Universidade do Minho
Escola de Engenharia

Programação Orientada aos Objetos

Grupo 36

Daniel José Silva Furtado A97327
Lara Beatriz Pinto Ferreira A95454
Nuno Miguel Leite da Costa A96897

A97327

A95454

A96897

2 de maio de 2023

Índice

1 Introdução

O seguinte relatório é referente ao projeto prático, desenvolvido em Java, da Unidade Curricular de Programação Orientada a Objetos.

O objetivo deste projeto consiste em construir um sistema de *marketplace* **Vintage** que permite a compra e venda de artigos novos e usados de vários tipos.

No inicio, o principal foco foi o encapsulamento das estruturas de dados que utilizamos, respeitando a metodologia da Programação Orientada aos Objetos

Ao longo do desenvolvimento do projeto, enfrentamos alguns desafios ao implementar as funcionalidades específicas exigidas no enunciado, pelo que serão enunciadas as principais decisões tomadas pelo grupo na conceção do trabalho analisando as suas implicações e potenciais benefícios e/ou desvantagens.

2 Arquitetura de Classes

Com o objetivo de que seja possível desenvolver a aplicação é fundamental organizar dados numa estrutura que seja tanto compacta como de rápido acesso.

As classes elementares da nossa estrutura representam artigos, transportadoras, utilizadores e encomendas. Com o continuar do projeto, constatamos que seria indispensável 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.

No nosso trabalho, a composição está bastante presente na forma como as classes relacionam-se. Para conseguir manter o encapsulamento e para que não haja perda ou alteração dos objetos. Assim não há partilha de objetos entre classes e mantém o trabalho coeso e sem fugas de informação dos objetos que poderiam ser alterados. Para isso nos métodos que todas as classes necessitam, tal como os construtores, *getters* e *setters* todos os objetos são "clonados".

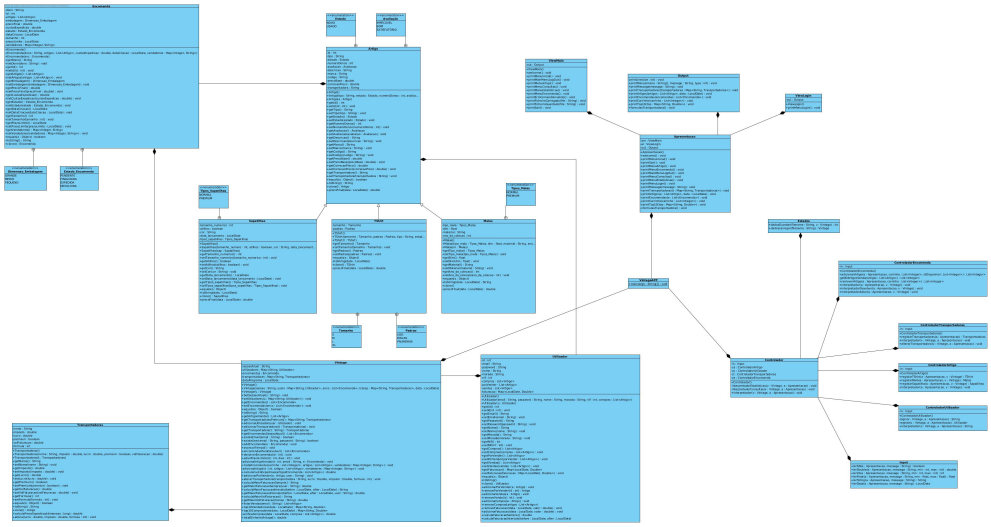


Figura 1: Diagrama de Classes

2.1 Model-View-Controller

Decidimos implementar a arquitetura Model-View-Controller (MVC) de forma a obter independência das várias camadas e para poder interagir com o utilizador de forma segura. A cada uma destas camadas foi atribuído um package único.

Desta forma, e com o intuito de controlar o modelo do programa, todos os dados das classes referidas em cima são agregados numa Base de Dados, que tem como nome VintageAPP.

2.2 VintageAPP

A seguinte classe representa a main do programa, permitindo assim a sua execução. Esta classe contém os vários módulos do MVC e executa o controlador.

- **Vintage** v - Modelo da Aplicação
- **Apresentação** a - Vista da Aplicação
- **Controlador** c - Controlador da Aplicação

3 Model

A identificação das entidades é feita através de um identificador inteiro (ID), que é único para cada entidade. De forma a garantir a unicidade dos identificadores, foi criada a variável *nextID* que corresponde ao identificador a atribuir da próxima vez que um construtor seja invocado. Quando é criada uma nova entidade, esta variável é incrementada.

3.1 Vintage

Esta classe é responsável pelo modelo do programa, é possível aqui encontrar todas as estruturas do nosso programa.

- **String** sessaoAtual - Representa o utilizador atual
- **Map<String, Utilizador>** utilizadores - Lista de todos os utilizadores existentes
- **List<Encomenda>** encomendas - Lista de todas as encomendas realizadas
- **Map<String, Transportadoras>** transportadoras - Lista de todas as transportadoras existentes
- **LocalDate** dataPrograma - Data da execução do programa

3.2 Artigo

Esta classe contem a informação de cada Artigo.

- **int** id - ID do artigo
- **String** tipo - Tipo de artigo
- **Estado** estado - Estado em que o artigo se encontra
- **int** numeroDonos - Numero de donos que o artigo teve previamente
- **Avaliação** avaliacao - Avaliação do artigo
- **String** descricao - Descrição do artigo

- **String** marca - Marca do artigo
- **String** codigo - Codigo do artigo
- **double** precoBase - Preço base do artigo
- **double** correcaoPreco - Desconto a aplicar ao preço base do artigo
- **String** transportadora - Transportadora do artigo

3.2.1 Malas

Esta classe extende da classe Artigo para descrever um artigo do tipo *Malas* com os seguintes atributos.

- **TiposMalas** tipoMala - Tipo da Mala, pode ser Normal ou Premium
- **float** dim - Dimensão da mala
- **String** material - Material da mala
- **int** anoDaColecao - Ano da coleção da mala

3.2.2 Sapatilhas

Esta classe extende da classe Artigo para descrever um artigo do tipo *Sapatilhas* com os seguintes atributos.

- **int** tamanhoNumerico - Tamanho das sapatilhas
- **boolean** atilhos - Verifica se a sapatilha tem atilhos
- **String** cor - Cor da sapatilha
- **LocalDate** dataLancamento - Data de lançamento da sapatilha
- **TiposSapatilhas** tiposSapatilhas - Tipo das Sapatilhas, pode ser Normal ou Premium

3.2.3 Tshirt

Esta classe extende da classe Artigo para descrever um artigo do tipo *Tshirt* com os seguintes atributos.

- **Tamanho** tamanho - Tamanho da tshirt, pode ser S, M, L, XL
- **Padrao** padrao - Padrão da tshirt, pode ser Liso, Riscas ou Palmeiras

3.3 Utilizador

Esta classe contem a informação de cada Utilizador.

- **int** id - ID do utilizador
- **String** email - Email do utilizador
- **String** password - Password do utilizador
- **String** nome - Nome do utilizador
- **String** morada - Morada do utilizador
- **int** nif - NIF do utilizador
- **List<Artigo>** compras - Lista de artigos comprados
- **List<Artigo>** porVender - Lista de artigos por vender
- **Map<LocalDate, Double>** faturacao - Valor que o utilizador faturou no respetivo dia, apenas serão guardados os dias em que existem vendas associadas ao utilizador

3.4 Encomenda

Esta classe contem a informação de cada Encomenda.

- **String** dono - Dono da encomenda
- **int** id - ID da encomenda
- **List<Artigo>** artigos - Lista dos artigos
- **DimensaoEmbalagem** embalagem - Dimensão da encomenda
- **double** precoFinal - Preço final da encomenda
- **double** custosExpedicao - Custos de encomenda
- **EstadoEncomenda** estado - Estado da encomenda
- **LocalDate** dataCriacao - Data de criação da encomenda
- **int** tamanho - Tamanho da encomenda
- **LocalDate** prazoLimite - Prazo limite da encomenda
- **Map<Integer,String>** vendedores - Lista de vendedores

3.4.1 Realizar uma encomenda

Para realizar uma encomenda é mostrado ao utilizador todos os artigos disponíveis para venda, ou seja, os artigos que outros utilizadores colocaram à venda. Para o utilizador adicionar algum artigo à encomenda criamos um carrinho temporário que armazena todos os *ID*'s dos artigos introduzidos pelo utilizador. Quando o cliente pretende terminar a encomenda todos esses *ID*'s do carrinho são associados aos respetivos artigos e esses artigos são removidos da lista de vendas dos outros vendedores. Também são adicionados à lista de vendas dos utilizadores respetivos, o preço do artigo também é adicionado à lista de faturação do vendedor no respetivo dia e é guardado o *email* do vendedor associado ao respetivo *ID* do artigo. Do lado do comprador os artigos também são adicionados à lista de compras. A encomenda é concluída com o estado Expedida.

3.4.2 Devolver uma encomenda

Para devolver uma encomenda foi necessário percorrer todos os *ID*'s dos artigos associados ao vendedor e repor novamente o artigo na lista de artigos a vender, também é removido o artigo da lista de vendas e no lado da pessoa que o comprou os artigos são removidos da lista de compras. O estado da encomenda fica como Devolvida e não é removida da lista de encomendas do programa.

3.4.3 Finalizar uma encomenda

Para tornar o estado da encomenda Finalizada, acontece quando a data atual do programa é posterior à data de prazo limite da encomenda. Após avançar a data do programa são percorridas todas as encomendas e as suas datas de prazo limite são verificadas. Caso a data já esteja ultrapassada e o estado da encomenda não seja Devolvida então o estado da encomenda passa a Finalizada.

3.5 Transportadoras

Esta classe contem a informação de cada Transportadora.

- **String** nome - Nome da transportadora
- **double** imposto - Imposto cobrado pela transportadora
- **double** lucro - Lucro ganho pela transportadora
- **boolean** premium - Verifica se é transportadora premium
- **double** volFaturacao - Volume de faturação da transportadora
- **int** formula - Numero da formula para o calculo do custo ????

4 Files

Na secção *Files*, temos dois métodos *dadosGuardar* e *dadoscarregar*, que são responsáveis pela leitura e escrita de objetos **Vintage** em arquivos.

O método *dadosGuardar* recebe um nome de arquivo e um objeto **Vintage** como argumentos. Ele cria um objeto FileOutputStream para abrir o arquivo em modo de escrita, em seguida, cria um ObjectOutputStream para escrever o objeto **Vintage** no arquivo. Por fim, o método fecha o ObjectOutputStream. Se ocorrer algum problema durante a abertura do arquivo, ele retorna o valor 1, e se ocorrer algum problema durante a escrita, retorna o valor 2. Caso a operação seja bem-sucedida, retorna 0.

Por outro lado, o método *dadoscarregar* recebe um nome de arquivo como parâmetro. Ele cria um objeto FileInputStream para abrir o arquivo em modo de leitura, em seguida, cria um ObjectInputStream para ler o objeto **Vintage** do arquivo. Por fim, o método fecha o ObjectInputStream e retorna o objeto **Vintage** lido.

5 Controller

5.1 Controlador

A seguinte classe é responsável por gerenciar a interação do utilizador com o sistema, processar os comandos e de controlar o fluxo do programa.

- **ControladorArtigo** ca - Esta classe é responsável por lidar com as operações relacionadas ao registro de diferentes tipos de artigos, como T-Shirts, Malas e Sapatilhas.
- **ControladorUtilizador** cu - Esta classe é responsável por controlar as operações relacionadas aos utilizadores do sistema.
- **ControladorTransportadoras** ct - Esta classe é responsável por controlar as operações relacionadas a transportadoras do sistema.
- **ControladorEncomenda** ce - Esta classe é responsável por controlar as operações relacionadas a encomendas do sistema.
- **Input** in - Esta classe é responsável por lidar com a entrada de dados fornecida pelo utilizador.

Além disto, a classe *Controlador* apresenta o método *interpretador*. Este é responsável por realizar a interação com o utilizador e processar os comandos fornecidos.

Dentro do método *interpretador*, há um loop que verifica o estado atual da sessão do utilizador. Se não houver sessão ativa, é exibido o menu inicial com opções como login/registo, salvar/carregar dados em arquivo, criar transportadora, avançar no tempo, entre outros. Se houver uma sessão ativa, é exibido o menu principal com opções como fazer logout, consultar estatísticas, consultar produtos, colocar produto para vender, fazer encomenda, devolver encomenda, entre outros.

6 View

6.1 Apresentação

A seguinte classe é responsável por exibir informações e interagir com o utilizador através de mensagens de texto.

- **ViewMain** am - Esta classe é responsável por exibir menus e mensagens relacionadas à interface principal da aplicação Vintage.
- **ViewLogin** al - Esta classe é responsável por exibir menus e mensagens relacionadas ao processo de login e registo.
- **Output** out - Esta classe é responsável por exibir informações e mensagens no terminal de saída.

7 Descrição da Aplicação

Para a interação com o utilizador foi escolhida uma interface baseada em linha de comandos.

7.1 Menu Boas-Vindas

Após o utilizador entrar no programa, é apresentado um menu de boas-vindas. A partir daí, o cliente pode pressionar qualquer tecla para continuar.

Figura 2: Menu Boas-Vindas

7.2 Menu Principal

Este menu é o primeiro menu ao qual o utilizador tem acesso, logo após o menu de boas-vindas.

Neste menu, o cliente dispõe das seguintes opções:

1. **Login/Registar**

   Menu que permite fazer login ou registar um utilizador

2. **Guardar o Estado da Vintage para um ficheiro .dat**

   Guarda o estado atual do programa para um ficheiro .dat

3. **Carregar o Estado da Vintage para um ficheiro .dat**

   Carrega o estado do programa a partir de um ficheiro .dat

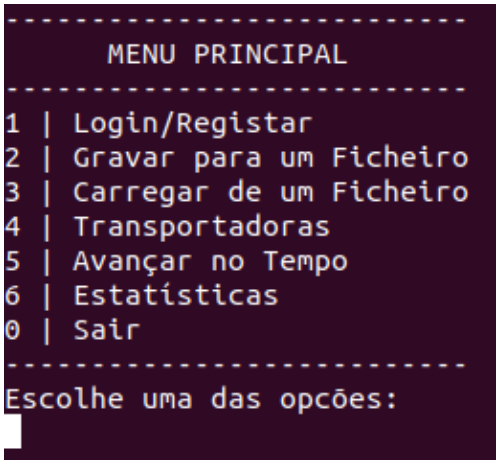


Figura 3: Menu Principal

4. **Transportadoras**

   São apresentadas duas opções: uma para criar uma transportadora e outra para editar a transportadora.

5. **Avançar no Tempo**

   Permite ao utilizador avançar no tempo

6. **Estatísticas**

   Menu que permite ao utilizador analisar as estatísticas

7.3 Menu Login

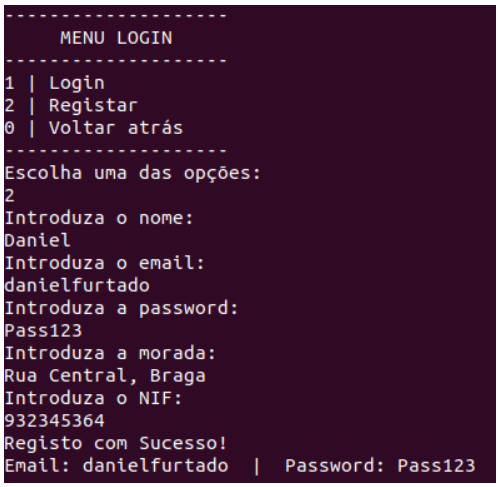


Figura 4: Menu Login

Neste menu, o cliente dispõe das seguintes opções:

1. **Login**

Para um utilizador que já esteja registado fazer login, apenas é necessário o email e a password.

2. **Registar**

   Para um utilizador se registar, é necessário preencher o nome, o email, a password, a morada e o número de identificação fiscal (NIF).

7.4 Menu Utilizador

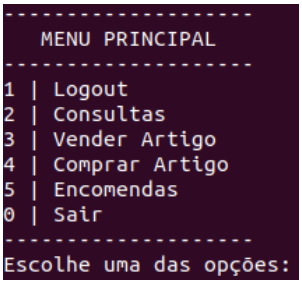


Figura 5: Menu Utilizador

Neste menu, o cliente dispõe das seguintes opções:

1. **Logout**

   Permite ao utilizador efetuar *logout*

2. **Consultas**

   Menu que permite consultar dados da aplicação sobre o utilizador.

3. **Vender Artigo**

   Menu que permite colocar artigos a venda.

4. **Comprar Artigo**

   Menu que permite iniciar uma nova encomenda.

5. **Encomendas**

   Menu que permite devolver uma encomenda ou adiar a data de devolução.

7.5 Menu Estatísticas

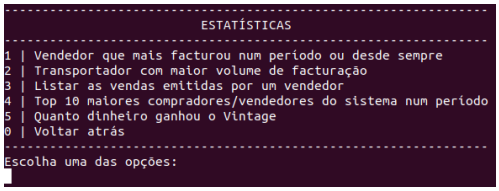


Figura 6: Menu Estatísticas

Neste menu, o cliente dispõe das seguintes opções:

1. **Vendedor que mais faturou num período ou desde sempre**

   A estratégia para resolver esta estatística foi consultar todos os utilizadores do programa e percorrer as "faturas"de cada utilizador, ou seja, verificar para os dias que estão dentro do período escolhido e somar o dinheiro que foi vendido nesses dias. Após obter o total de faturação de cada utilizador de cada cliente é escolhido o vendedor com maior vendas.

2. **Transportadora com maior volume de faturação**

   Para determinar qual a transportadora com maior volume de faturação apenas é necessário comparar a variável de cada transportadora, que representa o lucro obtido pela mesma desde a sua criação. Depois apenas temos que selecionar a transportadora com maior valor.

3. **Listar as vendas emitidas por um vendedor**

   Nesta estatística decidimos que não faria sentido "listar as encomendas emitidas por um vendedor", tal como era referido no enunciado do trabalho prático, visto que, cada vendedor apenas vende artigos e cada encomenda tem artigos de vários vendedores. Então calculamos as vendas emitidas por um vendedor. Para isso, apenas apresentamos a lista de artigos vendidos que pertence ao utilizador correspondente ao valor de *email* introduzido como *input*.

4. **Top 10 maiores compradores/vendedores dos sistema num período**

   A estratégia para mostrar os 10 maiores compradores/vendedores num período foi: após ser introduzido como *input* o número de dias anteriores que quer incluir no período, são verificados todos os artigos vendidos e comprados para cada utilizador e serão verificados nas encomendas quando os produtos foram vendidos, caso a data esteja entre o período estipulado, então é acumulado o valor desse produto de venda nesse momento. Depois é calculado o top 10 de utilizadores.

5. **Quanto dinheiro ganhou a Vintage** O dinheiro que a aplicação Vintage ganhou desde o início do programa é calculado através da soma do valor de todas as encomendas já realizadas desde o início da aplicação.

