

Projektplanung

Aufbau der GUI (Skizzen)

Welche Fenster wird es geben?

- Registrieren Page
- Login Page
- Home Page
- Berg suchen Page
- Wetter Page
- Profil Page
- Berge gemacht Page

Wie sehen diese grob aus?

Registrieren



Vorname



Nachname

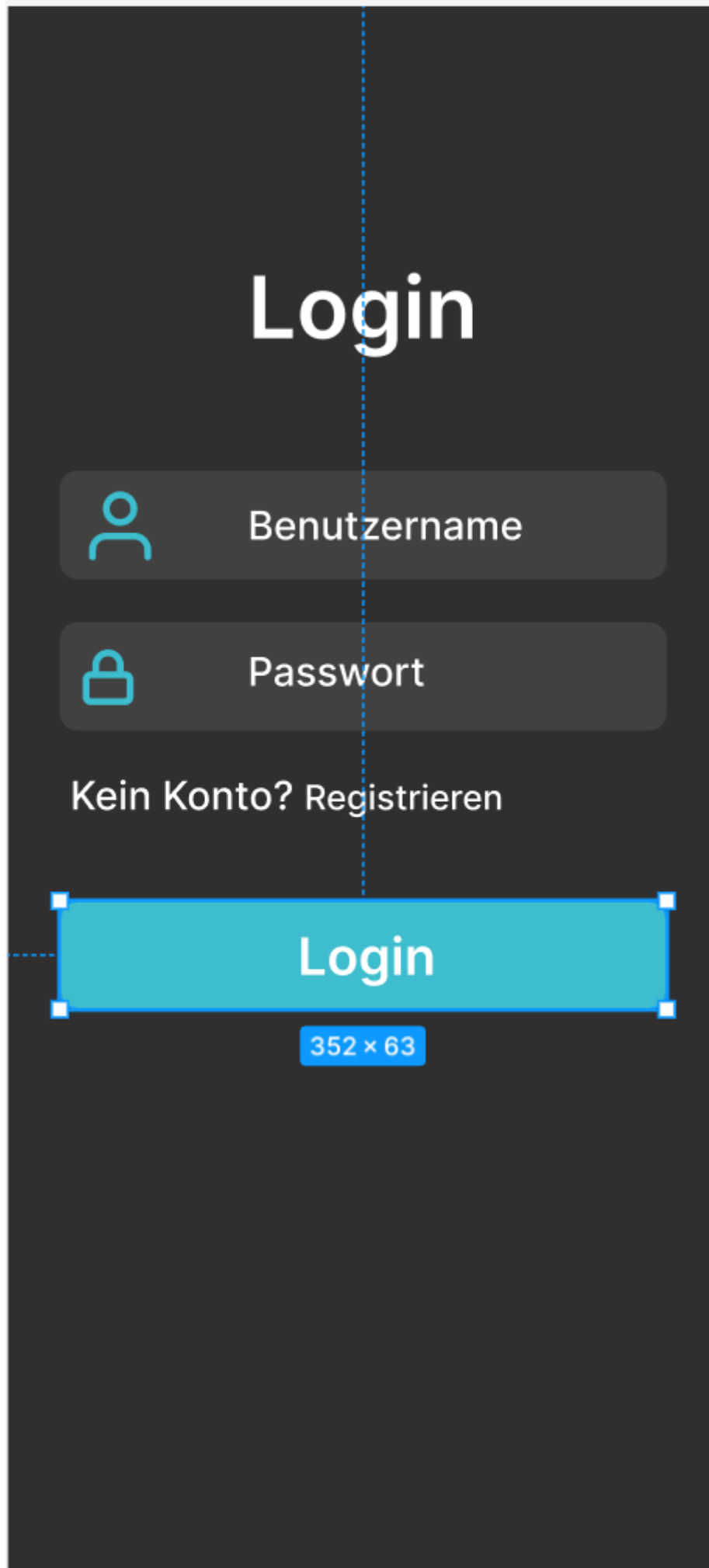


Benutzername

