UNIVERSIDADE D COIMBRA

Projeto de Sistemas Distribuídos

ScoreDei!

Trabalho realizado por:

- João Silva (2019216753)
- Pedro Martins (2019216826)
- Mário Lemos (2019216792)

Índice

Arquitetura do sistema:

Na **Figura 1** é possível observar uma descrição sucinta da arquitetura usada para elaborar este projeto:

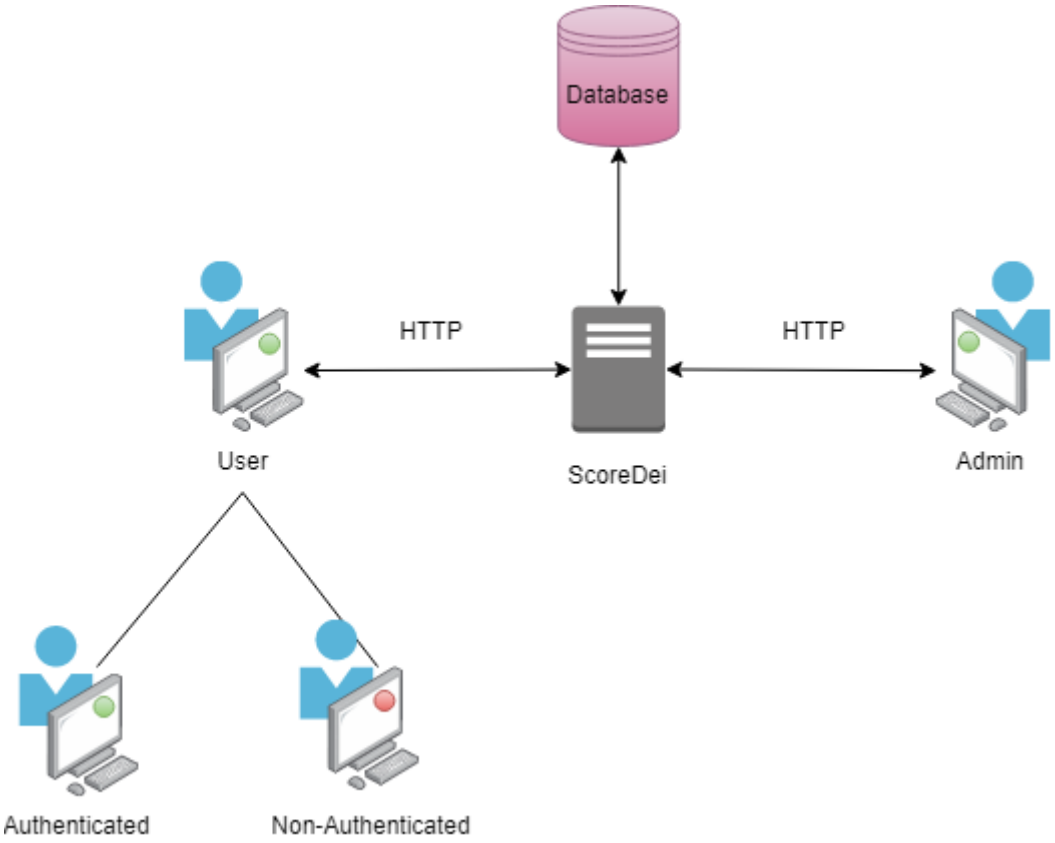

Figura 1 – Arquitetura geral do sistema

Primeiramente, recorremos a diversas tecnologias para compor o todo do projeto. Usamos PostgresSQL + pgAdmin de forma a poder estabelecer e enviar pedidos à nossa base de dados. A nível de servidor usamos a API JPA (Java Persistance API) para integrar as nossas classes na base de dados previamente mencionada, Spring Boot para o website, com as seguintes dependências: thymeleaf, SpringBootWeb, SpringBootDev, unirest e gson, sendo os dois últimos necessários para estabelecer uma conexão com a API sports.io para obter informação relativa a jogadores e equipas da vida real, com o propósito de inserir estes na base de dados. Os primeiros três dizem respeito às vistas HTML e ao desenvolvimento do projeto no conceito website desejado.

Analisando os diferentes tipos de acesso possíveis de acontecer no website, o **Admin** é uma conta cujas permissões indicam que este tem poder administrativo sob a plataforma, contudo um utilizador pode-se distinguir em duas categorias, o que afeta as suas funcionalidades no *ScoreDei*, **autenticado** e **não autenticado**. As diferenças entre vistas e controladores são visíveis nas **Figuras 2, 3 e 4**, onde é demonstrado as diferenças de acesso consoante o tipo de cliente a usá-lo.

