Projet Développement JAVA V 2023-2024



Une image contenant fleur

Description générée automatiquement avec une confiance moyenne

Produced by Popadiuc Claudiu

Supervise by Mr Bilal Ben Abdelkader

**Architecture Overview**

The One-Piece API is a Spring Boot application designed to manage information about characters in the One-Piece universe. It uses a PostgreSQL database to store character data and provides various endpoints to perform CRUD operations on characters. Additionally, it includes features such as retrieving characters by crew, job, rank, and initiating fights between characters.

I'm going to explain the project for the character model because then it's the same principle for the other controllers.

**Controller** (be.helb.cpopadiuc.controller):

Responsible for handling HTTP requests and interacting with the service layer.

Implements the CRUD operations and additional functionalities like initiating fights.

**Model** (be.helb.cpopadiuc.model):

Defines the data structure of a Character, including its attributes and relationships with other entities like DevilFruit, Crew, Haki, and FightTactics.

**Repository** (be.helb.cpopadiuc.repository):

Uses Spring Data JPA to provide CRUD operations for the Character entity.

Includes custom query methods to retrieve characters based on different criteria.

**Service** (be.helb.cpopadiuc.service):

Contains business logic for character-related operations.

Interfaces between the controller and repository, handling transactions and additional processing.

**Database Configuration (application.properties):**

Configures the database connection properties, such as URL, username, and password.

Defines Hibernate DDL auto-configuration for schema updates.

**Technical Choices**

**Spring Boot:**

Chosen for its simplicity, convention over configuration, and built-in support for web applications.

**Spring Data JPA:**

Simplifies database operations and reduces boilerplate code.

**PostgreSQL Database:**

Selected for its relational database capabilities and compatibility with Spring Data JPA.

**Lombok:**

Used to reduce boilerplate code by automatically generating getters, setters, and other common methods.

**RestTemplate:**

Employed for making HTTP requests to an external fight API to initiate character fights.

**Front End**

**Cross-Origin Resource Sharing (CORS):**

Configured to allow requests from <http://localhost:3000> For the front end.

To launch the front end Instal react if you don't have it then you have a front-end folder, open it in the terminal and run the npm install command, then run the npm start command and the react page will launch automatically, and you'll see my beautiful front end.

Une image contenant fiction, Jeu PC, Art numérique, Visage humain

Description générée automatiquement

Une image contenant texte, Visage humain, personne, habits

Description générée automatiquement

**How to Use the API**

I'm going to explain the project for the character controller because then it's the same principle for the other controllers.

**Retrieve All Characters:**

Endpoint: http://localhost:8080/api/characters

Returns a list of all characters in the database.

**Add a New Character:**

Endpoint http://localhost:8080/api/characters/add

Body:

{

    "name": "Monkey D. Garp",

    "rank": "Vice Admiral",

    "job": "Marine",

    "bounty": 11,

    "imageUrl":"https://pm1.aminoapps.com/6952/a8f89be4e601acd4f9b32043801d6e93d8a40884r1-1280-720v2\_hq.jpg"

}

**Delete a Character by ID:**

Endpoint: http://localhost:8080/api/characters/2

Deletes the character with the specified ID.

**Retrieve Characters by Crew ID:**

Endpoint: http://localhost:8080/api/characters/byCrew/1

Returns characters belonging to the specified crew.

**Retrieve Characters by Job:**

Endpoint: http://localhost:8080/api/characters/byJob/Captain

Returns characters with the specified job.

**Retrieve Characters by Rank:**

Endpoint: http://localhost:8080/api/characters/byRank/yonko

Returns characters with the specified rank.

**Retrieve Characters with High Bounty and No Devil Fruit:**

Endpoint: http://localhost:8080/api/characters/highBountyAndNoDevilFruit

Returns characters with a bounty greater than 1,000,000,000 and no Devil Fruit.

**External API Integration** (CharacterController and RestTemplate):

Uses RestTemplate to interact with an external fight API for initiating character fights.

To test my second api you must start the 2 projects, the 1st main one called "back end" and the second called "secondaryAPi", start the 2 at the same time and to test the api you must test like this:

http://localhost:8080/api/characters/fight/Luffy/Zoro

**Database Connection:**

URL: jdbc:postgresql://localhost:5432/RestApiOnePiece

Username: claudiu

Password: test

**Tests**

For the tests you have a "test" package, go to the "CharacterControllerIntegrationTest" file and you can simply start the file and all the tests will be carried out on the db, for the deletion you will need to add a real character id to delete a character, it already does this on the first run but on the second run you will need to add another id.