1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Sistemas Distribuídos

Relatório do Projeto - Meta 2

scoreDEI: Resultados desportivos em direto

2021/2022

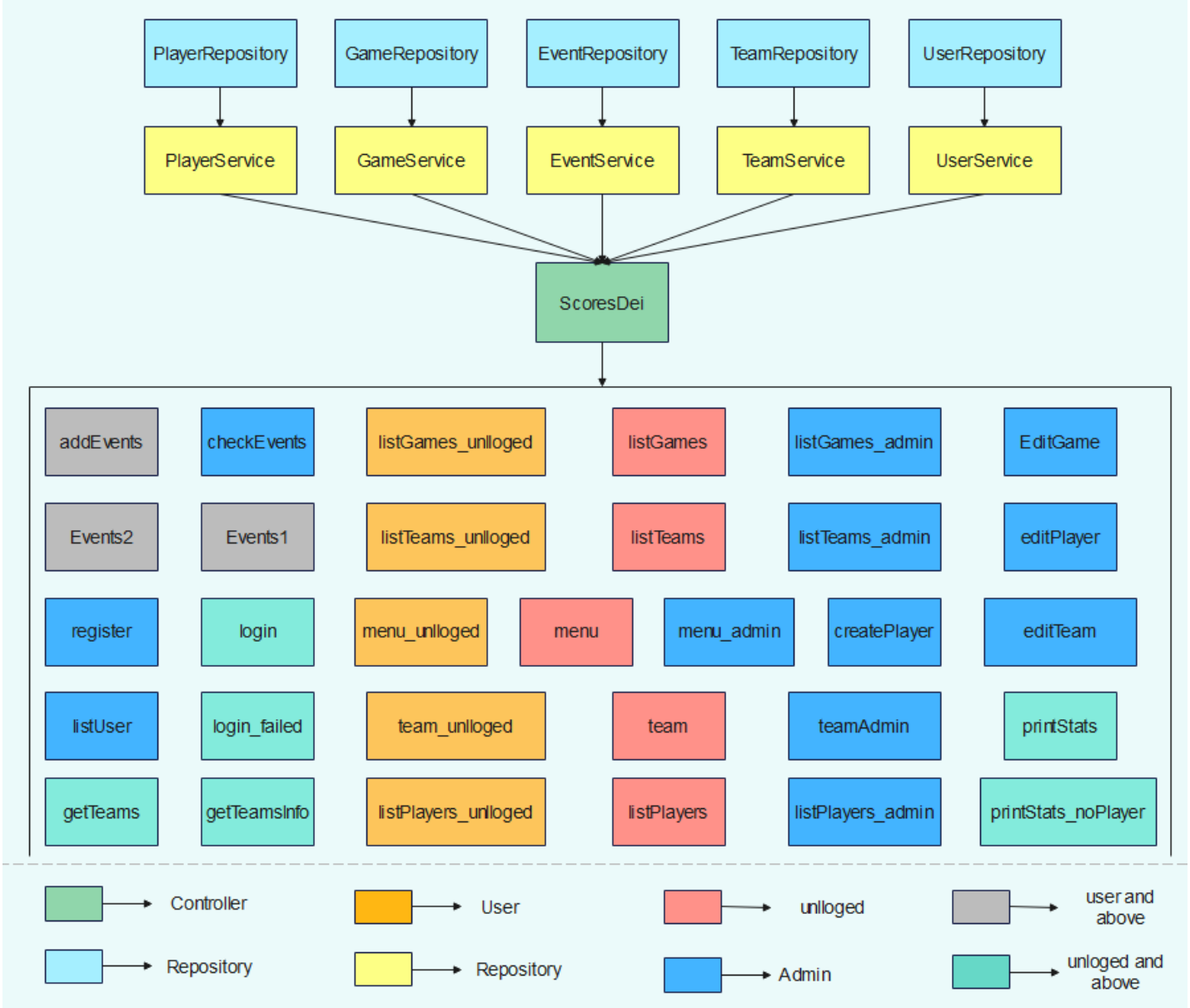
Licenciatura em Engenharia Informática

PL4 Davide Figueiredo Areias 2019219422 uc2019219422@student.uc.pt
PL4 Francisco Faria 2019227649 uc20192276492@student.uc.pt
PL4 Iago Bebiano 2019219478 uc2019219478@student.uc.pt

Índice

1. Arquitetura do Software

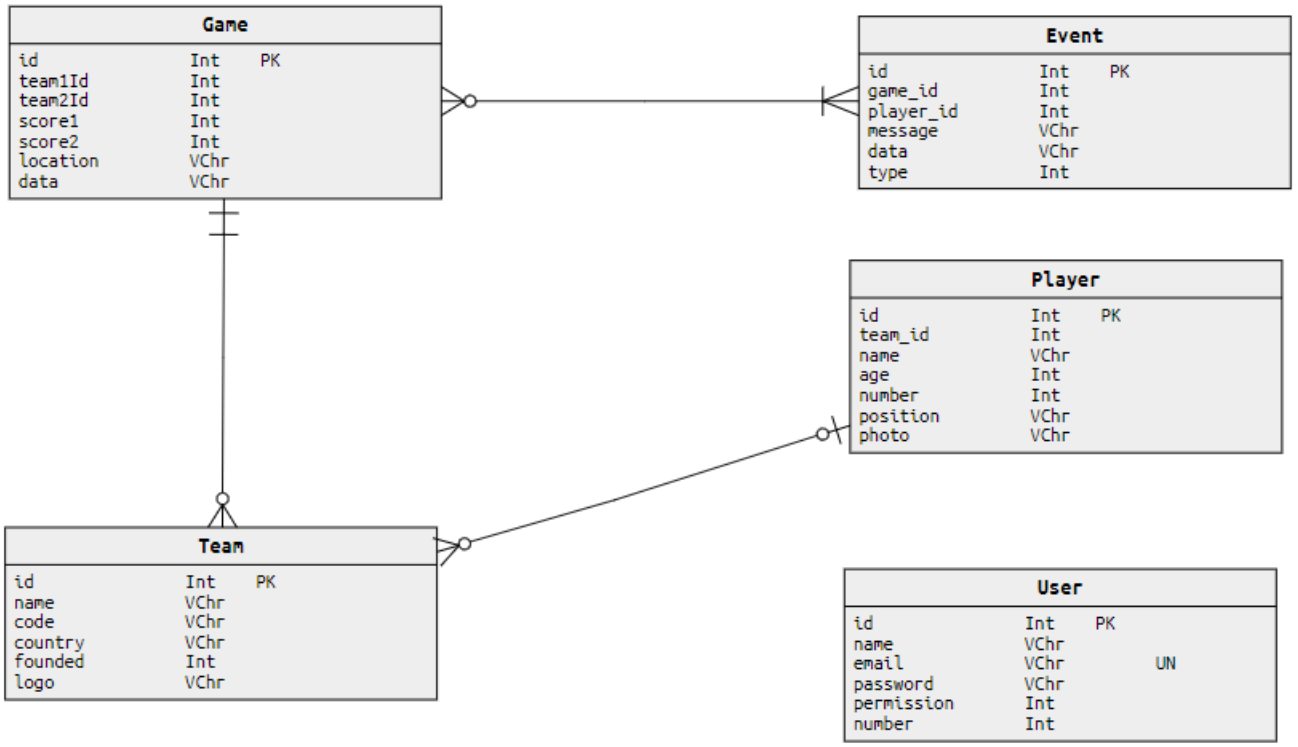
A arquitetura seguida foi a disponibilizada no enunciado, encontrando-se abaixo uma versão mais detalhada da mesma.



Optámos por um esquema de cores em vez de termos diversas conexões entre as várias views e os utilizadores ,visto que devido ao elevado número de views este iria ficar confuso.

1.1. Organização da Base de dados

Acima encontra-se a base de dados utilizada ao longo do nosso projeto criada



usando métodos JPA.

Importante mencionar que embora não tenhamos nenhuma relação entre o evento e o jogador o evento possui a variável player id através do qual pode encontrar o jogador associado se este existir.

1.2. Organização do Código

A nosso código está organizado da seguinte forma:

- Pasta data : local onde todas as classes de java se encontram;
- Demo/repository : todos os repositiries para as várias entities;
- Demo/services : todos os servicespara as várias entities;
- DataController : código correspondente ao controller;
- resources/templates : local onde se encontram todos os html usados separados por diversos folders.

2. BackEnd

2.1. Acessos a base de dados

Todos os nossos repositories são *crud* isto significa que já possuem diversas funcionalidades úteis no controlo da base de dados por exemplo:

- findAll():
  - que devolve uma lista com todos os elementos da base de dados;
- delete():
  - elimina o elemento dado como parâmetro da base de dados;
- findById():
  - recebe como parâmetro o id e dá retorno ao objeto que corresponde a esse id;
- save():
  - altera ou adiciona o elemento dado como parâmetro à base de dados.

Todas estas funcionalidades foram utilizadas indiretamente através de funções presentes nos serviços.

2.2. Permissões

A implementação das nossas permissões foi feita através de uma variável na session à qual demos o nome de *counter*. Para controle desta variável criámos duas funções: a setPermissions(int i) que define a variável e a checkPermissions() que dá retorno ao valor da mesma ou inicializa-a caso ainda não esteja inicializada.

Nós utilizamos o setPermissions() quando o user faz login definindo o counter com o valor de permissões que o mesmo tem.

Controlamos o acesso a users não só disponibilizando nos interface os links a que o mesmo tem acesso mas também direcionando o utilizador para a home sempre que este não tem permissões para o link a que está a tentar aceder.

No contexto do nosso trabalho counter igual a 0 é unloged, 1 é user e 2 é admin.

2.3. Login

Para a implementação do login tivemos de criar uma nova classe Login.java esta apenas tem como parâmetros o username e a password, é este objecto que é enviado para dentro do html e depois usado para validação.

Para procurar pelo user na base de dados tivemos de criar um sql que procura por Users com email igual ao dado por parâmetro comparando posteriormente a palavra passe se ambas forem as corretas altera-se o valor da variável counter no session e vai-se para a página de login.

2.4. Dados API

É criado um cliente que faz request das equipas da liga portuguesa de futebol, no caso estamos a optar por salvar apenas 4 equipas pois existe um limite de requests. De seguida, por cada equipa vamos fazer outro request dos jogadores da mesma e posteriormente salvar os dados na base de dados.

2.5. Criação de entidades

Em geral para a criação de todas as entidades usa-se sempre a mesma lógica, envia-se o objecto desejado para o html e depois de preenchido é submetido, adiciona-se a base de dados utilizando o save dado pelo Crud.

Apenas no caso dos players é que à uma pequena diferença visto que optamos por apenas adicionar estes dentro da página dos team details deste modo associando o player à team.

2.6. Edição das entidades

Mesma lógica que na criação mas o objecto que é enviado para o html é o que se pretende editar, de forma a obter este objecto é necessário fazer uma pesquisa por id usando a função mencionada anteriormente findById().

2.7. Lista das entidades

Neste caso não iremos enviar apenas um objecto para dentro do html mas sim uma lista obtida usando o findAll() mencionado anteriormente. Para casos em que só se pretende listar eventos específicos, por exemplo, no team details estes já estão associados ao objecto dado o Team (neste caso) ou seja não é necessário fazer a pesquisa, simplesmente, usar o método get das classes.

2.8. Estatísticas

Esta foi a etapa em que tivemos mais dificuldades, não só na criação dos diversos queries , visto que já a algum tempo que não trabalhamos com os mesmos, mas também de diversos bugs que ocorreram. Em geral todos os bugs que apareciam foram resolvidos após adicionar o parâmetro *nativeQuery* = *true* à chamada do query.

Ainda encontrámos outro erro e este estava relacionado com o nome dado ao objeto disponibilizado ao html sendo que este não podia ter o nome param.

Outra dificuldade que nós tivemos foi como ordenar os resultados obtidos, embora neste ponto já tivéssemos as diferentes queries com vários *order by* definidos, não sabíamos como disponibilizar ao utilizador ordenamentos tanto no sentido ascendente como descendente, após discutirmos o assunto durante algum tempo optámos por adicionar um

parâmetro extra a função este corresponde ao valor pelo qual os dados estavam ordenados, se este valor corresponde-se ao que se pretende ordenar então a ordem é invertida, se o valor do parâmetro for maior que 10 então os dados encontravam-se por ordem inversa ou seja volta-se a colocar por ordem ascendente.

3. Testes

Teste	admin	user	unloged	Link
Create User	☑	----------	----------	http://localhost:8080/register
Create Team	☑	----------	----------	http://localhost:8080/createTeam
Create Player	☑	----------	----------	http://localhost:8080/createPlayer?id=212
Create Game	☑	----------	----------	http://localhost:8080/createGame
Create Event	☑	☑	----------	http://localhost:8080/addEvent?id=2
Log in	☑	☑	☑	http://localhost:8080/login
Log out	☑	☑	----------	http://localhost:8080/logout
Statistics All	☑	☑	☑	http://localhost:8080/printStats?old_d=1&d=0
Statistics two teams	☑	☑	☑	http://localhost:8080/compTeams
List User	☑	----------	----------	http://localhost:8080/listUser
List Team	☑	☑	☑	http://localhost:8080/listTeams
List Player	☑	☑	☑	http://localhost:8080/listPlayers
List Game	☑	☑	☑	http://localhost:8080/listGames
List Event	☑	☑	☑	http://localhost:8080/gameDetails?id=2
Team details	☑	☑	☑	http://localhost:8080/team?id=211
Game details	☑	☑	☑	http://localhost:8080/gameDetails?id=2
Edit Team	☑	----------	----------	http://localhost:8080/editTeam?id=212

Edit Player	☑	----------	----------	http://localhost:8080/editPlayer?id=556
Check Events	☑	----------	----------	http://localhost:8080/checkEvents?id=2
Update	☑	☑	☑	http://localhost:8080/update

4. Conclusão

Neste trabalho obtivemos novas capacidades no que toca a construção de websites e manipulação de bases de dados em java. Concluímos que começar por definir as diversas classes inicialmente e só após isto começar os outros elementos de trabalho permitiu um trabalho mais fluido e eficaz, tal como uma maior facilidade de acesso à informação.

Embora tenham surgido diversos problemas ao longo da realização deste projeto, podemos afirmar que entregamos o nosso site satisfeitos e com novos conhecimentos, não só em a java mas também em html, que ficaram intrínsecos nas nossas mentes e prontos a ser usados no nosso futuro profissional.