Singidunum University

Faculty of Technical Sciences

Department of Software and Data Engineering

Title of the essay:

Final Exam Assignment – React.js app with spring boot

Subject:

Internet technology and web services

Professor:

Prof.Nebojša Bačanin Džakula

Student:

Aleksandar Milosevic

Belgrade, 2023

Welcome to the documentation for the Game Catalog website. This project aims to provide a platform for managing game-related information such as developers, platforms, and games. The website allows users to add, search, and browse through the catalog of games. The technologies used for building this project are Spring Boot for the back-end and ReactJS for the front-end. The data is stored in a MySQL database.

The main purpose of the Game Catalog website is to offer a user-friendly interface for users to explore and keep track of their favorite games. Users can discover new games, view details about specific games, and contribute by adding developers, platforms, or games. The use of Spring Boot and ReactJS provides a robust and efficient solution for handling the back-end logic and creating an interactive user interface.

This documentation will cover various aspects of the Game Catalog website, including its architecture, installation, setup , and more. So, let's get started and delve into the details of this project.

Front-End (ReactJS): The front-end component is responsible for presenting the user interface and interacting with the back-end API. It utilizes ReactJS to build dynamic and responsive web pages. Users can perform various actions such as adding developers, platforms, and games, as well as searching for games. The front-end component communicates with the back-end API to fetch and manipulate data.

Back-End (Spring Boot): The back-end component is developed using Spring Boot, which provides a robust framework for building RESTful APIs. It handles the business logic and data management for the Game Catalog application. The back-end interacts with a MySQL database to store and retrieve game-related information, including developers, platforms, and games.

Components:

Developer Controller:

Provides RESTful endpoints for CRUD operations related to developers.

Accepts HTTP requests from the front-end and communicates with the DeveloperRepository to perform database operations.

Game Controller:

Manages game-related operations and interactions with the front-end.

Handles multipart form data for game creation and updates.

Stores game images in the specified directory and maintains the image URLs.

Communicates with the GameRepository and PlatformRepository to interact with the database.

Platform Controller:

Handles RESTful endpoints for managing platforms.

Performs CRUD operations for platforms by communicating with the PlatformRepository.

Developer, Game, and Platform Models:

Represent the data structures for developers, games, and platforms, respectively.

Define the relationships between the models (e.g., a game belongs to a developer, a game can have multiple platforms).

Database (MySQL): The MySQL database is utilized to store the game-related data, including developers, games, and platforms. The data is persisted and retrieved using Spring Data JPA through the respective repositories (DeveloperRepository, GameRepository, PlatformRepository).

Overall Flow:

The user interacts with the front-end interface built with ReactJS.

The front-end sends HTTP requests to the appropriate endpoints in the back-end API.

The back-end controllers receive the requests and perform the necessary operations.

The controllers communicate with the corresponding repositories to store or retrieve data from the MySQL database.

The controllers return the appropriate responses to the front-end, containing the requested data or confirmation messages.

The front-end updates the user interface based on the received responses.

Prerequisites:

Install Node.js: Visit the official Node.js website (https://nodejs.org) and download the latest stable version for your operating system. Follow the installation instructions.

Install XAMPP: Download XAMPP from the Apache Friends website (https://www.apachefriends.org/index.html) and choose the appropriate version for your operating system. Install XAMPP and start the Apache and MySQL services.

Install Java Development Kit (JDK): Download and install the latest version of JDK from the Oracle website (https://www.oracle.com/java/technologies/javase-jdk11-downloads.html) following the provided instructions.

Front-End Setup:

Open a command prompt or terminal and navigate to your project's root directory.

Run the command npm install to install the project dependencies listed in the package.json file.

Once the dependencies are installed, run the command npm start to start the development server.

Open a web browser and access http://localhost:3000 to verify that the React application is running correctly.

Back-End Setup:

Download and install an Integrated Development Environment (IDE) such as IntelliJ IDEA or Eclipse.

