Sofia Reia fc59875
Diogo Sousa fc59792
Renan Silva fc59802

Relatório CSS: Fase 2

Arquitetura do Projeto:

Em relação à arquitetura do projeto, houve poucos elementos que sofreram alterações relativamente à primeira fase. Por causa dos filtros, foi preciso alterar os cartões de um enumerado para uma entidade, dado que é preciso saber quem atribuiu um cartão.

Para além disso, foi adicionado um nome a um Tournament e houve alguns atributos booleanos que passaram a enumerados pois apenas dois estados não eram suficientes (por exemplo, estado de um jogo, que passou a ser, criado, fechado e cancelado).

Para além disso, foram adicionados todos os UCs relativos aos torneios, e tentámos abstrair ao máximo o tipo do Tournament para ser agnóstico e suportar no futuro a adição de outros tipos de torneio, mas em alguns casos muito específicos (por exemplo na interface gráfica) por simplicidade foi feito cast para um PointTournament.

Para os filtros, inicialmente tínhamo-los implementados na camada de negócio, mas procurámos alternativas para a segunda fase, dado que era preciso adicionar mais filtros. Encontrámos as Specifications do Spring JPA que facilitavam bastante a composição de filtros, no entanto a falta de type safety fez com que procurássemos alternativas.

Utilizámos o QueryDSL, que para além de ter uma linguagem muito mais natural e fluida do que as Specifications, tem vantagens como a adição de type safety que tornaram a escrita destes filtros muito mais simples.

Escolha e justificação das decisões técnicas no desenho da interface Web:

Para a interface Web, responsável pela implementação dos filtros para as principais entidades, além das funcionalidade de login e de registo de resultados de jogos, optámos por uma organização de páginas simples com o necessário para implementar todos os casos de uso. Ao todo, são nove páginas navegáveis: uma para login, uma para o homepage, duas para o processo de registo de resultado de um jogo e cinco para os casos de uso de buscas e filtros das entidades principais.

No backend, foi implementado um @Controller que serve o cliente com vistas renderizadas no lado do servidor através da ferramenta Thymeleaf.

O utilizador, ao aceder à raíz do website (no caso, localhost:8080), é automaticamente redirecionado para página que corresponde ao caminho "/login", de modo a simular um sistema de autenticação, onde poderá autenticar-se com as credenciais: Username -> “admin”, Password -> “admin”. O endpoint Rest responsável pela autenticação trata do pedido e, em caso de sucesso, o utilizador é automaticamente direcionado para o caminho "/home" onde pode visualizar todos os botões correspondentes a todas as páginas disponíveis na aplicação.

O primeiro botão, na secção "Actions", com nome "Register Game Result" leva o utilizador para a página (em "/register-game-result") onde pode visualizar todos os jogos que podem ter o seu resultado registado.

Ao clicar no botão "Register" de um jogo, o utilizador será direcionado para a página onde poderá registar o seu resultado. O registo é feito por blocos, onde cada bloco encapsula informação relativa a um jogador (que o utilizador escolhe registar), que pode ter golos e cartões. O utilizador regista apenas o que for necessário (ou seja, todos os eventos que aconteceram no jogo, para cada jogador) e pode adicionar blocos para quantos jogadores quiser.

Ao clicar em "Submit", e se a operação tiver sido bem sucedida (request ao endpoint Rest responsável pelo registo de um jogo), aparecerá um "alert" em que poderá clicar em "Ok" para ser direcionado para a página anterior (com a tabela de todos os jogos em aberto).

Os outros botões correspondem às páginas de buscas/filtros das entidades principais da aplicação. Ao clicar em qualquer um deles, será direcionado para a página onde poderá visualizar uma tabela com as instâncias registadas da entidade escolhida. Por questões de simplificação e considerando que, no nosso exemplo, não haverão muitos registos, a tabela apresentará todas as instâncias registadas (apesar da lógica de negócio suportar paginação).

Acima da tabela, há um conjunto de campos (não obrigatórios) que podem ser preenchidos e cada um deles representa uma forma de filtrar os dados. Os tipos de dados podem ser: texto, número ou dropdown/select, facilmente identificáveis ao interagir com a interface.

