Amanda Menezes 2017124788
João Pedro Rosado Dionísio 2019217030
Miguel Pedroso 2019218176

Sistemas Distribuídos 2021/22
Licenciatura em Engenharia Informática
Junho, 2022

1. Introdução

O trabalho consistiu em desenvolver uma aplicação web capaz para a apresentação e gestão de dados esportivos. Os dados foram obtidos através de solicitações de API a um serviço externo [1] através da tecnologia REST. O principal objetivo deste projeto foi a integração destes dados à plataforma desenvolvida e implementação de funcionalidades complementares para a manipulação dos dados. A camada de dados e Frontend foram desenvolvidos com a framework Spring Boot e Thymeleaf e a programação da base de dados local foi feita com a utilização da Java Persistence Application Programming Interface.

2. Arquitetura da plataforma

A aplicação web foi construída de forma a integrar uma API externa para obter os dados e os importar a uma base de dados local num servidor Postgres. Para isso foi necessário utilizar solicitações em HTTP – como POST, GET e PUT – a fim de realizar operações e obter a informação necessária para cumprir as funcionalidades implementadas. A figura abaixo apresenta de forma ilustrativa o mecanismo de uma aplicação Web que se emprega a arquitetura REST como recurso.

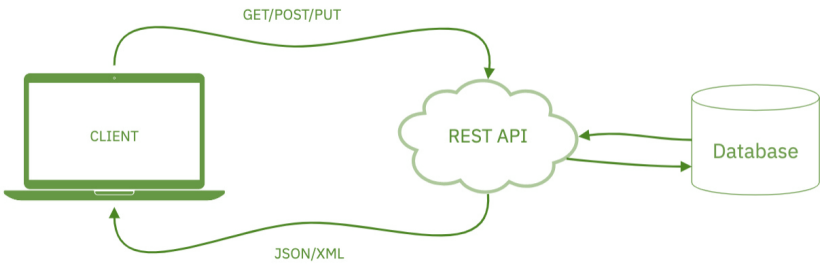

Figura 1. Arquitetura geral da plataforma scoreDEI.

O cliente quando realiza alguma tarefa que requer dados do recurso externo de API precisa realizar pedidos através do comando built-in toJson() como enunciado na expressão (1). que no Backend da aplicação se traduzem em comandos de HTTP como GET e POST. A API então recorre à base de dados própria para obter e assim transferir esses até o cliente final.

No caso particular deste projeto a resposta da API ao cliente é dada em JSON com a função . O ficheiro responsável pelas configurações do projeto em XML reúne todas as livrarias e recursos utilizados em anotações pelo Backend na Framework do Spring Boot. Desta forma, torna-se possível a permuta de dados uma vez que o ficheiro de configurações pom.xml reúne as anotações necessárias (em formato de beans) para receber e manipular estes dados.

(1)

```
response = Unirest.get(host + "/teams?" + s).header("x-apisports-key", x_apisports_key).asJson();
```

Um recurso de configuração automática do Spring Boot configura a aplicação automaticamente para determinadas dependências. Para todas as dependências disponíveis no classpath do projeto o Spring Boot criou os beans para cada uma delas. Estes beans no Spring são objetos que são instanciados e gerenciados pelo Spring por meio de containers Spring IoC.

Backend da plataforma

O esquema abaixo apresenta como o backend da aplição é dividido e qual foi o caminho tomado para a implementação das funcionalidades para serem apresentadas ao cliente final quando acessa à página Web.

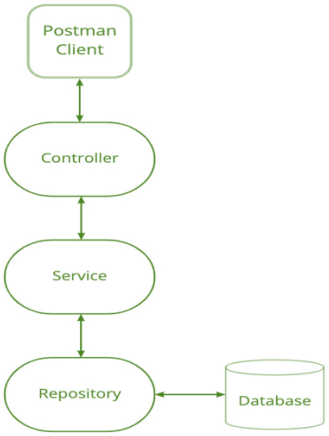

Figura 2. Arquitetura da organização de níveis do Spring Boot.

3. Database

A base de dados foi criada com a utilização de JPA que é uma interface que intercala os dados entre objetos em Java e uma base de dados relacional. Desta forma, este recurso age como ponte entre o modelo orientado a objetos e um sistema relaciona de base de dados.

O modelo de Entidade Relacionamento foi projetado com a utilização da ferramenta ONDA para a representação das tabelas e a relação entre elas.

