



Escola de Engenharia  
**Universidade do Minho**

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

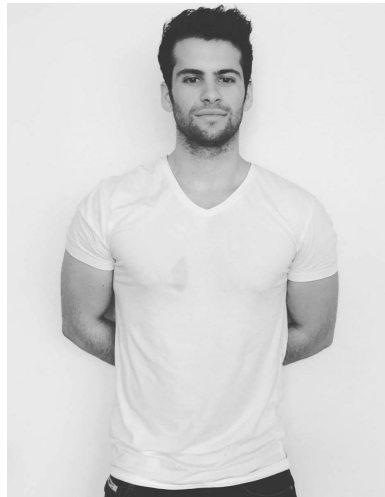
## UMeR

### *Serviço de transporte de passageiros*

### **Grupo 07**



Célia Figueiredo  
a67637



José Carlos Faria  
a67638



Márcia Costa  
a67672

Braga, 3 de Junho de 2017

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
<b>2</b>	<b>Arquitetura das Classes</b>	<b>2</b>
2.1	AtorInterface . . . . .	3
2.1.1	Ator . . . . .	3
2.1.2	Admin . . . . .	3
2.1.3	Cliente . . . . .	3
2.1.4	Motorista . . . . .	4
2.2	VeiculoInterface . . . . .	5
2.2.1	Veiculo . . . . .	5
2.3	Moto . . . . .	6
2.3.1	MotoFilaEspera . . . . .	6
2.4	CarroLig . . . . .	6
2.4.1	CarroFilaEspera . . . . .	6
2.5	Carrinha . . . . .	7
2.5.1	CarrinhaFilaEspera . . . . .	7
2.6	HistoricoInterface . . . . .	8
2.6.1	Historico . . . . .	8
2.7	Utils . . . . .	8
2.7.1	Meteorologia . . . . .	9
2.7.2	Trânsito . . . . .	9
2.8	Coordenadas . . . . .	9
2.9	BDInterface . . . . .	9
2.9.1	BD . . . . .	10
2.10	UMeRMenu . . . . .	10
2.11	UMeR . . . . .	10
2.12	UMeRApp . . . . .	11
<b>3</b>	<b>Funcionamento da Aplicação UMeR</b>	<b>12</b>
3.1	Menu Inicial . . . . .	12
3.2	Menu Inicial-Registo . . . . .	12
3.3	Menu Inicial - Iniciar Sessão . . . . .	13
3.3.1	Funcionalidades de Cliente . . . . .	13
3.3.2	Funcionalidades de Motorista . . . . .	16
3.3.3	Funcionalidades de Admin . . . . .	18
<b>4</b>	<b>Conclusões</b>	<b>22</b>
4.1	Trabalho Futuro . . . . .	22

## **Resumo**

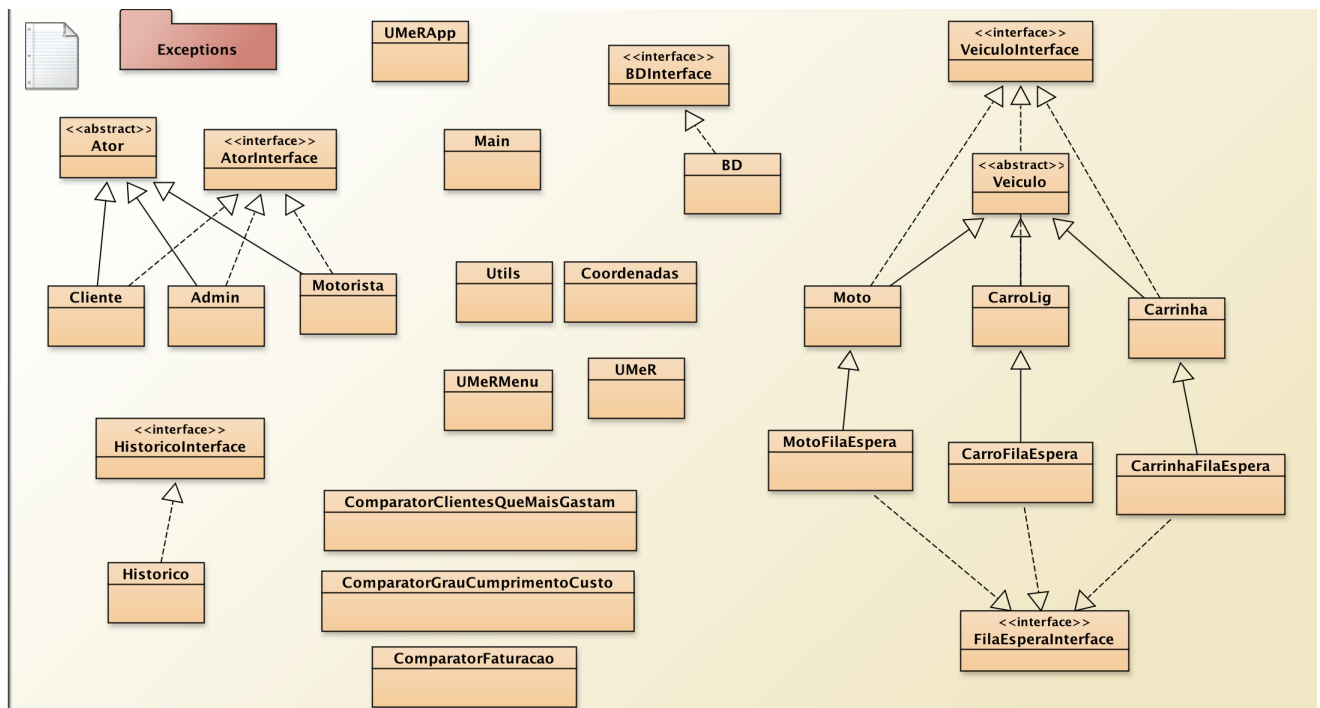
O documento descreve o trabalho efetuado para a realização do projeto, onde foi pedida a implementação de um serviço de transporte de passageiros *UMeR* com o uso da linguagem *JAVA* esta que é orientada aos objetos. O serviço de transportes de passageiros *UMeR* é um serviço que faculta aos seus utilizadores 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. 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.



