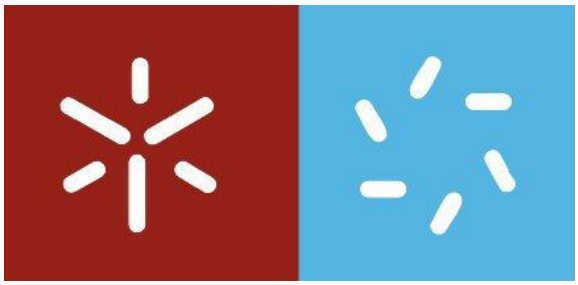


Universidade do Minho

Licenciatura em Ciências da Computação

2022/2023

POO – Trabalho Prático

Grupo 22

Alexandra Calafate (A100060) João Vieira (A100052)

Conteúdo

1 Introdução

Este projeto consistiu em desenvolver um simulador de comércio de artigos Vintage, mais especificamente sapatilhas, t-shirts, malas e outros. Tem como objetivos gerir utilizadores, compra e venda de artigos, transportadoras dos mesmos e encomendas.

Consideramos que o maior desafio foi na gestão de encomendas, sendo estas afetadas pelos possíveis saltos no tempo.

2 Classes

2.1 Artigos

```
private Tipo tipo;
private boolean estado;
private int ndonos;
private String descricao;
private String marca;
private String codigo;
private double preco;
private int correcao = 0;
private sapatilhas sapatilha;
private tshirts tshirt;
private malas mala;
private transportadora transp;
private boolean disponivel;
private static datemanager data = datemanager.getInstance();
DecimalFormat df = new DecimalFormat("#.##");
```

A classe artigos é onde se encontram os construtores dos mesmos, e gere toda a informação de um artigo. Possui várias características de um artigo, como o preço, descrição, estado de utilização, entre outros. As propriedades sapatilha, tshirt e mala são definidas de acordo com a propriedade tipo, construída num Enum.

Uma das funções da classe artigos é calcular o preço de um artigo dependendo dos seus diferentes atributos, sendo também necessário a data atual no programa, daí a sua definição nas propriedades.

Tem também os getters e setters das diferentes propriedades, e uma função ToString.

2.2 Sapatilhas

```
private boolean premium;
private int tamanho;
private boolean atacadores;
private String cor;
private int data;
```

A classe Sapatilhas é composta pelo seu construtor, getters e setters para os diferentes atributos. Tem a informação do tamanho do calçado, se é premium, qual o seu ano de edição, entre outros.

2.3 T-Shirts

```
private Tamanho tamanho;
private Padroes padrao;

enum Padroes {
    Liso,
    Riscas,
    Palmeiras
}

enum Tamanho {
    S,
    M,
    L,
    XL
}
```

A classe Tshirts tem os atributos representados acima, tal como os getters e setters, e o construtor do objeto tshirt.

2.4 Malas

```
private boolean premium;
private int dimx;
private int dimy;
private String material;
private int data;
```

A classe Malas tem o construtor e os getters e setters das propriedades aqui representadas;

2.5 Utilizadores

```
private String systemcode;
private String email;
private String password;
private String nome;
private String morada;
private int nfiscal;
private double lucro = 0;
private double prejuizo = 0;
private ArrayList<encomendas> encomendas;
private ArrayList<encomendas> pendentes;
private ArrayList<artigos> artavenda = new ArrayList<>();
private ArrayList<artigos> artvendidos = new ArrayList<>();
private ArrayList<artigos> artcomprados = new ArrayList<>();
```

A classe Utilizadores contém toda a informação de um utilizador registado no sistema. O **systemcode** é atribuído automaticamente após criação. Tem atributos designados para saber quanto o utilizador já gastou e quanto já lucrou ao longo do seu tempo na Vintage. Contém também várias listas, sendo **encomendas** para as encomendas realizadas, **pendentes** para as encomendas ainda no prazo de cancelamento (48h), **artavenda** designa todos os artigos que o utilizador tem à venda, **artvendidos** para os que já vendeu e **artcomprados** para os adquiridos ao longo do programa.

Para além dos habituais construtores, getters e setters, tem várias funções:

- **getCarrinho()** vai buscar o último elemento da lista **encomendas**, e caso este não esteja pendente, cria um carrinho novo;
- **listarArtigo()** adiciona um artigo à lista **artavenda**;
- **printListaArts()** imprime uma das listas **artavenda**, **artvendidos** ou **artcomprados**, dependendo dos argumentos;
- **artigoVendido()** verifica todos os artigos em **artavenda** e caso os mesmos já não estejam disponíveis, remove-os do stock e adiciona-os em **artvendidos**;
- **carrinhotem()** diz se o carrinho atual contém o artigo dado;
- **getTotalVendas()** e **getTotalComprado()** calcula o custo total de todos os artigos das respetivas listas;
- **imprimePendentes()** imprime a lista **pendentes;**

Tem também funções para clonar um utilizador e verificar se dois utilizadores são iguais.

2.6 Contas

```
private Map<String, utilizadores> contas;
private int utilcounter;
```

A classe Contas gere todos os objetos da classe utilizadores que sejam inseridos no sistema. É capaz de adicionar novos utilizadores, ir buscar um determinado utilizador, determinar o melhor vendedor atual, a transportadora com mais lucro, e listas dos utilizadores ordenadas por quem mais gastou e quem mais lucrou.

Esta é a classe onde é analisado e mandado para impressão todo o stock atual, ou seja, todos os artigos disponíveis à venda de cada utilizador.

Contém também os habituais getters e setters, o construtor e a função ToString.

2.7 Encomendas

```
private String nome;
private Dimensao dimensao;
private Estado estado;
private double preco;
private int numero_artigos;
private String codigo;
private datemanager data = datemanager.getInstance();
private LocalDate data_de_criacao;
ArrayList<artigos> artigos;
DecimalFormat df = new DecimalFormat(pattern:"#.##");
```

A classe Encomendas contém toda a informação de uma encomenda realizada por um determinado utilizador. Os seus atributos são o nome do utilizador, a dimensão da encomenda determinada pelo número de produtos, o seu estado de envio (pendente, finalizada ou expedida), o preço total com taxas aplicadas, a data de criação da encomenda e, claro, uma lista de todos os artigos encomendados.

Para além do construtor, getters e setters e ToString, tem também uma função para calcular o preço total da encomenda (tendo aqui peso as taxas das transportadoras e o estado dos artigos), funções de adicionar ou remover artigos da encomenda e diferentes funções de impressão da encomenda, para diversas ocasiões.

2.8 Gestor Encomendas

```
private Set<encomendas> encomendas;
DecimalFormat df = new DecimalFormat(pattern:"#.##");
```

A classe Gestor Encomendas armazena todas as encomendas realizadas no sistema. Fora os getters, setters, construtores e ToString, esta classe também é responsável por adicionar e remover uma encomenda, concluir uma encomenda, isto é, finalizá-la, e também calcular o lucro total da Vintage, pois é capaz de iterar por todos os artigos de todas as encomendas e as respetivas transportadoras.

2.9 Transportadora

```
private String transportadora;
private boolean premium;
private int margemlucro;
private double lucro;
```

A classe Transportadora é apenas composta pelos seus construtores, getters e setters. Contém informações da transportadora, como o nome, o estatuto premium, a sua taxa e o seu lucro na Vintage.

2.10 Gestor Transportadoras

```
private ArrayList<transportadora> transportadoras;
private int counter;
DecimalFormat df = new DecimalFormat(pattern:"#");
```

A classe Gestor Transportadoras funciona para a classe Transportadora da mesma maneira que a classe Contas funciona para a classe Utilizadores. Gere as diferentes transportadoras, e é capaz de as imprimir, com a possibilidade de as filtrar por transportadoras premium ou não, e calcular qual delas teve maior lucro até a atualidade.

Tem também funções de adicionar e remover transportadoras, tal como construtores, getters e setters.

