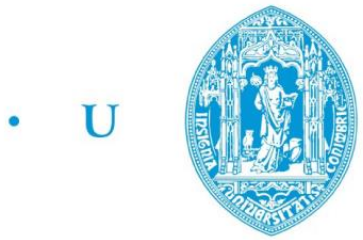

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

scoreDEI: Resultados desportivos em direto

Sistemas Distribuídos

Licenciatura em Engenharia Informática

2021/2022

Duarte Emanuel Ramos Meneses – 2019216949 – duartemeneses@student.dei.uc.pt

Patrícia Beatriz Silva Costa – 2019213995 – patriciacosta@student.dei.uc.pt

Índice

Introdução

Atualmente é cada vez mais usual a sociedade querer informação em tempo real. A cada dia que passa a necessidade de obter informação instantânea aumenta e isso serve de mote para este trabalho prático.

Com o evoluir da tecnologia, muitas áreas aproveitaram essa evolução e reinventaram-se. Uma dessas áreas foi sem dúvida o acompanhamento dos eventos desportivos. Há muitos anos, a única opção que existia era ouvir o relato de um jogo na rádio. Hoje em dia, qualquer smartphone consegue ter todas as informações de um jogo em tempo real, em qualquer parte do mundo.

Com isto em mente, este trabalho prático visa replicar uma plataforma que permite acompanhar resultados desportivos em direto com a contribuição dos utilizadores.

Ao longo deste relatório explicamos os nossos dilemas ao longo do desenvolvimento da aplicação, que opções tomamos, como está organizado o backend e o frontend e mostramos os testes realizados à plataforma.

Arquitetura da plataforma

Era-nos pedido que desenvolvêssemos uma plataforma capaz de disponibilizar resultados desportivos em direto. Para isso, criamos tanto a parte de *frontend* como de *backend*, de modo a que o cliente, através do frontend consiga interagir com a plataforma e envie pedidos ao *backend* que os recebe e processa em endpoints específicos. Tanto os clientes como os administradores (clientes com funcionalidades extra) acedem à plataforma via HTTP.

De um modo geral, a arquitetura da plataforma é a seguinte:

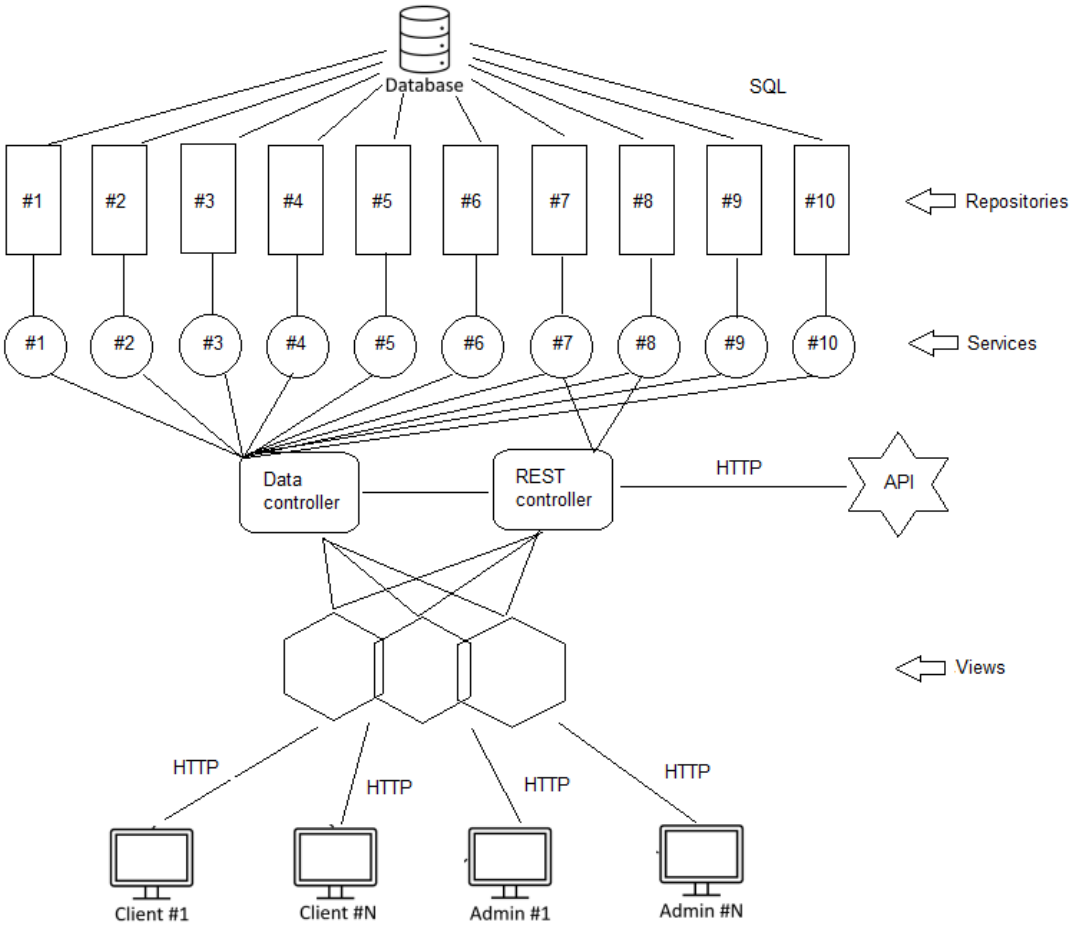

Figura 1 - Arquitetura da plataforma

Fica evidente pela imagem que os clientes comunicam via HTTP com as vistas que por sua vez vão comunicar com os controladores. Ambos os controladores comunicam com os serviços. Para cada serviço existe um repositório e cada repositório comunica com a base de dados.

