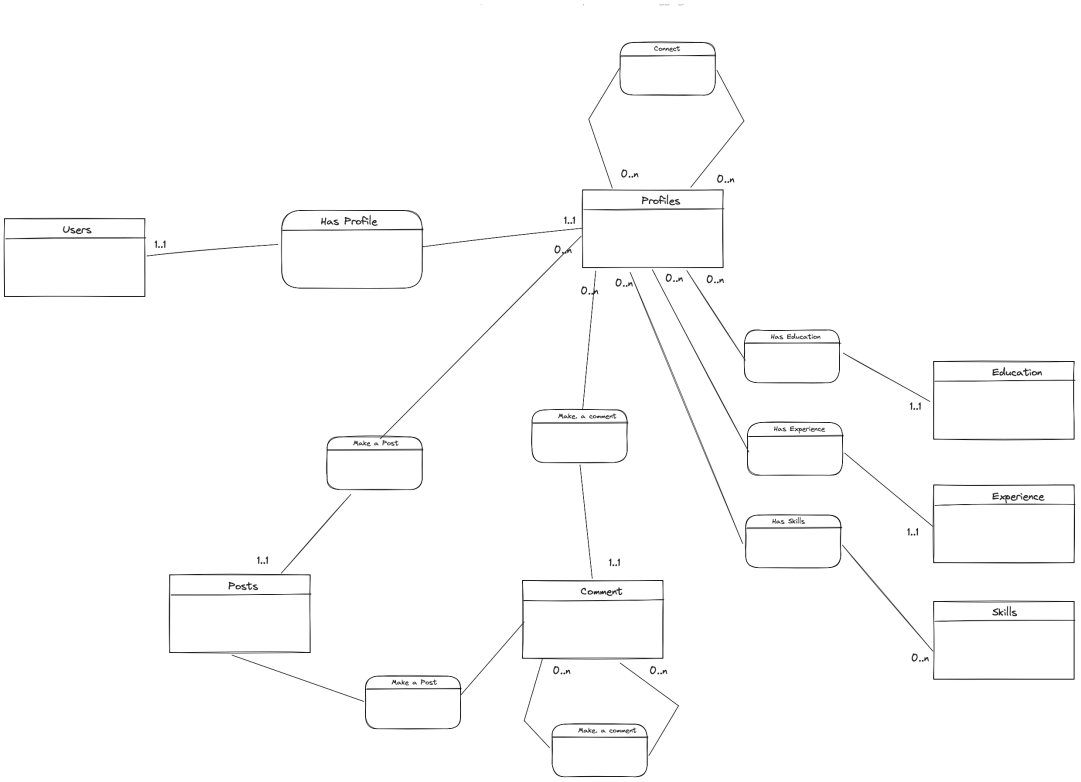
Projet 4AMS



Project for 4AMS - 2023-2024

Project Description

In this project, you are required to develop the backend for a LinkedIn clone utilizing a microservice architecture and Java language with Spring Boot (Java 21 and Spring Boot 3.2). The project requires you to implement microservices, REST APIs, and employ Spring Data JPA and Hibernate for storage. You are provided with a SQL schema and data from a monolith database. You will be segregating the existing SQL schema and test data, inherited from a monolithic application, according to the microservice best practices covered in the 4AMS course.

In your microservices you need to implement business rules. In particular, when there is an integrity constraint in the database you have to verify it in your code first.

For instance, if someone tries to add a Comment on a post that does not exist you have to return a 4xx response with a message in the body describing the problem.

Remember the best practices and apply them diligently. Ensure that each service should have its own database to ensure loose coupling and high cohesion. Moreover, services should be stateless and are designed to be resilient. This will be a challenging but highly rewarding project that will solidify your understanding of microservices using Java and Spring Boot.

Microservices to Develop

User Microservice (4 points)

This microservice will handle the task of managing users and their basic information, including passwords. Endpoints should be created for user registration, authentication, and information retrieval. Verify users are at least 18 years old when they register.

Profile Microservice (3 points)

The Profile microservice is tasked with managing user profiles, which include the user's education details, skills, professional experience, and connections with other users. Endpoints should include those for creating, updating, fetching, and deleting user profiles and connections.

Posts Microservice (4 points)

This will focus on managing posts and comments made by users on the posts. Endpoints should allow creating, updating, deleting and fetching posts and comments.

Backend For Frontend (BFF) Microservice (6 points)

This microservice is aimed at collating and presenting data from the other microservices to the frontend. It will need to make calls to the other microservices and consolidate the data for the frontend team.

Documentation (3 Points)

Document you APIs with OpenAPI / Swagger and provide a report explaining your approach and solution.

Docker (Bonus: 2 points)

Provide a docker compose file for building and deploying all microservices and the database.

Requirements from the Frontend Team

The following are the requirements coming from the frontend team regarding which operations they will need from your application backend.

Home Page API

This will list the posts to be displayed on the main page. Each post must include Author Name, Post Title, and limited content (first N characters), along with post's ID for navigation and user's ID to access the user's profile.

Post Page API

This API will provide full details of a single post, including Author name, Post title, as well as entire content. It should contain the IDs required to navigate to author's profile as well.

The API should also fetch and display the comments on the post, featuring comment content and author name for each comment as well as the date. Only one level of comments should be returned, which means not including replies to these comments.

User API

This API should allow the creation of a new user. Information such as username, email, password should be accepted while ensuring that the email and username are unique. After the user is created, it should return a confirmation along with User ID, for reference in future tasks.

This API should also allow to manage users, modify some of their information and modify their password.

Profile API

This endpoint retrieves user data based on the User ID provided. The profile data should include the user's username, email, and links to user's education, skills, experience, and connections.

This API also allows users to update their information regarding skills, experience and education.

User's Connections API:

This endpoint retrieves a list of connections for a particular user in a structured response, which includes the ID, username, and profile link for each connection.

This API should allow to manage the connections between users (create, read, update, delete).

Search Users API:

This endpoint allows users to search for other users by username or skills. The response should include their profile ID.

Artifacts to provide

- Documentation in the form of OpenAPI/Swagger needs to be provided to the frontend team regarding the API they have available. Ideally you should also provide a documentation for each of you microservices APIs.
- Example requests should be provided in the form of curl commands or http requests files usable in IntelliJ IDEA. Those example requests will exercice all requests available to the frontend.
- The application will have to be deployed in Docker using a docker-compose file. All microservices and the database server will have to be included in the docker-compose so that everything can be built and deployed together in one command.
- All source code will have to be provided for review. It should be buildable with Maven.
- A report should be provided that describe your approach and solution.