

# Sistemas Distribuídos

Relatório do Trabalho Prático 2

2023/2024



Trabalho realizado por:

- Diogo Ferreira, nº430751
- Rúben Farinha, nº48329

## Introdução

O objetivo do trabalho é implementar um sistema com arquitetura distribuída, que inclua um cliente (que vai servir de cliente geral e de cliente de administração dependente do tipo de utilizador que se autenticar, para isto utilizamos um menu de autenticação antes de poder aceder aos clientes geral e de administração) e um servidor.

Primeiramente temos um menu de autenticação em que qualquer cliente se pode registar ou autenticar caso já esteja registado. Depois dependendo do tipo do cliente podem fazer funções diferentes:

O **cliente geral** tem a possibilidade de:

- Registrar novos artistas, sendo que o artista fica em estado inativo até ser aprovado pelo administrador. Caso este artista já exista podemos atualizar a sua localização.
- Listar artistas (com estado ativo) pela sua localização ou tipo de arte.
- Procurar localizações onde existem artistas a atuar.
- Utilizando o `artist_id` podemos listar as datas e localização onde este já atuou ou irá atuar futuramente.
- Enviar e listar donativos a um artista.

O **cliente administrador** pode:

- Listar artistas por estado.
- Aprovar um artista, ou alterar os dados do mesmo.
- Alterar permissões de um user.
- Ter acesso a todas as funcionalidades referidas no cliente geral.

Por outro lado o **servidor** é responsável por:

- Serviço para as operações a disponibilizar a cada cliente, sendo que nas listagens do cliente geral mostra apenas artistas em estado ativo.
- Armazenamento persistente de dados.

Para isso utilizámos o spring para o desenvolvimento do projeto.

## Estrutura

Neste trabalho, foram implementados 2 módulos, um representa o cliente(geral e administrador), e o servidor. O módulo servidor contém todas as implementações que são necessárias para o bom funcionamento do sistema. Abaixo incluimos alguns dos métodos utilizados:

- Módulo Cliente:
  - **public void addUtilizador(String username, String password, String email):**
    - Este método irá adicionar um utilizador à base de dados.
  - **public void addArtista(String artist\_name, String artist\_type, Integer latitude, Integer longitude, String is\_acting, String approval):**
    - Este método irá adicionar um artista à base de dados.
  - **public Optional<Artista\_Entity> get\_ArtistaByID(Integer id):**
    - Este método retorna o anúncio que tem o id igual ao do parâmetro id

(se existir).

- **public List<Artista\_Entity> getAllArtistas():**
  - Este método retorna todos os artistas.
- **public List<Artista\_Entity> getAllByApproval (String approval):**
  - Este método retorna todos os anúncios de um certo estado de aprovação (aprovado ou não aprovado).
- **public List<Artista\_Entity> getAllAnunciosByType(String type):**
  - Este método retorna todos os artistas de um tipo de arte dado por type.
- **public List<Artista\_Entity> getAllAnunciosByLatitude (Integer latitude):**
  - Este método retorna todos os artistas com latitude definida pelo parâmetro latitude. Temos ainda um método igual mas para a longitude.
- **public List<Artista\_Entity> getAllByActing(String acting)**
  - Este método retorna todos os artistas que se encontrem a atuar.
- **public void approve\_state(Integer id):**
  - Este método permite ao administrador aprovar um artista, tornando o seu estado ativo.
- **public void changetype(String username):**
  - Este método permite ao administrador aprovar um utilizador através do seu username.

## Problemas encontrados e observações

Apesar de termos concluído o trabalho e praticamente todas as funcionalidades do mesmo, ficaram duas por tratar, sendo elas listar os locais futuros onde artistas vão atuar e os locais passados.

