

# Algoritmo e Estrutura de Dados

## *T1 Boleias Partilhadas*

*2MIEIC1\_E*

*(20 de Novembro de 2016)*

Bárbara Silva	<b>up201505628@fe.up.pt</b>
Igor Silveira	<b>up201505172@fe.up.pt</b>
Julieta Frade	<b>up201506530@fe.up.pt</b>

# *Índice*

ShareIt	2
Solução Implementada	3
Diagrama de Classes (UML)	4
Casos de Utilização	5
Classes	5
Ficheiros	6
Programa	7
Dificuldades	8
Distribuição do Trabalho pelos Elementos do Grupo	9

# *ShareIt*

ShareIt é um sistema para a gestão de uma rede social de partilha de boleias baseado no conceito de carpooling e ridesharing. Há dois tipos de utilizadores: os registados no sistema e aqueles que utilizam o sistema ocasionalmente. Entre os utilizadores, há aqueles que desejam disponibilizar as suas viaturas, e aqueles que não têm viaturas para partilhar, mas apenas partilham as viagens. Os utilizadores com viaturas podem criar viagens e os utilizadores que possam ter interesse em partilhar a viagem toda ou trechos das viagens, podem candidatar-se aos lugares disponíveis. Sendo construído em torno do conceito de redes sociais, o sistema também privilegia a formação de grupos de partilha de viagem entre pessoas próximas entre si (“buddies”).

# *Solução Implementada*

## **1) Register**

## **2) Login**

### a) ADMIN

#### i) Users

(1) Sort by Username

(2) Sort by Name

#### ii) Trip Record

(1) Collect Payment

#### iii) Transactions

#### iv) Relationships

#### v) Stops

#### vi) Search

(1) Users

(a) By ID

(b) By Username

(2) Trips

(a) By Driver

(b) By Month

(3) Transactions

(a) By User

(b) By Month

#### vii) Run Trip

### b) DRIVER

#### i) Account

(1) Deposit

#### ii) Create Trip

#### iii) Add Buddy

### c) PASSENGER

#### i) Account

(1) Deposit

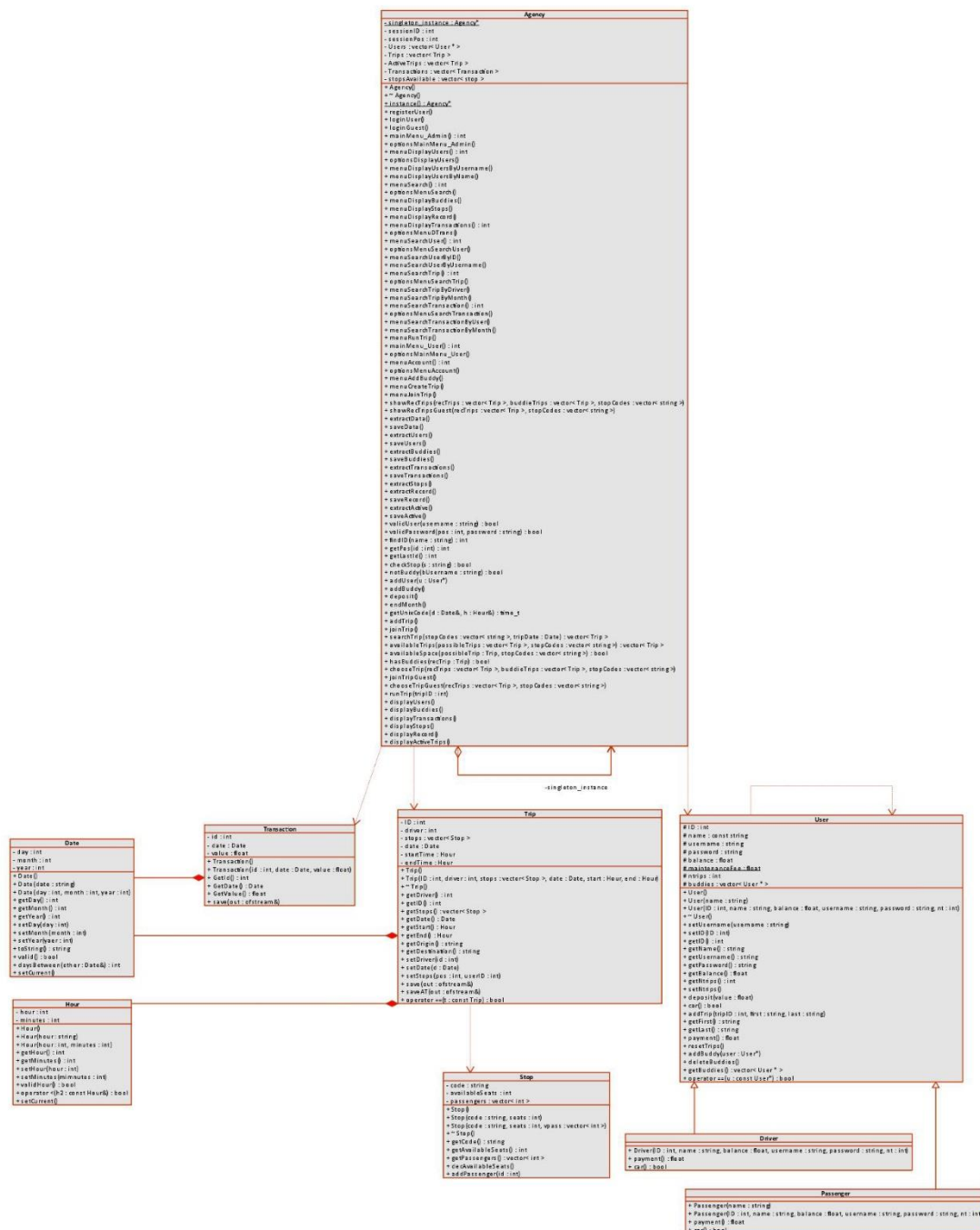
#### ii) Join Trip

#### iii) Add Buddy

## **3) Guest**

### a) Join Trip

### Diagrama de Classe (UML)



# Casos de Utilização

## Classes

### Agency

Esta é considerada a classe principal. Foi implementada com o padrão de projetos de software Singleton. Este padrão garante a existência de apenas uma instância de uma classe, mantendo um ponto global de acesso ao seu objeto.

