UNIVERSIDADE DE COIMBRA

SD – scoreDei

2019216646 – António Correia

2019216764 – João Monteiro

2019216809 – Miguel Faria

Índice

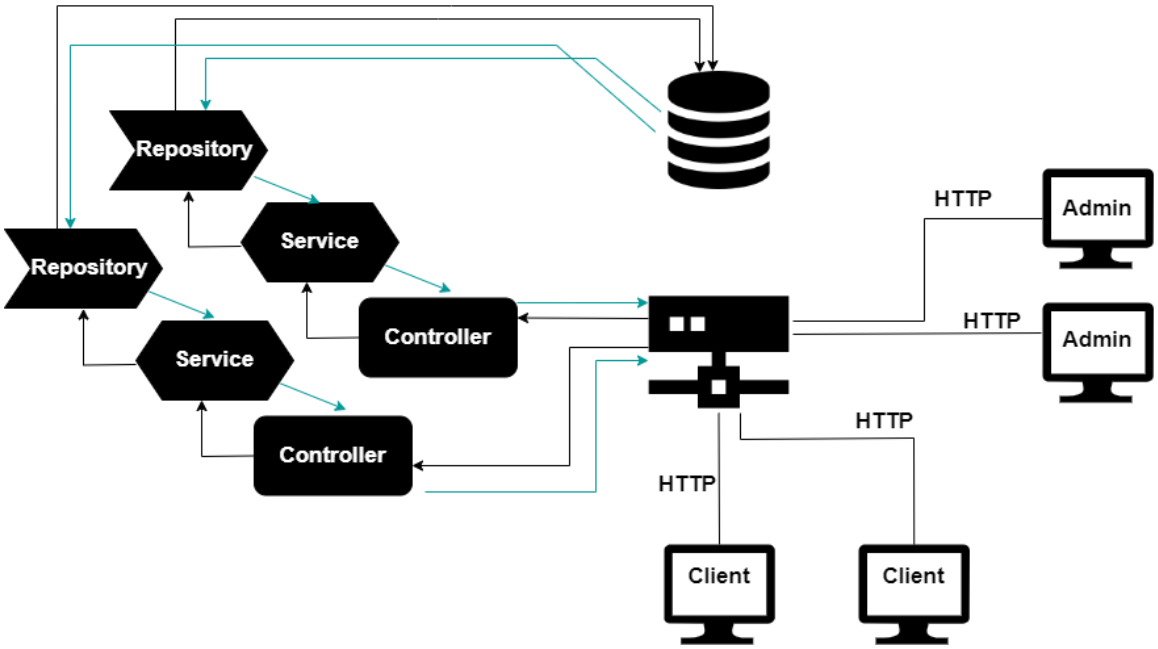
Introdução

Este projeto tem por objetivo desenvolver e adquirir conhecimentos sobre Spring Boot, Thymeleaf, Java Persistence Application Programming Interface (JPA) e integração com o serviço externo api-sports.io.

Foi desenvolvida a aplicação scoreDEI, que pretende simular uma plataforma de jogos de futebol da Primeira Liga, onde é possível acompanhar estes jogos, criar eventos e obter informações sobre equipas e jogadores, bem como funcionalidades de *admin*.

O seguinte relatório tem por objetivo identificar e explicar as funcionalidades desenvolvidas, bem como a utilização da plataforma. Estão também presentes as tarefas elaboradas e identificados os testes realizados à plataforma e consequentes resultados.

Arquitetura da plataforma



Este diagrama tem por fim apresentar de forma simplificada a arquitetura da plataforma, na qual clientes e *admins*, através de *http requests*, irão impor sobre a plataforma as operações ilustradas de modo a obter os conteúdos da base de dados.

A plataforma encontra-se organizada por controladores, serviços e repositórios. Os controladores estão associados por tabela, possuindo cada um funcionalidades relacionadas, com a exceção de alguns que melhoram a experiência da aplicação, nomeadamente no que diz respeito ao menu, gestão de erros e preenchimento da base de dados. Os serviços permitem fazer a articulação entre os controladores e os repositórios, de modo a aumentar a segurança da plataforma; com esse intuito procurou-se também recorrer ao uso de *Forms* que permitem fazer comunicação entre o *backend* e *frontend*. Os repositórios realizam as transações com a base de dados, devolvendo, inserindo, removendo ou atualizando dados.

Os controladores são:

- ErrorController
- EventController
- LoginController
- MatchController
- MenuController
- PlayerController
- PopulateController
- StatisticsController
- TeamController
- TeamPlayerController

Os Serviços são:

- EventService
- LoginService
- MatchService
- PlayerService
- TeamPlayerService
- TeamService

Os Repositórios são:

- EventRepository
- LoginRepository
- MatchRepository
- PlayerRepository
- TeamPlayerRepository
- TeamRepository

Endpoints do tipo GET

ErrorController:

/error – destinado a todos os erros que ocorram durante o uso da plataforma.