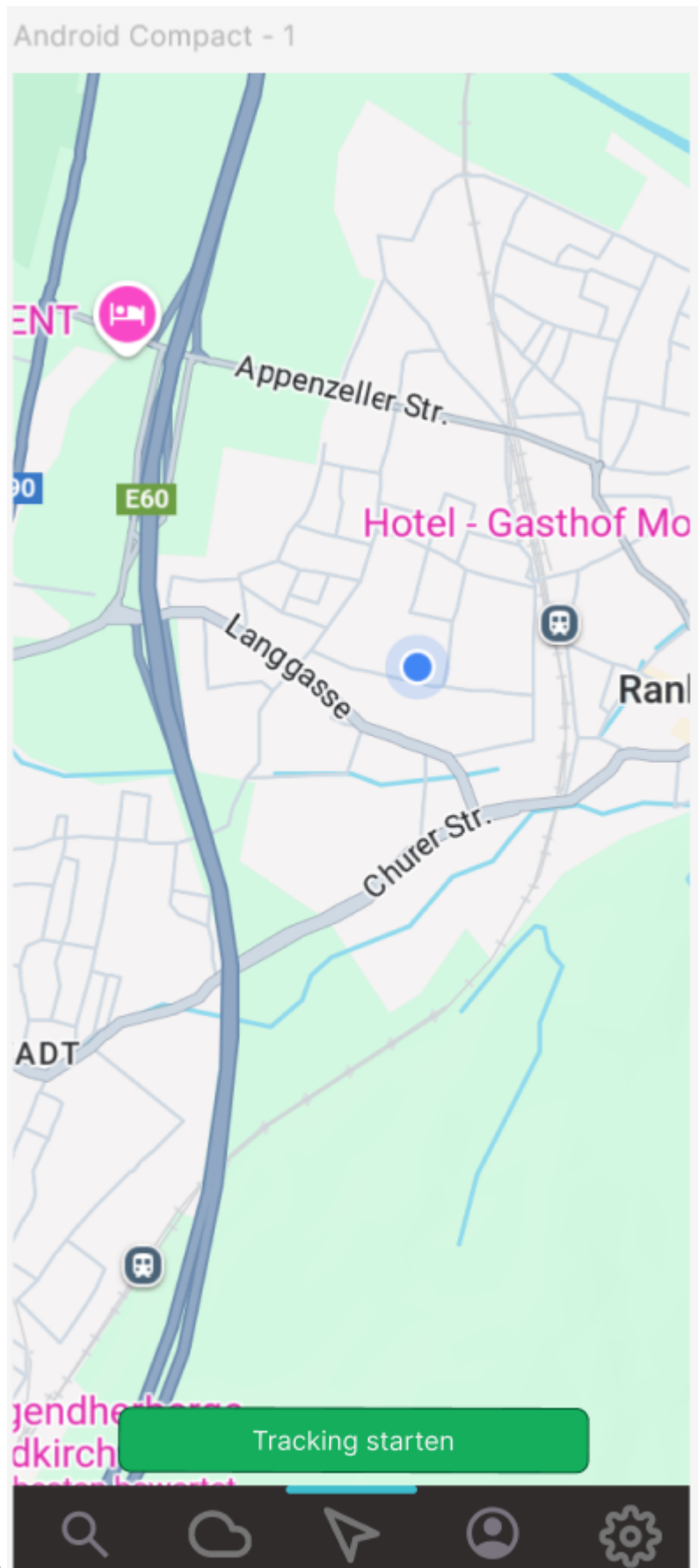


Passwort

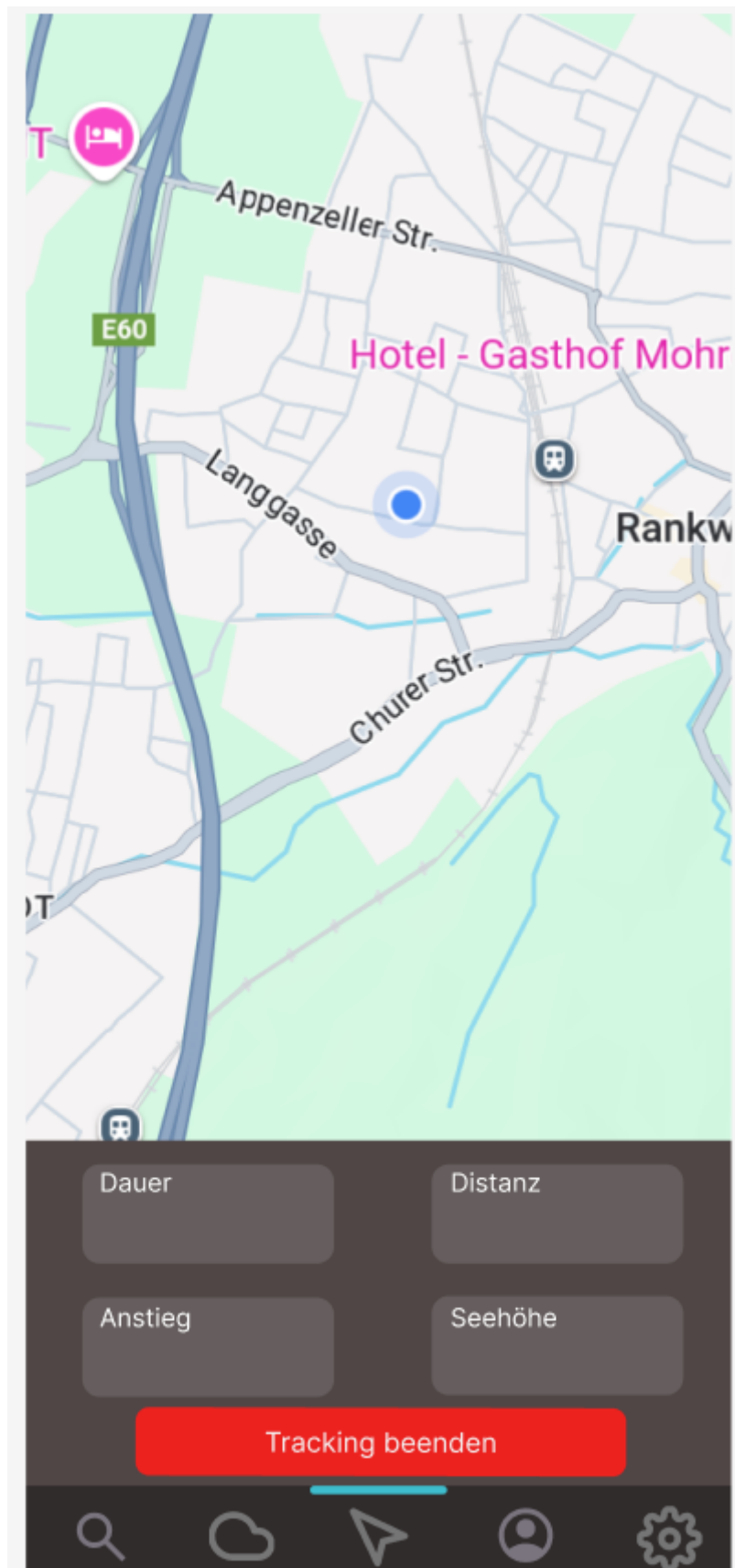
Registrieren




- Login Page



- Home Page



- Tracking gestartet Page








