit zeigt.

Zustand

Ein Zustand beschreibt die Fahrbarkeit eines Trails. Die Daten werden durch den Server anhand der letzten Meldungen errechnet.

Shop

Shops zeigen Informationen über Werkstätte und Händler an. Diese sind wie die Destinationen keine Nutzer-generierten Inhalte, sondern kostenpflichtige Einträge.

Bewertung

Innerhalb von Traildevils können Trails und Shops mit einer Wertung von 1 – 5 Sternen bewertet werden. Das Objekt ist ein vom Server berechneter Durchschnitt der Sternbewertungen.

2.2 User Journeys

Die Schlüsselszenarien für die Traildevils App wurden als User Journeys festgehalten. Von diesen insgesamt fünf Journeys wurde bis zum Projektende die drei Kernszenarien vollumfänglich, die restlichen zwei teilweise implementiert.

Die User Journeys beinhalten einfach gehaltene Personas, welche aus vorgegebenen Interessen bestimmte Aktionen vornehmen. Diese Interessen, Bedürfnisse und Aktionen werden in den Journeys definiert und beschrieben. Anhand der folgenden fünf Journeys wurden Job Stories abgeleitet, welche als Implementierungstasks betrachtet werden.

2.2.1 Journey 1: Trails in Davos

Danny, 26, ist übers Wochenende mit seinem zwei besten Freunden in Davos. Heute, Samstag, ist die Gruppe bereits eine längere Tour mit einigen anspruchsvollen Trails gefahren. Bei einem Bier besprechen die Freunde, was für eine Tour, respektive was für Trails, sie morgen fahren wollen. Da Danny und seine Freunde sich in Davos und Umgebung nicht auskennen, startet Danny die Traildevils App. Auf einer Kartenansicht blendet er alle Aktivitäten in seiner Umgebung aus, ausser den Trails und Touren. Die Gruppe befindet den Wolfgangtrail am Weissfluhjoch als spannend. Danny wählt den Trail an und gelangt zur Detailseite des Trails. Hier erhält Danny wichtige Informationen zum Trail, dessen Beliebtheit und Zustand.

Dank diesen Merkmalen und Kennzahlen entscheidet die Gruppe, diesen Trail morgen zu fahren.

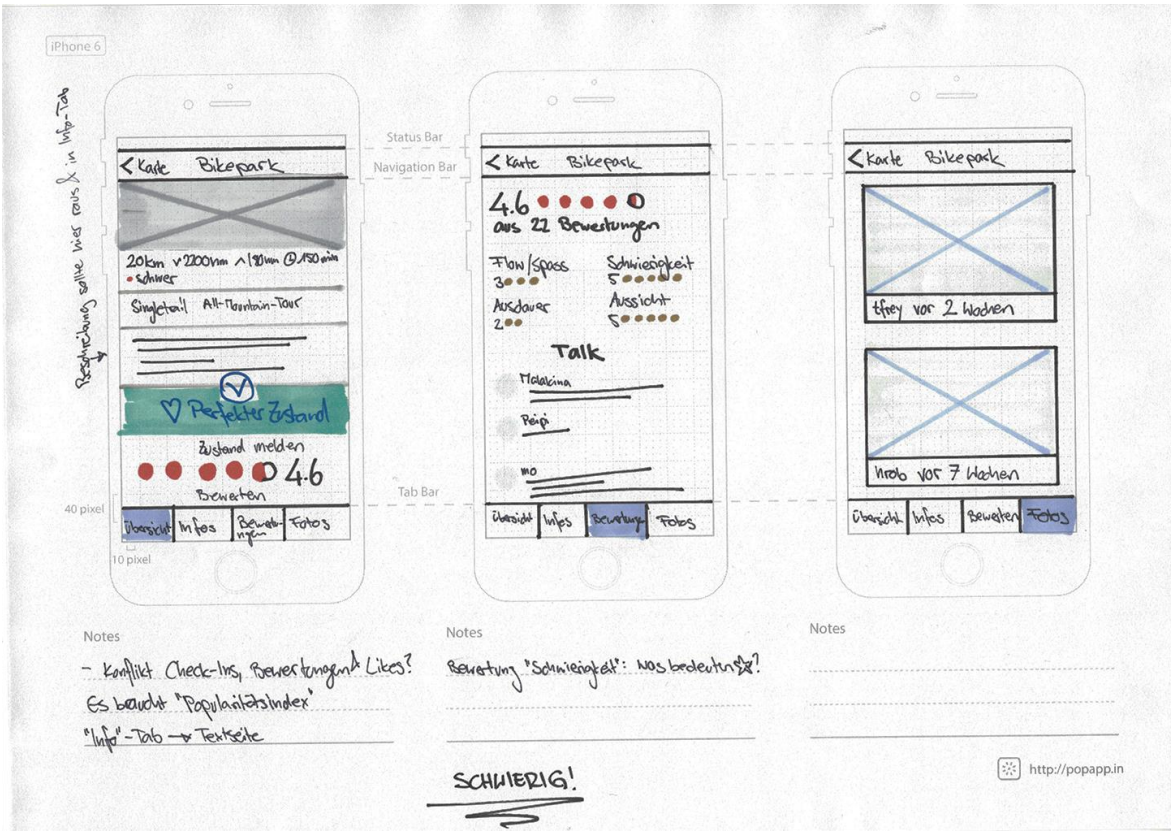

Abbildung 2: Skizzen zur Umsetzung der Journey «Trails in Davos»

2.2.2 Journey 2: Gestauchte Felge

Claudio, 31, verbringt eine Woche in der Lenzerheide um ein paar Touren zu fahren, aber vor allem um sein neues Santa Cruz Nomad auf einer anständigen Downhillstrecke zu fahren. Gegen Ende des ersten Tages dann der Schock: bei einer Landung hat es die Felge seines Vorderrads so verzogen, dass er das Problem nicht selbst beseitigen kann. Da er den Urlaub aber nicht abbrechen will, startet er die Traildevils App, um nach einem Bike-Shop für Santa Cruz zu suchen. Auf der Kartenansicht wählt er den Shop-Layer an und sieht, dass es mehrere Shops in seiner Nähe hat. Er wählt einen der Shops an und sieht daraufhin die wichtigsten Informationen zum Shop, wie die Adresse oder die Öffnungszeiten. Da der Shop mit 4.8 Sternen eine sehr gute Bewertung erhalten hat, klickt Claudio auf die Telefon-Nr. des Shops, um telefonisch kurz abzuklären, ob die Reparatur möglich ist. Glücklicherweise kann diese schon am nächsten Morgen erledigt werden, sodass Claudio und sein Santa Cruz bereits am nächsten Tag wieder zum *shredden* bereit sind.

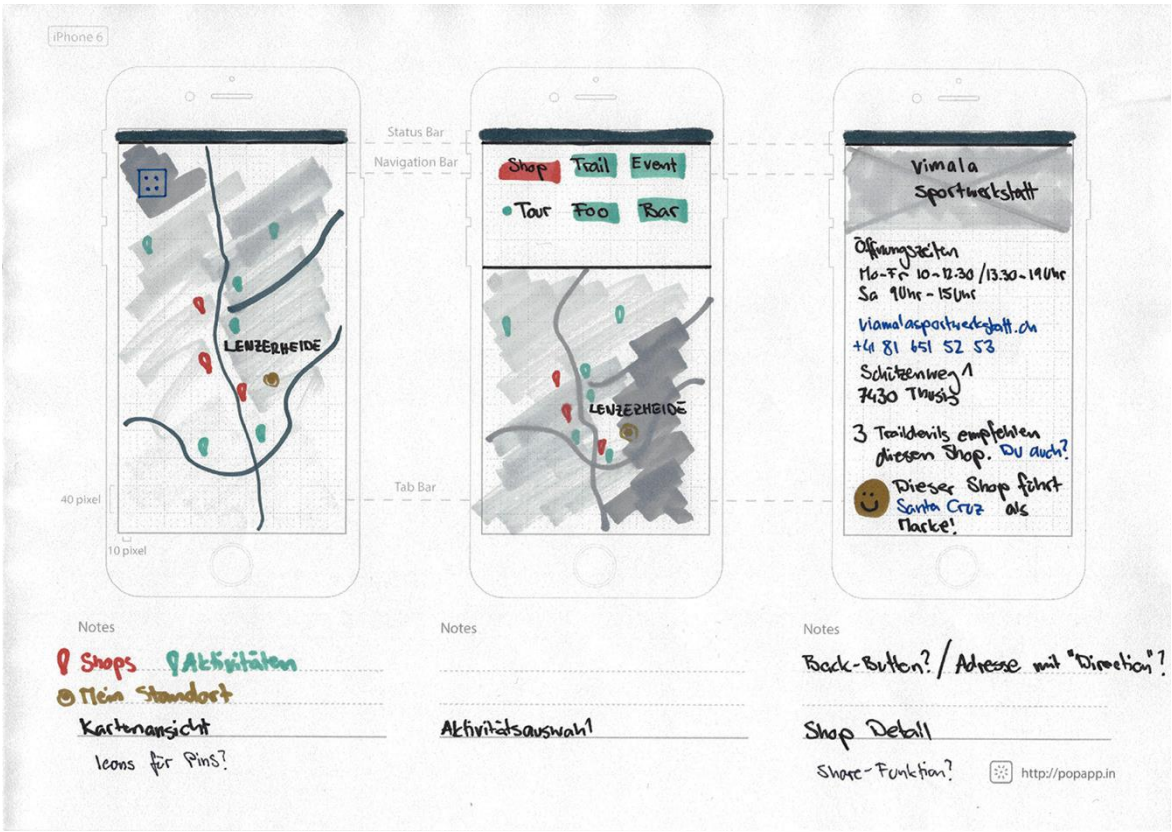

Abbildung 3: Skizzen zur Umsetzung der Journey «Gestauchte Felge»

2.2.3 Journey 3: Bewerten eines gefahrenen Trails

Danny und seine Freunde sind heute, Sonntag, in Davos unterwegs und haben gerade einen von Traildevils empfohlenen Trail absolviert. Die Gruppe erholt sich bei Nussgipfel und Panaché in einem Davoser Kaffee, bevor es nach Hause geht. Danny, der die Fahrt mit Strava aufgezeichnet hat, stoppt den GPS-Tracker und speichert die Aktivität bei Strava.

Kurz darauf vibriert sein iPhone aufgrund einer Notification von Traildevils. »Wir haben gesehen, dass du den Älplisee Trail gefahren bist! Wie wars?« erscheint auf seinem Display. Da gerade Holzschlag in den Davoser Wäldern ist und Teile des Trails nicht befahrbar sind, beschliesst Danny den Zustand des Trails als *schlecht* zu melden, damit weitere Traildevils vorgewarnt sind.

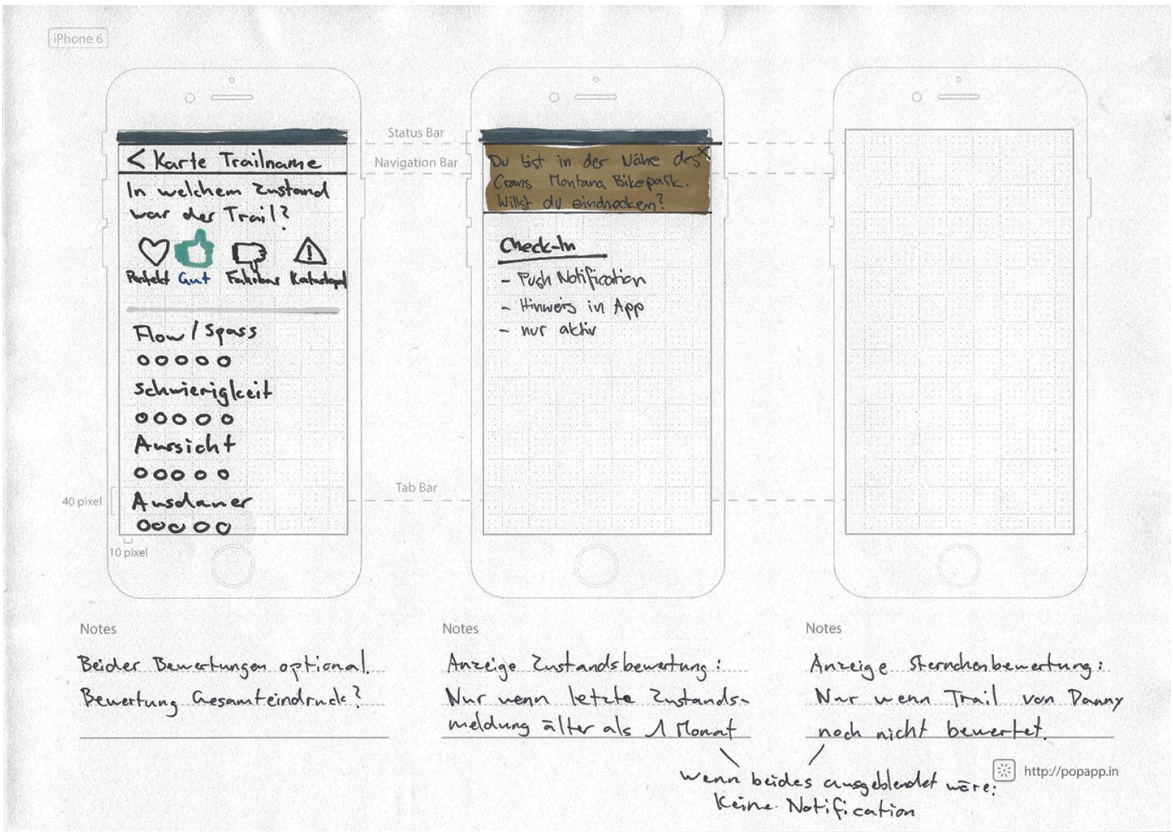

Abbildung 4: Skizzen zur Umsetzung der Journey «Bewerten eines gefahrenen Trails»

2.2.4 Journey 4: Umfrage Lenzerheide

Diana, 32, sitzt am Donnerstagmorgen im Zug zur Arbeit noch Chur und vertreibt sich die Zeit damit, in der Traildevils App nachzuschauen ob die Trails in der Umgebung in gutem Zustand für einen allfälligen Ausflug am Wochenende sind. Als sie die Karte in der App nach Trails absucht, fällt ihr am unteren Rand eine kleine Meldung auf, die sie zur Teilnahme an einer Umfrage über ihr Lieblingsgebiet in der Nähe, die Lenzerheide, auffordert. Da sie oft in diesem Gebiet zu Besuch ist, sich dafür interessiert und gerade Zeit hat, beschliesst sie an der Umfrage teilzunehmen und klickt auf die Meldung, wodurch die Umfrage gestartet wird.