Import your Spring Boot project into the IDE.

Make sure the necessary dependencies are specified in your project's pom.xml file. For MySQL connectivity, you should have the mysql-connector-java dependency included.

Configure the database connection properties in the application.properties file, specifying the MySQL database URL, username, and password.

In my case database name is: game_catalog username:root and password empty

Database Setup:

Open the XAMPP Control Panel and start the MySQL service if it's not already running.

Launch a web browser and access http://localhost/phpmyadmin to open the phpMyAdmin interface.

If required, import any SQL scripts or execute database migration scripts to set up the necessary tables and data.

Running the Project:

In your IDE, run the Spring Boot application. It will start the back-end server on http://localhost:8080.

With the React development server already running, you can access the application in the web browser at http://localhost:3000.

Test the functionality of your game project, ensuring that the front-end communicates correctly with the back-end APIs.

Component Relationships:

Home.js uses the SearchBar.js component and renders a list of games.

SearchBar.js communicates with the API to fetch and filter games.

Home.js, ViewGame.js, and EditGame.js navigate to other components based on user actions using the useNavigate hook from react-router-dom.

EditGame.js and ViewGame.js retrieve game details from the API based on the gameId parameter.

AddGame.js interacts with the API to add a new game.

Data Models:

Platform: Represents a gaming platform and includes an id, name, and a list of associated games.

Game: Represents a game and includes attributes such as id, title, description, releaseDate, genre, imageUrl, developer, and a list of associated platforms.

Developer: Represents a game developer and includes attributes such as id, and name. It also has a list of associated games.

Controller Classes:

PlatformController: Handles requests related to platforms and defines RESTful APIs for creating, retrieving, updating, and deleting platforms. It communicates with the PlatformRepository.

GameController: Handles requests related to games and defines RESTful APIs for creating, retrieving, updating, and deleting games. It communicates with the GameRepository and PlatformRepository. It also includes methods for handling image file upload and retrieval.

DeveloperController: Handles requests related to developers and defines RESTful APIs for creating, retrieving, updating, and deleting developers. It communicates with the DeveloperRepository.

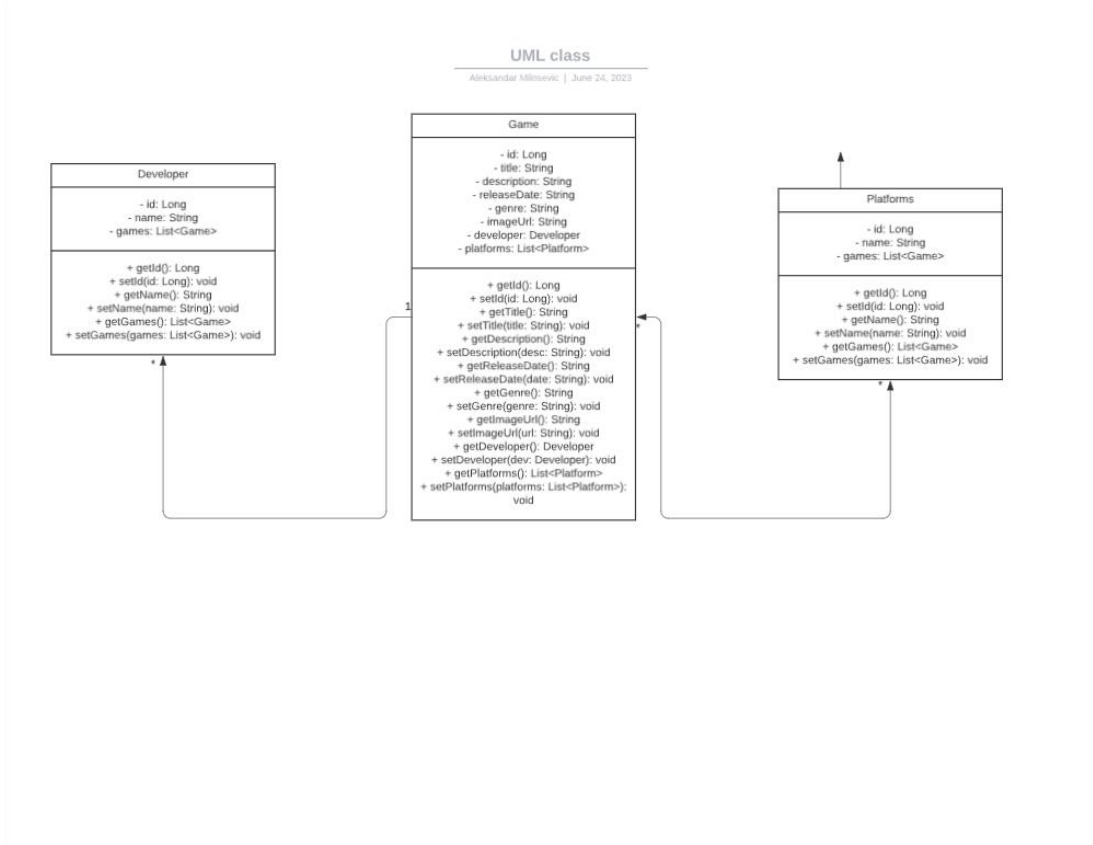
Repositories:

PlatformRepository: Provides methods for CRUD operations on the Platform entity.

GameRepository: Provides methods for CRUD operations on the Game entity.

DeveloperRepository: Provides methods for CRUD operations on the Developer entity.

Exception Handling:



The code includes custom exception classes (PlatformNotFoundException, GameNotFoundException, DeveloperNotFoundException) for handling exceptions when a resource is not found.

Sample Requests and Responses:

The code includes sample request mappings for various endpoints, such as /platform, /game, and /developer, with corresponding request methods (POST, GET, PUT, DELETE).

It also includes methods for retrieving all platforms/games/developers, retrieving a single platform/game/developer by ID, and updating existing platforms/games/developers.

The GameController includes methods for handling image file upload and retrieval.

Project Summary: The Game Catalog project is a Spring Boot-based web application that provides a backend implementation for managing a catalog of games, developers, and platforms. It exposes RESTful APIs to perform CRUD operations on games, developers, and platforms, and allows for the creation and retrieval of images associated with games.

The project utilizes Spring Boot, JPA (Java Persistence API), and Hibernate for data persistence and database interactions. It includes three main entities: Game, Developer, and Platform, with appropriate relationships defined between them. The Game entity contains attributes such as title, description, release date, genre, and an image URL. It is associated with a Developer entity and can have multiple Platform entities.

The backend functionality includes API endpoints for creating, retrieving, updating, and deleting games, developers, and platforms. Sample requests and responses are provided to demonstrate the usage of each API.

Future Improvements: Here are some potential future improvements or enhancements that can be considered for the game:

User Authentication and Authorization: Implement user authentication and authorization mechanisms to secure the API endpoints and restrict access based on user roles and permissions.

User Reviews and Ratings: Add functionality for users to leave reviews and ratings for games. This can enhance the user experience and provide valuable feedback for other users.

Game Recommendations: Incorporate recommendation algorithms to suggest games to users based on their preferences, browsing history, or ratings.

Social Features: Integrate social features such as user profiles, friend lists, and the ability to share game recommendations or achievements with friends.

Game Statistics and Analytics: Collect and analyze game usage data to generate insights and statistics, such as the most popular games, average playtime, or user demographics. This can help in making data-driven decisions and improving the game catalog.

Improved Image Handling: Enhance the image handling functionality by implementing image resizing, cropping, or thumbnail generation to optimize image storage and delivery.

Mobile Application Development: Develop a mobile application to complement the web-based platform, allowing users to access the game catalog on their mobile devices.