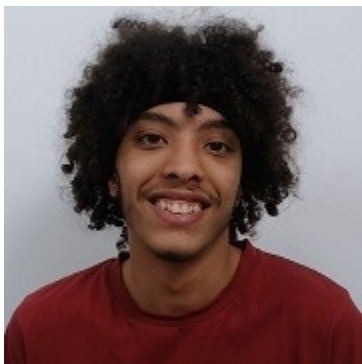




TRAZ AQUI!

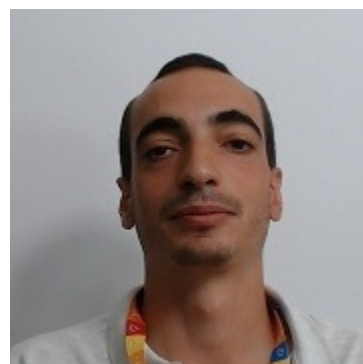
Programação Orientada a Objetos 2019/2020
(Grupo 7)



Alexandre Ferreira Gomes
A89549



Marco Avelino Teixeira
Pereira A89556



Manuel Jorge Mimoso
Carvalho A69856

INTRODUÇÃO

Com este trabalho pretendíamos desenvolver uma aplicação que permitisse aos seus Utilizadores fazer encomendas a Lojas e que estas fossem entregues a nós através de entregadores que podiam ser Voluntários ou Transportadoras especializadas para o serviço.

Sendo o foco deste trabalho a implementação dos vários mecanismos aprendidos na Unidade Curricular como o uso de herança, interfaces e o modelo MVC para permitir uma mais fácil gestão do programa, tivemos uma agradável surpresa ao saber o quão mais simples é trabalhar na linguagem Java sendo uma das maiores preocupações o encapsulamento dos objetos de modo a não expor a informação contida nestes a possíveis ataques que a alterassem e podessem facilmente modificar o balanço na conta de um utilizador ou o preço de produtos, entre outros.

ESTRUTURAÇÃO

Model

Para o desenvolvimento deste trabalho prático decidimos criar 3 classes base sendo elas Loja, Utilizador, Entregador, utilizando esta última abstract com o objetivo de evitar a repetição de código servindo para guardar as várias variáveis comuns entre Voluntário e Entregador, tendo estas 3 extends uma classe abstrata que seria herdada por todas as classes base de seu nome BasicInfo, esta classe irá conter as variáveis de instância comuns às 3 classes acima referidas e os respectivos métodos, estas variáveis são o código, o nome e a password em forma de String e ainda um Point2D que representam a posição de cada entidade numa grelha 2D e ainda uma lista de Strings que representa as mensagens que serão apresentadas a cada entidade quando está faz sign in/up.

Além desta simples estrutura temos ainda várias classes de serviço como é o exemplo da classe LinhaDeEncomenda e Encomenda, assim como TriploPedido, etc.

Cada uma das classes demonstradas tem a respetiva interface implementada que permite uma melhor descrição e uma maior generalização do programa por nós criado além de fornecer uma simples forma de saber quais as funções que a classe que a implementa pode usar (API).

Temos assim em baixo apresentado um diagrama com apenas as variáveis de instância de cada uma destas e o seu relacionamento.

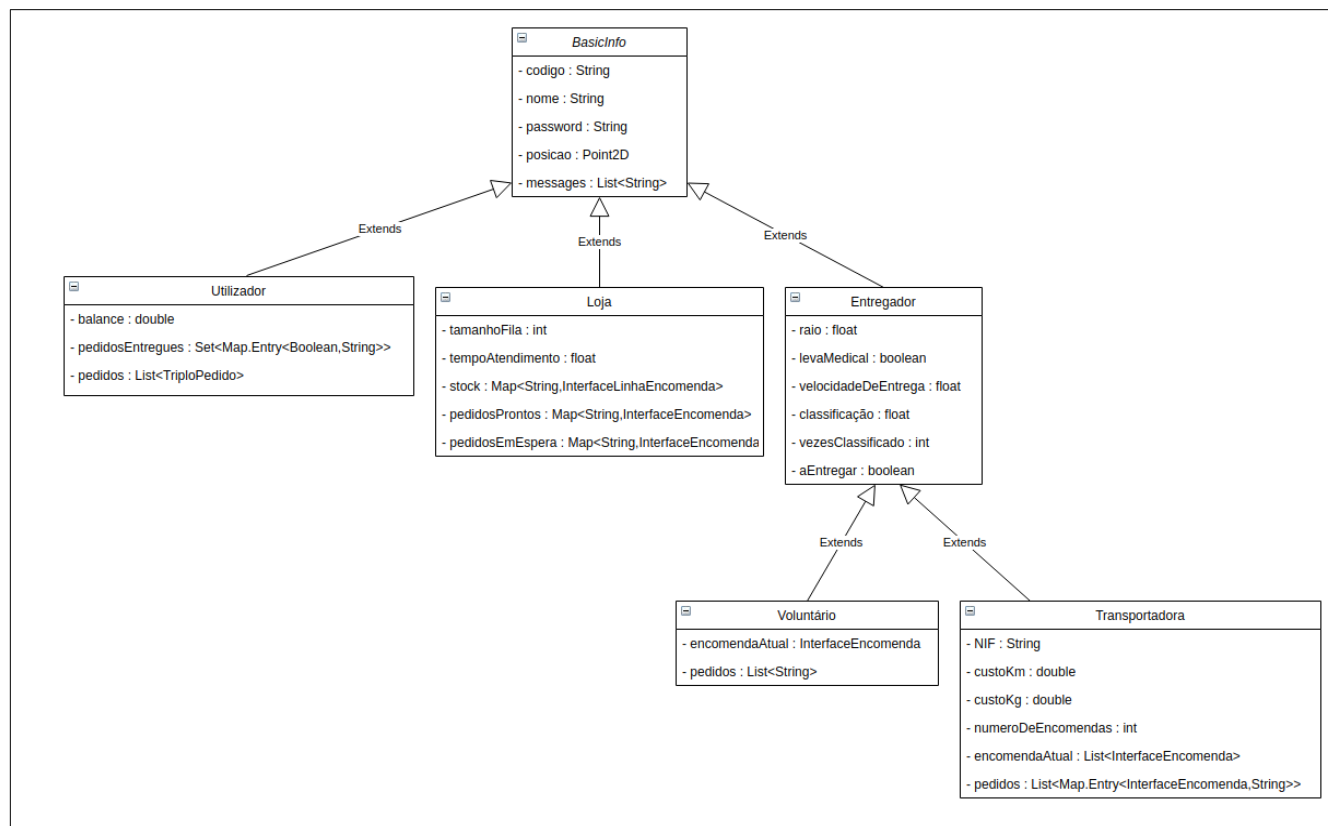


Figure 1: Diagrama de dependência de classes base

Depois da definição destas classes base tivemos a preocupação com a definição de classes onde estivessem agrupadas as várias entidades dentro de um respetivo map, para isso decidimos criar as classes Utilizadores, Lojas e Entregadores, estas tinham apenas dentro delas um Dicionário que mapeava junto os códigos de acesso a cada entidade para a respetiva classe que continha a informação deste.

Assim ao aceder a utilizadores tínhamos lá dentro toda a informação de utilizadores agrupada num Map.

Sendo estas 3 classes bastante “straightforward” de perceber não iremos colocar um diagrama com as classes pois elas não se interligam,.

Ora, por questões de conveniência assim como organização do trabalho decidimos criar uma classe que irá conter cada uma destas 3 anteriormente faladas assim como a hora em que o estado atual se encontrar, essa classe age como um banco de dados por isso o seu nome é Data.

View

Neste ponto atual já temos a parte mais complicada definida, que é o Model, por isso todas as classes dentro deste estão organizadas em diferentes packages para permitir uma mais fácil compreensão e organização de código.

A View é responsável por mostrar ao user todo o output sendo assim esta não tem variáveis de instância e contem as várias funções de pedido de input, decidimos assim chamar-lhe Printer.

Controller

O controller é aquilo que define o que o Model irá realizar estando assim definido com 2 variáveis de instância o Printer(View) e Data(Model).

Este é responsável por organizar o envio de informação ao Model sobre o método a realizar e irá enviar à View a informação que o método devolver.

Sendo assim o Controller é a classe responsável por todo o input.

Relação MVC

A relação MVC entre Model, View e Controller foi a primeira abordagem que tivemos deste, apresentada na UC de LI3, em que o Model é semelhante ao apresentado em POO servindo este apenas para executar operações pedidas pelo Controller, não sabendo nada sobre a View e vice-versa. Por outro lado a definição da relação entre View é, como em LI3, algo muito estático servindo apenas para operações de output e não podendo fazer input, para este temos o Controller, o Controller pede à View para apresentar menus e resultados de operações no Model que o justifiquem e para guardar o input do utilizador chamando depois o devido método ao Model com a informação lida do utilizador.

Diagramas

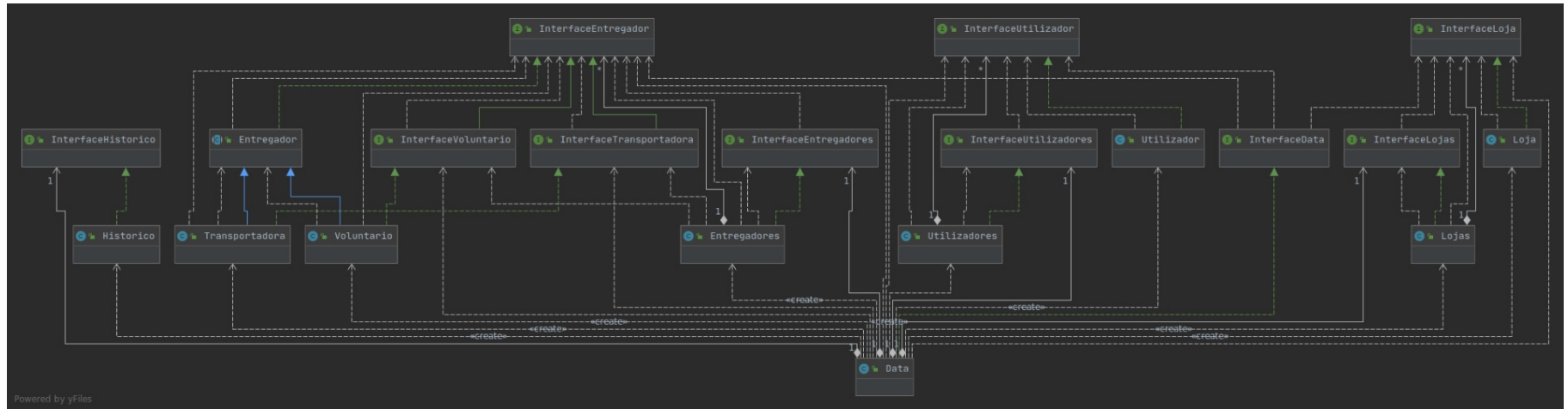


Figure 2: Diagrama de Classes

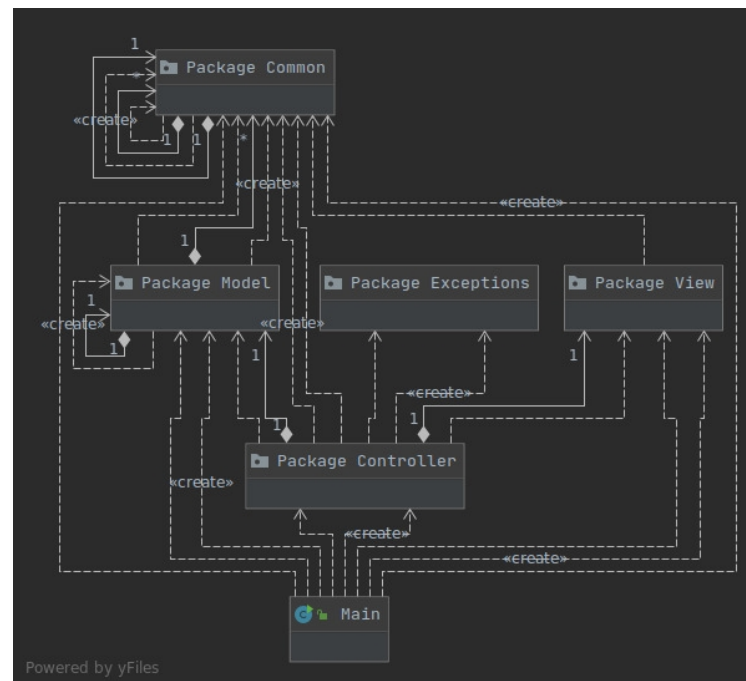
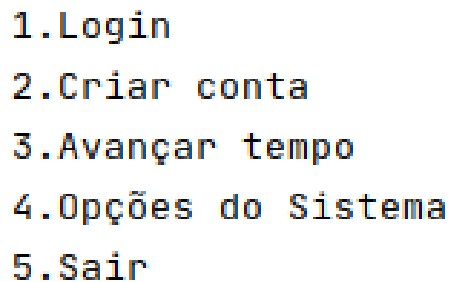


Figure 3: Diagrama de Packages

Funcionalidades

Menu Inicial



```
1.Login
2.Criar conta
3.Avançar tempo
4.Opções do Sistema
5.Sair
```

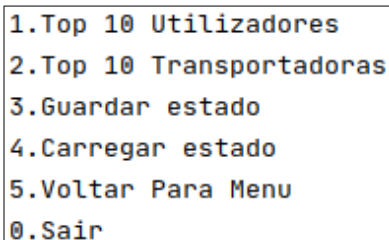
Figure 4: Menu Inicial

A primeira interação que o utilizador tem com o programa é menu inicial. Este possui as opções acima apresentadas na figura 4: login, criar conta, avançar tempo, opções do sistema e sair.

O Login é usado para entrar numa conta que já tenha sido previamente criada, caso contrario, após falha nas credenciais, é apresentado um novo prompt, dando a oportunidade de fazer parte da aplicação através da criação de uma nova conta.

A criação de conta é usada para novas entidades que desejam usar a aplicação. Esta mesma, varia consoante o uso pretendido, sendo assim possível serem distinguidas entre contas de utilizador, voluntário, transportadora ou loja.

Avançar no tempo, foi a maneira que foi implementada para simular a passagem do tempo, sendo assim pedido o tempo desejado a avançar.



```
1.Top 10 Utilizadores
2.Top 10 Transportadoras
3.Guardar estado
4.Carregar estado
5.Voltar Para Menu
0.Sair
```

Figure 5: Menu de Sistema

Opções do Sistema, são opções/funcionalidades disponiveis e comuns a todas a entidades, como a apresentação de informações estatísticas relativamente ao top 10 de utilizadores ou transportadoras, a gravação do estado atual da aplicação ou o seu carregamento a partir do ficheiro de dados.

Sair termina a execução do programa.

Menu de conta

Conta Utilizador:

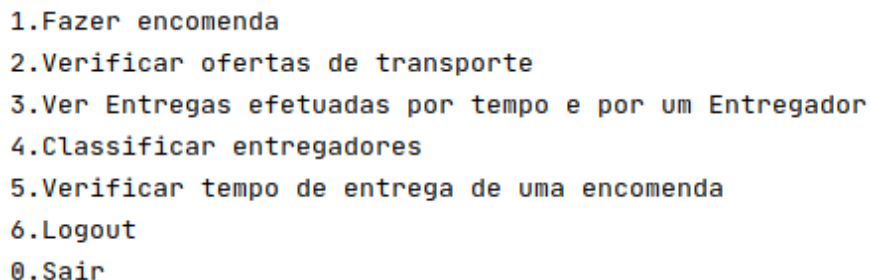
- 
- 1.Fazer encomenda
 - 2.Verificar ofertas de transporte
 - 3.Ver Entregas efetuadas por tempo e por um Entregador
 - 4.Classificar entregadores
 - 5.Verificar tempo de entrega de uma encomenda
 - 6.Logout
 - 0.Sair

Figure 6: Menu Conta Utilizador

Fazer encomenda é a funcionalidade que permite o utilizador construir a sua lista de compras para uma loja. A lista de produtos disponíveis é apresentada em forma de tabela com paginação, podendo ser interagida a partir de comandos específicos, permitindo navegar para uma página à escolha, avançar ou retroceder uma página, adicionar ou remover produto e finalizar encomenda.

Ofertas de transporte é a secção onde é apresentada todas as ofertas e orçamentos de entrega para várias transportadoras, a partir das quais o utilizador pode aceitar ou rejeitar. No caso de alguma oferta ser aceita, essa transportadora recebe uma notificação de sucesso, enquanto que todas as outras recebem uma notificação de oferta rejeitada.

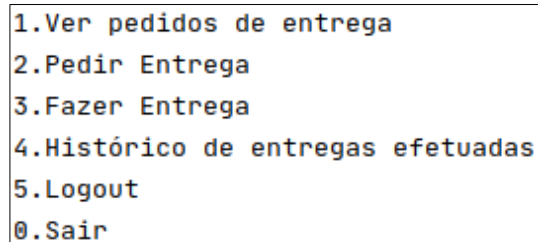
A funcionalidade 3 da figura 6 é um histórico de entregas, em que o utilizador pode usar filtros se o desejar, como tempo e/ou entregador.

