Reflection on PG3402 Microservices Exam

Project Overview

This project focused on building a food shopping list application that tracks items, provides nutritional data, and recommends healthier options. It integrated the Mattilsynet API for nutritional information, RabbitMQ for messaging, and Consul for service discovery and configuration, all within a microservices-based architecture.

Key Decisions and Technologies

To simplify the system and better support households, I decided to remove the user service. Instead of managing individual accounts, the application now supports multiple shopping carts, making it more flexible for shared use. The backend was developed with Java and Spring Boot, ensuring scalability and reliability. RabbitMQ handled asynchronous communication between services, decoupling their interactions. Consul was used to centralize configuration and manage service discovery. On the frontend, React provided a dynamic and responsive user experience, while Docker ensured consistent deployment across different environments.

Challenges and Insights

While microservices enabled scalability, they also introduced complexity, particularly in managing service communication and maintaining data consistency. RabbitMQ made messaging efficient but required careful attention to message delivery and processing to preserve system integrity. Consul simplified configuration management but came with a learning curve and additional setup. Removing the user service not only reduced system complexity but also aligned the design with real-world usage, showing how simplifying architecture can lead to better outcomes.

Conclusion

This project offered valuable lessons on balancing the benefits and challenges of microservices and messaging systems. Simplifying the design by focusing on multiple shopping carts instead of user accounts proved to be an effective decision, keeping the system aligned with its purpose and improving usability for households.