



Escola de Engenharia
Universidade do Minho

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA
Mestrado Integrado em Engenharia Informática
Programação Orientada aos Objetos

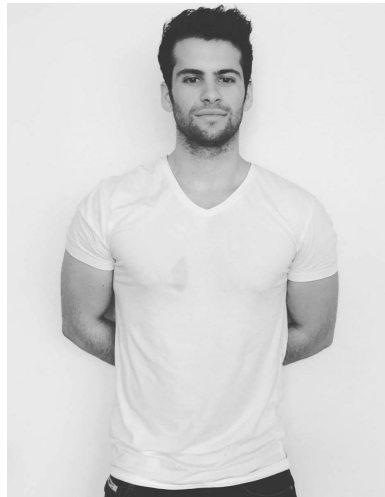
UMeR

Serviço de transporte de passageiros

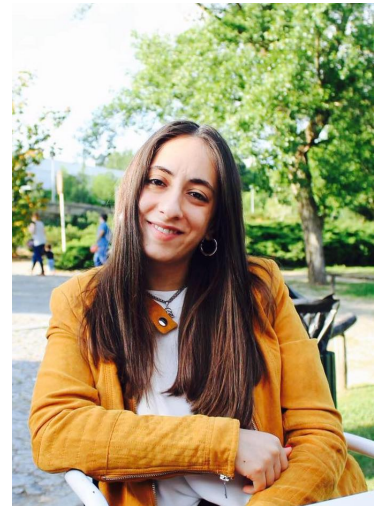
Grupo 07



Célia Figueiredo
a67637



José Carlos Faria
a67638



Márcia Costa
a67672

Braga, 2 de Junho de 2017

Conteúdo

1	Introdução	1
2	Descrição geral do projeto	2
2.1	UMeR	2
2.1.1	Actores do sistema	2
2.1.2	Os táxis UMeR	3
2.1.3	Fazer uma viagem no UMeR	4
2.1.4	Motoristas individuais vs Empresas de Táxi	4
2.1.5	Viaturas com Fila de Espera	4
3	Arquitetura das Classes	5
3.1	AtorInterface	6
3.1.1	Ator	6
3.1.2	Admin	6
3.1.3	Cliente	6
3.1.4	Motorista	6
3.2	Veiculo	8
3.3	Moto	8
3.3.1	MotoFilaEspera	8
3.4	CarroLig	9
3.4.1	CarroFilaEspera	9
3.5	Carrinha	9
3.5.1	CarrinhaFilaEspera	9
3.6	Historico	10
3.7	Utils	10
3.7.1	Meteorologia	10
3.7.2	Trânsito	10
3.8	Coordenadas	10
3.9	BD	11
3.10	UMeRMenu	11
3.11	UMeR	11
3.12	UMeRApp	11
4	Funcionamento da Aplicação UMeR	13
4.1	Menu Inicial	13
4.1.1	Registo	13
4.1.2	Iniciar Sessão	14

Resumo

O documento descreve o trabalho efetuado para a realização do projeto, onde foi pedida a implementação de um serviço de transporte de passageiros *UMeR* com o uso da linguagem *JAVA* esta que é orientada aos objetos. O serviço de transportes de passageiros *UMeR* é um serviço que faculta aos seus utilizadores a possibilidade de realizarem uma viagem desde um determinado local até ao local destino. A viagem pode ser efetuada por diferentes tipos de veículos: motos, carros e carrinhas e, naturalmente que diferentes tipos de veículos acarretam diferentes preços, velocidades distintas e também oscilações no que diz respeito ao número de lugares livres para que seja possível efetuarem as viagens como desejado. A aplicação em causa abrange todos os mecanismos desde a criação de utilizadores, motoristas, veículos, assim como, a marcação de viagens, a sua realização e a atribuição do seu valor correspondente.