Classificação dos entregadores, é a qualidade de prestação do serviço, quer de transportadores quer de voluntários. Classificações só podem ser atribuídas a entidades que prestraram serviço ao utilizador.

A funcionalidade 5 permite ao utilizador verificar o tempo restante de entrega da encomenda. Caso seja feito um avanço no tempo, este tempo é por sua vez também atualizado.

Logout permite regressar ao menu inicial e sair para terminar aplicação.

Conta Voluntário:

A screenshot of a menu for a volunteer account. The menu is displayed in a white box with a black border. It contains six items, each preceded by a number. The first item is '1.Ver pedidos de entrega', the second is '2.Pedir Entrega', the third is '3.Fazer Entrega', the fourth is '4.Histórico de entregas efetuadas', the fifth is '5.Logout', and the sixth is '0.Sair'.

```
1.Ver pedidos de entrega
2.Pedir Entrega
3.Fazer Entrega
4.Histórico de entregas efetuadas
5.Logout
0.Sair
```

Figure 7: Menu Conta Voluntário

Os voluntários, selecionando a primeira opção apenas conseguem visualizar encomendas que ainda não foram aceites, as que se encontrem no seu raio de acção e vão de acordo com a sua licença de transporte de produtos médicos. A partir desta listagem, o voluntario pode efetuar o pedido e posteriormente a entrega (2 e 3).

Por fins estatísticos é possível visualizar o todo o de entregas efetuadas ou num determinado periodo.

Tal como mencionado anterior no menu do utilizador, logout sai da conta e regressa ao menu inicial, enquanto que sair termina a execução da aplicação.

Conta Transportadora:

- 1.Ver pedidos de entrega
- 2.Calcular preco de transporte
- 3.Propor Entrega
- 4.Verificar Pedidos Propostos
- 5.Fazer Entrega
- 6.Histórico de entregas efetuadas
- 7.Total Faturado
- 8.Logout
- 0.Sair

Figure 8: Menu Conta Transportadora

Tal como os voluntarios, os pedidos de entrega apresentados por (1) são apenas aqueles aos quais a transportadora possui os requisitos necessários para o serviço, como o raio de acção e a licença de transporte de produtos médicos.

A transportadora pode fazer uma proposta de entrega ao utilizador (3), ao qual este poderá ou não aceitar. Caso seja aceite, a transportadora poderá então prodecir à sua entrega (5) imediatamente ou esperar por novas encomendas. Estas são feitas por ordem sequencial de pedido, seguindo uma rota desde a sua localização até ao ponto ao de recolha do item, e deste até ao utilizador que realizou a encomenda.

Por fins informativos e estatísticos é dada a opção de calculo de faturação da possível entrega (2), faturação total (7) e historico total ou por data de entregas (6).

Logout (8) e sair (0) é igual aos outros menus já referidos anteriormente.

Conta Loja:

```
1.Atualizar Stock  
2.Logout  
3.Sair
```

Figure 9: Menu Conta Loja

As funcionalidades da loja baseiam-se no controle de stock e passam por, poder adicionar novos produtos, mudar as quantidades e os preços, remover o produto, passar para a página seguinte ou recuar a página, ir para uma página específica e terminar todas as alterações. Tal como pode ser visto nas imagens abaixo.

```
[a] Adicionar Produto | [s] Mudar Quantidade | [c] Mudar Preço | [r] Remover Produto
```

Figure 10: Gestão de Stock (Parte 1)

```
[n] Proxima Pagina | [p] Pagina Anterior | [(numero)] Ir para pagina | [q] Terminar Alterações
```

Figure 11: Gestão de Stock (Parte 2)

Conclusão

Neste trabalho conseguimos, na nossa opinião, cumprir todos os requisitos sendo o mais importante o encapsulamento de dados, que é como sabemos uma das mais importantes partes de programar em linguagem orientada a objetos. Além disso facilmente conseguimos expandir código sem ter de reescrever muito do que está feito graças à implementação que decidimos utilizar.

O que poderia ser melhorado era a implementação de MVC para estar de acordo com o ensinado na UC de POO, além disso poderíamos também melhorar a rota das transportadoras pois embora exista uma ela não é otimizada.

Este trabalho fez-nos perceber o quão mais simples a linguagem Java é comparada a outras mais low level, dizemo isto pois Java tem sempre uma implementação para tudo e não temos de nos preocupar com problemas de memória pois Java trata da memória por nós.