Running Events - Trabalho 2 Sistemas Distribuídos

Autores:
Miguel Horta - nº42731
Vasco Barnabé - nº42819

Janeiro de 2022

1 Introdução

Neste trabalho foi solicitada a realização de um sistema com arquitetura distribuída, um sistema de registo de inscrições em eventos e cronometragem dos tempos de corrida.

Este sistema é cosntituido por 3 módulos:

- **Aplicação Cliente** - para que o utilizador possa registar e consultar eventos, efetuar inscrições em eventos e obter classificações dos mesmos, finais ou parciais;
- **Aplicação Sensor** - aplicação que transmite a leitura da passagem de um dado participante por determinado ponto num dado evento, para o Backend/Servidor (Aplicação Serviços), transmitindo um *chipID*, o tempo exato de passagem (um timestamp) e o local;
- **Aplicação Serviços** - Aplicação Backend/Servidor, onde se encontram todos os serviços disponíveis às aplicações cliente e sensor. Esta aplicação é responsável pelas funcionalidades e armazenamento.

2 Implementação

2.1 Aplicação Cliente

Operações disponíveis:

- **Obter a lista de eventos numa data:** o cliente obtém os eventos que se realizam na data que inserir;
- **Registar novo evento:** o cliente pode registar um novo evento introduzindo o nome para o novo evento, o seu tipo (atletismo de pista, atletismo de estrada ou trail) e a data em que este se irá realizar;
- **Lista de inscritos num evento:** o cliente obtém uma lista com os participantes inscritos num evento específico;

- **Situação de atletas em corrida:** o cliente pode consultar o número de participantes que já atingiram determinado ponto na prova;
- **Classificação:** o cliente pode obter a classificação final ou parcial de um evento, consoante pretenda verificar as classificações dos participantes que já terminaram a prova, ou a classificação num determinado ponto intermédio da prova;
- **Inscrição de Participante:** o cliente regista um novo participante num evento existente.

Nota: Na aplicação Serviços, todas as operações foram implementadas com a segurança de retornar alguma informação de erro ao cliente sempre que algum dado seja inserido incorretamente. Por exemplo, se o cliente tentar consultar eventos numa determinada data e não existirem eventos a realizar-se nessa data, o sistema retorna uma mensagem sugestiva ao cliente do que aconteceu; sempre que o cliente insira um ponto intermédio que não existe; etc...

2.2 Aplicação Sensor

Operações disponíveis:

- Transmitir Leitura: transmite ao backend/servidor uma leitura, simulando a passagem de um participante por um certo ponto de um determinado evento.

A leitura envia ao servidor um chipID, um timestamp e um locationID. LocationID é a identificação do ponto intermédio (ou inicial ou final) por onde o participante passou; o timestamp é o registo do tempo exato em que o participante passou aquele ponto, para que sejam efetuados cálculos precisos do tempo de prova de cada participante e para que este possa ser mapeado ao longo do prova; por fim, o chipID, que se trata de um identificador único para aquele participante naquela prova. O servidor é responsável por, através do chipID, identificar o participante e em qual prova este está a participar, para que se possa registar a leitura.

Este chipID tem o seguinte formato(exemplo): "E1P3". Este exemplo identifica o participante com o dorsal nº3 a participar no evento com o id=1.

Todos estes dados são inseridos pelo utilizador do sensor.

2.3 Aplicação Serviços

Como já foi referido, esta é a aplicação responsável por todas as funcionalidades do sistema e pelo armazenamento.

Foram criados dois objetos, objeto **evento** e objeto **participante**, de modo a implementar todas as funcionalidades pedidas. Esta aplicação gere, consoante os pedidos que sejam efetuados, tanto para obtenção como para alteração de dados, todos os eventos e participantes neles inscritos.

Estes pedidos são realizados através do **Protocolo HTTP**, onde foi utilizado o método **GET** para qualquer pedido que apenas solicitasse a obtenção de dados, e o método **POST** para qualquer pedido que exigisse a alteração ou criação de dados.

Disponibiliza uma API REST para a Aplicação Sensores (um serviço), para que esta possa efetuar registar as leituras que efetua relativas às passagens dos participantes em cada ponto do evento.

Classes:

- **Event** - Entidade, implementação do objeto evento;
- **EventController** - Controlador que direciona cada pedido relacionado com eventos ao serviço específico para a resolução desse pedido;
- **EventService** - Implementação dos serviços que retornam ao utilizador os eventos que se realizam numa determinada data; a lista dos participantes inscritos num determinado evento; as classificações de cada evento; e ainda o registo de novos eventos;
- **EventRepository** - Repositório onde se encontram definidos métodos que permitem obter informação sobre os eventos da Base de Dados.
- **Participant** - Entidade, implementação do objeto participante;
- **ParticipantController** - Controlador que direciona cada pedido relacionado com participantes ao serviço específico para a resolução desse pedido;
- **ParticipantService** - Implementação do serviço que retorna ao utilizador o número de participantes que já passaram por determinado ponto em determinado evento, que regista a inscrição de um novo participante, e que regista uma nova leitura de um sensor;
- **ParticipantRepository** - Repositório onde se encontram definidos métodos que permitem obter informação sobre os participantes da Base de Dados.

3 Armazenamento

Para o armazenamento de dados deste serviço, foi utilizada uma Base de Dados em **PostgreSQL**. Os dados encontram-se divididos em duas tabelas:

- **table_event** - tabela onde se encontram registados todos os eventos existentes, juntamente com os seus tipos e datas de realização;
- **participants_table** - tabela que regista todos os participantes inscritos nos eventos, os seus chips identificadores, os seus tempos de prova e todas as restantes informações pessoais.

4 Execução do Projeto

4.1 Requisitos

- Java
- Apache Maven
- PostgreSQL

Executar as 3 aplicações em terminais separados:

4.2 Aplicação Cliente

1. Aceder à pasta `Client_Module`
2. Executar :`mvn compile`
3. Executar: `mvn spring-boot:run`

4.3 Aplicação Sensor

1. Aceder à pasta `Sensor_Module`
2. Executar :`mvn compile`
3. Executar: `mvn spring-boot:run`

4.4 Aplicação Services

1. Aceder à pasta `Services`
2. Executar :`mvn compile`
3. Executar: `mvn spring-boot:run`

5 Conclusão

Com a realização deste trabalho, foi criado um **Sistema de eventos** que permite inscrições em eventos e cronometragem dos tempos de corrida, entre outras funcionalidades. Foi-nos possível esclarecer de melhor forma o funcionamento de uma **REST API** com framework **Spring**, construindo uma aplicação executável e utilizando outras ferramentas como **PostgreSQL** como sistema gerenciador de banco de dados, e **Apache Maven** como ferramenta de automação de compilação.

Este sistema foi implementado com o máximo de atenção no que toca a tolerância a falhas, no sentido de suportar erros da parte do utilizador, e retribuir com uma resposta sugestiva, para que o utilizador tenha uma maior facilidade em interagir de forma correta com a aplicação.

É importante referir ainda que esta foi mais uma oportunidade de construir um projeto com uma dimensão considerável na linguagem **Java**, consolidando e aumentando os nossos conhecimentos sobre a linguagem, bem como sobre todas as restantes tecnologias e ferramentas utilizadas.

Para concluir, resta apenas referir que todas as funcionalidades pedidas para este programa foram implementadas e se encontram funcionais, à exceção das funcionalidades com valorização superior referidas no enunciado.