

Projeto de POO - UMER

Grupo 52

Vítor Castro (A77870), Marcos Pereira (A79116), Sérgio Jorge (A77730)

(da esquerda para a direita)



Resumo

Neste relatório faremos uma análise do projeto de Programação Orientada aos Objetos, no qual o objetivo era desenvolver um programa, em Java, que fizesse a gestão da *UMER*, uma empresa de transporte de passageiros. Assim, este documento apresenta detalhadamente a abordagem tomada ao problema proposto pela equipa docente da UC.

Conteúdo

1	Introdução	2
2	Problema	2
3	Solução	3
3.1	User	4
3.2	Vehicle	4
3.3	Trip	4
3.4	IO	4
3.5	Implementação - UMER	4
3.6	Resultado Final	5

4	Manual de Utilização	5
4.1	Criar entidades	6
4.1.1	Cliente	7
4.1.2	Condutor	7
4.1.3	Veículo	7
4.2	Cliente	8
4.2.1	Iniciar sessão	8
4.2.2	Obter registo de viagens	9
4.2.3	Iniciar viagem	9
4.2.4	Avaliar motorista	10
4.2.5	Terminar sessão	10
4.3	Condutor	11
4.3.1	Iniciar sessão	11
4.3.2	Obter registo de viagens	12
4.3.3	Alterar disponibilidade	12
4.3.4	Terminar sessão	12
4.4	Veículo	13
4.4.1	Ver finanças	13
4.5	Outras funcionalidades	14
4.5.1	Guardar o estado do programa	14
4.5.2	Associar condutor a veículo	14
4.5.3	5 piores condutores	14
4.5.4	10 clientes que mais gastam	15
4.5.5	Avançar no tempo	15
5	Conclusões	15

1 Introdução

Este projeto foi realizado com o objetivo de desenvolver um programa responsável por toda a gestão de uma empresa de taxis. Foram, então, propostas pelos professores algumas funcionalidades com contexto real enquadradas no tema, às quais o programa deve responder com sucesso. A implementação destas permitiram consolidar e adquirir conhecimentos ao nível da sintaxe de programação em Java e também incentivaram à exploração de estruturas de dados e de APIs características desta linguagem. Assim, de modo a facilitar a compreensão do projeto, o relatório está dividido da seguinte forma:

Secção 2 : Problema;

Secção 3 : Solução;

Secção 4 : Conclusão.

2 Problema

Neste projeto de POO pede-se para desenvolver um programa capaz de auxiliar na gestão de uma empresa de transporte de pessoas. Assim, este deve ser capaz de:

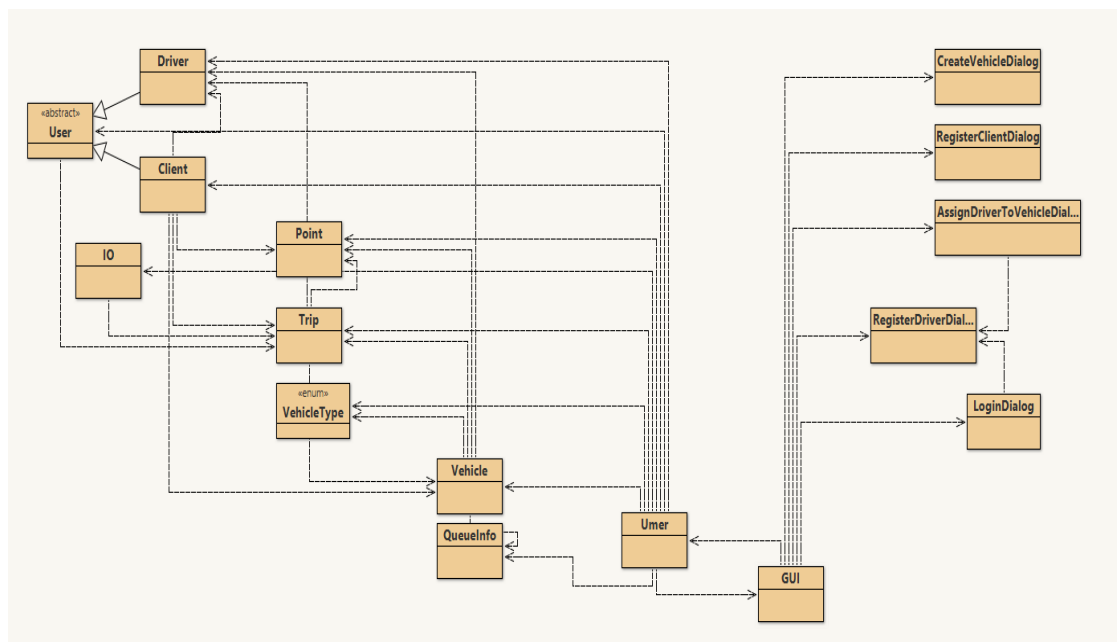
- Registrar um utilizador (cliente ou motorista);
- Implementar login no sistema;
- Criar viaturas;
- Associar motoristas a viaturas;

