Aplicação TrazAqui!

Trabalho realizado por:

João Figueiredo Martins Peixe dos Santos

Luís Filipe Cruz Sobral

Paulo Silva Sousa

A89520
João Santos

A89474
Luís Sobral

A89465
Paulo Sousa

Projeto POO
2019/2020
Universidade do Minho

Índice

1 Introdução

Nesta unidade curricular foi-nos proposta a implementação de uma aplicação de entrega de encomendas aos utilizadores, por parte de Transportadoras e Voluntários.

Inicialmente, o foco prioritário deste projeto foi o encapsulamento das estruturas de dados por nós utilizadas, respeitando a metodologia da prograamação orientada aos objetos.

Ao longo do desenvolvimento deste projeto, consideramos que o maior desafio foi a implementação de certas funcionalidades na aplicação impostas pelo enunciado, mantendo, ao mesmo tempo, todas as outras operacionais.

2 Arquitetura de Classes

De modo a desenvolver a aplicação é necessário organizar dados numa estrutura que, apesar de compacta, seja de rápido acesso.

As classes elementares da nossa estrutura representam utilizadores, lojas, transportadoras, voluntários e encomendas.

Com o avançar do projeto, verificamos que seria necessária uma classe que agregasse estas classes para que estas pudessem interagir entre si e, ainda, precisaríamos de uma classe capaz de interagir com o utilizador. Assim, decidimos proceder à implementação do modelo MVC não só para obter independência das várias camadas (Modelo, Vista e Controlador), mas também para poder interagir com o utilizador de forma segura.

Assim, todos os dados destas classes são agregados numa Base de Dados geral à qual chamamos de GestTrazAqui, que controla o Modelo do programa. Além disso, temos a classe Interpretador responsável pelo controlo do programa e a classe apresentação responsável pela vista.

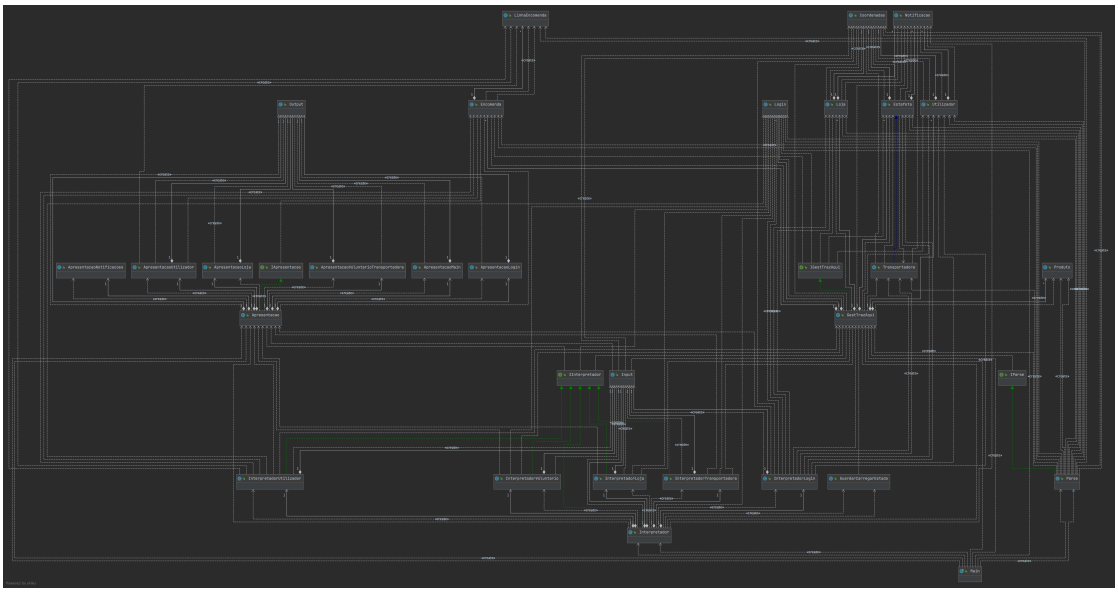


Diagrama de Classes

2.1 TrazAquiApp

Esta classe é a main do programa. É a classe que contêm os vários módulos do MVC e que executa o interpretador.

