

Dokumentation

comfycoffee

Mobile Web

Name: Thomas Wedler, Katharina Garrecht
E-Mail: inf2920@hs-worms.de, inf2930@hs-worms.de
Studiengang: Mobile Computing
Matrikelnummer: 673324, 673342
Fachsemester: 2

Betreuer: Prof. Dr. Elisabeth Heinemann

Einreichung: 31.01.2018

Inhaltsverzeichnis

1 Einleitung	4
1.1 Ziel	4
1.2 Unique Selling Proposition (USP)	4
2 Konzeption	5
2.1 Zielgruppe	5
2.2 Personas	5
2.2.1 Primärpersona	5
2.2.2 Sekundärpersona	6
2.3 Anforderungsanalyse	7
2.3.1 Muss-Anforderungen	7
2.3.2 Soll-Anforderungen	8
2.3.3 Kann-Anforderungen	8
2.4 Ablaufplan	9
2.5 Flow Chart	9
2.6 Wireframes	11
2.7 MockUps	12
2.8 Prototyp	12
3 Entwicklung	13
3.1 React Native	13
3.2 Setup	13
3.3 Module	14
3.4 Arbeitsaufteilung	15
3.4.1 Gemeinsame Aufgaben	16
3.4.2 Aufgaben von Katharina Garrecht	16
3.4.3 Aufgaben von Thomas Wedler	17
3.5 Vorstellung der App	18
3.5.1 Startseite	18
3.5.2 Kartenansicht	19
3.5.3 Kartenansicht - ausgewählt	20
3.5.4 Listenansicht	21
3.5.5 Infos Café	22
3.5.6 Bewertungen Café	23
3.5.7 Lexikon	24

3.5.8	Lexikon Detailansicht	25
3.6	Readme	26
3.6.1	Voraussetzungen	26
3.6.2	Testing und Debugging	26
4	Ausblick	27
4.1	Testkonzept	27
4.2	Marketing	28
4.2.1	Produktpolitik	28
4.2.2	Preispolitik	28
4.2.3	Distributionspolitik	28
4.2.4	Kommunikationspolitik	29
4.2.5	Retention Marketing	29
5	Fazit	30
5.1	Lessons Learned	30
5.2	React Native	30
5.3	Cross-Plattform vs. Nativ	31
Eidesstattliche Erklärung		32

1 Einleitung

Starbucks, wohin das Auge blickt - das bedeutet Massenabfertigung an der Kasse, Kaffee mit Zusatz- und Konservierungsstoffen, Steuervermeidungstaktiken, angeblich ungerecht behandelte Mitarbeiter und unklare Fairness im Kaffeeanbau. Die kleinen, gemütlichen Kaffees verschwinden Aufgrund des hohen Konkurrenzdrucks von billigeren Anbietern immer mehr aus dem Stadtbild. Fair gehandelter Kaffee und exotische Sorten werden immer seltener angeboten.

1.1 Ziel

Unser Bestreben war es daher, kleinen lokalen Cafés wieder mehr Aufmerksamkeit zu schenken. Unsere App *comfycoffee*, die dies umsetzt, verfolgt deshalb vor allem zwei große Ziele:

- Finde Geheimtipps für lokale Cafés im Umkreis
- Lerne neue Kaffeesorten und Zubereitungsarten kennen

Comfycoffee soll auf kleine lokale Cafés aufmerksam machen und im Idealfall deren Kundenstamm erhöhen. Kaffeeliebhaber können die großen Ketten mit ihrem “0815”-Kaffee umgehen, lokale Cafés entdecken und neue Erfahrungen machen. Zusätzlich ist die App als Lexikon nutzbar, indem sie den Nutzer über Kaffeesorten informiert.

1.2 Unique Selling Proposition (USP)

Der USP von *comfycoffee* betont die oben genannten Ziele:

“Anzeige lokaler Cafés und damit nachhaltige Unterstützung der lokalen Kaffeewirtschaft gegen die großen Ketten”

Durch das Bekanntmachen kleiner Cafés kann deren Umsatz erhöht werden, so dass sie sich eher gegen große Ketten wie Starbucks und Co. behaupten können.

2 Konzeption

In der Konzeptionsphase wurden die Zielgruppen von comfycoffee herausgearbeitet und analysiert. Personas aus der Zielgruppe repräsentieren potentielle Nutzer der App. Anschließend wurden die Anforderungen an die App definiert und das Design entworfen.

2.1 Zielgruppe

Die Zielgruppe für comfycoffee besteht aus Menschen, die gerne Kaffee trinken und dabei offen für Neues sind.

Die primäre Zielgruppe besteht aus Kaffeeliebhabern, wie z.B. Studenten oder Arbeitnehmern. Sie genießen die Atmosphäre der kleinen Kaffees, legen Wert auf Fair-Trade und besitzen eine “Support your locals”-Mentalität. Das bedeutet, sie unterstützen und schätzen lokale Angebote.

Die sekundäre Zielgruppe sind Urlauber und Touristen, die zu Besuch in einer fremden Stadt sind und lokale Geheimtipps entdecken wollen.

2.2 Personas

Bei der Ausarbeitung der Personas wurde eine Vertreterin der primären und ein Vertreter der sekundären Zielgruppe modelliert.

2.2.1 Primärpersona

Laura Wagner vertritt die primäre Zielgruppe (vgl. Abb. 2.1). Sie...

- ...ist 25 Jahre alt
- ...ist ledig
- ...studiert Sozialpädagogik
- ...ist extrovertiert und spontan
- ...geht gerne mit Freunden im kleinen Café um die Ecke was trinken, will aber auch andere Cafés entdecken

- ...legt Wert auf Fairtrade- und Biowaren, gemütliche und individuelle Cafés, kurze Wege
- ...ist technikaffin, d. h. sie bewegt sich mit ihrem Smartphone (Samsung Galaxy A5 (Android)) souverän in den Sozialen Medien



Laura Wagner (25)
Studentin der Sozialpädagogik

Eigenschaften
ledig • extrovertiert • spontan • technikaffin

Mentalität

- Geht gerne mit Freunden im kleinen Café um die Ecke was trinken, will aber auch andere Cafés entdecken
- Legt Wert auf Fairtrade- und Biowaren, gemütliche und individuelle Cafés, kurze Wege
- Smartphone: Samsung Galaxy A5 (Android)

Abbildung 2.1: Primärpersona Laura Wagner

2.2.2 Sekundärpersona

Roland Reisenweber vertritt die sekundäre Zielgruppe (vgl. Abb. 2.2). Er...

- ...ist 42 Jahre alt
- ...ist verheiratet
- ...arbeitet als Journalist bei Frankfurter Allgemeine
- ...ist extrovertiert und offen für Neues
- ...arbeitet gerne an öffentlichen Orten
- ...besitzt ein ausgeprägtes soziales Bewusstsein
- ...ist gerne mit seiner Frau auf Reisen (innerhalb der EU)

- ...will regionale Kulturen kennenlernen
- ...ist weniger technikaffin, d. h. er nutzt sein iPhone nur zum telefonieren



Roland Reisenweber (42)
Journalist bei Frankfurter Allgemeine

Eigenschaften
verheiratet • offen für neues • wenig technikaffin

Mentalität

- Arbeitet gerne an öffentlichen Orten
- Soziales Bewusstsein
- Ist gerne mit seiner Frau auf Reisen (innerhalb der EU)
- Will regionale Kulturen kennenlernen
- Smartphone: iPhone 7 (iOS)

Abbildung 2.2: Sekundärpersona Roland Reisenweber

2.3 Anforderungsanalyse

In der Anforderungsanalyse haben wir die Muss-, Soll- und Kann-Anforderungen von *comfycoffee* definiert.

Learnings: Dabei war die Herausforderung, die Anforderungen exakt zu formulieren und in Pakete zu unterteilen, die klein genug waren, um sie in der im Ablaufplan definierten Zeit abzuarbeiten (vgl. Kapitel 2.4).

