Projekt Auftrag üK 223

1. Ausgangslage

Euer Team wurde beauftragt, die Social Media-Website «OurSpace» zu entwickeln. Ihr werdet als Team im Rahmen des üKs einen Teil dieser Website erarbeiten. Auf dieser Website sollen mehrere User Blog-Posts erstellen können. Zudem haben Admins auf weiten Teilen der Applikation zusätzliche Berechtigungen.

2. Auftrag

Ziel dieses Projekt ist es sein eine Full-Stack Komponente mithilfe von React, SpringBoot und PostgreSQL zu erstellen, welche unten aufgeführten Bedingungen erfüllt. Dabei wird auf Multiuser-fähigkeit, Sicherheit sowie Dokumentation der Arbeit geachtet. Als Vorgabe erhaltet ihr ein funktionierendes rudimentäres Full-Stack Projekt. Dieses Projekt enthält Funktionalitäten um bestehende User einzuloggen und ermöglicht das Erstellen, Bearbeiten und Löschen von Usern. Login Funktionalität sowie einfaches Routing wurde ebenfalls implementiert.

3. Abgabe

- **Tag 3, 16.30 Uhr**: Dokumentation (PDF-Dokument von Punkt 6.4) in Frontend Repository
- **Tag 5, 16.30 Uhr**: Rest des des Projektauftrags

4. Allgemeine Anforderungen

4.1 Funktionale Anforderungen:

User Rollen & Privilegien:

- Bestehende Rollen und Autoritäten können genutzt oder erweitert werden, um Autorisierung für die restlichen Anforderungen zu implementieren und zu testen.
- Die persönlichen Informationen eines Users sind nur für Administratoren oder den User selbst zugänglich. Admins können ausserdem andere Benutzer bearbeiten, erstellen und löschen. Beides muss im Frontend möglich sein.

Frontend

Im Minimum enthält die Applikation:

- Login-Page, die öffentlich zugänglich ist (bereits vorhanden)
- Eine öffentlich zugängliche Homepage (bereits vorhanden)
- eine Homepage für alle eingeloggten User
- Eine Admin Page (nur für Admins zugänglich).
- Mindestens eine Komponente, um gruppenspezifische Funktionalitäten im Frontend zu ermöglichen.

Für die Bewertung des Auftrags sind Benutzerfreundlichkeit der UI und Visualisierung nicht relevant. Es wird jedoch Wert auf korrektes Rechte-Management und Error Handling gelegt (User-Feedback).

4.2 Security

- Jeder gruppenspezifische REST-Endpoint soll nur mit sinnvollen Autoritäten zugänglich sein. Diese sind selbst zu definieren und wird mit automatisierten Tests überprüft.
- Es gibt Bereiche des Front-Ends die nur für eingeloggte Benutzer zugänglich sind.
- Es gibt Bereiche des Front-Ends die nur für Admins zugänglich sind.
- Der Authentifizierung-Mechanismus wird mit JSON-Web-Tokens implementiert. (bereits vorhanden)

5. Gruppenspezifische Aufgabe

Diese Use-Cases werden von zugeteilten Gruppen implementiert. Dazu wird mindestens eine neue Spring Boot Entity und dazugehörige Klassen (wie Controller, DTO etc.) sowie eine oder mehrere Pages in der Frontend Applikation benötigt.

4.1 User Profile

- UC1: User erstellt eigenes Profil (Adresse, Geburtsdatum, Profilbild-URL und Alter).
- UC2: User liest, aktualisiert oder löscht eigenes Profil.
- UC3: Administrator liest, aktualisiert oder löscht beliebige UserProfile.
- UC4: Administrator sucht, filtert und sortiert UserProfiles (mit Pagination).
- UC5: System stellt sicher, dass nur Besitzer oder Admin Zugriff auf ein UserProfile haben.

4.2 Blog Posts

- UC1: User erstellt neuen Blogpost (Text, Titel, Kategorie, Autor).
- UC2: User bearbeitet oder löscht eigenen Blogpost.
- UC3: Administrator bearbeitet oder löscht beliebigen Blogpost.
- UC4: Alle Nutzer (auch anonyme/uneingeloggte Gäste) lesen Blogposts mit Pagination und Sortierung.
- UC5: System stellt sicher, dass nur Autor eines Blogposts oder Admin Änderungen durchführen dürfen.

