Dokumentation

Begleitendes Projekt
FH Campus 02
Business Software Development
SS 2021

Gruppe 2
Theresa Dietinger
Georg Praßl
Florian Reisinger
Denis Schüle
Günther Walenta

Datum	Version	AutorInnen	Beschreibung
08.03.2021	0.1	Georg Praßl	Initiale Erstellung
15.03.2021	0.2	Denis Schüle	Use Case Erweiterung
20.03.2021	0.3	Georg Prassl	Kommunikations Protokoll
20.03.2021	0.4	Theresa Dietinger	Mock Up Screens
20.03.2021	0.5	Günther Walenta	Prüfung Entwicklung, Client, Backend
20.03.2021	0.6	Florian Reisinger	Sicherheitskonzept
20.03.2021	0.7	Denis Schüle	Offline Online Verfügbarkeit von Use Cases
21.03.2021	1.0	Denis Schüle	Abgabe 01
22.03.2021	1.1	Günther Walenta	Adaptionen nach erster Abgabe
25.04.2021	1.2	Denis Schüle	Preabgabe - Finalisierung
14.05.2021	1.3	Günther Walenta	Deployment
15.05.2021	1.4	Theresa Dietinger	Problem- und Applikationsbeschreibung
16.05.2021	1.5	Theresa Dietinger	Gesamtkorrektur
17.05.2021	1.6	Theresa Dietinger	App-Screenshots
17.05.2021	1.7	Denis Schüle	Erweiterung Umsetzung, Finalisierung

Inhalt

1. Einleitung

1.1 Problembeschreibung

Der Boom des Kletterns, ins Besondere des Bouldern, hat in den letzten Jahren auch Österreich erreicht. Auf Grund dieses Trends wurden mehrere Boulderhallen, darunter Bloc House, Boulderclub und Newton in Graz eröffnet. Diese große Auswahl an Boulderhallen spiegelt die Nachfrage an unterschiedlichsten Boulderrouten wider. Hier tritt das Problem auf, dass der Überblick über den aktuellen Stand eines Boulderers schnell verloren geht. Es ist schwierig sich zu merken in welcher Halle Boulder bereits versucht oder geschafft wurden – umso schwieriger ist es sich mit seinen Freunden aktuell zu halten. Um dieses Problem zu lösen wird die mobile Applikation „Skill Crack“ vorgestellt.

1.2 Applikationsbeschreibung

Die Applikation “Skill-Crack" dient zum Erfassen von Bouldern und Vergleichen und Messen des persönlichen Fortschritts mit anderen Nutzern. Dabei ist es möglich sich als Nutzer zu registrieren und anzumelden. Angemeldete Nutzer können Boulder erfassen, erfasste Boulder auswählen und Interaktionen mit diesen Bouldern, erfassen. Nutzer können die Interaktionen anderer Nutzer einsehen und mit einem Like-Button und einer Kommentarfunktion darauf reagieren. Somit ist es möglich den Überblick über das aktuelle Geschehen im Sport von sich selbst und seinen Freunden zu behalten.

Die Idee zur Applikation stammt von Theresa Dietinger, welche selbst eine leidenschaftliche Boulderin ist und eine App, um den aktuellen Stand einzusehen, für sinnvoll hält. Daher verfügt die Gruppe über das nötige Domainwissen, um die Applikation auf die Nutzergruppe auszurichten. Den Entschluss die Idee der Applikation umzusetzen wurde in der Gruppe via Mehrheits-Entscheidung getroffen.

1.3 Technologie Auswahl

1.3.1 Front-End

Als Programmiersprache wird im Front-End TypeScript eingesetzt. Bei der Technologie Auswahl setzen wir im Front-End auf React Native[1]. Die Auswahl begründet sich durch die von mehreren Mitgliedern bereits im beruflichen Feld gesammelten Erfahrung mit diesem Framework sowie dieser Programmiersprache.

1.3.2 Back-End

Im Back-End wird ein Node.js[2] Server eingesetzt, welcher die Daten in einer Free Hosting Datenbank persistiert. Diese Kombination ist vom Team gewählt, da bereits in einer im beruflichen Umfeld gute Erfahrungen mit dieser gemacht wurden. Zudem kann durch diese Auswahl im Back-End dieselbe Programmiersprache wie im Front-End verwendet werden. Die Authentifizierung wird über den allgemein gültigen Standard OAuth 2.0[3] durchgeführt, da diese von Experten als sicher angesehen wird.

[1] https://reactnative.dev/
[2] https://nodejs.org/en/
[3] https://oauth.net/2/

1.3.3 Entwicklung

Für die Entwicklung der Applikation werden Visual Studio Code[4], IntelliJ Idea[5] und Android Studio[6] eingesetzt, der Code wird in einem Github Repository[7] (https://github.com/Greeny1992/GMC_SS21_BoulderApp) gesichert. Zum Testen der Anwendung auf mobilen Endgeräten wird expo.io[8] bzw. ein Emulator in der Android Studio Umgebung eingesetzt. Die Dokumentation ist in diesem Dokument mithilfe von Microsoft Word erstellt.

1.3.4 Plattform Ausrichtung

Generell sollte die Applikation auf beiden Plattformen (Android & IOS) bereitgestellt werden. Im Zuge dieser Arbeit wird die Applikation Android kompatibel implementiert. Speziell wird der Fokus auf Handys und weniger auf Tablets gelegt. Da die Zielgruppe tendenziell Handys präferiert.

1.4 Zweck des Dokumentes

Das Dokument dient als Dokumentation der Applikation “Skill-Crack", welche im Zuge der Lehrveranstaltung “Grundlagen Mobile Computing” erstellt worden ist. Neben der Technologieauswahl werden die Ziele und Use-Cases beschrieben. Des Weiteren werden die Herausforderung der mobilen App Entwicklung: Datenübertragung, Datensicherung, Kollisionsfreiheit, der Betrieb der Applikation (Testen & Entwicklung) im Zusammenhang mit “Skill-Crack” betrachtet.

1.5 Umfang des technischen Prototypens

Im Rahmen des in der LV umgesetzten Projektes werden folgende Aspekte beschrieben, jedoch nicht implementiert:

- Authentifikation
- Registrierungs-Prozess
- Change-Management für Boulder-Editierung

2. Ziele & nicht Ziele

2.1 Ziele

2.1.1 Nutzer Management Registration

Der Nutzer soll die Möglichkeit haben sich zu registrieren, um es zu ermöglichen userspezifische Informationen zu speichern und ihm in weiterer Folge zur Verfügung zu stellen - dies wird durch einen Dialog bestätigt.

2.1.2 Nutzer Management Anmeldung

Durch die Anmeldung erhält der Nutzer Zugang zu seinen spezifischen Informationen, welche er in der Applikation erfasst hat.

2.1.3 Nutzer Management Abmeldung

Der Nutzer soll die Möglichkeit haben sich abzumelden, dadurch wird die Nutzer-Session geschlossen.

