ÜK 223 Dokumentation – Image Gallery

Thema: MUA-Team 5

Autoren: Fabiano Marino, Juliana Ivankovic, Maximilian Nöthe

Inhaltsverzeichnis

1 Einleitung

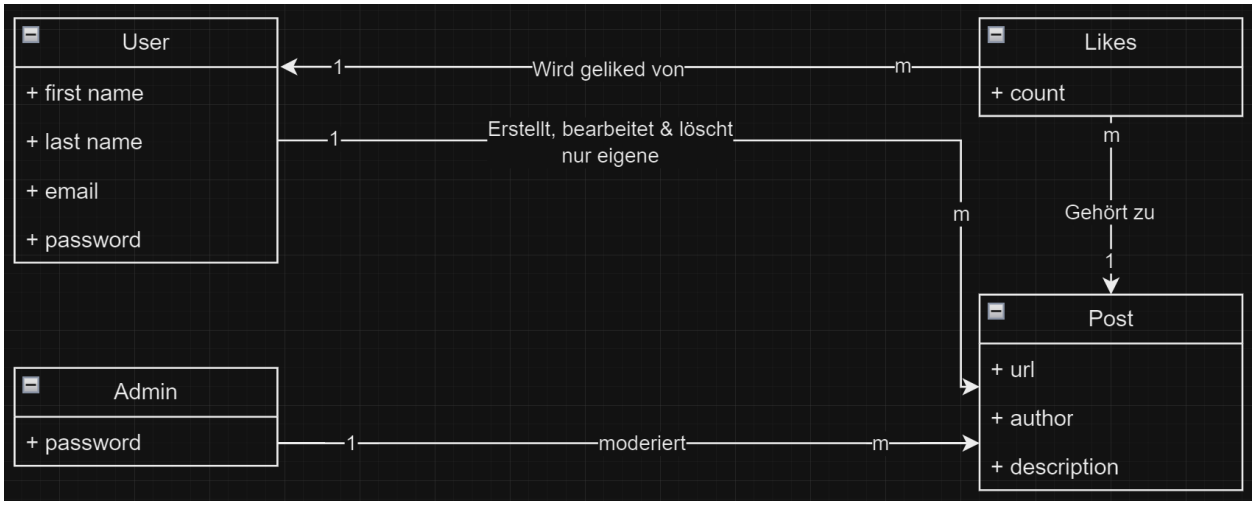
Im üK 223 war es unsere Aufgabe eine Fullstack Multi-User-Applikation für die Social Media Webiste «OurSpace» zu entwickeln. Das Ziel am Ende dieses üK's ist es, dass wir eine objektorientierte Multi-User-Applikation entwerfen, die notwendigen Datenbankanpassungen vornehmen, die Applikation testen und dokumentieren können. Zusätzlich hat jedes Team eine spezifische Gruppenfunktion. Unser Team, bestehend aus Fabiano, Juliana und Maximilian, hatte die zusätzliche Funktion eine Image-Gallery zu implementieren. Nach dem Einloggen hat der User die Möglichkeit alle erstellten Image-Posts zu sehen, eigene Image-Posts zu erstellen, zu bearbeiten und zu löschen, sowie andere Image-Posts zu liken. Diese Posts werden von einem Admin moderiert.

Diese Dokumentation dient zur Übersicht des Projektes, hier sind alle UML's, die für das Verständnis wichtig sind, dargestellt und beschrieben. In den Repositories des Front- und Backends befinden sich die Setup-Anleitungen in Form eines README. Ausserdem gibt es eine Swagger Dokumentation, die wir für unsere Endpoints verwenden.