1. Introdução

O presente relatório documenta o trabalho prático referente a Unidade Curricular de Programação Orientada aos Objectos pertencente ao plano de estudos do 2º ano do Mestrado Integrado em Engenharia Informática. Ao longo do semestre aprendemos a lidar com a linguagem de Programação orientada a objetos (também conhecida pela sua sigla *POO*.) que é um modelo de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos. Neste projeto foi-nos proposto o desenvolvimento de um serviço de transporte de passageiros *UMeR*, utilizando para tal a linguagem *JAVA* que é orientada a objetos. Em termos gerais, pretendemos desenvolver uma aplicação em *JAVA*, que é uma linguagem diferente das linguagens de programação convencionais, que são compiladas para código nativo, a linguagem *JAVA* é compilada para um bytecode que é interpretado por uma máquina virtual (Java Virtual Machine, mais conhecida pela sua abreviação *JVM*). É pretendido que a aplicação permita que um utilizador consiga realizar uma viagem num dos taxis da *UMeR* e que de certa forma consiga guardar toda a informação útil desde o registo de cada utilizador até aos extratos de viagens num determinado período, por exemplo. Ainda de realçar, é suposto conseguirmos os registos dos preços das viagens, assim como, marcação de viagens. Tudo isto, iremos mostrar ao longo do relatório de forma clara e objetiva.

2. Descrição geral do projeto

2.1 UMeR

Pretende-se que a aplicação a ser desenvolvida dê suporte a toda a funcionalidade que permita que um utilizador realize uma viagem num dos táxis da **UMeR**. O processo deve abranger todos os mecanismos de criação de utilizadores, motoristas, automóveis e posteriormente a marcação das viagens, a realização das mesmas e respectiva imputação do preço. Pretende-se também que o sistema guarde registo de todas as operações efectuadas e que depois tenha mecanismos para as disponibilizar (exemplo: viagens de um utilizador, extracto de viagens de um taxi num determinado período, valor facturado por um taxi num determinado período, etc.).

Cada perfil de utilizador deve apenas conseguir aceder às informações e funcionalidades respectivas.

- Os clientes dos táxis UMeR poderão:
 1. solicitar uma viagem ao táxi mais próximo das suas coordenadas;
 2. solicitar uma viagem a um táxi específico;
 3. fazer uma reserva para um táxi específico que, de momento, não está disponível.
- Os motoristas poderão:
 1. sinalizar que estão disponíveis para serem requisitados;
 2. registar uma viagem para um determinado cliente;
 3. registar o preço que custou determinada viagem.

2.1.1 Actores do sistema

Existirão dois tipos distintos de actores no sistema, que partilham a seguinte informação:

- email (que identifica o utilizador);
- nome;
- password;
- morada;
- data de nascimento.

Cliente

O Cliente representa a pessoa que solicita e efectua uma viagem de táxi. O cliente está sempre numa determinada localização (expressa em x e y, isto é, num espaço 2D) e escolhe um táxi específico ou então solicita o táxi mais perto que esteja disponível. O cliente tem também uma relação de todas as viagens que fez, com toda a informação relativa à viagem.

Motorista - colaborador da UMeR

O motorista conduz o táxi e além da informação atrás referida tem também dados relativos a:

- grau de cumprimento de horário estabelecido com o cliente, dado por um factor entre 0 e 100;
- classificação do motorista, dado numa escala de 0 a 100, calculada com base na classificação dada pelo cliente no final da viagem;
- histórico das viagens realizadas;
- número de kms já realizados na UMeR;
- informação sobre se está ou não disponível em determinado momento, isto é, se está ou não a trabalhar.

2.1.2 Os táxis UMeR

O ecossistema do UMeR contempla diferentes tipos de viaturas de aluguer (táxis). Neste momento estão em funcionamento os seguintes tipos de viaturas:

- carros ligeiros;
- carrinhas de nove lugares;
- motos.

Cada um destes tipos de viaturas tem associada:

- uma velocidade média por km;
- um preço base por km;
- um factor de fiabilidade, que determina a capacidade da viatura cumprir o tempo acordado com o cliente. Sempre que se realiza uma viagem é calculado (através da invocação de um `random()`) a capacidade de o veículo cumprir com o tempo acordado com o cliente. Este factor tem um efeito multiplicador sobre o tempo fornecido ao cliente.