- Um cliente pode solicitar uma viagem escolhendo uma viatura específica ou a mais próxima de si;
- Classificar o motorista, após a viagem;
- Registrar um utilizador (cliente ou motorista);
- Possibilidade do cliente conseguir ver as viagens que já fez;
- Possibilidade do motorista conseguir ver as viagens que já fez;
- Indicar o total faturado pela viatura ou pela empresa;
- Listar os 10 clientes que mais gastam;
- Listar os 5 motoristas que apresentam mais desvios entre valores previstos para a viagem e o valor final faturado;
- Gravar o estado do programa em ficheiro.

3 Solução

A nossa solução foi implementada com base em diferentes classes dos quais destacamos:

- User;
 - Driver;
 - Client;
- Vehicle;
- Trip;
- IO.



3.1 User

Esta é uma superclasse que define variáveis como: email, nome, password, morada e data de nascimento do utilizador. A razão para a implementação de uma hierarquia é porque todos os campos referidos são comuns a motoristas e a clientes pelo que não há a necessidade de repetir código em diferentes classes. Está definida como abstrata porque não haverá necessidade de instanciar Users.

Driver A classe referente aos motoristas inclui as variáveis definidas na superclasse mas também o grau de cumprimento de horário, a classificação do motorista, o total de quilómetros feitos por este na empresa e um booleano que informa sobre a disponibilidade do motorista.

Client Na classe de clientes são também incluídas as variáveis definidas em User. Além disso, define-se o ponto/localização do cliente e o dinheiro total gasto por este. Nesta classe,

3.2 Vehicle

Em Vehicle, classe relativa aos objetos Veículos da empresa, são definidas variáveis como: finanças do veículo, posição, ID, tipo de veículo, motorista atual e dois booleanos relativos a fila de espera e ocupação.

3.3 Trip

Na classe Trip estão as variáveis relativas às viagens e, a cada viagem, deve estar associado um veículo, um motorista, dois pontos (origem e destino), duração estimada, duração real, custo da viagem, hora de partida e chegada.

3.4 IO

Esta é a classe responsável pelas operações de Input/Output do programa. Assim, através dela, é-nos possível guardar o estado da aplicação em determinado momento e é também possível recuperar esse estado posteriormente. O estado é guardado em ficheiros presentes na diretoria do programa.

3.5 Implementação - UMER

Na implementação deste programa, optamos por criar uma interface gráfica do utilizador (GUI) de modo a facilitar a interação da aplicação com as funcionalidades exigidas. Usamos três *hashmaps* para guardar a informação relativa a motoristas, clientes e veículos. A escolha recaiu no uso deste tipo de estrutura de dados por nos possibilitar ganhos de performance relativamente a listas. Por sua vez, as listas são usadas, no nosso programa, para guardar a informação relativa a viagens já completadas e viagens a decorrer.

Primeiramente, esta classe é responsável por métodos relativos a *login* e *logout*, instancia ou cria objetos das classes motoristas, clientes e veículos e adiciona-os às estruturas de dados respetivas e trata de associar um condutor a um veículo. É, então, que se torna possível a criação ou o começo de uma viagem.

