Running Events - Trabalho 1 Sistemas Distribuídos

Autores:
Miguel Horta - nº42731
Vasco Barnabé - nº42819

Novembro de 2021

1 Introdução

Neste trabalho foi solicitada a implementação de aplicações **servidor** e **cliente** que permitam, desde vários pontos do país, o acesso ao serviço. Este serviço deve permitir o registo de eventos e de participantes nesses eventos, entre outras funcionalidades.

2 Implementação

2.1 RunningEventsServer

A classe `RunningEventsServer` realiza, neste serviço, o papel de servidor, criando a ligação ao RMI e à Base de Dados, usando as credenciais de acesso a esta (credenciais fornecidas num ficheiro do tipo **properties**).

2.2 RunningEventsClient

A classe `RunningEventsClient` é o cliente utilizado como ferramenta de interação que o utilizador usa para usufruir do serviço fornecido. Serviços do servidor disponíveis para o cliente:

- Registar um participante num determinado evento já existente;
- Consultar eventos numa determinada data;
- Registar um evento;
- Listar os participantes inscritos num evento;
- Registar o tempo de prova de um participante;
- Obter a classificação geral de um evento, seja esta absoluta ou por género (masculino e feminino);
- Obter o Top3 de cada evento num determinado escalão e género.

2.3 RunningEventsImpl

A classe `RunningEventsImpl` contém a implementação dos serviços apresentados anteriormente, métodos que podem ser remotamente acedidos pelo cliente. É nesta classe que é realizada o tratamento dos dados e a interação do servidor com a base de dados, onde são realizadas consultas e atualizações da mesma.

2.4 PostgresConnector

A classe `PostgresConnector` é responsável por realizar a conexão à base de dados **PostgreSQL**.

2.5 RunningEvents

A interface `RunningEvents` contém definidos os métodos que podem ser remotamente solicitados pelo cliente, para executar as operações na base de dados (através do servidor).

2.6 Client_Request e Server_Answer

Classes criadas para facilitar o transporte de dados entre **Cliente** e **Servidor**. Os dados que o cliente necessita de enviar para o servidor são enviados num objeto `Client\_Request` e os dados que o Servidor retorna para o cliente são enviados num objeto `Server_Answer`. Assim, informações como os dorsais dos participantes, os seus nomes, os nomes dos eventos, etc...são facilmente transportados num objeto, evitando assim a passagem de vários argumentos em cada invocação de método remoto.

3 Execução

1. **importar** o ficheiro `tables.sql` para a base de dados;
2. **alterar** as credenciais de acesso à base de dados no ficheiro `credentials.properties`, que se encontra na pasta `resources`;
3. **compilar** todas as classes `RunningEventsClient` e `RunningEventsServer` com os comandos "javac -d build/classes -classpath build/classes src/t1/RunningEventsClient.java"e "javac -d build/classes -classpath build/classes src/t1/RunningEventsServer.java";
4. **executar** o **RMIRegistry** com o comando "rmiregistry -J-classpath -Jbuild/classes port";
5. **iniciar** o servidor, num novo terminal, com o comando "java -classpath build/classes:resources/* t1.RunningEventsServer port";
6. **iniciar** o cliente, num novo terminal, com o comando "java -classpath build/classes:resources/* t1.RunningEventsClient localhost port";

4 Conclusão

Como conclusão para este trabalho, foi-nos possível construir uma aplicação cliente-servidor, podendo esta servir como um serviço para uma empresa de gestão de eventos desportivos.

Foi uma oportunidade de utilização de **RMI**, bases de dados **PostgreSQL** e **serialização de dados**, utilizando os conhecimentos adquiridos nas aulas desta disciplina, Sistemas Distribuídos.