Projektidee:

Eine App, mit der du ganz einfach checken kannst, was da eigentlich grade über dir lang fliegt. Und nebenbei für den Sammelspaß, kann man noch die gesichteten Flugzeuge sammeln und mit seinem Flugzeug begeisterten Freunden vergleichen.

Kernfunktion:

Die Möglichkeit Flugzeuge in dem eigenen Sichtfeld genauer unter die Lupe nehmen zu können

Randinformationen:

- Programmiersprache: Kotlin
- OS: Android
- Gruppen Größe: 1

Komponenten:

Aktivities:

- Main Screen
- Map
- Aircraft details
- Collection
- Profile

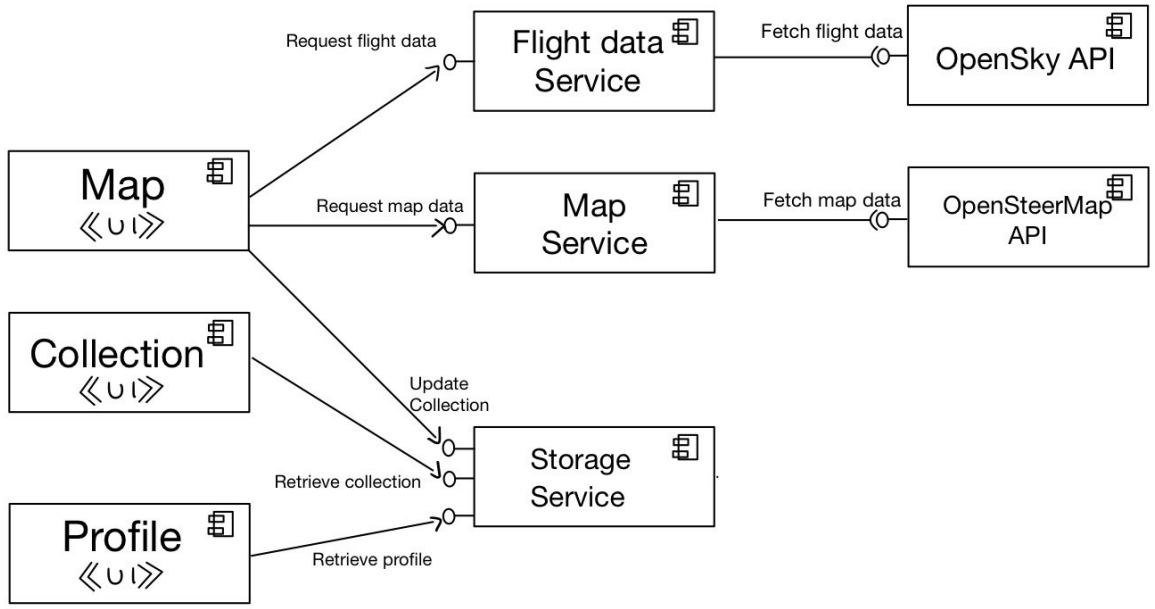
Services:

- API-Aufruf für die Flugdaten
- API-Aufruf für Kartenmaterial
- Speichern des gefundenen Flugzeugs

Content Providers:

- SQLite database
- OpenStreetMap REST API
- Opensky REST API

Komponentendiagramm:



Komponenten-Beschreibung

Activities:

1. Main Screen
    - Funktion: Startet die App und bietet dem Nutzer Zugang zu den Hauptbereichen (Karte, Sammlung, Profil).
    - API: Keine externe API, interne Navigation zu anderen Activities.
    - Test: UI-Tests, um sicherzustellen, dass alle Buttons korrekt zu den Activities navigieren.
2. Map
    - Funktion: Zeigt die Karte mit der Position des Nutzers und der darüberfliegenden Flugzeuge.
    - API: Ruft die OpenStreetMap API und die OpenSky API ab, um Flug- und Standortdaten zu aktualisieren.
    - Test: UI-Tests für die Kartendarstellung und Unit-Tests für die korrekte Darstellung der Flugzeuge und der eigenen Position.
3. Aircraft Details
    - Funktion: Zeigt die Details des ausgewählten Flugzeugs an.
    - API: Holt Flugzeugdetails aus dem Speicher-Service.
    - Test: Unit-Tests für die Anzeige korrekter Daten, UI-Tests für korrekte Layoutdarstellung.

4. Collection
    - Funktion: Zeigt alle bisher gesichteten Flugzeuge an.
    - API: Ruft Daten aus der SQLite-Datenbank ab.
    - Test: Datenintegritäts-Tests, um sicherzustellen, dass alle Daten korrekt angezeigt werden. Unit Test zur Überprüfung ob einzelne Methoden wie ranking korrekt funktionieren.
5. Profile
    - Funktion: Zeigt Benutzerinformationen und App-Einstellungen.
    - API: Intern, keine externe API.
    - Test: UI-Tests für die Einstellungen und Benutzerprofilanzeige.

