Hochschule **Rosenheim**
University of Applied Sciences

Vorlesung GUI

Übungsblatt 10: Architektur

---

1. Theoriefragen

1. Was versteht man unter einer Schichtenarchitektur? Welchen Vorteil hat sie?
2. Wie kann die Schichtenarchitektur in einer großen Applikation skalieren?
3. Nenne drei Merkmale für ein gutes Schnittstellen-Design und erkläre, was man darunter versteht.
4. Wieso ist es sinnvoll, einen `services` Ordner in seinem Frontend zu haben? Wozu dient er?

---

2. Ziel der Übung

In dieser Übung entwerft und beginnt ihr die Umsetzung einer Vue.js-Frontendarchitektur für eine bestehende REST-API. Ihr analysiert die API, strukturiert euer Projekt in Säulen und Schichten und setzt erste Teile des UI um.
Der Fokus liegt auf Modularisierung, Wiederverwendbarkeit und sauberer Projektstruktur, nicht auf vollständiger Funktionalität.

3. Basis: RealWorld Spring Boot API

Wir verwenden die Java 21 und Spring Boot 3 Implementierung der Real World Example App:

- https://github.com/1chz/realworld-java21-springboot3
- API-Spezifikation: https://1chz.github.io/realworld-java21-springboot3/
- Die API bietet:
    - Registrierung/Login (JWT-basiert)
    - Artikel anzeigen/veröffentlichen/favorisieren
    - Benutzerprofile
    - Kommentare

4. Analyse & Architektur-Design

Analysiert die API:

- Wie ist das Backend strukturiert?
- Was haltet ihr von der Backend Architektur? Was findet ihr gut/schlecht? Würdet ihr etwas anders machen?
- Welche Endpunkte gibt es?
- Welche Datenstrukturen (DTOs) werden erwartet/geliefert?

Entwerft eure Projektstruktur im Frontend:

- Welche Säulen braucht ihr?
- Wie wollt ihr innerhalb der Säulen euren Code unterteilen?
- Welche Komponenten braucht ihr?
- Welche Composables und Services braucht ihr?

5. Implementierung Frontend

Setzt das Backend lokal auf, sieht dazu auch die Readme des Projekts:
https://github.com/1chz/realworld-java21-springboot3?tab=readme-ov-file#getting-started
Hinweis: Ihr benötigt Java 21, damit das Projekt läuft.

Erzeugt eure Paketstruktur aus Aufgabe 4 in einem neuen Vue.js Projekt.

Überlegt euch einen Use Case, den ihr implementieren wollt. Orientiert euch dabei an den Endpunkten, die ihr in Aufgabe 4 analysiert habt.

Zum Beispiel:

- Registrieren eines neuen Benutzers
  - Input Felder für die Eingabe der Daten
  - Backend Aufruf
  - Meldung bei erfolgreicher Registrierung
- Laden aller bestehender Artikel
  - Liste oder Card Darstellung der Artikel basierend auf den Daten aus der Backend Antwort
  - Backend Aufruf
  - Hinweise:
    - Dazu müsst ihr ein valides Token mitschicken! Falls ihr nicht wisst, wie das funktioniert, habe ich euch im Anhang `curl` Befehle angehängt, die ihr verwenden könnt.
    - Zum Start der Applikation habt ihr keine Artikel, ihr müsst für die Anzeige also erstmal welche erzeugen. Falls ihr nicht wisst, wie das funktioniert, habe ich euch im Anhang `curl` Befehle angehängt, die ihr verwenden könnt.

Ihr seid in der Gestaltung der UI völlig frei, in der Übung geht es vor allem um eine saubere Software-Architektur.

6. Optional: Implementiert weitere Use Cases

Wenn ihr früher fertig seid, sucht euch weitere Use Cases und API Endpunkte, die ihr implementieren könnt.

Anhang

Um ein Token zu erzeugen, müsst ihr euch registrieren und einloggen:

Registrieren

```shell
curl -X POST http://localhost:8080/api/users \
  -H "Content-Type: application/json" \
  -d '{
    "user": {
      "username": "maxi",
      "email": "maxi@example.com",
      "password": "securepassword"
    }
  }'
```

Einloggen

```shell
curl -X POST http://localhost:8080/api/users/login \
  -H "Content-Type: application/json" \
  -d '{
    "user": {
      "email": "maxi@example.com",
      "password": "securepassword"
    }
  }'
```

Als Antwort bekommt ihr euer Token:

```shell
"user": {
  ...
  "token": "eyJhbGciOiJIUzI1NiIsInR5..."
}
```

Erzeugen eines neuen Artikels:

```shell
curl -X POST http://localhost:8080/api/articles \
  -H "Content-Type: application/json" \
  -H "Authorization: Token YOUR_JWT_TOKEN_HERE" \
  -d '{
    "article": {
      "title": "My First Article",
      "description": "An intro to realworld backend",
      "body": "This is my article body. It supports **Markdown**!",
      "tagList": ["java", "spring", "realworld"]
    }
  }'
```

Nach demselben Schema könnt ihr auch alle anderen Endpunkte des Backend via `curl` aufrufen, um euch Testdaten zu erzeugen.