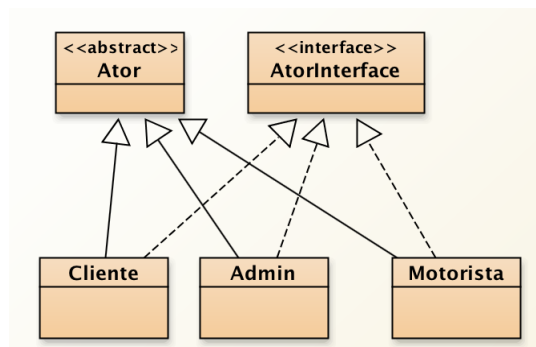
**Figura 2.1:** Esquema de Classes do BlueJ

## 2.1 AtorInterface

A interface *AtorInterface* define todos os métodos que os utilizadores da aplicação deverão ter definidos. Se no futuro for pretendido adicionar novos tipos de atores ao sistema, essas classes deverão também elas implementar esta interface. Desta forma poder-se-á inserir novos tipos de utilizadores à base de dados sem dificuldades. Como se pode visualizar na base de dados os utilizadores (motoristas, clientes e admins ) são coleções do tipo *AtorInterface*. Pode-se portanto adicionar os novos tipos de atores a cada uma destas coleções.

### 2.1.1 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 2.2:** 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;
```

### 2.1.2 Admin

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

### 2.1.3 Cliente

Esta classe corresponde a um *Ator* do tipo *Cliente* onde estarão todos os dados herdados da classe *Ator*, à qual se acrescenta uma variável de instância para guardar a localização e outra para indicar se o cliente está ou não em viagem.

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

### 2.1.4 Motorista

Esta classe corresponde a um *Ator* do tipo *Motorista* onde estarão todos os dados herdados da classe *Ator*. A esta classe foram acrescentadas diversas variáveis de instância para poderem ser guardadas informações acerca de grau de cumprimento de horário, a classificação do motorista esta que será uma média de todas as classificações efetuadas por um Cliente em determinada viagem. Decidimos acrescentar uma variável para diferenciar se um motorista estaria em horário de trabalho, isto é se está no local de trabalho para efetuar uma viagem. Por outro lado existe a variável de disponibilidade, esta que informa se o motorista está ocupado noutra viagem e portanto não poderá efetuar outra.

Foi também acrescentada a variável do tipo *VeiculoInterface* para guardar informações relativas ao veículo que cada motorista poderá ter. A variável do tipo *Historico* serve para saber se o motorista está a fazer uma viagem neste momento ou não, se for null é porque está livre, senão cria um histórico. Quando a viagem é terminada a variável disponível passa a true e *viagemEmProcesso* passa a null.