2 Domänenmodell

2.1 Gruppen-spezifisches Domänenmodell

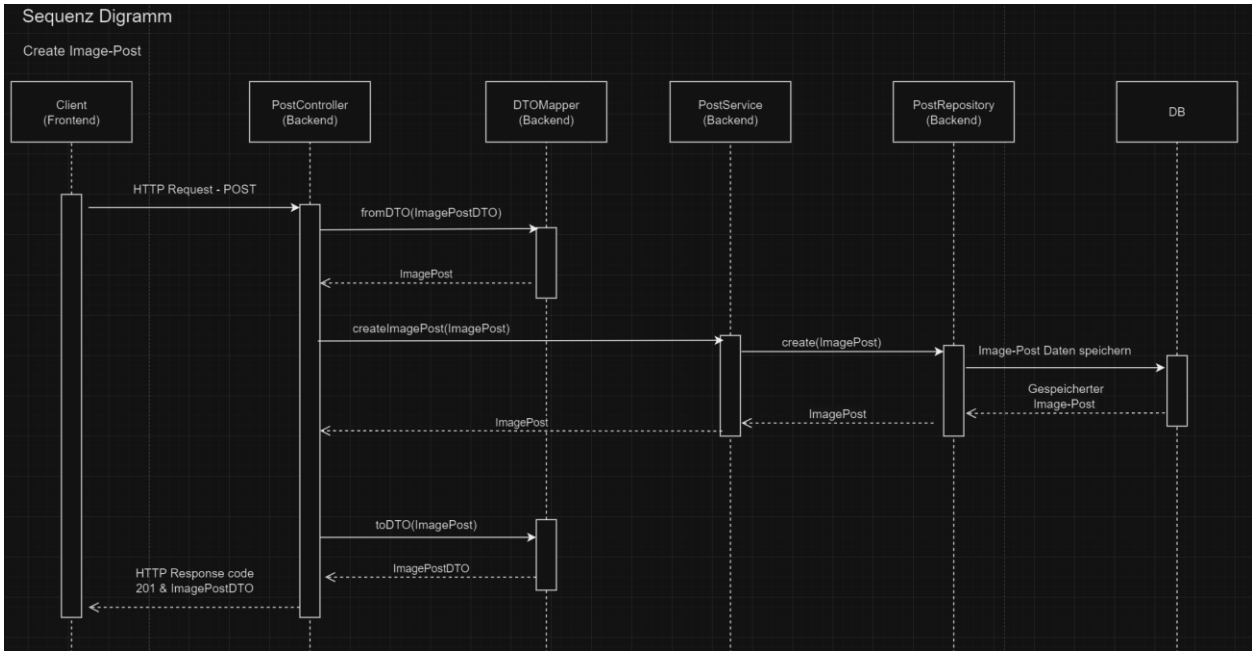
In unserem Domänenmodell wird grafisch dargestellt welche Entitäten unsere Domäne beinhaltet. Die Beziehungen wie auch die Eigenschaften zwischen den Entitäten werden hier angezeigt.



3 Sequenz-Diagramm

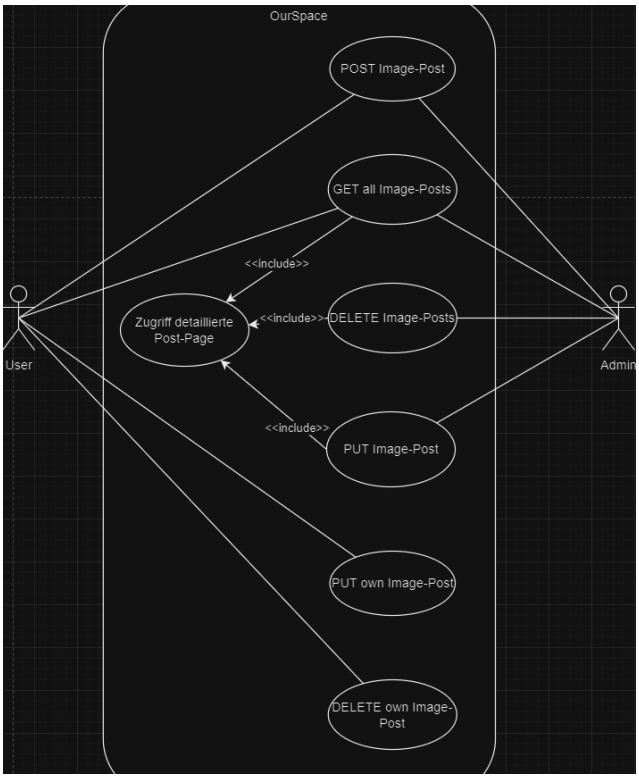
3.1 Visualisierung einer Backend-Funktionalität