Matterhorn

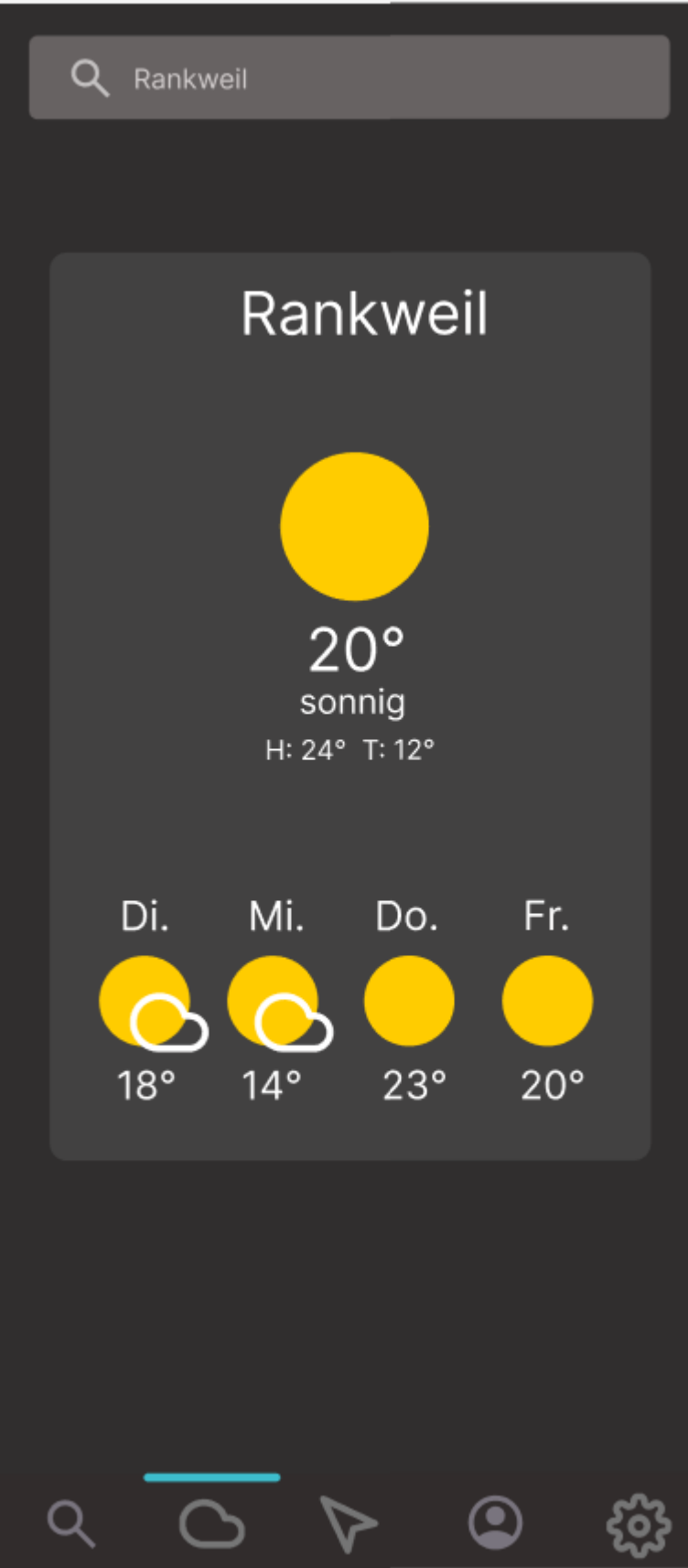
Höhe

Ort

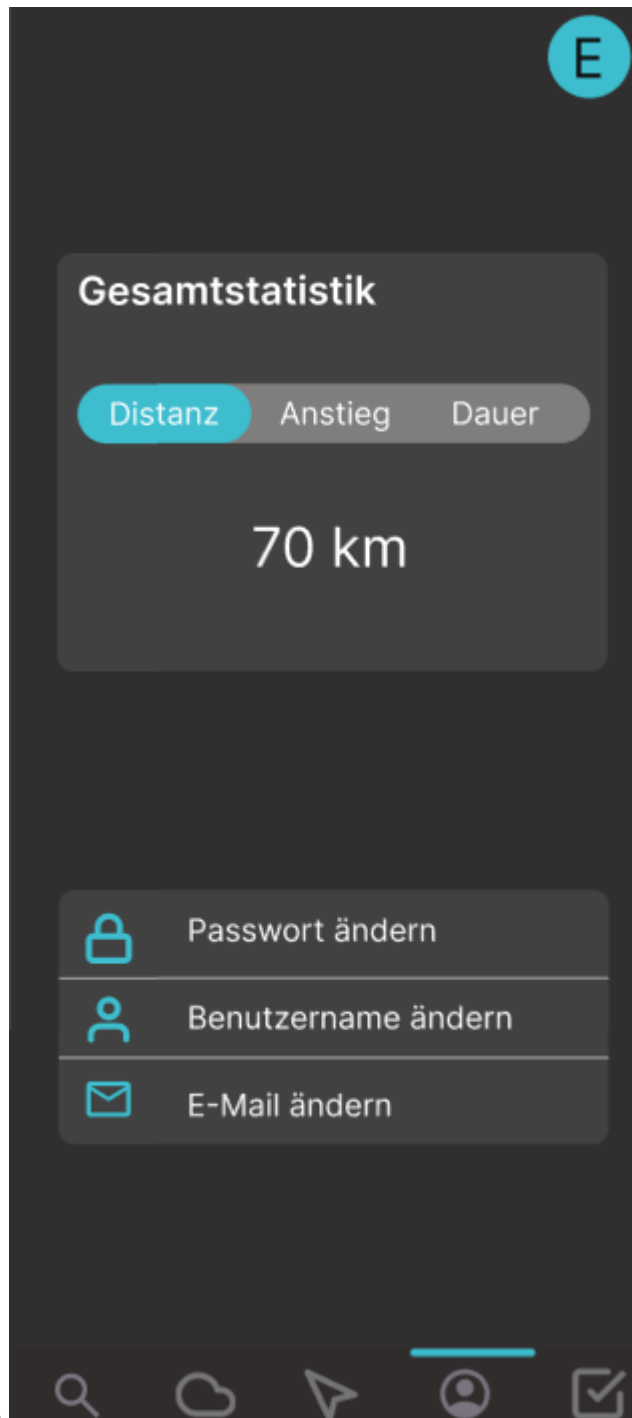
Beschreibung



- Berg suchen Page



- Wetter Page



- Profil Page
- Berge gemacht Page Diese Skizze haben wir noch nicht erstellt, braucht aber auch keine richtige Skizze, da es nur eine Liste ist

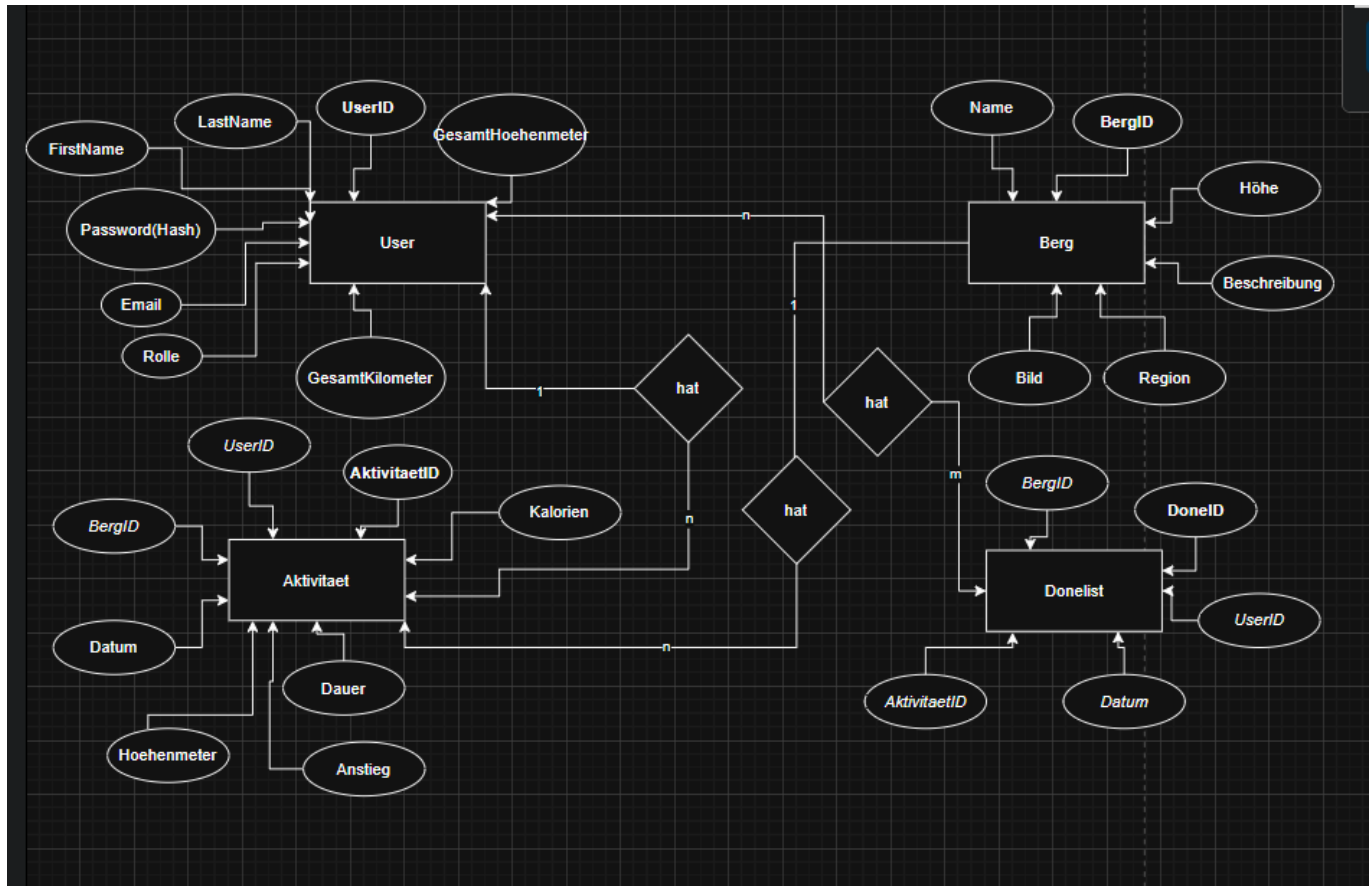
Wie ist die Benutzernavigation geplant?

- Von der Login Page gelang man zu der Home Page (Und die Page wo direkt angezeigt wird ist entweder die Berge suchen Page oder die Navigation Page)
- Bei der Page angekommen kann man unten navigieren zwischen den verschiedenen Seiten (Home, Berge suchen, Wetter, Profil, Berge gemacht)
- Auf der Navigation Page kann man auf tracking starten und dann kommt ein grauer Bereich mit den Informationen (siehe Skizze)
- Bei WetterPage kann man die Stadt auswählen und dann wird das Wetter angezeigt
- Bei der Search Mountain Page kann man einen Berg eingeben und dann wird ein Bild von diesem angezeigt die Höhe und der Ort des Berges auch gibt es ein Button um den Berg zu der DoneList