2.3.1 Muss-Anforderungen

Die Muss-Anforderungen wurden auf das Nötigste heruntergebrochen, um die App sinnvoll bedienen zu können. Vorgabe war, den Bewegungssensor (Gyroskop) und GPS zu benutzen. Daraus ergaben sich die folgenden Muss-Anforderungen:

- Lokale Cafés in der Nähe anzeigen

- Kartenansicht in Google Maps
- Listenansicht der Cafés im Umkreis
- Standortdaten via Google API
- Kaffelexikon
- Einträge über Zubereitungsarten
- Schüttelfunktion (Benutzung des Bewegungssensors) für Zufallswahl einer Zubereitungsart

2.3.2 Soll-Anforderungen

Die Soll-Anforderungen erweitern den Funktionsumfang der App um nützliche Inhalte. Diese waren:

- Navigation zu ausgewähltem Café über die App *Google Maps*
- Listenansicht mit den wichtigsten Standortinformationen
- Detailseite mit mehr Infos über das Café (die Informationen sind abhängig davon, was die Google API zur Verfügung stellt)

2.3.3 Kann-Anforderungen

Die Kann-Anforderungen ergänzen die App durch weitere Informationen und Features, die der Nutzer nicht zwingend erwartet, aber als angenehm empfindet. Die Kann-Anforderungen waren:

- Mehr Daten über zusätzliche APIs
- Eigene Bewertungen der Cafés durch die App
- Eigene Standorte vorschlagen
- Erweitertes Lexikon (z. B. mit Kaffeesorten, etc.)
- Profil des Nutzers anlegen

2.4 Ablaufplan

Anschließend wurde ein Ablaufplan (vgl. Abb. 2.3) ausgearbeitet und festgelegt. Dabei sollte die Designphase mit Wireframes, Flow Charts, MockUps und einem interaktiven Prototyp bis zum 22.11.2017 abgeschlossen sein. Für die Einarbeitung in das Entwicklungsframework, React Native (vgl. Kapitel 3.1), wurde eine Woche eingeplant. Bis zum 29.11.2017 sollte eine “Hello World”-Anwendung laufen, um die Grundzüge des Frameworks zu beherrschen. Zwei Wochen, also bis zum 13.12.2017, waren für die Muss-Anforderungen, eine Woche für die Soll-Anforderungen (bis 20.12.2017) geplant. Die Kann-Anforderungen sollten im Januar 2018 abgeschlossen sein. Das folgende Diagramm veranschaulicht den Ablauf.



Abbildung 2.3: Ablaufplan der Konzeption und Entwicklung von *comfycoffee*

Learnings: Die Aufwandsschätzung zu Beginn gestaltete sich als schwierig, da keine Erfahrung mit React Native und der Cross-Plattform-Entwicklung bzw. allgemein der App-Entwicklung vorhanden war.

2.5 Flow Chart

Das Flow Chart stellt die Navigation durch die App und die verschiedenen Views dar (vgl. Abb. 2.4). Von der Startseite aus kann der Nutzer wählen, ob er die Karte mit den Cafés in der Nähe oder das Kaffee-Lexikon aufrufen will.

In der Karte befindet sich eine Bottom Navigation Bar, mit der von der Karte zur Listenansicht und zurück gewechselt werden kann. Klickt der Nutzer in der Karte auf einen Marker oder in der Liste auf einen Eintrag, so wird er auf die Detailseite des jeweiligen Café geleitet. Dort erhält er Informationen wie Adresse, Öffnungszeiten und Bewertungen des Cafés. Über einen Button wird die Navigation zu dem Café in Google Maps geöffnet. Das Navigationskonzept wurde in der Entwicklung abgeändert, um den Design-Guidelines von Android und iOS zu entsprechen. Aus der Bottom Navigation Bar wurde in Android eine Tab Bar. Weiterhin wird auf der Detailseite eine Tab bzw.

Bottom Navigation Bar angezeigt, um von der Seite des Cafés zu den Bewertungen zu wechseln.

Das Lexikon beinhaltet verschiedene Kaffeesorten. Beim Auswählen einer Kaffeesorte wird der Nutzer auf eine Unterseite geleitet, die Details zu der jeweiligen Kaffeesorte enthält. Schüttelt der Nutzer das Smartphone, während er sich auf der Übersichtsseite des Lexikons befindet, wird zufällig eine Kaffeesorte ausgewählt und der Nutzer hat die Möglichkeit, sich zu der genannten Detailseite navigieren zu lassen.

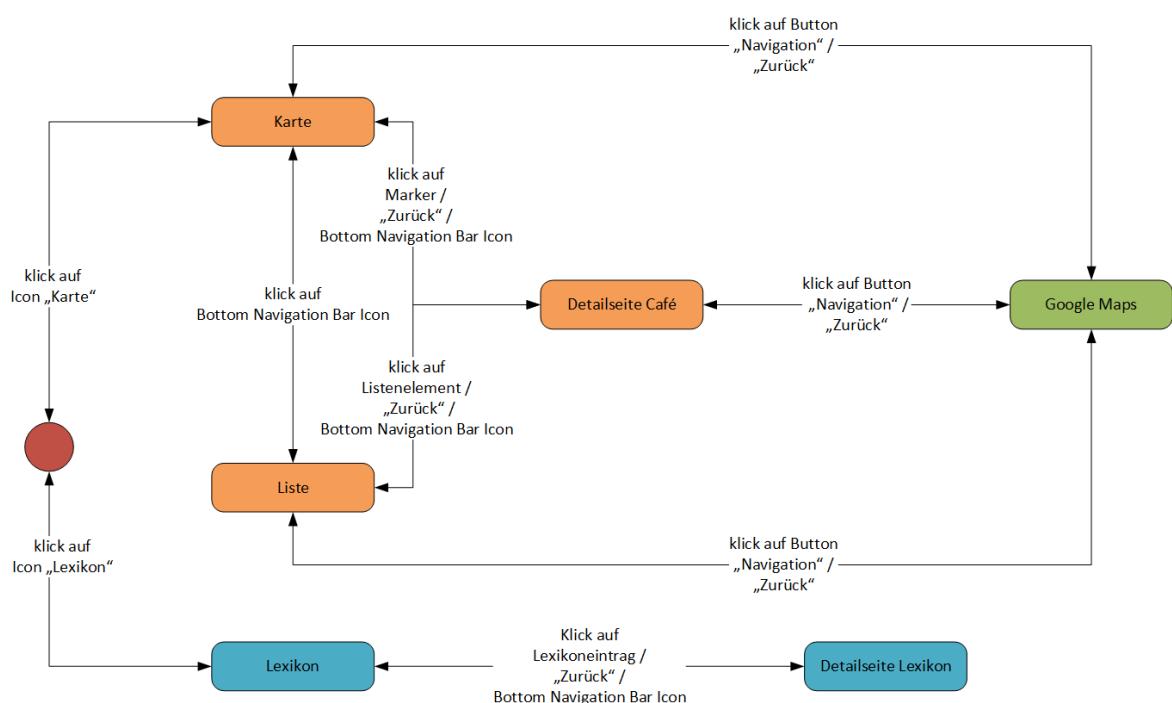


Abbildung 2.4: Erster Entwurf des Flow Charts von comfycoffee

2.6 Wireframes

Nachdem die Anzahl und Anforderungen an die Views feststanden, wurden diese auf Papier in Form von Wireframes visualisiert. Dabei wurden die Elemente pro View bestimmt und deren Position auf den einzelnen Seiten festgelegt (vgl. Abb. 2.5).