Das ist eine Darstellung von einem Request, wenn ein User einen neuen Image-Post erstellen möchte.



4 Use-Case-Diagramm

4.1 Gruppen-spezifische Funktionalitäten der Full-Stack-Applikation



Unser Use-Case-Diagramm stellt dar was ein User / Admin in unserer Applikation in der Lage ist auszuführen.

5 Testing Strategie

5.1 Verwendete Tools

Für die Tests verwenden wir Cypress und Postman, um die generelle Funktionalität der selbst implementierten Endpoints zu testen. Cypress wird dabei zur Automatisierung der End-to-End-Tests verwendet, während Postman hauptsächlich als component Testing verwendet wird, um die restlichen Endpoints zu testen.

Wir entschieden uns das Testen der weiteren Endpoints mit Postman zu machen, da wir alle schon Erfahrungen damit haben und wir somit keine Zeit aufwenden müssen eine neue Testing-Art zu erlernen. In Postman erstellen wir eine Testing-Collection, in der man sieht, wie und welche Endpoints genau getestet wurden. Diese Collection findet man im Repository des Backends.

5.2 Übersicht der getesteten Endpoints

Alle erstellten Endpoints werden getestet. Dies schliesst alle CRUD-Operationen (Create, Read, Update, Delete) und die damit verbundenen Authentifizierungsanforderungen ein. Dabei orientieren wir uns an unserem Use-Case-Diagramm, weil dort beschrieben ist was die verschiedenen Benutzer können und wir so die Enpoints definieren können.

Endpoints die getestet werden (Als User / Admin):

- GET /imagePosts
- GET /imagePosts/imagePostId
- GET /imagePosts /user/userId
- POST /imagePosts/create
- PUT /imagePosts/id
- DELETE /imagePosts/id

Ein besonderer Fokus liegt auf dem Testen der Zugriffsbeschränkungen, um sicherzustellen, dass nur autorisierte Benutzer bestimmte Funktionen ausführen können (z. B. Bearbeitung durch Admins oder den Ersteller eines Image-Posts).

5.3 Erwartetes Verhalten der Endpoints

Der erwartete Output jedes Endpoints muss im Voraus definiert sein, wir entschieden uns auf diese Definition:

Erfolgreiche Operationen sollen den korrekten HTTP-Statuscode (wie 200 für erfolgreiche GET-Anfragen oder 201 für erfolgreich erstellte Ressourcen) und die erwarteten Daten zurückgeben.

Fehlerhafte Operationen, wie ein unbefugter Zugriff oder ungültige Eingabedaten, sollen entsprechende Fehlermeldungen und HTTP-Statuscodes (wie 401 für nicht authentifizierte oder 403 für unbefugte Zugriffe) generieren.

Auch der erfolgreiche und fehlerhafte Zugriff auf Endpoints wird mit mehreren Benutzern und verschiedenen Rollen getestet, um sicherzustellen, dass die Berechtigungen korrekt umgesetzt sind.

