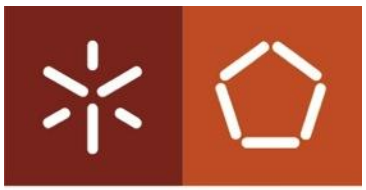

Universidade do Minho

Programação Orientada aos Objetos

UMeR

Grupo 27

A64951 - Miguel José Almeida Campos

A74260 - Luis Miguel da Cunha Lima

A74033 - João Pedro Carvalho Gomes

Índice

1. Introdução

No âmbito na Unidade Curricular de Programação Orientada aos Objetos foi proposta a realização de um projeto prático, denominado UMeR.

Este projeto consiste na realização de uma aplicação em JAVA que faça a gestão do serviço de transporte de passageiros. Neste relatório apresentamos várias das decisões que foram tomadas pelo nosso grupo durante o desenvolvimento deste projeto, evidenciando os passos que usamos para programar a dita aplicação.

Desde o início achámos o projeto extremamente aliciante o que nos motivou a trabalhar para o seu desenvolvimento, e tendo como principal objetivo criar uma interface eficiente, onde o utilizador pudesse interagir na UmeR. Procurámos desde cedo trabalhar para completar o nosso objetivo.

Seguidamente apresentamos o desenvolvimento onde vamos expor as nossas decisões e a parte chave do nosso código.

2. UmeR

Como referido anteriormente, o projeto baseia-se na gestão do serviço de transporte de passageiros e intitula-se como podemos observar, por UMeR.

Inicialmente pensámos no projeto como um esboço de forma a conseguir visualizar todos os elementos do mesmo, o que mais tarde ajudou no desenvolvimento das classes e respetivas variáveis de instância.

Apresentamos, então, o diagrama final do nosso projeto:

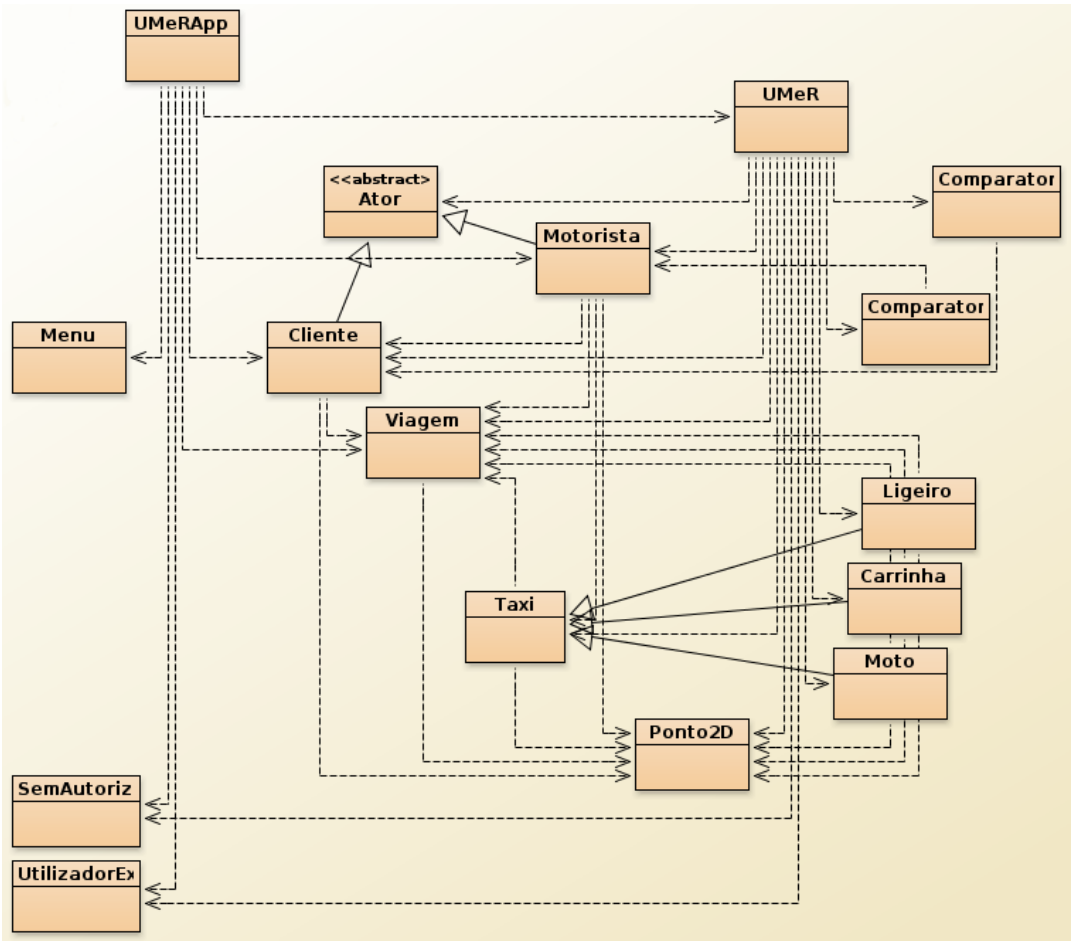

Figura 1. Diagrama das Classes

3. Decisões Tomadas

Para melhor organização do trabalho, optámos por definir as seguintes classes:

- Actor
- Cliente
- Motorista
- Taxi
- Ligeiro
- Carrinha
- Moto
- Viagem
- Ponto2D
- Menu
- UMeRApp
- UMeR

Dentro destas classes existem variáveis corretamente estruturadas, e uma vez que não nos foi dito qual a dimensão do projeto, optámos por usar HashMap, uma vez que o esta estrutura se mostra mais eficiente em bases de dados mais pequenas.