Existem ainda alguns tipos de viaturas que possibilitam a existência de uma fila de espera de marcações. Quando o táxi não está disponível (por exemplo, pelo facto do condutor estar fora do horário de trabalho) é possível para essas viaturas aceitarem reservas de clientes. As reservas serão satisfeitas por ordem de chegada. Uma viatura sabe sempre a localização (em x e y) onde está. Quando realiza um serviço desloca-se para as coordenadas indicadas pelo cliente e fica aí parado até que seja solicitado um novo serviço.

2.1.3 Fazer uma viagem no UMeR

O processo de fazer uma viagem no UMeR segue as seguintes regras:

1. o cliente indica as coordenadas x e y em que se encontra;
2. o cliente decide se pretende chamar um táxi específico ou então solicitar o que está mais próximo;
3. por uma questão de simplificação, os UMeR deslocam-se sempre em linha recta pelo que o cálculo da distância entre o cliente e o táxi é feito pela cálculo da distância euclidiana (exemplo: se o cliente estiver em (0,0) e o táxi em (2,2) a distância é de 2.8284 kms);
4. após ser calculada a distância consegue-se saber, dadas as características do táxi, quanto tempo demora a chegar ao cliente e depois ao destino que o cliente solicita;
5. o táxi indica ao cliente qual o custo estimado da viagem, tendo em conta o deslocamento que é necessário efectuar, e o tempo total de viagem;
6. de acordo com a fiabilidade do carro (e de outros factores que pode considerar: a destreza do condutor, as condições meteorológicas, etc.) é calculado o tempo real da viagem. Se a diferença for superior a 25% do tempo estimado, então o preço a cobrar é o combinado com o cliente. Se a diferença for igual ou inferior a 25% o valor é ajustado para o valor real em função do tempo decorrido;
7. o táxi fica no ponto definido como fim da viagem à espera de nova solicitação de serviço;
8. após a viagem o cliente pode dar uma nota ao motorista e fica com o documento relativo à viagem guardado na sua área pessoal.

2.1.4 Motoristas individuais vs Empresas de Táxi

Numa primeira fase o UMeR foi pensado para condutores que conduziam a sua viatura e fazem serviço de transporte de clientes. No entanto, e devido ao sucesso do negócio, foram criadas empresas que possuem várias viaturas (em teoria de diversos tipos) e que empregam vários motoristas. A gestão que é feita destas empresas implica que após a criação da empresa se possam adicionar viaturas e motoristas, e que uma mesma viatura possa ser conduzida por motoristas diferentes (em tempos diferentes).

2.1.5 Viaturas com Fila de Espera

O UMeR possibilita que algumas viaturas possam ser definidas como possuindo uma fila de espera, que possibilita que quando a viatura está indisponível os pedidos sejam adicionados a uma fila de espera. Quando o veículo fica disponível são executadas todas as viagens inseridas na fila de espera, pela ordem de chegada. Os veículos com fila de espera possibilitam este comportamento, sendo que os demais não exibem este comportamento e nessa situação se a viatura estiver indisponível não será candidata a efectuar viagens.

3. Arquitetura das Classes

Neste capítulo falaremos do esqueleto da aplicação UMeR, serão abordadas as classes presentes na aplicação assim como os atributos e funcionamento de cada uma e também as decisões tomadas.

