PG5100
Enterprise programmering 1
Final exam - Spring 2022

Assignment

The final exam for this course is a 48-hour home exam. You will be working individually, this is not a group exam.

The assignment is to create a working API for an animal shelter. You will not be building the frontend, just the API. You will build the server, according to the following requirements:

You must write the server in Kotlin.

You must use Spring Boot to build the API, using Rest Controllers.

You must use Spring Data JPA to persist data, using Entity classes and Repositories. You should be using Flyway to manage creating initial structures, and not letting JPA manage the DDL. You will be persisting data to a Postgres database inside a Docker container.

You must secure the application using Spring Security. You will handle login via a username and password sent in the body of a POST request, which, once authenticated, will return a JWT token inside a cookie to the user. All other requests should use this token for authorisation. You will need to set different authority levels (Admin, User, etc), and set which authorities have access to which endpoints.

You must implement integration testing using Spring's testing tools, TestContainers, as well as MockK if you need to mock anything.

Specifications

The server is for managing data for an animal shelter, that would be used for API calls by a frontend (you are not implementing the frontend).

The API should provide the following functionality:
- Retrieve all animals
- Retrieve a single animal
- Add a new animal
- Update an animal
- Delete an animal

The Animal class should include relevant fields, such as Name, Animal Type, Breed, Age, Health, etc. I leave the details to your imagination.

Additionally, the application should handle authorization and authentication.

The base url for the API should be /api
- /api/shelter for endpoints specific to managing the animal shelter
- /api/user and /api/authentication should be the base urls for authentication and authorization

You will use the docker-compose.yml, application.yml, and application-test.yml provided at the end of this document.

Delivery

You will deliver the project in Wiseflow as a single zip file. This zip file should contain:

- The source code (DO NOT INCLUDE THE TARGET DIRECTORY)
- Documentation
  - All endpoints, and how to use them along with important details, like how I access the endpoints via Postman
  - Information on how to log in
  - An overview, no more than 500 words, give details on anything you weren't able to implement or get working

Grading

Minimum to get an E:

A working API that handles all of the shelter functionality, using the appropriate HTTP methods (GET to retrieve, POST to create, PUT to update, DELETE to delete), written in Kotlin.

Minimum to get an D:

Full data persistence, using Spring Data JPA. Data migrations using Flyway.

Minimum to get an C:

Fully secure the application using Spring Security.

Minimum to get an B:

Full implementation of Unit Tests, using mocks with MockK

Minimum to get an A:

Full implementation of Integration Tests, including full end-to-end testing.

application-test.yml (for testing)

```
spring:
  jpa:
    database: POSTGRESQL
    show-sql: true
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQLDialect
  datasource:
    url: jdbc:tc:postgresql:14.2-alpine:///localdevdb
    username: localdevuser
    password: pirate
    driverClassName: org.testcontainers.jdbc.ContainerDatabaseDriver
```

application.yml

```
spring:
  jpa:
    database: POSTGRESQL
    show-sql: true
    hibernate:
      ddl-auto: none
    properties:
      hibernate:
        dialect: org.hibernate.dialect.PostgreSQLDialect
  datasource:
    url: jdbc:postgresql://localhost:5432/localdevdb
    username: localdevuser
    password: pirate
    driverClassName: org.postgresql.Driver
  sql:
    init:
      platform: localdevdb
```

docker-compose.yml

```
version: '3.1'
services:
  postgres-db:
    container_name: 'postgres-db'
    image: postgres:alpine
    ports:
      - '5432:5432'
    environment:
      - POSTGRES_DB=localdevdb
      - POSTGRES_USER=localdevuser
      - POSTGRES_PASSWORD=pirate
```