[4] https://code.visualstudio.com/
[5] https://www.jetbrains.com/de-de/idea/
[6] https://developer.android.com/studio
[7] https://github.com/
[8] https://expo.io/

2.1.4 Boulder erfassen

Der Nutzer soll die Möglichkeit haben neue Boulder mit dazugehörigen Metadaten zu erfassen.

2.1.5 Boulder auswählen

Dem Nutzer sollen erfasste Boulder angezeigt werden, die Auflistung soll nach Schwierigkeitsgrad, Location und Farbe gefiltert werden können.

2.1.6 Boulder-Aktion erfassen

Der Nutzer soll die Möglichkeit haben Interaktionen mit erfassten Bouldern zu erstellen. Dabei kann er den Status (Like/Unlike) und einen Kommentar hinterlegen.

2.1.7 Boulder-Aktion andere Nutzer einsehen und interagieren

Der Nutzer soll die Möglichkeit haben Interaktionen von anderen Nutzern einzusehen und mittels Like-Button darauf reagieren.

2.2 Nicht Ziele

2.2.1 Direkte Kommunikation zwischen Nutzern

In der Applikation soll es keine direkte Kommunikation zwischen Nutzern geben (z.B. Chat-Funktion).

3. Konzept

3.1 Übersicht

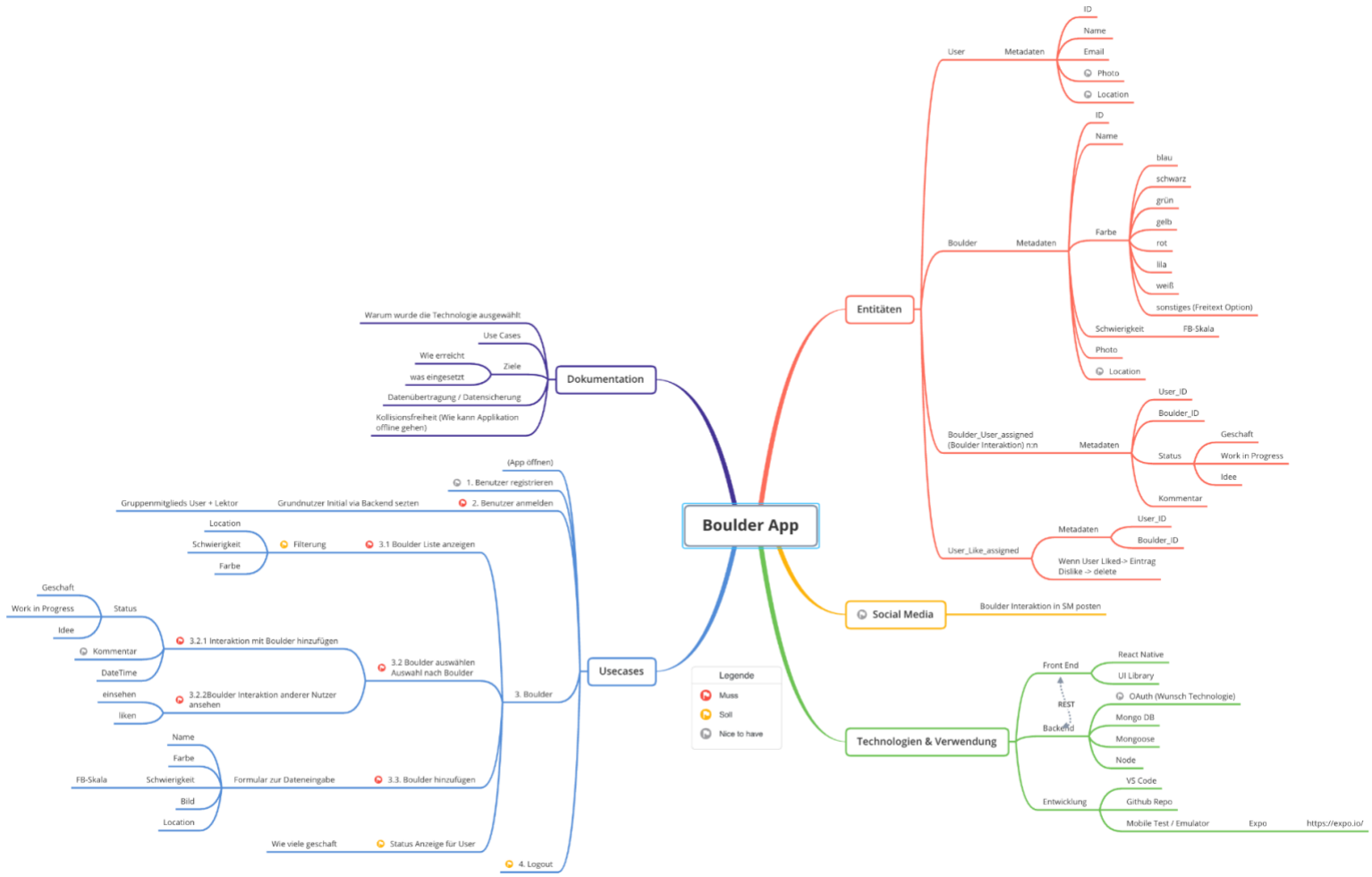

Abbildung 1 Konzept

3.2 Mock Up Screens

3.2.1 Home Screen

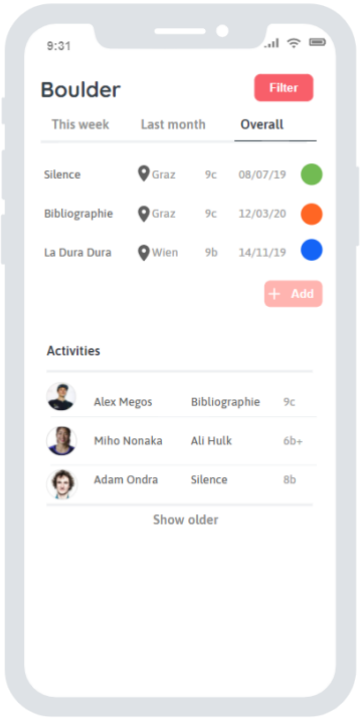

Abbildung 2 Mock-Up-Screen Home Screen

3.2.2 Neuen Boulder hinzufügen

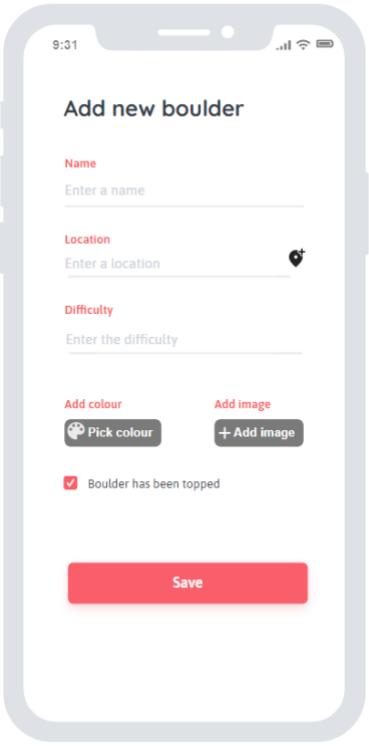

Abbildung 3 Mock-Up-Screen Boulder hinzufügen

3.2.3 Zusammenhang der Screens

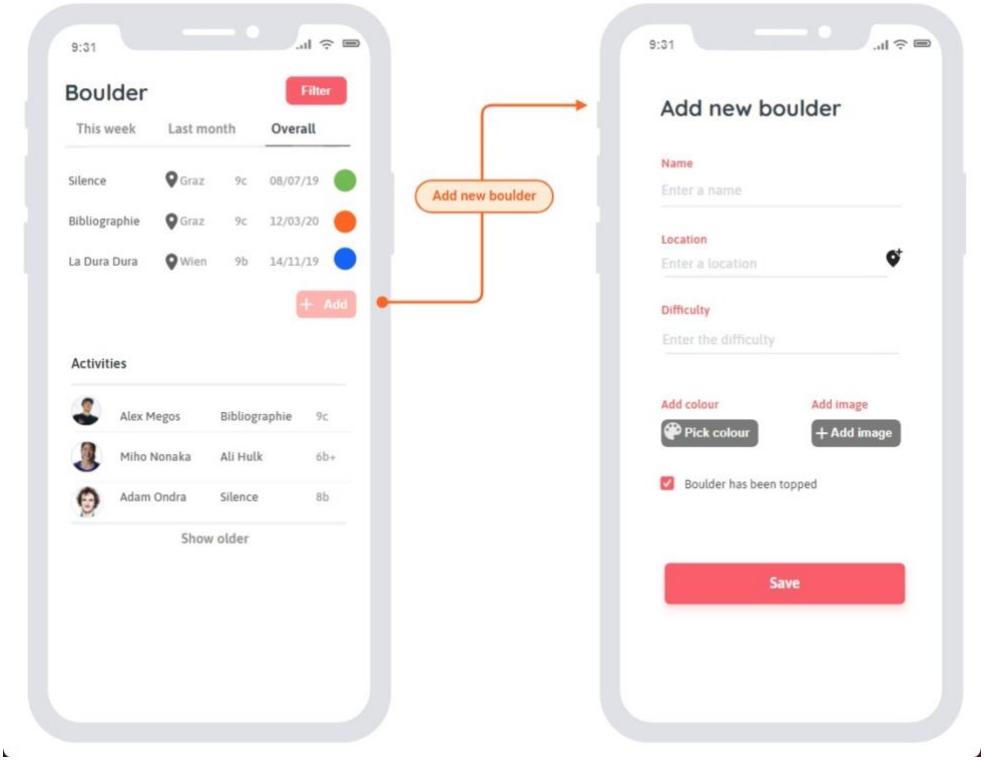