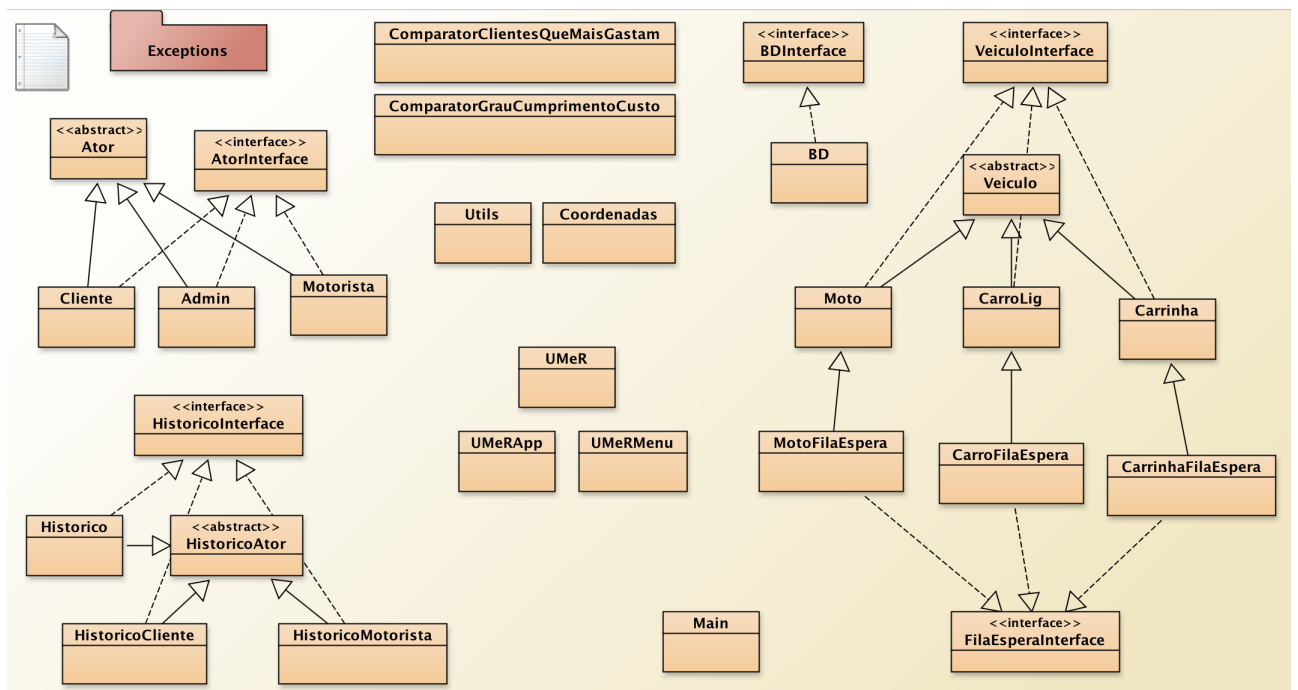


Figura 3.1: Esquema de Classes do BlueJ

3.1 AtorInterface

A interface AtorInterface servirá

3.1.1 Ator

A classe *Ator* é uma classe abstrata e servirá como “modelo” para outras classes que dela herdem, não podendo ser instanciada por si só. Para ter um objeto de uma classe abstrata é necessário criar uma classe mais especializada que herda dela e então instanciar essa nova classe. Neste caso foram criadas as classes *Admin*, *Cliente* e *Motorista* que herdam a informação que está na classe superior (*Ator*).

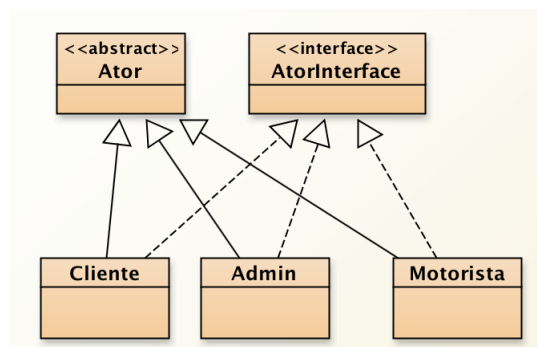


Figura 3.2: Classes

As variáveis de instância da classe abstrata *Ator* são apresentadas de seguida:

```
private String email;  
private String nome;  
private String password;  
private String morada;  
private LocalDate dataNascimento;
```

3.1.2 Admin

Na classe *Admin* estarão todos os dados herdados da classe *Ator*.