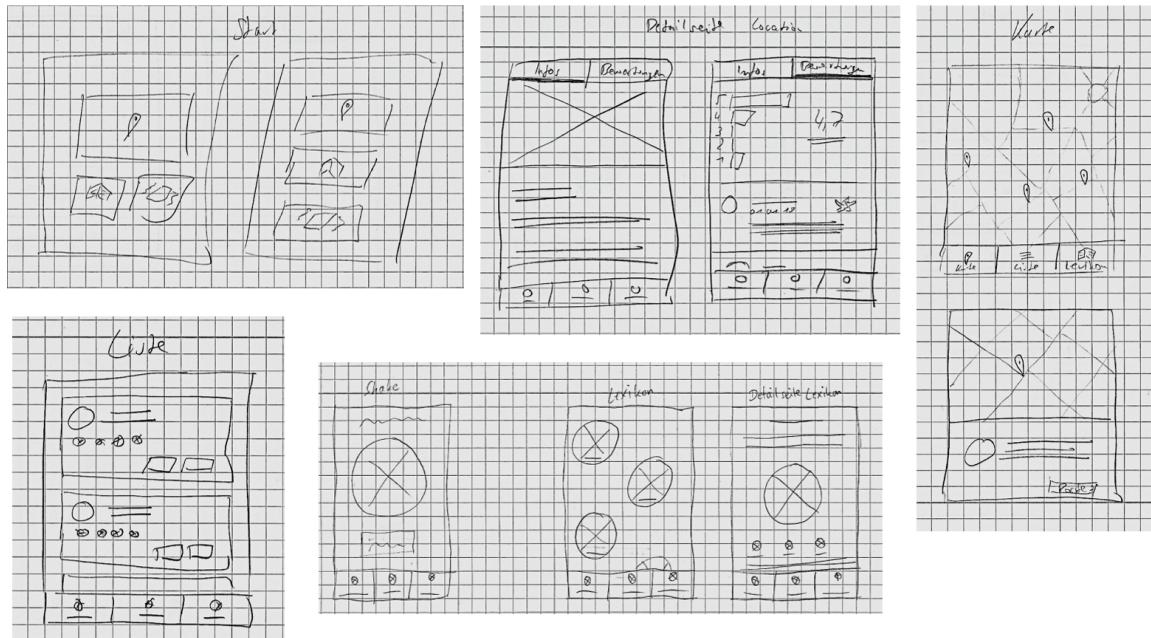


Abbildung 2.5: Wireframes von comfycoffee

2.7 MockUps

Aus den Wireframes wurden MockUps mit dem MockUp-Tool Axure konfiguriert. Zu jeder View wurde ein MockUp erstellt. Im Gegensatz zu den Wireframes sind hier schon Farben, Bilder und Texte beispielhaft enthalten (vgl. Abb. 2.6).



Abbildung 2.6: Ansicht der Detailseite eines Kaffees als MockUp

2.8 Prototyp

Anhand der MockUps und des Flow Charts wurde in Axure ein interaktiver Prototyp erstellt. Er dient zum Durchklicken durch die App und um einen ersten Eindruck in Sachen Usability zu gewinnen. Der Prototyp ist online verfügbar unter <https://jzd0d2.axshare.com>.

3 Entwicklung

Mit dem Abschluss der Konzeptionsphase begann die technische Umsetzung der App. Nachdem ein passendes Framework zur Entwicklung der App auf mehreren Plattformen gefunden wurde, konnte eine sinnvolle Projektstruktur mit allen benötigten Bibliotheken angelegt werden. Die Implementierung einiger Freatures wurde dabei parallel durchgeführt. Weiterhin wird in diesem Kapitel der aktuelle Stand der App beschrieben. Eine Readme inklusive Installationsanleitung schließt die technische Seite des Projekts ab.

3.1 React Native

Nach kurzer Diskussion und dem Abwägen von Vor- und Nachteilen entschieden wir uns dafür, das *React Native* Framework zu benutzen. React Native stellt somit das technische Kernelement des Projekts dar. Als ein von *Facebook* entwickeltes und verwaltetes Repository erfreut es sich mit über 58.000 Sternen auf *GitHub* großer Beliebtheit. Demnach steht bereits eine große Community hinter dem Cross-Plattform-Framework. Die Entwicklung mit React Native erfolgt fast ausschließlich über JavaScript. Als Cross-Plattform-Framework ermöglicht React Native den Zugriff auf native Komponenten via JavaScript und profitiert somit von einer besseren Performance als hybride Frameworks. Allerdings erscheint es trotz seiner Popularität technisch noch nicht sehr etabliert. Darüber hinaus benötigen die nativen Komponenten meist noch Anpassungen per Hand für Android und iOS.

3.2 Setup

Um React Native einem Projekt hinzuzufügen stehen zwei Möglichkeiten zur Verfügung. Eine davon ist der *Quick Start*, welcher nach erfolgreichem Ausführen einen QR-Code anzeigt. Über das Smartphone kann der Code gescannt und die App aufgerufen werden. Vor- und Nachteil gleichermaßen ist, dass die Entwicklungsumgebungen *Android Studio* für Android und *Xcode* für iOS nicht installiert sein müssen. Allerdings ist es somit auch nicht möglich, selbsterstellte React Native Module hinzuzufügen. Demnach entschieden wir uns für die zweite Variante, um zusätzlich Zugriff auf eigene Module zu haben. Hierbei wird React Native über die Kommandozeile und den Befehl 'create-react-native-app' initialisiert.

Grundvoraussetzungen hierbei sind eine funktionierende Version von *Node.js* sowie das Facebook Tool *Watchman*. Für die Verwaltung von Abhängigkeiten wird bei der Installation von React Native der Paketmanager *Yarn* hinzugefügt, der Zugriff auf die über den *Node Package Manager (npm)* bereitgestellten Module ermöglicht. Zusätzlich müssen die beiden Entwicklungsumgebungen Android Studio und Xcode zum Kompliieren des nativen Codes bereit stehen. Testing und Debugging der App kann somit entweder über die in Android Studio und Xcode integrierten Simulatoren erfolgen oder aber direkt über ein verbundenes Smartphone.

Unsere Wahl für einen Code Editor fiel auf *Visual Studio Code* von Microsoft, da es einen einfachen Einstieg bietet und sowohl für Windows als auch für Mac verfügbar ist. Als Versionierungssystem benutzten wir Bitbucket, welches die Verwaltung von Git-Repositories ermöglicht. Des Weiteren richteten wir ein *Trello-Board* ein, auf dem das Projektmanagement abgebildeten werden sollte.

Learnings: Obwohl die Konzeption unserer App auf einen eher geringen Umfang schließen lässt wurden doch sehr viele Tools benötigt, um eine Entwicklung zu ermöglichen. Für diese Projektgröße stellte sich auch das Projektmanagement-Tool Trello als zu aufwändig in der Pflege heraus.

3.3 Module

Für die Implementierung einiger Funktionalitäten stehen bereits zahlreiche Komponenten und Pakete aus der Community zu Verfügung. Zusätzlich zu den bereits in React Native integrierten Modulen werden folgende in der App verwendet:

- **axios:** Erleichtert die Kommunikation mit beliebigen APIs, da es entsprechende Hilfmethoden zur Verfügung stellt.
- **react-native-maps:** Eine benutzerdefinierte React Native Komponente, welche die Integration einer Google Maps Karte ermöglicht. Leider ein sehr undokumentiertes Modul, welches große Probleme bei der Integration aufwirft.
- **react-native-permissions:** Bietet eine Schnittstelle zur Abfrage von unterschiedlichen Berechtigungen, beispielsweise dem Zugriff auf den aktuellen Standort.
- **react-native-sensors:** Ermöglicht den direkten Zugriff auf den Bewegungssensor des entsprechenden Endgeräts - leider ein ebenfalls eher undokumentiertes Modul.

- **react-native-star-rating**: Die visuelle Darstellung eines Werts (wie beispielsweise einer Bewertung) in Form von Sternen verpackt in einer React Native Komponente.
- **react-native-vector-icons**: Eine Sammlung thematisch geordneter Icon-Bibliotheken, welche nach dem Import direkt zur Verfügung stehen.
- **react-navigation**: Ein alternatives Navigationskonzept als React Native Modul aus der Community. Dieses ersetzt das in React Native integrierte Konzept und ermöglicht eine einfache Implementierung diverser Navigationselemente.

Während der Entwicklung wurden einige zusätzliche Module eingebunden, welche allerdings nicht den gewünschten Anforderungen entsprachen:

- **react-native-material-bottom-navigation**: Die ebenfalls in den Material Design Guidelines vertretene Bottom Navigation für Android verpackt in einer React Native Komponente. In dieser App wird allerdings die Navigation in Android über Tabs bevorzugt.
- **react-native-stars**: Eine Alternative zu dem verwendeten react-native-star-rating Modul. Da dieses Modul weniger Konfigurationsmöglichkeiten bietet als sein Pendant ist es obsolet.
- **react-native-google-places-autocomplete**: Die reduzierte Variante des react-native-maps Moduls. Allerdings ist der Funktionsumfang deutlich zu gering, so bietet es lediglich autocomplete für die Google Places API.

Durch die zahlreichen verfügbaren Komponenten lässt sich eine React Native App sehr stark modularisieren. Es können bei Bedarf die jeweils benötigen Funktionalitäten isoliert angefordert und eingebunden werden. Einzig undokumentierte und redundante Module stellen einen zusätzlichen Aufwand dar. Diese müssen zunächst herausgefiltert und getestet werden, um die beste Lösung zu finden.

3.4 Arbeitsaufteilung

Die Aufteilung der anstehenden Aufgaben sahen wir als notwendig an, um die geplanten Deadlines einhalten zu können. Unumgänglich war allerdings eine gemeinsame Code-Basis, in die zusätzliche Funktionalitäten unabhängig voneinander integriert werden

konnten. Hierzu legten wir durch *Pair Programming* zusammen eine geeignete Grundlage.