Abbildung 4 Mock-Up-Screen neuer Boulder

3.3 Datenmodell

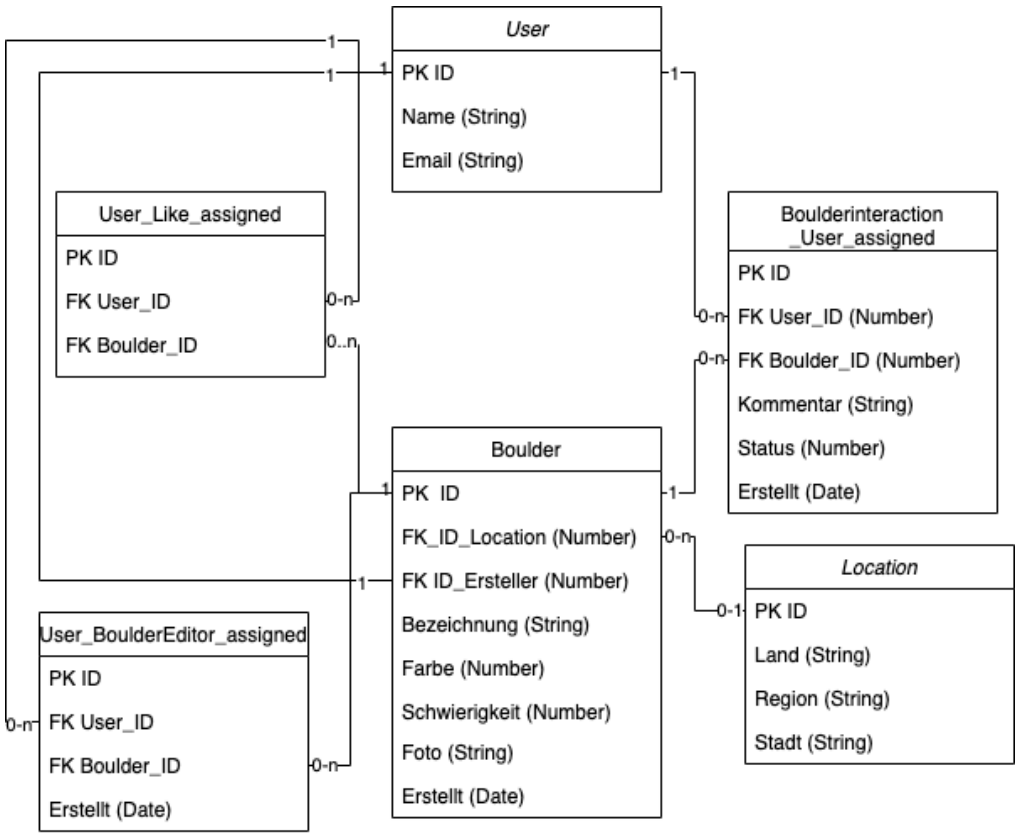

Abbildung 5 Datenmodell

4. Use-Cases

4.1 Übersicht

Tabelle 1 Use-Case Übersicht & Verfügbarkeit

Use-Case	Online-Verfügbarkeit	Offline-Verfügbarkeit	Optional
01_UC-Benutzer Registrieren	x		
02_UC-Benutzer anmelden	x	x	
03_UC-Boulder Liste anzeigen	x	x	
04_UC-Boulder Interaktion auswählen	x		
05_UC-Boulder auswählen	x	x	
06_UC-Interaktion mit Boulder hinzufügen	x		
07_UC-Boulder Interaktion anderer Nutzer ansehen	x	(x)	
08_UC-Boulder hinzufügen	x	x	
09_UC-Benutzer abmelden	x	(x)	
10_UC-Boulder bearbeiten (Kollision im Backend)	x	x	
11_UC-Boulder liken	(x)	(x)	x

(x) -> zusätzlich umgesetzt

4.2 Use-Case Beschreibung

01_UC-Benutzer registrieren

Verfügbar: nur Online

Beschreibung:

Ausgehend vom Startbildschirm der Applikation hat der Nutzer die Möglichkeit sich zu registrieren. Durch einen Button-Klick öffnet sich das Registrierungsformular. Nach der Eingabe folgender Nutzer-Metadaten:

- Name – Textfeld, String
- Passwort – Textfeld, String
- E-Mail-Adresse – Textfeld, String, Überprüfung auf Mail-Format
- Location (Land und Stadt) - Dropdown / Location Libary
- Foto - Upload

wird das Formular validiert, sollte der Nutzer bereits existieren wird dieser auf seinen bestehenden Account hingewiesen, ansonsten wird der Nutzer über die erfolgreiche Registration informiert und auf den Anmeldescreen weitergeleitet.