É importante realçar que apenas aparecem três vistas por uma questão de comodidade ao ver a imagem. O *RESTcontroller* apenas está conectado a dois serviços uma vez que apenas necessita do serviço dos Jogadores e das Equipas. Esse controlador é o que nos permite importar equipas e jogadores de uma API externa. Já no outro controlador, é onde temos a maioria dos endpoints aos quais os clientes se vão conectar através das suas ações no frontend. Para facilitar a visualização da imagem, representamos cada serviço e cada repositório por um número.

De um modo geral, a nossa arquitetura de backend resume-se à imagem seguinte (UML).

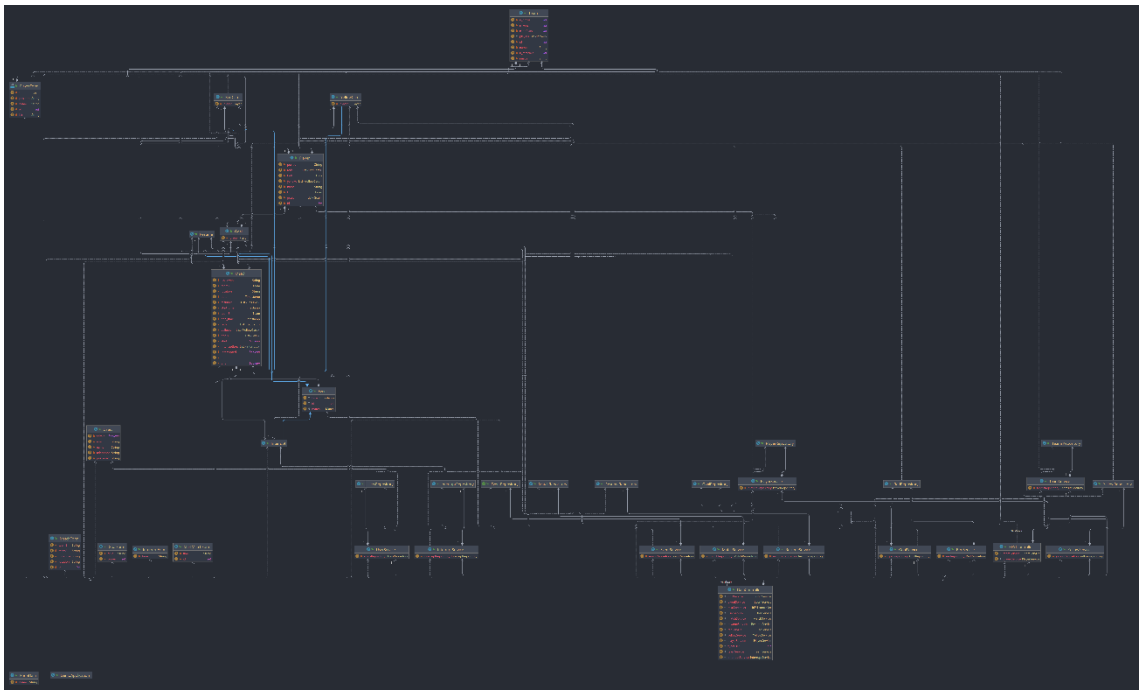

Figura 2 - Diagrama UML do backend

Em termos de frontend, a nossa plataforma contém várias páginas web. Estas relacionam-se entre si da seguinte maneira:

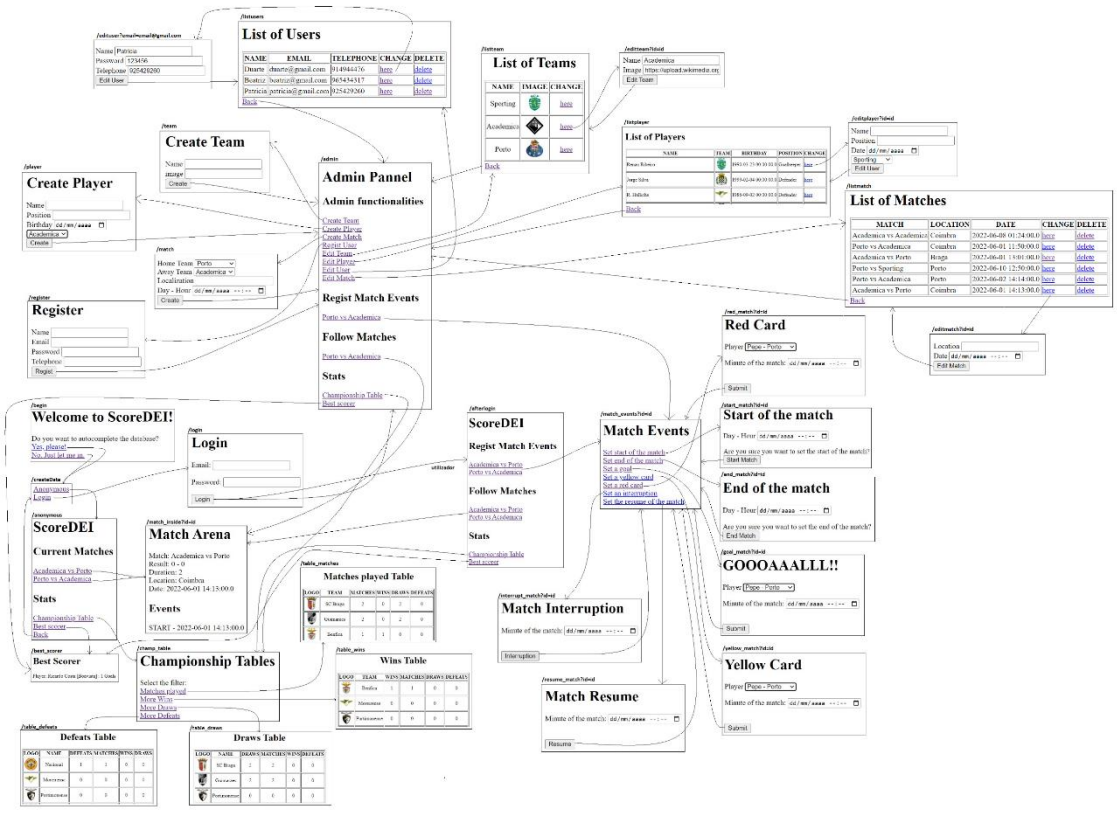

Figura 3 - Vistas e a sua interligação

Estas últimas duas imagens, para uma melhor visualização, estarão em anexo junto a este relatório.

Backend da plataforma

É aqui no *backend* que se realiza a ponte entre a base de dados e o *frontend* com o qual o cliente interage. Para ter tudo bem estruturado, optamos por ao início criar um modelo de dados de entidade-relacionamento para depois ser mais fácil organizar as classes. Para uma melhor visualização e compreensão do modelo por nós desenvolvido, segue em anexo a este relatório a imagem que apresentamos abaixo.

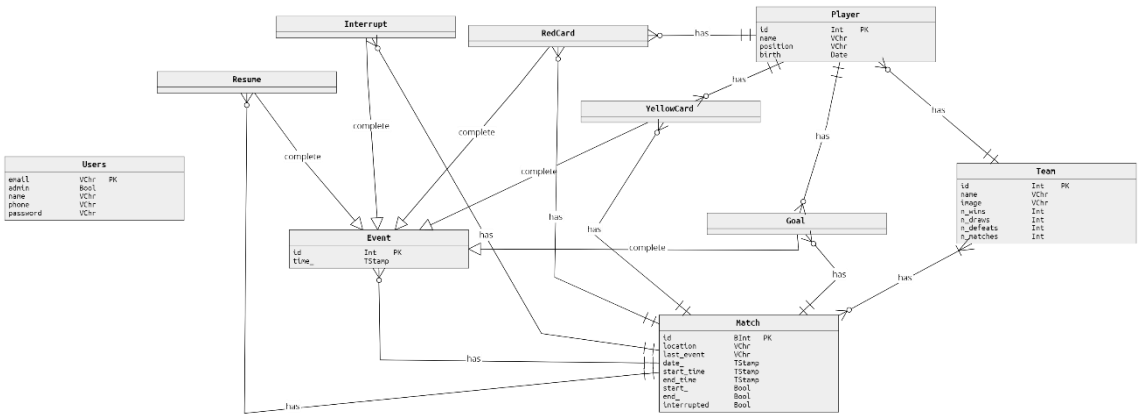

Figura 4 - Diagrama ER

Tal como podemos observar, o utilizador não tem ligação a nenhuma outra entidade. Isto acontece uma vez que o utilizador não se relaciona com outras classes. Apenas serve para estar registado na plataforma e conseguir aceder às funcionalidades correspondentes ao seu estatuto.

Já todas as outras classes estão interligadas. Optamos por criar uma superclasse "Event" e gerar a partir de hereditariedade todos os tipos de eventos. Isto permite distinguir o tipo de eventos e tê-los todos na mesma tabela para os poder selecionar por jogo e ordenar de forma cronológica mais facilmente.

De resto, tal como era espectável, uma equipa pode ter vários jogadores mas um jogador só pode pertencer a uma equipa. Um jogo tem duas equipas e pode ter todo o tipo de eventos mas cada evento só pode estar relacionado com um jogo em específico.

Dos atributos das entidades consideramos relevante esclarecer o significado do "last_event" no Match. Esta variável serve apenas para guardar no jogo a data e hora do último evento reportado naquele jogo. Isto fará com que não seja possível reportar eventos anteriores aos já reportados.

Tendo este modelo implementado no Java, começamos por desenvolver as diferentes funcionalidades da plataforma. Para isso, criamos um repositório e um serviço para cada entidade do diagrama ER acima. Com isto, conseguimos criar queries e aceder à base de dados o que nos possibilitou começar a programar as várias tarefas que a plataforma devia conseguir fazer.

Para cada uma delas, criamos um endpoint que, estando ligado a uma função, vai desenvolver um certo número de operações para realizar a funcionalidade pretendida. De seguida explicamos de forma sucinta como cada funcionalidade está implementada.

Criar utilizadores / equipas / jogadores / jogos

Para criar um novo elemento de cada uma destas categorias, deve-se aceder ao endpoint correspondente ("/register", "/team", "/player", "/match"). Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se é administrador). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página correspondente à criação do pretendido onde deve preencher com as informações necessárias. Essas informações, ao chegarem ao "POST" do endpoint vão ser verificadas. Se estiverem em conformidade com o pretendido, cria-se o novo objeto com esses dados e, através do serviço correspondente, guarda-se no repositório devido, sendo o cliente redirecionado para a página do administrador. Já se as informações não estiverem certas, o cliente é automaticamente redirecionado para a mesma página para voltar a introduzir os dados corretos. No caso de ser criação de jogadores ou jogos, a página da criação vai já apresentar as equipas que constam na base de dados (as que o jogador pode pertencer / o jogo pode ter). Isso acontece, pois, no "GET" do endpoint "/player" e "/match" é atribuído um modelo com todas as equipas disponíveis através de uma query do serviço de equipas.

