PGR209
Exam
Report

The goal of this project was to develop a complete web application for a fictional company, CandyCo, that manages customers, orders, products, and addresses. The application includes a backend built with Spring Boot and a frontend using Thymeleaf for data display. The database is PostgreSQL, managed via JPA and Flyway. This project provided an opportunity to explore various technologies and concepts, such as REST APIs, unit and integration testing, and frontend development.

The project began by setting up a basic Spring Boot application with necessary dependencies, such as JPA, Thymeleaf, and Testcontainers. We started by defining the data models for Customer, Order, Product, and CustomerAddress. Then, we implemented REST APIs for each entity, focusing on basic CRUD operations. After establishing backend functionality, we used Thymeleaf to build frontend pages for displaying customers and orders.
We implemented an initData service to populate the database with test data, which made testing functionality easier. Testing was an important part of the project, and we wrote both unit and integration tests to ensure the APIs functioned as expected. Finally, we used Postman to manually test all endpoints and documented them for future reference.

The separation between backend and frontend made it easier to structure the project. Spring Boot proved to be a powerful platform for quickly developing APIs, and Thymeleaf integrated seamlessly with the backend for data presentation. Testcontainers made it easy to test against a real PostgreSQL database without risking production data. Using Postman to test API endpoints was also very helpful, as it allowed us to identify and fix issues quickly. Additionally, the initDataservice was effective in generating test data, providing a simple way to verify that the database models and relationships worked as expected.

One challenge we encountered was ensuring data consistency in the database, especially in the relationship between Customer and CustomerAddress. We noticed that not all customers had addresses and had to adjust
the initData service to fix this. Another challenge was handling specific error

responses for missing resources, which required implementing custom exceptions and a common RestExceptionHandler.
Additionally, we struggled with formatting the frontend using Thymeleaf and CSS. After several iterations and testing with different design options, we were able to produce a final result that was both visually appealing and functional.
Testing and debugging in Postman was also time-consuming, but it helped us better understand how clients and servers interact.
We decided to use Java Faker to make fake data for our project. This turned out to be a bit of a struggle, due to the fact that it produced more numbers and letters than what our database allowed. We had to make several adjustements to our database with migration files, and change the code on our Faker program, to make sure it only produced numbers and letters accordingly to our database model´s limits.

For this project we used code written during the semester, and the website Stackoverflow to search for clues when debugging our program.

This project was a valuable learning experience, strengthening our skills in Spring Boot, Thymeleaf, database management, and testing. While we faced some challenges, we were able to resolve them and deliver a functional application. If we were to do something differently, we would spend more time on planning and designing the frontend from the beginning.
Overall, we are satisfied with the result and feel that the project demonstrates a solid understanding of full-stack development.