Bemerkung: wird im Zuge der LV nicht implementiert

02_UC-Benutzer anmelden

Verfügbar: Online & Offline

Offline Voraussetzungen:

Benutzer war mind. einmal angemeldet, nicht abgemeldet und hat seinen Speicher nicht gecleart.

Beschreibung:

Der Nutzer kann vom Startscreen der Applikation mittels Button Klick, oder nach erfolgreicher Registration, sein Nutzernamen und Passwort im Anmeldeformular eintragen. Nach positiver Validierung der Eingabe, wird der Nutzer auf seinen persönlichen Homescreen der Applikation weitergeleitet, sollte die Validierung

fehlgeschlagen sein, wird das Formular zurückgesetzt und der Nutzer wird über den fehlgeschlagenen Anmeldeversuch informiert.

03_UC-Boulder Liste anzeigen

Verfügbar: online & offline

Offline Voraussetzungen:

Benutzer war mind. einmal angemeldet und hat die Liste geladen.

Offline Einschränkungen:

Stand der Boulder-Liste entspricht dem, des letzten Logins.
Es werden nur jene Boulder lokal gespeichert, mit welchen der User interagiert hat:

- erstellt
- bearbeitet
- Boulder Interaktion erfasst

Lokal gespeicherte Daten werden bei der nächsten Netzwerkverbindung mit dem Backend synchronisiert. Hat es Änderungen gegeben wird der User darüber informiert und er hat die Möglichkeit diese zu überschreiben oder seine Änderungen zu verwerfen.

Voraussetzung:

- erstmalige Registration

Beschreibung:

Am persönlichen Homescreen der Anwendung wird dem Nutzer eine Liste der erfassten Boulder angezeigt. Der Nutzer hat die Möglichkeit die Liste mithilfe von Dropdowns nach Location (Land und Stadt) - Dropdown/ Location Library zu filtern. Durch Selektion der Filter wird die Liste entsprechend gefiltert. Sind noch keine Boulder erfasst wird eine entsprechende Information angezeigt.
Die generelle Boulder-Liste, welche dem Nutzer angezeigt wird, wird am Endgerät gespeichert, um sie für eine Offline-Nutzung zur Verfügung zu stellen.

04_UC-Boulder-Interaktionen auswählen

Verfügbarkeit: nur online

Voraussetzung:

- erstmalige Registration
- Nutzer ist angemeldet

Beschreibung:

Am persönlichen Homescreen der Anwendung wird dem Nutzer eine Liste der (eigens) erfassten Boulder-Interaktionen angezeigt, dabei handelt es sich um die letzten drei Aktivitäten, durch Klick darauf kann der Nutzer die Detailinformationen ansehen und hat die Möglichkeit direkt auf die Detailansicht des dazugehörigen Boulder zu wechseln.

05_UC-Boulder auswählen

Verfügbarkeit: online & offline

Offline Voraussetzungen:

Benutzer war mind. einmal angemeldet und hat die Boulderliste Liste geladen.

Voraussetzung:

- erstmalige Registration
- Nutzer ist angemeldet

Beschreibung:

Der Nutzer hat die Möglichkeit aus der Liste der erfassten Boulder einen auszuwählen und sich zu diesem die Metadaten (Name, Farbe, Schwierigkeit, Location (Land, Ort)) in einer Detailansicht anzeigen zu lassen.

06_UC-Interaktion mit Boulder hinzufügen

Verfügbarkeit: nur online

Voraussetzung:

- erstmalige Registration
- Nutzer ist angemeldet
- Boulder ist ausgewählt

Beschreibung:

Ausgehend von der Detailansicht eines Boulders soll der Nutzer die Möglichkeit haben eine Boulder Interaktion zu erfassen. Mittels Button-Klick öffnet sich das Eingabeformular für die Boulder Interaktion, um folgende Metadaten zu erfassen:

- Status (Dropdown Auswahl: “Geschafft”,”Work in Progress”,”Idee”)
- Kommentar (Mehrzeiliges Textfeld)

07_UC-Boulder Interaktion anderer Nutzer ansehen

Verfügbarkeit: nur online

Bedingung:

- erstmalige Registration
- Nutzer ist angemeldet
- Boulder ist ausgewählt

Beschreibung:

Auf der Detailansicht des Boulders werden dem Nutzer alle Boulder-Interaktionen von allen Nutzern angezeigt, die zu diesem Boulder erfasst worden sind. Der Nutzer kann an Boulder-Interaktionen mittels Like-Button sein Interesse bekunden.

08_UC-Boulder hinzufügen

Verfügbarkeit: online & offline

Offline Voraussetzungen:

Offline Einschränkungen:

Ein Boulder wird lokal gespeichert, beim nächsten Zugriff auf das Netzwerk werden lokale Datensätze mit der Datenbank synchronisiert.
Lokal gespeicherte Daten werden bei der nächsten Netzwerkverbindung mit dem Backend synchronisiert. Hat es Änderungen gegeben wird der User darüber informiert und er hat die Möglichkeit diese zu überschreiben oder seine Änderungen zu verwerfen.

Bedingung:

- erstmalige Registration
- Nutzer ist angemeldet

Beschreibung:

Am persönlichen Homescreen der Anwendung hat der Nutzer die Möglichkeit neue Boulder hinzuzufügen, mittels Button Klick öffnet sich das Eingabeformular für einen Boulder mit den Metadaten:

- Name - Textfeld, String
- Farbe - Dropdown, String Enum

- Schwierigkeit - Dropdown, String Enum
- Bild - Upload
- Location (Land und Stadt) - Dropdown/ Location Library

Nach dem Speichern der Daten wird der Nutzer auf die Detailansicht des Boulders weitegeleitet. Während der Eingabe hat der Nutzer die Möglichkeit das Erstellen mittels “Abbrechen” Button zu beenden, dabei wird er auf den persönlichen Homescreen weitergeleitet und die Eingaben werden verworfen.

09_UC-Logout

Verfügbarkeit: online

Bedingung:

- erstmalige Registration
- Nutzer ist angemeldet
- Boulder ist ausgewählt

Beschreibung:

Am persönlichen Homescreen der Anwendung hat der Nutzer die Möglichkeit sich mittels Buttons von der App abzumelden, dabei wird die aktive Session geschlossen und der Nutzer auf den Startscreen der App weitergeleitet.