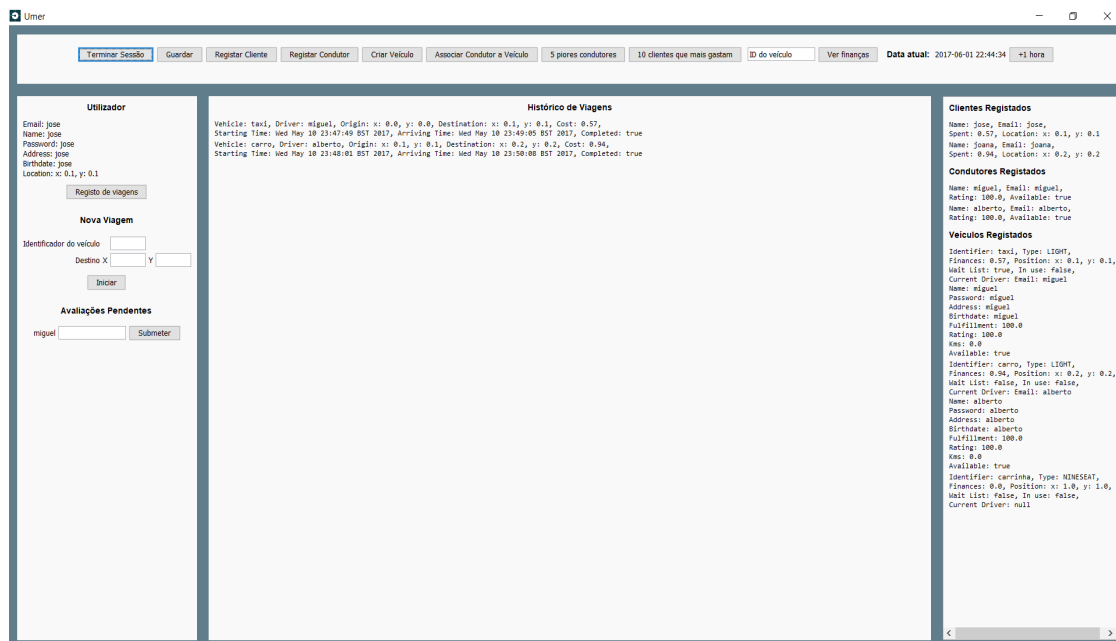
No que diz respeito à criação de viagens, é verificado se há algum veículo disponível ou se o que foi pedido pelo cliente está disponível. Em caso afirmativo, a viagem é adicionada às listas de viagens, as posições dos objetos são atualizadas, as finanças e o custo são adicionadas ao veículo e ao cliente e o cliente avalia o condutor. Em caso negativo, se existir algum veículo com fila de espera, o cliente é adicionado à sua fila de espera. É importante referir que a nossa aplicação faz uso de um relógio que nos permite marcar a hora de início e de fim da viagem. Assim, é-nos possível saber quando a viagem acaba pelo que, nesse momento, podemos tirar a viagem da lista de viagens a decorrer. Está também implementado o método *fastforward* que adianta o relógio.

Para cliente e motorista terem acesso às viagens que já fizeram percorre-se a lista do historial de viagens e vê-se em que viagens é que estes participaram e é apenas feito um ToString dessa informação.

Em relação aos tops, decidimos converter as *hashmaps* de clientes e de motoristas para *arraylists* que, posteriormente, são ordenados. A seguir, passamos os N primeiros elementos dos *arraylists* (já ordenados) para *linkedhashmaps* aos quais damos toString.

O estado da aplicação é guardado a partir de um botão na GUI que chama os métodos responsáveis por esse efeito, presentes na classe IO. Relativamente a carregar o estado, é feita a chamada aos métodos, automaticamente, quando a GUI é aberta.