Nota: Sendo que o login é efetuado através do email e password, não podem existir dois utilizadores com o mesmo email. Para garantir que isso acontece, verificamos sempre na altura da criação de utilizadores se já existe algum com o mesmo email. Só se cria um novo, se não existir.

Editar utilizadores / equipas / jogos / jogadores

Para editar um objeto, deve-se aceder ao endpoints correspondente ("/listusers", "/listteam", "/listplayer", "/listmatch"). Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se é administrador). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página correspondente e, ao clicar em editar, vai para a de edição do objeto correspondente onde deve preencher com as informações relativas a este. Os dados que o cliente preenche, ao chegarem ao "POST" do endpoint (chamado pelo do edit) vão ser verificados. Se estiverem em conformidade com o pretendido, edita-se o objeto e, através do serviço deste, guarda-se no repositório correspondente, sendo o cliente redirecionado para a página em que se listam todos os objetos daquela classe. Já se as informações não estiverem certas, o cliente é automaticamente redirecionado para essa página, mas o objeto não é editado.

Eliminar utilizadores / jogos

Para eliminar um utilizador ou um objeto, deve-se aceder ao endpoints correspondente ("/listusers", "/listmatch"). Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se é administrador). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página correspondente e, ao clicar em

eliminar, a função ligada ao endpoint (“/deleteuser”, “/deletematch”) vai eliminar o objeto. No caso de ser utilizador, é logo eliminado. Já no caso de ser jogo, vai-se verificar se este já começou. Se assim foi, não se pode eliminar o jogo. Após a tentativa de eliminação ou a eliminação mesmo, o cliente é redirecionado para a página em que se listam todos os objetos daquela classe.

Login de utilizadores

Para um utilizador se ligar à plataforma deve efetuar o login. Para isso, tem de aceder à página cujo endpoint é “/login” e preencher os campos (email e password). Os dados que o cliente preenche, ao chegarem ao “POST” do endpoint vão ser comparados com os dos utilizadores existentes na base de dados. Se existirem, o utilizador é redirecionado para a página de utilizador correspondente ao seu estatuto e uma variável que guarda o tipo de utilizador (explicado mais à frente) é atualizada. Caso contrário, é redirecionado para a página inicial da plataforma.

Criar eventos para um jogo

Quando existem jogos a decorrer ou futuros é apresentada uma lista destes aos utilizadores registados e ao administrador. Quando estes clicam para criar eventos num jogo, são redirecionados para o endpoint “/match_events?id={id}”, sendo o id dentro de chavetas o id do jogo em questão. Se um utilizador anónimo tentar aceder a este endpoint é automaticamente redirecionado para a página inicial. Se conseguir aceder a este endpoint, tem ao seu dispor vários links. Cada um leva-o a um tipo de evento diferente:

Início / Fim do jogo

Para definir qualquer um destes eventos basta clicar no link presente na página cujo endpoint é “/match_events?id={id}”. Cada link leva-nos a novos endpoints (“/start_match?id={id}”, “/end_match?id={id). Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se não é anónimo). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página pretendida e tem de introduzir a hora a que aquele evento ocorreu e clicar no botão de submeter. Caso a hora do evento seja posterior à última ocorrida, o serviço do jogo vai mudar o estado deste. Para começar o jogo, este ainda não pode ter começado e para acabar, este tem de já ter começado. Ao acabar o jogo, a função “POST” ligada ao endpoint “/end_match” vai atualizar o número de jogos realizados por ambas as equipas, calcular quem venceu/ perdeu/ empatou e atualizar também esses valores nas equipas correspondentes. Após a criação destes eventos, o cliente é redirecionado para a página correspondente ao seu estatuto enquanto utilizador.

Interrupção / Retoma do jogo

Para definir qualquer um destes eventos basta clicar no link presente na página cujo endpoint é “/match_events?id={id}”. Cada link leva-nos a novos endpoints (“/interrupt_match?id={id}”, “/resume_match?id={id}”). Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se não é anónimo). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página pretendida e, tem de introduzir a hora a que aquele evento

ocorreu e clicar no botão de submeter. Caso a hora do evento seja posterior à última ocorrida, o serviço do jogo vai mudar o estado deste. Para interromper o jogo, este tem de já ter começado e não estar interrompido. No caso de retomar, o jogo tem de já ter começado e tem de estar interrompido. Após a criação destes eventos, o cliente é redirecionado para a página correspondente ao seu estatuto enquanto utilizador.

Golo

Para definir que existiu um golo basta clicar no link correspondente presente na página cujo endpoint é "/match_events?id={id}". Esse link levar-nos-á a "/goal_match?id={id}". Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se não é anónimo). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página pretendida e, tem de escolher, entre os jogadores das duas equipas e que ainda não levaram vermelho, qual foi o autor do tento certeiro, tem de introduzir a hora a que aquele evento ocorreu e tem de clicar no botão de submeter. Caso a hora do evento seja posterior à última ocorrida, os serviços de golo e de jogo vão adicionar este golo. Após a criação do golo, o cliente é redirecionado para a página correspondente ao seu estatuto enquanto utilizador.