3.1.3 Cliente

Na classe *Cliente* estarão todos os dados herdados da classe *Ator*.

```
private Coordenadas loc; //localização atual do cliente  
private boolean emViagem;
```

3.1.4 Motorista

Na classe *Motorista* estarão todos os dados herdados da classe *Ator*.

```
private int grauCumprimentoHorario; //0-100  
private int classificacao; //0-100  
private double totalKms;
```

```
private boolean disponivel;//verifica se está disponível ou não
private boolean horarioTrabalho; //verificar se está no horário de trabalho
private double destreza; //valor entre 0.5 e 1.9
private VeiculoInterface veiculo;
private Historico viagemEmProcesso;
private int totalViagens;
```

Decidimos que a destreza do motorista seria atribuída através da invocação de um `random()` que gera valores entre 0,5 e 1,9 afim de gerar alguma aleatoriedade nos tempos obtidos das viagens efetuadas.

```
this.destreza = Utils.generateRandom(0.5f, 1.9f);
```

3.2 Veiculo

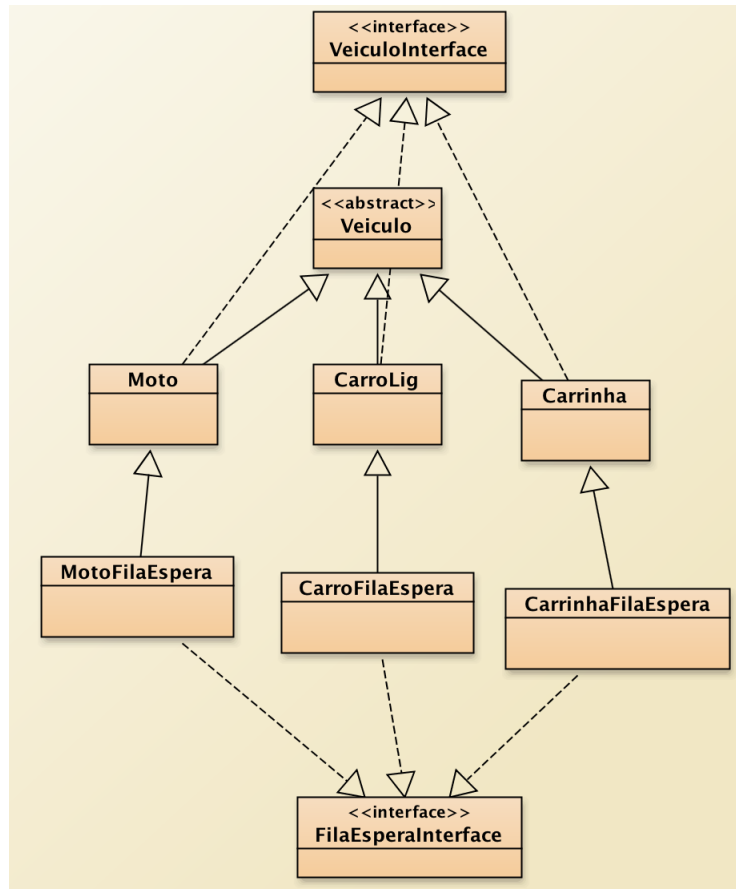


Figura 3.3: Classes

```
private String matricula;
private String marca; //Acrescentou-se a variável de instância marca,
para o cliente poder escolher um carro com base na marca do veiculo
```

```
private float fiabilidade; //0 a 2 random()
private Coordenadas loc;
```

3.3 Moto

```
private static final int lugaresLivres = 1;
private static final double vm = 40.5;
private static final double precoPorKm = 2.1;
```

3.3.1 MotoFilaEspera

```
private List<Cliente> filaClientes;
```

3.4 CarroLig

```
private static final int lugaresLivres = 4;  
private static final double vm = 65;  
private static final double precoPorKm = 3.5;
```

3.4.1 CarroFilaEspera

```
private List<Cliente> filaClientes;
```

3.5 Carrinha

```
private static int lugaresLivres = 8;  
private static final double vm = 55;  
private static final double precoPorKm = 5.1;
```