10_UC-Boulder bearbeiten (Kollision im Backend)

Verfügbarkeit: online & offline

Offline Voraussetzungen:

Benutzer war mind. einmal angemeldet und hat die Liste geladen.

Offline Einschränkungen:

Die Änderungen am Boulder werden lokal gespeichert, beim nächsten Zugriff auf das Netzwerk werden lokale Datensätze mit der Datenbank synchronisiert. Lokal gespeicherte Daten werden bei der nächsten Netzwerkverbindung mit dem Backend synchronisiert. Hat es Änderungen gegeben wird der User darüber informiert und er hat die Möglichkeit diese zu überschreiben oder seine Änderungen zu verwerfen.

Bedingung:

- erstmalige Registration
- Nutzer ist angemeldet
- Boulder ist ausgewählt

Beschreibung:

Am Detailscreen Boulder hat der Nutzer die Möglichkeit, mittels Button Klick das Bearbeiten-Formular mit folgenden Metadaten zu öffnen und Änderungen vorzunehmen:

- Name - Textfeld, String
- Farbe - Dropdown, String Enum
- Schwierigkeit - Dropdown, String Enum
- Bild - Upload
- Location (Land und Stadt) - Dropdown/ Location Library

Nach dem Betätigen durch einen Speicher-Button werden die Daten, wenn eine Netzverbindung besteht, in der Datenbank persistiert. Verfügt das Endgerät

aktuell über keine Netzverbindung werden die Änderungen lokal gespeichert und bei der nächsten Netzwerkverbindung entsprechend persistiert.

11_UC-Boulder liken (optional)

Verfügbarkeit: online

Bedingung:

- erstmalige Registration
- Nutzer ist angemeldet
- Boulder ist ausgewählt

Beschreibung:

Am Detailscreen Boulder hat der Nutzer die Möglichkeit, mittels Button-Klick das Interesse an dem Boulder zu bekunden bzw. dieses zu widerrufen.

5. Herausforderungen der mobilen App Entwicklung

5.1 Daten Zugriff / Gerätesteuerung

Der Zugriff auf die Gerätesteuerung muss vom Nutzer erlaubt werden. Ansonsten können zugehörige Funktionen nicht genutzt werden. Hierbei muss darauf geachtet werden, dass die Einschränkung die Nutzung der Applikation nicht generell einschränkt.

5.1.1 Location (Optional)

Bei der Verwendung der Google-API zur automatischen Erkennung des Standortes, muss der Nutzer den Zugriff bestätigen. Wird die Zustimmung des Nutzers erteilt, kann als Default-Wert der derzeitige Standort gesetzt werden, der natürlich abgeändert werden kann. Wird die Zustimmung nicht erhalten, bleibt der Default-Wert leer und der Nutzer muss manuell den Standort hinzufügen.

5.2 Nutzer Change am Smartphone

Unterschiedliche Nutzer können die Applikation verwenden. Der jeweilige Nutzer kann sich mit seinen eigenen Zugangsdaten anmelden und die Applikation verwenden. Möchte ein anderer Nutzer die Applikation am gleichen Smartphone verwenden muss sich der eingeloggte Nutzer zuvor abmelden.

6. Kommunikationsprotokoll

Bei dem Kommunikationsprotokoll handelt es sich um TCP/IP. Um eine gesicherte Datenübertragung zu gewährleisten wird in weiterer Folge TLS eingesetzt. Als API wird die von React Native zur Verfügung gestellte Fetch API verwendet. Diese API wird von den gängigsten Betriebssystemen unterstützt und garantiert somit eine reibungslose Kommunikation mit dem Backend. Bei dieser Fetch-API wird mittels JSON-Objekte kommuniziert.

6.1 Kollisionsmanagement

Bearbeiten zwei Nutzer gleichzeitig einen Boulder und sind nicht mit dem Netzwerk verbunden, kommt es bei einem zeitgleichen Upload zu einer Kollision.
Der Nutzer, der einen nicht bekannten Change eines anderen Nutzers überschreiben würde, wird auf diesen Umstand hingewiesen und muss mittels Schaltfläche das Überschreiben des anderen Wertes erzwingen.

6.2 API-Endpoints

Tabelle 2 API-Endpoints

Entität	Route	Methode	Funktion
User	/user/	POST	Prüft Login des Users
Boulder	/boulder/:userId	GET	Retourniert alle Boulder mit den dazugehörigen Likes des Users
	/boulder/	POST	Erstellt einen Boulder
	/boulder/:boulderId	PUT	Aktualisiert Boulder
Boulder Interaction	/boulderInteraction	POST	Erstellt neue Boulder-Interaction
	/boulderInteraction/:boulderid	GET	Retourniert alle Boulder-Interaction zu Boulder
	/boulderInteraction/:interactionId	PUT	Aktuallisiert Boulder-Interaction
Like	/like/:boulderId	POST	Erstellt Like für User für aktuellen Boulder
	/like/:boulderId	DELETE	Löscht Like für User für aktuellen Boulder

7. Sicherheitskonzept

Im Rahmen der LV ist dieser Bereich nicht umgesetzt, es wird mit Dummydaten gearbeitet.

Als Sicherheitskonzept bzw. Zugriffskontrolle wird OAuth 2.0 - “Open Authorization 2.0” verwendet. Die Autorisierung erfolgt über diese “sichere - API”, welche ein offenes Standardprotokoll ist. Weiters soll dazu auch eine 2-Faktor-Authentifizierung auf Basis von React Native implementiert werden. Auch das Google Captcha wird als zusätzlicher Sicherheitsaspekt integriert. Als optionale bzw. zusätzliche Authentifizierung soll eine biometrische Authentifizierung über den Fingerabdrucksensor implementiert werden, um eine optimale Sicherheit zu gewährleisten.

7.1 Login Handling

Beim Login wird ein Bearer-Secure-Token übertragen. Für die Dauer dessen Gültigkeit wird die Kommunikation zwischen Endgerät und Backend mit diesem identifiziert. Nach Ablauf der Gültigkeit ist eine erneute Anmeldung nötig. Am Endgerät werden außer dem verschlüsselten Token keine Anmeldeinformationen gespeichert.

7.2 E-Mail-Verschlüsselung

Die E-Mail-Adressen der User werden in der Datenbank mittels RSA verschlüsselt.

7.3 Localstorage

Die Verschlüsselung der Daten, welche am Endgerät gespeichert werden, wird zukünftig im Rahmen einer Weiterentwicklung mit Realm umgesetzt. Dies ermöglicht das verschlüsselte Speichern von Daten am Endgerät.