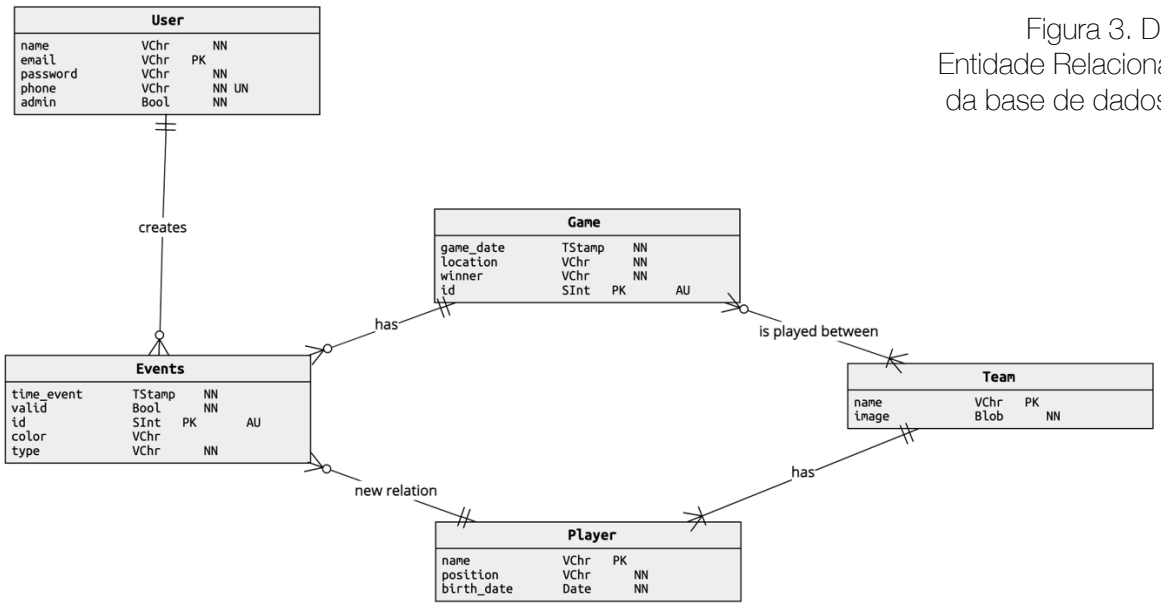

Figura 3. Diagrama Entidade Relacionamento da base de dados local.

Um utilizador autenticado pode criar eventos como a incidência de uma cartão amarelo ou vermelho, uma interrupção no jogo ou registar um golo de um jogador. Seus atributos são:

- Nome: username do utilizador dado por uma string.
- Email de registo
- Palavra passe de registo
- Telemóvel de registo
- Admin: identifica se o utilizador é administrador ou não através de um boolean.

O Evento, por sua vez, é dado por uma entidade que apresenta as três ações possíveis mencionadas anteriormente: cartão, interrupção ou golo. Portanto ele está associado à entidade Jogo pelo relacionamento many-to-one, representado pela anotação @ManyToOne em JPA. Além disso, como um evento pode se passar a um jogador específico no caso de ocorrência de um cartão ou golo, a entidade Evento também se relaciona com Jogador da forma many-to-one. Suas instâncias são:

- Hora do evento: é dado por uma timestamp do momento exato em que o evento ocorreu.
- Válido: permite mostrar se um evento foi validado por um administrador sobre sua ocorrência.
- ID: é a chave primária da tabela e por isso é única para cada evento registado.
- Color: corresponde ao evento de incidência de cartão e pode ser 'yellow' ou 'red'.
- Type: indica o tipo de evento ocorrido e pode ser 'card', 'goal' e 'interruption'

No caso do Jogo, é competido entre duas equipas (entidade Team) e por isso o relacionamento é do tipo many-to-many com a anotação @ManyToMany. É detalhado por:

- Data: data dada por um timestamp em SQL do dia e horário do jogo.
- Localização: estádio em que ocorreu ou ocorre o jogo.
- Winner: a equipa vencedora.
- ID: identificação única de cada jogo que corresponde a uma chave forasteira para a tabela Evento.

A Equipa, por sua vez, é uma entidade que se relaciona a um Jogador da forma one-to-many pela razão de uma equipa ser constituída por vários jogadores. É caracterizada por:

- Nome da equipa, que é uma chave forasteira para a tabela Jogo.
- Imagem do brasão da equipa

Por fim, a entidade Jogador apresenta uma tabela constituída por:

- Nome do jogador, que é uma chave forasteira para a tabela de Evento e Equipa.

- Data de nascimento
- Posição em que joga na sua respetiva equipa.

Todas as entidades foram devidamente implementadas e definidas nas suas respetivas classes utilizando a anotação @Entity.

4. Repository e Service

Todas as classes, consequentemente entidades, que compõe a aplicação devem ter enunciadas as queries para a base de dados com a anotação @Query no ficheiro do seu respetivo repositório. Desta forma uma query fica associada a um método específico, possibilitando sua invocação no DataController que será discutido mais adiante.