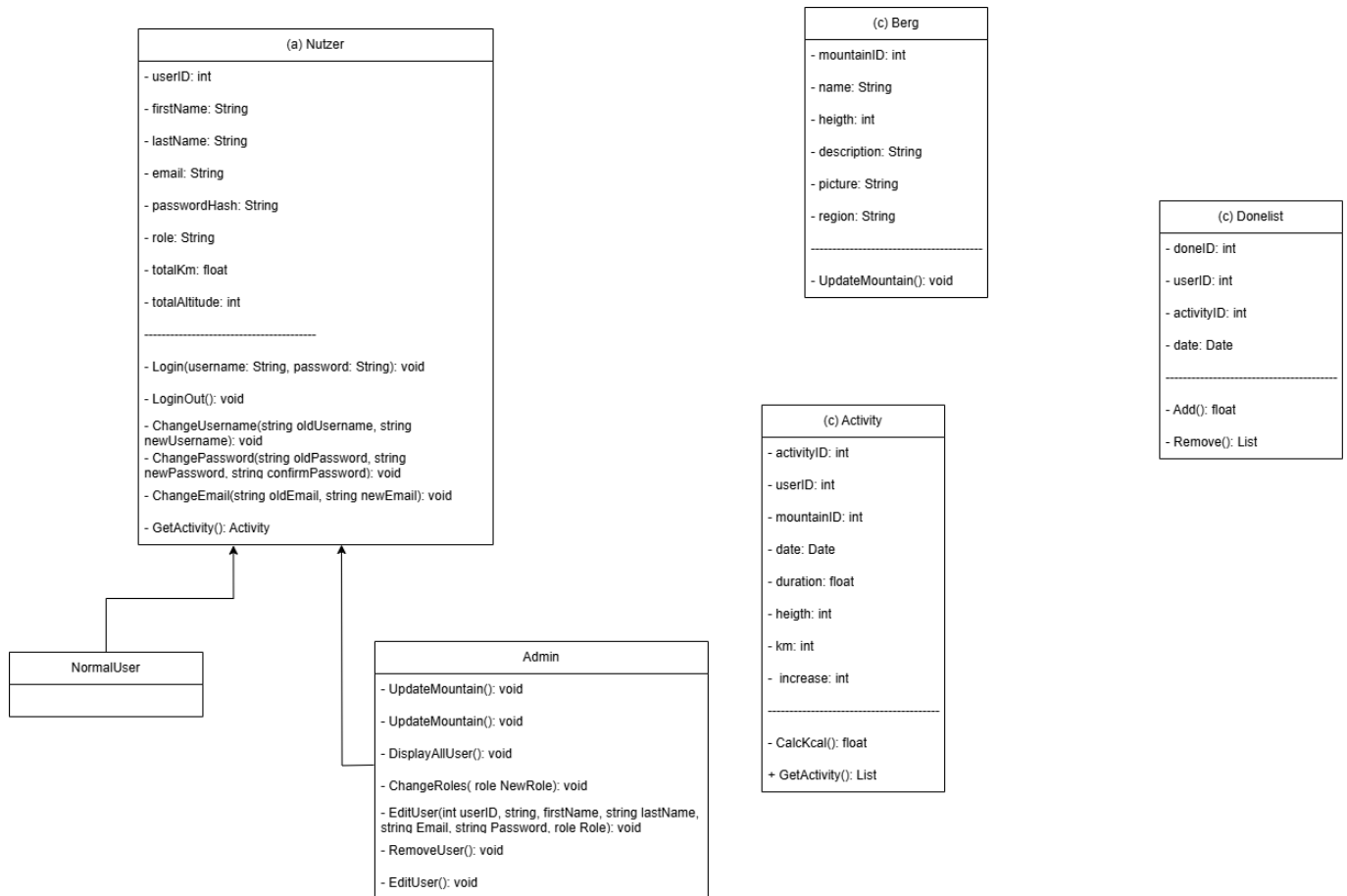
hinzuzufügen

- Bei der Profile Page sieht man seine Aktivitäten und kann Sachen machen wie den Namen ändern, das Passwort ändern
- Bei der Done Page sieht man einfach nur welche Berge man gemacht hat und kann diese auch löschen

Aufbau des Programms/der Datenbank



hier ist das ER-Diagramm zu sehen.



hier ist das Klassendiagramm zu sehen.

wie arbeiten sie zusammen?