3.6 Resultado Final



4 Manual de Utilização

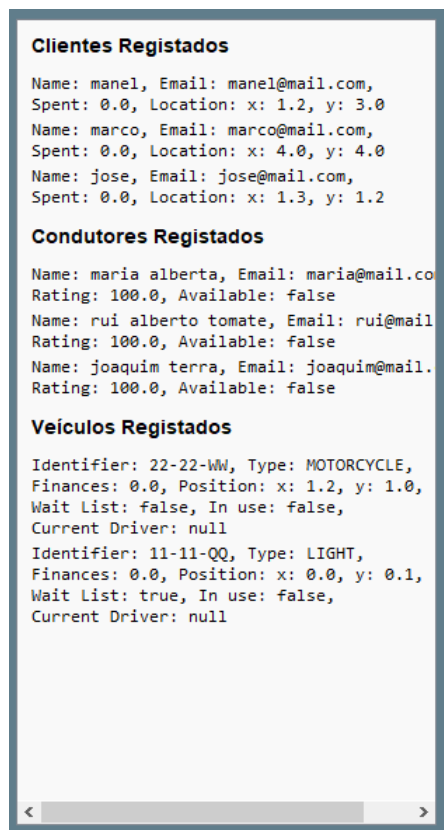
De seguida, se destacam os elementos que permitem proceder à utilização completa do programa, sendo que cada passo será acompanhado da devida ilustração. Todas as funcionalidades são apresentadas na mesma janela, sendo por isso fácil ter uma perspetiva completa do que acontece a cada momento.

- Criar entidades;
 - Cliente;
 - Condutor;
 - Veículo;
- Cliente;
 - Iniciar sessão;
 - Obter registo de viagens;
 - Iniciar viagem;
 - * Para o veículo mais próximo;
 - * Para o veículo específico.
 - Avaliar motorista;

- Terminar sessão.
- Condutor;
 - Iniciar sessão;
 - Obter registo de viagens;
 - Alterar disponibilidade;
 - Terminar sessão.
- Veículo;
 - Ver finanças.
- Outras funcionalidades.
 - Guardar o estado do programa;
 - Associar condutor a veículo;
 - 5 piores condutores;
 - 10 clientes que mais gastam;
 - Avançar no tempo.

4.1 Criar entidades

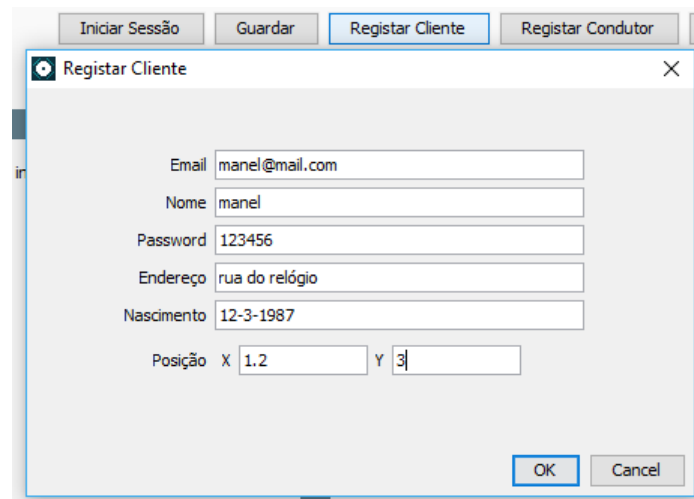
Foram disponibilizados botões específicos para cada entidade, podendo esta ser um Cliente, um Condutor ou um Veículo. Procedeu-se desta forma pois é mais intuitivo para um possível futuro utilizador. Todos as entidades registadas serão apresentadas no lado direito da janela do programa, como é visível na seguinte figura.



Entidades registadas

4.1.1 Cliente

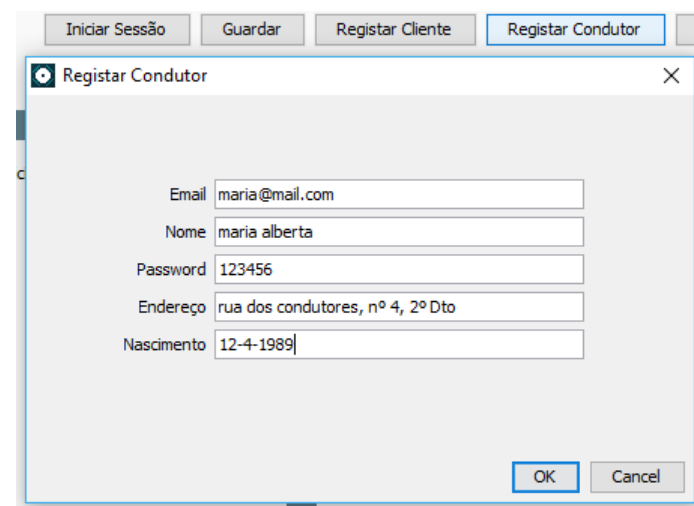
Para proceder à criação de um cliente, bastará clicar no botão "Registrar Cliente", facilmente identificável no topo da interface. Depois disso, o utilizador terá de preencher os campos "Email", "Nome", "Password", "Endereço" (morada), "Nascimento" (data em qualquer formato) e "Posição" (X e Y). De seguida, clicará em "OK" e ficará registado no sistema.