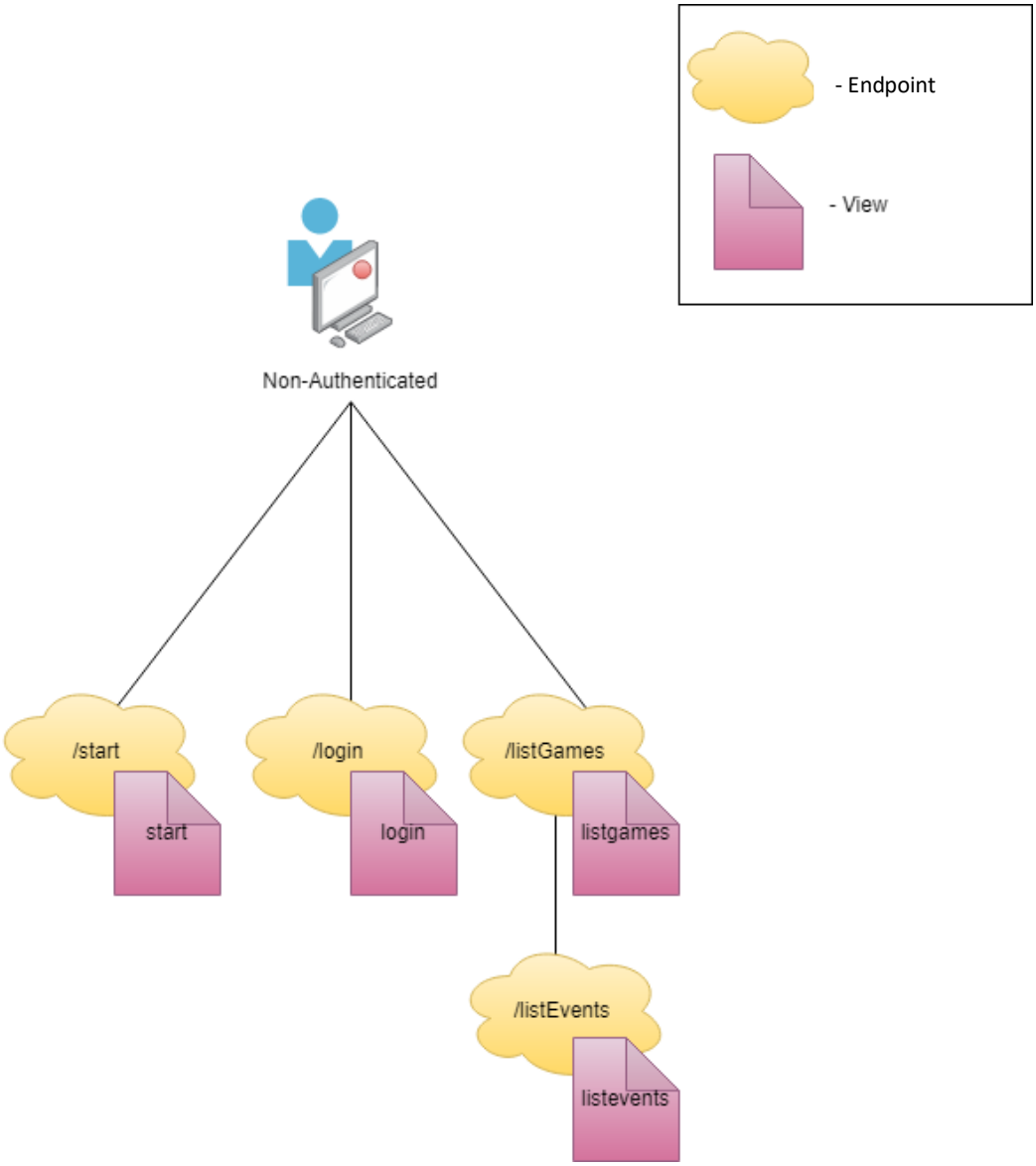

Figura 2 – Acessos e vistas de um utilizador não autenticado

Na **Figura 2** temos presentes as páginas e endpoints acessíveis por um utilizador que não se encontra autenticado. Daqui podemos descrevê-las:

- **/start (start):** Corresponde à página inicial do website. Dá acesso direto aos controladores do seu nível (/login e /listGames).
- **/login (login):** Representa a página de autenticação do cliente. Deverá assim introduzir um nome e palavra-passe que constem da base de dados. Em caso de ou o nome ou a password estarem incorretos, uma mensagem de erro é apresentada na vista.
- **/listGames (listgames):** Constitui a lista de jogos presentes na base de dados. O cliente pode diretamente visualizar os eventos de um jogo (/listEvents).
- **/listEvents (listevents):** Lista completa dos eventos associados a um jogo. O controlador recebe por parâmetro o id do jogo em questão, podendo assim ver o registo dos mesmos.

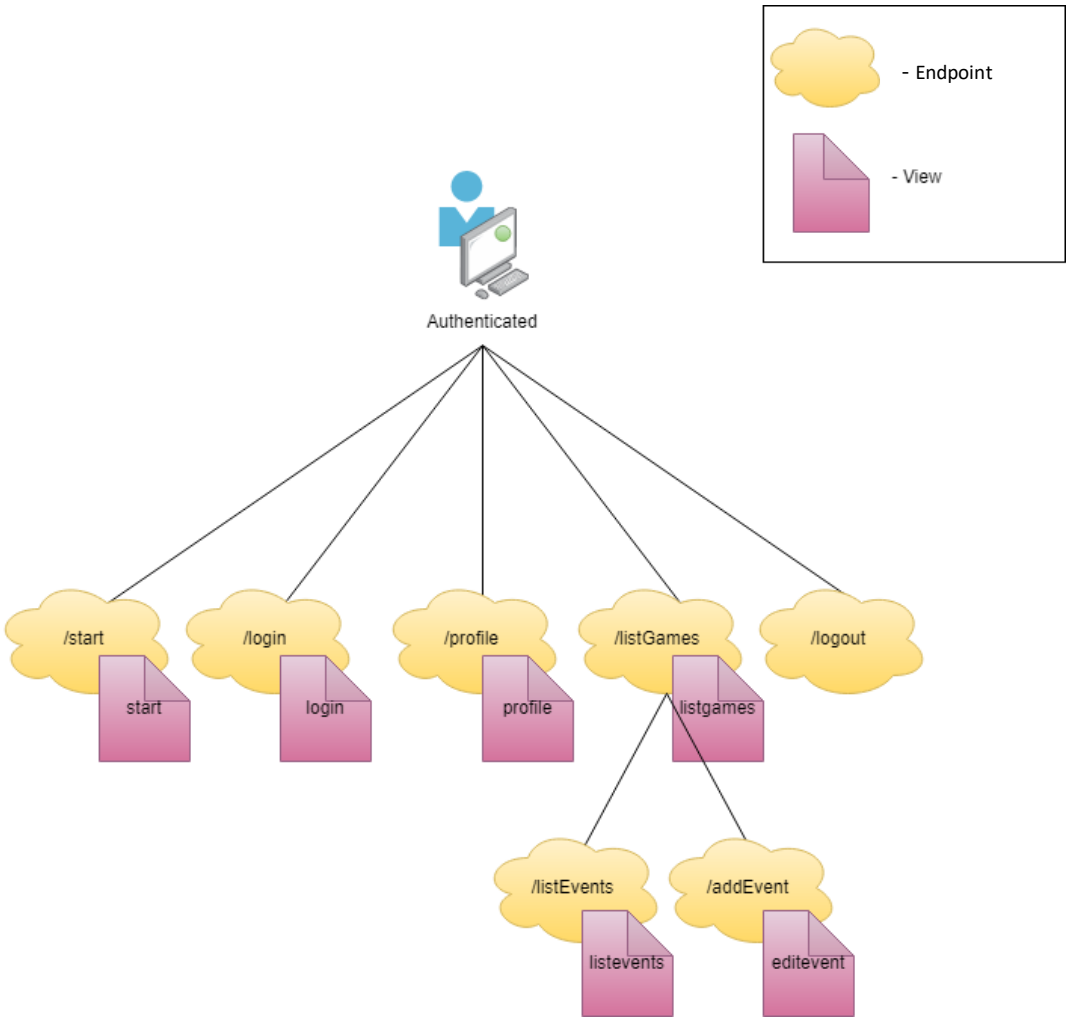

Figura 3 – Acessos e vistas de um utilizador autenticado

Na **Figura 3** estão presentes os endpoints e vistas aos quais um utilizador (não administrador) autenticado consegue aceder. Tendo já explicado o /start e /login, iremos enunciar as características dos outros mencionados:

- **/profile (profile**): Página de perfil do utilizador com a seguinte informação: nome, endereço de e-mail e número de telemóvel.
- **/listGames (listgames):** Tal como descrito anteriormente, só que agora o cliente também pode adicionar eventos a um jogo (/addEvent), selecionando-o e fornecendo assim o id do jogo em questão.
- **/addEvent (editevent):** Permite adicionar um evento a um jogo, podendo o mesmo representar diferentes acontecimentos que influenciam o jogo em questão (terão de ser validados por um administrador da plataforma (/validEvents)).
- **/logout:** Controlador sem vista, uma vez que a plataforma simplesmente remove a sessão do utilizador, devolvendo-o à página inicial como um utilizador não autenticado na plataforma.