```
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);
```

## 2.2 VeiculoInterface

A interface `VeiculoInterface` define todos os métodos que os veículos da aplicação deverão ter definidos. Se no futuro for pretendido adicionar novos tipos de veículos ao sistema, essas classes deverão também elas implementar esta interface. Desta forma poder-se-á inserir novos tipos de veículos à base de dados sem dificuldades. Como se pode visualizar na base de dados os veículos (carros, motos, carrinhas, etc ) são coleções do tipo `VeiculoInterface`. Pode-se portanto adicionar os novos tipos de veículos à coleção.

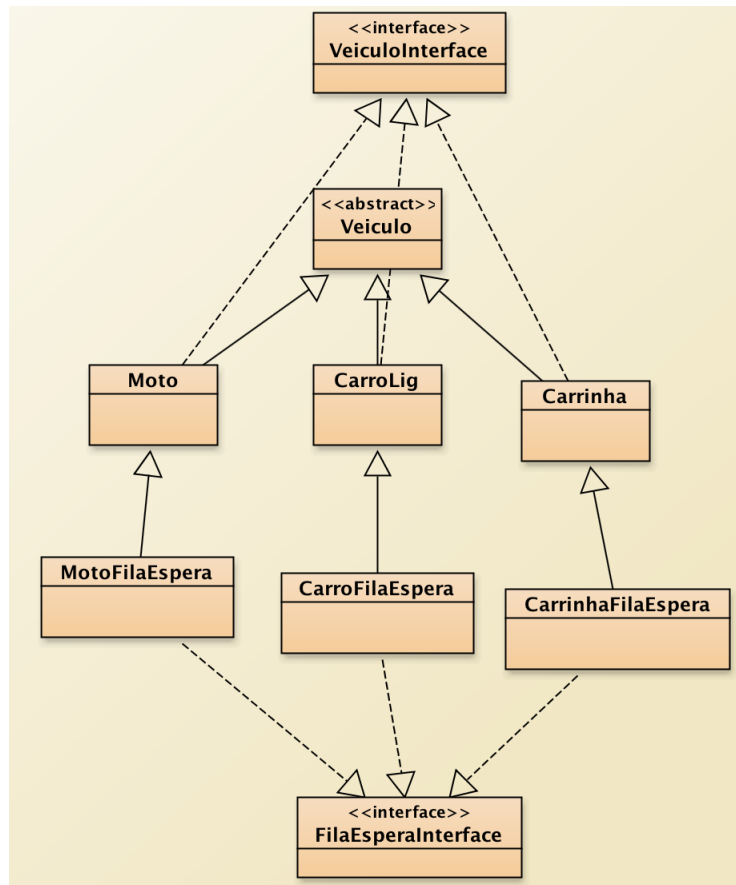


Figura 2.3: Classes

### 2.2.1 Veiculo

A classe *Veiculo* 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 *Moto*, *CarroLig* e *Carrinha* que herdam a informação que está na classe superior (*Veiculo*).

Esta classe terá como variáveis de instância a matrícula para identificação do veículo, a marca, a fiabilidade do veículo, o tempo de viagem, este será também um valor random dependente do tipo de veículo e por fim a localização do veículo.

```
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;
```

## 2.3 Moto

Esta classe corresponde a um *Veiculo* do tipo *Moto* onde estarão todos os dados herdados da classe *Veiculo*. A esta classe foram acrescentadas variáveis estáticas e com valor final, pois serão valores iguais para todas as instâncias deste objeto e com valores sempre iguais.

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

Foi acrescentado o método *calcFiabilidade()* para gerar aleatoriedade no cálculo da fiabilidade de um veiculo. Este método é invocado aquando da criação de um novo tipo de veiculo. Este método está implementado em todas as classes que dependem da classe *Veiculo*.

```
public static float calcFiabilidade(){
    float minX = 0.9f;
    float maxX = 1.2f;
    Random rand = new Random();
    float finalX = rand.nextFloat() * (maxX - minX) + minX;
    return finalX;
}
```

### 2.3.1 MotoFilaEspera

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

## 2.4 CarroLig

Esta classe corresponde a um *Veiculo* do tipo *CarroLig* onde estarão todos os dados herdados da classe *Veiculo*. A esta classe foram acrescentadas variáveis estáticas e com valor final, pois serão valores iguais para todas as instâncias deste objeto e com valores sempre iguais.

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

### 2.4.1 CarroFilaEspera

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

## 2.5 Carrinha

Esta classe corresponde a um *Veiculo* do tipo *Carrinha* onde estarão todos os dados herdados da classe *Veiculo*. A esta classe foram acrescentadas variáveis estáticas e com valor final, pois serão valores iguais para todas as instâncias deste objeto e com valores sempre iguais.

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

### 2.5.1 CarrinhaFilaEspera

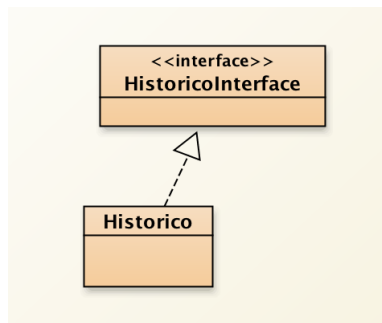
```
private List<Cliente> filaClientes;
```

## 2.6 HistoricoInterface

A interface `HistoricoInterface` define todos os métodos que os históricos da aplicação deverão ter definidos. Se no futuro for pretendido adicionar novos tipos de históricos ao sistema, essas classes deverão também elas implementar esta interface. Desta forma poder-se-á inserir novos tipos de históricos à base de dados sem dificuldades. Como se pode visualizar na base de dados os históricos são coleções do tipo *HistoricoInterface*. Pode-se portanto adicionar os novos tipos de históricos à coleção.

### 2.6.1 Historico

Esta classe serve para guardar dados pretinentes relativos a um histórico. Decidiu-se que seria necessário guardar o email do cliente e do motorista. Assim como uma data de inicio de viagem, a distancia da viagem, dados relativos ao custo e tempo: estimados e reais. O estado do tempo e transito assim como as coordenadas de origem e destino, a classificação do motorista. E ainda a indicação se um histórico histórico que está guardado na base de dados está terminado, quando se listam os históricos só apresentamos os que tem essa variável a true.



**Figura 2.4:** Classes

```
private String emailCliente;
private String emailMotorista;
private LocalDateTime dataDeInicioDeServico;
private double distancia;
private double tempoEstimado;
private double tempoReal;
private double valorEstimado;
private double valorCobrado;
private String estadoTempo;
private String estadoTransito;
private boolean terminado;
private Coordenadas origem;
private Coordenadas destino;
private int classificacao;
```

## 2.7 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.

### 2.7.1 Meteorologia

Esta classe é usada para gerar fatores de aleatoriedade no cálculo real do tempo de viagem.