Services:

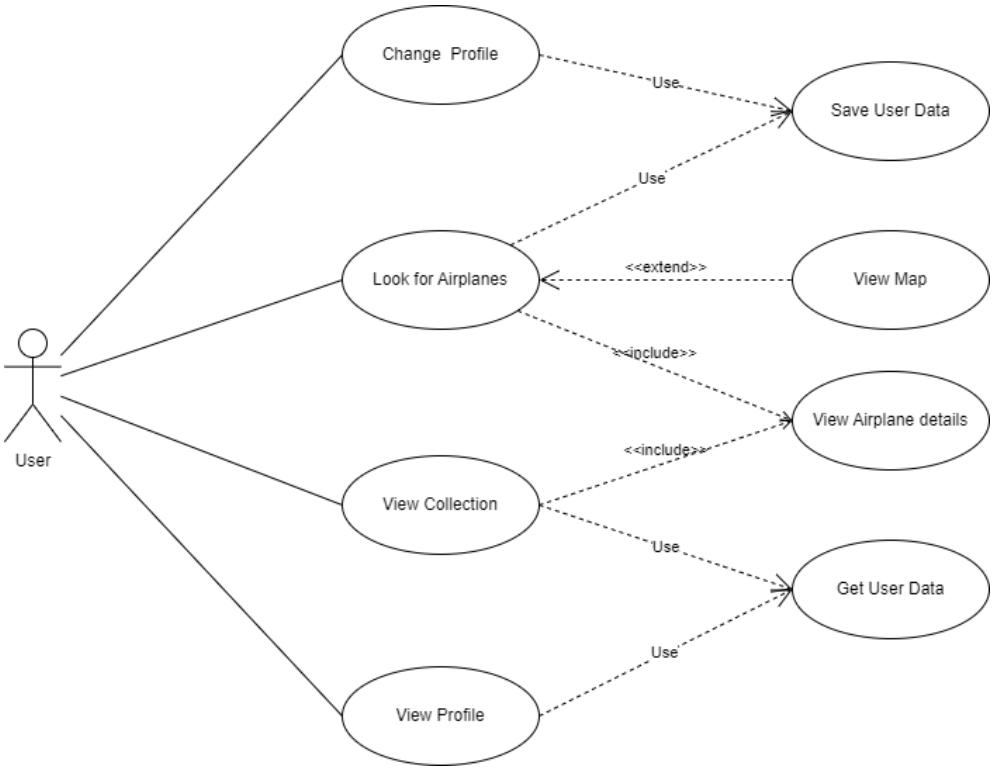
1. API-Service für Flugdaten
    - Funktion: Kommuniziert mit der OpenSky API, um Daten über Flugzeuge in der Umgebung zu erhalten.
    - API: Stellt eine Schnittstelle für die Kommunikation mit der OpenSky API bereit.
    - Test: Integrationstests, um die erfolgreiche API-Kommunikation und die korrekte Datenverarbeitung zu prüfen. Tests mit Mock-APIs zur Prüfung der Netzwerkstabilität.
2. API-Service für Kartenmaterial
    - Funktion: Ruft das Kartenmaterial von OpenStreetMap ab.
    - API: Bindet die OpenStreetMap API ein, um Karten zu aktualisieren und darzustellen.
    - Test: Integrationstests zur Prüfung der API-Abfragen und ordnungsgemäßen Darstellung der Karte.
3. Speicher-Service
    - Funktion: Speichert und lädt gesichtete Flugzeuge in der SQLite-Datenbank.
    - API: Stellt eine Schnittstelle zum Speichern und Abrufen von Flugzeugdaten bereit.
    - Test: Unit-Tests zur Überprüfung der Datenkonsistenz und Speicherung in der Datenbank.

Content Providers:

1. SQLite Datenbank
    - Funktion: Speichert gesichtete Flugzeuge lokal.
    - API: Schnittstelle zum Speicher-Service, um Daten zu speichern und abzurufen.
    - Test: Es wird davon ausgegangen, dass die einfachen Selects und Inserts funktionieren
2. OpenStreetMap REST API
    - Funktion: Bereitstellung des Kartenmaterials für die Karte.
    - API: REST-API-Schnittstelle für Kartenmaterial.
    - Test: Verbindungstests und Mock-API-Tests zur Sicherstellung der API-Stabilität.
3. Opensky REST API
    - Funktion: Liefert Flugdaten für Flugzeuge in der Nähe.
    - API: REST-API-Schnittstelle zur Abfrage von Flugdaten.
    - Test: Verbindungstests und Mock-API-Tests zur Sicherstellung der API-Stabilität.

Testplan (Die Tests der einzelnen Komponenten sind in der Komponenten-Beschreibung aufgeführt)

Use-Case-Diagramm:



Tests

Alle Services sind mit einem Interface ausgestattet, gegen welches sie getestet werden.

Change Profile / View Profile:

Integrationstests zur Überprüfung ob nach Änderung der Profildaten die daten gespeichert und wieder korrekt aufgerufen wird. Dabei wird das Zusammenspiel der Profil- und der Datenbankkomponente getestet.

Look for Airplanes:

Eine Mischung aus Integrationstests und Mocking Tests, um zu überprüfen, ob die Daten durch den Map Service und den Flight Data Service korrekt an die Map Komponente weitergegeben werden und dann von dieser korrekt verarbeitet werden. In diesem Fall sind die API aufrufe gemocked, da es nicht viel Sinn ergibt API aufrufe im Test zu machen.

View Collection:

Integrationstests, um festzustellen, ob die in der Map gesichteten Flugzeuge tatsächlich in der Collection auftauchen. Dadurch wird das Zusammenspiel der Collection, Datenbank und Map Komponente getestet.