Department of Electronical Engineering,
Telecommunications and Computers

Project Report (Phase 2)

Group 6:

47718: Pedro Diz (a47718@alunos.isel.pt)

48259: Vasco Branco (a48259@alunos.isel.pt)

48264: João Pereira (a48264@alunos.isel.pt)

Professors: Paulo Pereira and Pedro Pereira
2023/2024 Summer Semester

Index

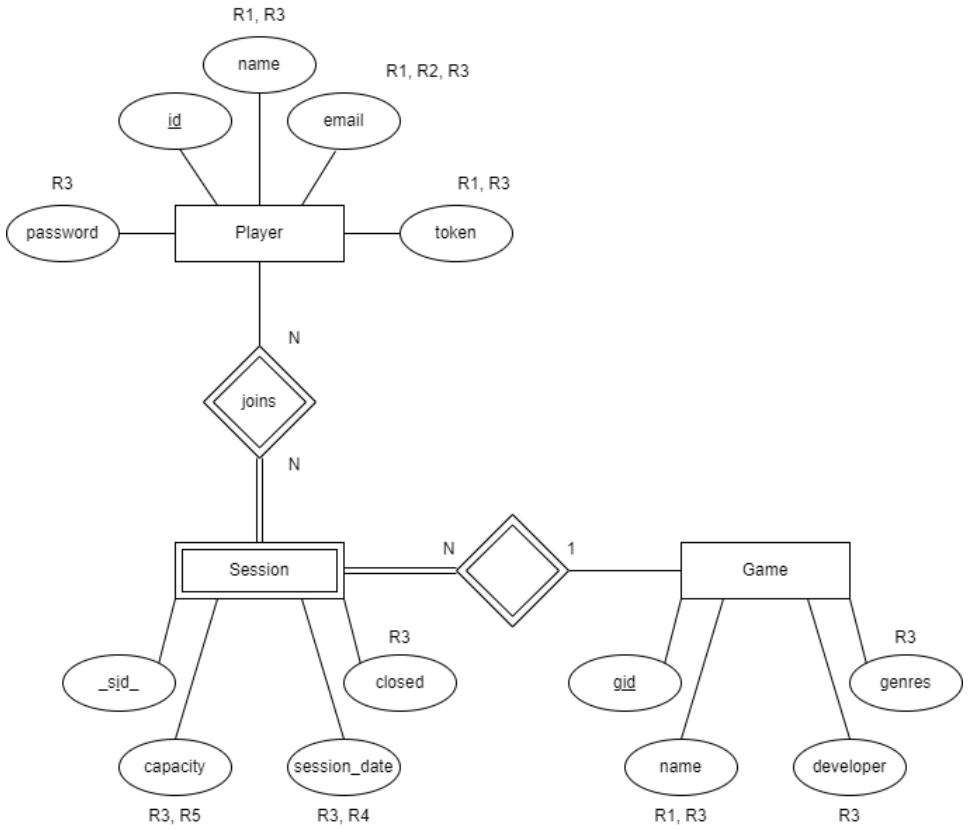
Introduction

The aim of this work is to implement an information system that manages video game sessions.

The application domain consists of these entities:

- **Player**: A **Player** is characterized by having a **unique number**, a **name**, and an **email address**.
- **Game**: A **Game** is characterized by having a **unique number**, a **unique name**, a **developer**, and a **set of associated genres**.
- **Session**: A **Session** is characterized by having a **unique number**, the **number of players involved in the session**, the **session start date**, the **game**, and the **associated players**.

Conceptual Model



R1 - Unique;
R2 - Email format (abcd@aaa);
R3 - Not null;
R4 - Date format (YYYY-MM-DD HH:MM:SS);
R5 - Value greater than 1 (> 1).

Figure 1 – Conceptual Model

The **model** contains the **three entities** mentioned. Note that **Session** is a weak entity from **Player** and **Game** because, if there are no players or game, there is no session.