```
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";
```

### 2.7.2 Trânsito

Esta classe é usada para gerar fatores de aleatoriedade no cálculo real do tempo de viagem.

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

## 2.8 Coordenadas

Esta classe guarda a localização de um utilizador. O construtor vazio está a ser usado na aplicação para gerar localizações aleatórias quando se criam os utilizadores.

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

As coordenadas 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;
}
```

## 2.9 BDInterface

A interface BDInterface define todos os métodos que as classes que geram dados da aplicação deverão ter definidos.

Se no futuro se pretender guardar a informação numa base de dados (Oracle por exemplo), esta classe deverá implementar a BDInterface. Desta forma, a mudança de classes que geram os dados guardados é muito simples uma vez que o resto da aplicação continuará a usar os mesmos métodos(definidos na BDInterface) que usa neste momento independentemente da forma como a classe implementa os métodos definidos pela interface.

### 2.9.1 BD

As coleções clientes, motoristas e admins são do tipo Map e estão organizados pela chave que é o respetivo email. Uma vez que se efetuam muitas pesquisas, sobre estes dados com a chave email, a melhor opção seria um HashMap uma vez que a pesquisa pela chave é instantânea.

Para a coleção de veículos foi utilizada também um HashMap em que a chave é a matrícula e os motivos são os mesmos descritos em cima. Apesar de neste momento não se efetuar muitas pesquisas pela chave, imagina-se que no futuro tal poderá acontecer.

A coleção de históricos escolhida foi o Set, pois não devem ser permitidos históricos repetidos. Uma vez que algumas das funcionalidades da aplicação implicavam percorrer os dados por ordem (data), pareceu-nos indicado usar um TreeSet e desta forma ter os dados guardados e ordenados de forma a facilitar as funcionalidade do programa.

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

## 2.10 UMeRMenu

Esta classe tem como objetivo facilitar a criação de menus e a leitura da opção introduzida pelo utilizador. Facilitando em muito o trabalho da classe principal *UMeRApp*.

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

## 2.11 UMeR

Esta classe é o coração da aplicação, uma vez que é nela que está centralizada a lógica das gestão de viagens. É nesta classe que estão definidos os métodos para o calculo do tempo real, estimado, custo real e estimado, bem como métodos para devolver o motorista mais perto e estatísticas da aplicação. Sempre que necessário a UMeR delega a gestão dos dados(ler, guardar, atualizar, apagar) na classe da base de dados (BDInterface). Após login efetuado com sucesso a variável de instância atorloggado é inicializada com o utilizador que acabou de iniciar sessão. É com base nesta variável que a interface da aplicação(UMeRApp) decide quais os menus a apresentar.

```
private BDInterface baseDeDados;  
private AtorInterface atorLoggado;  
private int tentativasDeLoginFalhadas;
```

Para o calculo do tempo real de uma viagem decidimos que este seria a multiplicação dos factores de fiabilidade, destreza, transito e meteorologia com a duração estimada da viagem.

Para o cálculo do custo real da viagem, decidimos que se a diferença entre o tempo estimado e o real fosse superior a 25% então o cliente pagaria o custo estimado, se o tempo real fosse inferior 25% do estimado o valor da viagem seria recalculado.

Este método apenas está a beneficiar o cliente, pois se o motorista demorar mais do que o esperado, o cliente pagará o custo estimado, por outro lado se o motorista demorar menos 25% do tempo estimado o valor é ajustado, sendo que pagará menos do que o previsto.

```
public double custoRealViagem(){
    if((duracaoEstimada/DuracaoReal) >= 1 && Real < estimada * 0.75){
        return (custoEstimado(distancia, precoUnitario) *
            duracaoRealDaViagem / duracaoEstimada);
    }
    else {
        return custoEstimado(distancia, precoUnitario) ;
    }
}
```

## 2.12 UMeRApp

Esta é a classe responsável pela interface apresentada a um utilizador, para que possa desfrutar ao máximo da aplicação, como veremos no capítulo seguinte. É também a classe responsável por gravar o estado da aplicação e ler esse mesmo a cada vez que a aplicação é fechada e reiniciada, respectivamente.

```
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;
```

## 3. Funcionamento da Aplicação UMeR

### 3.1 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 3.1:** Menu Inicial

### 3.2 Menu Inicial-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 3.2:** 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

## 3.3 Menu Inicial - Iniciar Sessão

### 3.3.1 Funcionalidades de 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 3.3: Menu de Cliente

#### 1. Solicitar Viagem

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 específico.