Criação de um Cliente

4.1.2 Condutor

Junto do botão para criação de um Cliente existe também o botão "Registrar Condutor", através do qual será possível criar um novo Condutor. Obviamente, a posição do motorista não tem interesse, pelo que não é pedida.



Criação de um Condutor

4.1.3 Veículo

Como nos anteriores, existe também um botão para a criação de um veículo - "Criar veículo". Será pedido o "Identificador" (do Veículo), o "Tipo" do mesmo (LIGHT para carros 5 lugares, NINESEAT para carros 9 lugares e MOTORCYCLE para motos), a "Posição" em que se encontra e o utilizador deverá seleccionar o campo "Lista de Espera" caso pretenda que aquele veículo tenha essa característica.

Registrar Condutor

Identificador: 11-11-QQ

Tipo: LIGHT

Posição X: 0 Y: 0.1

Lista de espera: ☒

OK Cancel

Criação de um Veículo

4.2 Cliente

Nos seguintes pontos se descrevem todas as funcionalidades disponíveis apenas quando o cliente inicia a sua sessão. O cliente com sessão iniciado terá a sua zona pessoal, apresentada no lado esquerdo do ecrã, cujo aspeto está presente na imagem seguinte.

Utilizador

Email: manel@mail.com
 Name: manel
 Password: 123456
 Address: rua do relógio
 Birthdate: 12-3-1987
 Location: x: 1.2, y: 3.3

Registo de viagens

Nova Viagem

Identificador do veículo:

Destino X: 1.2 Y: 3.3

Iniciar

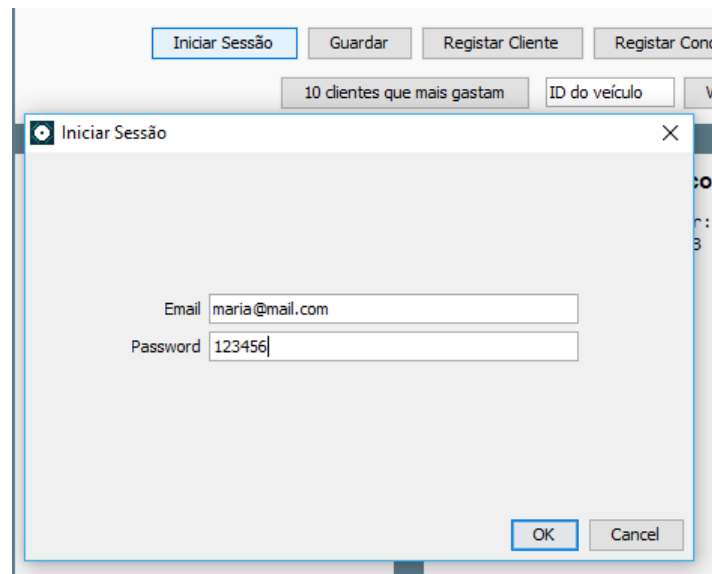
Avaliações Pendentes

maria@mail.com Submeter

Área de cliente

4.2.1 Iniciar sessão

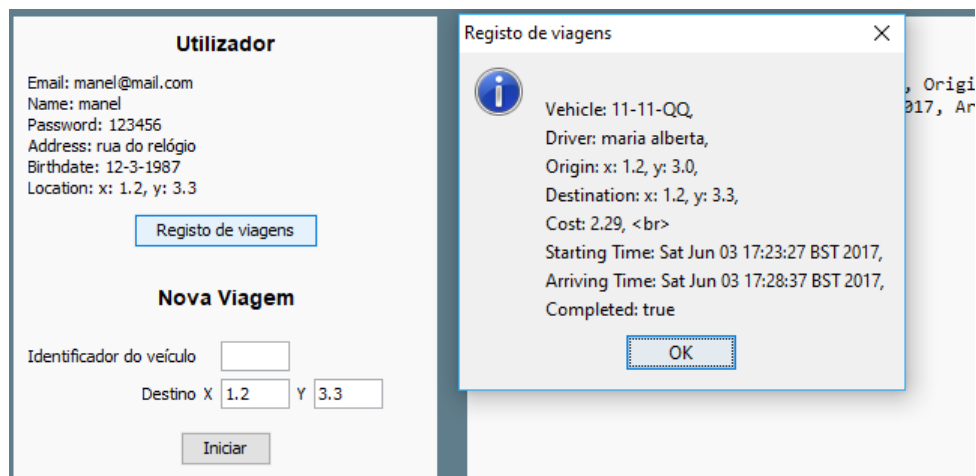
O cliente, após clicar no botão "Iniciar Sessão", no topo da janela, deverá introduzir os dados pedidos e clicar em "OK".



Início de sessão de um Cliente

4.2.2 Obter registo de viagens

Na sua área pessoal, poderá clicar em "Registo de Viagens". Desta forma, será apresentada uma janela com a informação relativa a todas as suas viagens.



Registo de viagens de um Cliente

4.2.3 Iniciar viagem

Para o veículo mais próximo O cliente poderá iniciar uma viagem com o veículo mais próximo e, nesse caso, não precisará de inserir qualquer identificador de veículo mas apenas a localização de destino. Automaticamente o Veículo mais próximo será pedido.

Nova Viagem

Identificador do veículo

Destino X Y

Pedido de viagem a Veículo mais próximo

Para o veículo específico O cliente poderá iniciar uma viagem para um veículo específico, introduzindo o ID do veículo desejado.

Nova Viagem

Identificador do veículo

Destino X Y

Pedido de viagem a Veículo específico

4.2.4 Avaliar motorista

O cliente, após o término da viagem, avaliará o condutor no local "Avaliações Pendentes", presente na sua área pessoal. O valor inserido de avaliação deverá ser de 0 a 100.

Avaliações Pendentes

maria@mail.com

Avaliação de um Condutor

4.2.5 Terminar sessão

Poderá, no fim da sua utilização, o cliente proceder ao término da sessão, no mesmo local onde iniciou sessão, sendo que agora o botão mostra "Terminar Sessão".

Terminar Sessão Guardar 10 clientes que mais

Utilizador

Email: manel@mail.com
 Name: manel
 Password: 123456
 Address: rua do relógio
 Birthdate: 12-3-1987
 Location: x: 1.2, y: 3.3

Registo de viagens

Nova Viagem

Identificador do veículo
 Destino X Y

Avaliações Pendentes

maria@mail.com

Terminar sessão

4.3 Condutor

Serão, nos pontos seguintes, descritas todas as funcionalidades disponíveis a um Condutor registado.

4.3.1 Iniciar sessão

O condutor, após clicar no botão "Iniciar Sessão", no topo da janela, deverá introduzir os dados pedidos e clicar em "OK".