Estruturas

- *GestTrazAqui* c - Modelo da aplicação
- *Interpretador* i - Controlador da aplicação
- *Apresentacao* a - Vista da aplicação
- *Parse* parse - Classe responsável pelo parse dos logs
- *Login* l - Classe responsável pela informação do Login

2.2 GestTrazAqui

Esta é a classe responsável pelo modelo do programa. Aqui encontram-se todas as estruturas do nosso programa. Esta classe implementa a interface IGestTrazAqui.

Atributos

- *Map<String, Utilizador>* users - Lista dos utilizadores registados na aplicação
- *Map<String, Loja>* lojas - Lista das lojas registadas na aplicação
- *Map<String, Estafeta>* estafetas - Lista de voluntários e transportadoras registados na aplicação
- *Map<String, Encomenda>* encomendas - Lista de encomendas
- *Map<String, Produto>* produtos - Lista de produtos disponíveis
- *Map<String, Login>* loginMap - Lista de logins da aplicação
- *int[]* randomTraffic - Array com valores aleatórios de trânsito
- *int[]* randomWeather - Array com valores aleatórios de meteorologia
- *int[]* randomQueue - Array com valores aleatórios de tamanho de filas de espera

Decidimos utilizar Maps em vez de Sets nas nossas estruturas devido à enorme vantagem de, durante a execução dos métodos, apenas guardarmos os códigos de cada elemento (encomenda, utilizador, etc.) e utilizarmos isso como key para aceder ao Map. Além disso, optamos pela utilização de HashMaps em vez de TreeMaps porque a ordenação dos TreeMaps não nos traz vantagens neste caso, sendo então os tempos dos HashMaps mais vantajosos.

Temos também três arrays de inteiros que correspondem à intensidade do trânsito, meteorologia e tamanho das filas. Mais tarde, na função calculaTempo, acedemos ao array numa posição aleatória para obtermos as condições.

2.2.1 Utilizador

Esta é a classe com a informação de um utilizador.

Atributos

- *String* codigoUtilizador - Código de utilizador
- *String* nome - Nome do utilizador
- *Coordenadas* gps - Localização do utilizador
- *double* precoMax - Preço máximo aceite pelo transporte de uma encomenda
- *Set<String>* encomendas - Set de códigos de encomenda do utilizador
- *Set<String>* standBy - Set de códigos de encomenda em standBy (à espera de aceitação de um voluntário, por exemplo)
- *List<String>* notificacoes - Lista de notificações a apresentar

Para organizar as encomendas utilizamos dois Set, no primeiro guardamos todos os códigos de encomenda em estado normal e, no segundo, guardamos todas as encomendas em StandBy (i.e. à espera de aprovação de um voluntário ou à espera de aprovação de uma rota).

Para as notificações utilizamos uma List(ArrayList), porque a ordem pela qual as notificaçoes são inseridas é a mesma pela qual são impressas.

2.2.2 Loja

Esta é a classe com a informação de uma loja.

Atributos

- *String* storeCode - Código da loja
- *String* storeName - Nome da loja
- *Coordenadas* gps - Localização da loja
- *boolean* hasQueueInfo - Booleano que indica se a loja envia informação da fila
- *double* queueTime - Tempo na fila de espera
- *double* queueSize - Número de pessoas na fila de espera
- *List<String>* prods - Lista de produtos disponíveis na loja
- *Set<String>* encomendas - Set de códigos de encomenda efetuadas na loja
- *List<String>* notificacoes - Lista de notificações a apresentar

Tal como na classe acima, utilizamos um Set para as encomendas e uma List(ArrayList) para as notificações.

2.2.3 Estafeta

Esta é a classe com a informação de uma transportadora ou voluntário.