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

Menu para inserir as coordenadas

```
*****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

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

```
*****Lista de Motoristas*****
1 - Email: manuel@manuel.com | Nome: Manuel César | Classificação: 70 | Grau de cumprimento de Horário: 98 | Total
2 - Email: antonio@antonio.com | Nome: António César | Classificação: 90 | Grau de cumprimento de Horário: 100 | Total
3 - Email: joana@joana.com | Nome: Joana César | Classificação: 89 | Grau de cumprimento de Horário: 50 | Total
4 - Email: maria@maria.com | Nome: Maria César | Classificação: 80 | Grau de cumprimento de Horário: 67 | Total
5 - Email: rita@rita.com | Nome: Rita César | Classificação: 50 | Grau de cumprimento de Horário: 63 | Total
6 - Email: carlos@carlos.com | Nome: Carlos César | Classificação: 70 | Grau de cumprimento de Horário: 73 | Total
7 - Email: acres@acores.com | Nome: João César | Classificação: 37 | Grau de cumprimento de Horário: 50 | Total
0 - Sair
```

Selecione uma das opções acima ou 0 caso pretenda voltar ao menu anterior

Menu com os dados dos táxis disponíveis



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 3.4:** 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 3.5:** Menu de Erro: Cliente está em viagem

## 2. Visualizar Histórico de viagens

O Cliente poderá visualizar o seu histórico de todas viagens efetuadas e poderá escolher como quer ver a informação ou entre datas ou todas as viagens.

```

*****Menu Historico*****
Escolha uma das seguintes opções:
  1 - Visualizar todos
  2 - Entre duas datas
  0 - Sair
*****
Opção:

```

Menu para escolher como  
visualizar histórico

```

Insira a data inicial (yyyy-mm-dd):
2017-01-01
Insira a data final (yyyy-mm-dd):
2017-06-02
Motorista: acores@acores.com | Data da viagem: 2017-03-27T00:00 | Distancia: 50.0 | Tempo da Via

```

Menu ver histórico entre datas

### 3. Classificar Viagens

O menu para classificação de viagem, servirá para o cálculo da média de classificação geral do Motorista.

```
*****Lista de Viagens por classificar*****
1 - Motorista: manuel@manuel.com | Data da viagem: 2017-06-03T16:34:38.415 | Distancia: 27.71463235330627 | Tem
0 - Sair
Selecione uma das opções acima ou 0 caso pretenda voltar ao menu anterior

*****Classificar Viagem*****
Motorista: manuel@manuel.com | Data da viagem: 2017-06-03T16:34:38.415 | Distancia: 27.71463235330627
Introduza classificacao (1 - 100):
80
```

Menu para Cliente classificar a viagem

Menu para classificação da viagem

### 4. Atualizar Localização

O menu para atualizar localização serve para o cliente após fazer uma viagem, atualizar para uma nova localização em que se encontra para efetuar uma nova viagem.

```
Introduza a coordenada x:
34
Introduza a coordenada y:
43
Introduziu as coordenadas x: 34.0 y: 43.0 com sucesso!
Localizacao atualizada
```

**Figura 3.6:** Atualizar Localização

### 5. Ver dados pessoais

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 3.7:** Menu de visualização e edição dos dados Pessoais

### 3.3.2 Funcionalidades de 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 3.8:** Menu de Motorista

#### 1. Gerir Viagens

Após escolher gerir uma viagem será apresentado o menu com a opção de terminar viagem, caso Ao efetuar este passo o motorista passa a estar num disponível, isto é fica apto para receber novas viagens.

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

Menu do motorista para terminar uma viagem

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

#### 2. Gerir Horário de trabalho

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

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 - 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 do motorista para terminar horario de trabalho

Menu após iniciar horário de trabalho

### 3. Visualizar Histórico de Viagens

O menu Visualizar Histórico de viagens permite ao motorista escolher em que tipo de formato prefere ver o histórico ou entre datas ou o total.

```

Hsitorico de Viagens efetuadas:
Cliente email: daniel@daniel.com | Data da viagem: 2017-04-27T00:00 | Distancia: 20.0 | Tempo da Viagem estimado: 15.0 |
Cliente email: celia@celia.com | Data da viagem: 2017-03-27T00:00 | Distancia: 50.0 | Tempo da Viagem estimado: 50.0 | T
Cliente email: marcia@marcia.com | Data da viagem: 2017-02-27T00:00 | Distancia: 50.0 | Tempo da Viagem estimado: 100.0

*****Menu - Motoristas*****

```

**Figura 3.9:** Menu de visualização de Histórico de viagens

### 4. Visualizar 10 melhores Clientes

Esta opção permite ao motorista visualizar quais os seus melhores 10 clientes, isto é os clientes que gastam mais dinheiro em viagens.