Iniciar Sessão Guardar Registrar Cliente Registrar Cond

10 clientes que mais gastam ID do veículo V

Iniciar Sessão

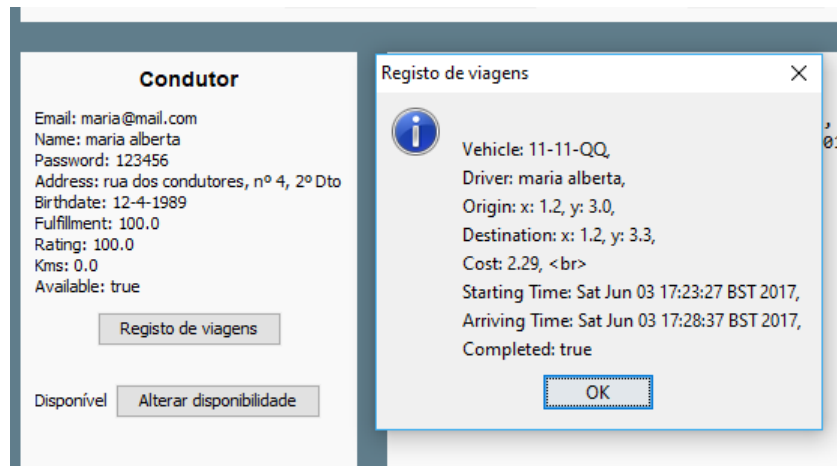
Email

Password

Início de sessão de um Condutor

4.3.2 Obter registo de viagens

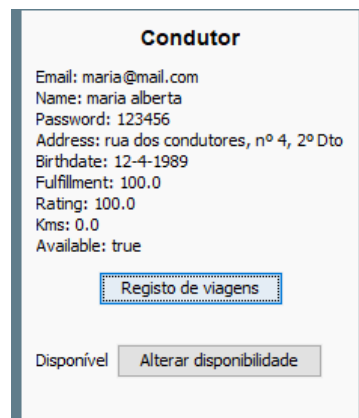
Na sua área pessoal, poderá clicar em "Registo de Viagens". Desta forma, será apresentada uma janela com a informação relativa a todas as suas viagens.



Registo de viagens de um Condutor

4.3.3 Alterar disponibilidade

O condutor poderá tomar conhecimento da sua disponibilidade atual e também alterá-la, através do botão "Alterar Disponibilidade", presente na sua área pessoal.



Disponibilidade de um Condutor

4.3.4 Terminar sessão

Poderá, no fim da sua utilização, o condutor proceder ao término da sessão, no mesmo local onde iniciou sessão, sendo que agora o botão mostra "Terminar Sessão".

Terminar Sessão
Guardar
10 clientes que mais

Utilizador

Email: manel@mail.com
Name: manel
Password: 123456
Address: rua do relógio
Birthdate: 12-3-1987
Location: x: 1.2, y: 3.3

Registo de viagens

Nova Viagem

Identificador do veículo
Destino X Y

Iniciar

Avaliações Pendentes

maria@mail.com

Submeter

Terminar sessão

4.4 Veículo

Cada veículo disponível está representado na lateral direita, sendo também apresentadas informações relativas às suas finanças, posição, lista de espera, disponibilidade e condutor.

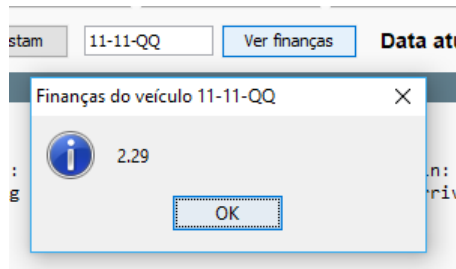
Veículos Registrados

Identifier: 22-22-WW, Type: MOTORCYCLE,
Finances: 0.0, Position: x: 1.2, y: 1.0,
Wait List: false, In use: false,
Current Driver: null
Identifier: 11-11-QQ, Type: LIGHT,
Finances: 2.29, Position: x: 1.2, y: 3.3,
Wait List: true, In use: false,
Current Driver: Email: maria@mail.com
Name: maria alberta
Password: 123456
Address: rua dos condutores, nº 4, 2º Dto
Birthdate: 12-4-1989
Fulfillment: 100.0
Rating: 100.0
Kms: 0.0
Available: true

Informação dos veículos

4.4.1 Ver finanças

É possível consultar as finanças de um veículo, colocando o seu ID no campo "ID do Veículo" no topo da janela, clicando de seguida em "Ver finanças".



