Project documentation for the portfolio exam

Tim Knüttel (5123101) Adrian Stelter (5123031)
Laurin Neubert (6824016) Jakob Hämmelmann (5123113)

February 15, 2025

1 Introduction of the Project Topic

1.1 Domain

The domain of our project is a social media platform like X (formerly Twitter), Threads or Bluesky. Our goal for the project is to create a platform that enables users to interact with one another through means of text posts. The main objective here is to give users the opportunity to interact not only by posting their own posts, but also to give feedback through likes and comments. Tags help the users to find posts based on interests or topics.

1.2 Use Cases

1.2.1 Posts

The central and most important use case for our project is the ability for users to post text posts. When posting, users should be able to fill their post with content, also it should be visible which user posted at what exact time.

1.2.2 Replies & Likes

Users should be able to interact with posts directly by replying to and "liking" them. This gives users the ability to converse on the platform and discuss their favorite topics. Additionally, replying keeps all the different posts together, so there can be an easy understanding of which post is a new topic and which post was written in response to an already existing topic.

1.2.3 Tags

Tags can be used to link a topic to the created post. By searching for posts tagged with the desired topic, users can easily find posts that are relevant to what they wish to converse about. This is designed to increase the usability and also user retention.

2 Description of the Software Architecture

3 Explanation of the API technology

For this project, we have chosen a RESTful API architecture with Spring Boot. This decision is based on several factors. REST is a widely adopted industry standard that significantly simplifies integration with other systems. Additionally, its stateless communication enables high scalability, which is particularly beneficial for large-scale applications. Furthermore, REST offers great flexibility, as it is compatible with various frontend technologies and mobile applications. Another crucial advantage is its ease of implementation: Spring Boot provides comprehensive support for REST controllers, authentication, and persistence, making development more efficient.

This technology brings several benefits to our project. REST is simple and lightweight, as it relies on JSON over HTTP, which facilitates both implementation and debugging. The stateless architecture also ensures efficient resource utilization. Moreover, REST integrates seamlessly with Spring Data JPA, enabling a smooth connection with the database.

Despite these advantages, some limitations and trade-offs need to be considered. One of the biggest challenges is the lack of real-time communication. Unlike WebSockets or GraphQL subscriptions, REST does not provide built-in support for real-time updates. Additionally, there is a risk of over-fetching or under-fetching, where clients may receive more or fewer data than necessary, potentially impacting performance—especially with large datasets. Another issue is state management: Since REST is stateless, handling complex transactions across multiple API calls can be challenging.

Despite these limitations, the benefits of a RESTful API with Spring Boot outweigh the drawbacks for our use case, ensuring a robust, flexible, and highly scalable solution for our project.

4 Implementation Details

As a framework we chose Quarkus and therefore REST Easy. Already collecting experience with Quarkus is a benefit for the members of our project. Furthermore, from a future-proofing perspective, Quarkus is ready for the use as a base for micro-services, meaning the application being split into multiple small parts. To implement mapping we went with MapStruct. We choose it because it is a modern approach to mapping as well as being generated instead of manually written, this reduces the possibilities of user error in the implementation. For authentication, we chose to implement a custom middleware that interacts with the THWS Authentication System and takes the Bearer token it generates. This decision relies on the ever increasing popularity of using bigger providers like Google or GitHub as the sole authentication source.

5 Testing Strategy

6 Learning Outcomes and Reflection

Laurin's notes: next time, more time planning before we start coding (esp. when the API-technology is not yet picked); smaller goals that can be achieved quicker so we can support earlier in the process if someone needs help; utilising Git to it's full potential (namely creating milestones and issues;)

Caption

This article was drafted and refined using GPT-4 based on an outline containing related information. The GPT-4 output was reviewed, revised, and enhanced with additional content. It was then edited for improved readability and active tense, partially using Grammarly.