DOKUMENTATION M223

Jasmin Jeyakumar & Dominic Heule
Gruppe 6

Inhalt

1 Projektauftrag

1.1 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. Die Posts sind zugänglich für die Öffentlichkeit. Die 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.

1.2 Auftrag

Ziel dieses Projekt ist es sein eine Full-Stack Komponente mithilfe von React, SpringBoot und PostgreSQL zu erstellen, die die unten aufgeführten Bedingungen erfüllt. Dabei wird auf Multiuser-fähigkeit und Sicherheit sowie Dokumentation der Arbeit geachtet. Als Vorgabe erhalten Sie 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.

1.2.1 Pflicht-Anforderungen

1.2.1.1 Funktionale Anforderungen

User Rollen & Privilegien:

- Bestehende Rollen und Autoritäten werden bearbeitet oder erweitert, um untenstehende Anforderungen zu erfüllen 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.

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.

Security

- Jeder REST-Endpoint soll nur mit sinnvollen Autoritäten zugänglich sein. Dies 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)

Gruppenspezifische Aufgabe

Diese Teil-Aufgaben werden selbstständig von Gruppen oder Einzelpersonen

durchgeführt.

6. Custom List

- 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.

1.2.1.2 Nicht-Funktionale Anforderungen

Implementation

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

Testing

- Generelle Funktionalität aller selbst implementierten Endpoints wird mit
- Cypress (zwingend), 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 Errorfall wird getestet.
  - Für diese Fälle werden Use-Cases nach UML-Standard beschrieben.

Multiuserfähigkeit

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

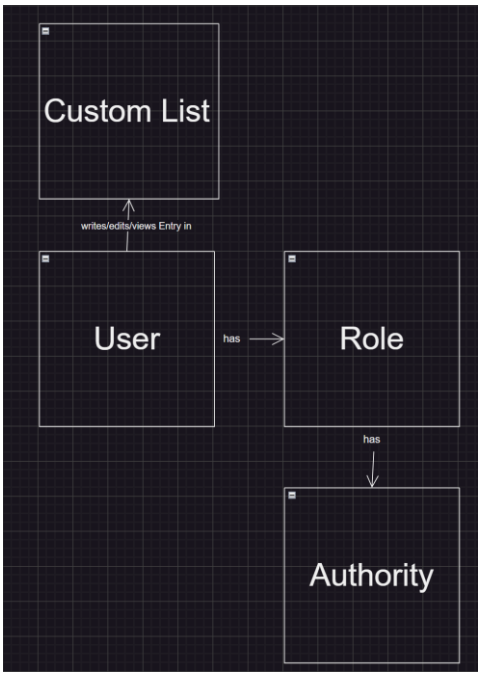
Dokumentation

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.
- Ein Domänenmodell das die Applikation als Ganzes beschreibt (Also die allgemeinen Domänen sowie die Domäne der jeweiligen Gruppe).
- Eine Beschreibung der Testing Strategie inkl. Use-Case Beschreibung für einen ausführlich getesteten Endpoint (siehe «Testing»).
- Ein Use-Case-Diagramm aller gruppenspezifischen Funktionalitäten der Full-Stack-Applikation.
- Ein Sequenz-Diagramm, das eine der verlangten Backend-Funktionalitäten visualisiert.
- Swagger wird verwendet, um automatisierte Dokumentation der Endpoints zu generieren.

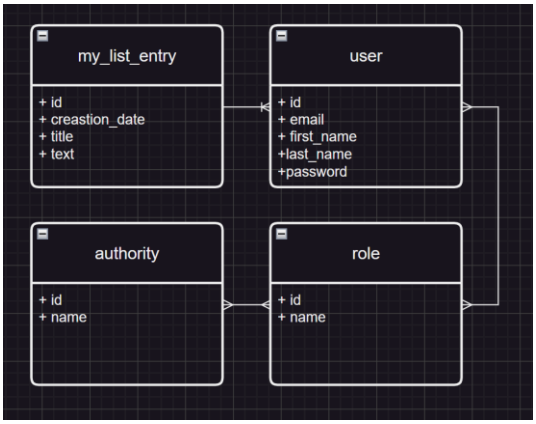
2 Domain Modell

Unser Project ist folgendermassen aufgebaut: wir haben ziemlich zentral einen User dieser hat eine Rolle mit gewissen Autoritäten, die mit dieser Rolle in Verbindung stehen. Dieser User sieht, erstellt, bearbeitet und löscht Einträge in einer Liste. Hier ist das entsprechende Domain Modell:



3 ERD

Unser Daten werden in einer PostgreSQL Datenbank abgespeichert, diese Datenbank hat folgende Entities: my_list_entry, user, role und authorities. So sie das Entity Relation Diagramm dazu aus:



4 Testing

In unserem Project werden wir verschiedene Testarten verwenden. All unsere Tests sind in unserem Git Repository abgelegt: https://github.com/doeme07/M223_Jasmin_Dominic_Backend

4.1 Funktionale Tests

4.1.1 E2E Tests

Die E2E Tests werden anhand von dem Tool Cypress durchgeführt. Diese Tests werden den ganzen Prozess unserer Applikation als ein fiktiver User simulieren. Ebenfalls werden verschiedene Simulationen durchgeführt mit verschiedenem User, um die Richtigkeit der Rollen und Autoritäten im ganzen Programm sicher zu stellen.

4.1.2 Allgemeine Cypress Tests im Frontend

Ebenfalls werden wir mit Cypress die Benutzerfreundlichkeit und Rückmeldung der Applikation Testen. Im Beispiel von einen falschen Dateneintrag in ein Formular, wird erwartet das die entsprechende Rückmeldung vom Frontend richtig ausgelöst wird.

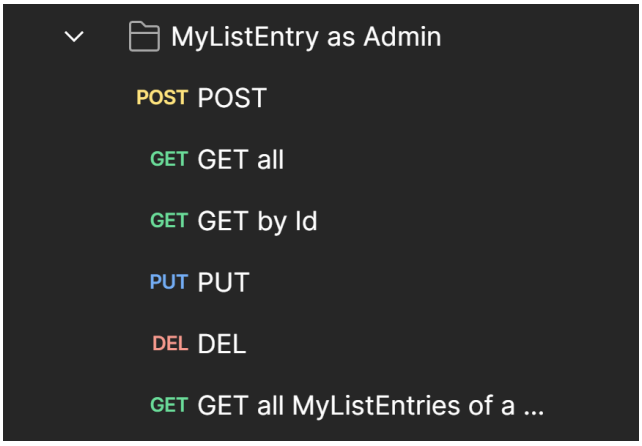
4.1.3 Performance

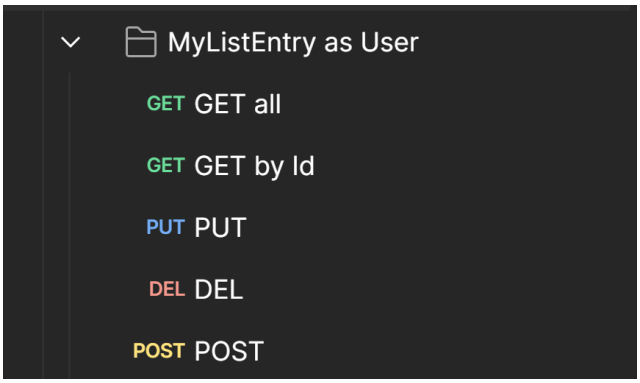
Ebenfalls werden wir die Performance unserer Applikation testen, anhand von Stresstests, diese Tests sollen die Performance unserer Applikation sicherstellen, wenn mehrere Requests gleichzeitig ans Backend geschickt werden mit Postman. Ebenfalls wird durch diese mehrere Requests die Daten Konsistenz unserer Applikation getestet. Diese Tests werden mit PGBench durchgeführt.

4.1.4 Component Tests

Unsere Component Tests werden wir anhand von Postman durchführen. Wir werden die einzelnen Endpoints testen, einerseits mit verschiedenen Usern, um sicherzustellen, ob die Rollen und Autoritäten richtig zu den Usern gebunden sind. Durch diese Tests erfahren wir, ob das Verhalten unserer Applikation bei verschiedenen Usern richtig ist. Unsere Postman Test werden auch die richtige Struktur unserer Responses vom Backend kontrollieren.

So werden unsere Postman Tests aussehen:





4.2 Nicht-Funktionale Tests

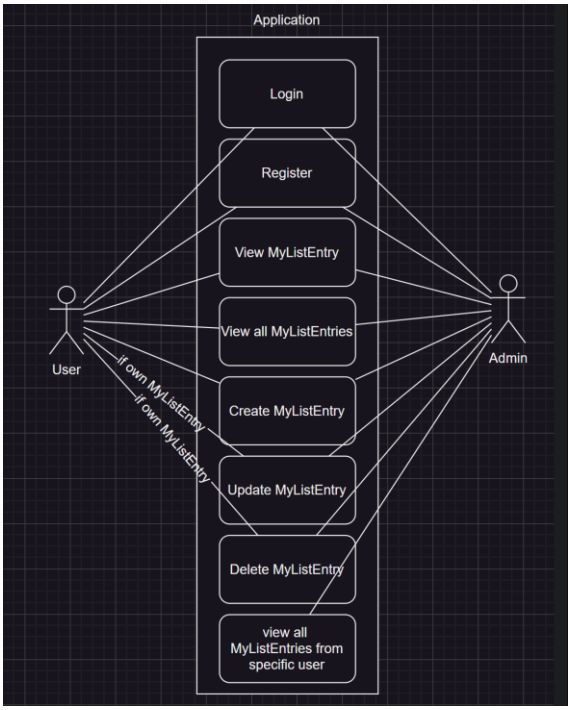
4.2.1 Lighthouse

Ebenfalls werden wir unsere Applikation anhand von Lighthouse verifizieren, Grund dafür ist, dass wir die Performance überprüfen und sehen wollen in welchen Aspekten unsere Applikation sich verbessern kann, falls die Werte noch nicht unseren Erwartungen entsprechen.

5 Use-Case Diagramm:

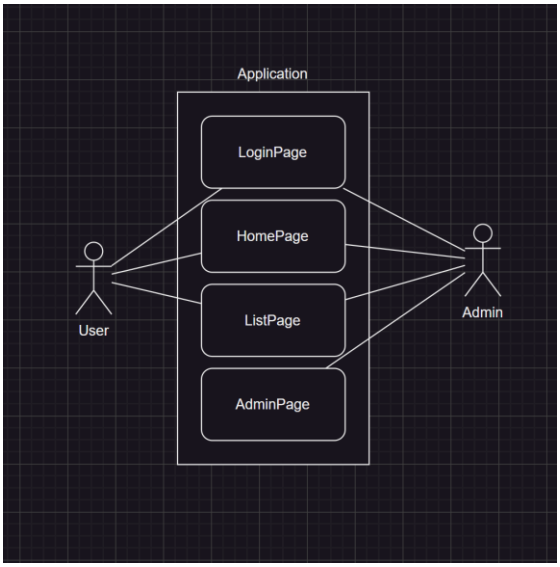
5.1 Backend

Das ist das Use-Case Diagramm zu unserer Applikation. Hier sieht man auf welche Funktionalitäten, welcher User Zugriff hat.



5.2 Frontend

Hier sieht man das Use-Case Diagramm vom Frontend. Hier sind die Pages aufgeführt, auf die ein gewisser User Zugriff hat.



6 Use-Case Definition:

6.1 User

Das ist die Use-Case Definition aus der Sicht eines Users, für das Erstellen eines MyListEntrys:

Akteur:	User
Beschreibung:	Ein normaler User erstellt einen Eintrag in die Custom List.
Vorbedingungen:	• Der User ist eingeloggt • Der User hat die richtige Rolle (User) • Die User-Rolle hat die richtigen Autoritäten.
Nachbedingungen:	• MyListEntry Daten sind abgespeichert • User wird mitgeteilt, ob es erfolgreich oder nicht war.
Normaler Verlauf:	1. Der User meldet sich an. 2. Der User klickt in der Navigations-Leiste auf MyList. 3. Der User drückt auf das «+» Icon, um eine MyListEntry zu erstellen. 4. Der User füllt den Titel, Text, Datum und Wichtigkeit. 5. Anschliessend drückt der User auf «save» und die Daten des Formulars werden abgespeichert.
Alternativer Verlauf:	1. Wenn die MyListEntry nicht dem Format entsprechen oder fehlen, funktioniert es nicht und es kommt eine Fehlermeldung. 2. Wenn es die MyListEntry bereits gibt, werden sie nicht abgespeichert und es kommt eine Fehlermeldung.
Ausnahmen:	Keine

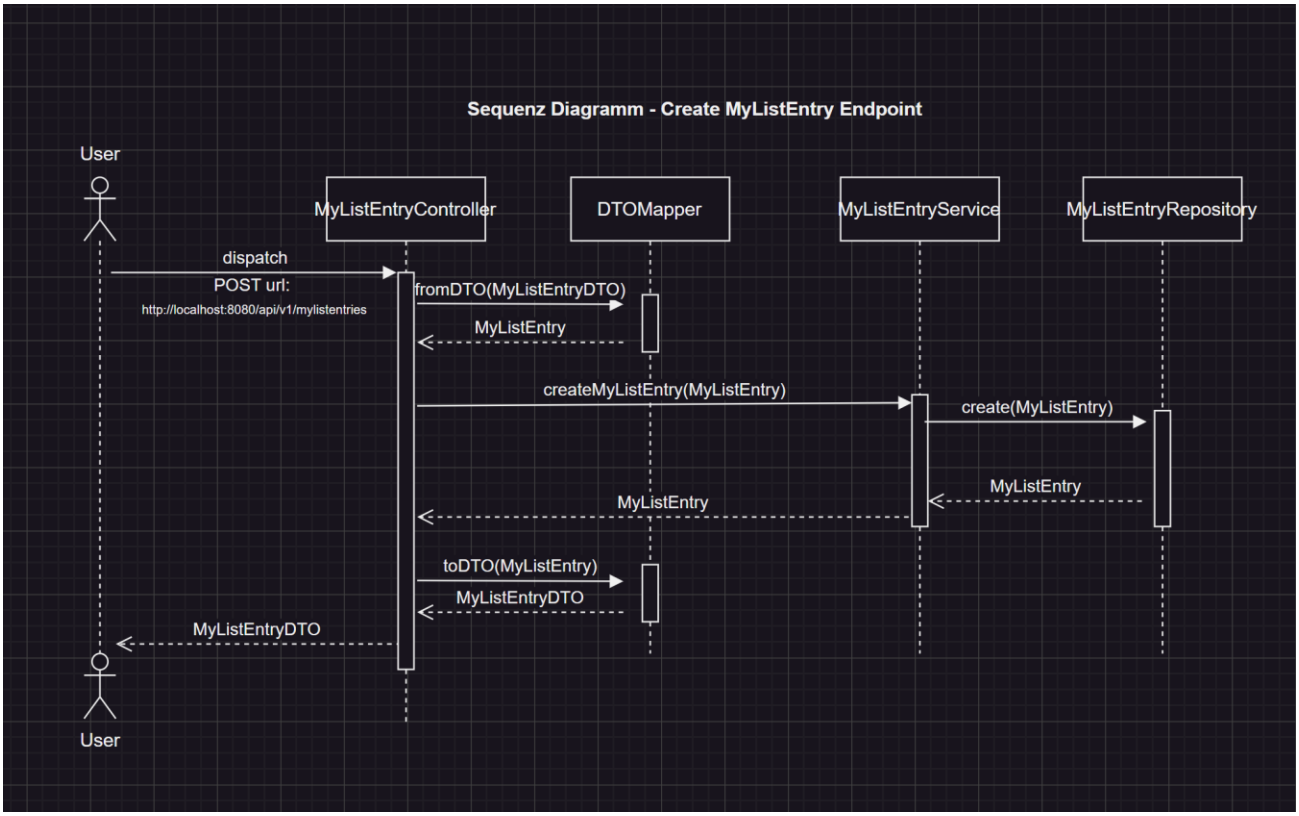
6.2 Admin

Das ist die Use-Case Definition aus der Sicht eines Admins, für das Erstellen eines MyListEntrys:

Akteur:	Admin
Beschreibung:	Ein Admin erstellt einen Eintrag in die Custom List.
Vorbedingungen:	• Der Admin ist eingeloggt • Der Admin hat die richtige Rolle Admin • Die Admin-Rolle hat die richtigen Autoritäten.
Nachbedingungen:	• MyListEntry Daten sind abgespeichert • Admin wird mitgeteilt, ob es erfolgreich oder nicht war.
Normaler Verlauf:	6. Der Admin meldet sich an. 7. Der Admin klickt in der Navigations-Leiste auf MyList. 8. Der Admin drückt auf das «+» Icon, um eine MyListEntry zu erstellen. 9. Der Admin füllt den Titel, Text, Datum und Wichtigkeit. 10. Anschliessend drückt der Admin auf «save» und die Daten des Formulars werden abgespeichert.
Alternativer Verlauf:	3. Wenn die MyListEntry nicht dem Format entsprechen oder fehlen, funktioniert es nicht und es kommt eine Fehlermeldung. 4. Wenn es die MyListEntry bereits gibt, werden sie nicht abgespeichert und es kommt eine Fehlermeldung.
Ausnahmen:	Keine

7 Sequenz Diagramm:

So sieht unser Sequenz Diagramm, in diesen wird der Prozess beschrieben wie ein MyListEntry erstellt wird.



8 Swagger:

So sieht unsere Swagger Dokumentation aus. Hier sind unsere Endpoints aufgeführt:

!!! Aus einer Diskussion mit Luca sind wir auf die Einigung gekommen, dass die Swagger Dokumentation erst am Ende des üKs angeschaut wird. !!!