Esta classe tem os vetores: *Users* (vetor de todos os usuários registrados), *Trips* (vetor de todas as viagens adicionadas ao sistema), *ActiveTrips* (vetor de viagens registradas mas ainda não realizadas), *Transactions* (vetor de todas as transações dos usuários para a agência), *stopsAvailable* (vetor com as paragens disponíveis na agência).

### User

Esta classe tem como atributos: *ID*, *username*, *password*, *balance* (saldo na conta do usuário), *buddies* (vetor de ids dos buddies do usuário), *ntrips* (numero de trips realizadas por o usuário), *Maintenancefee* (manutenção mensal).

Esta classe tem como derivadas a classe Driver e a classe Passenger.

### Trip

Esta classe tem como atributos: *ID*, *driver* (id do driver da viagem), *stops* (paragens desta viagem), *date* (dia da viagem), *startTime* (hora de início da viagem), *endTime* (hora de fim da viagem).

### Stop

Esta classe tem como atributos: *code* (o código da paragem), *availableSeats* (nº de lugares disponíveis na viatura, ou seja, passageiros que pode entrar nessa paragem) e *passengers* (id dos passageiros que estão dentro da viatura nessa paragem).

### Transactions

Esta classe tem como atributos: *id* (id do usuário relativo a esta transação), *date* (dia da transação), *value* (valor transferido).

### Date

Esta classe tem como atributos: *day*, *month* e *year*.

### Hour

Esta classe tem como atributos: *hour* e *minutes*.

## *Ficheiros de texto*

### **Users.txt**

Lista de todos os usuários no formato:

- *ID ; nome ; 1 se driver, 0 se passenger; balance ; username; password ; numero de viagens*

### **Record.txt**

Lista de todas as viagens adicionadas ao sistema no formato:

- *ID da viagem ; ID do driver ; paragem de inicio ; paragem de fim ; data ; hora de início ; hora de fim*

### **ActiveTrips.txt**

Lista de viagens adicionadas ao sistema mas ainda não realizadas no formato:

- *ID da viagem ; ID do driver ; [ código da primeira stop , numero de lugares disponíveis na viatura nessa stop, (ids dos passageiros dentro da viatura nessa stop); (repetir para todas as stops da trip) ] ; data ; hora de inicio ; hora de fim*

### **Stops.txt**

Lista de todas as stops disponíveis no Sistema no formato:

- *Código da Paragem ; Nome da Paragem*

### **Transactions.txt**

Lista de todas as transações dos usuários para a agência no formato:

- *ID do usuário ; data ; valor da transação*

### **Buddies.txt**

Lista das relações entre os usuários no formato:

- *ID do usuário; buddies desse usuário*

## *Programa*

O programa começa com as opções *Register*, *Login* e *Guest*.

No registo, é perguntado ao usuário se se vai registar como driver, o username desejado, a password (sendo pedida duas vezes para efeitos de comprovação visto que no ecrã não é mostrada a password como medida de segurança) e o nome. Não é possível adicionar usuários com o mesmo username. Depois de registado, o login é feito automaticamente.

O login pode ser feito como administrador com as credencias - username: admin, password: admin. O administrador tem acesso a toda a informação da agência e é aquele que tem acesso à opção correr uma viagem. Esta opção mostra no ecrã uma viagem a decorrer, mostrando também quais os usuários que entram e saem em cada paragem.

Quando é feito o login como driver ou como passenger é possível ver a informação da conta e adicionar um buddy. O que difere entre os dois é que no driver há a opção criar viagem, enquanto que no passenger há a opção juntar-se à viagem. Não é possível criar viagem para datas inválidas ou anteriores à atual. Quando um passageiro diz de onde a onde quer a viagem e quando, é lhe mostrada uma lista de viagens que passam nos sítios pretendidos nesse dia, dando prioridade às viagens que contêm buddies deste utilizador.

Quando se entra como guest, é se redirecionado imediatamente para a opção juntar-se a uma viagem. Os guests são guardados no sistema (trips e transações) com o id -1.

Os usuários têm uma conta onde podem depositar dinheiro. Este dinheiro vai ser recolhido mensalmente pelo administrador. A manutenção mensal é 10€. O driver paga a manutenção e recebe o número viagens em euros, o passenger paga a manutenção e 2€ por viagem e o guest paga sempre no final de cada viagem e o valor é 1.5€ por paragem.



## *Dificuldades*

Ao longo do projeto não houve grandes dificuldades. As maiores dúvidas surgiram na interligação de toda a informação. A classe mais desafiante foi a Trip, pois adicionar passageiros à viagem e correr a viagem são funções um pouco trabalhosas.

## *Distribuição de Trabalho Pelos Elementos*

O trabalho foi dividido igualmente por todos os membros. A distribuição do trabalho foi feita da seguinte forma:

- *Bárbara Silva* – interface de utilização; intercalação dos menus; registos e logins; adicionar passageiro a uma viagem.
- *Igor Silveira* – exceções; obtenção da hora atual para a robustez do programa; correr uma viagem; documentação no Doxygen.
- *Julieta Frade* – extrair e guardar informação dos ficheiros; display de informação; ordenação e procura.

O que não foi mencionado em cima foi feito em parceria.