PGR209
Fall 2024

Start Date: 18/11/2024, 10.00
Due date: 09/12/2024, 10.00

Final Exam

Requirements

The PGR209 exam is a two-person exam. Students are responsible for finding their own partners.

The exam shall be delivered as a zip file, containing all source code and reflections. You MUST run “mvn clean” before delivery, and you MUST NOT include the /target folder in your zip file.

Your exam consists of the following:

You are to build a backend application to support an e-commerce company called Chicken Direct, which sells chickens directly to the customer. You are to use Java, Spring Boot, Spring Data JPA, Flyway, and TestContainers to build this application. Use of JavaFaker or Lombok is optional.

As a database, you must use Postgres, with Flyway to handle migrations. Use the attached docker-compose.yml file, or create your own.

The application should handle the following functionality:

Chicken Direct should be able to manage all of the Customers, Products, Orders, Customer Address via http requests. This includes creating, deleting, and fetching. All of these should be handled via Spring Data JPA, and should be defined as entity classes.

Products should have a name, description, price, status, and quantity on hand. Chicken Direct should be able to manage all of this via http requests.

Customers should have a name, phone number, email, and a way to see their order history. Additionally there should be a Customer Address that is stored in a separate table and uses a separate class. A Customer can have multiple Addresses.

Orders should have a list of Products, how many of each product, a shipping charge for the order, the total price of the order, whether it has shipped, the customer, and a shipping address.

Fetching a customer should show their addresses and order history.
Fetching a customer should show the customer and shipping address.
Placing an order should update the status and quantity on hand of a product, and the system should not allow products to be ordered that are out of stock.

All of this functionality should be available via http requests.

The system should return custom exceptions for requests, such as customer not found, product not found, etc.

The application should include both unit tests and integration tests using test containers, and should use Jacoco to generate test coverage reports.

The application should use repositories, services, and controllers. Controllers should never talk to repos, and services should not talk to repos other than their own.

Students should include a report, no more than 500 words, reflecting on their experiences with the exam. What worked, what didn't work, what resources did you use. One report per group, not per student.

All resources are available for this exam, including websites like stack overflow, or code written during the semester. Copying another group's work is NOT allowed, nor is asking other groups for solutions.

Grading criteria:

E - All application functionality is working, but there is no database.

D - All application functionality is working with JPA using a postgres database

C - All application functionality is working with JPA, and the application includes custom exceptions, and includes at least 30% test coverage

B - All application functionality is working with JPA, and the application includes custom exceptions, and includes at least 50% test coverage, including both integration tests and unit tests, using testcontainers.

A - All application functionality is working with JPA, and the application includes custom exceptions, and includes at least 70% test coverage, including both integration tests and unit tests, using testcontainers, and the application should include logging

Credit will be given for adding additional functionality, such as generating front end content or order receipts via Thymeleaf, providing postman requests, implementing pagination, or implementing custom queries.

Attachment: docker-compose.yml

```
services:
  db:
    container_name: "postgresql"
    image: postgres:alpine
    ports:
      - '5432:5432'
    environment:
      - POSTGRES_DB=appdb
      - POSTGRES_USER=appuser
      - POSTGRES_PASSWORD=pirate
```