Mit Supabase oder postgresSQL wird die Datenbank online erstellt und mit Swagger wird auf die Datenbank zugegriffen, am Anfang werden wir direkt von flutter auf die Datenbank zugreifen da es schneller geht und währenddessen werden die Endpunkte nach und nach erstellt.

- Mit einem öffentlichen Server der als Rest-API fungiert, wird die Kommunikation zwischen der App und der Datenbank ermöglicht.
- Die App sendet Anfragen an den Server, auf dem Server läuft die ganze Zeit die Swagger Python Datei, die die Anfragen entgegennimmt und die Datenbank abfragt.
- Der Server hat eine öffentliche IP-Adresse, die in der App hinterlegt ist, um so die Abfragen zu ermöglichen.

Wie werden die Mindestanforderungen umgesetzt?

- **Git**: Mit der aktiven Verwendung von GIT wird diese Mindestanforderung gedeckt. Auch wird hier die Planung durchgeführt mit den Tickets.
- **Klassendiagramme**: Werden davor geplant (z.B. Benutzer, Berg, Donelist).
- **Grafische Anwendung**: 3 Fenster werden wir auf jeden Fall haben (Login, Berge suchen, Wetterbericht, Live Tracking).
- **Vererbung & abstrakte Klassen**: Werden für gemeinsame Funktionalitäten verwendet.
- **Interfaces**: Werden für Services implementiert.
- **API Dokumentation**: Wird laufend von uns durchgeführt.
- **Unit Tests**: Tests werden nach Abschlüssen von Klassen/Methoden programmiert oder davor. Der Fokus liegt auf kritischen Methoden.

- **Logging:** Wird während dem Programmieren implementiert.
- **Bonus:** Hintergrund-Thread für Routing-Tracking.

Datenbank:

- **Tabellen:** Mind. 3 Tabellen (Berg, User, DoneList, Aktivitaet).
- **Select Joins:** Werden für Datenabfragen wie das Laden der DoneList verwendet.
- **2 Rollen (admin, user):**
 - Admin hat Lese-/Schreibzugriff auf alles.
 - User hat nur Leserechte und begrenzte Schreibrechte.
- **SQL Datenbank:** PostgreSQL (geplant/vorerst).
- **REST Interface/Swagger:** Wird für GET/POST-Operationen implementiert.
- **Unit Tests:** Werden nach Abschluss geschrieben, um die Funktionalität zu überprüfen.
- **Logging:** Wird während der Entwicklung implementiert.

Welche Features sind ein Muss?

- **Live Tracking mit Karte** (Dauer, Höhenmeter, Distanz, Anstieg, kcal verbrannt).
- **Liste mit gemachten Bergen** (abhaken können).
- **UserLogin.**
- **Gesamtstatistik vom Account** (wie viel km gesamt gelaufen, Höhenmeter).
- **Aktivitäten speichern** (Wann war es, Dauer, Distanz, Anstieg).
- **Berge suchen können** (Daten wie Höhe des Berges).
- **Wetterbericht von Städten.**

Welche Features sind Erweiterungen (nice-to-have), wenn genügend Zeit bleibt?

- **Lebensbestätigung** (alle zwei Stunden bestätigen, dass man lebt - Sound, Vibration).
- **Wetteranalyse** (Regen beginnt in 2 Stunden).
- **SOS-Knopf** (Notfall).
- **Achievements.**

Wie möchten wir das Ganze grob umsetzen?

- **Framework:** Flutter
- **APIs für Funktionen:**
 - **Karten & GPS-Tracking:** OpenStreetMap API.
 - **Wetterdaten:** OpenWeather API.
- **Datenbank:** PostgreSQL.
- **Zusammenarbeit:** Über GitHub.