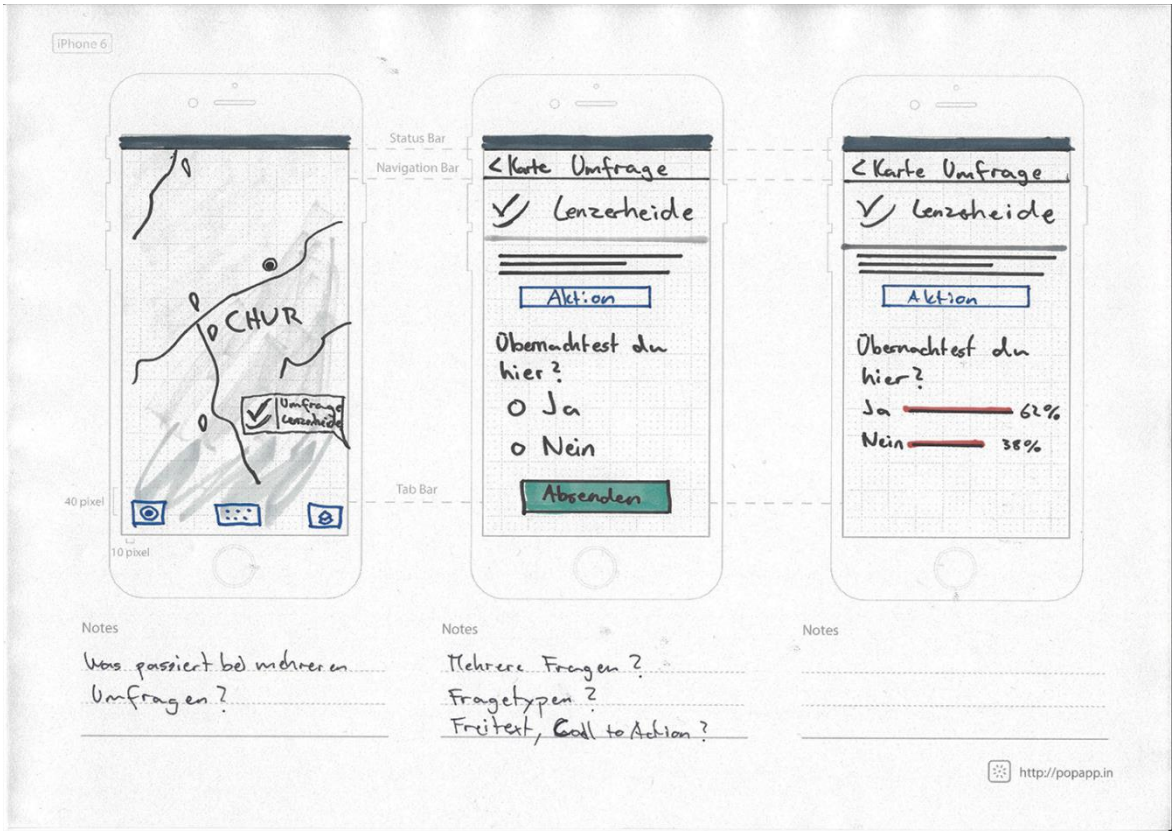

Abbildung 5: Skizzen zur Umsetzung der Journey «Umfrage Lenzerheide»

2.2.5 Journey 5: Teilnahme an einem Live-Event

Claudios Urlaub in der Lenzerheide ist natürlich so geplant, dass er die zur gleichen Zeit stattfindende WM live mitverfolgen kann. Er befindet sich im Zielraum und genehmigt sich ein quellfrisches Calanda, als sein Samsung Galaxy S5 vibriert. Traildevils hat erkannt, dass Claudio in der Nähe des Events ist und sendet daher eine Benachrichtigung. Die App bietet ihm die Möglichkeit sich am Event “einzuchecken”, damit andere Traildevils seine Anwesenheit sehen. Claudio entscheidet sich denn Check-In zu überspringen, und landet auf dem Live-Stream des Events. Er sieht dabei Fotos anderer Nutzer, kurze Videos von Traildevils, Kommentare zu gewissen Jumps und eine Umfrage, gestellt von der MTB-Destination Lenzerheide.

2.3 Funktionale Anforderungen

Die Anforderungen an die App wurden aufgrund der offenen Aufgabenstellung laufend verändert. Daher wurde ein agiler Ansatz gewählt, bei welchem die Anforderungen laufend definiert und priorisiert werden.

Auf Basis der erarbeiteten User Journeys wurden Tasks erstellt. Diese Tasks wurden als Job Stories [2] in der Form «Wenn ich _____, möchte ich _____, damit ich _____.» erfasst. Des Weiteren wurden zusammen mit dem Kunden für alle eingeplanten Backlog Items Akzeptanzkriterien erfasst, welche als Definition of Done (DoD) [3] dienten.

Im Projektmeeting vom 1. März 2016 wurde gemeinsam entschieden, dass diese Job Stories, zusammen mit auf GitHub erfassten Akzeptanzkriterien, als Funktionale Anforderungen genügen.

Eine ausführliche Liste über alle bearbeiteten Backlog Items ist im Anhang (Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.**) zu finden.

2.4 Nicht-funktionale Anforderungen

In der folgenden Tabelle werden die Nicht-funktionalen Anforderungen an den Prototypen der Traildevils App aufgeführt.

Plattform (Abnahmetest: Usability Test)	Die App soll auf Android und iOS Smartphones lauffähig sein. Primäre Plattform ist jedoch iOS.
	In dieser Arbeit wird die App auf folgenden Smartphones getestet: OnePlus One, iPhone 6 Plus
Performance (Abnahmetest: Usability Test)	Die App muss in unter 3 Sekunden gestartet sein und auf Nutzereingaben reagieren.
	Die Nutzer empfinden das navigieren innerhalb der App als flüssig.
Wartbarkeit (Abnahmetest: Code Review)	Die App soll soweit möglich mit dem Kunden bekannten Web-Technologien umgesetzt werden (JavaScript, HTML, CSS, ASP.NET).
	Für die eingesetzte Technologie muss auf dem bevorzugten OS des Kunden eine Entwicklungsumgebung vorhanden sein.
Sicherheit (Abnahmetest: Code Review)	Sensitive Login-Daten müssen lokal und während der Übertragung gesichert sein.
UI/UX (Abnahmetest: Usability Test)	iOS sowie Android Nutzer müssen die App ohne Schwierigkeiten bedienen können.
	Plattformspezifische Aktionen (z.B. Zurück-Button bei Android) müssen unterstützt sein.
	UI soll das Farbschema der Traildevils Webseite anwenden.
Verfügbarkeit (Abnahmetest: Usability Test)	Die App bietet vollen Funktionsumfang, solange das Smartphone Verbindung zum Internet und Lokalisierungsdiensten hat.
	Die App bietet ein sinnvolles Subset an Funktionen an, solange keine Lokalisierungsdienste aber Internet verfügbar sind.
Internationalisierung	Die App ist nur auf Deutsch verfügbar, soll jedoch relativ einfach um weitere Sprachen erweitert werden können.
	Die App muss auf den Schweizer Markt zugeschnitten sein.

Tabelle 4: Übersicht der NFRs

2.5 Technologie-Evaluation

Ein wichtiger Teil dieser Arbeit war die Technologie-Evaluation. Eine möglichst einfache Cross-Plattform-Fähigkeit war eine der Anforderungen an die Projektarbeit. Um diese Evaluation durchführen und verschiedene Technologien miteinander vergleichen zu können, wurde ein Kriterienkatalog entwickelt. Mit Hilfe dieses Kriterienkatalogs wurden vier aktuell verbreitete Technologien bewertet. Jedes Kriterium wurde mit einer Punktzahl zwischen 5 und 1 bewertetet, wobei fünf der beste Wert ist. Für jede Technologie ergab sich daraus ein Indexwert. Die Technologie mit dem höchsten Index stellt auch die am besten geeignete Technologie dar.

Die zu evaluierenden Technologien/Framework sind Cordova, NativeScript, React Native und Xamarin. Diese Technologien verfolgen drei deutlich unterschiedliche Ansätze, die vor der Evaluationsauswertung kurz erklärt werden.

Die Evaluation wurde im Februar 2016 aufgrund der zu diesem Zeitpunkt verfügbaren und gültigen Informationen erstellt. Xamarin wurde in der Bewertung schlechter eingestuft, da es die einzige Closed Source- und Lizenzpflichtige Technologie war. Zwischen der Evaluation und der Fertigstellung dieser Projektarbeit wurde Xamarin (das Unternehmen) von Microsoft übernommen (Februar 2016), die Lizenzkosten abgeschafft und eine vollständige Integration in Visual Studio gewährleistet (April 2016). Zusätzlich ist die Veröffentlichung der gesamten Code-Basis geplant und kommuniziert, zu diesem Zeitpunkt (Juni 2016) aber noch nicht umgesetzt. Diese Umstände hätten wahrscheinlich ein anderes Ergebnis der Technologie-Evaluation ergeben. Da dies aber zu einem späten Zeitpunkt geschah, wurde keine Re-Evaluation vorgenommen.

2.5.1 Cross Compiling

Der Cross Compiling Ansatz ist seit ein paar Jahren geläufig und wird produktiv eingesetzt. Der wohl bekannteste Player im mobilen Bereich ist Xamarin. Beim Cross Compiling Ansatz wird der geschriebene Source Code in den von der gewünschten Plattform unterstützen Source Code transpiliert. Für iOS würde also z.B. der Source Code in Objective-C transpiliert (Source-to-Source Compiling) und läuft so effektiv nativ unter iOS. Für Android

wird Mono als Virtual Machine und Intermedia Language (IL) Code ausgeliefert. Es wird hier also kein Java Code generiert, sondern nach wie vor .NET-Technologie ausgeführt.

Dieser Ansatz hat den Vorteil, dass Entwickler eine Code Basis in einer Programmiersprache entwickeln können und diese anschliessend trotzdem ohne Performanceeinbussen auf unterstützten Plattformen einsetzen können, welche sonst keine Kompatibilität für diese Programmiersprache bieten würden.

Xamarin

Xamarin ist seit dem Frühjahr 2016 ein Teil von Microsoft und beschäftigt sich sehr mit der Cross-Plattform-Fähigkeit des gesamten .NET-Frameworks und hat daher grosse Anteile an der Entwicklung von Mono.

Mit Xamarin wird Code auf Basis von C# und dem .NET-Framework entwickelt. View-Controls werden in einem Superset von XAML geschrieben. Für iOS transpiliert Xamarin den Source Code zu Objective-C, bei Android wird jedochkein richtiger Cross Compiling Ansatz verfolgt, sondern es wird Intermediate Code zusammen mit einer Mono Virtual Machine ausgeliefert. Da Mono allerdings sehr fortgeschritten ist, sind keine Performance Einbussen spürbar.

2.5.2 Embedded WebView

WebViews sind der wohl älteste und bekannteste Versuch eine Code-Basis auf verschiedene Betriebssysteme lauffähig zu machen. Eine native App bietet die Rahmenbedingungen für die zu entwickelnde App und stellt ein WebView Control zur Verfügung. In diesem Control wird der geschriebene Code ausgeführt. Das Control hat dieselben Fähigkeiten und Eigenschaften wie ein Browser. Somit wird effektiv eine Web-App entwickelt und auf dem Gerät ausgeführt.

Von den drei untersuchten Ansätzen ist dieser wohl der etablierteste. Daher gibt es auch diverse Anbieter, die Tools und Frameworks zur Verfügung stellen.

Cordova

Der bekannteste Anbieter ist PhoneGap. Das kleine, aus den USA stammende, Start-Up hat ein Framework entwickelt, das native APIs wie z.B. die Kamera oder Geolokalisierung

als JavaScript APIs in der WebView exponiert. PhoneGap wurde 2011 von Adobe übernommen und weitergeführt. Inzwischen hat Adobe den Quellcode des Produktes an die Apache Foundation übergeben, welche das Projekt seither unter dem Namen Apache Cordova weiterführt. Adobe bietet zusätzliche Module und kostenpflichtige Dienstleistungen im Umfeld von Cordova.

2.5.3 JavaScript Runtime Engine

Der jüngste Ansatz baut ebenfalls auf Webtechnologien auf. Die gesamte Code-Basis wird in JavaScript geschrieben. Das native Framework stellt nicht mehr eine ganze WebView sondern lediglich eine JavaScript Runtime zur Ausführung des Codes zur Verfügung. Unter iOS kommt hier JavaScriptCore (Webkit), unter Android V8 (Chrome) zum Einsatz. In einer XML-ähnlichen Syntax wird das UI deklarativ geschrieben und nativ gerendert. Dazu stellt das Framework einen eigenen View Manager zur Verfügung. Dieser View Manager kommuniziert mit der JS Runtime Engine, in welcher der restliche Code ausgeführt wird.

Die populärsten Vertreter dieses Konzept sind Telerik und Facebook mit NativeScript respektive React Native.

NativeScript

Telerik hat im Frühjahr 2015 Native Script vorgestellt. Ein UI-Control in Native Script wird mit XML und Styles mittels einem CSS-Subset geschrieben – diese Dateien sind getrennt. Die UI zugehörende Logik («Code behind») ist in einer dritten Datei gespeichert, einer JavaScript-Datei. Die CSS und die HTML-Datei werden nativ kompiliert und sind daher sehr performant. Das JavaScript-Modul wird in dem eigenen Thread ausgeführt.

React Native

Ebenfalls im Frühjahr 2015 hat Facebook React Native veröffentlicht. Im Gegensatz zu NativeScript werden UI-Controls in einem einzigen JavaScript-Modul geschrieben, der Ansatz bleibt aber sehr ähnlich.

Der wesentliche Unterschied zwischen den Lösungen von Telerik und Facebook ist nicht technischer Natur, sondern die Community-Unterstützung. Während Telerik zwar eine sehr saubere Entwickler-Dokumentation zur Verfügung stellt, konnte NativeScript die

Entwickler-Community nicht für sich gewinnen. React Native punktet genau hier; die Anzahl an Tutorials, Guides, Dokumentation und Beispielen scheint schier grenzenlos zu sein. Viele Best Practices und Bibliotheken können zudem von React.js übernommen werden. Diese sind eigentlich für den Browser bestimmt, solange jedoch das DOM nicht modifiziert wird, sind sie grösstenteils auch unter React Native gültig.

2.5.4 Evaluationsmatrix

	React Native	41	NativeScript	38	Xamarin	39	Cordova	29
Entwicklungsumgebung	OS X, Windows & Linux	3	OS X, Windows, Linux	5	OS X, Windows	4	OS X, Windows, Linux	5
Cross-Plattform	iOS, Android ((noch keine Feature-parity)	3	iOS, Android	4	iOS, Android, Windows 10	5	iOS, Android, Windows 10	5
Technologien	JavaScript, React, Flexbox (Subset)	4	JavaScript, XML, CSS (Subset)	4	XAML, C# (keine Web-technologien)	1	JavaScript, HTML5, CSS	5
Native Performance	Natives UI, aber JS	4	Natives UI, aber JS	4	Alles Native	5	Web-App, nicht nativ	1
Native Karten	Js	4	Nein, nur einfaches Google Maps	1	Ja	5	Nein	2
Native UI Komponenten	Vieles Out-of-the-Box, viele Packages	4	Vieles out-of-the-Box, wenige Plugins	2	Vieles Out-of-the-Box, grosser COmponent	4	Vieles Out-of-the-Box	2
Native Look & Feel	Ja	5	Ja	5	Ja	5	Nein	1
Dokumentation	Gute Doku, viele Community Beiträge	4	Gute Doku von Telerik	3	Gute Doku, grosses Hilfe-Forum	4	Teilweise veraltet, wenig (aktuelle) Community	2
BC in neuen Releases	Minimal und nur in Major Releases	5	Minimal und nur in Major Releases	5	Minimal und nur in Major Releases	5	Mehrfach in Core	1
Kosten / Lizenz	BSD (OSS)	5	Apache Licence 2.0 (OSS)	5	Rund $1000 / Jahr (Closed Source)	1	Apache Licence 2.0 (OSS)	5
Status	Aktive Entwicklung durch Facebook (GitHub)		Aktive Entwicklung durch Telerik (GitHub)		Aktive Entwicklung durch Xamarin		Aktive Entwicklung durch Apache	
Veröffentlichung	März 2015		Mai 2015		Mai 2011		Oktober 2011	

Tabelle 5: Evaluationsmatrix

2.5.5 Evaluationsergebnis

Am Ende der Evaluation zeigte sich, dass **React Native** die besten Voraussetzungen mitbringt, um die Anforderungen in diesem Projekt zu erfüllen. Es erfüllte die wichtigsten Kriterien wie gute Cross-Plattform Möglichkeiten und die Nähe zu Webtechnologien.

Das technologisch sehr ähnliche NativeScript fällt aufgrund der schwachen Community und Verbreitung hinter React Native zurück. Ebenfalls schwach war die Unterstützung für (native) Karten-Applikationen, was in einer prototypischen Umsetzung getestet wurde.

Xamarin bietet zwar die beste und am weitesten entwickelte Plattform für Cross-Plattform Entwicklung, fällt jedoch aufgrund der Distanz zu gewohnten Webtechnologien, den Kosten und dem Closed Source Ansatz zurück.

Cordova kann nicht überzeugen, da eine Webview nicht die gewünschten Vorteile gegenüber einer mobilen Webseite bringen kann.

3 Umsetzung

3.1 Code Style Guidelines

Als Basis für die Code Style Guidelines wurden Guidelines von Airbnb übernommen. Airbnb hat sehr gründliche und ausgeklügelte Guidelines, daher werden diese im JavaScript Ökosystem oft wiederverwendet.

Für die Airbnb-Plattform wird voll auf React gesetzt, daher gibt es auch spezifische Guidelines zur Entwicklung mit React, welches auch in dieser Arbeit zur Anwendung kommt. Diese Guidelines sind extrem strikt und brauchten vor allem zu Beginn der Arbeit Eingewöhnungszeit. Beispielsweise sollen zustandslose Komponenten und Komponenten ohne Lifecycle-Methoden pure Funktionen anstatt ES6 Klassen verwenden.

