Relatório de Implementação do Sistema de Gestão de Artistas de Rua

Sistemas Distribuídos

Thawila Simbine - 49183

Luís Borges - 47297

Évora, Janeiro de 2023

Introdução

O trabalho consiste na implementação de um sistema que possa ajudar a Associação de Artes e Espectáculos TáNaForja quer melhorar o sistema de gestão de artistas SeekArtist. Para tal, necessita que o sistema se torne mais robusto, escalável e seguro.
Neste trabalho foram implementadas 2 aplicações: servidor (SeekServ) e cliente geral junto de admin (SeekERAdmin) de acordo com os parâmetros listados no enunciado. Em nossa implementação da aplicação decidimos seguir a dica deixada no enunciado de ao invês de implementar duas aplicações clientes, implementar só um e diferenciar o acesso de acordo com os dados de autenticação.

Arquitetura

Tal como descrito na introdução, o nosso sistema é composto por duas partes principais: o servidor (SeekServ) e o cliente (SeekERAdmin), onde os clientes Geral e Admin são diferenciados pelos dados de autenticação.

A comunicação entre as aplicações clientes e servidor é feita por meio de Sockets TCP onde o servidor corre no endereço local da máquina (localhost) e o cliente pode se conectar com o servidor usando o endereço local da máquina também caso as duas aplicações sejam executadas na mesma máquina ou usando o endereço do servidor na rede caso as aplicações sejam executadas em máquinas diferentes.

Toda comunicação entre as aplicações é feita a partir da troca de texto e números inteiros. Na implementação das funções de comunicação usaram-se as classes mais básicas para ficheiros java (InputStream e OutputStream), sem uso de classes mais complexas como ObjectInputStream e ObjectOutputStream, dentre outras. O uso das classes InputStream e OutputStream dão mais controle para o caso em que seja necessário fazer a aplicações (cliente e servidor) se comunicar com outras aplicações (cliente e servidor) implementadas em outras linguagens de programação, como python por exemplo.

Padrão de comunicação

Tal como descrito na seção acima, as aplicações usam Sockets TCP para estabelecerem comunicação, mas para que a comunicação seja eficaz um padrão de comunicação foi implementado de modo a garantir que o servidor entenda o pedido do cliente e o servidor responda corretamente aos pedidos do cliente.

Usando como base o modelo de funcionamento de da arquitetura REST, criamos um cojunto de constantes em que cada uma descreve ou uma função/funcionalidade que o cliente pode requisitar ao servidor ou descreve uma resposta que o servidor pode dar ao cliente. As constantes são encontradas no ficheiros ***Contants.java***, que se encontra no diretório contants de ambas aplicações, tanto cliente quando servidor. Eis exemplo de constantes que podem ser encontradas nesse ficheiro:
Constantes que o cliente pode enviar ao servidor na requisição de uma funcionalidade:

- `PEDIR_REGITAR_UM_ARTISTA = 100;`
- `LISTAR_ARTISTAS_COM_OPCAO_FILTROS = 101;`
- `LISTAR_LOCALIZACOES_COM_ARTISTAS_A_ATUAR = 102;`
- `LISTAR_DATAS_E_LOCALIZACÕES_ONDE_ARTISTA_ATUOU = 103;`

Constantes que o servidor pode retornar como resposta:

- `SUCESSO = 200;`
- `ERRO = 400;`

Quando um cliente solicita ao servidor por uma funcionalidade cuja resposta são dados, então servidor não envia nenhuma das constantes acima, somente envia os dados requisitados. As constantes só são retornadas para requisições de modificação de dados (CREATE e UPDATE e DELETE), pois requisições de Listagem de dados, os dados retornados já servem como resposta.

Nota: Todas as constantes usadas nas comunicações são números inteiros.

Então quando um cliente solicita por um serviço (autenticar-se por exemplo), ele envia ao servidor a constante que indica que o cliente está a requisitar por tal funcionalidade e em seguida envia os dados de autenticação (username e password). O servidor recebe os dados da

requisição, determina a resposta e envia para o cliente. Durante o tempo que o servidor fica determinando a resposta, o cliente fica em espera até receber a resposta do cliente.
Esse padrão de comunicação todo está implementado no ficheiro ***ServicoDeMensagens.java***, na pasta *servicos* tanto na aplicação cliente assim como na servidor.