```

***** Top 10 clientes *****
Nome: Márcia | email: marcia@marcia.com | total gasto (euros): 300.0
Nome: Celia | email: celia@celia.com | total gasto (euros): 291.34462500186197
Nome: Daniel Faria | email: daniel@daniel.com | total gasto (euros): 40.0

*****Menu - Motoristas*****

```

**Figura 3.10:** Menu de visualização de 10 melhores Clientes do Motorista

### 5. Registar Veiculo

Esta opção permite ao motorista associar um veiculo a si mesmo. Só é permitido que um motorista tenha um veiculo associado a si.

```

*****Registar Veiculos*****
Escolha uma das seguintes opções:
  1 - Moto
  2 - Carro Ligeiro
  3 - Carrinha
  4 - Moto Com Fila de Espera
  5 - Carro Ligeiro Com Fila de Espera
  6 - Carrinha Com Fila de Espera
  7 - Voltar para trás
  0 - Sair
*****
Opção: |

Matricula: rt-54-87
Marca: bmw
Veiculo Registado na UMeR!
*****Menu - Motoristas*****

```

Menu para escolher tipo de veiculo

Menu de inserção de dados no novo veiculo

## 6. Remover Veiculo

Esta opção permite ao motorista remover um veiculo associado a si.

```
Veiculo removido
*****Menu - Motoristas*****
```

**Figura 3.11:** Menu para remoção do veiculo do motorista

### 3.3.3 Funcionalidades de Admin

O Administrador da aplicação tem a funcionalidade principal de visualizar estatísticas, tais como ver uma lista de todos os utilizadores atualizados, os veiculos, o histórico, a lista dos 10 clientes que mais gastam, os motoristas com mais desvios de tempo e custo e ainda poderá ver e editar dados pessoais.

```
***** Menu - Admin *****
Escolha uma das seguintes opções: |
  1 - Ver Lista dos utilizadores registados
  2 - Ver Lista dos Veiculos Registados
  3 - Visualizar Histórico de viagens
  4 - Ver Lista dos clientes que mais gastam
  5 - Ver Lista de motoristas com mais desvios de tempo
  6 - Ver Lista de motoristas com mais desvios de custo
  7 - Faturacao por Motorista
  8 - Ver Dados Pessoais
  0 - Sair
*****
Opção:
```

**Figura 3.12:** Menu do Administrador

Mostramos de seguida o exemplo de execução de cada opção do menu:

## 1. Ver lista dos utilizadores Registados

```
***** Lista de Clientes *****
Nome: Adriana Pereira | Email: adriana@adriana.com
Nome: Nuno Faria | Email: nuno@nuno.com
Nome: Daniel Faria | Email: daniel@daniel.com
Nome: Rui | Email: rui@rui.com
Nome: Célia | Email: celia@gmail.com
Nome: Xavier Francisco | Email: xavier@xavier.com
Nome: Márcia | Email: marcia@marcia.com
Nome: Filipa Faria | Email: filipa@filipa.com
Nome: Andreia Faria | Email: andreia@andreia.com
Nome: Celia | Email: celia@celia.com
Nome: Luis | Email: luis@luis.com
Nome: Nazare Faria | Email: nazare@nazare.com
Nome: Carlos Faria | Email: carlosfaria@gmail.com
Nome: Carla Faria | Email: carla@carla.com

***** Lista de Motoristas *****
Nome: João César | Email: acores@acores.com
Nome: Carlos César | Email: carlos@carlos.com
Nome: Rita César | Email: rita@rita.com
Nome: Maria César | Email: maria@maria.com
Nome: Joana César | Email: joana@joana.com
Nome: António César | Email: antonio@antonio.com
Nome: Manuel César | Email: manuel@manuel.com

***** Menu - Admin *****
Escolha uma das seguintes opções:
1 - Ver Lista dos utilizadores registados
2 - Ver Lista dos Veiculos Registados
3 - Visualizar Histórico de viagens
4 - Ver Lista dos clientes que mais gastam
5 - Ver Lista de motoristas com mais desvios de tempo
6 - Ver Lista de motoristas com mais desvios de custo
7 - Faturacao por Motorista
8 - Ver Dados Pessoais
0 - Sair
*****
Opção:
```

**Figura 3.13:** Menu do Administrador

## 2. Ver lista dos Veiculos registados

```
***** Lista de Veiculos *****
Matricula: qt-76-22 | Marca: Mercedes | Tipo: CarroLig
Matricula: 23-az-20 | Marca: Honda | Tipo: Moto
Matricula: 24-az-28 | Marca: Honda | Tipo: Carrinha
Matricula: qt-34-22 | Marca: Renault | Tipo: Carrinha
Matricula: 25-76-22 | Marca: Fiat | Tipo: Moto
Matricula: 23-az-24 | Marca: Toyota | Tipo: CarroLig
Matricula: 25-tr-19 | Marca: BMW | Tipo: CarroLig
Matricula: 09-bm-09 | Marca: Audi | Tipo: CarroLig
Matricula: 25-fb-20 | Marca: Volvo | Tipo: Carrinha

***** Menu - Admin *****
```

**Figura 3.14:** Ver lista dos veiculos registados

### 3. Visualizar histórico de viagens

Neste menu o Administrador poderá visualizar todo o histórico ou escolher entre datas, neste caso mostramos como funciona entre datas:

```
Insira a data inicial (yyyy-mm-dd):
2017-05-10
Insira a data final (yyyy-mm-dd):
2107-05-30
Cliente email: carlosfaria@gmail.com | Motorista email: maria@maria.com | Data da viagem: 2017-06-12T00:00 | Distancia: 20.0 | T
Cliente email: celia@gmail.com | Motorista email: maria@maria.com | Data da viagem: 2017-06-03T00:00 | Distancia: 40.0 | Tempo c
Cliente email: andreia@andreia.com | Motorista email: antonio@antonio.com | Data da viagem: 2017-06-02T00:00 | Distancia: 130.0
Cliente email: luis@luis.com | Motorista email: manuel@manuel.com | Data da viagem: 2017-05-30T00:00 | Distancia: 100.0 | Tempo
Cliente email: nazare@nazare.com | Motorista email: manuel@manuel.com | Data da viagem: 2017-05-29T00:00 | Distancia: 100.0 | Te
Cliente email: nuno@nuno.com | Motorista email: manuel@manuel.com | Data da viagem: 2017-05-27T00:00 | Distancia: 50.0 | Tempo c
Cliente email: carla@carla.com | Motorista email: rita@rita.com | Data da viagem: 2017-05-23T00:00 | Distancia: 100.0 | Tempo de
Cliente email: filipa@filipa.com | Motorista email: joana@joana.com | Data da viagem: 2017-05-14T00:00 | Distancia: 12.0 | Tempc

*****Menu Historico*****
Escolha uma das seguintes opções:
1 - Visualizar todos
2 - Entre duas datas
0 - Sair
*****
Opção:
```

**Figura 3.15:** Ver lista de histórico de viagens

### 4. Ver lista dos 10 clientes que mais gastam

Esta funcionalidade permite listar o top dos 10 melhores clientes da aplicação. Este top é construído com base nos históricos terminados (viagens em precurso não são contabilizados para este cálculo).

```
***** Top 10 clientes *****
Nome: Márcia | email: marcia@marcia.com | total gasto (euros): 300.0
Nome: Celia | email: celia@celia.com | total gasto (euros): 150.0
Nome: Célia | email: celia@gmail.com | total gasto (euros): 120.0
Nome: Andreia Faria | email: andreia@andreia.com | total gasto (euros): 90.0
Nome: Carla Faria | email: carla@carla.com | total gasto (euros): 90.0
Nome: Luis | email: luis@luis.com | total gasto (euros): 90.0
Nome: Nazare Faria | email: nazare@nazare.com | total gasto (euros): 90.0
Nome: Nuno Faria | email: nuno@nuno.com | total gasto (euros): 90.0
Nome: Xavier Francisco | email: xavier@xavier.com | total gasto (euros): 70.0
Nome: Daniel Faria | email: daniel@daniel.com | total gasto (euros): 40.0

***** Menu - Admin *****
```

**Figura 3.16:** Ver lista dos 10 clientes que mais gastam

### 5. Ver lista de motoristas com mais desvios de tempo

A lista apresentada é ordenada por ordem crescente do grau de cumprimento de tempo. Este valor é calculado com base nos históricos (terminados). Para cada histórico se o tempo estimado for maior que o tempo real, o grau será 100, se o tempo estimado for menor que o tempo real o grau é calculado pela divisão entre o tempo real e o tempo estimado. O grau de cumprimento por motorista é a média do grau de cumprimento dos seus históricos.

Quanto maior o grau de cumprimento de tempo, melhor será para o cliente uma vez que este grau indica que o motorista efetua viagens com um tempo menor ou igual daquele que foi estimado.

Para o administradores da plataforma quanto menor o grau de cumprimento de tempo pior uma vez que as viagens demoraram mais tempo do que o estimado e por consequência possibilitará a realização de menos viagens (menos faturação).

```

***** Top 5 Motoristas com mais desvios de tempo *****
Nome: Joana César | email: joana@joana.com | Grau de cumprimento de horario: 50
Nome: João César | email: acores@acores.com | Grau de cumprimento de horario: 50
Nome: Rita César | email: rita@rita.com | Grau de cumprimento de horario: 63
Nome: Maria César | email: maria@maria.com | Grau de cumprimento de horario: 67
Nome: Carlos César | email: carlos@carlos.com | Grau de cumprimento de horario: 73

***** Menu - Admin *****

```

**Figura 3.17:** Ver lista de motoristas com mais desvios de tempo

## 6. Ver lista de motoristas com mais desvios de custo

A lista apresentada é ordenada por ordem crescente do grau de cumprimento de custo. Este valor é calculado com base nos históricos (terminados). Para cada histórico se o custo real for maior que o custo estimado, o grau será 100, se o custo real for menor que o custo estimado o grau é calculado pela divisão entre o custo real e o custo estimado. O grau de cumprimento por motorista é a média do grau de cumprimento dos seus históricos.

Quanto menor o grau de cumprimento de custo, melhor será para o cliente uma vez que este grau indica que o motorista efetua viagens com um custo menor do que aquele que foi estimado.

Para o administradores da plataforma quanto menor o grau de cumprimento de custo pior uma vez que as receitas serão menores em relação às receitas estimadas.

---

