**Prototyp: Detailliertere Informationen**

Auf den folgenden Seiten sind detaillierte Informationen wie zum Beispiel die verwendete Software-Architektur, Screenshots der einzelnen Activities (Ansichten innerhalb der App) etc. aufgeführt.

Die App ist grundsätzlich so aufgebaut, dass sie sich in Frontend und Backend unterscheiden läss.

**Frontend:**

Für das Frontend wurde Android Studio also die App selbst benutzt:

* Android Studio
* Programmiersprache: Java
* Gradle für dependencies

Im aktuellen Prototyp v1.0.0 gibt es einen Homescreen, drei seperate Activities, die sich über die Buttons: Food, Time und Skills erreichen lässt. Außerdem gibt es im Home Screen einen Login Button, über den man sich registrieren oder anmelden kann. Alle Activities sind bis jetzt Mockups und verfügen über keine tiefgehende Funktionalität.

**Backend:**

Für das Backend wurde Springboot benutzt, welches ein mächtiges Framework ist und sich ideal für den Zweck einer App eignet. Springboot bietet unter anderem folgende Vorteile:

* Schnelle Projektinitialisierung​
* Umfangreiche Auto-Konfiguration​
* Eingebaute HTTP-Server (Tomcat, Jetty)​
* Einfache Abhängigkeitsverwaltung​
* Unterstützung für Mikroservices​
* Umfangreiche Spring Ökosystem-Integration​
* Vielfältige Datenzugriffsoptionen​
* Sicherheitskonfigurationen leicht gemacht​
* Umfangreiche Testunterstützung​

**Versionskontrolle:**

Für die Versionskontrolle haben wir uns für Git entschieden. Ein vorläufiges Repository ist öffentlich zugänglich unter dem folgenden Link:

https://github.com/tomrussler/project-cologne-team-rfh

​

​

​

​

​

​

​

**Frontend (Android App)**

**Technologien:**

**Android Studio:**

* + Als Hauptentwicklungsumgebung.

**Kotlin oder Java:**

* + Als Programmiersprachen. Kotlin wird von Google empfohlen und bietet moderne Sprachfeatures.

**Jetpack Libraries:**

* + Bietet nützliche Tools und Bibliotheken, die die Entwicklung von Android-Apps vereinfachen, z.B. Room für die Datenbankinteraktion, ViewModel für die Datenhaltung und Navigation Component für die Navigation zwischen App-Bereichen.

**MVVM-Architektur:**

* + Das Model-View-ViewModel (MVVM) Muster unterstützt eine klare Trennung von Belangen und erleichtert die Entwicklung und Wartung.

**Backend (Server)**

**Technologien:**

**Spring Boot:**

* + Für die Erstellung des Backend-Services, bietet Unterstützung für die Entwicklung von RESTful Web Services.

**Sicherheitsmechanismen:**

* + OAuth2 für Authentifizierung und JWT für sichere Übertragung von Informationen zwischen Client und Server.

**Datenbank:**

* + PostgreSQL für relationale Datenhaltung oder MongoDB, falls eine NoSQL-Datenbank bevorzugt wird. Dies hängt von deinen spezifischen Anforderungen ab (z.B. Beziehungen zwischen Daten, Skalierbarkeit).

**Hibernate ORM:**

* + Für die Abbildung von Java-Klassen auf Datenbanktabellen, falls eine relationale Datenbank verwendet wird.

**Architekturübersicht:**

1. **Client-Schicht (Android App):** Nutzerinteraktionen, UI/UX, Anfragen an das Backend.
2. **Business-Logik-Schicht (Backend):** Verarbeitung von Geschäftslogik, Authentifizierung, Autorisierung, Datenverarbeitung.
3. **Datenzugriffsschicht (Backend):** Datenbankoperationen, ORM / Datenbankzugriff, Modelldefinitionen.
4. **Datenschicht (Datenbank):** Speicherung von Benutzerdaten, Transaktionen, Beiträgen etc.

**Kommunikation:**

* Die Kommunikation zwischen der Android App und dem Backend erfolgt über RESTful APIs. Das Backend stellt Endpunkte zur Verfügung, über die die App Daten abfragen oder senden kann.
* Daten werden typischerweise im JSON-Format ausgetauscht.
* Sicherheitsaspekte wie Authentifizierung und Autorisierung sind über das Backend zu regeln, wobei Technologien wie HTTPS, OAuth2 und JWT zum Einsatz kommen.