Organização do projeto

A aplicação servidor está organizada em 5 diretórios segundo o padrão MVC, nomeadamente:

- *app* - Diretório com o ficheiro de inicialização do programa **App.java**
- *constants* - Diretório com os ficheiros para definição de constantes, **Constants.java** e **TipoDeUtilizador.java**
- *modelo* - Diretório com as entidades da aplicação, ou seja, classes de definição de dados;
- *repositorio* - Diretório com as classes repositórios da aplicação, isto é, as classes que implementam métodos para comunicação direta com a base de dados. As outras classes invocam métodos destas classes para realizar operações sobre a base de dados;
- *servicos* - Diretório com as classes serviços da aplicação, isto é, as classes que implementam toda lógica de negócio de cada requisição feita pelo cliente;
- *servidor* - Diretório com as classes controladoras da aplicação. Elas mapeiam as requisições dos clientes aos serviços do diretório de serviços.

Funcionalidades implementadas

A aplicação cliente permite ao **utilizador geral** efetuar as seguintes operações:

- Inscrever-se no sistema, indicando o username, email e password pretendidos. (Não é necessário se increver, caso já existir um utilizador com o mesmo username ou email).
- Autenticar-se, usando o seu `username` e `password`. Só é permitido efetuar as operações seguintes após uma correta autenticação.
- Pedido de registo de um novo artista (se o utilizador encontrar um artista na rua, pede para este entrar no sistema). Aqui, o artista fica com um estado de não aprovado até ser aprovado por um administrador. É considerado o facto de, possivelmente, o artista já ter sido introduzido por outro utilizador. Neste caso, deve ser adicionada apenas a localização (caso ainda não tenha sido anteriormente).

- Listar artistas, com filtros (opcionais – no uso, não na implementação) por localização e arte.
- Listar localizações onde existem artistas a atuar.
- Para um determinado artista, identificado por um artistID, listar as datas e localiza- ções onde este já atuou.
- Para um determinado artista, identificado por um artistID, listar a data e localização da próxima (futura) atuação.
- Enviar um donativo a um artista, indicando o valor monetário.
- Dar uma classificação (rating) a um artista.
- Ver, na listagem de artistas, o rating de cada um.

A aplicação cliente permite ao **administrador** efetuar as seguintes operações:

- Autenticar-se, usando o seu `username` e `password`. Só permite efetuar as operações seguintes o utilizador que efetuar uma correta autenticação e confirmação da permissão (ser do tipo administrador).
- Dar permissão de administrador a um User.
- Listar artistas por estado.
- Aprovar um artista que ainda não tenha sido aprovado.
- Consultar e alterar as informações de um artista.
- Todas as operações permitidas pelo cliente geral .

Nota: Todas as funções do cliente estão implementadas e funcionais.

Quanto ao **servidor**, ele só responde as requisições feitas pelo clientes e não implementa nenhuma regra especial, como replicação para tolerância a falhas ou `pusblish/subscribe` onde cliente pode pedir para ser notificado sobre novos artistas ou atuações inseridos, em tempo real.

Configuração do projeto:

- A comunicação entre os Sockets do programa é feita usando a porta 49183.
- O projeto usa maven como gestor de dependências para cada um dos módulos. Portanto para execução da aplicação, basta a execução do comando: `maven exec:java` pela linha de comando no diretório das aplicações.

- A aplicação cliente recebe argumentos opcionais pela linha de comando, que são endereço do servidor e porta de execução do servidor caso esta seja alterada.
- As configurações de conexão da base de dados estão no ficheiro app.proprieties que encontra-se no diretório servidor do servidor.

Conclusão

A implementação do sistema utilizando Sockets proporciona uma comunicação eficiente entre os clientes e o servidor. As funcionalidades atendem às necessidades da Associação de Artes e Espectáculos TáNaForja, proporcionando uma gestão eficaz dos artistas de rua no Alentejo.