EventController:

/allmatches/{match}/regevent – registar um evento no jogo com id {match}.
/allmatches/{match}/manageevent – validar eventos no jogo com id {match}.

LoginController:

/ –não colocando nenhum conteúdo no *url*, é redirecionado para o menu.
/login – vista onde é inserido as credenciais para fazer login.
/reguser – vista onde o *admin* regista um novo *user*.

MatchController:

/regmatch – vista onde o *admin* regista um novo jogo.
/allmatches – vista que apresenta todos os jogos.
/allmatches/{match} – vista de acompanhamento de um jogo com id {match}.
/allmatches/{match_id}/managematch – admin gere dados do jogo.

MenuController:

/menu – menu principal que permite navegar a plataforma.

PlayerController:

/regplayer – vista onde o *admin* pode registar um novo jogador.
/allplayers – vista que lista todos os jogadores existentes na base de dados.
/allplayers/{player_id} – vista com informação do jogador de id {player_id}.
/allplayers/{player_id}/manageplayer – *admin* gere dados do jogador.

PopulateController:

/populate – acesso ao serviço externo para preenchimento da base de dados.

StatisticsController:

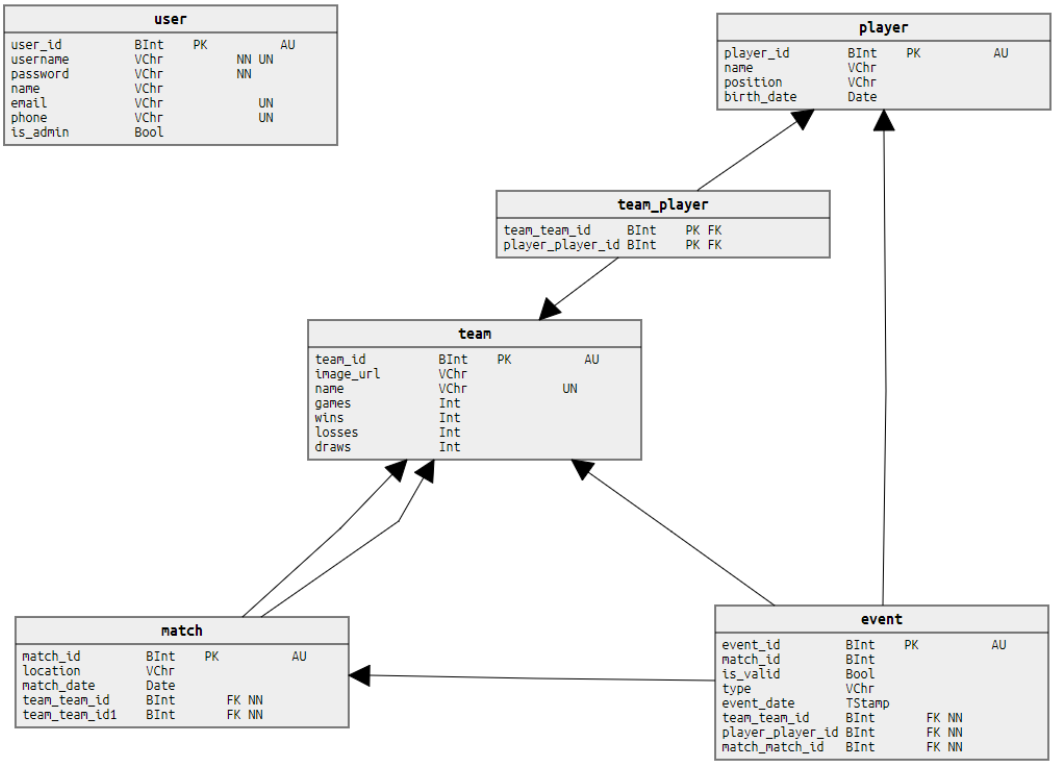
/statistics – vista de apresentação de todas as estatísticas indicadas.

TeamController:

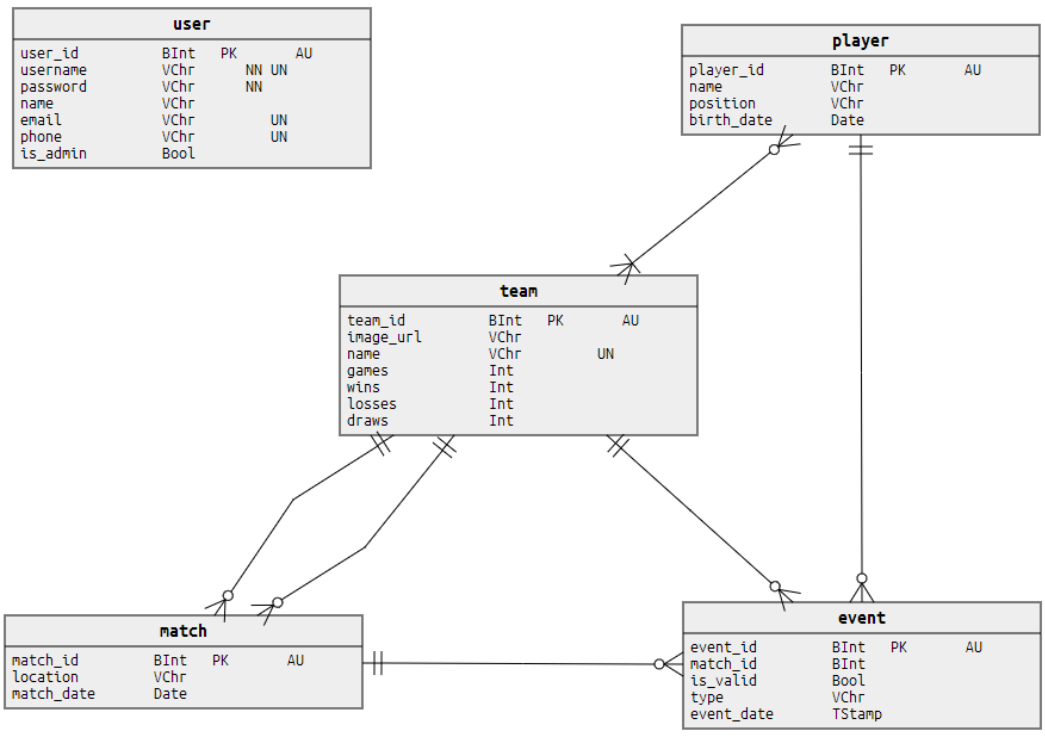
/regteam –*admin* regista uma equipa
/allteams – vista que lista todas as equipas existentes na base de dados.
/allteams/{team_id} – apresentação detalhada da equipa de id {team_id}
/allteams/{team_id}/manageteam – *admin* gere uma equipa

Entity Relation Diagram

Physical



Conceptual



É de notar que muitos dos nomes presentes nas tabelas não estão de acordo com ficheiro SQL usado para gerar as tabelas; isto deve-se ao facto da ferramenta usada, *Onda.dei*, gerar estes nomes de forma automática, não sendo possível alterá-los.

A tabela *user* não se encontra ligada a mais nenhuma tabela; isto deve ao facto de não estar associada a mais nenhuma entidade, sendo os seus valores independentes, usados apenas para verificar permissões.

A tabela *team* possui uma ligação de 0 para n e 1 para n com *player*, uma vez que, ao registar uma equipa nova, esta pode não possuir nenhum jogador inicialmente; contudo a inserção de um jogador obriga a indicação da equipa à qual pertence. Esta ligação irá gerar uma tabela, *team_player*, permitindo guardar as associações entre equipa e jogador.

A tabela *events* possui ligações de 1 e apenas 1 para 0 e n com as tabelas *team*, *player* e *match*, em consequência de existirem eventos nos quais é necessário saber os agentes envolvidos. Com o objetivo de evitar um número alargado de tabelas, todos os eventos são alocados numa mesma, contudo nem todos irão necessitar de identificação de *player* ou *team.* Nestas situações, estes valores são colocados a 0, sendo que todos os *ids* possuem um auto incremento, que é inicializado a 1.

Match possui duas ligações com *team*: desta forma é possível na ocorrência de uma match saber as duas equipas envolvidas.

Backend

Acesso ao serviço externo

Para popular a base foi usada como recurso a *api* api-sports.io. Para realizar esta tarefa é necessário fazer uso de uma *api key*, contudo esta já se encontra no código fonte não sendo necessário um *admin* criar uma conta para executar esta operação.

Distinção entre tipos de utilizadores

Para realizar a distinção entre um *admin*, um utilizador registado e um utilizador não registado, foi criado um *token* de acesso. Ao realizar o login, a informação recebida da base de dados indica o estatuto do utilizador. Um *token* é criado onde nele esta presente este estatuto, sendo de seguida guardado como cookie para ser acedido por diferentes *endpoints*. Se um *endpoint* é restrito, a cookie gerada é acedida, onde se verifica se este possui permissão para entrar; caso isto não se verifique, é redirecionado para uma página de erro onde é indicado que este não possui permissão para realizar aquela tarefa. Para distinguir um utilizador não registado, o mesmo é feito nas páginas nas quais é necessário login, verificando-se se a cookie necessária está presente; caso não esteja, este é também redirecionado para a mesma página de erro. É de notar que ao usar a plataforma não aparecem as ferramentas para aceder a locais aos quais um *user* não possui permissão para entrar, sendo apenas possível ocorrer as situações mencionadas acima se um utilizador tentar aceder alterando o *url*.