Zur Sicherstellung der Einhaltung dieser Guidelines wurde ESLint verwendet und in den Continuous Integration-Prozess integriert. ESLint ist ein modulares Lint-Werkzeug für JavaScript und JSX, für welches Regeln von Airbnb geschrieben wurden.

Neben ESLint wurde auch Flow von Facebook verwendet. Flow ist ein statischer Typechecker, der dabei hilft Fehler in JavaScript Applikationen zu finden. Dabei versucht Flow

das schwache Typensystem von JavaScript mit statischen Typen zu ergänzen. Alle Files welche mit einer `@flow` Annotation versehen sind, werden automatisch geprüft.

3.2 UX / UI Konzept

3.2.1 Paper Prototype

Zur Sicherstellung einer guten User Experience und einer konsistenten Designsprache wurde vor der Implementation ein Paper Prototype (Anhang A) entwickelt.

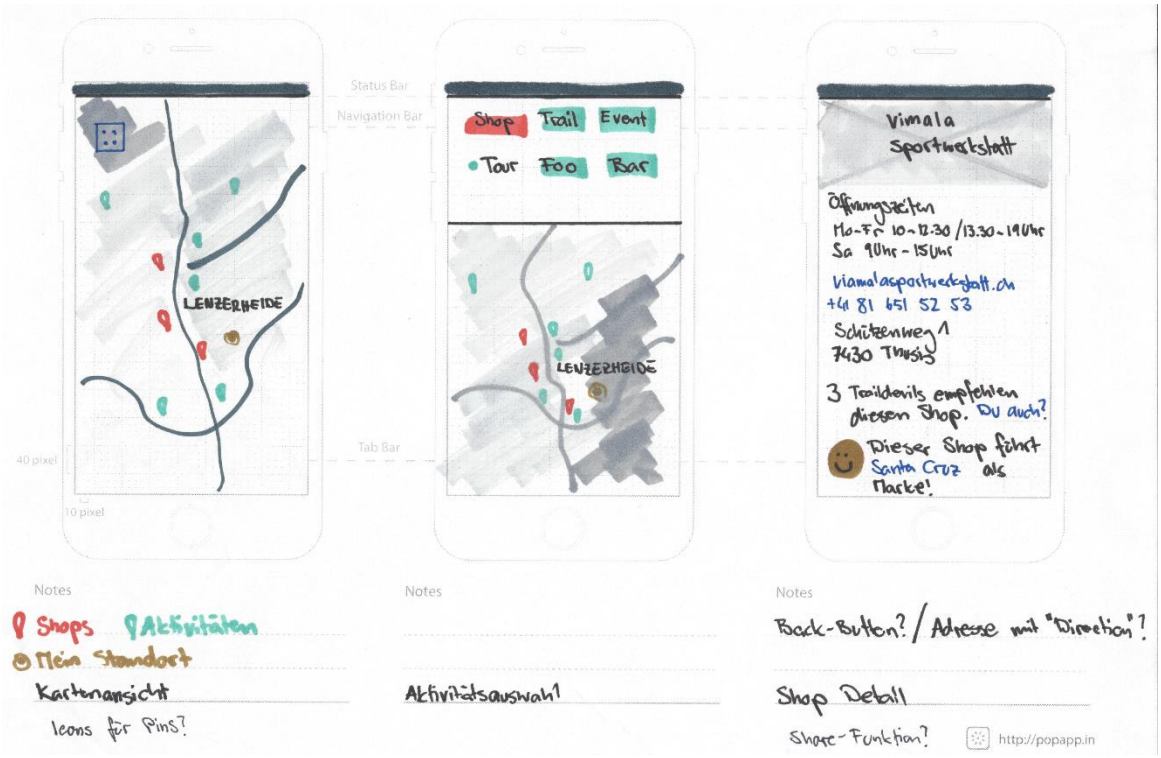

Abbildung 6: Auszug aus dem Papierprototypen

Dieser Prototyp wurde zusammen mit Daniel Demel, Senior Art Director bei der Firma Namics, hinterfragt [4]. Die Diskussion und Inputs waren sehr hilfreich und schon in einem frühen Stadium richtungsweisend. Dabei zeigte sich, dass das Konzept des Paper Prototypes gut umsetzbar ist. Die wichtigsten Änderungen, die aus dem Gespräch hervorgingen, sind die folgenden:

Kein Onboarding

Die App soll keinen Onboarding-Prozess verlangen und somit die Einstiegshürden minimieren. Das bedeutet auch, dass weder Login noch Registrierung vom Nutzer verlangt

werden. Die Nutzung der Karte und Betrachtung der Trails und Shops benötigt keine Authentifizierung. Daher soll der Login- oder Registrierungsprozess erst angestossen werden, wenn effektiv Nutzerdaten benötigt werden. Das ist zum Beispiel bei der Abgabe von Zustandsmeldungen der Fall.

Heat-Map

Eine Möglichkeit Aktivität auf der Plattform zu visualisieren, ist eine Art Heat-Map. Diese Heat-Map kann verschiedene Ausprägungen und Faktoren haben. So könnten aktive Nutzer auf der Karte eingezeichnet, oder aber oft befahrene Trails visuell gekennzeichnet werden. Einzelne Regionen auf der Karte werden dadurch hervorgehoben und wirken interessanter.

Overlay anstatt Navigation

Anstatt eine neue Ansicht zu öffnen, soll die Preview bei einem Klick zu einem Vollbild-Overlay expandieren. Durch die Animation sind die Geschehnisse für den Nutzer nachvollziehbar und klar. Die Karte verschwindet nicht, sondern wird nur temporär verdeckt.

Suchverlauf und Quick-access

Der Suchverlauf und Quick-access Funktionalität (Kategorien, favorisierte Trails, etc.) kann und soll direkt in die Suche integriert werden. Beim Anwählen der Suche sollen initial die letzten ausgewählten Suchergebnisse angezeigt werden. Nach dem Tippen eines neuen Suchbegriffs werden diese durch aktuelle Resultate ersetzt.

Zusätzliche Nutzer-generierte Inhalte

Um den Community-Fokus zu stärken ist es zwingend nötig Fotos schnell und einfach hochladen zu können. Andere Nutzer sollen diese Fotos auch kommentieren können. Werden nun Benachrichtigungen («Claudio hat dein Foto kommentiert») mit ein bezogen, wird das Communitygefühl und der Austausch zwischen den Nutzern gestärkt.

3.2.2 Cross-Plattform Problematik

Bei Cross-Plattform Apps gibt es für das User Interface zwei Möglichkeiten. Eine davon ist, konsequent auf native UI Controls zu setzen. Das birgt aber einige Gefahren. Nicht alle Patterns sind auf allen Plattformen abgebildet, so können gewisse Elemente oder Nutzerprozesse auf einzelnen Plattformen nicht so gut funktionieren wie auf anderen. Auch ist es schwierig, die eigene Designsprache den Nutzern zu vermitteln.

Das Hauptargument für native Controls ist meist, dass Nutzer sich auf der Plattform auskennen und daher Parallelen ziehen und bereits gelernte Effekte wiedererkennen. Als Gegenargument kann aufgeführt werden, dass Nutzer mit Plattformen wie Facebook, Instagram oder eben Traildevils ebenfalls vertraut sind und spezifische Verhaltensmuster schon gelernt haben.

Grundsätzlich ist Demel im Experteninterview der Meinung, dass ein natives Look & Feel nicht essentiell ist. Viele Elemente, gerade in einer App wie Traildevils, sind spezifisch auf die Use-Cases zugeschnitten. Daher ist es wichtig, dass die App eine konsistente Designsprache über verschiedene Medien einhält.

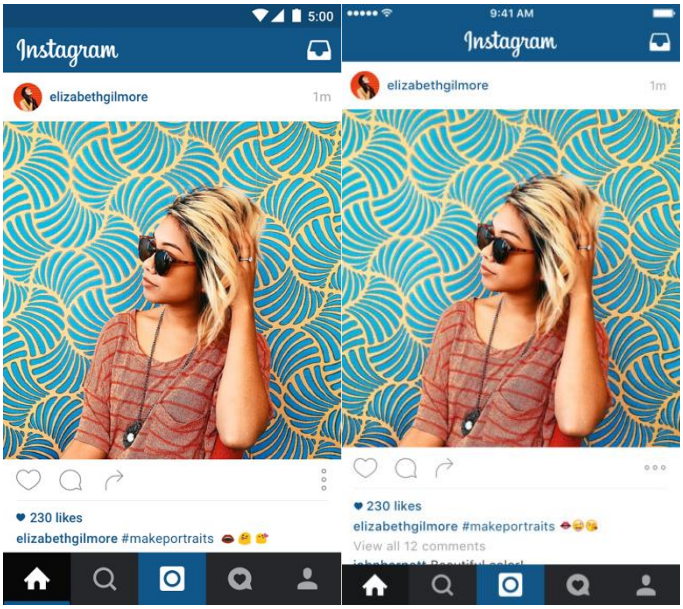

Abbildung 7: Instagram unter Android und iOS im Direktvergleich

Um diese These zu überprüfen, wurden die sehr populären Apps Facebook, Instagram, WhatsApp, Google Maps, 20min und Snapchat miteinander verglichen. Dabei hat sich gezeigt, dass keine dieser Apps auf native Controls aufbaut. In all diesen Apps werden meistens Elemente verwendet, die besser zur allgemeinen Designsprache der Plattform passen. Nutzer, welche die Webseite bereits kennen finden sich so schnell zurecht.

3.2.3 Suche

Die Suche ist das zentrale Element der Traildevils App. Allerdings ist es keine reine Suche, sondern bietet auch schnelle Zugangspunkte zu oft verwendeten Inhalten.

Wird die Suche angewählt, werden initial die letzten ausgewählten Objekte wie Trails oder Destinationen angezeigt. Ergänzend besteht die Möglichkeit, einen Zugriff auf weitere Elemente, wie favorisierte Trails, in dieser Liste zu bieten. Erst bei der Eingabe eines neuen Suchbegriffs werden die Objekte durch die Resultatliste ersetzt. Dabei wird nach Objekten von Traildevils und nach Ortschaften (Geolokalisierung) gesucht.

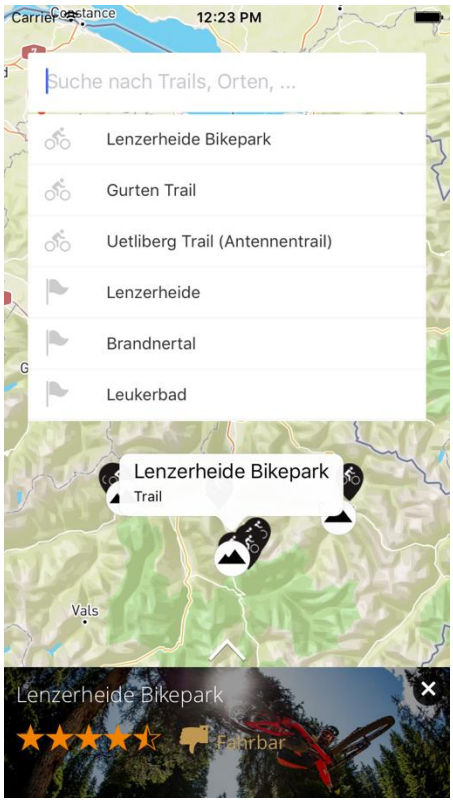

Abbildung 8: Anzeige des Suchverlaufs als Schnellzugriff

Die Suchresultate sind dabei keine eigene Ansicht, sondern werden über die Kartegelegt. Das gibt dem Nutzer ein stärkeres Gefühl der Kontrolle über die App, da für ihn klar ist, was passiert.

3.2.4 Karte

Die App hat Zugriff auf die aktuelle Nutzerposition. Sobald diese verfügbar ist, wird die Karte auf diese Koordinaten zentriert und die aktuelle Position markiert. Das dient als Orientierungshilfe für den Nutzer und zeigt ihm direkt Inhalte in der näheren Umgebung an. Die unterschiedlichen Daten auf der Karte werden durch unterschiedliche Marker differenziert dargestellt. So wird ein Shop mit Werkzeug, ein Trail mit einem Mountainbiker und eine Destination mit einer Bergkette symbolisiert.

Beim Klick auf einen Marker wird eine Textblase zum Marker geöffnet, um den aktiv ausgewählten Marker zu deklarieren. Ausserdem wird die erwähnte Preview am unteren Rand eingeblendet. Diese enthält die essentiellen Informationen über das ausgewählte Objekt.

3.2.5 Navigation

Für die Navigation der App wurden in der Konzeptionsphase einige Ideen verfolgt. Da aber klar war, dass im Rahmen dieser Arbeit lediglich die Bike Map umgesetzt werden kann, gab es zu wenige Inhalte für eine Off-Canvas oder Tab-Navigation. Daher musste auch ein einfacheres Navigationskonzept entworfen werden. Kartenobjekte werden einerseits in einem Overlay geöffnet anstatt auf einer eigenen Seite und sind andererseits über die Suche und die Marker selektierbar. Der einzige Seitenwechsel findet für die Profilansicht des Nutzers statt. Dafür wurde auf das Konzept eines Floating Action Buttons [5] gesetzt. Der Button kann zwei unterschiedliche Zustände haben.

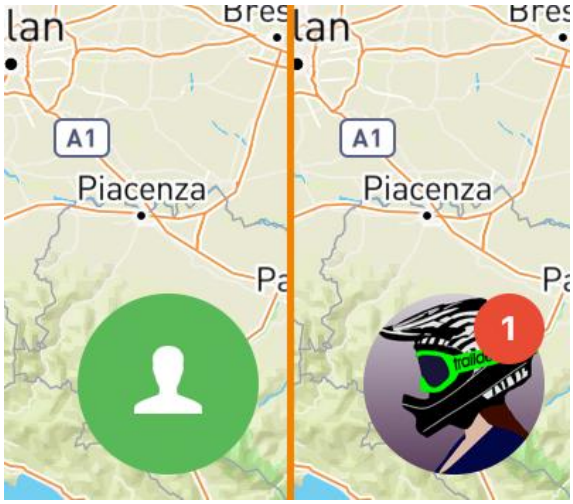

Abbildung 9: Die beiden möglichen Zustände des Floating Action Buttons

Ist der Nutzer nicht eingeloggt, wird ein Login-Icon im Button angezeigt und führt bei einem Klick auf die Registrierungsseite. Der zweite ist der eingeloggte Zustand. Dabei wird das Profilbild des Nutzers angezeigt, zusammen mit einem roten Hinweis bei ungelesen Benachrichtigungen. Bei einem Klick führt der Button nun zur Profilseite.

3.3 Architektur

React stellt kein vollständiges MVC- oder MVVM-Framework zur Verfügung, sondern bezeichnet sich selbst als View-Schicht in einer Model-View-Controller Architektur [6]. Das Entwicklungsteam muss die zu verwendende Applikationsarchitektur bewusst auswählen und umsetzen.

Für die Traildevils App wurde auf der UI-Schicht Flux mit Hilfe der Redux Bibliothek verwendet, was eine Alternative zu MVC darstellt. Für die Aufteilung der UI-Komponenten wurde die Atomic Design Methodologie verwendet, welche die Wiederverwendbarkeit von Komponenten erhöht.

Die UI-Schicht baut direkt auf einer Domainschicht auf, welche durch Domain-driven Design inspiriert ist. Auf diese verschiedenen Konzepte und einige Eigenheiten von JavaScript und dessen Ökosystem wird in den folgenden Kapiteln näher eingegangen.

3.3.1 Atomic Design

Atomic Design ist eine Methodologie zur Erstellung von Design Systemen [7]. Erstmals festgehalten wurde das Konzept von Brad Frost und fand schnell breite Unterstützung in der Frontendentwicklung.

Atomic Design ist nicht zwingend ein technologisches Prinzip, sondern stammt aus dem UI Design. Nichtsdestotrotz ist es empfehlenswert, das Prinzip durch alle Entwicklungsstufen und Ebenen durchzuziehen.