3.4.1 Gemeinsame Aufgaben

Das Fundament besteht aus dem Grundgerüst der App sowie der Navigation durch diese. Dabei besteht das Navigationskonzept aus einer übergeordneten *Stack Navigation* sowie mehreren kontextabhängigen, verschachtelten *Tab/Bottom Navigations*. Eine dem Flow Chart entsprechende Seitenstruktur ermöglicht es, Seiteninhalte isoliert zu bearbeiten. Die Implementierung der Karte wurde ebenfalls gemeinsam gelöst, allerdings nach Betriebssystemen getrennt. Sobald der Rahmen der App aufgebaut war arbeiteten wir an unterschiedlichen Funktionalitäten weiter.

Learnings: Bereits in diesem frühen Stadium konnten wir einige Aha-Momente für uns entdecken. Die Tabs und Bottom Navigation, so wie in unseren MockUps skizziert, ließ sich nur teilweise realisieren. Eine weitere Recherche ergab, dass der Aufbau der Navigation in React Native eine andere Lösung für diesen Fall vorgab und wir die Implementierung entsprechend anpassen mussten. Darüber hinaus stießen wir mit dem React Native Modul für die Google Maps Karte (`react-native-maps`) bereits auf ein undokumentiertes Modul aus der Community, welchem noch Weitere folgen sollten.

3.4.2 Aufgaben von Katharina Garrecht

Die Umsetzung der Inhalte des Lexikons und der Lexikon-Detailansicht sowie die Integration des Bewegungssensors übernahm Katharina Garrecht. Hierbei implementierte sie die im Prototypen definierten Elemente sowohl für die Übersichtsseite als auch für das Template der Kaffee-Detailseite. Die Detailseiten werden dabei mit in einer JSON Datei hinterlegten Inhalten gespeist. Auf der Übersichtsseite selbst ist der Bewegungssensor aktiv, welcher bei Erkennung einer Schüttelgeste zufällig eine der zur Verfügung stehenden Kaffee-Zubereitungsarten auswählt und vorschlägt.

Learnings: Die Integration des Bewegungssensors in die bestehende Navigation brachte einige Stolpersteine mit sich, die nur durch lange Recherchen und Umwege überwunden werden konnten. Auch die Einarbeitung in native UI Komponenten der jeweiligen Betriebssysteme war für die Umsetzung dieser Funktionalitäten notwendig.

3.4.3 Aufgaben von Thomas Wedler

Die Darstellung der Listenansicht aller nahen Cafés sowie die Detailansicht einzelner Cafés fiel in den Aufgabenbereich von Thomas Wedler. Ebenso Teil dieses Bereichs war die Implementierung der Geolocation zur Ermittlung des aktuellen Standorts. Über diese Geokoordinaten werden bei Bedarf die *Google Places API* und auch die *Google Maps API* angesprochen. Das Ergebnis ist situationsabhängig entweder eine Liste aller umliegenden Cafés mit rudimentären Informationen zu diesen oder aber ein ausführliches Objekt mit zahlreichen Detailinformationen über ein einzelnes Café.

Learnings: Für die Umsetzung der benötigten Funktionalitäten war eine intensive Einarbeitung in das Ecosystem sowie die bereitgestellten Schnittstellen der Google APIs unumgänglich. Ebenfalls hier mussten native UI Komponenten der jeweiligen Betriebssysteme korrekt identifiziert und implementiert werden.

3.5 Vorstellung der App

Im Folgenden wird die App mit ihren Views vorgestellt.

3.5.1 Startseite

Der Einstiegspunkt der App. Von dort aus ist die Navigation zur Umkreissuche oder zum Lexikon möglich (vgl. Abb. 3.1 und 3.2).

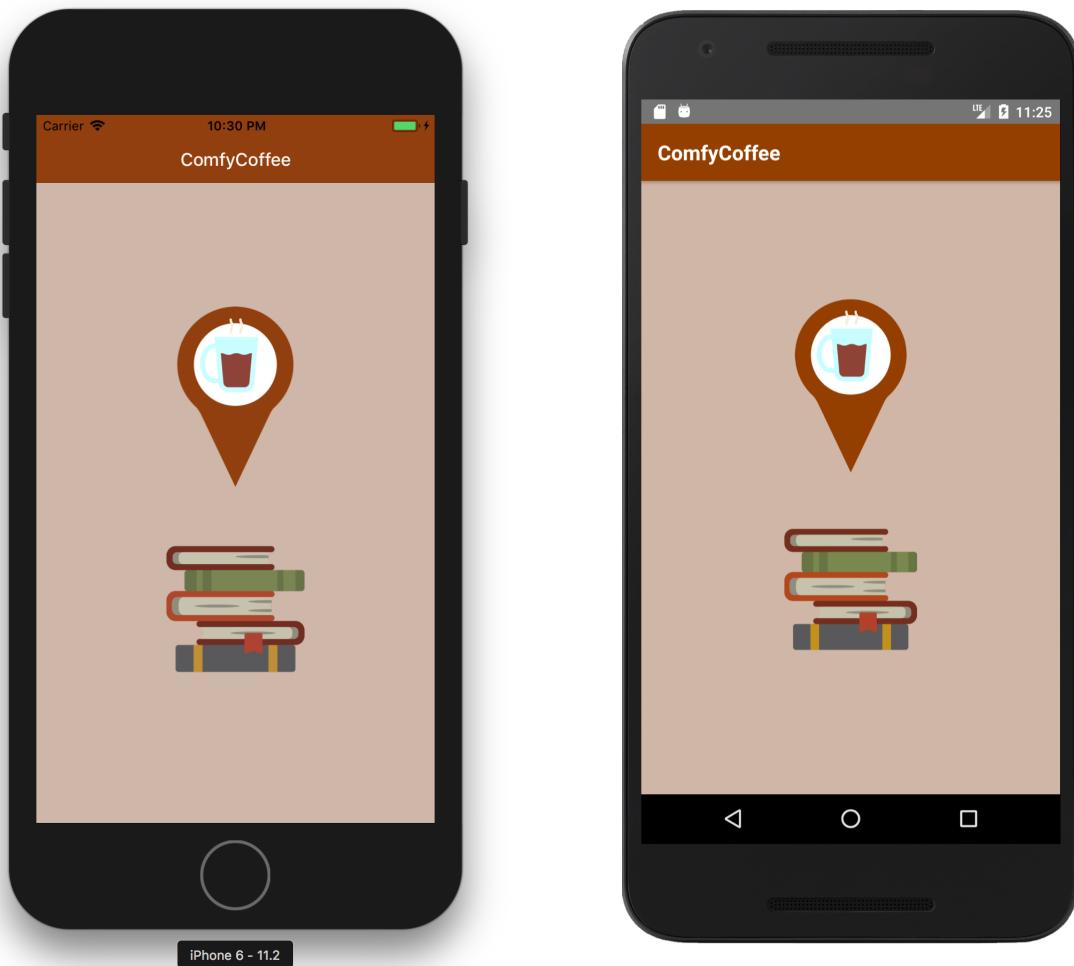


Abbildung 3.1: Startseite der App unter iOS Abbildung 3.2: Startseite der App unter Android

3.5.2 Kartenansicht

Hier werden auf einer interaktiven Google Maps Karte alle sich in der Nähe befindlichen Cafés einschließlich des eigenen aktuellen Standorts angezeigt (vgl. Abb. 3.3 und 3.4). Die hierfür benötigen Informationen werden über die entsprechenden Google APIs bezogen. Des Weiteren erreicht man von hier aus über die Tab Navigation in Android sowie die Bottom Navigation in iOS die Listenansicht der angezeigten Cafés.