Nas variáveis que representam datas usámos o GregorianCalendar.

Passamos então a explicitar e explicar todas as variáveis de instância da classe UMeR e de todas as suas sub-classes:

UMeR:

private Ator ator;
private HashMap<String, Cliente> cliente;
private HashMap<String, Motorista> motoristas;
private HashMap<String, Moto> motos;

Actor:

- private String email; // Forma de garantir unicidade
- private String nome;
- private String pass;
- private String morada;
- private GregorianCalendar data;

Cliente:

- private List<Viagem> historico; // Lista de viagens realizadas

Motorista:

- private int fator;
- private int classificacao;
- private double kmsTotal;
- private boolean disponibilidade;
- private List<Viagem> historico; // Lista de viagens realizadas
- private int idTaxi; // ID do taxi associado
- private Viagem viagem; // Viagem a decorrer
- private int totalViagens; // Numero total de viagens realizadas

Taxi:

- private int idTaxi; // Forma de garantir unicidade
- private double velocidade;
- private double precoKm;
- private double precoHora;
- private Ponto2D coordenadas; // Coordenadas atuais do taxi
- private List<Viagem> historico; // Lista de viagens realizadas
- private boolean temFila // Dispõe de fila de espera?
- private List<Viagem> filaDeEspera; // Lista de viagens em espera

Ligeiro:

- Nenhuma variável atribuida;

Carrinha:

- Nenhuma variável atribuida;

Moto:

- Nenhuma variável atribuida;

Viagem:

- private Ponto2D origem;
- private Ponto2D destino;
- private String motorista; //Email do motorista
- private String cliente; //Email do cliente
- private double duracao;
- private double custo;
- private double distancia;
- private GregorianCalendar data;

Ponto2D:

- private int x;
- private int y;

Prosseguimos assim, para a explicação da classe Menu. Para tornar possivel a interação com a aplicação optámos pelo uso/criação de menus. É de notar que todos os menus foram criados e manipulados através dos mesmos métodos de forma a garantir a máxima reutilização do código. Seguem-se alguns exemplos destes menus:

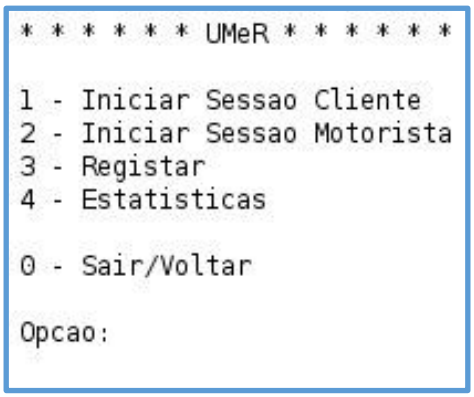

Figura 2. Menu Inicial

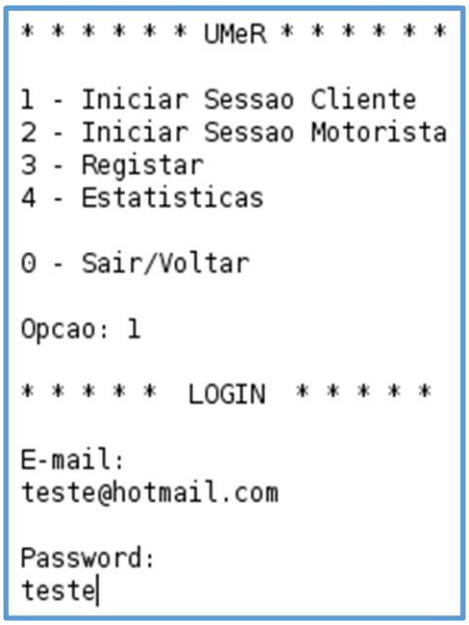

Figura 3. Menu Login

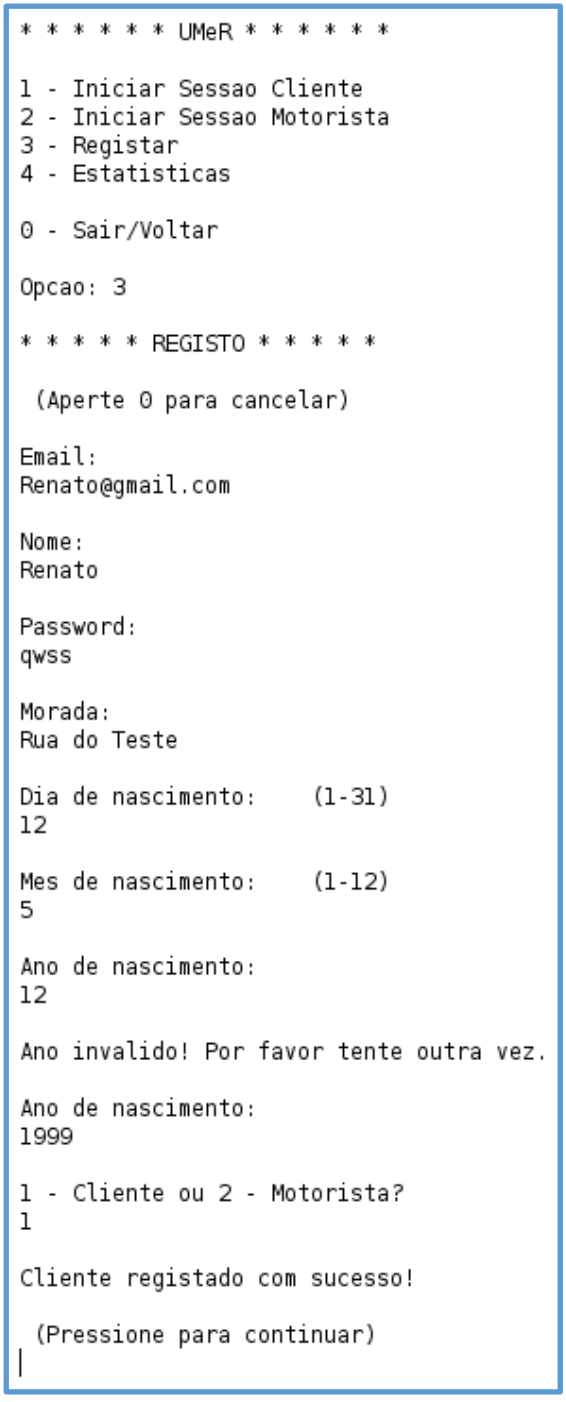

Figura 4. Menu Registo

```
* * * * * * UMeR * * * * * *

1 - Iniciar Sessao Cliente
2 - Iniciar Sessao Motorista
3 - Registar
4 - Estatisticas

0 - Sair/Voltar

Opcao: 1

* * * * *  LOGIN  * * * * *

E-mail:
sofi4@gmail.com

Password:
5665rrr

Log in efetuado com sucesso!

 (Pressione para continuar)


* * * * * * UMeR * * * * * *

1 - Ver Taxis Disponiveis
2 - Solicitar Viagem
3 - Fazer Reserva
4 - Avaliar Ultimo Motorista
5 - Ver Viagens Efetuadas

0 - Sair/Voltar

Opcao:
```

Figura 5. Menu Cliente

```
* * * * * * UMeR * * * * * *

1 - Iniciar Sessao Cliente
2 - Iniciar Sessao Motorista
3 - Registar
4 - Estatisticas

0 - Sair/Voltar

Opcao: 2

* * * * *  LOGIN  * * * * *

E-mail:
antunes1978@hotmail.com

Password:
4ntun3s

Log in efetuado com sucesso!

 (Pressione para continuar)


* * * * * * UMeR * * * * * *

1 - Sinalizar Disponibilidade
2 - Inserir Viatura
3 - Associar Viatura
4 - Registar Viagem
5 - Ver Viagens Efetuadas

0 - Sair/Voltar

Opcao:
```

Figura 6. Menu Motorista

4. Inserção Novos Tipos

Para acrescentar um novo tipo de viatura:

1. Criar uma nova classe com as variáveis de instância que representam as características desse tipo de viatura, respetivos construtores, gets, sets, equals, toString e clone;
2. Alterar o método inserirViatura da classe UmeR para fazer o scan das características referidas acima.

Para acrescentar um novo tipo de motorista:

1. Criar uma nova classe com as variáveis de instância que representam as características desse tipo de motorista, respetivos construtores, gets, sets, equals, toString e clone;
2. Alterar as estruturas, isto é, em vez de existir apenas uma com todos os motoristas, podiamos criar o número de estruturas correspondente ao número de diferentes tipos de motorista;
3. Alterar os métodos de registo da classe UmeR para fazer o scan das características referidas acima e inserir o novo motorista na estrutura mais apropriada;

5. Conclusão

Primeiramente, podemos concluir que o projeto foi interessante de executar e permitiu adquirir um maior conhecimento sobre a linguagem JAVA. Desenvolvemos a nossa capacidade de programação aprendendo, de uma maneira construtiva, a criar bases de dados complexas.

Todos os elementos mostraram empenho e deram sempre o seu melhor no decorrer do trabalho, ajudando-se sempre com os conhecimentos que possuíam. Ponderámos e considerámos que o nosso projeto final cumpriu o que era proposto.

Em suma, apesar de pequenas dúvidas nunca houve nenhum impedimento à resolução deste projeto, e deixou-nos interessados para no futuro tentar aprender ainda mais sobre Java.