3.5.1 CarrinhaFilaEspera

```
private List<Cliente> filaClientes;
```

3.6 Historico

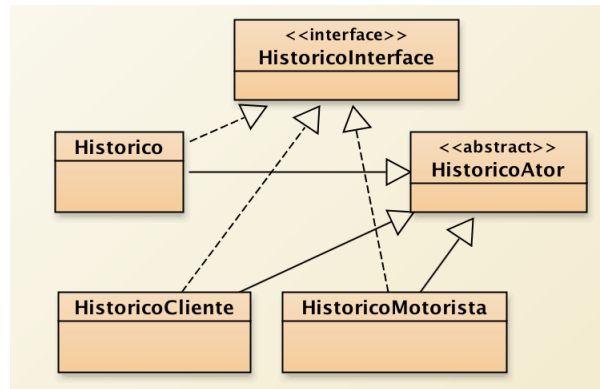


Figura 3.4: Classes

```
private String emailCliente;
private String emailMotorista;
```

3.7 Utils

Adicionalmente a classe *Utils* tem implementado um método que encripta a password. E um método que gera números random com intervalos de 0.1.

3.7.1 Meteorologia

```
public static final String sol = "Sol";
public static final String nevoeiro = "Nevoeiro";
public static final String granizo = "Granizo";
public static final String chuva = "Chuva";
public static final String neve = "Neve";
```

3.7.2 Trânsito

```
public static final String st = "Sem Transito";
public static final String tn = "Transito Normal";
public static final String mt = "Muito Transito";
```

3.8 Coordenadas

```
private double x;
private double y;
```

As cordenadas também são iniciadas com o método `random()`.

```
this.x=Utils.generateRandom(0f, 100f);
this.y=Utils.generateRandom(0f, 100f);
```

O método `getDistancia()` calcula a distancia euclidiana, este será um método importante na execução da simulação de uma viagem.

```
public double getDistancia (Coordenadas c){
    double distancia=0;
    distancia = Math.sqrt( Math.pow((this.x - c.getX()),2 ) +
                           Math.pow((this.y - c.getY()),2 ));
    return distancia;
}
```

3.9 BD

```
private Map<String, AtorInterface> clientes;
private Map<String, AtorInterface> motoristas;
private Map<String, AtorInterface> admins;
private Map<String, VeiculoInterface> veiculos;
private Set<Historico> historico;
```

3.10 UMeRMenu

```
private String titulo;
private List<String> opcoes;
private int op;
```

3.11 UMeR

```
private BDInterface baseDeDados;
private AtorInterface atorLoggado;
private int tentativasDeLoginFalhadas;
private Map<String, AtorInterface> atores;
```

3.12 UMeRApp

```
private static UMeR umer;
private static UMeRMenu menu_principal;
private static UMeRMenu menu_registar_atores;
private static UMeRMenu menu_motorista;
private static UMeRMenu menu_cliente;
private static UMeRMenu menu_dados_pessoais;
private static UMeRMenu menu_cliente_efetuarViagem;
private static UMeRMenu menu_admin;
private static UMeRMenu menu_registar_veiculos;
private static UMeRMenu menu_solicitarViagem;
private static UMeRMenu menu_inserir_coord_destino;
private static UMeRMenu menu_terminar_viagem;
private static UMeRMenu menu_terminar_horario_trabalho;
```

```
private static UMeRMenu menu_iniciar_horario_trabalho;  
private static UMeRMenu menu_proposta_viagem;  
private static UMeRMenu menu_historico;
```


4. Funcionamento da Aplicação UMeR

4.1 Menu Inicial

Esta é uma aplicação com uma interface para o utilizador muito simples, foi pensada de maneira a que o utilizador pudesse tirar o maior proveito da mesma, com comandos simples, tendo em conta que todos os menus funcionam à base de opções por números. Quando um utilizador executa a aplicação o primeiro menu a que está sujeito é o seguinte:

```
*****--Bem vindo à UMeR--*****
Escolha uma das seguintes opções:
  1 - Registar Utilizador
  2 - Iniciar sessão
  0 - Sair
*****
Opção: 1|
```

Figura 4.1: Menu Inicial

4.1.1 Registo

O utilizador consoante o seu tipo poderá efetuar um registo na aplicação. Será apresentado o seguinte menu para o registo de atores no sistema:

```
*****Escolha o tipo de utilizador a registar***
Escolha uma das seguintes opções:
  1 - Cliente
  2 - Motorista
  0 - Sair
*****
Opção: |
```