Das Prinzip schreibt vor, dass das User Interface vom kleinsten Element bis hin zum grössten entwickelt wird. So soll beispielsweise nicht von einer “Suchseite” aus gestartet werden, sondern von einem Button, einem Eingabefeld oder einem Textlabel. Diese kleinsten Einheiten nennen sich *Atome*. Diese sind kontextfrei, kennen also ihr Umfeld und ihre Funktion nicht.

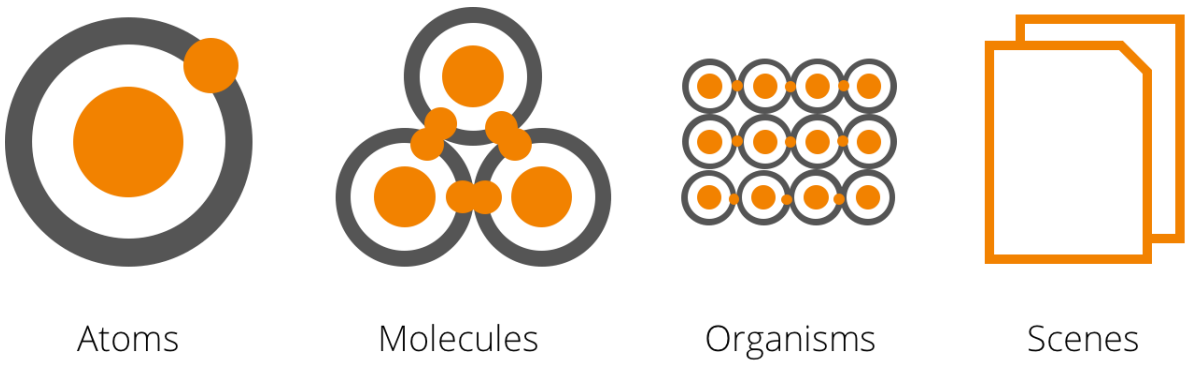

Abbildung 10: Aufbau eines Designsystems nach der Atomic Design Methodologie [8]

Werden mehrere Atome zusammengeführt, entstehen *Moleküle*. So entsteht beispielsweise aus einem Button, einem Eingabefeld und einem Textlabel eine Suchbox. Die Suchbox ist also eine Komposition aus verschiedenen, kleineren Einheiten.

Auch diese Moleküle bilden meist noch keine ganzheitlichen UI-Komponenten ab. Mehrere Moleküle verbinden sich zu einem *Organismus*. So wird aus einem Logo, einer Suchbox und einer Linkliste eine Header-Komponente. Ein Organismus bildet im Grunde den visuellen Kontext der UI-Komponente.

Verschiedene Organismen werden in *Templates* eingebunden, was den globalen Kontext widerspiegelt.

Wie bereits erwähnt, wird die Wiederverwendbarkeit erhöht. So kann ein Button auf verschiedenen Bereichen einer Seite verwendet werden, ohne dass er mehrfach designet oder entwickelt wurde.

3.3.2 Domain-driven Design

Die Domainschicht der Traildevils App wurde stark von Domain-driven Design beeinflusst. Domain-driven Design versucht Systeme oder Prozesse aus der realen Welt möglichst genau zu modellieren [9]. Dabei ist es wichtig, dass die Expertise der Domäne in der realen Welt korrekt verstanden wird. Zu diesem Zweck kann ein Domänenexperte beigezogen werden. Reale Objekte bilden die Basis für das Design der Software. So sind in der Traildevils App die zentralen Elemente Trails, Shops und Destinationen.

DDD bietet Unterstützung bei der Definition, wie die Objekte interagieren. Dafür werden die Domänenobjekte folgendermassen kategorisiert:

- **Werteobjekte** widerspiegeln Werte, die aber auch unterteilt werden können (beispielsweise kann ein Datum die Teile Tag, Monat und Jahr haben).
- **Entitäten** sind Objekte, die eine Identität haben. Eine Person ist z.B. eine Entität. So ist klar, dass zwei Personen mit demselben Namen *nicht* dieselbe Person sind.
- **Aggregate** sind Objekte, die andere Objekte enthalten. So macht zum Beispiel eine Zustandsmeldung keinen Sinn, ohne eine zugehörige Trail-Entität, welche bewertet wird.

Im Zusammenhang mit Domain-driven Design werden auch oft die folgenden Design Patterns verwendet:

- **Repositories** sind zuständig für die Persistenz (speichern, laden und suchen) der Daten. Ein Repository interagiert normalerweise mit einer Datenbank oder einem Webservice. Im Fall der Traildevils App kennen die Repositories allerdings nur den Gateway.
- **Factories** sind für das Erstellen von Domänenobjekten zuständig. Bekannt geworden ist das Factory Pattern durch die Gang of Four (GoF).
- **Services** sind Features, die Domänenobjekte verändern oder mit ihnen interagieren, ohne selbst Teil der Domäne zu sein.

In der Traildevils App wird zudem das **Gateway** Pattern verwendet. Dieses kapselt Zugriffe auf externe Systeme oder Ressourcen, in diesem spezifischen Fall die Zugriffe auf die Schnittstelle der Traildevils Plattform.

Bei Client-Server Architekturen kommt oft eine RESTful Architektur zum Einsatz, wobei JSON als Austauschformat verwendet wird. So auch im Fall der Traildevils App. Dabei ist der Server meistens die Komponente, welche die Datenhoheit behält. Richtet sich nun das Model nach der JSON-Struktur, läuft der Client Gefahr, dass im Falle einer Änderung in der Struktur der Client nicht mehr lauffähig ist. Genau diese Problematik untersucht die Publikation Model Your Application Domain, Not Your JSON Structures [10] der Universität Graz. Aus diesem Grund wurden für die Traildevils App bewusst die Domänenobjekte Trails, Shops und Destinationen unabhängig von der ausgelieferten JSON-Struktur entwickelt und entworfen. Die Domänenobjekte werden, wie oben bereits erwähnt, in Factories erstellt. Diese Factories kennen sowohl die Domänenobjekte als auch die JSON-Struktur, und wandeln diese einmal pro Aufruf um.

3.3.3 Event-Driven Architecture, Flux und Redux

Flux ist eine Applikationsarchitektur, die von Facebook verwendet wird, um Client-seitige Webapplikationen zu entwickeln. Durch die Verwendung eines unidirektionalen Datenflusses ist es eine optimale Ergänzung zu Reacts View-Komponenten [11].

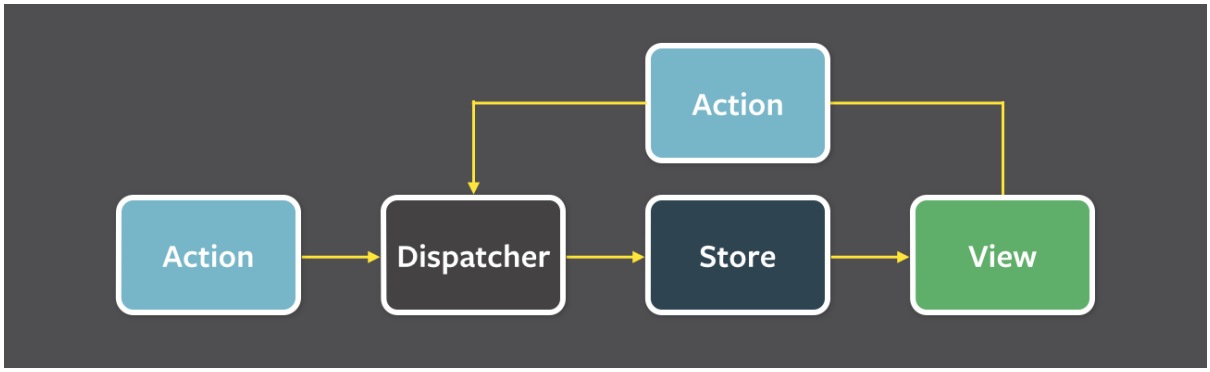

Abbildung 11: Unidirektionaler Datenfluss in einer Flux Architektur [12]

Das Architekturkonzept wird durch diverse Anbieter implementiert. Redux ist eine der populärsten Implementationen. Allerdings ist Redux nicht zwingend eine reine Flux-Implementation, sondern mischte Ansätze aus Flux und Elm [13].

Genau wie Flux konzentriert Redux die Aktualisierungslogik eines Models in einer bestimmten Schicht der Applikation (`stores` in Flux, `reducers` in Redux). Anstatt die Applikation direkt Daten mutieren zu lassen, werden diese Mutationen in Flux und Redux als einfach JavaScript Objekte beschrieben, genannt `actions`. In einem CQRS (Command Query Responsibility Segregation) Pattern kämen Actions wohl Commands gleich.

Eine gängige Beschreibung von Flux ist `(state, action) => state`. Nimmt man diese Beschreibung, ist Redux eine reine Implementation der Flux Architektur. Der Unterschied ist, dass anstelle eines Event-Dispatchers bei Redux lediglich reine Funktionen verwendet werden.

Redux basiert im Kern auf drei fundamentalen Prinzipien:

Single source of truth

Das Prinzip besagt, dass der Zustand der kompletten Applikation als Objektbaum in einem einzelnen Store gespeichert wird. Der Vorteil dieser «single source of truth» ist, dass Zustandsänderungen der Applikation atomar getrackt werden können.

State is read-only

Nach dem zweiten Prinzip gibt es nur einen Weg den Zustand zu ändern: eine `action` auszulösen, welche die Geschehnisse beschreibt. Dieses Prinzip gewährleistet, dass Views nur die Absicht zur Datenmutation mitteilen, den State aber nie direkt verändern.

Alle Datenmutationen finden zentral und in einer strikten Reihenfolge statt, daher kann es auch zu keinen Race Conditions kommen.

Changes are made with pure functions

Um zu beschreiben, wie sich der Zustand aufgrund ausgelöster Actions verändert, werden Reducer geschrieben. Reducer sind reine Funktionen, die den vorherigen Zustand und die ausgelöste Action entgegennehmen. Anstatt den mutierten Zustand zurückzugeben, wird ein neuer Zustand erzeugt.

3.3.4 Dependency Injection/Inversion of Control

Das Konzept von Dependency Injection und Inversion of Control ist sehr verbreitet. In etablierten Programmiersprachen wie Java, C# oder PHP deklariert eine Klasse ihre Abhängigkeiten als Annotations oder direkt in den Funktionsparametern. Ein IoC-Container kann diese z.B. via Reflection auflösen und die Abhängigkeiten direkt übergeben.

Dabei wird oft nicht gegen eine konkrete Implementation programmiert, sondern eine Abstraktion; also ein Interface. Die Konfiguration des IoC-Containers kann anhand verschiedener Faktoren entscheiden, welche konkrete Implementation verwendet werden soll.

JavaScript ist eine dynamisch typisierte Sprache und setzt auf prototypenbasierte Vererbung. Das Konzept von Interfaces ist in JavaScript unbekannt. Für die Aufteilung der Code-Einheiten und der Einhaltung von «Separation of Concerns» gibt es schon länger Bibliotheken wie CommonJS. ECMAScript 2015 bietet auch native Module Loader. Diese ermöglichen zwar das Deklarieren und das Laden von Abhängigkeiten, allerdings werden dabei immer konkrete Implementationen verlangt und bieten keine Möglichkeit zu inverser Kontrolle.

In der JavaScript-Welt wird oft argumentiert, dass aufgrund der interpretierten Natur der Sprache Dependency Injection nicht benötigt wird, da diese nur Schwächen von kompilierten Sprachen entgegentritt.

Dass dies nicht unumstritten ist, wird dadurch gezeigt, dass die meisten bekannten JavaScript-Frameworks einen Dependency Injection-Container zur Verfügung stellen, so

z.B. auch das von Google entwickelte Framework Angular. Dieses erlaubt es zwar Abhängigkeiten deklarativ zu definieren, es werden dabei aber immer die konkreten Implementationen verlangt. Das kann unter anderem bei der Generierung von Mocks in einem Testing-Umfeld zu Schwierigkeiten führen.

Die Thematik um Dependency Injection und Inversion of Control in JavaScript ist extrem gross und würde wohl den Umfang einer eigenen Bachelorarbeit rechtfertigen.

3.3.5 Babel

ECMAScript, oder kurz ES, ist eine Standardisierung der Sprachspezifikation von JavaScript. Diese Spezifikation wird laufend durch Ecma International erweitert und aktualisiert. Die konkrete Implementation der Spezifikation ist den Herstellern von JavaScript Interpretern überlassen. Im Juni 2015 wurde die sechste Version des Standards mit der offiziellen Bezeichnung ECMA-262, oder umgänglich ES2015 (ECMAScript 2015 Language Specification), abgenommen. Die neue Spezifikation bietet gegenüber den Vorgängern zahlreiche Vorteile wie Module Loading, das `class` Keyword oder Arrow Functions (ähnlich Lambdas in Java 8).

Noch wird diese Spezifikation nicht von allen Herstellern vollumfänglich unterstützt. Die Vorteile der neuen Version sind aber so gross, dass in grossen Projekten dennoch bereits voll daraufgesetzt wird. Um die Kompatibilität zu älteren Plattformen gewährleisten zu können, wird oft Babel verwendet. Babel ist ein JavaScript Source-to-Source Compiler, welcher ES2015 zu ES5 Syntax umwandelt. Dieser Schritt passiert nicht zur Laufzeit; es wird effektiv ein zusätzlicher Kompilationsschritt eingeführt, was sonst bei JavaScript nicht nötig ist. Babel hat sich in der Webindustrie zum de facto Standard für die Verwendung von ES2015 entwickelt. Anhand einer Konfigurationsdatei kann Babel sehr granular konfiguriert werden, welche ES2015 Merkmale umgewandelt werden sollen. Oft reicht aber die Verwendung eines “Presets” aus, welches eine sinnvolle Vorauswahl trifft. Im Rahmen dieser Arbeit wird das react-native Preset verwendet, welches von Facebook empfohlen und entwickelt wird.

3.3.6 Node und npm

Grossen Anteil an dem Popularitätsanstieg von JavaScript dürfte Node.js haben. Node.js ist eine JavaScript Laufzeitumgebung auf Basis von Chrome's V8 JavaScript Engine und erlaubt es JavaScript auch ausserhalb eines Browsers, z.B. in einem Server-Prozess, auszuführen. Zusammen mit Node wird auch der Node Package Manager, kurz npm, ausgeliefert. Das npm Register stellt über 250'000 wiederverwendbare JavaScript Packages zur Verfügung. Auch in dieser Arbeit kamen solche Pakete zum Einsatz und wurden mittels npm installiert und verwaltet.

Während npm es erlaubt Pakete sehr schnell und eifach zu installieren, bringt es auch Nachteile mit sich. So sind Pakete nicht nur innerhalb eines Vendor-Namespaces, sondern global gültig, was zu Namenskollisionen führen kann. Auch kann es durch vage Angaben der erforderlichen Paket-Versionen (oder Nichteinhaltung von SemVer) zu Korruption der eigenen Applikation kommen, wenn die Pakete aktualisiert werden.

In so einem Fall kann das Paket auf GitHub geforkt und somit eine eigene Kopie verwendet werden. Die Probleme können dann behoben und die Korrekturen via Pull Requests zurück an den ursprünglichen Entwickler gespielt werden. Um die Zeit zu überbrücken, in welcher das Problem zwar gefunden, aber noch nicht im offiziellen Paket behoben wurde, kann im Projekt die eigene Version des Pakets referenziert werden.

Im Rahmen dieser Arbeit wurden die Projekte `mirco0101/react-native-star-rating` und `deniaz/react-native-floating-text-input-labels` geforkt.

3.4 Kartenmaterial

Aufgrund der intensiven Nutzung von Kartenmaterial in der App haben diese direkten Einfluss auf die Funktionalität, die mit der App angeboten werden kann.

Zu Beginn der Projektarbeit war geplant, die Karte mit nativen Kartenmaterial (Apple Maps auf iOS, Google Maps auf Android) darzustellen. Um die Machbarkeit sicherzustellen, wurde in der Technologie-Evaluation die Kartenkomponente von React Native sowie eine Third-Party Komponente analysiert.

Des Weiteren wurde versucht, Karten in einer WebView mit der bekannten Leaflet.js-Bibliothek zu implementieren. Leaflet wird bereits auf der Traildevils Webseite verwendet. Aufgrund der sehr schlechten Performance der WebView wurde dieser Versuch aber wiederl abgebrochen.