```

***** Top 5 Motoristas com mais desvios de custos *****
Nome: Carlos César | email: carlos@carlos.com | Grau de cumprimento de custo: 77
Nome: João César | email: acores@acores.com | Grau de cumprimento de custo: 96
Nome: Rita César | email: rita@rita.com | Grau de cumprimento de custo: 100
Nome: Manuel César | email: manuel@manuel.com | Grau de cumprimento de custo: 100
Nome: Joana César | email: joana@joana.com | Grau de cumprimento de custo: 100

***** Menu - Admin *****
Escolha uma das seguintes opções:

```

**Figura 3.18:** Ver lista de motoristas com mais desvios de custos

## 7. Faturação por Motorista

Esta funcionalidade permite ver os dados entre datas ou a lista com todos os dados. Decidiu-se que faria mais sentido apresentar a faturação por motorista, visto que o motorista só tem um veículo então a informação seria a mesma.

---

```

***** Faturacao por Motorista *****
Nome: João César | email: acores@acores.com | Faturacao: 490,00
Nome: Manuel César | email: manuel@manuel.com | Faturacao: 270,00
Nome: Maria César | email: maria@maria.com | Faturacao: 157,00
Nome: Rita César | email: rita@rita.com | Faturacao: 90,00
Nome: António César | email: antonio@antonio.com | Faturacao: 90,00
Nome: Carlos César | email: carlos@carlos.com | Faturacao: 70,00
Nome: Joana César | email: joana@joana.com | Faturacao: 18,00

```

**Figura 3.19:** Ver faturação por motorista



## 4. Conclusões

Com a realização deste trabalho pode-se concluir que ainda há aspetos a melhorar, tais como a apresentação dos menus, poder-se-ia ter melhorado a organização, por exemplo as estatísticas que são acedidas pelo administrador deviam ter um menu de introdução de estatísticas.

Pode-se concluir que as funcionalidades de registar um utilizador quer cliente quer motorista, assim como a validação das suas credenciais está a funcionar. É possível a um motorista inserir uma nova viatura, ficando assim esta associada a esse motorista.

O cliente poderá solicitar uma viagem, bastando para tal inserir as coordenadas de destino, depois poderá escolher uma viatura específica ou então a mais próxima. Poderá ainda classificar uma viagem, mas só depois de esta ser terminada pelo motorista. No perfil de cliente é possível ver a listagem das viagens efetuadas, quer entre datas ou ver tudo.

No perfil de motorista também poderão ser visualizadas as listagens das viagens efetuadas quer por data quer ou a lista inteira, esta lista está ordenada de forma descendente. O cliente tem a opção de terminar uma viagem e atualizar se está em horário de trabalho.

No perfil de administrador é possível consultar o total faturado por motorista num determinado período ou a apresentação da lista completa. É possível visualizar a listagem do 10 clientes que mais gastam, e ainda a listagem dos 5 motoristas que apresentam mais desvios entre os valores previstos para as viagens e o valor final faturado.

O estado da aplicação é gravado num ficheiro para que seja possível retomar mais tarde a execução do programa.

Adicionalmente foram implementados os fatores de aleatoriedade para calcular o tempo real de uma viagem. Infelizmente não conseguimos utilizar as filas de espera dos veículos, assim como a criação de novas empresas. Percebemos também que os motoristas deveriam poder criar vários veículos e fazer a gestão do veículo que estão a utilizar no momento. Neste momento o motorista para reutilizar o veículo tem de o voltar a registar. O administrador deveria poder gerir clientes motoristas e veículos, neste momento só existe um administrador e não tem a possibilidade de inserir outros administradores. Tivemos um problema de interpretação que só foi percebido no fim e já não tivemos tempo para alterar que foi a indicação do total faturado por viatura e nós fizemos por motorista. Essa implementação seria possível, mas teríamos de não remover os veículos da coleção veículos quando o motorista faz remover veículo e seria necessário adicionar ao histórico a matrícula do veículo para poder buscar esses dados.

Para concluir, consideramos que as funcionalidades mais importantes foram implementadas com sucesso, embora ainda possam ser melhoradas.

### 4.1 Trabalho Futuro

Como trabalho futuro a aplicação poderá vir a permitir registo de empresas e a utilização das filas de espera dos veículos, para o primeiro ponto ter-se-ia de adicionar um novo tipo de utilizador e dar-lhe permissão para conseguir gerir os seus motoristas e veículos. Para o segundo

seria necessário alterar a requisição de um veículo e se ele permitisse fila de espera adicionar o cliente a essa fila e alterar o estado do cliente para em fila de espera, caso o veículo estivesse ocupado.

O administrador poderá gerir clientes, motoristas e veículos. Poder-se-à fazer alterações de modo a que seja possível indicar o total faturado por veículo.