Figura 4.2: Menu de Registo na Aplicação

Os dados pedidos no ato de registo quer de clientes ou motoristas é o mesmo, no entanto serão guardados de acordo com o tipo inicialmente escolhido.

```
Nome: Pedro Costa
Email: pedro@pedro.com
Password: pedro
Morada: Rua do Pedro
Data de nascimento (YYYY-MM-DD): 1989-09-04|
```

Exemplo de registo de um motorista

```
Nome: Carolina Patrocinio
Email: carolina@carolina.com
Password: carolina
Morada: Rua da Fama
Data de nascimento (YYYY-MM-DD): 1987-05-30|
```

Exemplo de registo de um cliente

4.1.2 Iniciar Sessão

Cliente

O Cliente tem à sua disposição várias funcionalidades tais como: Solicitar uma viagem; Visualizar o histórico de viagens; Classificar viagens e ainda Ver e Editar dados pessoais.

```
*****Menu - Cliente*****
Escolha uma das seguintes opções:
  1 - Solicitar Viagem
  2 - Visualizar Histórico de viagens
  3 - Classificar Viagens
  4 - Atualizar Localizacao
  5 - Ver Dados Pessoais
  0 - Sair
*****
Opção:
```

Figura 4.3: Menu de Cliente

Ao escolher "Solicitar Viagem" será pedido ao Cliente para introduzir as Coordenadas de destino. Após a inserção das Coordenadas é pedido ao Cliente para escolher se quer o táxi que se encontra mais próximo dele ou se prefere escolher um táxi que permite fila de espera se estiver ocupado.

```
*****Inserir Coordenadas de destino*****
Escolha uma das seguintes opções:
  1 - Inserir Coordenadas
  0 - Sair
*****
Opção: 1
```

Menu para inserir as coordenadas

O Cliente após escolher a opção 1, é-lhe apresentado um menu com os dados do táxi que se encontra mais perto da localização do cliente. De seguida é dada a opção de aceitar fazer a viagem ou não.

```
***** Taxi mais próximo: *****
Motorista: João César | Veiculo: qt-76-22
Duração Estimada da Viagem (minutos): 52,33
Custo Estimado da Viagem (euros): 198,44

*****Dados Estimados da viagem *****
Escolha uma das seguintes opções:
  1 - Aceito
  2 - Não Aceito
  0 - Sair
*****
Opção: 1
```

Menu com os dados do taxi mais próximo

```
*****Menu - Solicitar Viagem*****
Escolha uma das seguintes opções:
  1 - Escolher táxi mais Próximo
  2 - Escolher táxi específico
  0 - Sair
*****
Opção: 2
```

Menu após a inserção de coordenadas: Escolha de táxi

```
*****Lista de Motoristas*****
1 - Email: manuel@manuel.com | Nome: Manuel César | Classificação: 0 | C
0 - Sair

Selecione uma da opções acima ou 0 caso pretenda voltar ao menu anterior
|
```

Menu com os dados dos táxis disponveis

Se aceitar fazer a viagem aparecer-lhe-á o seguinte menu, com a informação do tempo que o táxi demorará até chegar à localização do cliente.

```
0 táxi deverá demorar cerca de 14,57 minutos até à sua localização

*****Menu - Cliente*****
Escolha uma das seguintes opções:
  1 - Solicitar Viagem
  2 - Visualizar Histórico de viagens
  3 - Classificar Viagens
  4 - Ver Dados Pessoais
  0 - Sair
*****
Opção: |
```

Figura 4.4: Menu de Viagem aceite

Caso o cliente tenha efetuado uma viagem e esta ainda estiver a decorrer, não poderá solicitar outra viagem sem que a atual tenha terminado. Será mostrada uma mensagem como a seguinte:

```
Nao pode requisitar outra viagem porque esta a efetuar uma viagem neste momento

*****Menu - Cliente*****
Escolha uma das seguintes opções:
  1 - Solicitar Viagem
  2 - Visualizar Histórico de viagens
  3 - Classificar Viagens
  4 - Ver Dados Pessoais
  0 - Sair
*****
Opção: |
```