Para utilizar queries desenvolvidas em SQL foi preciso utilizar o parâmetro 'nativeQuery=true' em cada um dos métodos criados. Abaixo estão descritos brevemente os principais métodos implementados para a plataforma.

Esses repositórios estão restritamente relacionados aos Services que funcionam como classes de definição dos métodos. Tais métodos são então invocados pelo Data Controller da aplicação a produzir os resultados que serão expostos nas páginas Web. Abaixo estão descritas algumas queries implementadas no repositório das respetivas entidades.

a. Game

i. getGames():

Selecionar a identificação (id) dos jogos que estão a acontecer no momento com o método Para isso os jogos selecionados possuem a timestamp entre o atual momento e duas horas. Este método devolve uma lista de inteiros.

ii. getCurrentGames():

Seleciona toda a informação presente na tabela para os jogos que estão a ocorrer no momento – num intervalo de duas horas.

iii. getGoalsAndLocations():

Seleciona os jogos e suas respetivas localizações recorrendo à coluna da tabela "location".

iv. getGamesIds():

Seleciona os ids do jogos que estão a ocorrer no momento e devolve uma lista de inteiros.

b. Event

As principais queries realizadas foram métodos built-in invocados com recurso a JPA, como por exemplo a inserção de novos eventos à base de dados com o método save(). Os eventos ao serem registados são inválidos por defeito e isto é feito definindo esses valores dentro do método responsável por invocar o save().

i. selectFalseEvents():

Seleciona eventos que ainda não foram validados por um administrador.

c. Team

i. getTeamResults(String name):

Responsável por selecionar as estatísticas de um time e coloca-las a vista na página que reúne as métricas de comparação entre times.

5. Observações do Data Controller

O Data Controller é responsável por interligar os Services associados às Entidades com a página web em HTML. Cada método enunciado invoca métodos e manipula objetos a fim de disponibilizar o dado numa página ou utilizar dado que foram dados num input de uma página.

- Para a integração com a API foi preciso recorrer às respostas das solicitações em HTTP em JSON como referido anteriormente para exportar a informação quanto aos times e jogadores. Os objetos dos times foram passados com a função getJSONArray() que retorna o dado em formato de objetos de equipas. Toda vez que a aplicação é iniciada pelo servidor é preciso conectar-se à API uma vez que a base de dados local não mantém os dados de forma permanente. Por isso o utilizador deve primeiramente realizar o "Create Data" para obter a base de dados originada pela API e assim utilizar as funcionalidades implementadas.

- O mesmo passo é feito para retornar um array com objetos de jogadores que serão colocados posteriormente na tabela de Players. Desta forma todas as informações são direcionadas a cada instância das Entidades.

- Para a listagem de objetos das entidades é preciso utilizar a anotação @GetMapping que será responsável por realizar a solicitação HTTP GET associada ao método da query. Este método é realizado em recurso aos Services das classes que os definem. Isto é feito para os casos de listar eventos, eventos a serem validados pelo administrador, estatísticas e listar jogos entre equipas.

- No caso da anotação @PostMapping esta foi definida utilizada como recurso para salvar o administrador, eventos, jogos e novas equipas. Ou seja, todos os métodos que recorre ao insert na base de dados é preciso realizar uma solicitação HTTP do tipo POST.

- O método setrole(int) é utilizado para o login para guardar a sessão, ou seja, associa um token de forma segura a uma sessão e esse token é por defeito 0. Se um utilizador é registado e se faz login esse token passa a ser 1 e se esse utilizador for um administrado esse token passa a 2. Toda vez que se faz um login é chamado o método.

6. Detalhes de implementações extra

Foi implementado um sistema de permissões que utilizou variáveis de sessão para permitir que certas páginas só podem ser acedidas por utilizadores registados ou administradores. Ao fazer registo de um user, é-lhe atribuído um determinado "role", que define as permissões que esse user tem. Nas páginas que têm acesso restrito, é efetuada a verificação desse role. Caso tenha o role necessário, o user tem acesso à página. Caso contrário, é redirecionado para uma página de erro.

7. Descrição e resultados de testes realizados

Teste	Resultado
Importar dados API externa	pass
Criar/alterar equipa	pass
Criar/alterar jogador	pass
Criar/gerir jogo	pass
Validar eventos num jogo	pass
Registar evento num jogo	pass
Acompanhar jogos a decorrer	pass
Consultar informação dos jogos realizados pelas equipas (vitórias, derrotas, empates, total de jogos)	pass
Consultar estatísticas sobre os marcadores de golos	pass
Consultar informação sobre jogos entre duas equipas específicas	pass
Registar utilizadores	pass
Login	pass

8. Referências

[1] API utilizada: api-sports.io

[2] Building Java Projects with Maven, disponível em: https://spring.io/guides/gs/maven/