7.4 Kollisions-Management

Nutzer haben die Möglichkeit lokal gespeicherte Boulder auch ohne Netzwerkverbindung zu bearbeiten. Beim Laden des Home-Screens wird geprüft ob Daten zur Synchronisation lokal gespeichert sind und eine aktive Netzwerkverbindung besteht.
Im Backend wird überprüft, ob das LastChangeDate (wird ausschließlich vom Server bei der Bearbeitung gesetzt und beim Boulder gespeichert) mit dem des zu Bearbeitenden Boulders in der Datenbank übereinstimmt. Ist dies der Fall werden die Daten direkt persistiert und das Backend sendet einen 200 Statuscode. Stimmen die LastChangeDate des Boulders aus der Datenbank und dem vom Frontend übermittelten Boulder nicht überein (passiert wenn ein andere Nutzer den Boulder aktualisiert hat, während der Nutzer seine Daten bearbeitet hat), sendet das Backend den Statuscode 409. Das Frontend verarbeitet dies und zeigt den Kollisions-Management Screen an, welcher dem Nutzer die Information gibt, dass eine Kollision entstanden ist. Der Nutzer hat nun die Möglichkeit, die bestehenden Änderungen zu überschreiben oder seine eigenen Anpassungen zu verwerfen. Das Überschreiben erfolgt via „force“ Parameter, welcher dem EditBoulder Objekt mitgesendet wird.

8. Deployment

8.1 Frontend

Die Client Applikation wird über den Google Play Store, ebenso wie über den Apple Store bereitgestellt. Neue Releases werden mit dem Push-Prinzip auf den Endgeräten deployed, da durch die indirekte Nutzerkommunikation so Fehler vermieden werden können. Aufgrund der Ausrichtung wird hierbei der Fokus auf dem Google Play Store liegen und die Optimierungen für Android vorgenommen.

8.2 Backend

Für das Hosting des Backends sowie der Datenbank werden die jeweiligen Services der Amazon Web Services gemietet. Die Kosten dieser Services skalieren mit der tatsächlich verwendeten Leistung (Anzahl an Anfragen, Speichermenge, etc.). Für den Einstieg kann hier mit ungefähr 350€ im Monat gerechnet werden. Diese Kosten steigen annähernd linear mit der Nutzeranzahl.

9. Umsetzung

9.1 Frontend Struktur

Durch die Umsetzung der Applikation mit React-Native ist die Struktur generell vorgeben. Um eine Trennung zu schaffen wurden der Code aufgeteilt in Componenten, API- Klasse, Services und Styles.

9.2 Typisierung

9.2.1 User

```
export interface IUser {
    userId: number,
    userEmail: string,
    userName?: string,
}
```

Abbildung 6 Typisierung User

9.2.2 News

```
export interface INews {
    title:string;
    image:string;
    description:string;
    link:string;
}
```

Abbildung 7 Typisierung News

9.2.3 Lookup Werte

```
export interface IBasic {
    id:number,
    name: string,
    key:number
}
Denis Schüle, a month ago | 1 author (Denis Schüle)
export interface IColor extends IBasic {
    value: string
}

Denis Schüle, a month ago | 1 author (Denis Schüle)
export interface IDifficulty extends IBasic {
}

Denis Schüle, a day ago | 1 author (Denis Schüle)
export interface ILocation {
    id:number,
    country:string,
    region:string
    key:number
}
```

Abbildung 8 Typisierung Basis - Location

```typescript
export interface ILocation {
    id:number,
    country:string,
    region:string,
}

export interface ILocationFilterValues {
    countries:string[],
    region:string[]
}
```

Abbildung 9 Typisierung Location

```typescript
export interface IStatus extends IBasic {
    id: number,
    name: string,
    icon: string
}
```

Abbildung 10 Typisierung Status

9.2.4 Boulderinteraktion

```typescript
export interface IBoulderInteraction {
  boulder_id:number,
  user_id:number,
  title: string,
  status:number,
  comment:string,
  created: Date,
  id?:string
}

Denis Schüle, 3 days ago | 1 author (Denis Schüle)
export interface INewBoulderInteraction{
  boulderId:number,
  userId:number,
  title:string,
  comment:string,
  status:number

}
Denis Schüle, 3 days ago | 1 author (Denis Schüle)
export interface IUpdateBoulderInteraction extends INewBoulderInteraction{
  interactionId:number
}
export type BoulderInteractionFormData = {
  id:string;
  title: string;
  comment: string;
  status:number;
  boulder_id:string;
  user_id:string;
};
Denis Schüle, 3 days ago | 1 author (Denis Schüle)
export class BoulderInteraction implements IBoulderInteraction{
  boulder_id: number;
  user_id: number;
  userName:string;
  title: string;
  status: number;
  comment: string;
  created: Date;
  id: string;
  constructor(boulder_id:number, user_id:number,userName:string, title:string ='',status:number=1,comment:string='',created:Date =new Date(),id:string ='')
    this.boulder_id = boulder_id;
    this.user_id    = user_id;
    this.userName   = userName;
    this.title      = title;
    this.status     = status;
    this.comment    = comment;
    this.created    = created;
    this.id         = id;
  }
  isEqual(action:IBoulderInteraction):boolean {
    return this.user_id === action.user_id && this.boulder_id === action.boulder_id && this.created === action.created
  }


}      Denis Schüle, a month ago via PR #31 • basic form
```

Abbildung 11 Typisierung Boulderinteraktion

9.2.5 Boulder

```
export interface IBoulder{
        id:number,
        title:string,
        color: number,
        difficulty: number,
        location_id:number,
        like:boolean,
        lastChangeTimestamp:Date;
        lastEditor:string;
}
export type BoulderFormData = {
        title: string;
        color: number,
        difficulty: number,
        location_id:number,
        boulder_id:number;
        like:boolean;
        id:number;
        lastChangeTimestamp:Date;
        lastEditor:string;

}
Denis Schüle, 3 days ago | 1 author (Denis Schüle)
export interface INewBoulder {
        creatorId:number,
        name:string,
        colour:number,
        difficulty: number,
        locationId: number
}
Denis Schüle, a day ago | 1 author (Denis Schüle)
export interface IEditBoulder {
        userId:number,
        name:string,
        colour:number,
        difficulty: number,
        locationId: number,
        force:boolean,
        boulderId:number,       Denis Schüle, a day ago via PR #44 • async handling
        lastChangeTimestamp?:Date ,
        lastEditor?:string,
}
Denis Schüle, a day ago | 2 authors (Denis Schüle and others)
export class Boulder implements IBoulder{
        id:number;
        title:string;
        color: number;
        difficulty: number;
        location_id:number;
        like:boolean;
        lastChangeTimestamp:Date;
        lastEditor:string;
        constructor(
                id:number,
                title:string,
                color: number,
                difficulty: number,
                location_id:number,
                like:boolean,
                lastChangeTimestamp:Date,
                lastEditor:string,
        ){
                this.id=id,
                this.title      =title;
                this.color      = color;
                this.difficulty = difficulty;
                this.location_id=location_id;
                this.like       =like;
                this.lastEditor = lastEditor
                this.lastChangeTimestamp= lastChangeTimestamp;
        }

}
```