Figura 4.5: Menu de Erro: Cliente está em viagem

É oferecida aos Clientes e a todos os outros utilizadores a possibilidade de verem os seus dados e editarem todos os campos com a exceção do email.

```
***** Dados Pessoais *****
Email: marcia@marcia.com
Nome: Márcia
Password: *****
Morada: Rua da Televisão
Data de Nascimento: 1993-12-24
Localização:
  x: 64.0 y: 30.6
Em viagem: true
***** Dados Pessoais *****

*****Opções Dados Pessoais*****
Escolha uma das seguintes opções:
  1 - Editar Dados
  0 - Sair
*****
Opção:
```

Figura 4.6: Menu de visualização e edição dos dados Pessoais

Motorista

O motorista na UMeR poderá registrar e remover o seu veículo, gerir viagens é para iniciar uma viagem que esteja pendente; gerir horário de trabalho serve para não aceitar viagens quando não está a trabalhar. Poderá ver o histórico das viagens efetuadas, quais os seus melhores 10 clientes e ver e editar os seus dados pessoais. O menu apresentado será o seguinte:

```
*****Menu - Motoristas*****
Escolha uma das seguintes opções:
  1 - Gerir Viagens
  2 - Gerir Horário de trabalho
  3 - Visualizar Histórico de viagens
  4 - Visualizar 10 melhores clientes
  5 - Registrar Veiculo
  6 - Remover Veiculo
  7 - Ver Dados Pessoais
  0 - Sair
*****
Opção: 1|
```

Figura 4.7: Menu de Motorista

Após escolher gerir uma viagem será apresentado o menu com a opção de terminar viagem, caso Ao efetuar este passo a viagem passa a estar disponível no histórico quer no cliente quer no motorista.

```
*****Gestão de Viagem*****
Escolha uma das seguintes opções:
  1 - Terminar viagem em processo
  0 - Sair
*****
Opção:
```

Menu do motorista para terminar uma viagem

Na opção de gerir horário de trabalho é apresentado o seguinte menu em que o motorista o poderá terminar.

```
*****Horário de trabalho *****
Escolha uma das seguintes opções:
  1 - Terminar horario de trabalho
  0 - Sair
*****
Opção:
```

Menu do motorista para iniciar horario de trabalho

Viagem terminada :)

```
*****Menu - Motoristas*****
Escolha uma das seguintes opções:
  1 - Gerir Viagens
  2 - Gerir Horário de trabalho
  3 - Visualizar Histórico de viagens
  4 - Visualizar 10 melhores clientes
  5 - Registrar Veiculo
  6 - Remover Veiculo
  7 - Ver Dados Pessoais
  0 - Sair
*****
Opção: 2
```

Menu após terminar viagem

Terminou o seu horário de trabalho :)

```
*****Menu - Motoristas*****
Escolha uma das seguintes opções:
  1 - Gerir Viagens
  2 - Gerir Horário de trabalho
  3 - Visualizar Histórico de viagens
  4 - Visualizar 10 melhores clientes
  5 - Registrar Veiculo
  6 - Remover Veiculo
  7 - Ver Dados Pessoais
  0 - Sair
*****
Opção: |
```

Menu após terminar horário

Após terminar o horário, os menus mudam de modo a que o motorista possa voltar ao trabalho, sendo criados para tal os seguintes menus:

```
*****Horário de trabalho *****
Escolha uma das seguintes opções:
    1 - Iniciar horário de trabalho
    0 - Sair
*****
Opção:
```

Menu do motorista para terminar horario
de trabalho

Admin

```
Iniciou o seu horário de trabalho :)

*****Menu - Motoristas*****
Escolha uma das seguintes opções:
    1 - Gerir Viagens
    2 - Gerir Horário de trabalho
    3 - Visualizar Histórico de viagens
    4 - Visualizar 10 melhores clientes
    5 - Registar Veiculo
    6 - Remover Veiculo
    7 - Ver Dados Pessoais
    0 - Sair
*****
Opção:
```

Menu após iniciar horário de trabalho

5. Conclusão

Aspetos a melhorar seriam a apresentação dos menus, poder-se-ia ter melhorado a organização, assim como o