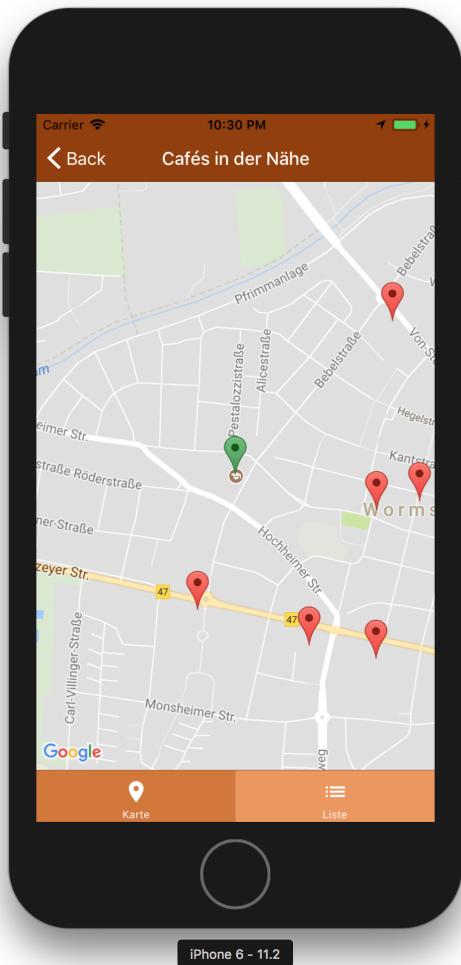


Abbildung 3.3: Kartenansicht der App unter iOS

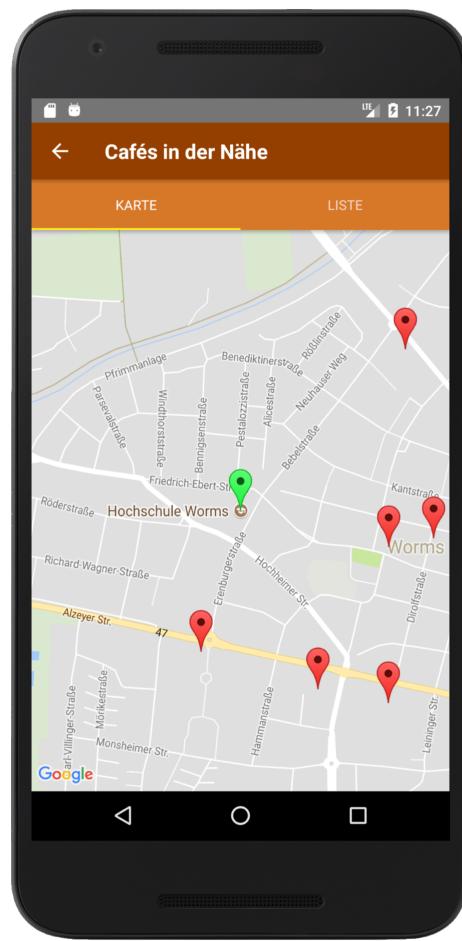


Abbildung 3.4: Kartenansicht der App unter Android

3.5.3 Kartenansicht - ausgewählt

Ein Klick auf einen Marker öffnet ein *Bottom Sheet*, in welchem die wichtigsten Informationen über das ausgewählte Café wie etwa Entfernung, Bewertung, Öffnungszeiten oder Adresse angezeigt werden. Innerhalb dieses Sheets stehen Buttons zur Navigation auf die Detailseite des Cafés oder direkt zu Google Maps mit dem Café als hinterlegtem Ziel zur Verfügung (vgl. Abb. 3.5 und 3.6).

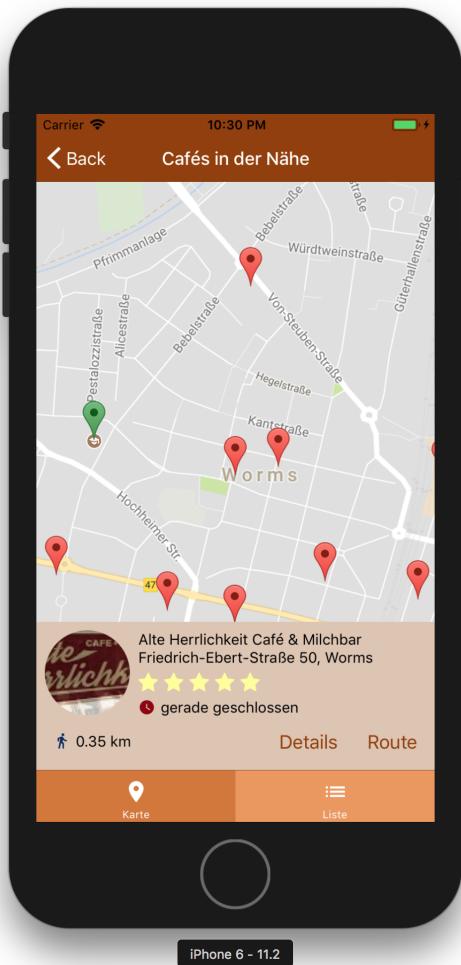


Abbildung 3.5: Kartenansicht (ausgewählt) der App unter iOS

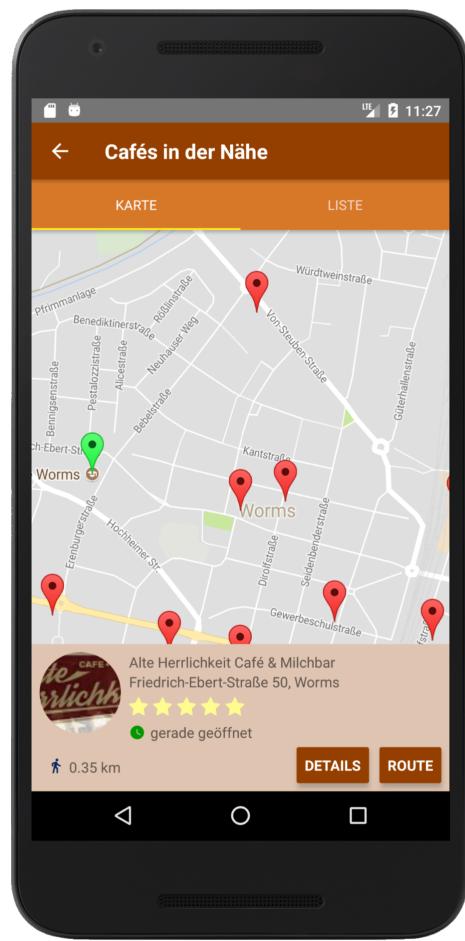


Abbildung 3.6: Kartenansicht (ausgewählt) der App unter Android

3.5.4 Listenansicht

Die bereits in der Kartenansicht dargestellten Standorte der Cafés werden hier in einer übersichtlichen Liste gezeigt (vgl. Abb. 3.7 und 3.8). Jeder Listeneintrag enthält die gleichen Informationen und Interaktionsmöglichkeiten wie das Bottom Sheet aus der Karte, da es sich hierbei technisch um die gleiche, eigens konstruierte React Native Komponente handelt.

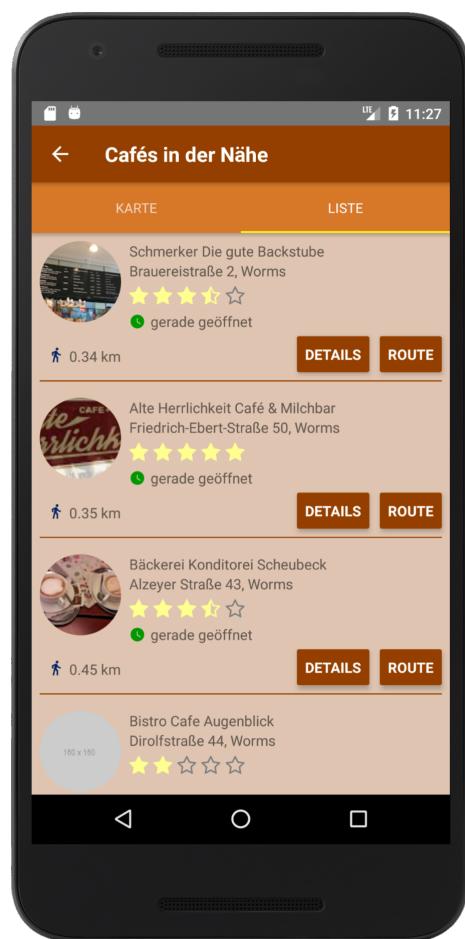
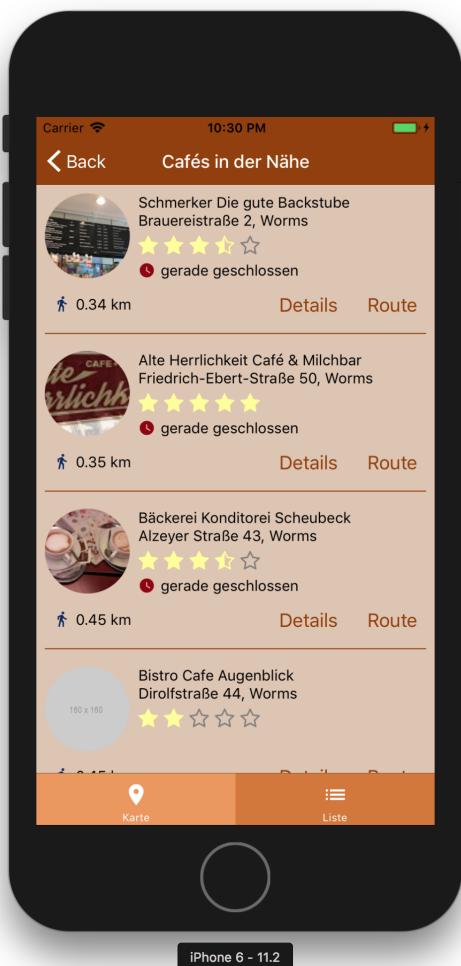