Atributos

- *String* code - Código do estafeta
- *String* name - Nome do estafeta
- *String* type - Tipo do estafeta (Transportadora ou Voluntários)
- *Coordenadas* gps - Localização do estafeta
- *double* raio - Raio de ação do estafeta
- *double* velocidade - Velocidade do estafeta
- *double* numKm - Número de quilómetros viajado pelo estafeta
- *boolean* isFree - Indica se o estafeta está disponível
- *boolean* isMedic - Indica se o estafeta transporta encomendas médicas
- *boolean* occup - Indica se o estafeta está ocupado
- *double* classificacao - Classificação do estafeta
- *int* numCla - Número de classificações
- *Set<String>* encomendas - Set de códigos de encomenda transportados
- *List<String>* notificacoes - Lista de notificações a apresentar

De novo, tal como na classe Utilizador, utilizamos um Set para as encomendas e uma List(ArrayList) para as notificações.

2.2.4 Transportadora

Esta é a classe com a informação de uma transportadora. Esta extende a classe Estafeta.

Atributos

- *int* nif - NIF da transportadora
- *double* taxaKm - Taxa aplicada por cada Km viajado
- *double* taxaPeso - Taxa aplicada por cada Kg da encomenda
- *int* numEncomendas - Máximo de encomendas que pode transportar
- *Set<String>* rota - Set com a rota definida

2.2.5 Produto

Esta é a classe com a informação de um produto.

Atributos

- *String* prodCode - Código do produto
- *String* name - Nome do produto
- *double* weight - Peso do produto
- *double* price - Preço do produto
- *boolean* isMedic - Indica se o produto é médico

2.2.6 Encomenda

Esta é a classe com a informação de uma encomenda.

Atributos

- *String* encCode - Código da encomenda
- *String* userCode - Código do utilizador
- *String* transpCode - Código do estafeta
- *String* storeCode - Código da loja
- *double* weight - Peso da encomenda
- *boolean* isMedic - Indica se a encomenda é médica
- *LocalDateTime* data - Data da encomenda
- *boolean* aceiteLoja - Indica se a loja aceitou a encomenda
- *boolean* entregue - Indica se a encomenda foi entregue
- *double* preco - Preço da encomenda
- *double* tempoEntrega - Tempo de entrega
- *boolean* standBy - Indica se a encomenda está em standBy
- *List<LinhaEncomenda>* linha - Lista com as linhas da encomenda

2.2.7 LinhaEncomenda

Esta é a classe com a informação de uma linha de encomenda.

Atributos

- *String* productCode - Código do produto
- *String* description - Descrição do produto
- *double* quantity - Quantidade
- *double* unitPrice - Preço unitário

2.2.8 Login

Esta é a classe com a informação de um login.

Atributos

- *String* code - Código de Login
- *String* password - Password de Login
- *double* tipoConta - Tipo de Conta
- *double* nome - Nome do Cliente

2.2.9 Coordenadas

Esta é a classe com a informação da localização utilizada pelos diferentes clientes da aplicação.

Atributos

- *double* latitude - Latitude da coordenada
- *double* longitude - Longitude da coordenada

2.2.10 Notificação

Esta é a classe com a informação de uma notificação.

Atributos

- *String* not - Notificação
- *int* type - Tipo de Notificação
- *String* estCode - Código do estafeta para classificar

2.3 Files

Nesta secção incluimos as duas classes responsáveis por ficheiros.

A primeira é a classe Parse. Nesta classe é feito o parse do ficheiro *logs.txt* onde se encontram disponíveis os dados iniciais do programa, disponibilizado pelo professor. Além disso também é feito parse do ficheiro *produtos.txt* criado pelo grupo, que contém todos os produtos disponíveis para serem utilizados pelas lojas. Esta classe implementa a interface IParse.

A outra é a classe GuardarCarregarEstado. Esta classe guarda e lê o estado da aplicação a partir de um ficheiro *GestTrazAqui.dat*.

2.4 Interpretador

Esta é a classe responsável pelo controlador do programa, ou seja, é a classe que interage com o utilizador. Esta classe implementa a interface IInterpretador.

Atributos

- *Input* in - Classe que recebe input
- *InterpretadorLogin* intL - Classe que controla o menu de login
- *InterpretadorUtilizador* intU - Classe que controla o menu de utilizador
- *InterpretadorVoluntario* intE - Classe que controla o menu de voluntario
- *InterpretadorTransportadora* intT - Classe que controla o menu de transportadora
- *InterpretadorLoja* intLj - Classe que controla o menu de loja

2.5 Apresentação

Esta é a classe responsável pela vista do programa, ou seja, é a classe que devolve resultados visuais ao utilizador. Esta classe implementa a interface IApresentacao.