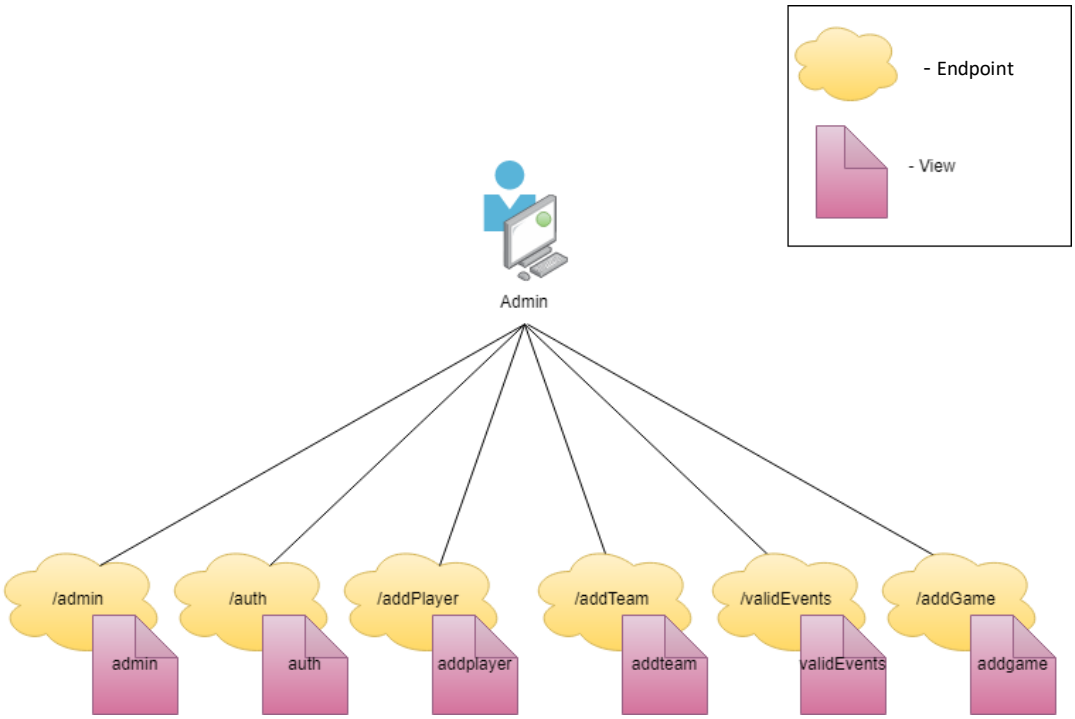

Figura 4 – Acessos e vistas exclusivas de um administrador

Na **Figura 4** estão representados os endpoints e vistas exclusivas de um administrador do ScoreDei. Este também tem acesso a todas as funcionalidades (endpoint e vistas) de um cliente autenticado. De seguida, apresenta-se uma descrição destes endpoints:

- **/admin (admin):** Página de administração, que contém acesso direto a todos os outros controladores em questão para este tipo de utilizador.
- **/auth (auth):** Permite adicionar um novo utilizador à base de dados, sendo necessário fornecer um nome, número de telemóvel, e-mail e se o cliente será um administrador ou não do ScoreDei.
- **/addPlayer (addplayer):** Tem o propósito de adicionar um novo jogador a uma equipa, fornecendo um nome, data de nascimento e posição.
- **/addTeam (addteam):** Adiciona uma equipa à base de dados, fornecendo um nome e imagem para a mesma.
- **/validEvents (validevents):** Serve para validar os eventos introduzidos por utilizadores autenticados, ou seja, todos os eventos terão de passar primeiro por um administrador de forma a serem aprovados e, assim, adicionados ao jogo em questão.
- **/addGame (addgame):** Página para adicionar um jogo futuro. Deverá fornecer como informação as equipas a participar no mesmo, localização e data para a ocorrência do mesmo.

Diagrama de Entidades-Relacionamento e uso da API:

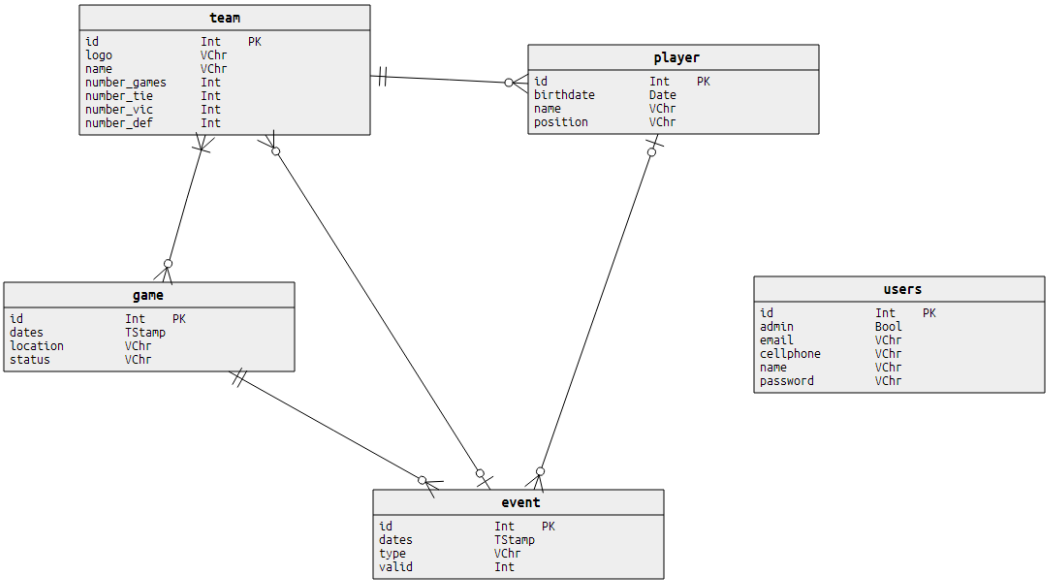

Figura 5 – Diagrama de entidades-relacionamento para a base de dados

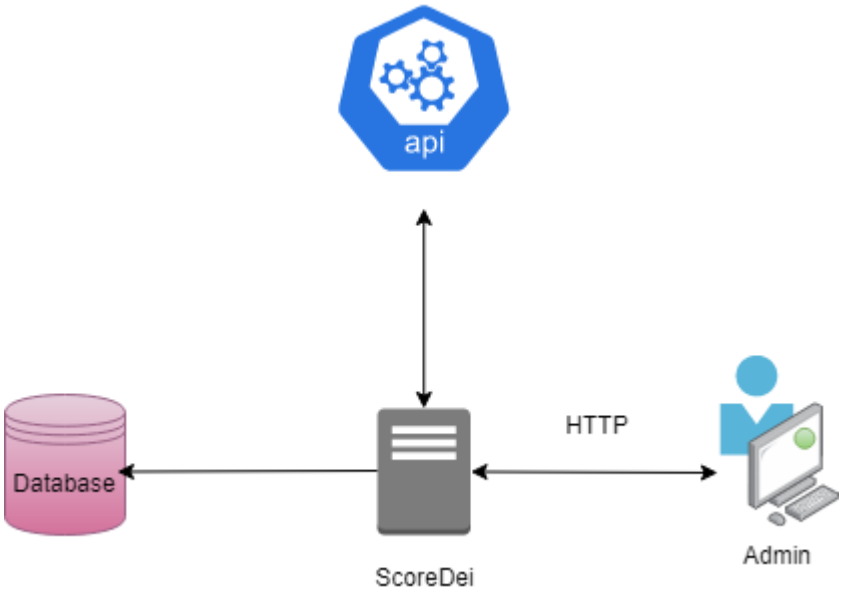

Figura 6 – Uso de API externa no ScoreDei

Implementação do Backend:

Classes, repositórios e serviços:

Iremos agora indicar como se encontra organizado o backend do ScoreDei. Primeiramente, temos os ficheiros .java que constam da pasta data, ou seja, as classes do nosso projeto:

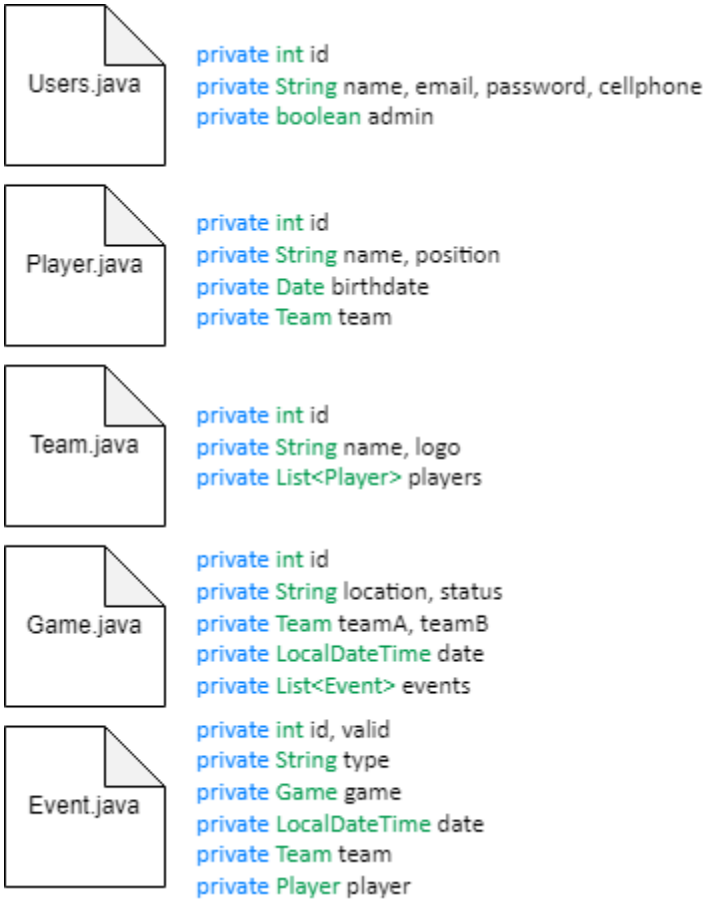

Figura 7 – Classes a usar para o ScoreDei

Para cada um destes ficheiros temos um repositório e serviço correspondente. No primeiro, encontra-se a criação da tabela corresponde na nossa base de dados e as queries SQL necessárias para obter a informação que precisamos desta. De seguida os serviços agrupam os requests necessários para executar estas queries, ou seja, de acordo com a sua nomencleatura, os métodos correspondentes vão acionar queries diferentes que constam no seu repositório, por exemplo, no UsersRepository.java temos uma query, que é invocada através do método "findByName(String s)", que devolve um utilizador da base de dados cujo nome seja equivalente ao argumento s fornecido. Este método é invocado no seu serviço (UsersService.java), num método com nome idêntico. Resumidamente, os serviços são o ponto de contacto entre os nossos controladores e utilizadores e o nosso repositório, ou seja, base de dados.

Queries criadas:

Foram várias as queries criadas se forma a poder retirar da base de dados a informação necessária para garantir o bom funcionamento do ScoreDei. De seguida apresentam-se os métodos criados para tal, assim como uma breve descrição do propósito de cada uma das queries:

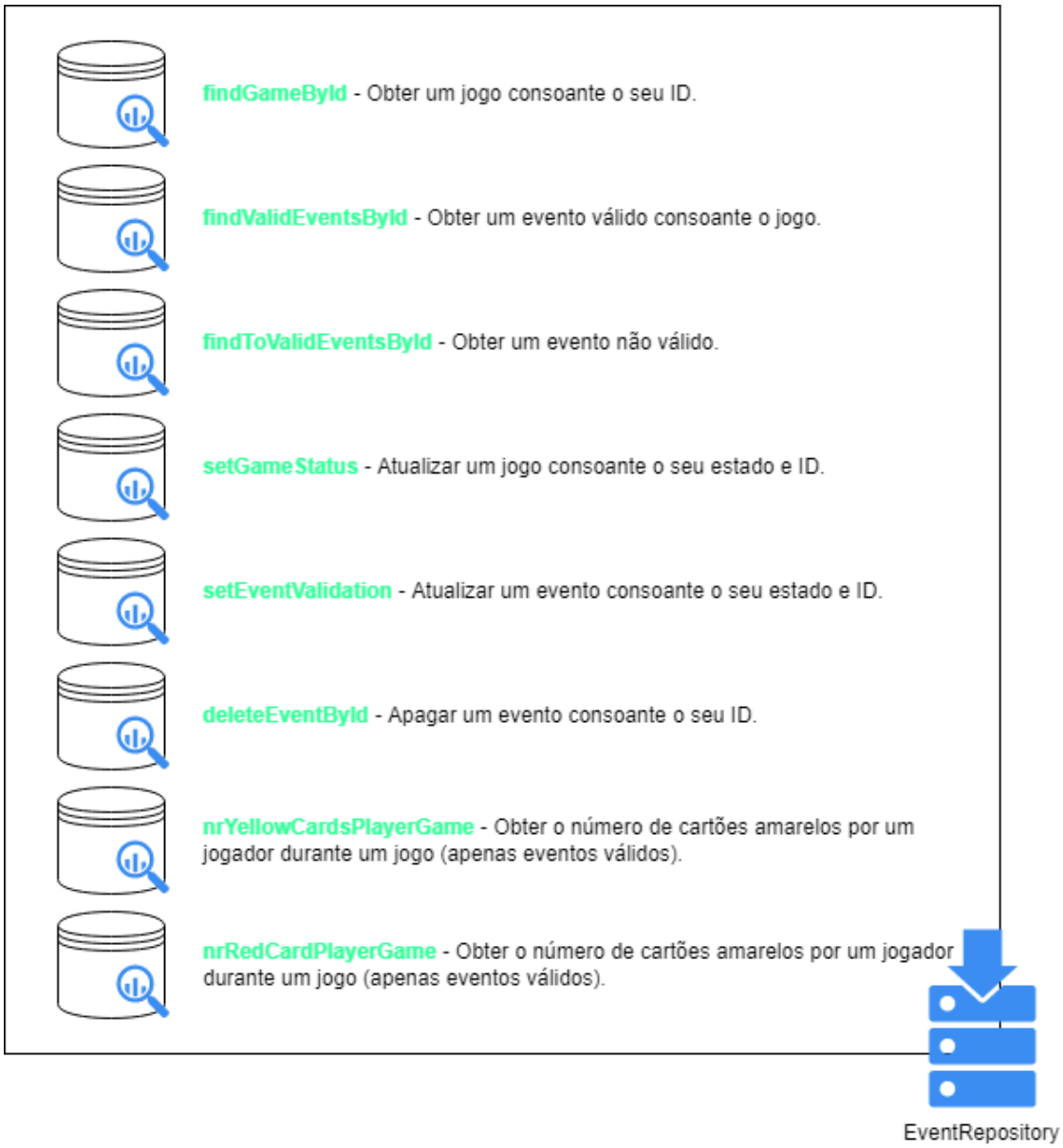

Figura 8 – Queries para o EventRepository

Como é possível observar na **Figura 8**, encontram-se presentes 8 queries para este repositório. As duas últimas representam um auxílio ao cálculo de estatísticas. Ter em conta que "Obter" é equivalente a "select", "Atualizar" a "update" e "Apagar" a "delete".

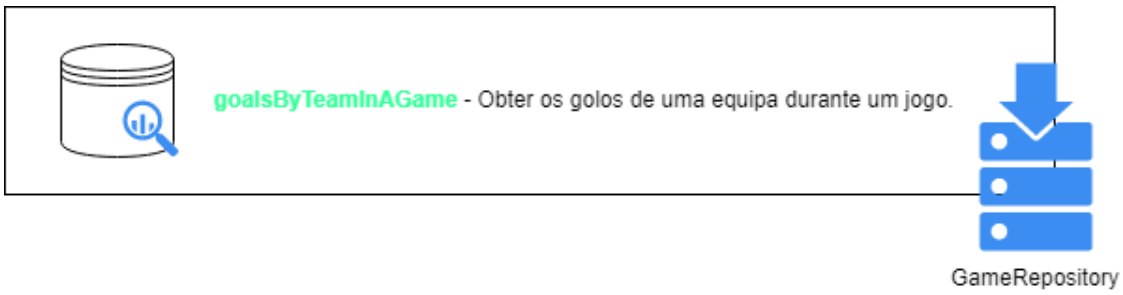

Figura 9 – Queries para o GameRepository

Tal como é possível observar na **Figura 9**, este repositório tem apenas 1 query, usada para obter informação necessária ao cálculo de estatísticas.

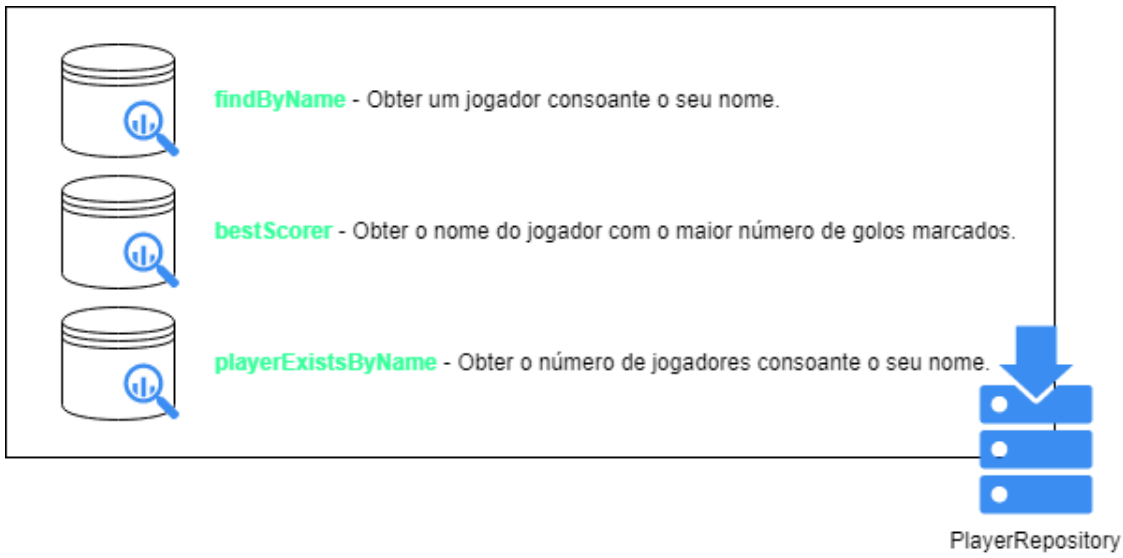

Figura 10 – Queries para o PlayerRepository

Na **Figura 10** temos as 3 queries que existem para obter informação relativa à tabela de jogadores na base de dados. De realçar que a segunda, cujo método é **bestScorer(),** é utilizada no cálculo de estatísticas (neste caso o melhor marcador consoante os jogos efetuados, e eventos válidos adicionados).

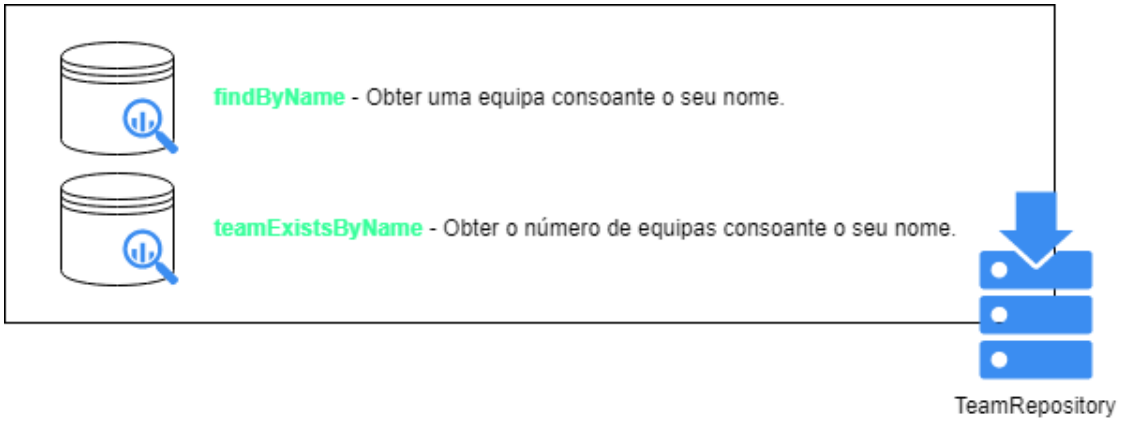

Figura 11 – Queries para o TeamRepository

Como consta da **Figura 11,** existem duas queries para este repositório, sendo ambas a operação "select" na base de dados.