Abbildung 3.7: Listenansicht der App unter iOS

Abbildung 3.8: Listenansicht der App unter Android

3.5.5 Infos Café

Diese Seite zeigt die bereits im Bottom Sheet vorhandenen Informationen über das ausgewählte Café. Außerdem sind weitere Details wie Telefonnummer, Internetadresse oder auch Bilder des Cafés - soweit über die Google API bereit gestellt - verfügbar (vgl. Abb. 3.9 und 3.10). Auch hier erreicht man über einen Button das Café als Zieladresse direkt in Google Maps. Darüber hinaus enthält die Tab Navigation respektive Bottom Navigation einen Link zur Bewertungsseite des Cafés.

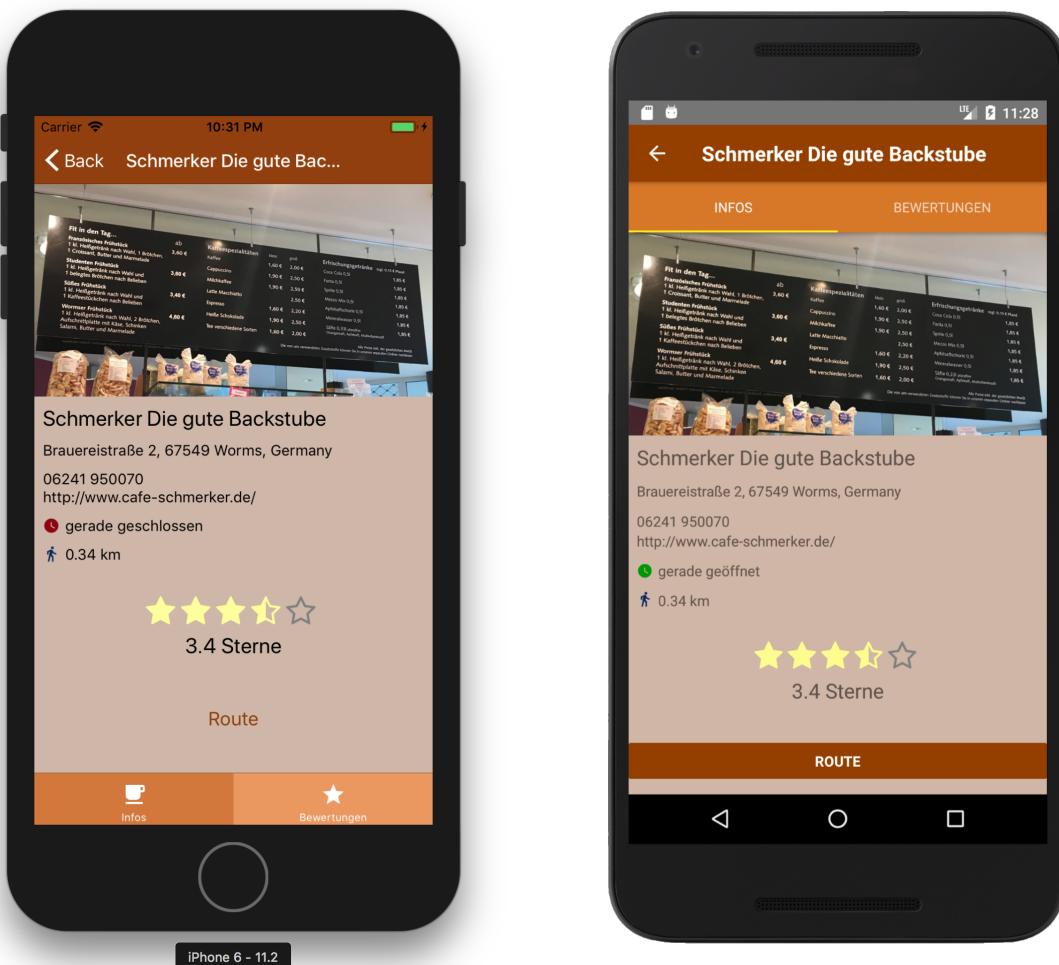


Abbildung 3.9: Infoseite eines Cafés der App unter iOS

Abbildung 3.10: Infoseite eines Cafés der App unter Android

3.5.6 Bewertungen Café

Alle über die Google API verfügbaren Bewertungen des ausgewählten Cafés werden hier in einer Liste angezeigt. Dabei erhält man Infos wie den Wert der abgegebenen Bewertung, den Zeitraum, den Bewertungstext sowie den Namen des Rezensenten (vgl. Abb. 3.11 und 3.12).

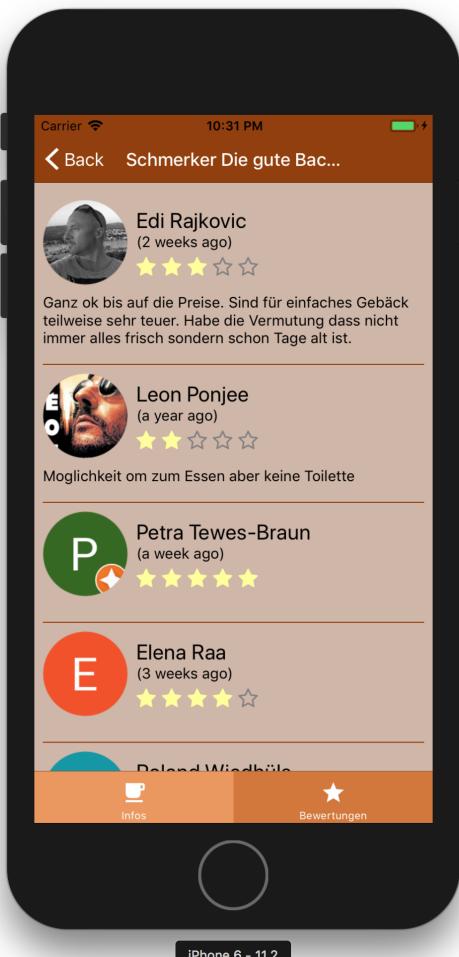


Abbildung 3.11: Bewertungsseite eines Cafés der App unter iOS

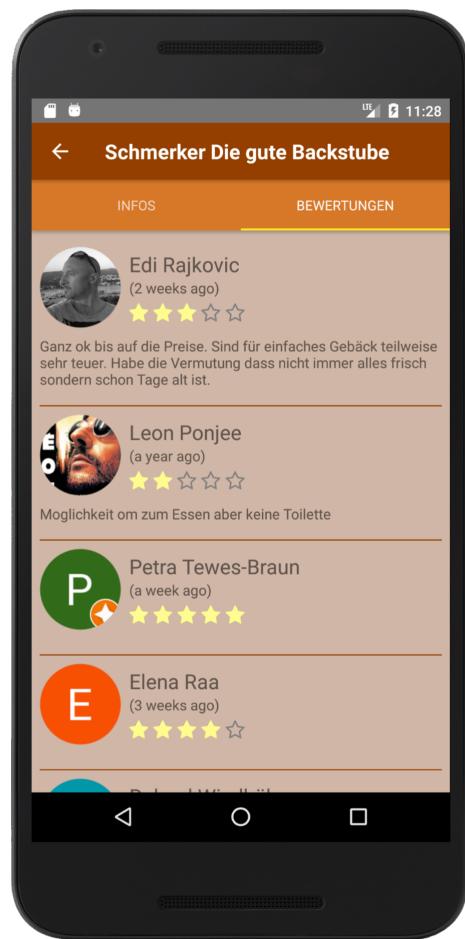


Abbildung 3.12: Bewertungsseite eines Cafés der App unter Android

3.5.7 Lexikon

Die Übersichtsseite des Lexikons zeigt alle in der App hinterlegten Zubereitungsarten mit Namen und Icon (vgl. Abb 3.13 und 3.14). Ein Klick auf eines dieser Elemente führt auf die zugehörige Detailseite (vgl. Kapitel 3.5.8). Außerdem ist dies die einzige Seite, auf welcher der Bewegungssensor aktiv ist. Wird eine Schüttelgeste ausgeführt, so wird eine zufällig ausgewählte Zubereitungsart in einem Alert angezeigt. Von diesem aus kann man ebenfalls entweder direkt zur vorgeschlagenen Detailseite gelangen oder den Alert wieder entfernen.