Während der Entwicklung traf der Praxispartner, die Entscheidung voll auf OpenStreetMap-Karten von Mapbox zu setzen. Diese Umstellung brachte einige Funktions- und Performanceeinbussen mit sich. Der Vorteil von Mapbox ist allerdings die Qualität der Outdoor-Karten und die zukünftige Möglichkeit, Kartenmaterial offline speichern zu können.

3.4.1 Mapbox Karten (Third Party Komponente)

Die Vektorkarten von Mapbox, basierend auf Open Street Map, sind über die Third Party Komponente `react-native-mapbox-gl` [14] in React Native verfügbar. Die Komponente bildet dabei einen Wrapper um die nativen SDKs von Mapbox auf Android und iOS, der die Schnittstellen als JavaScript APIs exponiert.

Kartenmaterial	Qualitativ gute Outdoorkarten mit Höhenlinien dafür keine Satellitenbilder.
Performance	Gute Performance auch bei vielen Markern. Werden geladene Marker bearbeitet, bricht die Performance ein.
Clustering	Noch nicht verfügbar, aber in aktiver Entwicklung. Kann auch nicht durch den reinen Clusteralgorithmus `supercluster` [15] kompensiert werden, aufgrund der oben beschriebenen Performance Probleme.
GPS Tracks	Unterstützung für GPS Tracks ist gegeben, kann jedoch auch zu Perfomance Problemen führen, wenn Tracks dynamisch nachgeladen werden.
Offline Karten	Im Verlaufe dieser Arbeit wurde eine API freigegeben, welche die offline Speicherung einzelner Regionen erlaubt.
Lizenz/Kosten	Gratis bis zu 50'000 Kartenzugriffen, danach kostenpflichtig mit verschiedenen Abo Modellen [16].

Tabelle 6: Analyse der Mapbox Karten

3.4.2 Native Karten (React Native Komponente)

React Native stellt von Haus aus die Komponente `MapView` [17] zur Verfügung, um die nativen Karten von iOS (Apple Maps) und Android (Google Maps) zu nutzen. Sie bietet

jedoch nur sehr rudimentäre Funktionalität und es wird auch von Facebook selbst empfohlen, für kartenintensive Anwendungen eine Community Komponente zu verwenden (Kapitel 0).

3.4.3 Native Karten (Third Party Komponente)

Mit der Komponente `react-native-maps` [18] hat die React Native Community die `MapView` Komponente (Kapitel 3.4.2) um sehr viele Funktionen erweitert.

Kartenmaterial	Gewohnt hochwertiges Kartenmaterial von Google und Apple Maps.
Performance	Sehr gute Performance, die man sich von nativen Karten gewohnt ist. Performanceprobleme können jedoch bei einer sehr grossen Anzahl Marker auftreten.
Clustering	Nicht verfügbar, könnte aber z.B. durch die Komponente `supercluster` [15] kompensiert werden, da die nativen Karten keine Performanceprobleme beim Verändern von Markern haben.
GPS Tracks	Unterstützung für GPS Tracks ist gegeben.
Offline Karten	Keine Möglichkeiten vorhanden.
Lizenz/Kosten	Gratis

Tabelle 7: Analyse Third Party Komponente für native Karten

3.5 API

Die serverseitige Umsetzung der API für den Zugriff der App auf die Daten von Traildevils, war nicht Bestandteil dieser Arbeit. Diese Umsetzung ist Sache des Auftraggebers und basiert auf der in dieser Arbeit erarbeiteten API Definition (Anhang **Fehler! Verweisquelle konnte nicht gefunden werden.**). Die API Definition wurde während der Entwicklung laufend vom Team spezifiziert, vom Auftraggeber kontrolliert und abgenommen.

Um die fehlende Schnittstellenimplementierung der Traildevils Infrastruktur zu kompensieren, wurde ein dreiteiliges Mocking Konzept erarbeitet.

3.5.1 Apiary

Mit Apiary wurde ein standardisiertes Online-Tool zur Spezifikation und gleichzeitigem Mocking von APIs genutzt (Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.**). Aufgrund der Spezifikation stellt das Tool automatisch einen Mock-Server zur Verfügung. Dieser Mock-Server wurde vor allem zu Beginn der Arbeit genutzt, um schnell statische Daten mit der App abrufen zu können.

3.5.2 Traildevils API

Der Auftraggeber hat bereits während der Arbeit mit der Umsetzung von einzelnen Endpunkten der Schnittstelle begonnen. Diese Endpunkte wurden, wenn möglich, direkt angesprochen.

3.5.3 Ghostrider Proxy

Um den Apiary-Mock sowie die bereits verfügbare Traildevils API zu verbinden und der App einen zentralen Endpoint zu bieten, wurde der «Ghostrider Proxy» erstellt. Es handelt sich dabei um eine Eigenentwicklung in JavaScript, welche die API Zugriffe der App entgegennimmt und je nach Konfiguration an Apiary oder Traildevils weiterleitet. Durch diesen zwischengeschalteten Proxy konnte stehts zwischen Live-Daten und Mock gewechselt werden, ohne das Anpassungen an der App nötig waren. Eine Instanz des Proxys lief während der Entwicklungsphase auf einem gratis Container von Heroku.

Ein weiterer Nutzen des Proxys war die Möglichkeit für «intelligentes Mocking». Schnittstellen mit unterschiedlichen oder dynamischen Rückgabewerten, wie z.B. ein Login oder die Bewertung eines Objektes, konnten nicht über den statischen Apiary-Mock gemacht werden. In solchen Fällen verarbeitet der Proxy die API Zugriffe direkt ohne Kommunikation zu anderen Systemen.

Ghostrider wurde mit Web Framework Express entwickelt und ist auf GitHub unter https://github.com/deniaz/ghostrider veröffentlicht.

4 Ergebnisse

4.1 Vergleich Vision und Ergebnisse

Die Interaktion zwischen Nutzer und der Traildevils Plattform sollte auf einem mobilen Endgerät einfacher gestaltet werden, um die Nutzeraktivität zu erhöhen. Mit der App wurde der Grundstein gelegt, genau das zu erreichen. Es ist sehr einfach und schnell möglich, für Nutzer Trails zu finden und diese mit Inhalten wie Zustandsmeldungen oder Bewertungen anzureichern. Diese Prozesse sind gegenüber dem Stand der Plattform im Februar 2016 deutlich optimiert. Im Idealfall kann ein Trail neu mit fünf Klicks bewertet werden, was nur einige Sekunden in Anspruch nimmt. Die Wahrscheinlichkeit, dass diese Daten von unterwegs erfasst werden, hat sich somit erhöht.

Personalisierte und geobasierte Inhalte wurden im Rahmen dieser Arbeit zwar konzipiert, aber nicht umgesetzt. Im Usability Test (Kapitel 4.2.4) wurde gezeigt, dass personalisierte Funktionalität erwünscht ist. Bereits absolvierte Trails, massgeschneiderte Vorschläge und Top-Inhalte stehen dabei zu Oberst auf dem Wunschzettel. Anstelle personalisierter Inhalte wurde sehr viel Aufwand in die Kartenunterstützung investiert. So wurde im zweiten Code Sprint auf Mapbox umgestellt. Viel Aufwand wurde in Thematiken wie Offlinespeicherung, Clustering oder Tracks investiert, ohne dass diese Themen im Prototyp schlussendlich implementiert wurden. Ebenfalls wurde viel Zeit investiert, das Design ansprechend zu gestalten. Viele Elemente durchliefen mehrere Designiterationen bis das Team und der Auftraggeber zufrieden waren.

Der entwickelte Prototyp zusammen mit den erarbeiteten Konzepten bieten eine solide Basis für die in der Vision festgehaltenen Funktionalitäten.

4.2 App

4.2.1 Umfang

Im Rahmen dieser Arbeit wurden die User Journeys «Trails in Davos» (Kapitel 2.2.1), «Gestauchte Felge» (Kapitel 2.2.2) und «Bewerten eines gefahrenen Trails» (Kapitel 2.2.3) implementiert. Die dritte Journey unterscheidet sich zwischen Vision und Implementation in der fehlenden Integration von Strava und Push Benachrichtigungen. Beide Funktionalitäten erfordern starke Zusammenarbeit und hohe Abstimmungsaufwände zwischen der App und dem Traildevils Server.

Zusätzlich zu diesen drei User Journeys wurde sehr viel Zeit in Kartenfunktionalität investiert, welche in diesem Ausmass nicht Bestandteil der definierten und abgenommenen User Journeys war.