Abbildung 12 Typisierung Boulder

9.3 Verwendete Libaries

9.3.1 Frontend

```
"@react-native-async-storage/async-storage": "^1.15.2",
"@react-native-community/checkbox": "^0.5.7",
"@react-native-community/cli": "^5.0.1-alpha.2",
"@react-native-community/netinfo": "^6.0.0",
"@react-native-community/toolbar-android": "0.1.0-rc.2"
"@react-native-masked-view/masked-view": "^0.2.3",
"@react-native-picker/picker": "^1.15.0",
"@react-navigation/native": "^6.0.0-next.2",
"@react-navigation/stack": "^6.0.0-next.9",
"moment": "^2.29.1",
"portable-fetch": "^3.0.0",
"react": "17.0.1",
"react-hook-form": "^7.2.1",
"react-native": "0.64.1",
"react-native-btr": "^2.1.2",
"react-native-dropdown-picker": "^5.1.0",
"react-native-eject": "^0.1.2",
"react-native-elements": "^3.3.2",
"react-native-gesture-handler": "^1.10.3",
"react-native-image-picker": "^3.3.2",
"react-native-reanimated": "^2.1.0",
"react-native-safe-area-context": "^3.2.0",
"react-native-screens": "^3.1.0",
"react-native-vector-icons": "^8.1.0",
"url": "^0.11.0"
"@babel/core": "^7.12.9",
"@babel/runtime": "^7.12.5",
"@react-native-community/eslint-config": "^2.0.0",
"@types/jest": "^26.0.20",
"@types/react-native": "^0.64.0",
"@types/react-test-renderer": "^16.9.2",
"babel-jest": "^26.6.3",
"eslint": "^7.14.0",
"jest": "^26.6.3",
"metro-react-native-babel-preset": "^0.64.0",
"react-test-renderer": "17.0.1",
"typescript": "^3.8.3"
```

Abbildung 13 Frontend Libraries

https://reactnavigation.org/docs/

9.3.2 Backend

```
"body-parser": "^1.19.0",
"cookie-parser": "^1.4.5",
"cors": "^2.8.5",
"dotenv": "^9.0.2",
"express": "^4.17.1",
"express-validator": "^6.11.1",
"morgan": "^1.10.0",
"mysql2": "^2.2.5"
```