5.4 Use-Case Beschreibung für einen detailliert getesteten Endpoint

Use-Case Beschreibung	
Use-Case name:	Create Image-Post
Actor(s):	User, Admin
Description:	Ein Benutzer oder ein Admin sollte in der Lage sein, einen neuen Image-Post zu erstellen. Zuvor muss man allerdings als Benutzer/Admin eingeloggt sein, um einen Post erstellen zu können. Danach sollte man auf einem Button klicken können der den User/Admin auf eine andere Page weiterleitet. Auf dieser Page kann man dann die entsprechenden Daten für das Erstellen eines Posts eintragen und danach auf einen weiteren Button klicken, welcher ein Request ans Backend versendet. Bei einem erfolgreichem Request sollte der Benutzer/Admin den Post auf der Homepage sehen.
Preconditions:	• Benutzer ist eingeloggt. • Benutzer hat gültige Zugangsdaten. • Der Button führt zu einer neuen Page • In der neuen Seite sollte man Daten hineinschreiben können • Die geschriebenen Daten sollten abgespeichert werden • Der neue Post ist auf der Homepage sichtbar
Postconditions:	• User / Admin füllen das Formular aus • Der Post wurde erfolgreich erstellt • Alle Änderungen wurden in der Datenbank gespeichert.
Normal Course:	1. User loggt sich ein. 2. Button klick 3. Formular erscheint 4. Formular ausfüllen 5. Submit Button klicken 6. Daten werden in der Datenbank gespeichert 7. Neuer Post erscheint auf der Homepage
Alternative Course:	• Wenn der User nicht eingeloggt ist, wird eine Pop-up Dialog angezeigt und zur Login-Seite aufgefordert. • Wenn der Benutzer keine Berechtigungen hat, wird der Zugriff verweigert.
Exceptions:	• Keine Berechtigung für den User führt zu einer Fehlermeldung.

6 Swagger Dokumentation

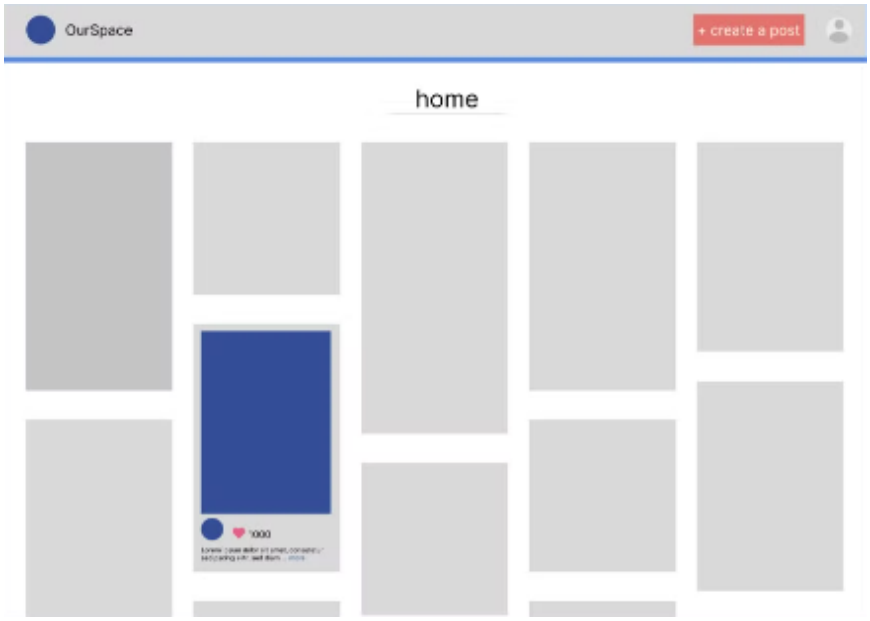
All unsere Endpoints wurden individuell dokumentiert, die Dokumentation ist in der Swagger-UI dargestellt. Um die Dokumentation zu sehen, muss man das Backend des Projektes Lokal aufgesetzt haben und starten.

Die Swagger Dokumentation ist hier zugänglich.

7 Frontend Mockup

Zusätzlich hat unser Team Mockups für das Frontend erstellt. Die Mockups zeigen die Ansichten unserer Homepage mit allen Image-Posts, die Detailansicht eines Image-Posts sowie die Ansicht, aller Image-Posts eines Users.

Homepage:



Alle Image-Posts eines Users:

Detailansicht: