Dokumentation Gruppe 2 ÜK 223

Autoren: Tahssin, Gabriel, Adam

Inhaltsverzeichnis

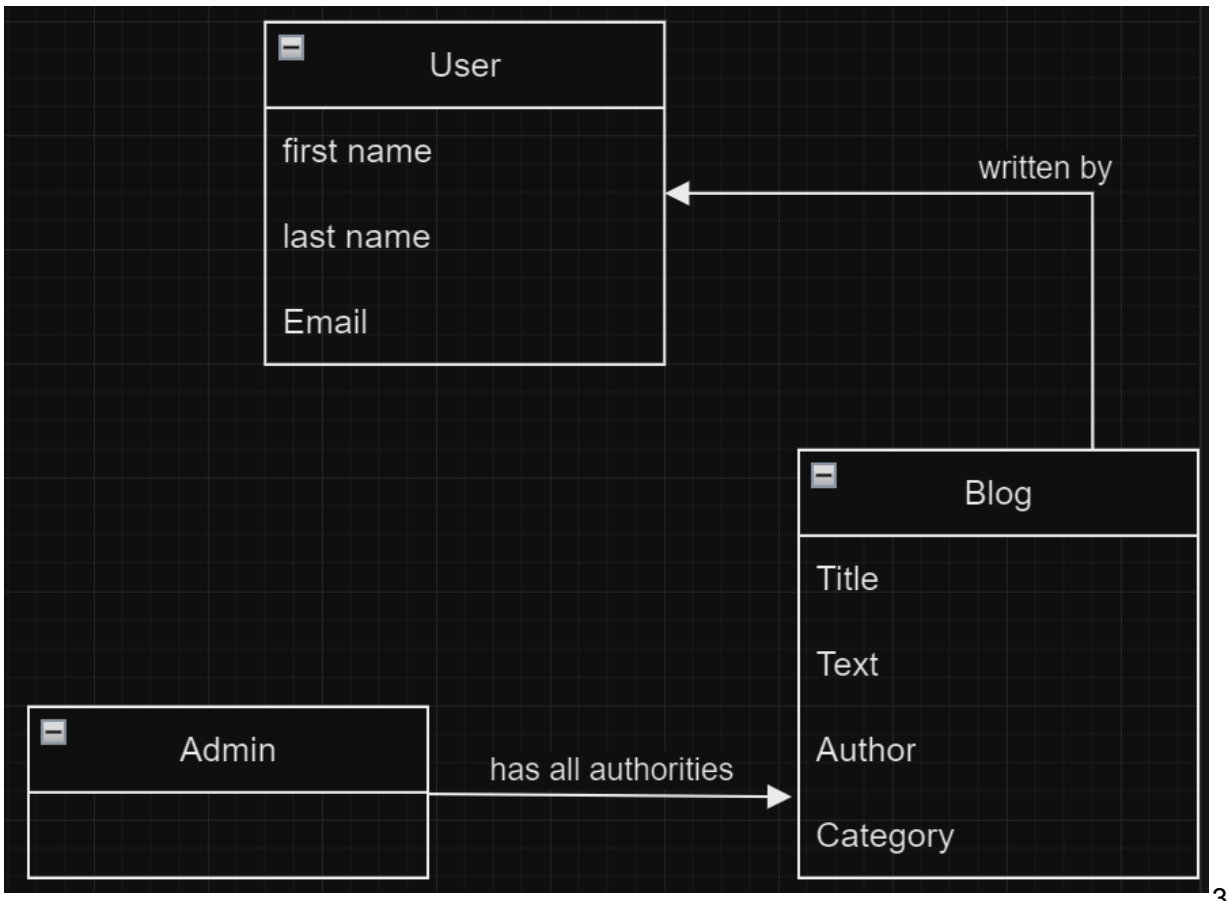
1. Projektauftrag

Ziel dieses Projekt ist es eine Full-Stack Komponente mithilfe von React, Spring Boot und PostgreSQL erstellen.
Als Vorgabe erhielten wir ein funktionierendes 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 wurden ebenfalls implementiert. Als Team 2 hatte unsere Gruppe, Adam Swiderski, Tahssin Al-Khatib und Gabriel Arocha, die Aufgabe die Entität des Blog Posts zu integrieren.

2. Domänenmodell

2.1 Beschreibung der Domänen

In einem Domänenmodell werden auf einem hohen Niveau die Beziehungen zwischen einzelnen Entitäten in unserer App dargestellt. Das Domänenmodell wird vor allem gebraucht, um es Kunden zu präsentieren. Unser Domänenmodell sieht folgendermassen aus:

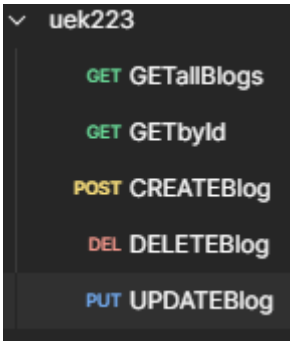


3. Testing

3.1 Testing Strategie

Für unser Projekt werden wir alle CRUD Endpoints testen. Wir werden JUnit, Postman und Cypress brauchen. Alle Tests sind in unserem Github Repo für das Frontend (https://github.com/Gabriel-A-NY/uek223-team2-blogpost-frontend_official) abgelegt. Unsere Strategie liegt darin, E2E-Tests mit Cypress zu absolvieren. Zudem wollen wir die einzelnen Endpoints mit Postman testen und einzelne Komponenten mit JUnit.

Unsere Tests werden für die Endpoints auf Postman stattfinden:



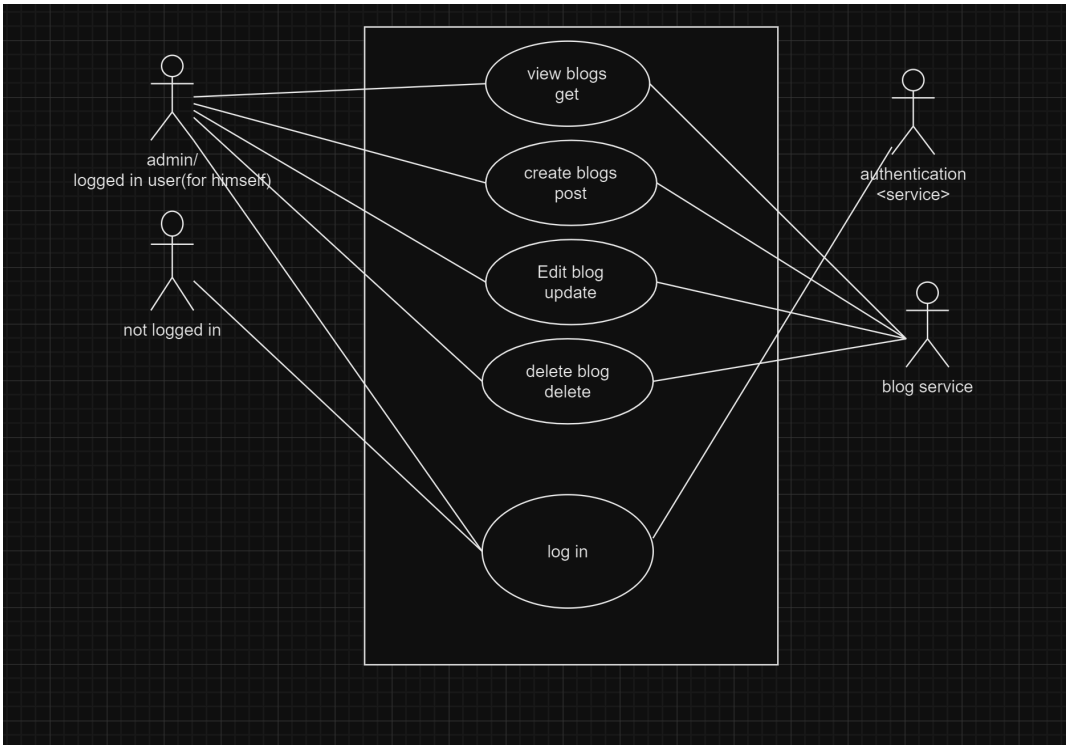
Unsere einzelnen Endpoints sollen die grundlegenden CRUD-Methoden ausführen:
retrieveById => einen bestimmten Blog Post anschauen
retrieveAll =>alle Blogs ansehen
createBlogPost => Blog Post erstellen
updateBlogPost => Blog Post bearbeiten
deleteBlogPost => Blog Post löschen

3.2 Genutzte Tools

JUnit, Postman, Cypress. Die Verwendung dieser Tools ist für uns sinnvoll, da wir mit Postman und JUnit schon gearbeitet haben. Wir werden mit Cypress das Frontend testen. Es war eine Anforderung in der Projektbeschreibung Cypress zu nutzen.

3.3 Use-Case Diagramm

In einem Use-Case Diagramm werden mögliche Rechte und Autoritäten von einem externen User und einem internen Nutzer getrennt. Hier ist unser Use-Case Diagramm.



3.4 Use-Case Beschreibung

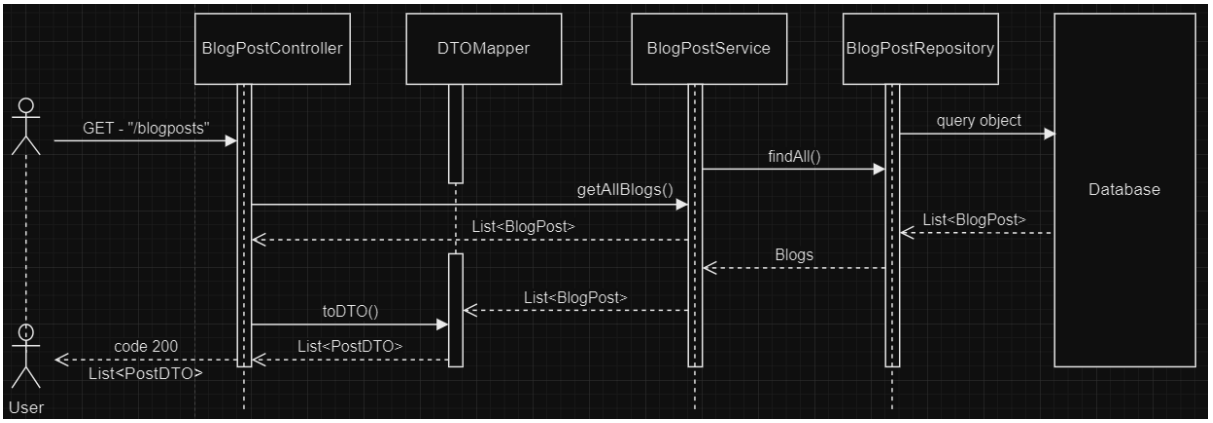
Akteur:	User
Beschreibung:	Der User erstellt einen neuen Blog
Vorbedingungen:	• User ist eingeloggt • User hat korrekt Rolle • User hat entsprechende Autorität (POST in diesem Fall)
Nachbedingung:	• Blog wird erstellt und in der DB gespeichert • User wird über Status informiert (erfolgreich oder fehlgeschlagen) • Status Code 201
Normales Verhalten:	1. User loggt sich ein 2. User drückt “+” Button um einen Blog zu erstellen 3. Der User füllt das Formular für das Erstellen des Blogs vollständig aus 4. Das Formular wird nach dem drücken des “save” Buttons gespeichert 5. Der erstellte Blog wird auf der Homepage angezeigt
Alternatives Verhalten:	1. Falls das Formular nicht vollständig ausgefüllt wird, wird der User dazu aufgefordert alles auszufüllen 2. Wenn ein Blog bereits existiert, wird das Formular entsprechend nicht gespeichert
Mögliche Ausnahmen:	keine

4. Sequenzdiagramm

4.1 Sequenzdiagramm Beschreibung

Ein Sequenzdiagramm ist nützlich, um den detaillierten Programmfluss eines Programms zu verstehen. In diesem Sequenzdiagramm stellen wir den Aufruf eines
Endpoints dar, um alle Blog Posts zu erhalten.

4.2 Visualisierung unseres Sequenzdiagramms:



5. Swagger

Wir haben die Endpoints unseres Backends in Swagger dokumentiert. Man kann die Dokumentation bei einem laufendem Backend unter http://localhost:8080/swagger-ui/index.html einsehen