2021/2022
João Rouxinol Nº44451
André Rato Nº45517

Running Events – v1.0

Introdução

No âmbito da disciplina de Sistemas Distribuídos, foi proposta a realização de um trabalho prático que consiste num serviço de registo de eventos e participantes em vários tipos de corridas. Foi pedida a implementação de um cliente e servidor que permitam a realização de funções como:

- Inscrever participantes num evento;
- Registar novos eventos;
- Registar tempo de um participante num evento;
- Consultar classificações nos eventos;

É também pedido que a serviço criado utilize uma solução de Middleware estudada nas aulas e que seja compatível com uma BD em Postgresql, de modo a ser compatível com a plataforma de "alunos.di.uevora.pt".

Desenvolvimento e Solução Encontrada

De modo a desenvolver um serviço que cumpra todas as funcionalidades pedidas foi criado um programa na linguagem Java e foi utilizado Java RMI (invocação remota de métodos) como solução de Middleware. Para realizar conexões, inserções e consultas à BD foi utilizado JBDC (Java Database Connectivity).

Classes:

As Classes criadas foram as seguintes:

- Inscricao – Interface Remota;
- InscricaoImpl – Objeto Remoto utilizado para invocação remota, implementa os métodos da interface remota Inscricao, cada objeto InscricaoImpl representa uma conexão à BD, quer seja para uma consulta, inscrição, registo de tempo, etc;
- PostGresConnect – Classe responsável pela conexão a uma BD Postgresql e com os métodos necessários para inserções e consultas às BD;
- Server – Classe que representa o servidor do serviço;
- Client – Classe que representa o cliente do serviço;

Métodos:

Os métodos utilizados no serviço são os seguintes:

Classe InscricaoImpl:

- convertToHhMmSs(int time) – Converte tempo em segundos para o formato HH:MM:SS;
- novaInscriçao(String s) – Envia um pedido ao método acessDatabase() referente à inscrição de um novo participante;
- getProva(String d) – Envia um pedido ao método acessDatabase() referente à consulta das provas numa data;
- getInscritos(String d) – Envia um pedido ao método acessDatabase() referente à consulta dos inscritos numa prova;

- novoTempo(String s) – Envia um pedido ao método acessDatabase() referente ao registo de um novo tempo numa prova;
- getClassificacoes(String s) – Envia um pedido ao método acessDatabase() referente à consulta das classificações numa prova;
- getTop(String s) – Envia um pedido ao método acessDatabase() referente à consulta do Top 3 numa determinada prova;
- acessDatabase(int op, String s) – Cria um objeto PostGresConnect de modo a realizar uma conexão à BD para realizar uma operação na BD passada como argumnto op, que pode ser uma consulta ou inserção;

Classe PostGresConnect:

- connect() - Estabelece uma conexão à BD;
- disconnect() - Termina a conexão à BD;
- getStatement() - Devolve o Statement, para ser usado para criar as Queries Postgresql;

Classe Server:

- main() - Coloca o servidor online na porta especificada no ficheiro config.properties e dá Bind ao objeto remoto;

Classe Client:

- genreChecker(String genre) – Verifica se o género é correto (m/f);
- convertToSeconds(String time) – Converte o tempo em formato HH:MM:SS para um inteiro, que representa o tempo em segundos;
- tierPicker(int choice) – Retorna o escalão correspondente à opção passada no argumento choice;
- menu() - Imprime o menu de opções e lê a opção escolhida pelo utilizador;
- main() - Conecta-se ao servidor e procura o objeto remoto, é também responsável pela leitura de todas os dados necessários para a interação com a BD (como leitura de nomes de eventos, tempos de prova ou escalões);

Tabelas BD:

Para a representação deste sistema, foi utilizada uma BD PostGres, nela foram criadas as seguintes tabelas:

- inscricoes – representa as várias inscrições de participantes em eventos, e inclui o numero de dorsal, único para cada participante num evento;
- eventos – representa os vários eventos do serviço, inclui também a data do evento, bem como o tipo do mesmo
- tempos – representa os vários tempos feitos por participantes nas várias provas;

As Queries utilizadas para a criação das tabelas na BD para o correto funcionamento do serviço são as seguintes:

Inscrições:

```
CREATE TABLE inscricoes(
        nomeparticipante varchar(255),
        nomeevento varchar(255),
        genero char(1),
        escalao varchar(20),
```

```
    dorsal int,
    PRIMARY KEY (nomeparticipante, nomeevento, dorsal),
    FOREIGN KEY (nomeevento) REFERENCES eventos
);
```

Eventos:

```
CREATE TABLE eventos(
    nomeevento varchar(255) PRIMARY KEY,
    dataevento date,
    tipo varchar(255)
);
```

Tempos:

```
CREATE TABLE tempos(
    nomeevento varchar(255),
    dorsal int,
    genero char(1),
    escalao varchar(20),
    tempo int,
    PRIMARY KEY (nomeevento, dorsal),
    FOREIGN KEY (nomeevento) REFERENCES eventos
);
```

Instruções de Utilização

De modo a utilizar este serviço é necessário fazer os seguintes passos:

1. Criar uma BD Postgresql com as tabelas necessárias (descritas acima);
2. Compilar todas as classes do serviço (Inscricao, InscricaoImpl, PostGresConnect, Server e Client);
    - Feito executando o comando “javac -d build/classes -classpath build/classes src/t1_sd/*classe a compilar*.java” num terminal na pasta base do projeto;
3. Executar o comando “rmiregistry -J-classpath -Jbuild/classes XXXX” onde XXXX será o numero de um porto disponível;
4. Alterar, no ficheiro config.properties, os valores de:
    - **db** para o nome da base de dados;
    - **host** para o endereço onde está a base de dados;
    - **user** para o nome de utilizador;
    - **password** para a password de acesso;
    - **regHost** para o endereço onde está o servidor;
    - **regPort** para a porta onde o cliente se deve conectar;
    - **regPortServer** para a porta onde o servidor se deve conectar;
5. Executar o servidor;
    - Feito executando o comando “java -classpath build/classes:lib/postgrestql-42.3.2.jar t1_sd.Server” num terminal na pasta base do projeto;
6. Executar o cliente;
    - Feito executando o comando “java -classpath build/classes:lib/postgrestql-42.3.2.jar t1_sd.Client” num terminal na pasta base do projeto;