Cartão amarelo

Para definir que existiu um cartão amarelo basta clicar no link correspondente presente na página cujo endpoint é "/match_events?id={id}". Esse link levar-nos-á a "/yellow_match?id={id}". Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se não é anónimo). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página pretendida e, tem de escolher quem levou o amarelo, de entre os jogadores das duas equipas e que ainda não levaram vermelho, tem de introduzir a hora a que aquele evento ocorreu e tem de clicar no botão de submeter. Caso a hora do evento seja posterior à última ocorrida, o serviço do jogo vai adicionar este amarelo, bem como o serviço de cartão amarelo. Quando um jogador recebe uma cartolina amarela, a nossa plataforma vai automaticamente ver quantos amarelos aquele jogador já tem no jogo em questão. Se este amarelo tiver sido o segundo, a aplicação cria logo o vermelho e o jogador deixa de estar disponível para os próximos eventos desse jogo. Após a atribuição do amarelo, o cliente é redirecionado para a página correspondente ao seu estatuto enquanto utilizador.

Cartão vermelho

Para definir que existiu um cartão vermelho basta clicar no link correspondente presente na página cujo endpoint é "/match_events?id={id}". Esse link levar-nos-á a "/red_match?id={id}". Aqui é verificado se o cliente que está a tentar realizar esta operação tem permissões para o fazer (se não é anónimo). Se não tiver, é reencaminhado para a página inicial da aplicação. Já se tiver, é redirecionado para a página pretendida e, tem de escolher quem levou o vermelho, de entre os jogadores das duas equipas e que ainda não foram expulsos, tem de introduzir a hora a que aquele evento ocorreu e tem de clicar no botão de submeter. Caso a hora do evento seja posterior à última ocorrida, o serviço do jogo vai adicionar este vermelho, bem como o serviço de cartão vermelho. Após a atribuição da cartolina encarnada, o cliente é redirecionado para a página correspondente ao seu estatuto enquanto utilizador.

Nota: Qualquer evento que se adiciona vai atualizar o campo que guarda a hora do último desse jogo.

Acompanhar um jogo

Quando existem jogos a decorrer é apresentada uma lista destes aos utilizadores em qualquer uma das vistas de utilizador (anónimo, registado e administrador). Quando o cliente clica para acompanhar um jogo, é redirecionado para o endpoint "/match_inside?id={id}", sendo o id dentro de chavetas o id do jogo em questão. Na função deste endpoint, atribuímos vários modelos para estarem disponíveis no HTML para apresentar ao cliente. Para a informação do nome do jogo, da localização e da data, apenas precisamos do jogo em si, uma vez que este contém estas informações. No que diz respeito ao tempo de jogo, resultado e eventos já tem que existir um pequeno processamento antes. Para calcular o tempo de jogo, pegamos na hora atual e calculamos a diferença entre essa e a de início do jogo. Para o resultado, utilizamos o serviço dos golos para nos retornar uma *string* que contenha os golos daquelas duas equipas, naquele jogo em específico. Já para os eventos, vamos buscar através do serviço destes os eventos do jogo em questão. Tendo estas informações, o HTML já consegue mostrar o que se pretende.

Estatísticas

Era-nos pedido que a partir dos dados dos jogos, obtivéssemos algumas estatísticas. Estas estão disponíveis em qualquer uma das vistas de utilizador (anónimo, registado e administrador). Podemos ver listagens ordenáveis das equipas e o melhor marcador:

Listagens ordenáveis

Para ver as listagens ordenáveis das equipas basta clicar no link que diz "Championship Table", correspondente à página cujo endpoint é "/champ_table". Essa página apresenta quatro links, todos eles correspondentes a tabelas diferentes. Podemos ver listagens das equipas ordenadas pelo número de jogos realizados, pelo número de vitórias, empates e até de derrotas. Cada página recebe da função "GET" o *array* de equipas proveniente do serviço destas que já vem ordenado segundo o critério escolhido.

Melhor marcador

Para ver o melhor marcador basta clicar no link que diz "Best Scorer", correspondente à página cujo endpoint é "/best_scorer". Essa página recebe da função "GET" o jogador correspondente ao melhor marcador bem como o de número de golos marcados. Estes valores são obtidos através do serviço de golos que nos devolve inicialmente o número de golos de cada jogador ordenada em ordem decrescente. Com isto, a partir do primeiro valor, vamos procurar o jogador que tem aquele número de golos na sua conta.

Importação de dados da API externa

Sempre que se entender necessário, a nossa plataforma permite a importação de equipas e jogadores de uma API externa. Isso faz-se na página inicial cujo endpoint é "/begin" e basta clicar no link que diz "Yes, please!". Ao fazer isto, a função associada ao endpoint "/completeDB" (chamada pelo HTML anterior) vai chamar as funções *getTeams* e *getPlayers* do *RESTcontroller*. Nessas funções é estabelecida conexão à API e são obtidos em formato *json* os dados pretendidos. Posteriormente, esses dados são tratados e convertidos no tipo devido e são criados os objetos em causa.

Decidimos obter as informações da Primeira Liga do campeonato português da época 2020/2021. No caso das equipas, basta um acesso à API e obtemos todos os dados. Já para os jogadores, como estes são retornados num sistema paginado, optamos por fazer 10 acessos (obter 10 páginas) de jogadores da liga desse ano. Com isto, nem todas as equipas apresentam o mesmo número de jogadores.

Balanço das funcionalidades

De todas as funcionalidades, a que revelou ser mais desafiante foi a da importação de dados da API. Talvez por ser algo relativamente novo, tivemos de pesquisar bastante e ver tutoriais para perceber como o fazer.