Atributos

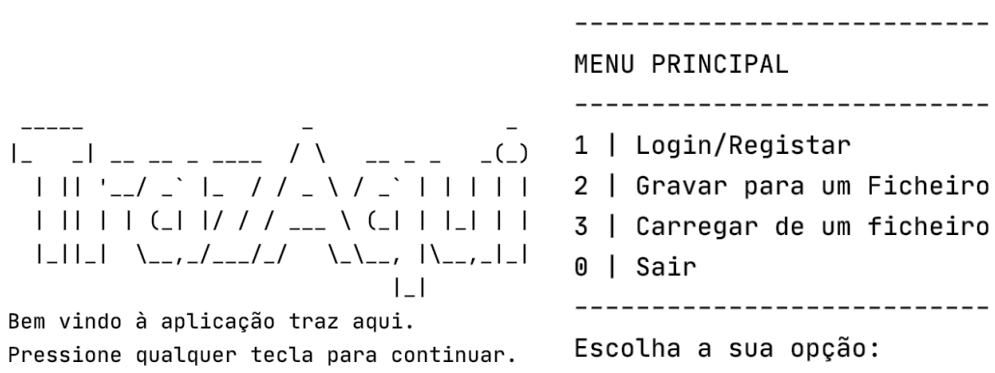
- *ApresentacaoMain* am - Classe responsável pela apresentação do menu principal
- *ApresentacaoLogin* al - Classe responsável pela apresentação do menu de login
- *ApresentacaoUtilizador* au - Classe responsável pela apresentação do menu de utilizador
- *ApresentacaoVoluntarioTransportadora* avt - Classe responsável pela apresentação do menu de voluntário e de transportadora
- *ApresentacaoNotificacao* an - Classe responsável pela apresentação das notificações
- *ApresentacaoLoja* alj - Classe responsável pela apresentação do menu de loja
- *Output* out - Classe que devolve output de modo geral

3 Descrição da Aplicação

O modo de interação com o utilizador escolhido foi uma interface de linha de comandos.

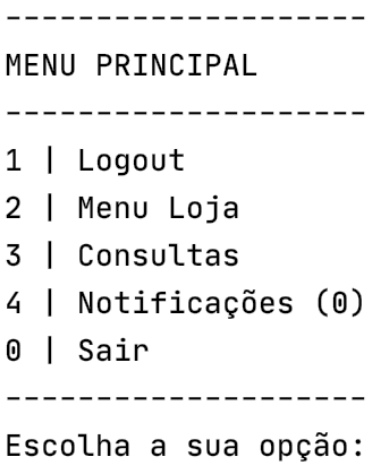
3.1 Menu Principal

Quando o cliente entra no programa depara-se com o menu de boas vindas. Daí, este pode premir qualquer tecla para exibir o menu principal.



No menu principal o cliente tem as seguintes opções:

- 1 - Fazer Login ou Registar um novo cliente
- 2 - Gravar o Estado atual do programa para um ficheiro .dat
- 3 - Carregar o Estado do programa a partir de um ficheiro .dat



Após o Login ser efetuado, surgem novas opções possíveis:

- 1 - Efetuar Logout da conta
- 2 - Menu de funcionalidades para cada tipo de cliente
- 3 - Menu que permite consultar dados estatísticos da aplicação
- 4 - Menu com notificações de encomendas para o cliente

3.2 Menu Login

```
--------------------
MENU LOGIN
--------------------
1 | Login
2 | Registar
0 | Voltar atrás
--------------------
Escolha a sua opção:
```

No menu de Login o cliente tem duas opções:

- 1 - Esta opção permite ao clienre fazer login na sua conta para utilizar a sua aplicação
- 2 - Permite registar uma nova conta na aplicação

Na nossa aplicação assumimos que todos os clientes carregados dos logs têm, por definição, o seu código de utilizador/transportadora/voluntário/loja como password.