Integrity Restrictions:

- ‘Email’ has the following format: abcd@aaa
- ‘Email’ and ‘Name’ are ‘Player’ candidate keys
- ‘Token’ is an unique attribute
- ‘Session_date’ has the following format: YYYY-MM-DD HH:MM:SS
- ‘Name’ is ‘Game’ candidate key

Physical model

The physical model can be found here.

All the tables are based on the conceptual model.

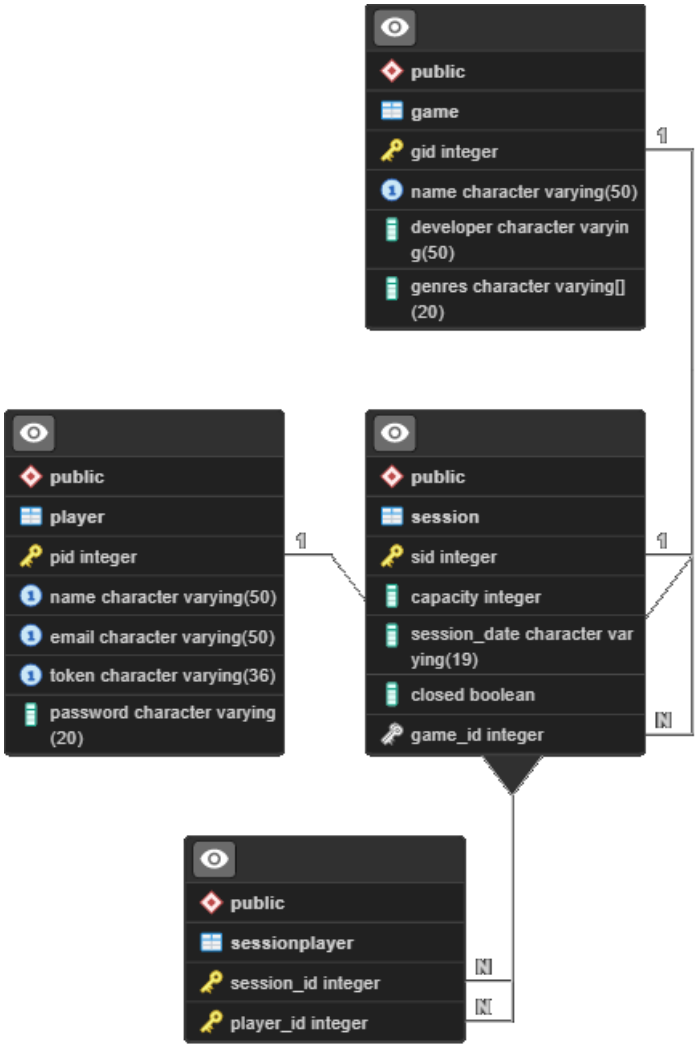


Figure 2 – Physical model

Open-API specification

The Open-API specification can be found here.

All the API's routes were documented using this specification.

Details of an API request

The request reaches the server, and the server then routes the request to the appropriate handler.

After this happens, the following steps are carried out:

- The handler executes a function called errorAwareScope that executes the code needed to fulfill the request. In the event of an error, this function handles the error via the exceptionHandler function.
- Within the function mentioned above, the first thing to do is extract the parameters present in the URI. The token is also extracted if a request requires authentication.
- Next, the JSON is deserialized if the request has a body.
- The associated service is called, which validates the parameters passed on. Within this service, the repository is called to persist, read, and change data.
- The result from the service is then encapsulated in an outputModel class, that class is serialized to JSON and the response is attributed the according status code.

Connection Management

Whenever we interact with the DBMS, we fetch a connection via the getConnection() function and utilize the use{} function, which automatically closes the connection after running the code in its scope. As we never set autoCommit to false in the repository functions, all the code runs in a single transaction.