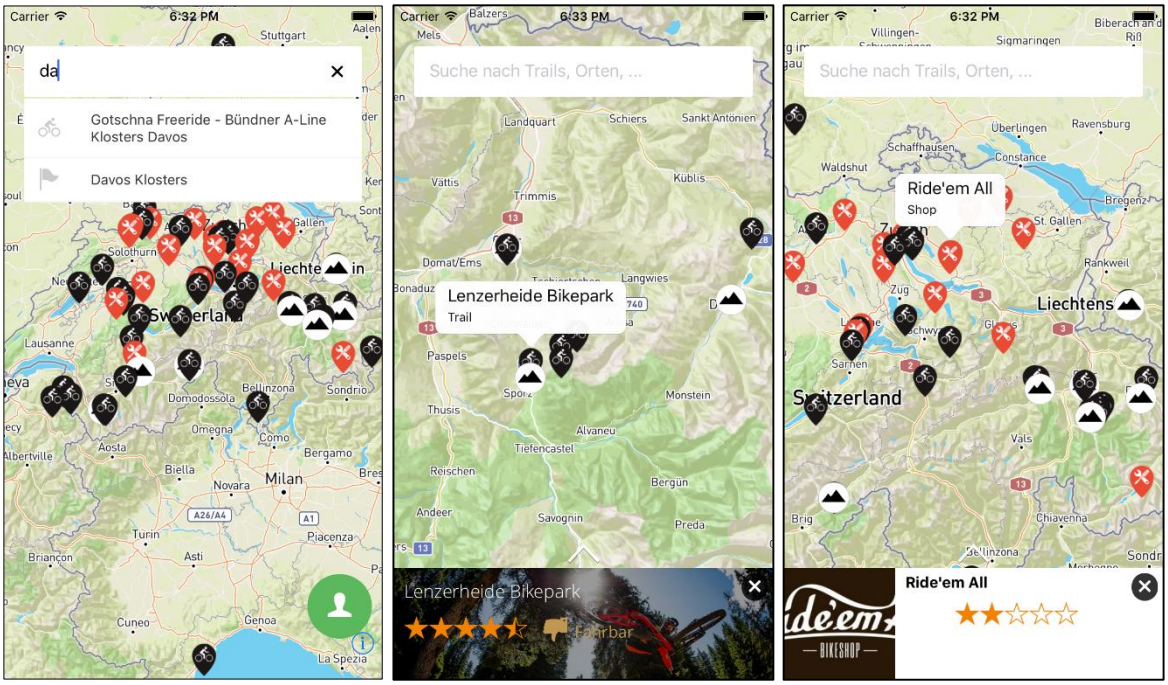

Abbildung 12: App-Screenshots der Kartenansicht

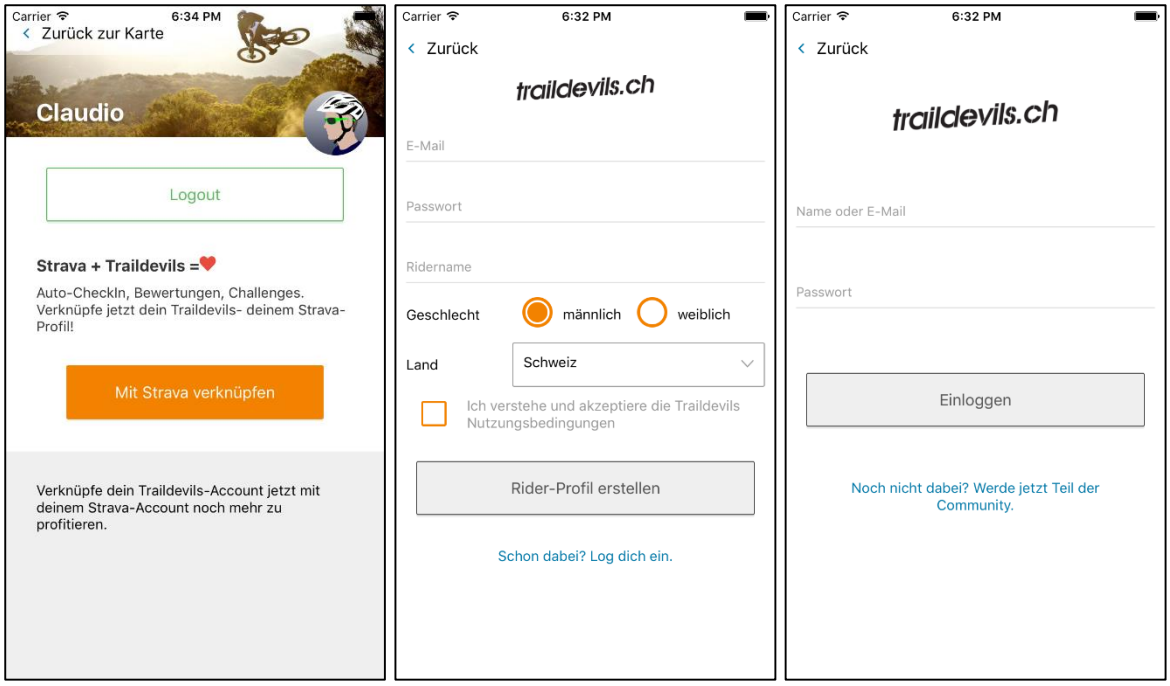

Abbildung 13: App-Screenshots der Account Aktivitäten

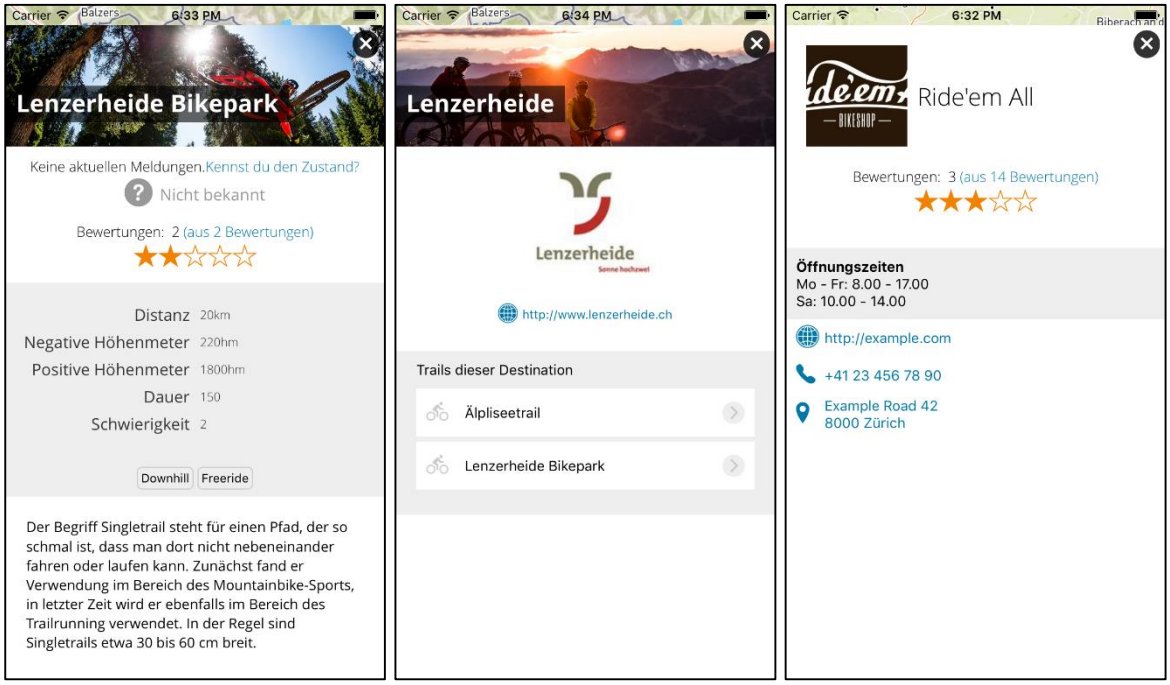

Abbildung 14: App-Screenshots der unterschiedlichen Detailansichten

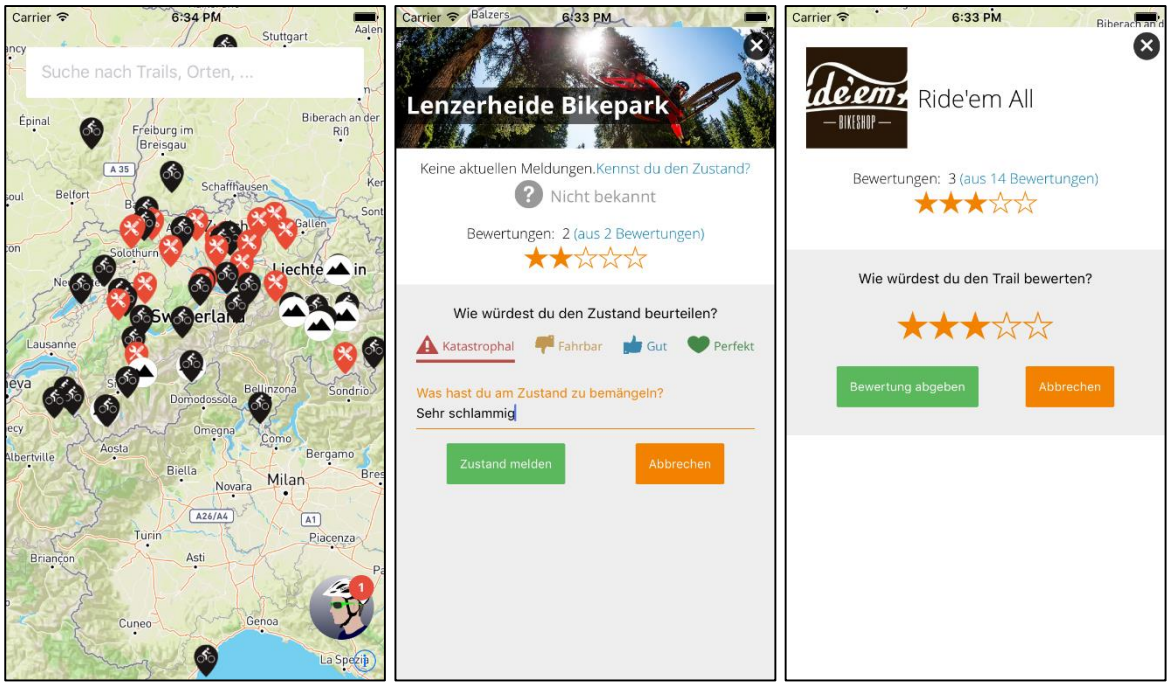

Abbildung 15: App-Screenshots der Möglichkeiten für eingeloggte Nutzer

4.2.2 Projektstruktur

Im Wurzelverzeichnis des Projekts sind folgende Dateien und Verzeichnisse zu finden:

```
android/
docs/
index.android.js
index.ios.js
ios/
src/
```

Die Verzeichnisse `android` und `ios` enthalten die Objective-C und Java Bindings für React Native. Darin wurden vom Projektteam lediglich Projektnamen, App Icons und Buildpläne verändert. Das Verzeichnis `docs` enthält User Manuals, die beim Setup und der Weiterentwicklung der App unterstützen. Des Weiteren ist die generierte API Dokumentation ebenfalls unter `docs` abgelegt.

Die beiden Dateien `index.android.js` und `index.ios.js` bilden die Einstiegspunkte für die plattformspezifischen Builds. Da im Falle von Traildevils keine Unterschiede für die Plattformen existieren, wird die App in `src/app.js` initialisiert und in den beiden index-Dateien eingebunden.

Die gesamte Codebasis des entwickelten Prototyps ist unter `src` gespeichert. Die Verzeichnisstruktur unter `src` sieht folgendermassen aus:

```
app.js
domain/
       collections/
       entities/
       factories/
       repositories/
       services/
infrastructure/
ui/
       actions/
       components/
       containers/
       reducers/
```

Diese Struktur wiederspiegelt den Schichtenaufbau des Prototyps. Code Einheiten aus der Domänenschicht liegen somit unter `src/domain`, UI-relevante Einheiten unter `src/ui`.

4.2.3 Codestatistik

Da ES2015 noch ein sehr junger Standard ist, gibt es fast keine Tools die gute Metriken errechnen. Für die Überprüfung der Code Qualität und das Auffinden von Code Smells wurden die Tools bitHound [19] und Scrutinizer [20] verwendet. Beide Tools verwenden einen Algorithmus und diverse Metriken (Komplexität, Kopplung, Kohäsion, etc.) zu einem Index zusammenzufassen. Die Skala von bitHound geht von 0 – 100 Punkte, diejenige von Scrutinizer von 0 bis 10.

bitHound	97 (very good)
Scrutinizer	9.72 (very good)

Tabelle 8: Messergebnisse zur Code Qualität

Diese Bewertungen zeigen, dass die Code Qualität hoch ist und wenig Technical Debt über die Projektdauer aufgebaut wurde.

Leider werden bei beiden Tools keine Werte für «Lines of Code» und «Number of Modules» berechnet. Daher wurden diese Metriken mit Hilfe von Bash-Skripts manuell berechnet. Die Anzahl der geschriebenen Codezeilen mittels `wc –w `find src –name *.js –print``. Die Anzahl Module ist äquivalent zur Anzahl aller Dateien, da jede JavaScript-Datei nur ein JavaScript-Modul enthält. Daher wurde dies mittels `find src –type f | wc –l` ermittelt.

Lines Of Code	16758
Number of Modules	126

Tabelle 9: Codestatistiken

Für den Entwicklungsprozess ebenfalls spannend, sind Auszüge aus dem Git-Repository.

Commits	449
Pull Requests	35

Tabelle 10: Git Statistiken

4.2.4 Testabdeckung

Für die Domainschicht der Traildevils App wurde zahlreiche Unit Tests geschrieben. Das verwendete Testing Framework Jest erlaubt es mit dem Befehl `jest --coverage` direkt die Testabdeckung generieren zu lassen.

Statements	Branches	Functions	Lines
92.46 %	74.39 %	93.94 %	96.02 %

Tabelle 11: Analyse der Testabdeckung

Mit Ausnahme der Codezweige ist die Abdeckung relativ hoch. Die tiefe Abdeckung bei Codezweigen kann beispielsweise dann auftreten, wenn in einer `if-else`-Condition lediglich ein Branch geprüft wird.

4.3 Usability Test

Um die mit dem Prototyp erarbeiteten Ergebnisse zu validieren, wurde der Prototyp einem Usability Test unterzogen. Dieser wurde mit vier Probanden durchgeführt. Drei Probanden sind iOS Nutzer, der vierte Android Nutzer. Alle vier sind, in unterschiedlichem Masse, am Mountainbikesport interessiert und involviert. Alle vier sind in der Altersgruppe zwischen 25 und 35 Jahren und gehören damit zur Zielgruppe von Traildevils und speziell der Traildevils App.

Abbildung 16: Impressionen aus dem Usability Test

Getestet wurden drei Key-Szenarien, mit welchen zu Beginn der Arbeit die Grundfunktionalität der App definiert wurde. Die gewählten Szenarien decken einen möglichst grossen Teil der umgesetzten App Funktionalität ab.

Das detaillierte Test Script ist im Anhang (Anhang B.1) zu finden.

Nachfolgend sind die wichtigsten Erkenntnisse, welche aus den verschiedenen Tests gewonnen werden konnten, zusammengefasst.

Die detaillierten Notizen zu jedem Test sind im Anhang (Kapitel A) zu finden.

Gute Cross-Plattform User Experience

Alle Tester, sowohl iOS oder Android Nutzer, waren in der Lage das grundsätzliche Bedienkonzept der App schnell zu erfassen und trafen keine plattformspezifischen Hürden an. Erwähnt wurde spezifisch die Einfachheit und Klarheit des User Interfaces.

Schlechte «Klickbarkeit»

Bei allen Testern zeigten sich Probleme beim Anwählen von Objekten in der App mit dem Finger. Die Objekte sollten grösser sein und/oder eine grössere Touchfläche bieten.

Irritation durch Mapbox Infobutton

Der blaue Infobutton im unteren Teil der Karte, der für die Akkreditierung von Mapbox und Open Street Map benutzt wird, verwirrte die Tester. Sie alle erwarteten weitere Kartenfunktionalität dort zu finden. Die Akkreditierung sollte an einem unauffälligeren Ort untergebracht werden wie z.B. eine Einstellungs- oder Impressumsseite, zusammen mit Informationen über die Datenherkunft (Traildaten von traildevils.ch, etc.).

Banner Blindness bei Previews

Die Previews, welche beim Klick auf einen Marker erscheinen, wurden von den meisten Testern im ersten Moment übersehen oder bewusst nicht angeklickt. Das traf insbesondere bei Previews zu, welche ein Vollbild als Hintergrund haben. Der Begriff Werbung wurde dabei explizit erwähnt.

GPS Tracks auf Karte gefragt

Eine Mehrheit der Tester hat vergeblich versucht, durch Zoomen auf der Karte den Trailverlauf ansehen zu können. Das Bedürfnis nach dieser visuellen Information ist klar vorhanden, da sie visuell schnell Informationen über die Distanz und Dauer des Trails geben.

Zu wenig Community Informationen

Die Tester fanden die Bewertungen und Zustandsmeldungen sehr nützlich, vermissten allerdings Detailinfos dazu wie Kommentartexte und Bilder. Dabei wurde hervorgehoben, dass vor allem der Konsum dieser Informationen interessant ist. Die Möglichkeit zur Abgabe dieser Informationen wird zwar erwartet, würde aber nur wenig genutzt werden.

Destinationen werden nicht beachtet

Die Detailseite der Destinationen wurde von fast keinem Tester während des Tests geöffnet. Die Destinationen wurden nur zu Navigationszwecken in der Suche genutzt. Die Trails wurden dann ausschliesslich über die Marker auf der Karte aufgerufen.

App ist zu passiv

Die Tester äusserten auf unterschiedliche Arten den Wunsch, aktiver Informationen von der App zu erhalten. Als Möglichkeiten wurden Personalisierungsfunktionen wie z.B. das Vorschlagen von Trails aufgrund des Aufenthaltsortes oder bereits gefahrener Trails genannt. Eine weitere Möglichkeit ist das Anbieten von Trail-Stories, die den Nutzer zum Ausprobieren eines neuen Trails animieren könnten.

5 Weiterentwicklungsmöglichkeiten

Im Laufe dieser Arbeit, vor allem während der Konzeptionsphase und der Umsetzung des Prototyps, wurden viele mögliche Ideen zur Weiterentwicklung der App entwickelt und dokumentiert.

Auf die wichtigsten und für den Auftraggeber interessantesten wird in den folgenden Kapiteln eingegangen.

5.1 Clustering

Die wohl grösste Einschränkung des Prototyps ist die Anzeige vieler Marker auf der Karte. Dies ist kein Performance-Problem, sondern ein visuelles; die grosse Anzahl Marker überfordert den Nutzer und schränkt die Übersicht ein. Ein bekanntes Pattern dagegen ist das

Clustering von Markern. Mehrere naheliegende Marker in einer Region werden zusammengefasst und zum Beispiel mit einem farbigen Kreis markiert. Beim Heranzoomen werden die einzelnen Marker angezeigt und nicht mehr die Cluster.

Aktuell ist eine Implementation für die nativen SDKs geplant, welche auch zeitnah im React Native Package eingebunden werden sollen. Laut dem Maintainer des Pakets ist der Release auf Q3 2016 geplant. Die Integration in den Traildevils Prototypen dürfte dann mit geringem Aufwand machbar sein.

5.2 Offline Speicherung

In den aktuellen Versionen des Mapbox Pakets ist es möglich, Kartenabschnitte offline zu speichern. Ohne einen zahlungspflichtigen Mapbox Account können bis zu 6000 Kacheln offline gespeichert werden. Das entspricht ungefähr dem Grossraum London, innerhalb der M25 Autobahn in den Zoomstufen 0 - 15.

Eine Offlinespeicherung der Karten macht nur Sinn in Kombination mit Offlinespeicherung der Traildevils Daten. Aktuell werden alle von der Traildevils-API abgefragten Daten in einem `EntityStore` zu Cachingzwecken gespeichert. Dieser Speicher ist lediglich im Arbeitsspeicher und daher nicht persistent. Dieser Store wäre eine Möglichkeit, um Offlinespeicherung zu implementieren. React Native bietet beispielsweise den `AsyncStorage`, welcher anstelle des Arbeitsspeichers als Basis des `EntityStores` dienen könnte.

Wichtig dabei ist es allerdings, eine gute Datensynchronisation zu konzipieren und zu implementieren. Zum Beispiel ist das Invalidieren von solchen Daten keine triviale Angelegenheit und sollte sauber geplant sein.

5.3 Community

In der Konzeptionsphase und erneut auch in den Usability Tests hat sich gezeigt, dass eine Stärke von Traildevils die Community und die Nutzer-generierten Inhalte sind. Dieser Effekt könnte über die App gestärkt werden.

Die Möglichkeit Kommentare und Fotos auf der Plattform zu teilen, stärkt die Benutzeraktivität auf der Plattform. Mit Benachrichtigungen wie «Claudio hat dein Foto kommentiert» könnte auch das Zusammengehörigkeitsgefühl gestärkt und ein Austausch zwischen den Nutzern etabliert werden.

5.4 Strava

Strava ist eine fest etablierte App im Mountainbike-Umfeld. Da Nutzer bereits Daten auf Strava speichern, welche per API zur Verfügung gestellt werden, ist eine Integration der Schnittstelle für Traildevils sehr attraktiv.

So kann ein Nutzer aufgefordert werden, gefahrene Trails auf Traildevils zu bewerten oder die bereits absolvierten Trails aufzulisten (Bucket-List). Ein weiterer Vorteil der Integration kann die Personalisierung von Traildevils sein. Zusammen mit den Nutzerinformationen von Strava können spezifische Routentipps abgegeben werden.

5.5 Push Benachrichtigung

Push Benachrichtigungen ergänzen die beiden vorherigen Verbesserungsmöglichkeiten der Community Features und der Strava Integration.

Ein Nutzer der App könnte per Push Benachrichtigung über Aktivitäten seiner Traildevils-Freunde informiert werden. Durch die Strava API können vom Nutzer gefahrene Trails quasi in Echtzeit registriert werden. Auf ein solches Ereignis hin, kann ein Nutzer aktiv zur Bewertung des Trails aufgefordert werden.

5.6 Feedback aus Usability Test

Aus den Resultaten des Usability Tests können weitere Verbesserungsmöglichkeiten abgeleitet werden.

Am Prototyp optimiert werden können die Klickbarkeit der Elemente und das Hinzufügen einer eigenen Impressums-/Informationsseite mit Informationen über die Mapboxkarte und die Herkunft der Traildevils Daten. Ohne Hürden können auch Versuche unternommen werden, die Bannerblindness der Preview-Elemente zu überwinden.

Die Integration der GPS Tracks und weiterer Community Inhalte wie Fotos und Kommentare sind ebenfalls sehr attraktiv, bedürfen aber einiges an Absprache mit der Traildevils-Schnittstelle.

6 Verzeichnisse

6.1 Abbildungsverzeichnis

6.2 Tabellenverzeichnis

6.3 Quellenverzeichnis

[1] Frontline Media GmbH, «frontline-media.ch,» [Online]. Available: http://www.frontline-media.ch/traildevils/. [Zugriff am 24 02 2016].

[2] A. Klement, «Replacing The User Story With The Job Story,» 12 11 2013. [Online]. Available: https://jtbd.info/replacing-the-user-story-with-the-job-story-af7cdee10c27#.p1bxdxudw. [Zugriff am 10 06 2016].

[3] D. Panchal, «What is Definition of Done (DoD)?,» 08 09 2008. [Online]. Available: https://www.scrumalliance.org/community/articles/2008/september/what-is-definition-of-done-(dod). [Zugriff am 10 06 2016].

[4] S. A. D. N. A. Daniel Demel, Interviewee, *Experten Interview Paper Prototype.* [Interview]. 30 03 2016.

[5] Google Inc., «Google Material,» Google Inc., [Online]. Available: https://material.google.com/components/buttons-floating-action-button.html. [Zugriff am 11 06 2016].

[6] Facebook's React Team, «React,» Facebook Inc., [Online]. Available: https://facebook.github.io/react/docs/why-react.html. [Zugriff am 15 06 2016].