Para as lojas, quando é registada uma nova, decidimos gerar uma lista aleatória de produtos para que esta possa ser utilizada pelos utilizadores.

3.3 Menu Utilizador

```
--------------------------------------
MENU UTILIZADOR
--------------------------------------
1 | Solicitar entrega de uma encomenda
2 | Aceder às encomendas
3 | Fazer encomenda
0 | Voltar atrás
--------------------------------------
Escolha a sua opção:
```

No caso do cliente decidir abrir o menu de utilizador depara-se com as seguintes opções:

- 1 - Esta opção permite ao utilizador solicitar a entrega de uma encomenda que esteja disponível, sendo-lhe atribuída um voluntário ou uma transportadora
- 2 - Permite aceder à lista de encomendas entregues, com limites de data mínima e máxima, e filtradas por voluntário, transportadora ou ambos
- 3 - Fazer uma nova encomenda numa loja à escolha

Ao solicitar a entrega de uma encomenda, se esta foi adicionada a uma rota por uma transportadora, irá ser questionado ao utilizador se este deseja que a encomenda seja entregue por essa transportadora. Caso o utilizador recuse, ou caso a encomenda não tenha sido adicionada a nenhuma rota, o sistema irá selecionar, de acordo com vários critérios, um voluntário ou uma transportadora para entregar a encomenda. Se for selecionada uma transportadora, o utilizador tem a opção de a aceitar ou não. Se for

selecionado um voluntário, a encomenda irá ficar em standby até este aceitar a entrega no seu menu.

Ao finalizar uma nova encomenda, esta terá de ser aceita pela loja, passando assim para a lista de encomendas a ser solicitadas.

Por último, quando a entrega é realizada é tambem calculado o tempo/preço da encomenda. Para calcular o tempo de entrega foram usados os seguintes fatores de aleatoriedade: condições climatéricas, trânsito e tamanho da fila de espera(caso a loja não forneça informação).

3.4 Menu Transportadora

```
---------------------------------------------------------------------
MENU TRANSPORTADORA
---------------------------------------------------------------------
1 | Sinalizar como disponivel/indisponivel para entregar encomendas
2 | Aceitar/Recusar Encomendas Médicas
3 | Aceder às encomendas
4 | Total faturado
5 | Classificação
6 | Definir rota de entregas
0 | Voltar atrás
---------------------------------------------------------------------
Escolha a sua opção:
```

No caso do cliente decidir abrir o menu de transportadora depara se com as seguintes opções:

- 1 - Sinaliza a transportadora com disponível ou indisponível para entregar uma encomenda
- 2 - Indica se aceita ou não encomendas médicas
- 3 - Permite aceder à lista de encomendas entregues pela transportadora, com limites de data mínima e máxima
- 4 - Calcula o total faturado pela transportadora na aplicação
- 5 - Apresenta a classificação da transportadora
- 6 - Permite à transportadora definir uma rota de distribuição de forma a transportar várias encomendas

Ao definir uma rota de entregas numa transportadora, se já houver encomendas na rota da transportadora o sistema verifica se todos os utilizadores já responderam ao pedido de entrega da transportadora. Caso isto se verifique a rota é efetuada (a rota da transportadora é esvaziada e todas as encomendas são entregues), caso contrário é possivel criar uma nova rota de entregas ou adicionar encomendas a uma rota já existente, tendo em conta o número máximo de encomendas que a transportadora pode transportar.

3.5 Menu Voluntário

```
-------------------------------------------------------------------
MENU VOLUNTÁRIO
-------------------------------------------------------------------
1 | Sinalizar como disponivel/indisponivel para entregar encomendas
2 | Aceitar Encomenda
3 | Aceitar/Recusar Encomendas Médicas
4 | Aceder às encomendas
5 | Classificação
0 | Voltar atrás
-------------------------------------------------------------------
Escolha a sua opção:
```

No caso do cliente decidir abrir o menu de voluntário depara se com as seguintes opções:

- 1 - Sinaliza o voluntário como disponível ou indisponível para entregar uma encomenda
- 2 - Permite ao voluntário aceitar ou não a entrega de uma encomenda a um utilizador
- 3 - Indica se aceita ou não encomendas médicas
- 4 - Permite aceder à lista de encomendas entregues pelo voluntário, com limites de data mínima e máxima
- 5 - Apresenta a classificação do voluntário

Quando uma encomenda é atribuída a um voluntário, o booleano occup (característico da classe Estafeta) passa a true, ficando, assim, o voluntário indisponível para qualquer outra encomenda. Quando executa a opção 2, o voluntário pode aceitar ou recusar entregar a encomenda, voltando assim o booleano occup a false, e o voluntário fica novamente disponível.

3.6 Menu Loja

```
---------------------------------------
MENU LOJA
---------------------------------------
1 | Atualizar Tempo de Fila de Espera
2 | Atualizar tamanho da Fila de Espera
3 | Aceitar compra
4 | Histórico de encomendas
0 | Voltar atrás
---------------------------------------
Escolha a sua opção:
```

No caso do cliente decidir abrir o menu de loja depara se com as seguintes opções:

- 1 - Atualiza o tempo de fila de espera na loja
- 2 - Atualiza o tamanho da fila de espera na loja
- 3 - Permite à loja aceitar ou não uma compra de um utilizador

- 4 - Permite aceder à lista de encomendas feitas na loja, com limites de data mínima e máxima

Neste menu, caso a loja ao ser registada não tenha informação sobre a fila de espera, a primeira e segunda opções do menu são omitidas.

Quando uma encomenda é efetuada na loja, esta tem a opção de aceitar ou recusar a encomenda. Caso a loja recuse a encomenda, esta é removida de todas as estruturas, caso aceite a encomenda, esta passa para a fase de ser solicitada pelo Utilizador.

3.7 Menu Consultas

```
---------------------------------
MENU CONSULTAS
---------------------------------
1 | Top Utilizadores do Sistema
2 | Top Transportadoras do Sistema
0 | Voltar atrás
---------------------------------
Escolha a sua opção:
```

No caso do cliente decidir abrir o menu de consultas depara se com as seguintes opções:

- 1 - Lista os 10 utilizadores que realizaram mais encomendas
- 2 - Lista as 10 transportadoras que realizaram mais quilómetros

3.8 Notificações

```
-------------------------------------------------------
                  Notificação (1/1)
-------------------------------------------------------
Tem uma compra pendente (e8380) do utilizador u78.
-------------------------------------------------------
         [1] Próxima | [2] Anterior | [0] Sair
-------------------------------------------------------
Escolha a sua opção:


-------------------------------------------------------
                  Notificação (1/1)
-------------------------------------------------------
Entrega da encomenda e6813 realizada com sucesso pela t
ransportadora t51
-------------------------------------------------------
[1] Próxima | [2] Anterior | [3] Classificar | [0] Sair
-------------------------------------------------------
Escolha a sua opção:
```

Por último, decidimos introduzir um sistema de notificações para que qualquer utilizador do programa tenha feedback do estado da sua encomenda e para que as lojas e os estafetas recebam informação sobre novas compras/encomendas.

Caso a encomenda seja referente a uma entrega por parte de uma transportadora ou um voluntário, o utilizador tem a possiblidade de classificar a entrega, através da opção 3.

4 Conclusão e Reflexão Crítica

De um modo geral, estamos satisfeitos com a nossa solução, tanto a nível de desempenho do programa, como a nível de interação com o utilizador.

Como tentamos uma abordagem de modo a implementar uma aplicação "real", a realização da entrega de uma encomenda revelou-se bastante inconveniente, pois todas as contas são geridas no mesmo terminal. No entanto, achamos que se fosse possível executar a aplicação em real time teria uma utilização bastante simples e prática.

Para concluir, os objetivos deste projeto foram atingidos e durante a sua realização foram de reter determinados pontos. Tais como, a necessidade de um código seguro, protegido do utilizador, como também a importância da organização das estruturas de dados, quais os algoritmos a implementar e como diferentes relações entre estes dois conceitos tiveram um efeito considerável nas funcionalidades do programa.