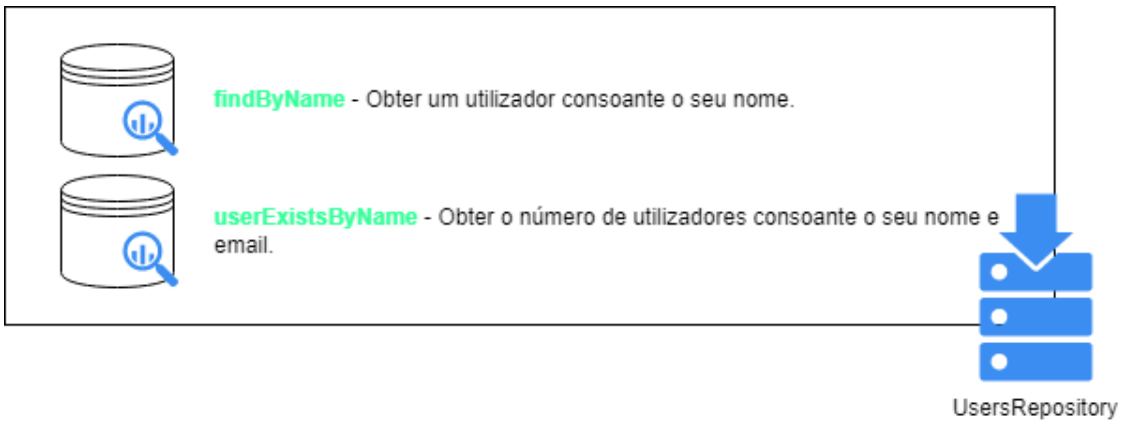

Figura 12 – Queries para o TeamRepository

Na **Figura 12** temos outra vez 2 queries, cujo propósito é semelhante àquele do repositório de equipas (**Figura 11**). Tal como já foi mencionado anteriormente, todas estas queries são usadas pelos serviços, de forma a poder interagir com a base de dados sempre que necessário.

Integração com a API:

Nesta secção iremos explicar como foi feita a integração com a API externa sports.io. Para isso faremos uso da **Figura 6,** relativa à arquitetura do sistema. Primeiramente, foi criada uma conta na API sports.io, de forma a obter uma "key" com a qual podemos efetuar "requests". Estes "requests" são efetuados por um administrador no endpoint /addTeam, onde este tem a opção de adicionar uma equipa da API externa ou ele próprio criar a sua equipa. Caso escolha a primeira opção, também poderá escolher entre importar 20 jogadores dessa equipa ou não. No final toda esta informação é guardada na base de dados, na tabela de equipas (e jogadores caso sejam importados os jogadores pela API).

Em termos de tecnologia, fazemos uso do unirest e do gson para efetuar esta procura de informação e armazenamento da mesma. Toda esta porção do código encontra-se no TeamService, com o método addTeam(). De seguida apresentamos a "key" e o "host" usados para os efeitos deste trabalho:

API Host: "v3.football.api-sports.io"
API Key: "ff63ac8a4fa763467c5f73b4bb747473"

Tratamento de erros:

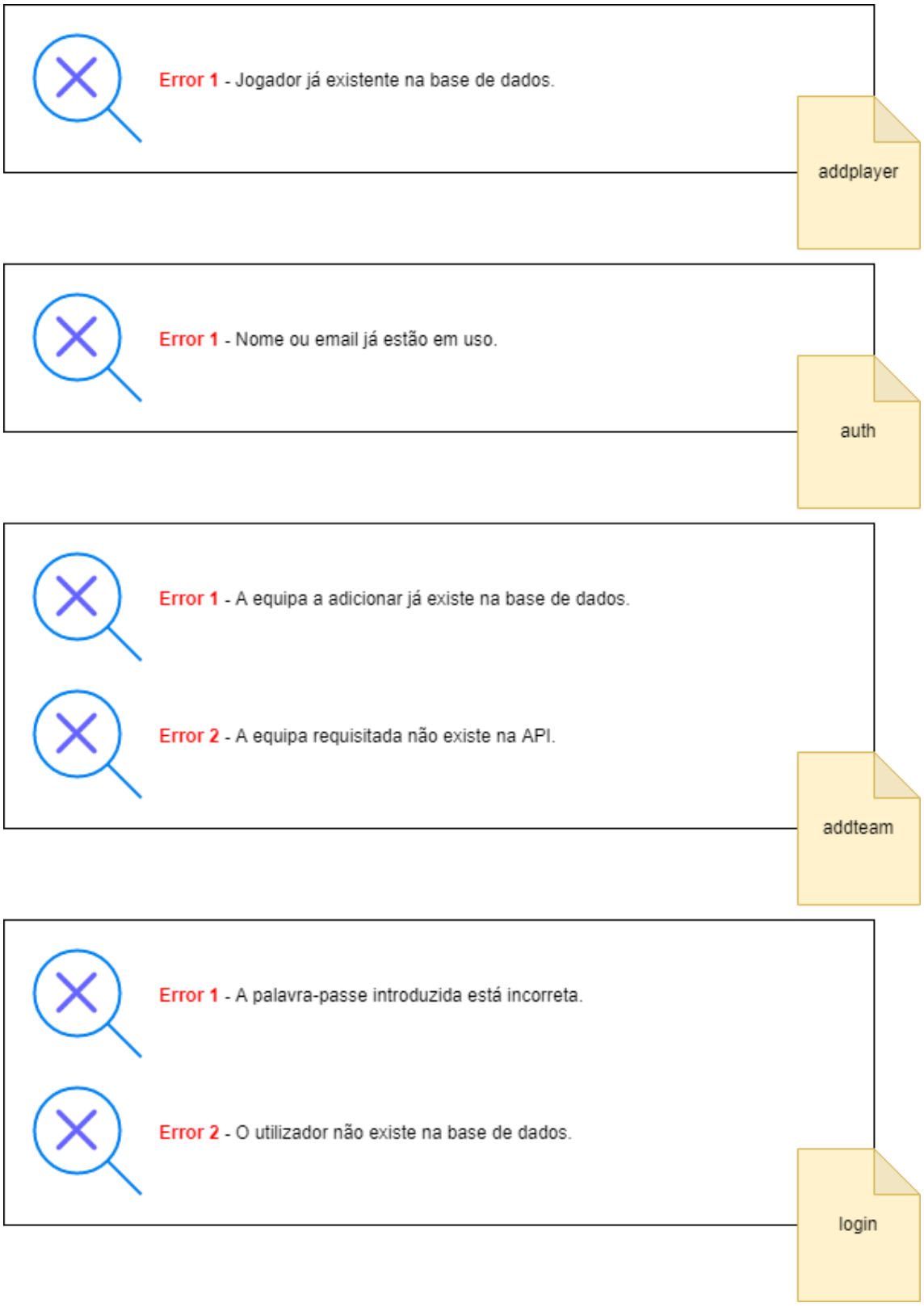

Figura 13 – Lista de erros para as vistas addplayer, auth, addteam e login

Figura 14 – Lista de erros para a vista editevent

Na **Figura 14** estão presentes os vários erros que podem occorrer ao interagir com a página editevent.

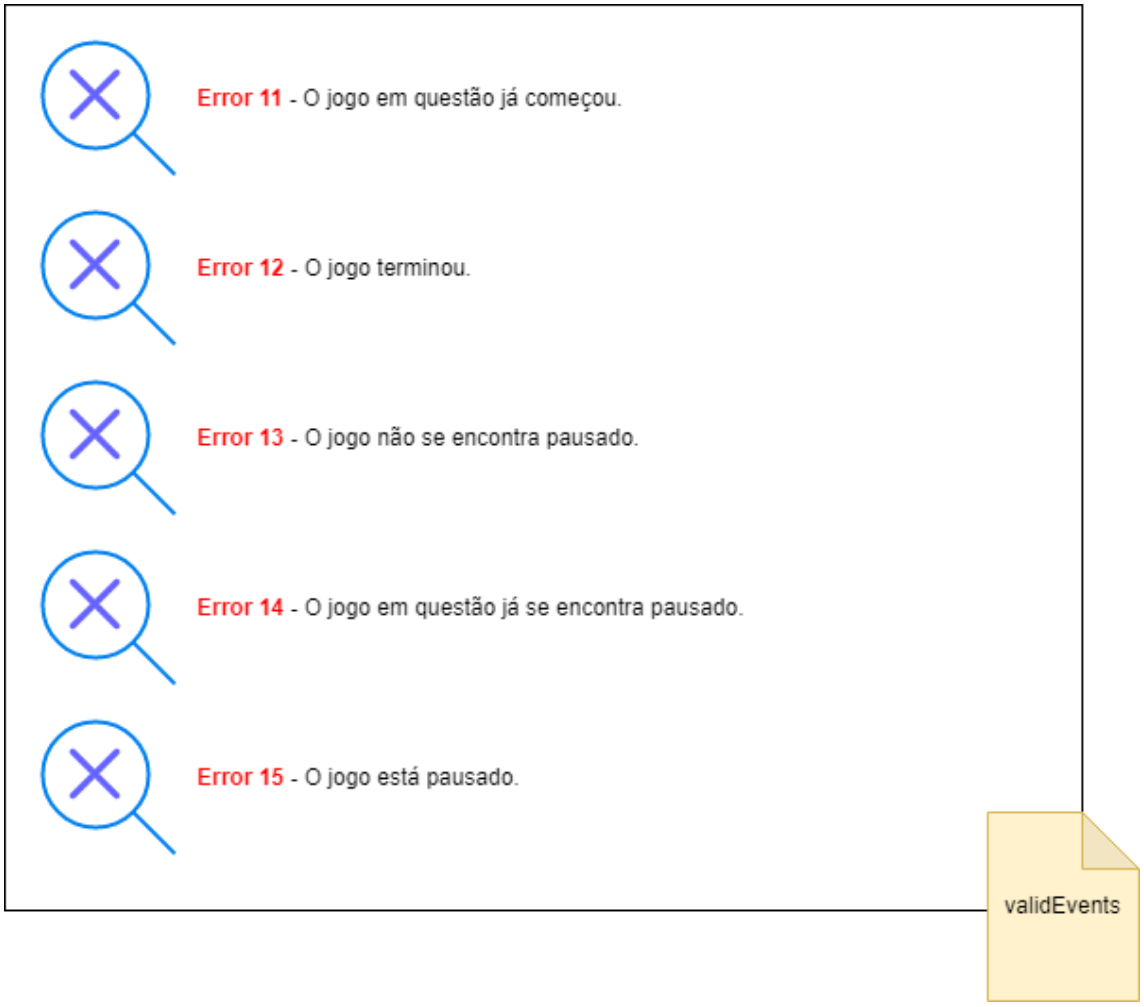

Figura 15 – Lista de erros para a vista validEvents

Todos os tipos de erros diferentes que estão demonstrados nas **Figuras 13, 14 e 15** apresentam mensagens de erro na vista do utilizador, na mesma página do mesmo. Estes têm uma grande importância uma vez que permitem manter os utilizadores bem informados do que podem ou não fazer no ScoreDei, assim como servir como ferramenta de "debug" sobre possíveis erros que possam originar da interação com as páginas e endpoints.

Controlo de acesso de utilizadores:

Por fim temos de garantir que cada tipo de utilizador diferente apenas tem acesso às funcionalidades que deve ter. Para isso primeiramente procedemos à criação de uma classe que armazene os dados do utilizador a utilizar a plataforma:

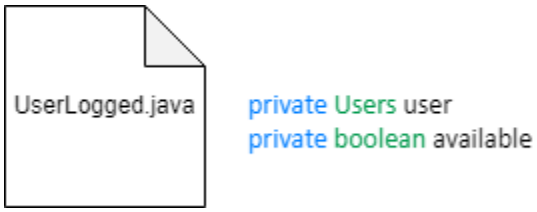

Figura 16 – Classe UserLogged para guardar dados de autenticação

Além dos setters e getters necessários, também possui o método **isAvailable()**, que indica ao programa se existe um utilizador autenticado ou não autenticado. Na variável user é guardado o utilizador, caso este esteja autenticado. Isto permite-nos obter informação sobre o mesmo, sobretudo sobre o nível de permissões do mesmo (administrador) sobretudo para fornecer em caso positivo, acesso ao painel de administrador e suas funcionalidades. Ao efetuar logout (endpoint /logout), a variável available volta a estar a false, e o utilizador é prontamente removido do sistema. É esta classe que permite estabelecer o controlo apresentado nas vistas e endpoints estabelecidos nas **Figuras 2, 3 e 4.**

Implementação das funcionalidades do controlador:

Todos os endpoints que constam do programa, e estão descritos nas **Figuras 2, 3 e 4,** fazem parte do controlador DataController.java, incluindo assim todas as funcionalidades do programa. A funcionalidade mais complexa aquando da sua implementação foi o estabelecimento e organização inicial, incluindo a base de dados. O ER da **Figura 5** representa o produto final contudo, foram realizadas múltiplas alterações, sobretudo ao tipo de relacionamentos entre objetos de classes diferentes (@ManyToMany, @ManyToOne, @OneToMany) tendo este ponto representado um grande custo de tempo até chegar a um estado de tabelas ideal para as futuras funcionalidades. Outra complicação depreendeu-se com o facto de registar os eventos, uma vez que recorremos a JavaScript para implementar as alterações necessárias da janela (explicado no capítulo à frente), necessitámos de aprender esta tecnologia, o que representou ao início algumas dificuldades até incorporar esta funcionalidade ao nosso gosto. Por fim, todo o trabalho desenvolvido com HTML e CSS necessitou, tal como o JavaScript, de alguma aprendizagem visto que estas linguagens até agora nos eram um pouco desconhecidas a um nível mais aprofundado.