4.3 Groups

- UC1: Administrator erstellt eine Gruppe (Mitglieder, Gruppenname, Motto, Logo).
- UC2: Administrator bearbeitet oder löscht eine Gruppe.
- UC3: User tritt einer Gruppe bei (max. eine Gruppe gleichzeitig erlaubt).
- UC4: Mitglieder und Administratoren sehen Gruppendetails inkl. Motto, Logo und Mitgliederliste.
- UC5: System listet Mitglieder einer Gruppe mit Pagination.

4.4 Events

- UC1: User erstellt ein Event (Gästeliste [Liste von User], Eventname, Datum, Ort).
- UC2: Event-Ersteller oder Admin bearbeitet Event und fügt Gäste hinzu.
- UC3: Alle User sehen Event-Informationen.
- UC4: System listet Gäste eines Events mit Pagination.
- UC5: System verhindert, dass Administratoren als Gäste beitreten.

4.5 Image Gallery

- UC1: User erstellt ein Image-Post (URL, Author, Beschreib, Likes (welche User haben dieses Bild geliked)
- UC2: User oder Administrator bearbeitet oder löscht ein Image-Post. (User nur eigenen)
- UC3: Eingeloggte User sehen Image-Posts anderer User.
- UC4: System listet alle Image-Posts eines bestimmten Users.
- UC5: User liked oder entfernt Like von einem Image-Post.

4.6 Custom List

- UC1: User erstellt einen Listeneintrag (Titel, Text, Erstellungsdatum, Wichtigkeit)
- UC2: User oder Administrator bearbeitet oder löscht einen Listeneintrag. (User nur eigenen)
- UC3: Eingeloggte User sehen Listeneinträge eines anderen Users.
- UC4: System listet Listeneinträge eines Users sortiert nach Wichtigkeit.
- UC5: User durchsucht eigene Listeneinträge nach Titel oder Wichtigkeit.

6. Nicht funktionale Anforderungen

6.1 Implementation

- Daten werden in einer PostgreSQL Datenbank persistiert, das OR-Mapping wird mit JPA realisiert. (bereits vorhanden)
- Ein Frontend mit React (Typescript) wird genutzt.
- Ein Backend Springboot (Java) wird genutzt.
- Der Sourcecode wird täglich in einem GIT-Repository commited.

6.2 Testing

- Generelle Funktionalität aller selbst implementierten Use-Cases wird mit Cypress, Postman und/oder JUnit getestet.
- Besonderer Wert wird auf das Testen von Zugriffsberechtigungen gelegt.
- Mindestens ein Use-Case wird ausführlicher mit Cypress getestet. Dies beinhaltet im Minimum:
  - Der Endpoint wird mit mehreren Usern & Rollen getestet
  - Mindestens ein Erfolgsfall und ein Error-Fall wird getestet.
  - Für diese Fälle werden Use-Cases nach UML-Standard beschrieben.

6.3 Multiuserfähigkeit

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

6.4 Dokumentation

Die Dokumentation umfasst im Minimum:

- Ein Readme (im Frontend-Code-Repo eingecheckt), welches die folgenden Informationen enthält:
  - Wie wird eure Applikation gestartet?
  - Welche Login Daten haben welche Rollen?
  - Wie können alle Tests ausgeführt werden?
  - Welche Frontend-URLs implementieren welche gruppenspezifischen Funktionalitäten?
- Swagger wird verwendet, um automatisierte Dokumentation der Endpoints zu generieren. (Grundlage bereits implementiert, erweitern um Endpoints zu dokumentieren)
- **Abgabe Tag 3, 16.30 Uhr:** Ein PDF (in Deutsch oder Englisch) welches die folgenden Informationen enthält:
  - Ein Domänenmodell das die Applikation als Ganzes beschreibt (Also die allgemeinen Domänen sowie die Domäne der jeweiligen Gruppe).
  - Eine Beschreibung der Testing Strategie. (genutzte Tools, Alle getesteten Endpoints, erwartetes Verhalten).
  - Use-Case Beschreibungen aller gruppenspezifische Funktionalitäten (UC 1-5 von oben).
  - Ein Use-Case-Diagramm aller gruppenspezifischen Funktionalitäten der Applikation. (UC 1-5 von oben)
  - Ein Sequenz-Diagramm, das einen Prozess im Backend visualisiert, der teil eines UseCases ist.