Os filtros podem ser combinados e depois basta clicar no botão "Filter" para que seja requisitada a view, onde os filtros vão como "query string" no URL do pedido e é devolvida a vista com os dados filtrados.

Escolha e justificação das decisões técnicas no desenho da interface JavaFX:

Na interface JavaFX, optámos por uma abordagem simples e concisa aos casos de uso.

Inicialmente, é apresentado um ecrã de Login (UC A). O acesso é possível utilizando as credenciais User -> “admin”, Password -> “admin”.

Daí, somos encaminhados para a Home Screen, que contém botões para aceder às listas de entidades: Players, Referees, Teams, Games e Tournaments.

Clicando em cada um desses botões, passamos para um ecrã com uma tabela na qual estão todas as instâncias da respetiva entidade. Nesse mesmo ecrã, podemos também adicionar instâncias, editar instâncias já existentes, fazer refresh da tabela, e, em certos casos, remover instâncias.

Para facilitar as interações entre entidades, sempre que era necessário adicionar, por exemplo, um Game a um Tournament, colocámos duas tabelas, uma de Entidades já associadas (ou seja, no exemplo, jogos do torneio) e outra de Entidades não associadas (no mesmo exemplo, outros jogos).

Um detalhe que vale a pena salientar sobre a aplicação nativa é o uso do okHTTP como alternativa ao HttpClient do Java. Utilizámos esta API ao invés do uso da API padrão do Java pela fluidez que a mesma apresenta assim como um encapsulamento nos tipos de retorno, facilitando a interação com a API Rest.

Players:

Para os jogadores é possível (UCs B, C):

- Registar um novo jogador dado o seu nome e posição de jogo preferida;
- Editar o nome, posição de jogo preferida e equipas de um jogador existente;
- Fazer Refresh;
- Remover um jogador.

Referees:

Para os árbitro é possível (UCs B, C):

- Registar um novo árbitro, dado seu o nome e a posse de certificado;
- Editar o nome e a posse de certificado de um árbitro existente;
- Fazer Refresh;
- Remover um árbitro.

Teams:

Para as equipas é possível (UCs E, F):

- Registar uma nova equipa, dado o seu nome;
- Editar o nome e jogadores de uma equipa existente;
- Fazer Refresh;
- Remover uma equipa.

Games:

Para os jogos é possível (UCs H, I):

- Criar um novo jogo, dadas as equipas e os seus jogadores, o Referee primário (e Referees secundários opcionais), um Address (apenas o campo Street é obrigatório) e uma data;
- Registar os resultados de um jogo -> a partir do registo dos Stats dos jogadores desse jogo, no menu para editar;
- Fazer Refresh;
- Remover um jogo.

Nos jogos, optámos por assumir que o utilizador os cria corretamente, fazendo apenas verificações no backend, dados alguns problemas com recursão infinita nos Observers ao tentar alterar de forma dinâmica os jogadores e equipas disponíveis no frontend.

Tournaments:

Para os campeonatos é possível (UCs J, M, L):

- Criar um novo campeonato, dado o seu nome;
- Editar um campeonato existente -> é feito em duas fases, caso o seu estado seja OPEN, podem ser adicionadas e removidas equipas, caso o seu estado seja IN_PROGRESS, é possível adicionar, remover ou cancelar jogos;
- Fazer Refresh;
- Remover um campeonato.

Para representar as tabelas, utilizámos o componente TableView. Para os outros componentes, utilizámos AnchorPane, HBox, VBox, ToolBar, Button, TextField e ComboBox para os dropdowns. Tentámos utilizar ids (fx:id) explicativos para facilitar a integração com os Controllers.

Visualmente, optámos por usar o prefHeight/prefWidth de modo a que os componentes tivessem um tamanho default adequado, mas não fossem muito restritos a mudanças nas janelas. Adicionámos também separadores e alterámos o tamanho do texto mais relevante para melhorar a visibilidade e os menus mais complicados (como o de criar um jogo) serem mais percetíveis. A fonte utilizada foi a JetBrains Mono, mas caso esta não esteja instalada é utilizada a default do JavaFX.