No que diz respeito ao outro tipo de funcionalidades, a que custou mais a implementar foi a de apresentar os eventos do jogo por ordem cronológica. Num primeiro momento, as nossas classes não estavam preparadas para isso, uma vez que não tínhamos a superclasse evento onde constam todos. Como tínhamos os eventos separados em várias classes, juntá-los posteriormente para ordenar não se revelou uma tarefa fácil. Por essa razão, decidimos criar a superclasse "Event" em que basta obter os eventos de um determinado jogo ordenando pelo id e eles vêm já ordenados cronologicamente.

Acesso de utilizadores

Decidimos que uma forma simples e eficaz de controlar o que cada utilizador pode ou não aceder é ter uma variável global no código. Em cada função temos uma verificação do valor dessa variável. Se as ações apenas podem ser acedidas por administradores, a variável tem de estar a 2. Se podem ser realizadas por utilizadores registados, pode estar a tudo menos a 0. Se, por outro lado, as ações poderem ser feitas por todos, inclusive utilizadores não registados, não existe verificação.

O programa inicia sempre com a variável a 0. Caso haja um login e o utilizador não seja administrador, a variável passa a 1. Caso seja administrador, passa a 2.

Frontend da plataforma

Na parte da arquitetura da plataforma já apresentamos as vistas e a forma como elas se relacionam.

Nesta secção do relatório, consideramos importante realçar que as páginas que apresentam estatísticas (como as das listagens e do melhor marcador), as dos acompanhamentos dos jogos e as que apresentam as listas de jogos a decorrer e futuros atualizam em tempo real (de 20 em 20 segundos) para que o cliente tenha uma verdadeira experiência de acompanhamento em direto sem ter de atualizar a página.

Testes realizados à plataforma

Testagem de funcionalidades

	Integração com serviços externos
Resultado esperado	Os alunos deverão criar uma funcionalidade de inicialização de dados da plataforma scoreDEI, a qual faz uma sequência de pedidos HTTP à API REST referida, no sentido de obter dados de equipas de futebol e dos seus jogadores.
Resultado obtido	Inicialmente o utilizador pode escolher integrar dados externos e completar a base de dados ou simplesmente entrar com os dados que já lá estejam.
Veredito	Aprovado

	Registar utilizadores
Resultado esperado	Deve ser possível efetuar o registo de utilizadores na plataforma. Deverá ser necessário preencher a seguinte informação: nome, email, password e contacto telefónico.
Resultado obtido	O administrador é capaz de registar/criar um utilizador preenchendo todos os campos apresentados (nome, email, password e contacto telefónico)
Veredito	Aprovado

	Criar/Gerir Equipas
Resultado esperado	Um administrador deve conseguir criar equipas, cada equipa terá um nome e uma imagem associada.
Resultado obtido	Um administrador consegue criar equipas preenchendo todos os campos apresentados (nome e imagem).
Veredito	Aprovado

	Criar/Gerir Jogadores
Resultado esperado	Um administrador deve conseguir criar jogadores, cada jogador terá um nome próprio, posição em que joga (defesa, lateral, médio, etc…), data de nascimento e deverá ser associada uma equipa já existente no sistema.
Resultado obtido	Um administrador consegue criar jogadores preenchendo todos os campos apresentados (nome próprio, posição em que joga, data de nascimento e equipa). As equipas a que pode ser atribuído são apresentadas, estando estas presentes no sistema.
Veredito	Aprovado

	Criar/Gerir Jogos de Futebol
Resultado esperado	Deve ser possível aos administradores criar jogos de futebol. Cada jogo deverá contemplar uma Equipa A e uma Equipa B, que deverão ser selecionadas a partir da lista de equipas registadas no sistema, localização do evento e a data em que ocorrerá o evento assim como a hora de início.
Resultado obtido	Um administrador consegue criar jogos escolhendo duas equipas (as equipas que podem ser escolhidas são apresentadas, estando estas presentes no sistema), e preenchendo o resto dos campos apresentados (localização, data e hora de início).
Veredito	Aprovado

Registar evento num jogo	Início e Fim de Jogo (Apenas possível para utilizadores registados)
Resultado esperado	Sinalizar o início e o final da partida.
Resultado obtido	Ao selecionar um jogo é possível iniciar ou terminar um jogo colocando a data e hora do acontecimento.
Veredito	Aprovado

Registar evento num jogo	Golo (Apenas possível para utilizadores registados)
Resultado esperado	Selecionar um jogador e reportar um golo desse jogador, e respetiva equipa, com uma data e hora associada.
Resultado obtido	Ao selecionar um jogador é possível reportar um golo desse jogador, já estando a sua equipa associada a si, preenchendo a data e hora do acontecimento.
Veredito	Aprovado

Registar evento num jogo	Cartão Amarelo (Apenas possível para utilizadores registados)
Resultado esperado	Selecionar um jogador e reportar um cartão amarelo desse mesmo jogador, com uma data e hora associada.
Resultado obtido	Ao selecionar um jogador é possível reportar um cartão amarelo desse mesmo jogador preenchendo a data e hora do acontecimento.
Veredito	Aprovado

Registar evento num jogo	Cartão Vermelho (Apenas possível para utilizadores registados)
Resultado esperado	Selecionar um jogador e reportar um cartão vermelho desse mesmo jogador, com uma data e hora associada.
Resultado obtido	Ao selecionar um jogador é possível reportar um cartão vermelho desse mesmo jogador preenchendo a data e hora do acontecimento.
Veredito	Aprovado

Registar evento num jogo	Jogo interrompido (Apenas possível para utilizadores registados)
Resultado esperado	Indicar que o jogo se encontra interrompido.
Resultado obtido	É possível indicar que o jogo se encontra interrompido ao escolher tal opção.
Veredito	Aprovado