Data Access

Our data access has been implemented with three interfaces. There is one interface for managing players, another for managing games, and another for managing sessions. There are two implementations of these interfaces. One of them is for performing CRUD operations in memory, whose names normally act in accordance with the following structure: Mem{entity} Repo, and the other is for performing CRUD operations on a Postgres database, where the names of the repositories follow the structure: Jdbc{Entity}Repo. The JDBC repositories receive as a parameter the PostgreSQL 'datasource' with which they would interact.

Error Handling

Backend exceptions are generated in all the main modules. For example:

- Repository cannot find a session with id = x.
- The Service fails validation.
- Api cannot extract the route parameter.

All these exceptions are then handled by the exceptionHandler used in the exceptionAwareScope{} function. Within this exceptionHandler function, the exception is converted to a status code using an associative map. A response is then created with a description of the problem. If an exception cannot be associated with an HTTP status code, it is converted to Internal Server Error status code.

Frontend

A Single Page Application (SPA) was developed, which loads the HTML content as the user navigates through the application. The structure of said SPA is the following:

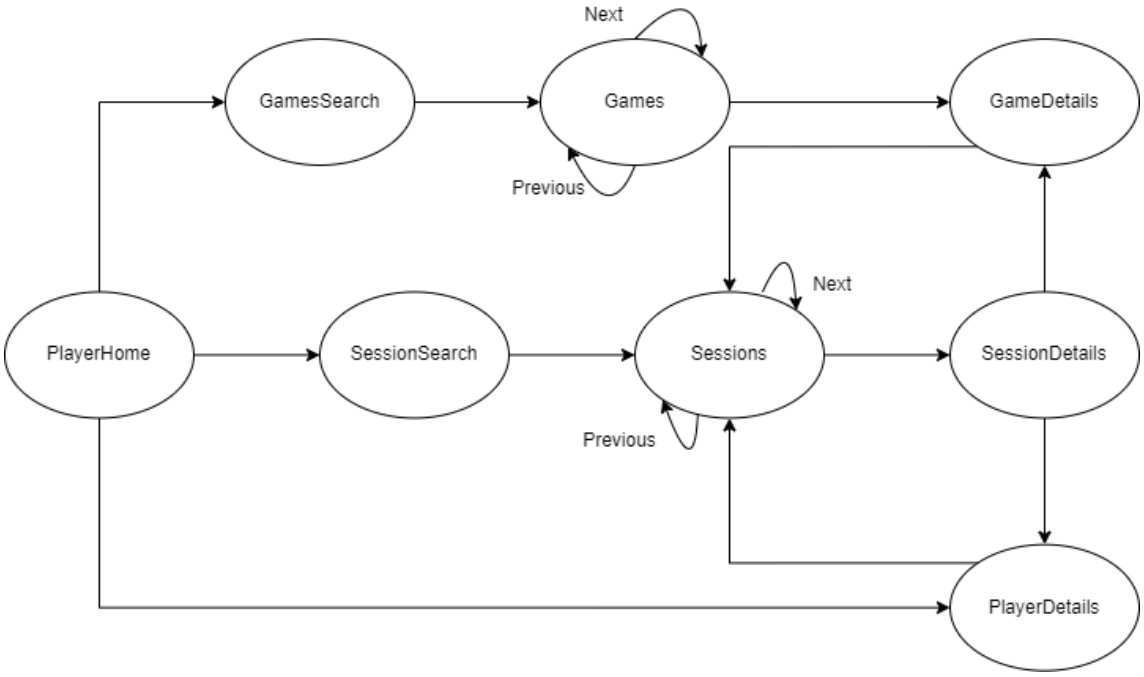


Figure 3 – SPA

Note that every element of navigation except PlayerHome has a direct navigation to PlayerHome.

Critical evaluation

A better analysis of using big transactions in the JDBC repo classes, specifically if there are issues due to concurrency.

In the second phase, we had to change backend API tests; now we use 'Mock Services' to accelerate the tests, and we do not create a client before making an HTTP request.

We separated our database into two parts, one for tests and the other for normal use of the app.

We had some difficulties using ‘DOM Tree’ on testing the frontend.