7.6 Menu Consultas

Este menu serve para o utilizador verificar quais os produtos que tem a vender, que comprou e que vendeu, tal como as transportadoras disponíveis.

Neste menu, o cliente dispõe das seguintes opções:

1. **Produtos a Vender**

   Apresenta a lista de todos os produtos que estão a venda do respetivo utilizador.

2. **Produtos Comprados**

   Apresenta a lista de todos os produtos que foram comprados do utilizador.

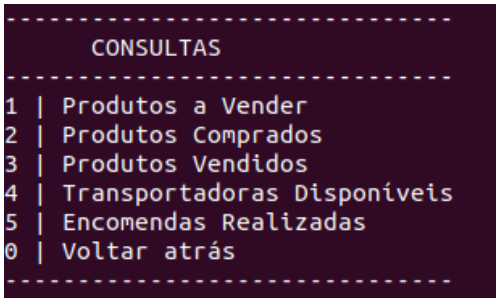


Figura 7: Menu Consultas

3. **Produtos Vendidos**

   Apresenta a lista de todos os produtos que foram vendidos pelo utilizador.

4. **Transportadoras Disponíveis**

   Apresenta a lista de todas as transportadoras disponíveis do programa.

5. **Encomendas realizadas**

   Apresentas as encomendas realizadas pelo utilizador.

7.7 Menu Encomenda

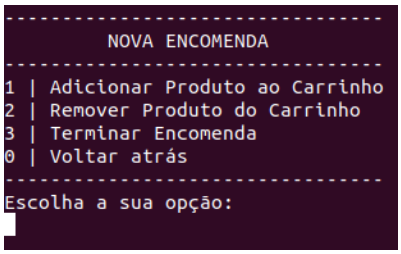


Figura 8: Menu Encomenda

Neste menu, o cliente dispõe das seguintes opções:

1. **Adicionar produto ao carrinho**

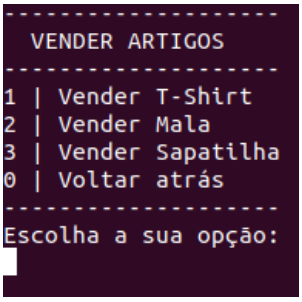
   Permite adicionar um artigo ao carrinho de compras através do ID do artigo.

2. **Remover Produto do Carrinho**

   Permite remover um artigo do carrinho de compras através do seu ID.

3. **Terminar Encomenda**

   Permite concluir a encomenda.



7.8 Menu Venda

Neste menu, o cliente dispõe das seguintes opções:

1. **Vender T-Shirt**

   Permite registar uma T-Shirt.

2. **Vender Mala**

   Permite registar uma Mala.

3. **Vender Sapatilha**

   Permite registar uma Sapatilha.

8 Conclusão

De um modo geral, consideramos que estamos bastante satisfeitos com o resultado obtido, visto que apesar de não termos cumprido com todos os requisitos exigidos, sentimos que foi possível aprofundar os conhecimentos adquiridos ao longo do semestre obedecendo ao paradigma da Programação Orientada a Objetos.

Durante a realização deste projeto, deparamos-nos com alguns desafios significativos. Em particular, ao tentar compreender o funcionamento das encomendas na aplicação Vintage e na organização do trabalho em si durante a fase inicial.

Um aspeto que poderíamos ter abordado diferente e mais específico é a forma das datas no programa, no nosso trabalho, usamos a biblioteca *LocalDate*, no entanto esta biblioteca apenas trata data até ao dia e não trata as horas. No enunciado é referido que o prazo limite de entrega da encomenda por defeito é 48 horas no entanto no nosso caso como usamos o *LocalDate* o prazo pré definido são 2 dias. Uma alternativa seria o uso do *LocalDateTime* que é mais específico.

Por fim, a aplicação Vintage demonstrou a relevância da POO na criação de sistemas eficientes e flexíveis. O trabalho proporcionou uma compreensão aprofundada dos conceitos e práticas da POO, preparando para desafios futuros no desenvolvimento de software.