1. Ausgangslage

Ziel ist es, ein Backend für eine Blog-Website zu erstellen, auf der mehrere User Blog-Posts erstellen können. Die Posts sind zugänglich für die Öffentlichkeit. Diese Page wird verwaltet von Admins, die Kategorien erstellen und verwalten können. Diese Kategorien können dann von Usern zu Blog-Posts zugeteilt werden. User können Teil einer Gruppe werden.

2. Auftrag

Ziel dieses Projekt ist es sein ein Backend mithilfe von SpringBoot und PostgreSQL zu erstellen, die die unten aufgeführten Bedingungen erfüllt. Dabei wird auf Multiuser-fähigkeit und Sicherheit geachtet. Als Vorgabe erhalten sie ein Skelet, indem die Grundlagen eines Spring Projekts erstellt wurde. Dieses Projekt enthält Funktionalitäten, um bestehende User einzuloggen und ermöglicht das manuelle Erstellen von Usern, Rollen und Autoritäten.

2.1 Funktionale Anforderungen

User Rollen & Privilegien

- Erstellen sie mehrere User, Rollen und Autoritäten durch die im Skelet ermöglichten Funktionalitäten oder ihre eigenen Endpoints. Überlegen sie dabei, welche Rollen und Autoritäten sinnvoll sind, um die untenstehenden Aufträge zu testen.
- User, Rollen und Autoritäten müssen in einer PostgreSQL Datenbank persistiert werden. Sie können erstellt, gelöscht und verändert werden, Änderungen werden in der Datenbank abgespeichert.
- Die Entscheidung, welche und wie viele Rollen und Privilegien sinnvoll sind, liegt bei ihnen.

Security

- Stellen sie mithilfe von Tests sicher, dass bei einem Login-Versuch der User mit den angegebenen Daten authentifiziert und autorisiert wird.
- Jeder REST-Endpoint ist nur zugänglich mit entsprechenden Autoritäten.

User Management

- Erweitern sie die Funktionalität des bestehenden Skelets, um typische CRUD Operationen an Usern über Endpoints zu ermöglichen.
- Neue User werden bereits in der Datenbank abgespeichert, ihre Passwörter sind jedoch noch nicht verschlüsselt. Implementieren sie eine Passwortverschlüsselung.
- Die persönlichen Informationen eines Users sind nur für Administratoren oder den User selbst zugänglich.

Gruppenspezifische Aufgabe
Diese Teil-Aufgaben werden selbstständig von Gruppen oder Einzelpersonen durchgeführt.

1. **User Profile**
   - Erstellen sie eine UserProfile Model, das zusätzliche Informationen über einen User enthält (Adresse, Geburtsdatum, Profilbild-URL, Alter etc.).
   - Jeder Eintrag in UserProfile kann eindeutig zu einem User zugeordnet werden.
   - Erstellen sie Endpoints in ihrer Applikation, um typische CRUD Operationen an UserProfile durchzuführen.
   - Der Endpoint mit dem alle UserProfiles angeteigt werden soll nur für Administratoren zugänglich sein. Dieser Endpoint soll Pagination benutzen und nach beliebigen Parametern des Userprofiles auf und absteigend sortierbar sein.
   - Das Profil eines Users ist nur für Administratoren oder den User selbst zugänglich.

2. **Blog Posts**
   - Erstellen sie eine Blog Model, das die Information eines Blogeintrags enthält (Text, Titel, Kategorie, Autor etc.)
   - Jeder Eintrag in Blog kann eindeutig zu einem User als AutorIn zugeordnet werden.
   - Erstellen sie Endpoints in ihrer Applikation, um typische CRUD Operationen an Blogs durchzuführen. Die GET Methode soll Pagination und Sorting nutzen.
   - Nur der/die AutorIn oder ein Administrator soll ein Blog bearbeiten oder löschen können
   - Selbst unauthentifizierte Benutzer sollen Blogs (mit GET-Methode) lesen können.

3. **Groups**
   - Erstellen sie eine Group Model, das Informationen über eine Gruppe von Usern enthält (Mitglieder, Gruppenname, Motto, etc.).
   - Jeder User kann nur in max. in einer Gruppe gleichzeitig Mitglied sein.
   - Erstellen Sie Endpoints in ihrer Applikation, um typische CRUD Operationen an Groups durchzuführen.
   - Erstellen Sie einen Endpoint der alle Mitglieder einer Gruppe auflistet. Benutzen sie dafür Pagination.
   - Nur ein Administrator soll eine Group erstellen, bearbeiten oder löschen können.
   - Nur Administratoren oder Mitglieder einer Gruppe können Informationen der Gruppe aufrufen/bearbeiten/löschen

4. **MyListEntry**
   - Erstellen sie ein MyListEntry Model, das Information über einen Listeneintrag enthält (Titel, Text, Erstellungsdatum, Wichtigkeit).
   - Jeder User kann mehrere Listeneinträge kreieren, jeder Listeneintrag gehört immer zu einem User.
   - Erstellen sie Endpoints in ihrer Applikation, um typische CRUD Operationen an Listeneinträgen durchzuführen.
   - Erstellen sie zusätzlich einen Endpoint, der alle Listeneinträge von einem beliebigen User nach Wichtigkeit sortiert ausgibt. Diese Liste, soll für alle eingeloggten User ersichtlich sein.
   - Nur der User, der ein Listeneintrag kreiert hat oder ein Administrator soll ein Listeneintrag bearbeiten oder löschen können.

2.2 Nicht funktionale Anforderungen:

Implementation

- Daten werden in einer PostgreSQL Datenbank persistiert, das OR Mapping wird mit JPA realisiert.
- Der Sourcecode wird täglich in einem GIT-Repository eingecheckt.

Testing

- Mindestens ein Backend-Endpoint wird ausführlich getestet. Dies beinhaltet im Minimum:
  - Implementierte Funktionalität wird mit Component-Tests in Postman überprüft.
  - Der Endpoint wird mit mehreren Usern/Rollen getestet
  - Mindestens ein Erfolgsfall und ein Error Fall wird getestet.
    Für diese Fälle werden Usecases nach UML Standard beschrieben

Multiuserfähigkeit

- Aspekte der Multiuserfähigkeit, wie z.B. Einhaltung der ACID-Prinzipien werden berücksichtigt

Dokumentation

- Die Dokumentation umfasst im Minimum:
  - Ein Readme (im Code-Repo eingecheckt), welches die wichtigsten Informationen zum Projekt so wie eine Setup-Anleitung für die Ausführung des Codes und Tests enthält.
  - Alle implementierten Endpoints sind beschrieben mithilfe von Swagger.
- Ein Domänenmodell wurde erstellt, das die Applikation als Ganzes beschreibt Also die allgemeinen Domänen sowie die Domäne der jeweiligen Gruppe).
- Ein Klassen-Diagramm wurde erstellt, das die implementierte Funktionalität beschreibt.
- Ein Sequenz-Modell, das einen implementierten Endpoint beschreibt.
- Eine UseCase Beschreibung für einen getesteten Endpoint (siehe oben).