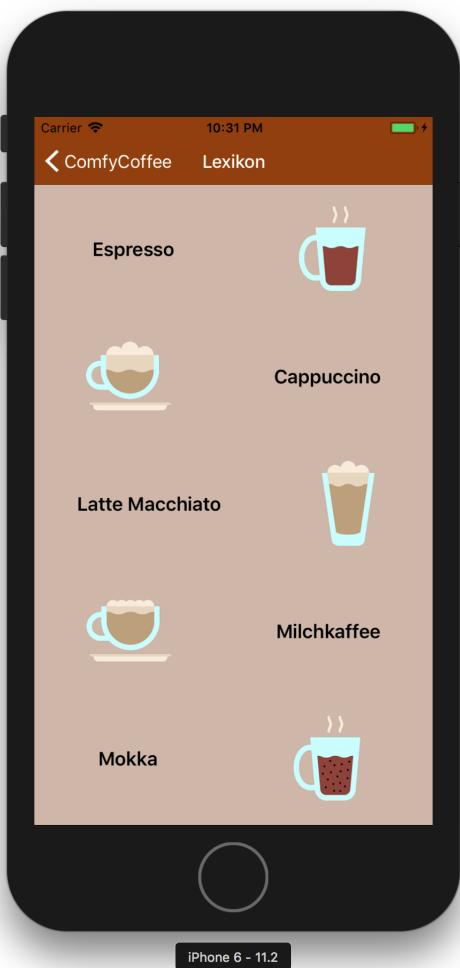


Abbildung 3.13: Lexikonseite der App unter iOS

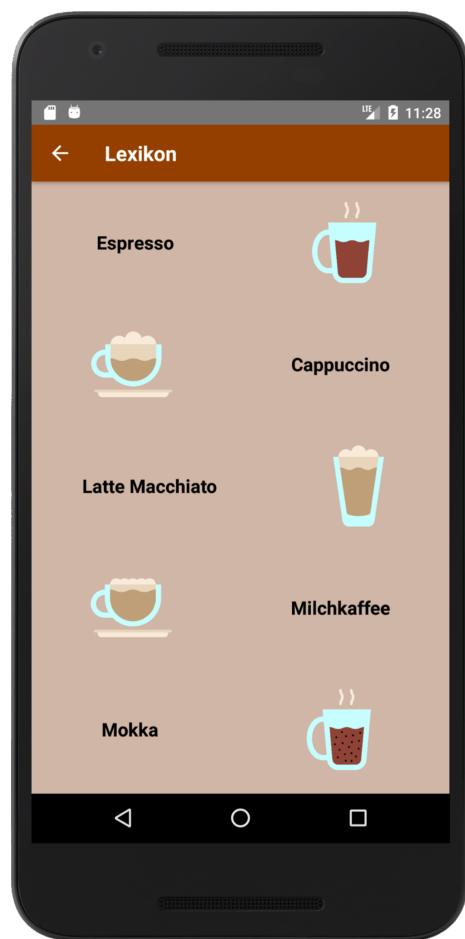


Abbildung 3.14: Lexikonseite der App unter Android

3.5.8 Lexikon Detailansicht

Hat man sich für eine Zubereitungsart entschieden, erscheint das Template dieser Seite gefüllt mit den passenden Informationen. Dazu gehören der Name, das Icon, eine Beschreibung des Ablaufs sowie einige Attribute wie Stärke des Getränks oder Menge des benötigten Wassers. Diese Informationen sind an die jeweils ausgewählte Zubereitungsart angepasst (vgl. Abb 3.15 und 3.16).



Abbildung 3.15: Detailansicht eines Lexikoneintrags der App unter iOS



Abbildung 3.16: Detailansicht eines Lexikoneintrags der App unter Android

3.6 Readme

comfycoffee ist eine Cross-Plattform App zur Anzeige lokaler Cafés und damit nachhaltigen Unterstützung der lokalen Kaffeewirtschaft. Das Repository ist öffentlich und unter folgendem Link erreichbar: <https://bitbucket.org/codemeleon/comfycoffee>. Das Projekt ist lizenziert unter der MIT Lizenz.

Weitere Informationen können der offiziellen React Native Dokumentation entnommen werden.

3.6.1 Voraussetzungen

Um an dem Projekt zu arbeiten müssen folgende Tools installiert sein:

- node.js (v8.6.0 wird empfohlen)
- npm (v5.3.0 wird empfohlen)
- Watchman
- Xcode (für iOS Builds - nur für MacOS)
- Android Studio (für Android Builds)

Es wird die Verwendung von *Visual Studio Code (VS Code)* als Code Editor empfohlen. Das VS Code Plugin *Prettier - Code formatter* sorgt hierbei für einen sauber formatierten Code. Weiterhin muss eine Verbindung mit dem Internet bestehen, um das Projekt bauen zu können.

3.6.2 Testing und Debugging

Das Testing und Debugging kann über die in Android Studio und Xcode integrierten Simulatoren erfolgen. Eine weitere Möglichkeit ist die direkte Verbindung zu einem Smartphone/iPhone über USB. Wird ein derartiges Endgerät erkannt erfolgt die Kompilierung direkt auf das Gerät. Hilfreiche Features wie JavaScript Debugging im Browser oder LiveReload können direkt im Simulator oder auf dem Endgerät aktiviert werden.

Relevante Befehle in der Kommandozeile sind hierbei:

react-native run-android

react-native run-ios

4 Ausblick

Während und nach der Entwicklung sollen Tests sicher stellen, dass die App gut bedienbar ist. Das Testkonzept hierfür wird in Kapitel 4.1 erläutert.

Anschließend spielt das Marketing eine wichtige Rolle, um die App erfolgreich unter die Zielgruppe zu bringen. Ansätze und Ideen hierfür sind in Kapitel 4.2 dargelegt.

4.1 Testkonzept

Das Testkonzept besteht aus einem *Usability Test*, der zusätzlich ein *Interview* und einen *A/B-Test* beinhaltet.

Für den Usability Test werden 10 bis 20 Versuchsteilnehmer aus der Zielgruppe benötigt. Unter Beobachtung sollen sie die App bedienen und mit ihr bestimmte Aufgaben lösen. Währenddessen sollen die Teilnehmer laut aussprechen, was sie gerade denken und tun (*Thinking Aloud*). So lassen sich unlogische Abläufe und unklare Bedienungen in der App herausfinden.

Das Interview wird als halb-strukturiertes Interview durchgeführt. Offene Fragen, die sich auf die App im Allgemeinen beziehen, lassen der Versuchsperson die Freiheit, ihre ganz persönlichen Eindrücke wiederzugeben. Diese Ergebnisse sind schwer zu vereinheitlichen und müssen mit entsprechender Umsicht behandelt werden, da sie sehr subjektive Empfindungen beinhalten, die sich unter Umständen nicht mit einem Großteil der Zielgruppe decken. Es muss stets abgewogen werden, ob und wie die Ergebnisse der offenen Fragen umgesetzt werden. Der geschlossene Teil des Interviews bezieht sich auf den A/B-Test im folgenden Abschnitt.

Beim A/B-Test wird ein Bestandteil der App in zwei unterschiedlichen Ausführungen getestet. Die Versuchspersonen werden in zwei Gruppen aufgeteilt, eine Gruppe erhält die App mit dem Bestandteil in der Version A, die andere die gleiche App, die sich nur darin unterscheidet, dass Bestandteil A durch B ausgetauscht wird. Im Fall von *comfycoffee* wäre zu testen, in welcher Form die Schüttelfunktion besser bei der Zielgruppe ankommt. Ursprünglich weist die Schüttelfunktion den Nutzer auf eine zufällige Kaffeesorte hin (vgl. Kapitel 2.3.1). Eine weitere Idee ist, über die Schüttelfunktion zufällig auszuwürfeln, wer von zwei oder mehreren Nutzern den Kaffee bezahlt. Präzise geschlossene Fragen sollen darüber Auskunft geben, welche Funktion von der Zielgruppe besser angenommen wird.

