Sistemas Distribuídos:

Relatório do primeiro trabalho prático

Henrique Rosa 51923 | Diogo Matos 54466

08-12-2023

Introdução:

O presente relatório foi realizado no âmbito da unidade curricular Sistemas Distribuídos integrante na Licenciatura em Engenharia Informática da Universidade de Évora, Colégio Luís António Verney.

O documento aborda de forma sucinta, todos os passos para a nossa implementação de um sistema para gerir atuações (performances) dos artistas de rua do Alentejo onde é permitido registar artistas e localizá-los quando estão a fazer as suas atuações. Tem se três aplicações que permitem o acesso ao serviço e às suas ações, sendo esses o Server, ClientAdmin e ClientGeneral. O serviço utiliza uma das soluções de Middleware estudadas nas aulas, sendo escolhido o Java RMI, de modo a ser compatível com a plataforma "alunos.di.uevora.pt".

Classes:

- Server: representa o servidor do serviço;
- PostGresCon: faz a conexão a uma BD PostGresSQL e com os métodos necessários para a consulta e inserção na BD. Esta classe foi retirada das aulas práticas, esta desenvolvida pelo professor José Saias;
- ClientAdmin: representa o cliente administrador do serviço, dando acesso às ações possíveis com chamadas dos métodos remotos;
- ClientGeneral: representa o cliente geral do serviço, dando acesso às ações possíveis com chamadas dos métodos remotos;
- ClientAdmImpl: objeto remoto usado para realizar chamadas remotas. Ele implementa os métodos definidos na interface remota Remote_ClientAdm. Cada instância de ClientAdmImpl está associada a uma conexão à BD, permitindo a execução de consultas;
- ClientGenImpl: objeto remoto usado para realizar chamadas remotas. Ele implementa os métodos definidos na interface remota Remote_ClientGen. Cada instância de ClientGenImpl está associada a uma conexão à BD, permitindo a execução de consultas e inserções;
- Remote_ClientAdm, Remote_ClientGen: Interfaces Remotas;

Tabelas BD:

```
CREATE TABLE Artists(
   artistID SERIAL PRIMARY KEY,
   name VARCHAR(255) NOT NULL,
   typeArt VARCHAR(255) NOT NULL,
   location VARCHAR(255) NOT NULL,
   acting BOOLEAN NOT NULL,
   status VARCHAR(20) NOT NULL
);

CREATE TABLE Performances(
   performanceID SERIAL PRIMARY KEY,
   artistID INT,
   date DATE,
   location VARCHAR(255) NOT NULL,
   FOREIGN KEY (artistID) REFERENCES Artists
);

CREATE TABLE Donations(
   donationID SERIAL PRIMARY KEY,
   artistID INT,
   value INT NOT NULL,
   FOREIGN KEY(artistID) REFERENCES Artists
);
```

Execução das aplicações:

Para utilizar o serviço deve seguir os seguintes passos (sempre dentro da pasta do projeto):

- Executar o comando num terminal à parte:
  → rmiregistry -J-classpath -Jbuild/classes port (sendo o port número do port disponivel)
- Compilar todas as classes do serviço com o comando:
  javac -d build/classes -classpath build/classes src/src/*.java
- Alterar, no ficheiro config.properties:
  → regPort para a porta onde o cliente se deve conectar, esta igual ao regPortServer;
  → regPortServer para a porta onde o servidor se deve conectar, esta igual ao regPort;
- Executar o servidor num terminal à parte com o comando:
  → java -classpath build/classes:lib/postgresql.jar src.Server
- Executar o cliente, um deles, ou o cliente geral ou o cliente administrador com o comando:
  → java -classpath build/classes:lib/postgresql.jar src.ClientGeneral
  OU
  → java -classpath build/classes:lib/postgresql.jar src.ClientAdmin

Funcionamento:

Todo o código fonte foi devidamente comentado com atenção às boas práticas de programação, pelo que o funcionamento das aplicações já está explícito aquando da leitura do código, logo, achámos pertinente não estar apenas a transcrever informação já presente na parte mais importante do projeto, o que leva este relatório a não apresentar muita informação relativamente ao funcionamento e implementação detalhada do projeto.

Instruções de utilização:

Aquando da execução da aplicação ClientGeneral e da aplicação ClientAdmin, com a correta ligação ao servidor, vão ser apresentadas na consola do terminal as várias ações disponíveis ao usuário por meio de uma lista numerada, onde, ao usuário, se pretender realizar uma ação, bastar-lhe-á escrever o número da lista correspondente à ação. Posteriormente ser-lhe-ão apresentadas mais instruções, pertinentes ao funcionamento da ação escolhida, pelo que o usuário terá apenas de seguir as instruções e escrever os dados nos formatos corretos (apresentados nas mesmas instruções).

Observações:

Alguns dos métodos implementados têm como objetivo alterar a base de dados, mais concretamente adicionando-lhe dados, pelo que achámos pertinente implementar uma convenção de formatação dos dados a inserir na BD, por ex: retirar espaços desnecessários pelo uso do método “trim()” e converter tudo para lowercase.

Todos os métodos (a que for pertinente), têm verificações quanto ao tipo de dados inserido, por ex: quando é necessário inserir um ID, há verificação se o input é ou não um número, sendo emitida uma mensagem de acordo.

Foi-nos dado a conhecer que a interface visual do programa não era objeto de grande incidência avaliativa, mas mesmo assim tomámos consideração pela experiência do utilizador e tentamos ao máximo deixar a interface legível e minimamente apresentável.

Conclusão:

Este trabalho prático permitiu-nos aplicar os conhecimentos lecionados nas aulas, mais especificamente as soluções de middleware para conexões cliente-servidor (Java RMI), aplicando a sua utilidade prática à problemática proposta no enunciado, tendo isso promovido a nossa aprendizagem do seu funcionamento e utilização, em visa à matéria lecionada. Sendo-nos possível implementar tudo o que nos era requerido no enunciado, ficou a faltar uma ação bónus na aplicação do Cliente Administrador, sendo esta a consulta e alteração da descrição e foto de um artista, enquanto que implementamos todas as outras tarefas de caráter opcional.