Implementação de vistas com atualização automáticas:

Definimos uma vista com atualização automática quando o programa atualiza a mesma com base nalguma alteração realizada. No nosso caso, isto acontece em duas vistas, **addTeam** e **editevent,** com uma breve descrição das mesmas presentes na **Figura 16**:

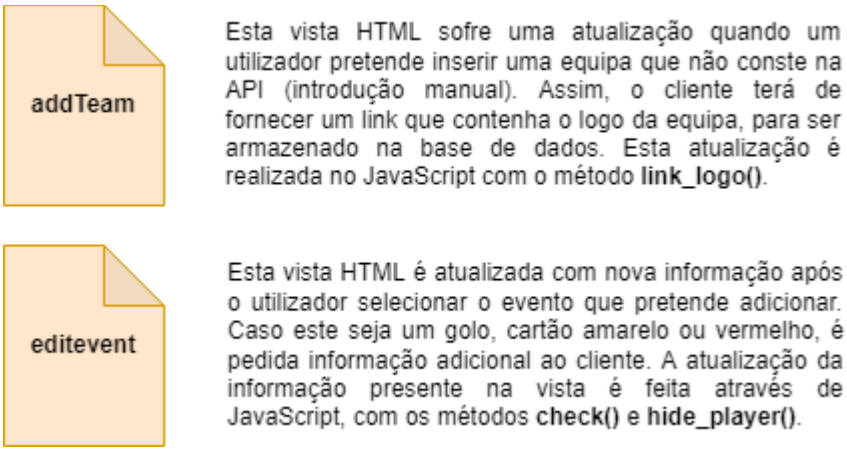

Figura 16 – Vistas com atualização automática

Testes efetuados à plataforma:

Ao longo deste projeto fomos efetuando diversos testes de forma a garantir a fiabilidade e boa funcionalidade dos sistemas implementados. Para isso, guardámos um registo destes testes, como consta na tabela seguinte:

ID	Descrição do teste efetuado	✓/✗
1	Como utilizador não registado consigo visualizar a lista dos jogos	✓
2	Como utilizador não registado consigo visualizar a lista de eventos de cada jogo	✓
3	Como utilizador não registado consigo visualizar as estatisticas dos jogos e melhores marcadores	✓
4	Ao efetuar login, se colocar um nome de utilizador inexistente, é apresentado um erro “The user dosen´t exist”	✓
5	Ao efetuar login, se utilizando um nome de utilizador correto, introduzir a password errada, é apresentada a mensagem de erro “Wrong password”	✓
6	Ao efetuar login como utilizador, se utilizar as credencias corretas é efetuado então o login com sucesso	✓
7	Estando logged in como utilizador, posso visualizar uma pagina “Profile”, onde se encontram informações detalhadas sobre o user	✓
8	Estando logged in como utilizador, posso visualizar a lista de todos os jogos	✓
9	Estando logged in como utilizador, posso visualizar as estatisticas dos jogos e melhores marcadores	✓
10	Estando logged in como utilizador, posso vizualizar a lista de eventos de cada jogo	✓
11	Estando logged in como utilizador, posso submeter um novo evento, escolhendo o seu tipo, a data em que se realizou, e outras especificações	✓
12	Estando logged in como utilizador, consigo dar log out com sucesso	✓
13	A qualquer momento é possivel carregar no logo “ScoreDei!” no canto superior esquerdo para regressar a página inicial	✓
14	Utilizando as credenciais corretas é possível efetuar login como admin	✓
15	Como admin, posso registar um novo user, seja este um utilizador normal ou um admin	✓
16	Ao registar um novo user é necessario fornecer informacção detalhada sobre o mesmo, existindo protecção sobre esta	✓
17	Depois de ser registado um user, é possivel efetuar login usando as suas credenciais	✓
18	Como admin, posso adicionar uma nova equipa	✓
19	Como admin, posso adicionar um novo jogador a uma equipa	✓
20	Como admin, posso criar um novo jogo	✓
21	Como admin, posso visualizar uma lista com todos os jogos	✓
22	Como admin, posso visualizar as estatisticas dos jogos e melhores marcadores	✓
23	Como admin, posso editar os dados de um jogador	✓
24	Como admin, posso editar o nome de uma equipa	✓
25	Como admin, posso visualizar a lista de eventos de cada jogo	✓
26	Como admin, posso submeter um novo evento para qualquer jogo	✓

27	Como admin, posso validar qualquer evento que tenha sido submetido por users ou admins	✓
28	Como admin, posso eliminar qualquer evento que tenha sido submetido por users ou admins	✓
29	Ao submeter um novo evento, caso este seja o primeiro do jogo, tem de ser obrigatoriamente “Start the Game”	✓
30	Só é possivel submeter outros eventos que não sejam “Start the Game” quando este jogo começar	✓
31	Validação das datas dos eventos	✗
32	Não é possivel submeter eventos depois do jogo ter terminado	✓
33	Não é possivel validar eventos sobre um jogo que já terminou, (caso de eventos que tenham sido submetidos antes deste terminar)	✓
34	Não é possível submeter eventos de “Resume de Game” se o jogo já estiver a decorrer; nem eventos de “Pause the Game” caso este já esteja pausado; nem eventos de “Start the Game” caso este já tenha começado	✓
35	Não é possível validar eventos de “Resume de Game” se o jogo já estiver a decorrer; nem eventos de “Pause the Game” caso este já esteja pausado; nem eventos de “Start the Game” caso este já tenha começado	✓
36	Como utilizador não registado, posso visualizar as estatisticas entre o confronto direto de equipas a minha escolha	✓
37	Como utilizador registado, posso visualizar as estatisticas entre o confronto direto de equipas a minha escolha	✓
38	Como admin, posso visualizar as estatisticas entre o confronto direto de equipas a minha escolha	✓
39	Não é possível visualizar as estatisticas entre duas equipas caso seja selecionado duas vezes a mesma equipa	✓