Abbildung 14 Backend Libraries

9.4 App Screenshots

9.4.1 Pre-Log-In

Abbildung 15 Pre-Login

9.4.2 Login

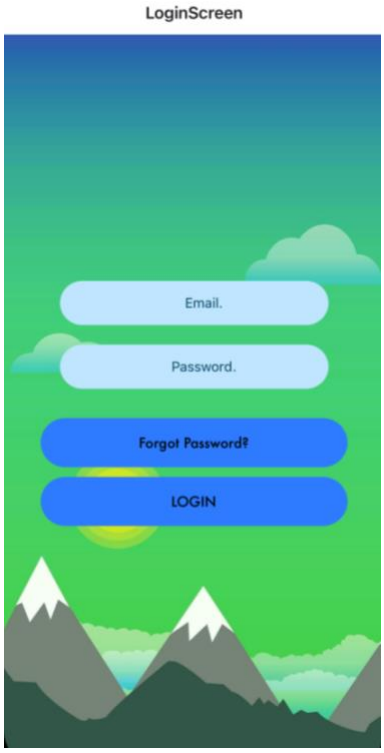

Abbildung 16 Login

9.4.3 Home

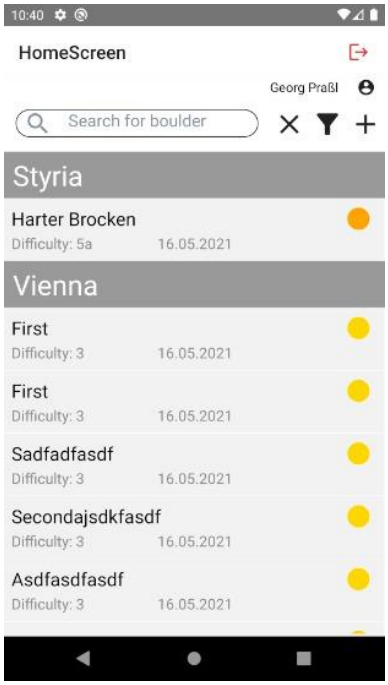

Abbildung 17 Home

9.4.4 Home – Region Filter

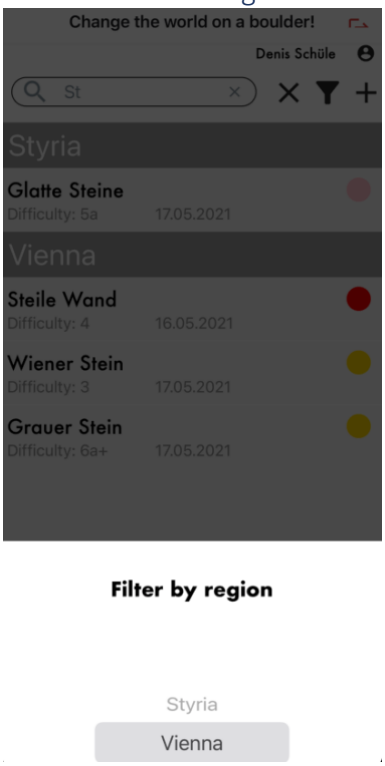

Abbildung 18 Home Filter Region

9.4.5 Detail-Boulder

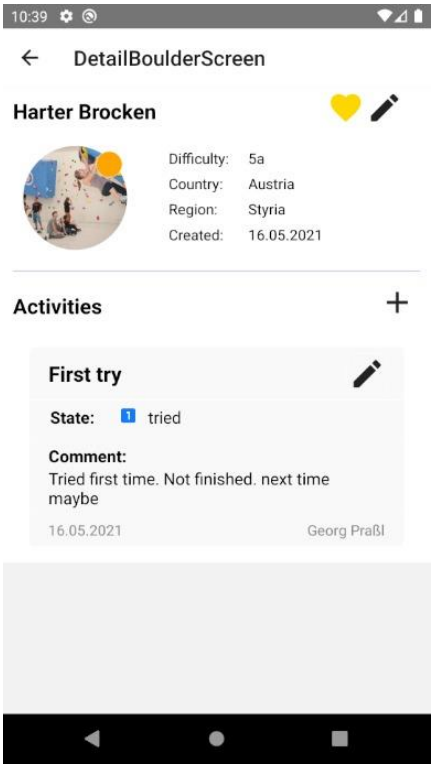

Abbildung 19 Detail-Boulder

9.4.6 Boulder Formular

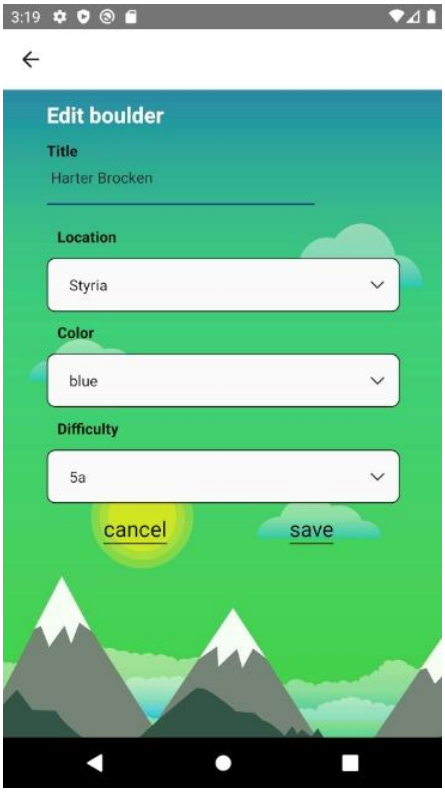

Abbildung 20 Boulder Formular

9.4.7 Activity Formular

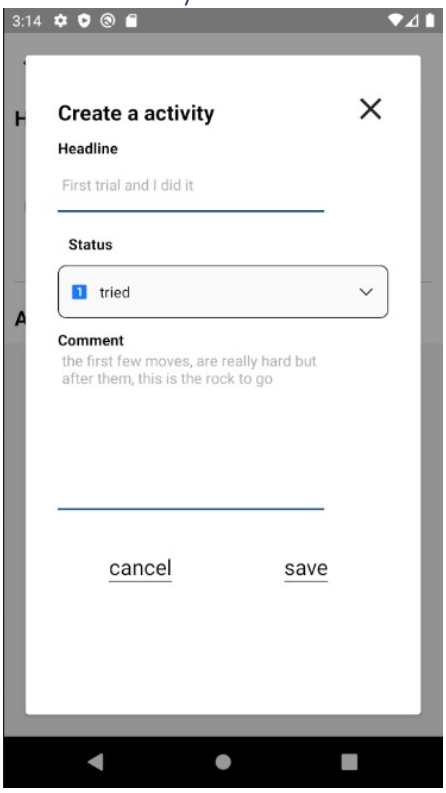

Abbildung 21 Activity Formular

9.4.8 Kollissions-Management

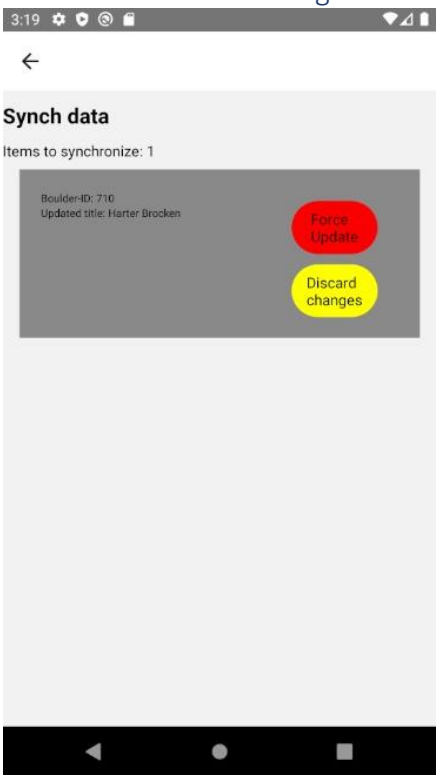

Abbildung 22 Kollisions-Management

10. Fazit

Die Entwicklung der Applikation via React-Native also als Cross-Plattform Applikation ist im gesamten mit einer funktionsfähigen App abgeschlossen worden. Im Zuge der Entwicklung gab es generelle Schwierigkeiten in Bezug auf die Entwicklungsumgebung in Bezug auf die unterschiedlichen Betriebssysteme der Rechner der Teammitglieder. Das Kollisions-Management war im Gegensatz zu den anfänglichen Annahmen keine Schwierigkeit und konnte generell problemlos umgesetzt werden.

Allgemein ist zu vermerken das die Umsetzung mit React-Native es ermöglicht die Applikation auf IOS und Android zu nutzen, hierzu ist es jedoch nötig plattformspezifische Anpassungen vorzunehmen um die Anwendung auf beiden Plattformen entsprechend nutzen. Aufgrund der Zielsetzung die Applikation auf Android auszurichten, wurde die Optimierung für IOS vernachlässigt.

11. Definitionen Akronyme Abkürzungen

API – Eine Programmierschnittstelle häufig nur kurz API genannt (von englisch application programming interface, wörtlich ‚Anwendungsprogrammierschnittstelle') ist ein Programmteil, der von einem Softwaresystem anderer Programme zur Anbindung an das System zur Verfügung gestellt wird.
Ausreichende Internetverbindung – Internet Verbindung die Bildübertragung in angemessener Zeit und Qualität erlaubt, diese Verbindung besteht während der gesamten Interaktion.
Button – Schaltfläche in der Oberfläche der Anwendung, durch die eine Aktion der Anwendung ausgelöst wird.
DB – Datenbank. Ein System zur elektronischen Datenverwaltung
Domänenwissen - Das ist das Wissen über betriebswirtschaftliche Prozesse, branchenspezifische Kenntnisse, sowie eine Sensibilität für mögliche Risiken
KI – Künstliche Intelligenz
MB – Die Bezeichnung Megabyte (MB) wird im Allgemeinen für die Bestimmung der Speicherkapazität von Speichern, primär für Speicherbausteine benutzt. Beim Mega handelt es sich um ein Präfix, der auf dem Dezimalsystem basiert: 10exp6, bzw. 1 Million.
Rich-Text-Field – Ein Eingabefeld welches das proprietäres Datenformat „Rich Text Format" RTF als Text entgegennimmt. Im Gegensatz zu „Plain text" (Einfachen Text), der nur die reinen Textzeichen transportiert, aber keinerlei Formatierungen wie zum Beispiel Schriftgrößen, -arten oder -auszeichnungen, enthält ein RTF-Dokument auch zahlreiche Textformatierungsmerkmale bis hin zu eingebetteten Grafiken, ohne andererseits an eine bestimmte Software gebunden zu sein. Praktisch alle Textverarbeitungssysteme können RTF-Dateien schreiben und lesen. Layouttreue ist dabei allerdings nicht gewährleistet, es kann beispielsweise zu veränderten Seitenumbrüchen auf dem Zielsystem kommen.
RSA-
Selektiert – Das entsprechende Element wird ausgewählt.
JSON – JavaScript Object Notation. Ist ein kompaktes Datenformat in einer einfach lesbaren Textform und dient dem Zweck des Datenaustausches zwischen Anwendungen
RSA - Ist ein asymmetrisches kryptographisches Verfahren, das sowohl zum Verschlüsseln als auch zum digitalen Signieren verwendet werden kann. Es verwendet ein Schlüsselpaar, bestehend aus einem privaten Schlüssel, der zum Entschlüsseln oder Signieren von Daten

verwendet wird, und einem öffentlichen Schlüssel, mit dem man verschlüsselt oder Signaturen prüft. Der private Schlüssel wird geheim gehalten und kann nicht mit realistischem Aufwand aus dem öffentlichen Schlüssel berechnet werden.

12. Abbildungsverzeichnis

13. Tabellenverzeichnis