[7] B. Frost, «bradfrost.com,» 06 10 2013. [Online]. Available: http://bradfrost.com/blog/post/atomic-web-design/. [Zugriff am 10 06 2016].

[8] B. Frost, «Atomic Design,» Juni 2016. [Online]. Available: http://atomicdesign.bradfrost.com/table-of-contents/.

[9] D. Haywood, «Methods & Tools,» [Online]. Available: http://www.methodsandtools.com/archive/archive.php?id=97. [Zugriff am 10 06 2016].

[10] M. Lanthaler und C. Gütl, «Model Your Application Domain, Not Your JSON Structures,» ACM, New York, Graz, 2013.

[11] Facebook React Team, «Flux Documentation,» [Online]. Available: https://facebook.github.io/flux/docs/overview.html. [Zugriff am 10 06 2016].

[12] Facebook, «Flux Docs,» Facebook, Juni 2016. [Online]. Available: https://facebook.github.io/flux/docs/overview.html#content.

[13] Redux, «Redux,» Redux, Juni 2016. [Online]. Available: http://redux.js.org/docs/introduction/PriorArt.html.

[14] Mapbox, «Github,» Juni 2016. [Online]. Available: https://github.com/mapbox/react-native-mapbox-gl.

[15] Mapbox, «Github,» Juni 2016. [Online]. Available: https://github.com/mapbox/supercluster.

[16] Mapbox, «Mapbox Abos,» 10 Juni 2016. [Online]. Available: https://www.mapbox.com/pricing/.

[17] Facebook, «Facebook Docs,» Juni 2016. [Online]. Available: https://facebook.github.io/react-native/docs/mapview.html.

[18] lelandrichardson, «Github,» Juni 2016. [Online]. Available: https://github.com/lelandrichardson/react-native-maps.

[19] bitHound Inc., [Online]. Available: https://www.bithound.io/. [Zugriff am 12 06 2016].

[20] Scrutinizer GmbH, «Scrutinizer,» [Online]. Available: https://scrutinizer-ci.com/. [Zugriff am 12 06 2016].

[21] Facebook, «Facebook Jest,» Facebook, Juni 2016. [Online]. Available: https://facebook.github.io/jest/.

[22] Facebook, «flowtype,» Facebook, Juni 2016. [Online]. Available: https://flowtype.org/.

[23] j. Foundation, «ESLint,» Open Source, Juni 2016. [Online]. Available: http://eslint.org/.

[24] a. blueprint, «api blueprint,» Juni 2016. [Online]. Available: https://apiblueprint.org/.

7 Glossar

Begriff	Erklärung	Mehr
Annotation	Annotationen sind syntaktische Metadaten in Quellcode. Oft mit dem Symbol @ als Präfix verwendet.	bit.ly/1PFgtGE
API Blueprint	API Blueprint ist eine Beschreibungssprache für Web APIs.	bit.ly/1Sf4uiz
Atomic Design	Methodologie für die Entwicklung von Design Systemen.	bit.ly/1AfjRtS
CommonJS	Definiert JavaScript Module und unterstützt bei der Definition und dem Laden von diesen.	bit.ly/1XBPnn1
CQRS	Command Query Responsibility Segregation ist ein Pattern, wobei für das Lesen und das Mutieren von Daten unterschiedliche Models verwendet.	bit.ly/1gDewIc
DDD	Domain-driven Design ist ein Ansatz der Software Entwicklung um Reale Systeme zu modellieren.	3.3.2
ECMAScript	ES ist eine Scriptsprache die von Ecma International standardisiert wird.	bit.ly/1NFZhPg
Elm	Elm ist eine funktionale Sprache zur deklarativen Entwicklung von Browser UIs.	bit.ly/2206hOF
Flux	Flux ist eine Applikationsarchitektur von Facebook zur Entwicklung von User Interfaces.	bit.ly/1Fnn2nf
Gang of Four (GoF)	Die Autoren des Software Engineering Buches «Design Patterns: Elements of Reusable Object-Oriented Software»	bit.ly/1svqPPa
Inversion of Control (IoC)	Das Auflösen von Abhängigkeiten wird von einem externen Service übernommen. Auch Hollywood Principle genannt.	bit.ly/1pCY23A
MVC	Model-View-Controller ist ein Pattern zur Entwicklung von User Interfaces.	bit.ly/1trYiVq
MVVM	Model-View-ViewModel ist eine Variation von MVC.	bit.ly/1VJ0oOi
OpenStreetMap	Kartenmaterial, welches unter offenen Lizenzen frei verfügbar ist.	bit.ly/1TUasCx
Race Condition	Wenn zwei Operationen in einer bestimmten Reihenfolge ablaufen sollen, diese aber nicht eingehalten wird und somit Inkonsistenzen entstehen.	bit.ly/1igqF4c

Reine Funktionen (Pure Functions)	Ansatz aus der funktionalen Programmierung. Der Rückgabewert einer Funktion wird alleine durch den Input definiert und hat keine Seiteneffekte.	bit.ly/25KKdGg
WebView	UI Control in welchem Webtechnologien ausführbar sind.	bit.ly/1NHYLQe
YAML	Data Serialisierungs Standard für Programmiersprachen.	bit.ly/1He4Z5q

Anhang

A Paper Prototype

A.1 Trails

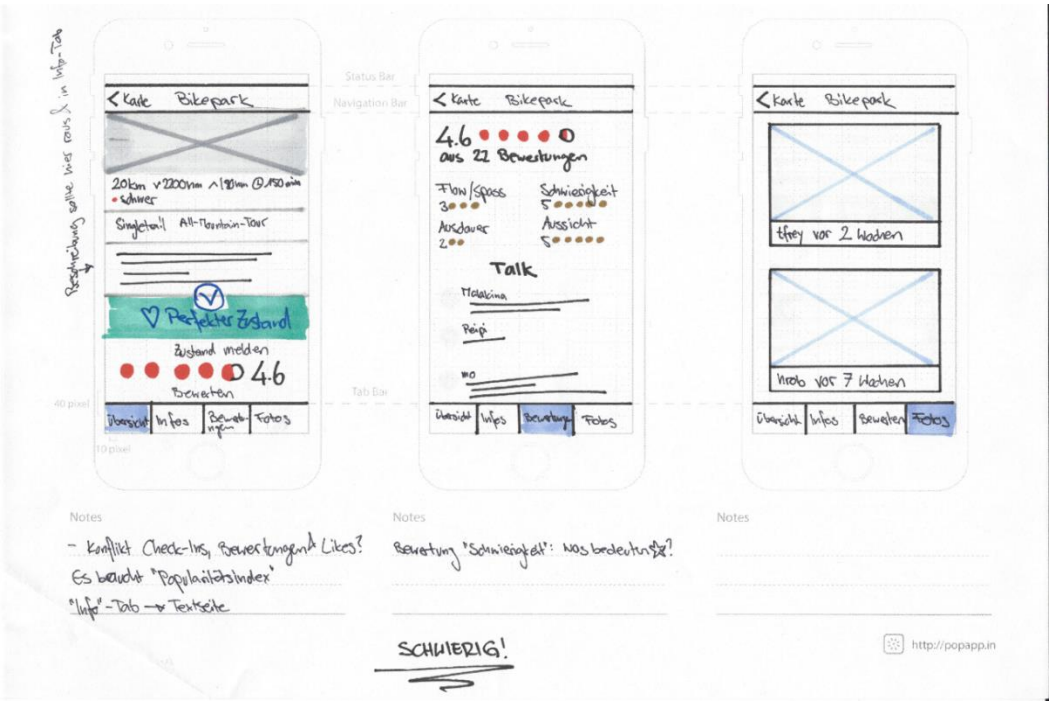

Abbildung 17: Paper Prototype Trails

A.2 Shops

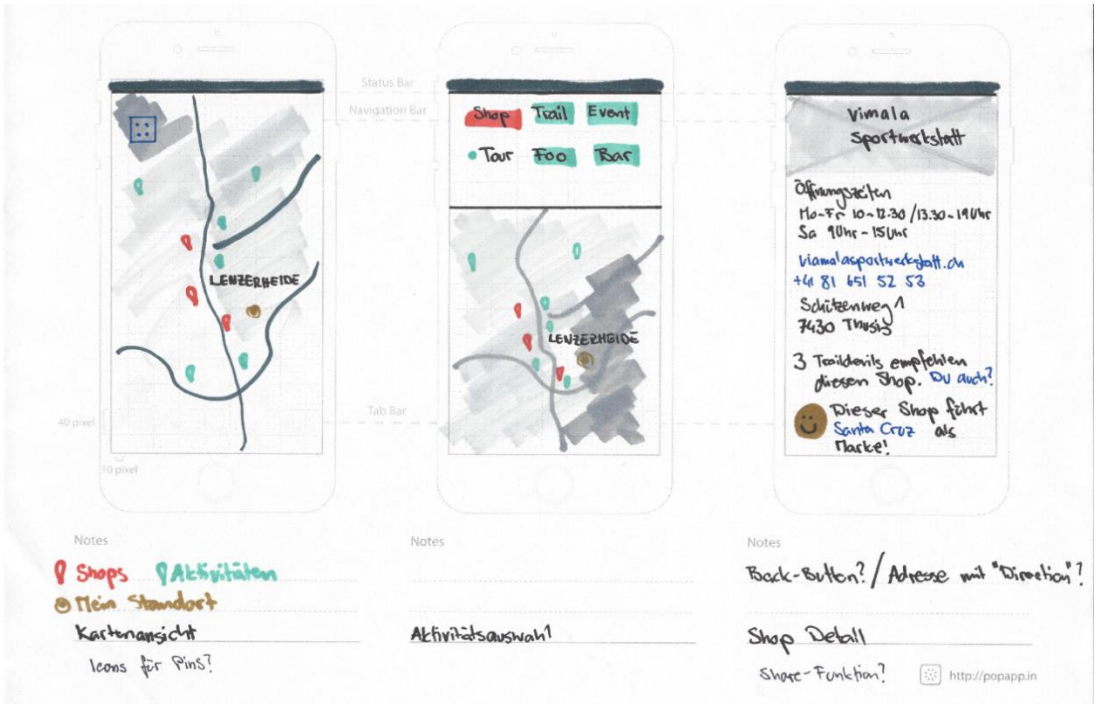

Abbildung 18: Paper Prototype Shops

A.3 Navigation

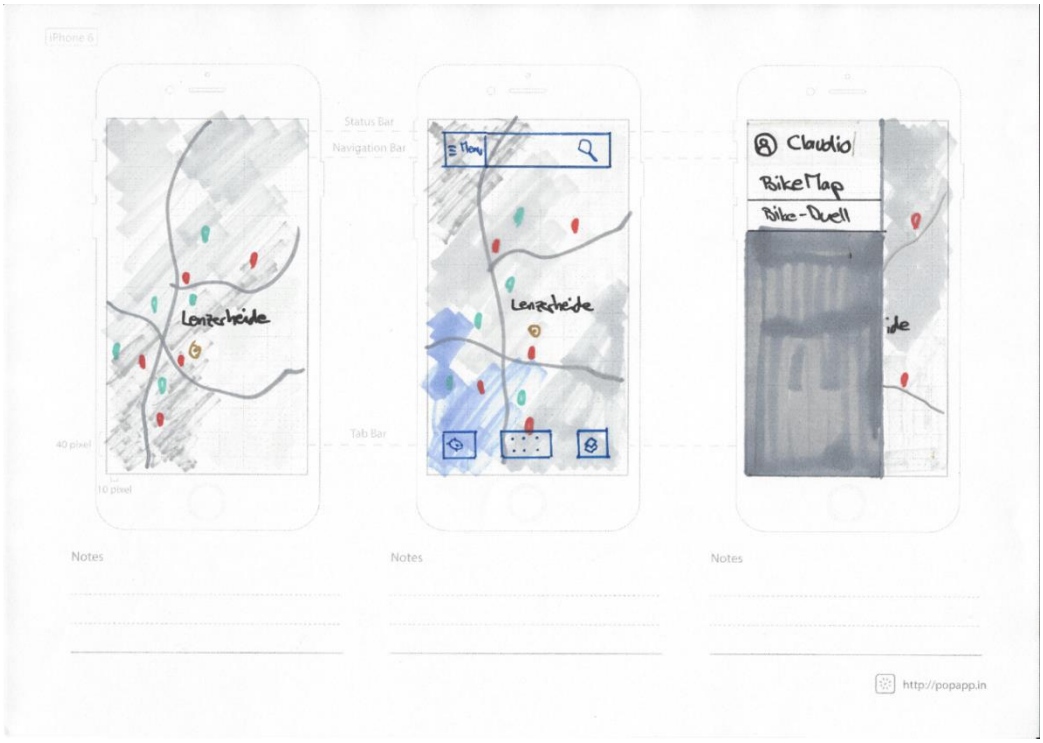

Abbildung 19: Paper Prototype Navigation

A.4 Profil

Abbildung 20: Paper Prototype Profil

A.5 Bewerten / Check-In

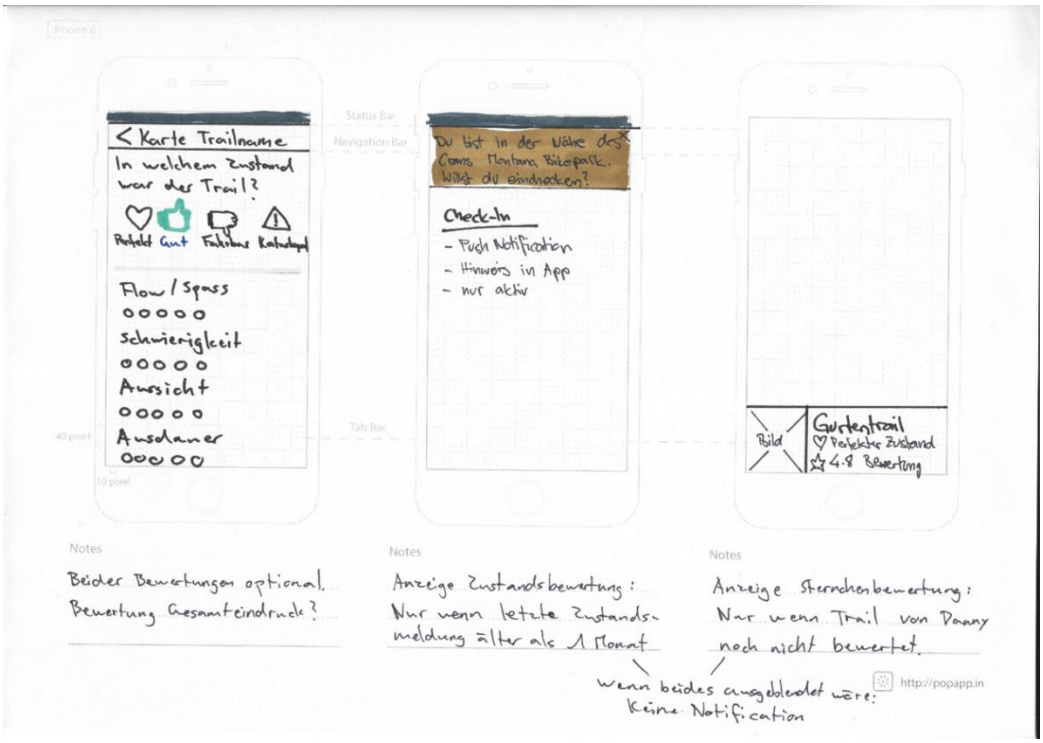

Abbildung 21: Paper Prototype Bewerten / Check-In

A.6 Umfrage

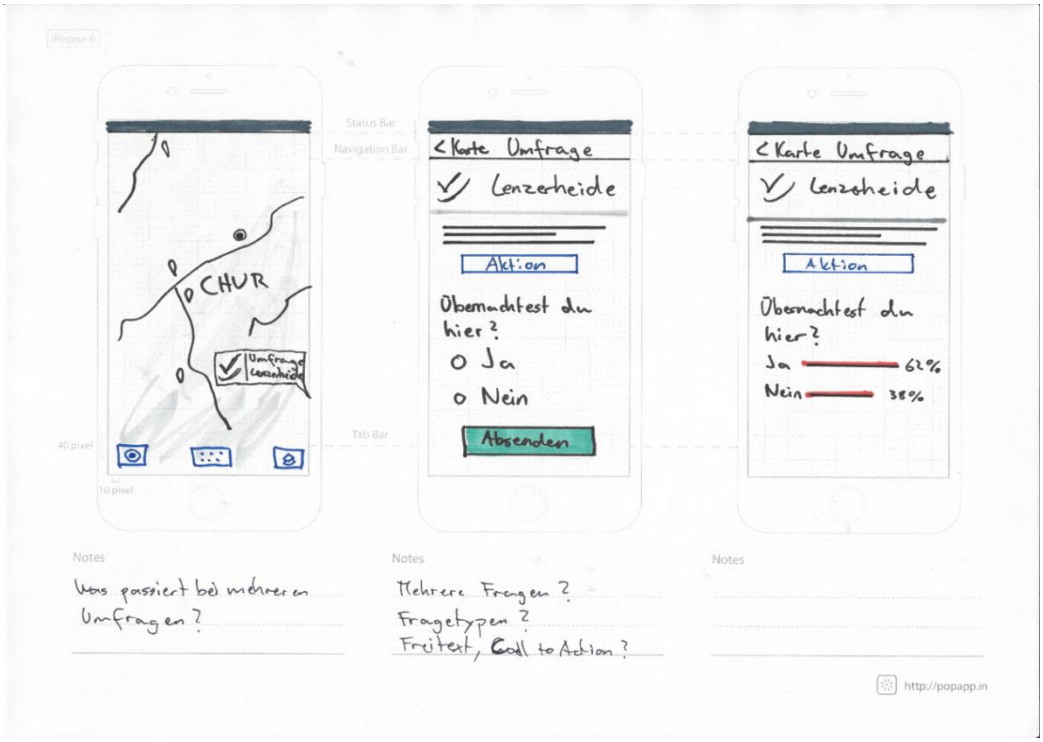

Abbildung 22: Paper Prototype Umfrage

A.7 Login

Abbildung 23: Paper Prototype Login

B Usability Test

B.1 Test Script

Usability Test Script

Traildevils App

Testgruppe: 20 – 40 Jahre alt, Biken mindestens als Hobby, iOS oder Android Nutzer

☐ Testgerät (iOS oder Android Smartphone, je nach Kenntnissen der Testperson) liegt bereit. Traildevils App ist installiert und eine Verknüpfung auf dem Startbildschirm erstellt. Traildevils Testaccount ist bereit und Testperson kennt Logindetails.