Learnings: Auch nach der Entwicklung ist noch viel zu tun. Regelmäßiges Testen erhöht die Usability der App und bringt den Entwickler auf den Boden der Tatsachen zurück.

4.2 Marketing

Beim Marketing wurden die Bereiche *Produktpolitik*, *Preispolitik*, *Distributionspolitik*, *Kommunikationspolitik* und *Retention Marketing* ausgearbeitet. Anhand von Analyse-tools, wie beispielsweise *Google Analytics*, können Statistiken zur App-Nutzung erfasst werden, anhand derer die App ausgewertet und weiter verbessert werden kann.

Learnings: Marketing ist ein eigenes Themengebiet mit vielen Facetten, die es zu meistern gilt, um eine App wirkungsvoll unter die Zielgruppe zu bringen. Hier wird viel Einarbeitung und Ausdauer verlangt, um erfolgreich zu sein.

4.2.1 Produktpolitik

comfycoffee wird als Cross-Plattform App für Android und iOS entwickelt. Eine zusätzliche Web-App für Desktopnutzer ist denkbar.

4.2.2 Preispolitik

comfycoffee wird als kostenfreier Download in den App-Stores zur Verfügung stehen. Cafés können einen höheren Platz in der Listenansicht der Cafés erkaufen, diese Einträge werden für den Nutzer kenntlich mit “Anzeige” markiert. Mit den Daten der Nutzer sowie der Cafés kann personalisierte Werbung innerhalb der App geschaltet werden.

Eine weitere Idee wäre, die App für einen geringen Preis in die App-Stores zu stellen. Dem Nutzer wird der Kauf schmackhaft gemacht, indem er Gutscheine von Cafés durch die App erhält, welche die Kosten für die App wieder aufwiegen.

4.2.3 Distributionspolitik

Die App wird in den App-Stores (*Google Play* für Android und *App Store* für iOS) sowie über den Browser (Web-App) zur Verfügung gestellt.

4.2.4 Kommunikationspolitik

Eine dedizierte Landing-Page ist die erste Anlaufstelle für comfycoffee. Weitere Social Pages dienen ebendiesem Zweck. Durch Social Media Werbung, besonders in “Support your Locals”-Communities und durch Influencer, soll die Bekanntheit der App gesteigert werden.

Ein eigener Blog soll die Nutzer informieren und neue Nutzer gewinnen. Ein “Kaffee-Reporter” fungiert als Gesicht für die App und stellt Themen wie der “Coffee of the Week”, die “Location of the Week” oder aktuelle Kaffee-Trends vor.

Offline-Werbung in Cafés vor Ort und in Zeitungen machen die App auch in Kreisen bekannt, die weniger in den Sozialen Medien unterwegs sind.

4.2.5 Retention Marketing

comfycoffee soll die Nutzer motivieren, die App regelmäßig zu nutzen. Hierfür können mit den Cafébesitzern, deren Cafés in der App aufgelistet werden, Gutscheine ausgehandelt werden, welche die Nutzer über die App erhalten und einlösen können.

Die Nutzer können durch den Besuch ihres Lieblingscafés Punkte erhalten, anhand derer sie in einer Rangliste nach oben steigen können. Die Top 10 der Besucher des jeweiligen Cafés werden in der App aufgelistet. Alternativ können die Nutzer Punkte erlangen, indem sie möglichst viele verschiedene Cafés besuchen. Durch diese Art der Gamification sollen die Nutzer Spaß an der App haben und diese durch den Ehrgeiz der Punktegewinnung möglichst oft nutzen.

Innerhalb der App kann auf bestimmte seltene Kaffeesorten aufmerksam gemacht werden, z. B. wenn sich der Nutzer gerade in der Nähe eines Cafés befindet, welches diese Sorte anbietet. Aktuelle Kaffee-Trends können zeitlich begrenzt in der App angesehen werden. Durch den Besuch eines Cafés können weitere Einträge im Lexikon freigeschaltet werden. Eine zeitliche Limitierung dieser Einträge fordert den Nutzer dazu auf, comfycoffee häufig zu nutzen.

5 Fazit

Die Konzeption und Umsetzung einer Cross-Plattform-App war für uns Neuland, ebenso wie das Ausarbeiten eines Marketingkonzepts (vgl. Kapitel 4.2). Im Folgenden sind unsere gewonnenen Erkenntnisse und ein Fazit zum Entwicklungsframework React Native sowie der Cross-Plattform-Entwicklung im Vergleich zur nativen Entwicklung aufgeführt.

5.1 Lessons Learned

comfycoffee war die erste Cross-Plattform App, die wir entwickelt haben. Auch mit dem Framework React Native haben wir unsere ersten Schritte gemacht (vgl. Kapitel 3.1). Dabei haben wir den Umgang mit APIs vertieft. Das Umsetzen von nativem Design in Android und iOS entsprechend den jeweiligen Design Guidelines birgt einige Herausforderungen. So haben Android und iOS teilweise die gleiche Bezeichnung für verschiedene Elemente, oder unterschiedliche Elemente mit dem gleichen Namen. Auch das Ansprechen von Hardwareschnittstellen (GPS, Bewegungssensor) war neu. Zu verstehen, wie diese funktionieren und benutzt werden können, bedeutet oft viel lesen und, je nach Aussagekraft der Dokumentationen, noch mehr ausprobieren.

Herausforderungen waren auch gegeben bei der Benutzung eines weniger etablierten Frameworks wie React Native (vgl. Kapitel 5.2).

Neu war außerdem die technische Konzeption mit der Implementierung der zuvor ausgearbeiteten, technischen Anforderungen. Die technische Konzeption steht der Produktkonzeption gegenüber, bei welcher die Zielgruppen, das Design und das Testkonzept ausgearbeitet sowie Marktanalysen durchgeführt werden. Den Teil der Produktkonzeption wurde schon in dem Modul Mobile Usability behandelt. Mit comfycoffee die Verbindung zwischen Produkt- und technischer Konzeption und Umsetzung herzustellen war eine spannende Herausforderung.

5.2 React Native

Das Cross-Plattform Framework React Native schlägt die Brücke zwischen Web- und mobiler Entwicklung. Durch die eingesetzten Webtechnologien wie JavaScript fiel uns durch unsere Affinität zur Webentwicklung der Einstieg nicht schwer. Dennoch hat

React Native einige Nachteile bei der Entwicklung. Das Debugging von CSS beispielsweise ist im Simulator nicht möglich. Weiterhin bestehen native Abhängigkeiten trotz der Cross-Plattform Eigenschaften des Frameworks. So mussten Libraries eingebunden und die Environment-Dateien separat per Hand konfiguriert werden. Es stehen online viele Module zur Erweiterung und Einbindung weiterer Elemente in React Native zur Verfügung, jedoch sind diese nicht immer ganz ausgereift oder gut dokumentiert.

5.3 Cross-Plattform vs. Nativ

Die Cross-Plattform Entwicklung mit React Native ist angelehnt an die Webentwicklung. Dies bietet gute Designmöglichkeiten dank CSS- und HTML-ähnlicher Strukturen. Der Hardwarezugriff ist dennoch gegeben und leicht anwendbar. Für eine kleine App wie comfycoffee wäre eine native Entwicklung für Android und iOS zu aufwändig gewesen. Die Vorteile der Cross-Platform Entwicklung - die Nutzung von Webtechnologien, die Designmöglichkeiten und der Hardwarezugriff - wiegen vor allem im Fall von comfycoffee die Nachteile auf. Die Entscheidung, womit eine App entwickelt werden soll, hängt zweifelsohne immer von den Anforderungen und Zielen der App ab. Wenn es machbar ist, würden wir jedoch lieber Cross-Platform bzw. hybrid statt nativ entwickeln.

Eidesstattliche Erklärung

Hiermit versichern wir, dass wir die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt haben. Alle Stellen, die wörtlich oder sinngemäß aus anderen Schriften entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit ist noch nicht veröffentlicht oder anderweitig für Prüfungszwecke vorgelegt worden.

Worms, den 31.01.2018