## Cliente





	username [PK] character varying (255) 	email character varying (255) 	password character varying (255) 	type character varying (255) 
1	pedro	pedromigalhas@gmail.com	migalhas	admin
2	Tiago	tiagomanteiga@gmail.com	manteiga	geral

Figura 1 – Exemplo de 2 utilizadores criados.

```
Bem-Vindo
|-----|
|  Insira a opção desejada:  |
|  1 - Login                 |
|  r - Registrar            |
|  s - Sair                  |
|-----|

-> 1
Insira os seguintes campos:
Username -> pedro
Password -> migalhas

Menu Admin
|-----|
|  Insira a opção desejada:  |
|  1 - Registrar artista    |
|  2 - Listar artista por localização |
|  3 - Listar artista por arte |
|  4 - Listar localizações com artistas |
|  5 - Listar locais atuados de artista |
|  6 - Listar próximas atuações de artista |
|  7 - Enviar Donativo      |
|  8 - Listar Donativos de artista |
|  A1 - Dar permissões admin a user |
|  A2 - Listar artistas por estado |
|  A3 - Aprovar artista      |
|  A4 - Alterar artista      |
|  s - Sair                  |
|-----|

->
```

Figura 2 – Utilizador pedro com login feito e como tem type admin entramos diretamente no menu para os administradores.

```
Bem-Vindo
|-----|
|  Insira a opção desejada:  |
|  1 - Login                 |
|  r - Registrar            |
|  s - Sair                  |
|-----|

-> 1
Insira os seguintes campos:
Username -> Tiago
Password -> manteiga

Menu Cliente
|-----|
|  Insira a opção desejada:  |
|  1 - Registrar artista    |
|  2 - Listar artista por localização |
|  3 - Listar artista por arte |
|  4 - Listar localizações com artistas |
|  5 - Listar locais atuados de artista |
|  6 - Listar próximas atuações de artista |
|  7 - Enviar Donativo      |
|  8 - Listar Donativos de artista |
|  s - Sair                  |
|-----|

->
```

Figura 3 – Utilizador tiago com login feito e como tem type geral entramos no menu cliente geral.

```

      Bem-Vindo
    _____
|      Insira a opção desejada:      |
|      1 - Login                      |
|      r - Registrar                  |
|      s - Sair                      |
|      _____                    |
|
-> r
Insira os seguintes campos:
Username -> Fabio20
Password -> nike
Email -> Fabio20nike@gmail.com
Registo com sucesso

      Bem-Vindo
    _____
|      Insira a opção desejada:      |
|      1 - Login                      |
|      r - Registrar                  |
|      s - Sair                      |
|      _____                    |
|
->
```

Figura 4 – Exemplo de registo de utilizador.

```

      Menu Cliente
    _____
|      Insira a opção desejada:      |
|      1 - Registrar artista          |
|      2 - Listar artista por localização |
|      3 - Listar artista por arte      |
|      4 - Listar localizações com artistas |
|      5 - Listar locais atuados de artista |
|      6 - Listar próximas atuações de artista |
|      7 - Enviar Donativo             |
|      8 - Listar Donativos de artista  |
|      s - Sair                      |
|      _____                    |
|
-> 3
Insira o seguinte campo:
arte -> Fado
[{"id":2,"name":"Ferreira","type":"Fado","latitude":1,"longitude":1,"acting":"Sim","approval":"inativo"}]

      Menu Cliente
    _____
|      Insira a opção desejada:      |
|      1 - Registrar artista          |
|      2 - Listar artista por localização |
|      3 - Listar artista por arte      |
|      4 - Listar localizações com artistas |
|      5 - Listar locais atuados de artista |
|      6 - Listar próximas atuações de artista |
|      7 - Enviar Donativo             |
|      8 - Listar Donativos de artista  |
|      s - Sair                      |
|      _____                    |
|
->
```

Figura 5 – Lista de todos os artistas que

## **Conclusão**

No geral, apesar de terem faltado duas funcionalidades consideramos que o trabalho ficou praticamente completo de acordo com o pedido pelo professor no enunciado. Em suma, neste segundo trabalho foi utilizada uma arquitetura diferente comparando com a que foi utilizada na primeira versão do mesmo, o que nos permitiu desenvolver as nossas competências, tanto na linguagem Java, como na cadeira de Sistemas Distribuídos.