Registar evento num jogo	Jogo resumido (Apenas possível para utilizadores registados)
Resultado esperado	Indicar que o jogo volta a estar a decorrer (após ter estado interrompido).
Resultado obtido	É possível indicar que o jogo volta a estar a decorrer apenas após ter estado como interrompido.
Veredito	Aprovado

	Acompanhar jogo
Resultado esperado	Um utilizador, que pode ser não autenticado, deverá conseguir visualizar os detalhes de um jogo a decorrer, incluindo resultado atual, tempo de jogo e listagem de eventos até ao momento.
Resultado obtido	Um utilizador, que pode ou não estar autenticado, consegue visualizar os detalhes de um jogo a decorrer, incluindo resultado atual, tempo de jogo e listagem de eventos até ao momento.
Veredito	Aprovado

	Consultar estatísticas (Possível a um utilizador que pode ou não estar autenticado)
Resultado esperado	A vista de estatísticas deverá apresentar ao utilizador, pelo menos as seguintes estatísticas: -> Listagens ordenáveis das equipas com informação sobre jogos realizados, nomeadamente: número de jogos, número de vitórias, número de derrotas, número de empates. -> Nome do melhor marcador de golos.
Resultado obtido	Ao selecionar a opção de “Championship Table” é possível observar uma tabela com as listagens ordenáveis das equipas com informação sobre jogos realizados com número de jogos, número de vitórias, número de derrotas, número de empates. Ao selecionar a opção “Best Scorer” é possível observar o nome do melhor marcador de golos ao escolher tal opção e caso este exista.
Veredito	Aprovado

Testagem contra possíveis erros

	Login inválido
Resultado esperado	Um utilizador ou administrador deve entrar apenas se a informação preenchida estiver correta.
Resultado obtido	Ao tentar entrar com um mail não registado ou password errada o cliente é redirecionado para a página inicial de forma a tentar entrar de novo.
Veredito	Aprovado

	Melhor marcador
Resultado esperado	Deve ser possível selecionar a opção para ver o melhor jogador sem que este dê erro caso não haja informação a apresentar.
Resultado obtido	Caso não exista dados de golos, a plataforma não deixa entrar no “Best scorer” ficando na mesma em página em que se encontra.
Veredito	Aprovado

	Marcar golo
Resultado esperado	Caso não haja jogadores para marcar golo, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso não exista nenhum jogador de ambas as equipas para marcar um golo ou o jogo não tenha sido iniciado, o cliente é redirecionado para a sua página inicial após o login.
Veredito	Aprovado

	Dar cartão amarelo
Resultado esperado	Caso não haja jogadores a quem dar o cartão ou o jogo não tenha sido iniciado, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso não exista nenhum jogador de ambas as equipas a quem dar um cartão amarelo ou o jogo não tenha sido iniciado, o cliente é redirecionado para a sua página inicial após o login.
Veredito	Aprovado

	Dar cartão vermelho
Resultado esperado	Caso não haja jogadores a quem dar o cartão ou o jogo não tenha sido iniciado, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso não exista nenhum jogador de ambas as equipas a quem dar um cartão vermelho ou o jogo não tenha sido iniciado, o cliente é redirecionado para a sua página inicial após o login.
Veredito	Aprovado

	Interrupção
Resultado esperado	Caso o jogo não tenha sido iniciado, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso se tente fazer uma interrupção, mas o jogo ainda não tenha começado, o utilizador/administrador é redirecionado para a sua página inicial após o login e o evento não fica guardado.

Veredito	Aprovado

	Resumo do jogo
Resultado esperado	Caso o jogo não tenha sido iniciado ou interrompido previamente, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso se tente resumir o jogo sem que este tenha sido interrompido ou não tenha sido iniciado, o utilizador/administrador é redirecionado para a sua página inicial após o login e o evento não fica guardado.
Veredito	Aprovado

	Terminar um jogo
Resultado esperado	Caso o jogo não tenha sido iniciado, não deverá ser possível que tal evento aconteça.
Resultado obtido	Caso se tente terminar o jogo sem que este tenha sido iniciado, o utilizador/administrador é redirecionado para a sua página inicial após o login e o evento não fica guardado.
Veredito	Aprovado

	Dados insuficientes para criação de um evento
Resultado esperado	Só deverá ser possível criar um evento se todos os dados forem preenchidos devidamente.
Resultado obtido	Caso os campos necessários não sejam preenchidos na criação de um evento o utilizador/administrador é redirecionado para a sua página inicial após o login e o evento não fica guardado.
Veredito	Aprovado

	Dados insuficientes para registo de um utilizador e criação de equipas, jogadores e jogos
Resultado esperado	Só deverá ser possível registar um utilizador e criar equipas, jogadores e jogos se todos os dados forem preenchidos devidamente.
Resultado obtido	Se os campos necessários não forem preenchidos, é feito um *refresh* à página de forma a representar que os dados estão em falta. O administrador deverá colocar de novo os dados de forma correta.
Veredito	Aprovado

	Dados insuficientes para edição de utilizadores, jogadores, equipas e jogos
Resultado esperado	Só deverá ser possível editar utilizadores, jogadores, equipas e jogos se todos os dados forem preenchidos devidamente.
Resultado obtido	Ao tentar editar, caso não sejam colocados todos os campos necessários é possível fazer "*Edit*" sendo redirecionada para a lista correspondente sem qualquer alteração.
Veredito	Aprovado

	Cartões
Resultado esperado	Um jogador que tenha levado cartão vermelho num jogo não deverá ser selecionado para mais nenhum evento desse mesmo jogo.
Resultado obtido	Após um jogador levar com um segundo cartão amarelo é lhe dado um cartão vermelho. Após levar com este ou mesmo com vermelho direto deixa de poder ser selecionado em qualquer evento possível naquele jogo.
Veredito	Aprovado

Conclusão

Hoje em dia, a informação instantânea é cada vez mais necessária a desejada. Foi com isso em mente que partimos para este trabalho prático.