Testperson über generelle Vorgehensweise und Art des Tests informieren:

- Getestet wird App, nicht Testperson → Keine Angst vor Fehlern
- Gedanken aussprechen ist wichtig → Nachvollziehbarkeit
- Bei Fragen nachfragen → Hinweis auf Probleme

☐ Anweisung: App vom Startbildschirm des Testgeräts öffnen und beobachten (noch nichts anklicken).

Fragen zum Erscheinungsbild der App stellen:

- Was fällt dir auf?
- Was kannst du hier machen?
- Verwirrt dich etwas?

Testfälle

Szenario durch die Testperson vorlesen lassen, Anweisung zur Ausführung geben und Testperson beobachten bis die Aufgabe erfüllt ist oder die Testperson keinen Weg zur Erfüllung der Anweisung mehr findet. Anschliessend zum nächste Szenario wechseln.

☐ Anweisung: Suche nach Informationen zur Destination ‘Lenzerheide’ und schau dir diese an.

Szenario: Du hast vor kurzen von Kollegen gehört, dass es in der Lenzerheide tolle Trails zum Biken geben soll. Du überlegst dir nächstes Wochenende einen Trail dort auszuprobieren und suchst deshalb nach Informationen zur Destination und den Trails um dich vorzubereiten.

- ☐ Anweisung: Suche Bikeshops in Zürich und wähle aufgrund der gefundenen Informationen einen für dich aus.

Szenario: Du bist umgezogen und wohnst jetzt im Zentrum von Zürich. Du möchtest dich über die Bikeshops dort informieren um zu sehen, wohin du dein Bike bringen kannst.

- ☐ Anweisung: Öffne die Detailseite des Bikepark Lenzerheide und hinterlasse eine „gute“ Bewertung sowie eine entsprechende Zustandsmeldung.

Szenario: Du hast gerade einen tollen Nachmittag im Bikepark Davos hinter dir und sitzt im Zug nach Hause. Weil dir der Bikepark super gefallen hat, möchtest du Andere darauf aufmerksam machen und hinterlässt eine gute Bewertung sowie Zustandsmeldung.

- ☐ Fragerunde und Abschluss

B.2 Durchführungsnotizen

B.2.1 Person 1

Alter: 30

Merkmale: Hobbybiker Downhill, iOS User

Erste Impressionen:

- Profilbutton fällt sofort auf → Tester registriert sich gleich, weil er auf mehr Inhalt und Funktionalität hofft.
- Nach der Registrierung im eingeloggten Zustand fällt der rote Notification-Hinweis sofort auf.
- Tester ist auf Profilseite leicht verwirrt, da er Strava nicht kennt und keine Erklärung sieht.
- Da die Karte auf die aktuelle Position des Testers reinzoomt, findet er im ersten Moment keinen Inhalt.

Szenario 1:

- Tester sucht sofort im Suchfeld nach Lenzerheide.
- Tester springt direkt über Suchresultat zur Destination Lenzerheide, übersieht jedoch das geöffnete Preview und öffnet Destination nicht.
- Er sucht Trails aufgrund des angezeigten Kartenausschnitts.
- Tester versucht wiederholt auf Marker-Flyout zu klicken um Details anzuzeigen, übersieht dabei zuerst das Preview.
- Tester beachtet bei den Trail-Details zuerst den Zustand und findet die Information super, er sieht auch sofort, dass er selber einen Zustand melden kann.
- Tester schaut für Entscheidung stark auf die aufgelisteten Fakten, fände aber weitere Fotos und GPS-Track auf der Karte nützlich.

Szenario 2:

- Tester sucht sofort im Suchfeld nach Zürich und springt über Suchergebnisse zum Ort.
- Er merkt schnell, dass die roten Marker für Shops stehen.
- Öffnet verschiedene im Kartenausschnitt angezeigte Shops und findet die Öffnungszeiten wichtig.
- Für weitere Infos würde er die Shop-Webseite aufrufen, das Wichtigste ist aber in der App ersichtlich.
- Bilder hält er hier für nicht so wichtig.

Szenario 3:

- Tester erkennt das Suchhistory angezeigt wird und nutzt sie um wieder auf der Karte zur Lenzerheide zu springen.
- Er findet schnell den Trail und kann ohne Problem bewerten und den Zustand melden.

Abschliessende Gedanken:

- Tester findet die App nützlich um Trails für den nächsten Ausflug zu suchen.
- App sei ansprechend, aber die Klickbarkeit der Elemente verbesserungswürdig.
- Tester denkt, Kommentare zur Bewertung wären gut; selbst abgeben würde er einen Kommentar eher nicht, aber dafür sehr gerne andere lesen.
- Tester kennt traildevils.ch primär vom Marktplatz und hat Bikemap erst jetzt durch die App richtig wahrgenommen.

B.2.2 Person 2

Alter: 31

Merkmale: Intensiver Hobbybiker, Android User

Erste Impressionen:

- Suchfeld und Profilbutton fallen gleich auf.
- App ist schön und einfach gehalten.
- Mapbox Infobutton ist sehr unscheinbar und sein Zweck nicht ganz klar.

Szenario 1:

- Tester sucht im Suchfeld nach Lenzerheide und klickt auf Suchergebnis-Eintrag des Ortes.
- Er sucht kurz den angezeigten Kartenausschnitt ab und öffnet dann verschiedene Trails.
- Tester merkt an, er möchte gerne GPS Tracks sehen.
- Er versucht in der Detailansicht auf die angezeigten Trail-Typen zu klicken und merkt dann, dass es nur zur Info ist.
- Er hätte gerne, dass nach Trail-Typen gefiltert werden kann.
- Tester ist durch die schwarze Farbe der Trail-Marker leicht verwirrt, da auf anderen Plattformen die Farben rot, schwarz, blau für die Schwierigkeit des Trails stehen.

Szenario 2:

- Tester sucht im Suchfeld nach Bikeshop Zürich und ist verwirrt, weil die Suche nichts findet und auch nicht sagt, dass keine Ergebnisse gefunden wurden.
- Er sucht in einem zweiten Anlauf nach dem Begriff „Zürich“ und findet so den Ort und Shop-Marker auf der Karte.
- Die Shop Infos in der Detailansicht entsprechen seinen Erwartungen.
- Tester merkt an, dass das ausgewählte Suchergebnis im Suchfeld den eingegebenen Suchtext überschreiben sollte.
- Tester merkt im ersten Moment nicht, dass die Adresse, Tel-Nr., etc. angeklickt werden können.

Szenario 3:

- Tester sucht wieder im Suchfeld nach Lenzerheide und wählt in den Ergebnissen direkt den Bikepark aus und öffnet über die Preview die Detailansicht.
- Er findet ohne Probleme den Ort für die Zustandsmeldung und loggt sich ein, da der Login-Screen erscheint.
- Tester ist verwirrt, weil nach dem Login wieder die Karte angezeigt wird und nicht die Detailansicht, auf der er davor war.
- Nach erneutem Öffnen der Detailansicht erledigt er die Zustandsmeldung und Bewertung problemlos.
- Er wünscht sich die Möglichkeit zur Bewertung und Zustandsmeldung einen Kommentar abzugeben.
- Es ist nicht richtig ersichtlich, dass bei schlechten Zustandsmeldungen ein Kommentar angegeben werden kann.

Abschliessende Gedanken:

- Elemente auf Profilseite verwirren, anstatt des prominent platzierten Strava-Buttons wären mehr Profilinfos erwünscht.
- Die Klickbarkeit der Elemente ist verbesserungswürdig.
- Performance bei Navigation in der App wird als ok empfunden.
- Tester wünscht sich eine intelligentere Suche (z.b. suche nach Bikeshop Zürich ermöglichen, die dann alle Bikeshops in Zürich unabhängig vom Namen findet).
- Das UI ist im Allgemeinen intuitiv.
- Wichtige fehlende Features, welche die Nützlichkeit der App einschränken: GPS Tracks anzeigen und herunterladen, Datenqualität muss aber stimmen → Sharing gpx z.B. mit Garmin Navi.
- Anzeigen der Aktualität für Zustände und Bewertungen wäre wichtig.

B.2.3 Person 3

Alter: 33

Merkmale: Hobbybiker Downhill, iOS User

Erste Impressionen:

- Tester fängt nach dem Öffnen der App sofort an auf der Karte herum zu navigieren und auf verschiedene Marker zu klicken.
- Tester denkt zuerst, er könne bei Mapbox Infobutton noch auf andere Karten wechseln.

Szenario 1:

- Tester nutzt sofort das Suchfeld, um nach Lenzerheide zu suchen.
- Er wählt den Ort Lenzerheide aus den Suchresultaten aus und merkt, dass die Karte zu diesem Ort springt.
- Tester öffnet die Detailansicht eines Trails und sieht schnell die wichtigsten Infos. Ist allerdings kurz durch die Zeitangabe verwirrt, da keine Einheiten (Minuten, Stunden, etc.) angegeben ist.
- Tester navigiert ein wenig auf der Karte und nutzt dann die Suchhistory, um erneut zum Ort Lenzerheide zu springen.
- Er hätte gerne GPS Tracks auf der Karte eingezeichnet, wenn er reinzoomt.

Szenario 2:

- Tester sucht im Suchfeld nach Zürich, springt über das Suchergebnis zum Ort und sieht sofort die Shop-Marker, auf welche er klickt.
- Er fände die Möglichkeit auf Shops zu filtern nützlich, wenn er spezifisch nach Shops sucht.
- Er merkt an, dass personalisierte Vorschläge für Shops aufgrund einer Heimadresse, welche im Profil hinterlegt ist, für diesen Task nützlich wäre.

Szenario 3:

- Tester findet den gewünschten Trail durch die Suche, sieht aber auf der Karte nicht welcher Marker angewählt ist, da das Marker-Flyout wegen eines Klicks auf die Karte schon wieder verschwunden war.
- Es ist ihm sofort klar, dass die Zustands- und Bewertungsmeldung mit einem Klick auf das entsprechende Info-Objekt gemacht werden können.

Abschliessende Gedanken:

- Profildaten sollten geändert werden können.
- Objekte relativ klein, was die Klickbarkeit verschlechtert.
- Previews mit dunklen Bildern werden im ersten Moment übersehen. Sobald diese das erste Mal wahrgenommen worden sind, stellt das kein Problem mehr dar.
- Bei den Trails und Destinationen wären Kontaktdetails noch nützlich (z.B. Tel-Nr. eines Notfallkontakts).

Person 4

Alter: 24

Merkmale: Intensiver Biker (fährt auch offizielle Rennen), Englisch sprechend, iOS User

Erste Impressionen:

- Die Ladeanzeige im Suchfeld verwirrt am Anfang, da noch keine Suche abgesetzt wurde.
- Profilbutton und Mapbox Infobutton fallen sofort auf.
- App wirkt übersichtlich.

Szenario 1:

- Tester fängt gleich mit Suche nach Lenzerheide im Suchfeld an und findet die Destination auf der Karte.
- Das angezeigte Preview wird zuerst als Werbung wahrgenommen und deshalb nicht sofort angeklickt.
- In der Detailansicht stechen der Zustand und die Bewertungen ins Auge und werden als nützlich wahrgenommen.
- Die angezeigten Trail-Typen werden zuerst als Buttons wahrgenommen und der Tester versucht sie anzuklicken.
- Tester öffnet einen Trail von der Detailseite der Destination aus und ist anschliessend verwirrt, dass beim Schliessen der Trail-Detailansicht nicht wieder die Destination angezeigt wird sondern die Karte.

Szenario 2:

- Tester sucht sofort über das Suchfeld nach Zürich, dabei empfindet er die Suchresultate direkt während der Eingabe als nützlich.
- Tester findet schnell einen Shop in der Nähe seines Wohnorts.
- Er kann sich für einen Shop anhand der vorhanden Infos entscheiden aber ihm fehlen weitere Bilder um ein besseres Gefühl für den Shop zu erhalten.

Szenario 3:

- Tester findet den Bikepark Lenzerheide schnell über die Suche.
- Er klickt sofort auf die Zustandsinformation um den Zustand zu melden.
- Beim Erscheinen der Registrierungsseite will sich der Tester zuerst mit seinem Login einloggen und merkt erst nach der Eingabe seiner Mailadresse, dass er auf die Loginseite wechseln muss, dabei ist ihm die Tastatur im Weg.
- Bei der Registrierung ist sich der Tester nicht sicher ob das Land seiner Herkunft gemeint ist oder ob er angeben muss, in welchem Land er an Trails, usw. interessiert ist.
- Nach dem Login prüft er sofort das Profil, um nach der angedeuteten Notification zu sehen.
- Die Notification ist im Profil nicht sofort ersichtlich, aber nach einem Moment ist ihm klar, dass Strava gemeint ist.
- Tester öffnet den Trail erneut und gibt ohne Probleme die Bewertung und Zustandsmeldung ab, dabei wünscht er sich die Möglichkeit, immer einen Kommentar mitgeben zu können. Er merkt aber an, dass er das meistens überspringen würde.

Abschliessende Gedanken:

- Tester möchte klarere Trennungsmerkmale zwischen offiziellen Trails und Destinationen und solchen, die von anderen Nutzern erfasst wurden.
- Er findet den Zustand und die Bewertung auf Trails nützlich und wichtig, die App macht das gut.
- Tester würde die App am Anfang nutzen um neue Trails zu finden die er noch nicht kennt.
- Die Strava Integration würde er wahrscheinlich nutzen.
- Personalisierung (auch in Zusammenhang mit Strava) wäre für ihn ein wichtiges Feature (z.B. Vorschläge für neue Trails basierend auf den Trails die er schon kennt).
- App hinterlässt grundsätzlich einen positiven Eindruck beim Tester.

- Beim Melden von Zuständen wäre ihm die Möglichkeit wichtig, Meldungen mit Kommentar und Bild zu versehen → Motto: «Schlamm zeigen».
- Er vermisst auch «Stories» oder «Highlights» als Anregungen, aktuell muss er eine Idee haben, was er machen will um Infos von der App zu erhalten.

Ride on.