CEN3024C Module 4 Project

For this project we will add Spring Data JPA-based repository classes with associated service, controller, and DTO classes in order to implement some basic APIs, using the springdoc-openapi tool to generate the API documentation.

Clone the GitHub Classroom assignment repo and use the following commands to create the postgresql database:

(note: if you already have a pizzarestaurant database in your postgresql installation, you should drop it before continuing. You may also want to back it up first using the pg_dump command described below)

```
C:\> c:\"program files"\postgresql\16\bin\psql -U postgres
...
postgres=# create database pizzarestaurant;
CREATE DATABASE
postgres=#\q

C:\> c:\"program files"\postgresql\16\bin\psql -U postgres
pizzarestaurant <db\m4pizzarestaurant.sql
```

The last command will restore the module 4 version of the pizza restaurant database (including the data) from the m4pizzarestaurant.sql file.

Here is the pg_dump command to back up a postgresql database:

```
C:\> c:\"program files"\postgresql\16\bin\pg_dump -U postgres dbname
>db\outputfilename.sql
```

After loading the database, create the connection to the data source in IntelliJ using the “+” button in the Database tool. Once you have done that you should be able to expand the database to view the schema and contents.

Examine the source code for the project and note the revised entity and repository classes from module 3. Note also the new service, controller, and DTO classes.

The program executes queries via the “testDatabaseRepositories” method in the main application class via a CommandLineRunner. This method has been modified as follows:

Various methods are called to test the new repository methods using the customerRepo, pizzaRepo, and pizzaOrderRepo parameters. Note that for the PizzaOrder repo, to access the "joined" data of the M:N relationship, we can just use an appropriate method name and JPA's query derivation feature will do the rest of the work for us, e.g.

```
System.out.println("FINDBYPIZZAPIZZATYPE");
pizzaOrderRepo.findByPizzaPizzaType("pepperoni").
```

There are 3 method calls that are annotated in red below, you will need to complete them for this assignment. The upper case println statements are there to help you locate your data in the output log, if you set your mouse focus to the output window pane you can use Ctrl-F to search for that text.

```
@Bean
public CommandLineRunner testDatabaseRepositories(
            CustomerRepository customerRepo,
            PizzaRepository pizzaRepo,
            PizzaOrderRepository pizzaOrderRepo) {
    return args -> {
        System.out.println("FINDALLCUSTOMERS");
        customerRepo.findAll().forEach(customer ->
                System.out.println(customer.toString()));
...

        System.out.println("FINDALLPIZZAORDERS");
        pizzaOrderRepo.findAll().forEach(order ->
                System.out.println(order.toString()));
        System.out.println("FINDBYPIZZAPIZZATYPE");
        pizzaOrderRepo.findByPizzaPizzaType("pepperoni").
                forEach(pizza -> System.out.println(pizza.toString()));
        System.out.println("FINDBYPIZZAPIZZATYPEIGNORECASE");
        pizzaOrderRepo.findByPizzaPizzaTypeIgnoreCase("PEPPERONI")
                .forEach(pizza -> System.out.println(pizza.toString()));

        System.out.println("FINDBYPIZZAPIZZASIZE");
        // add your method call here
        System.out.println("FINDBYPIZZAPIZZACRUSTTYPE");
        // add your method call here
        System.out.println("FINDBYCUSTOMERCUSTOMERLASTNAME");
        // add your method call here

        System.out.println("DONE");
    };
}
```

After completing this task, read through the Customer classes - CustomerController, CustomerService, CustomerConverter, and Customer DTO to familiarize yourself with the creation of the API endpoints. You will need to add appropriate code to the service, controllers, and DTOs for the Pizza and PizzaOrder classes to add two additional API endpoints:

- findbypizzatype in the Pizza classes.   (this one is easy)
- findbycustomername in the PizzaOrder classes. (this one is a little tougher).

Your DTOs should include all of the associated entity data _except_ the primary key value (e.g. the id field for the Pizza class and for the PizzaOrder class). Your API return data should also exclude these fields.

Find the comments marked with the placeholder "Insert your code here" to see where your code needs to be added.

Complete your coding and submit your repo to the GitHub classroom repo. Also submit screenshots showing your openapi docs data (the raw JSON) and the second showing the swagger UI where your new API endpoints are displayed (and subsequently tested, of course).