Era-nos pedido que desenvolvêssemos uma plataforma capaz de fornecer acompanhamento em direto de resultados desportivos (mais concretamente de futebol), alimentada por informações de utilizadores registados.

Num modo geral, pensamos ter cumprido o objetivo. Conseguimos, com mais dificuldade numas coisas que noutras, implementar tudo o que era requerido, pelo que consideramos que o trabalho prático correu bem na generalidade.

Num trabalho futuro, era interessante embelezar o *frontend* e reforçar a robustez do *backend*, embora neste momento a plataforma seja completamente funcional.

Referências

- Material fornecido pelos docentes [acedido em 01/06/2022]
- http://www.columbia.edu/itc/visualarts/r4100/html_example_pages/l_tables/9_tables_align_img.htm [acedido em 01/06/2022]
- https://www.computerhope.com/issues/ch001968.htm [acedido em 01/06/2022]
- https://www.w3schools.com/tags/tag_center.asp [acedido em 01/06/2022]
- https://rapidapi.com/blog/how-to-use-an-api-with-java/ [acedido em 31/05/2022]
- https://api-sports.io/documentation/football/v3 [acedido em 31/05/2022]
- https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html [acedido em 31/05/2022]
- https://www.javatpoint.com/java-string-to-date [acedido em 31/05/2022]
- https://www.javatpoint.com/java-get-current-date [acedido em 31/05/2021]
- https://stackoverflow.com/questions/16119421/thymeleaf-concatenation-could-not-parse-as-expression [acedido em 30/05/2022]
- https://stackoverflow.com/questions/16119421/thymeleaf-concatenation-could-not-parse-as-expression [acedido em 30/05/2022]
- https://www.delftstack.com/howto/java/how-to-sort-objects-in-arraylist-by-date-in-java/ [acedido em 30/05/2022]
- https://www.geeksforgeeks.org/java-program-to-sort-objects-in-arraylist-by-date/ [acedido em 30/05/2022]
- https://www.geeksforgeeks.org/optional-get-method-in-java-with-examples/ [acedido em 30/05/2022]
- https://www.baeldung.com/java-optional [acedido em 30/05/2022]
- https://stackoverflow.com/questions/50306445/java-optional-containing-an-array [acedido em 30/05/2022]
- https://www.javatpoint.com/java-timestamp-compareto-method [acedido em 29/05/2022]
- https://stackoverflow.com/questions/45279009/failed-to-convert-value-of-type-java-lang-string-to-required-type-int-for-i [acedido em 29/05/2022]
- https://www.w3schools.com/html/html_form_input_types.asp [acedido em 29/05/2022]
- https://pt.stackoverflow.com/questions/518547/erro-nullpointerexception-is-null [acedido em 29/05/2022]
- https://stackoverflow.com/questions/46640402/how-add-new-line-after-each-iteration-in-thymeleaf [acedido em 29/05/2022]
- https://www.baeldung.com/thymeleaf-iteration [acedido em 29/05/2022]
- https://www.baeldung.com/thymeleaf-select-option [acedido em 28/05/2022]
- https://stackoverflow.com/questions/44879987/fill-option-on-html-with-java-array-list [acedido em 28/05/2022]
- https://stackoverflow.com/questions/22928426/how-to-display-each-element-of-a-javascript-array-as-a-link [acedido em 28/05/2022]
- https://thewebdev.info/2022/01/17/how-to-set-a-javascript-array-as-options-for-a-select-element/ [acedido em 28/05/2022]
- https://www.w3schools.com/html/html_links.asp [acedido em 28/05/2022]
- https://www.baeldung.com/spring-data-derived-queries [acedido em 27/05/2022]
- https://www.javatpoint.com/password-hide-in-html [acedido em 27/05/2022]

- https://www.w3schools.com/tags/tag_select.asp [acedido em 27/05/2022]
- https://www.w3schools.com/howto/howto_css_login_form.asp [acedido em 26/05/2022]
- https://stackoverflow.com/questions/29191043/how-to-handle-a-request-with-multiple-parameters-on-spring-mvc [acedido em 26/05/2022]
- https://betterprogramming.pub/spring-security-basic-login-form-7c8f6e6e9f56 [acedido em 26/05/2022]
- https://hellokoding.com/spring-security-login-logout-thymeleaf/ [acedido em 26/05/2022]
- https://www.devmedia.com.br/sistema-de-autenticacao-web-utilizando-jpa/32128 [acedido em 26/05/2022]
- https://www.devmedia.com.br/java-web-criando-uma-tela-de-login-com-jpa-jsf-primefaces-e-mysql/32456 [acedido em 26/05/2022]
- https://stackoverflow.com/questions/2002993/jpa-getsingleresult-or-null [acedido em 26/05/2022]
- https://www.tabnine.com/code/java/methods/javax.persistence.Query/getResultList [acedido em 26/05/2022]
- https://spring.io/guides/gs/handling-form-submission/ [acedido em 26/05/2022]
- https://www.ictdemy.com/html-css/first-website/lists-in-html-and-a-table-example [acedido em 26/05/2022]