Relatório do segundo trabalho prático de Tecnologias Web

Henrique Rosa - 51923
Diogo Matos - 54466
16/01/2024

Introdução

Este trabalho foi desenvolvido como resposta ao enunciado do segundo trabalho prático da unidade curricular de Tecnologias Web do curso de engenharia informática. Consiste no desenho e implementação de uma aplicação web para gerir eventos de corrida.
A nossa aplicação inclui as seguintes funcionalidades:

- Uma divisão do site por perfis, com:
  → Uma parte pública, sem necessidade de login, onde é possível pesquisar eventos por nome ou data, participantes inscritos num evento e as suas classificações em determinado ponto;
  → Uma parte reservada ao perfil "ATLETA" onde, para além de fazer tudo o que se faz na parte pública, também é possível realizar inscrições em eventos, bem como consultar e fazer o pagamento das mesmas;
  → Uma parte reservada ao perfil "STAFF" onde, para além de fazer tudo o que se faz na parte pública, também é possível realizar registos de eventos e tempos de participantes nos eventos.

A solução foi desenvolvida em Java no no IDE IntelliJ IDEA da JetBrains.

Solução Implementada

De modo a encontrar uma solução adequada e satisfatória, fizemos primeiro uma análise dos conteúdos lecionados nas aulas, de modo a perceber que tecnologias e ferramentas nos seriam mais úteis para esta função. Acabamos por decidir que o serviço iria ser implementado seguindo as seguintes configurações:

- Solicitações utilizando Rest;
- Autenticação com Spring JPA;
- Acessos à base de dados PostgreSQL por meio de JDBC;
- Ficheiros JSP com conteúdo Web;
- Arquitetura MVC;

Classes

A nossa solução dispõe das seguintes classes:

- DataBaseConfig → Responsável pelas configurações da base de dados;
- SecurityConfig → Lida com as configurações de segurança da webapp, incluindo o tipo de autenticação e as permissões de acesso;
- SpringSecurityFormBasedJdbcUserDetailsServiceAuthApp → Inicializa a WebApp;
- PostGresConnect → Facilita a conexão e operações em uma base de dados PostgreSQL usando JDBC;
- Handler → Realiza pedidos e inserções na base de dados, além de processar os resultados desses pedidos;
- SpringSecurityController → Atua como o controlador principal da webapp, gerenciando os pedidos da interface, adicionando atributos ao modelo conforme necessário.
- Event → Classe que representa um modelo relacionado a um evento.
- User → Classe que representa um modelo relacionado a um user.
- UserRowMapper → Classe que mapeia registos de um ResultSet.
- UserAuthService → Classe responsável por carregar as informações do user durante o processo de autenticação.

Páginas JSP

- confirmations → Página onde são mostradas confirmações das operações realizadas
- error → Página de erro, quando um utilizador tenta o acesso a uma página para o qual ele não tem permissão.
- index → Página inicial da webapp.
- inscricoes → Página onde são mostradas todas as inscrições feitas por um user.
- joinevent → Página onde é registada uma inscrição num evento por um formulário.
- login → Página de login do site.
- newevent → Página onde é registado um novo evento.
- newinscricao → Página onde é mostrado ao "Atleta" eventos onde se pode inscrever.
- newtime → Página onde é registado o tempo de um atleta para um dado evento.
- newuser → Página onde é feito o registo de um utilizador.
- paymentevent → Página onde é feito o pagamento de um evento.
- registertime → Página onde é mostrado eventos para se efetuar o registo do tempo.
- submitinscricao → Página onde é mostrado o valor a pagar depois de se efetuar a inscrição num evento.

Base de Dados

Para o correto funcionamento da nossa solução, fizemos uso de uma BD PostGres, com as seguintes tabelas nela presentes:

- users → Contem informações sobre cada utilizador no sistema.
- events → Contem a informação sobre um evento, apenas os dados deste.
- user_roles → Contem o role de cada user.
- registrations → Contem cada uma das inscrições em cada um dos eventos.
- participant_times → Contem os tempos registados para cada Atleta em dado evento

Descrição do funcionamento do programa

Os seguintes passos numerados apresentam o funcionamento do sistema:

1. Criar uma BD PostGres com as tabelas descritas em cima.
2. Alterar config.properties e application.properties consoante a BD criada.
3. Compilar o projeto num IDE (como o IntelliJ) ou no terminal com o comando mvn compile.
4. Executar o projeto no IDEA ou então utilizando o comando "mvn spring-boot:run" num terminal.
5. Entrar no link http://localhost:8080 para aceder à página.

Balanço crítico

A nossa solução ao enunciado **cumpre** todos seguintes requerimentos do mesmo, embora tenhamos algumas falhas a apontar na nossa implementação:

- Alguns pedidos são feitos em **GET** em vez de **POST**, o que pode levar a alguns problemas de segurança.
- A autorização, embora funcional, não foi implementada da forma correta.
- Visualmente, a aplicação está um pouco "crua", pois não nos foi possível arranjar a parte visual do serviço tão bem quanto queríamos.