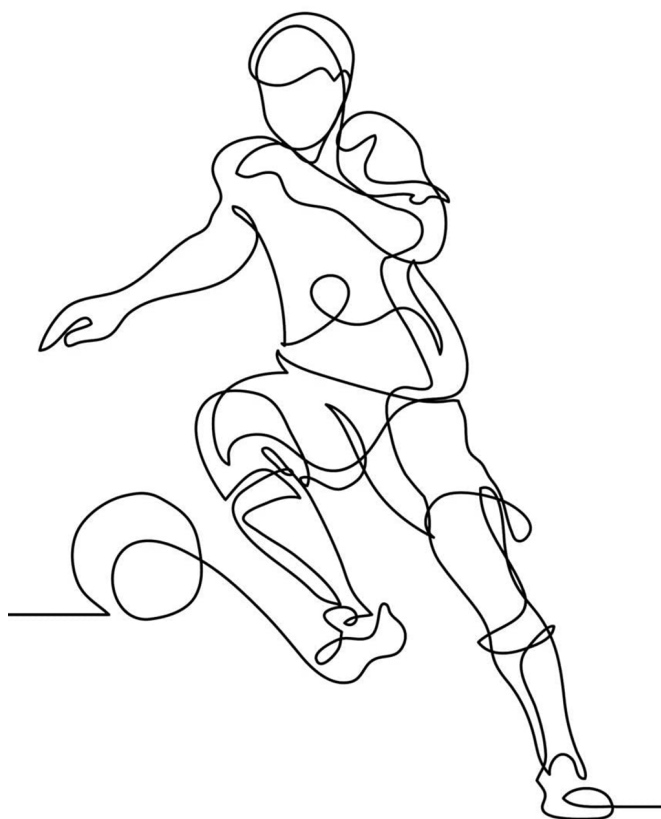




Sistemas Distribuídos 2021/22

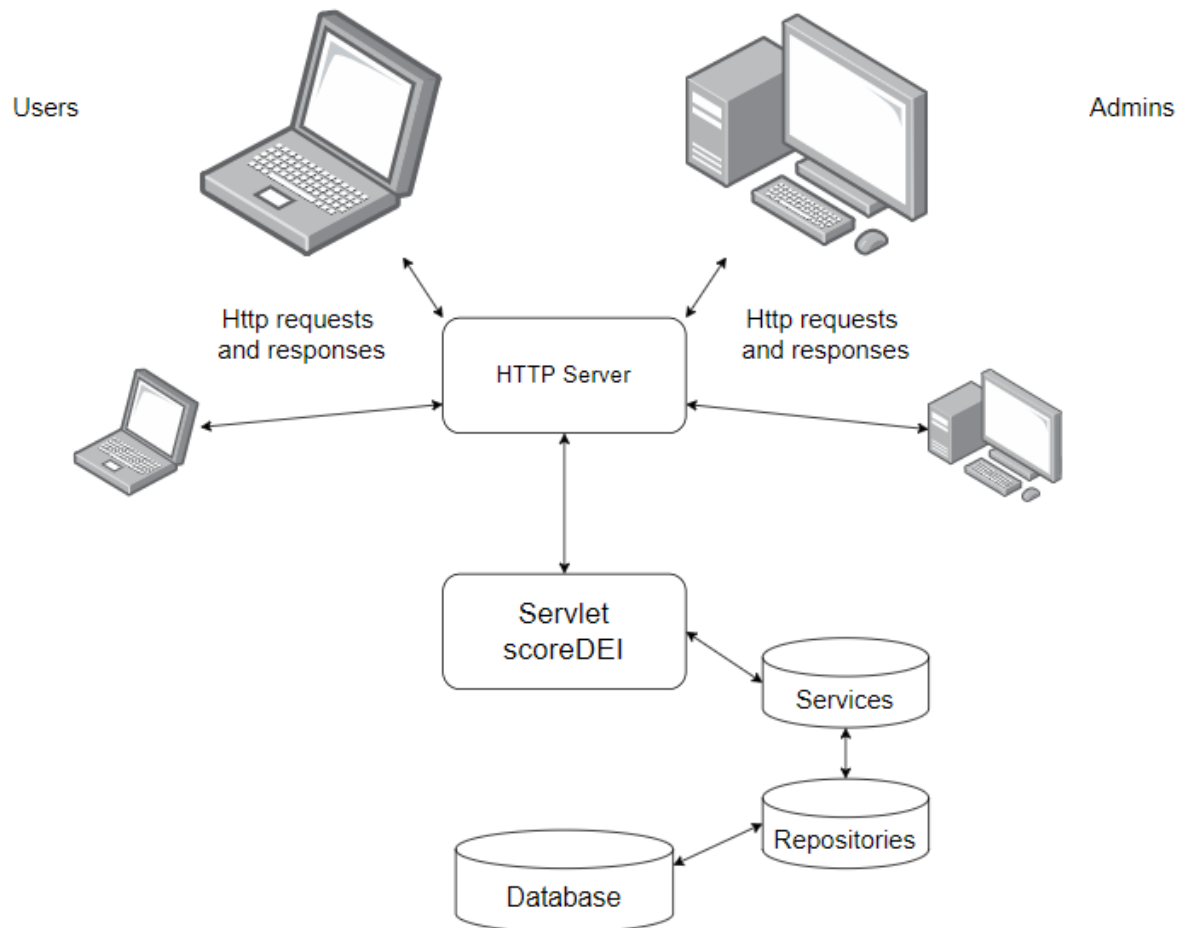
Trabalho prático “scoreDEI”

Projeto 2



Ana Beatriz Marques
2018274233, anafm@student.dei.uc.pt
Bárbara Gonçalves
2018295452, bsfg@student.uc.pt

Arquitetura



Ferramentas usadas

- Spring Boot: Java framework
- Thymeleaf: Java template para html
- Bootstrap: Formatação de html, css e javascript
- PostgreSQL: ferramenta de gestão de databases

Organização

- Data: Entities
- App: Controller, Services e aplicação
- Repository: repositories das entidades
- Form: auxiliares de formatação

Funcionamento Arquitetura

Usando as ferramentas acima referidas, realizamos a arquitetura de forma a ter um servidor disponível para diversos utilizadores simultaneamente. Segue um design pattern frontend e backend, sendo que o frontend representa o código que é executado no browser do utilizador, e o backend representa o conjunto de serviços e código que lidam com o tratamento de dados da aplicação.

Após conectar ao servidor através de *http://localhost:8080*, Spring Boot consegue controlar e redirecionar o utilizador para diferentes páginas com *@Controller* *DataController*. Conecta-se, portanto, à camada de visualização (HTML) e apresenta-os ao utilizador. Esta classe (*DataController*) contém o mapeamento da aplicação, sendo assim a camada de negócio. Tem também a capacidade de gerir e obter informação da camada de dados (*@Repository*), através de *@Service*. Mais detalhadamente, *@Repository* utiliza o princípio CRUD (create, read, update and delete) pelo que executa queries SQL nas tabelas presentes na database.

Um exemplo concreto é o login *http://localhost:8080/confirmLogin*:

O *DataController* recebe o *@ModelAttribute FormUser*, que contém o email e password necessários para autenticação. Chama depois o *userService*, que por sua vez chama o *userRepository* que executa uma query na Database e, caso as credenciais sejam válidas, retorna um utilizador da tabela. Se o login não for válido, o utilizador é avisado e redirecionado para uma página de erro.

Funcionamento Geral

A aplicação inicia, sendo possível fazer login ou ver os *jogosatuais(1)*.

Ao fazer login, entramos então no *menuUser(2)* ou *menuAdmin(3)*, consoante o utilizador seja um admin ou não. Seja qual for o menu, é sempre possível aceder às estatísticas.

1- Jogos Atuais: Permite os utilizadores não registados verem a lista de jogos a decorrer e os seus respectivos eventos. Podem também aceder às estatísticas.

2- Menu User: Os utilizadores que tenham feito login podem ver os jogos atuais e futuros, adicionar eventos nos mesmos, e ver os eventos de jogos que estão a decorrer.

3- Menu Admin: Contém as funcionalidades de admin, sendo estas adicionar, editar e listar users, jogadores, jogos e equipas.



Testes

Teste	pass/fail	notas
Organização dos Endpoints em Controladores	pass	Todos os endpoints da aplicação estão em DataController, a única excepção é os endpoints do SPORTS API
Criação, gestão de utilizadores e login	pass	
Criação e gestão de equipas	pass	A criação de equipa é feita unicamente através do SPORTS API
Criação e gestão de jogadores	pass	A criação de jogador é feita através do SPORTS API sendo editado depois a posição e a equipa a qual pertence.
Criação e gestão de jogos	pass	
Criação de eventos para um jogo	pass	Proteções de incompatibilidade de eventos realizada (se já começou não pode ser iniciado outra vez, etc)
Visualizar detalhes de um jogo	pass	
Estatísticas 3. a)	pass	
Estatísticas 3. b)	pass	
Importação de jogadores de API externa (só LEI)	pass	
Escolha de entidades para representar os dados e as suas relações e propriedades	pass	
Estruturação de código a nível de services e repositories	pass	
Consultas a dados através de queries JPQL	pass	
Tratamento de excepções	pass	