Relatório 1º Trabalho Prático
Sistemas Distribuidos
Room Rent

Trabalho realizado por:
Diogo Carreiro nº48729
Rodrigo Alves nº48681

Docente:
Professor José Saias

1. Introdução

Neste trabalho o objetivo é implementar as aplicações servidor, cliente geral e cliente de gestão que permitam, desde as mais variadas localizações, o acesso ao serviço para gerir os anúncios de quartos para arrendar, e também as respetivas solicitações de alojamento.
No cliente geral deve ser possível:

- registar novo anúncio do tipo oferta, ou do tipo procura, sendo que o anúncio fica em estado inativo até ser aprovado pelo gestor, e atribui-lhe um novo código único, que será devolvido como resultado da operação
- listar anúncios (com estado ativo) do tipo oferta (com filtros opcionais por preço, género e localização)
- listar anúncios (com estado ativo) do tipo procura (com filtros opcionais por preço, género e localização)
- listar todos os anúncios de um anunciante
- obter todos os detalhes de um anúncio, dado o seu identificador (aid)
- enviar nova mensagem ao anunciante de um anúncio, pelo identificador aid
- consultar as mensagens inseridas para um determinado anúncio

No cliente gestão é possível:

- listar anúncios por estado
- obter detalhes de um anúncio
- aprovar um anúncio, alterando o estado do mesmo para ativo
- alterar o estado de um anúncio.

Neste trabalho era pedida uma apliiação ciliente-servidor, através de uma solução de middleware apresentada nas aulas. Então nós usamos JavaRMI para essa solução.

2. Estrutura do Código

Para realizar este trabalho foi implementada uma interface remota “Anuncios”, juntamente com a sua implementação na classe AnunciosImpl, representando assim todos os métodos usados para este trabalho. Usámos também a classe anuncios_entity para criar objetos do tipo anuncio que contêm todos os atributos especificados.

3. Base de Dados

Na nossa bd inserimos 2 tabelas, uma relativa aos anuncios, que contém todos os atributos de um anúncio, e uma relativa às mensagens que são enviadas para cada um dos anúncios.

```
CREATE TABLE anuncios(
    aid serial,
    localizacao varchar(30),
    preco varchar(5),
    data varchar(40),
    genero varchar(20),
    anunciante varchar(30),
    tipo_alojamento varchar(20),
    tipo_anuncio varchar(20),
    estado varchar(10),
    primary key (aid)

);

CREATE TABLE mensagens(
    aid serial,
    remetente varchar(30),
    mensagens varchar(50),
    primary key (aid)
);
```

4. Servidor e Cliente

Como foi acima mencionado a comunicação entre o servidor e cliente foi feita através de JavaRMI usando o objeto remoto Anuncio.
A implementação do servidor foi quase concebida na totalidade na aula prática, porém levou algumas alterações de modo a ser ajustada ao trabalho em questão. Do mesmo modo, o Cliente sofreu também as alterações neessárias, incluindo a "construção" de um método para mostrar o menu (no terminal) do trabalho.

5. Métodos usados

int regist_anuncio(String localizacao, int preco, String genero, String anunciante, String tipo_alojamento, String tipo_anuncio) throws Exception: Este método serve para registar um anúncio na base de dados. Inicialmente o anúncio fica com o estado "Inativo".

LinkedList<Anuncio_Entity> listarAnuncios_oferta() throws Exception: Este método retorna a lista de anúncios cujo tipo é "Oferta".

LinkedList<Anuncio_Entity> listarAnuncios_procura() throws Exception: Este método retorna a lista de anúncios cujo tipo é "Procura".

LinkedList<Anuncio_Entity> listarAnuncios(String anunciante) throws Exception: Este método retorna a lista de anúncios de um determinado anunciante.

String get_details(int aid) throws Exception: Mostra todas as informações acerca de um dado anúncio cujo id é inserido pelo utilizador.

void send_message(String aid ,String message,String remetente) throws Exception: Envia uma mensagem para um anúncio cujo id é inserido pelo utilizador. Esta mensagem também possui o remetente.

LinkedList<String> get_messages(int aid) throws Exception: Retorna a lista com todas as mensagens associadas a um dado anúncio cujo id é inserido pelo utilizador.

LinkedList<Anuncio_Entity> listarAnuncios_ativos() throws Exception: Retorna a lista com todos os anúncios ativos presentes na base de dados.

LinkedList<Anuncio_Entity> listarAnuncios_inativos() throws Exception: Retorna a lista com todos os anúncios ativos presentes na base de dados.

String aprove_anuncio(String aid) throws Exception; String change_state(String aid) throws Exception: Método que aprova um dado anúncio, alterando o estado para "ativo".

String print_anuncio(int aid,String localizacao, int preco, String data, String genero, String anunciante, String tipo_alojamento, String tipo_anuncio, String estado) throws Exception: Método auxiliar que é utilizado para transformar em string todos os detalhes de um anúncio para ser printado em alguns métodos anteriores.

6. Problemas encontrados e observações

Na realização deste trabalho um dos maiores problemas que enfrentámos foi trabalhar com interfaces remotas, pois nunca tinhamos trabalhado com algo do género, excetuando nas aulas práticas. Outra dificuldade encontrada foi trabalhar com Postgres e conectar-nos à base de dados através do programa implementado pelo grupo. Isto deve-se ao facto de termos tido algumas dificuldades na utilização correta dos comandos do terminal. Posteriormente, em algumas situações tivemos alguns problemas de sintaxe, que levantavam exceções no código.

Para álem disso temos um pequeno problema na implementação do ficheiro “cliente_geral”, uma vez que as mensagens só são enviadas para anúncios cujo estado é “ativo”.

7. Comandos utilizados na execução do programa

Para compilar o projeto usámos a funcionalidade “Clean and Build” do NetBeans.

Em termos de execução utilizámos os seguintes comandos (cada um num terminal diferente):

```
rmiregistry -J-classpath -Jbuild/classes 7000
(Servidor) java -classpath build/classes:resources/postgresql.jar sd.rmi.Servidor 7000 localhost bd1 anunciante1 anuncios2022
(Cliente Geral) java -classpath build/classes sd.rmi.ClienteGeral localhost 7000
(Cliente Gestão) java -classpath build/classes sd.rmi.ClienteGestao localhost 7000
```