Atualização de forma automática

Para obter as estatísticas de vitórias, derrotas e empates, de forma correta, é analisada as estatísticas de cada equipa. Podendo obter estes valores apenas a partir dos eventos, foi determinado que seria menos intenso na base de dados manter estes valores na tabela de cada equipa. Um evento que indica o fim de uma partida, ao ser validado, resulta no cálculo dos golos de cada equipa nesse jogo, sendo de seguida feita a atualização dos resultados das equipas envolvidas, ou seja, o registo de eventos do tipo fim de partida, válidos, irão implicar uma atualização dos dados de uma equipa.

Frontend

Uma vez que não era o foco da plataforma, todo o *frontend* está implementado de forma bastante rudimentar, tirando certos aspetos os quais achamos necessários para facilitar o uso das funcionalidades. Para tal foi ainda necessário adquirir e apreender conhecimentos na área do HTML e JavaScript. Estes revelaram ser mais trabalhosos do que o que estava inicialmente planeado.

Testes

Testes	Pass/Fail	Resultado
Registar utilizador.	Pass	Utilizador registado com sucesso.
Aceder por url ao endpoint de registar user, por um user que não é admin ou não registado.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Criar Equipa.	Pass	Equipa criada com sucesso
Aceder por url ao endpoint de criar equipa, por um user que não é admin ou não registado.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Gerir Equipa.	Pass	É possível alterar dados e jogador da equipa com sucesso
Aceder por url ao endpoint de gerir equipa, por um user que não é admin ou não registado.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Criar Jogador.	Pass	Jogador criado com sucesso
Aceder por url ao endpoint de criar jogador, por um user que não é admin ou não registado.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Gerir Jogador.	Pass/Fail	É possível alterar os valores do jogador, contudo é necessário inserir sempre a data.
Aceder por url ao endpoint de gerir jogador, por um user que não é admin.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.

Criar Jogo.	Pass	O Jogo é criado
Aceder por url ao endpoint de criar jogo, por um user que não é admin ou não registado.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Gerir Jogo.	Pass/Fail	É possível alterar valores, contudo é necessário inserir sempre a data.
Aceder por url ao endpoint de gerir jogo, por um user que não é admin.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Gerir Evento.	Pass	O admin consegue validar e invalidar eventos.
Aceder por url ao endpoint de gerir evento, por um user que não é admin.	Pass	Apenas admin consegue aceder sendo os outros redirecionados.
Registar evento.	Pass	Evento criado.
Aceder por url ao endpoint de registar evento, por um user que não esta registado.	Pass	User não registado não consegue aceder.
Acompanhar jogo.	Pass	É possível acompanhar jogo
Consultar estatísticas.	Pass	É possível ver as estatísticas
Registar uma equipa com um nome que já existe.	Pass	É indicado que este nome já existe.

Registar um player com um nome que já existe.	Pass	É indicado que o nome já existe
Registar um player e não indicar equipa ou indicar uma que não existe.	Pass	É indiciado que o campo de equipa não esta correto.
Criar um jogo que no calendário atual não pode ocorrer.	Fail	Não são feitas verificações das datas. Pode existir o mesmo jogo duas vezes ao mesmo tempo.
Colocar url que não existe.	Pass	É indicado que algo correu mal ou chamar aquele url.
Criar um player ou jogo com formato da data errado.	Pass/Fail	Se a data não for inserida no formato correto, aparece uma mensagem de erro mas não indica qual.
Não preencher formulário de registar user, na totalidade ou de forma incompleta.	Pass	É indicado que algo esta errado ou imcompleto
Não preencher formulário de registar team, na totalidade ou de forma incompleta.	Pass	É indicado que a equipa já existe ou que os campos estão incorretos. É permitido criar equipa sem imagem.
Não preencher login ou com credenciais erradas.	Pass	É indicado que as credencias não estão corretas.
Tentar validar um evento que não pertence aquele jogo ou que não existe.	Pass	É indicado que o id não existe ou que existe mas não pertence aquele jogo.
Registar evento sem completar campos.	Pass/Fail	Se a data e o tipo de evento não foi selecionado, aparece uma mensagem erro mas não indica qual.

Conclusão

Com este projeto foi possível desenvolver e adquirir diversos conhecimentos, sobre a matéria mencionada, Spring Boot, Thymeleaf, Java Persistence Application Programming Interface (JPA), para além destes também nas áreas de HTML e JavaScript. Foi possível observar a quantidade de erros que podem ocorrer e a dificuldade em detetar os mesmos.

Após a conclusão deste projeto, constata-se que podia existir um maior desenvolvimento, principalmente no que diz respeito ao *frontend*, contudo, não sendo o foco deste trabalho, acreditamos que as funcionalidades se encontram desenvolvidas na sua totalidade.