Finanças do veículo por identificação

4.5 Outras funcionalidades

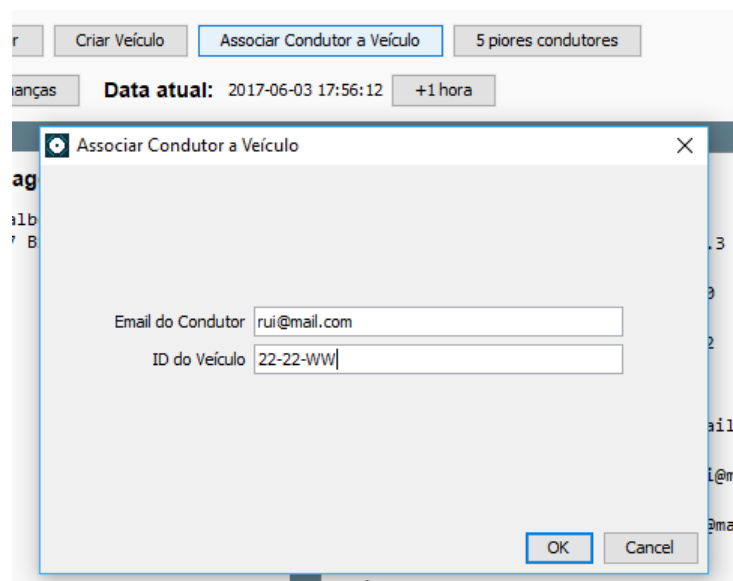
Outras funcionalidades foram disponibilizadas, algumas para permitir o teste do programa, outras para recolher informações. Estas funcionalidades podem ser acessadas no topo da interface e serão descritas de seguida.

4.5.1 Guardar o estado do programa

É possível guardar o estado do programa, através do botão "Guardar". Deve ser feito caso se queira guardar as informações da sessão atual.

4.5.2 Associar condutor a veículo

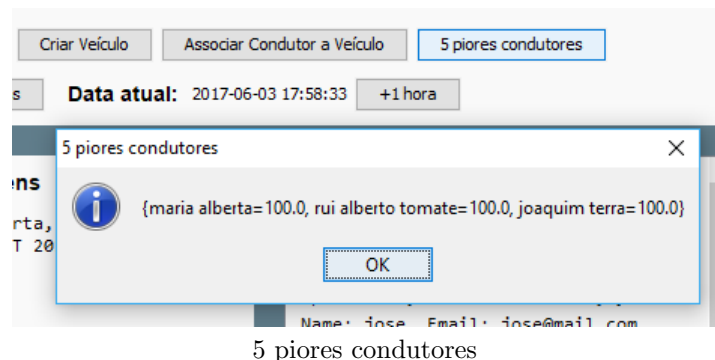
É possível associar um condutor a um veículo, tal como seria exigido numa situação de vida real. Deve ser introduzido o email do condutor e o identificador do veículo que conduzirá.



Associar condutor a veículo

4.5.3 5 piores condutores

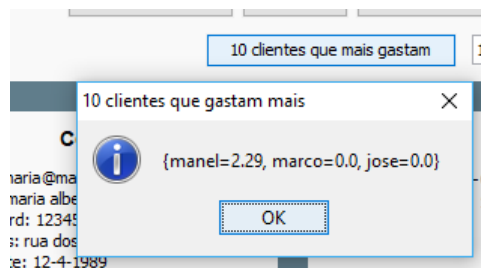
É possível consultar a lista dos 5 (ou dos existentes no sistema) piores condutores, através do botão "5 piores condutores".



5 piores condutores

4.5.4 10 clientes que mais gastam

É possível consultar a lista dos 10 (ou dos existentes no sistema) clientes mais gastadores, através do botão "10 clientes que mais gastam".



10 clientes mais gastadores

4.5.5 Avançar no tempo

É possível avançar 1 (uma) hora ao tempo real permitindo, por exemplo, o acelerar do término de uma viagem.

5 Conclusões

Este projeto serviu para aprofundarmos o conhecimento da linguagem JAVA, assim como as APIs que lhe estão associadas. Achemos que a realização de um trabalho deste tipo permite uma consolidação proveitosa da linguagem, não só em termos teóricos como também em termos práticos. Permite também melhorar as habilidades na resolução de problemas. No entanto, apesar de nos termos proposto a fazer todos os pontos do trabalho, acabamos por não ter tempo para implementar as empresas de taxis e o random dos condutores. Não temos dúvidas de que o conseguimos fazer, mas tivemos de dedicar tempo precioso a outras UCs. Fomos além do pedido e fizemos uma GUI, o que nos poderá ter custado tempo útil para fazermos os pontos pedidos que faltaram, mas acreditamos que ficamos a conhecer melhor outra parte da programação que até ao momento desconhecíamos e, assim sendo, cremos ter sido vantajoso este esforço que sabemos não contar para avaliação.