2.11 Colors

```
public static final String RESET = "\u001B[0m";
public static final String BLACK = "\u001B[30m";
public static final String RED = "\u001B[31m";
public static final String GREEN = "\u001B[32m";
public static final String YELLOW = "\u001B[33m";
public static final String BLUE = "\u001B[34m";
public static final String PURPLE = "\u001B[35m";
public static final String CYAN = "\u001B[36m";
```

Esta classe foi apenas definida de forma a tornar a interface (feita no terminal) mais apelativa e de melhor compreensão, dando diferentes cores a títulos, atributos, índices e outros.

2.12 Date Manager

```
private static datemanager instance;
private LocalDate currentDate;
```

Esta classe monitoriza a data atual do programa, e é capaz de a alterar, avançando uma quantia de dias fornecida pelo utilizador do Vintage.

2.13 MainV

Classe inicial do projeto. Chama os necessários construtores para, aquando testado o programa, já existirem alguns artigos, utilizadores, transportadoras e encomendas no sistema da Vintage. Após a construção de todos os objetos, estes são distribuídos organizadamente e é chamada a função **controlo.run()** da classe **Controlo**.

2.14 Controlo

A classe Controlo é onde ocorrem a maioria das ações. É responsável por apresentar os diferentes ecrãs da interface através da classe **Vintage**, e chamar diferentes funções dependendo das ações do utilizador. Aqui é possível fazer login, criar uma nova conta, ver que contas e transportadoras existem, aceder ao Menu Estatísticas, e ainda avançar no tempo.

A classe tem também uma função update, que mantêm o sistema a par de todos os processos que aconteceram, atualizando os parâmetros necessários.

2.15 Controlo Utilizador

A classe Controlo Utilizador está por trás de todas as operações realizadas dentro de uma determinada conta, isto é, desde que é efetuado o login e até o logout. Aqui é possível listar um artigo, comprar vários artigos, efetuar alterações no carrinho e concluir uma encomenda.

Pode-se verificar que artigos estão à venda por parte do utilizador, que artigos comprou e vendeu. É também possível ver se existem encomendas pendentes (dentro do prazo de cancelamento) e poder pedir reembolsos.

2.16 Controlo Estatísticas

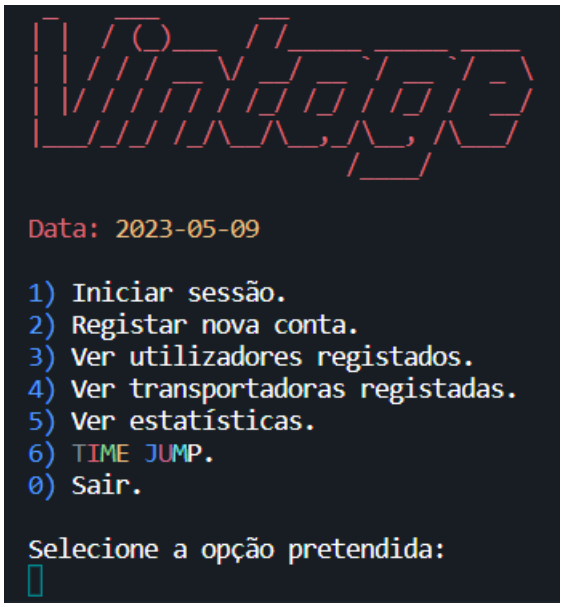
A classe Controlo Estatísticas monitoriza e efetua as necessárias mudanças por trás do Menu Estatísticas.

2.17 Vintage

Esta classe é responsável por apresentar os diferentes ecrãs da Vintage. Tem várias funções, como o **MenuInicial**, **MenuUtilizador**, **MenuEstatísticas**, entre outros. É a maior classe do projeto, visto que contém tudo o que um utilizador do Vintage vê. No ponto 3 serão apresentados alguns ecrãs desta classe.

3 O Projeto em Funcionamento

Ao longo da utilização do programa, são nos apresentados diversos ecrãs:



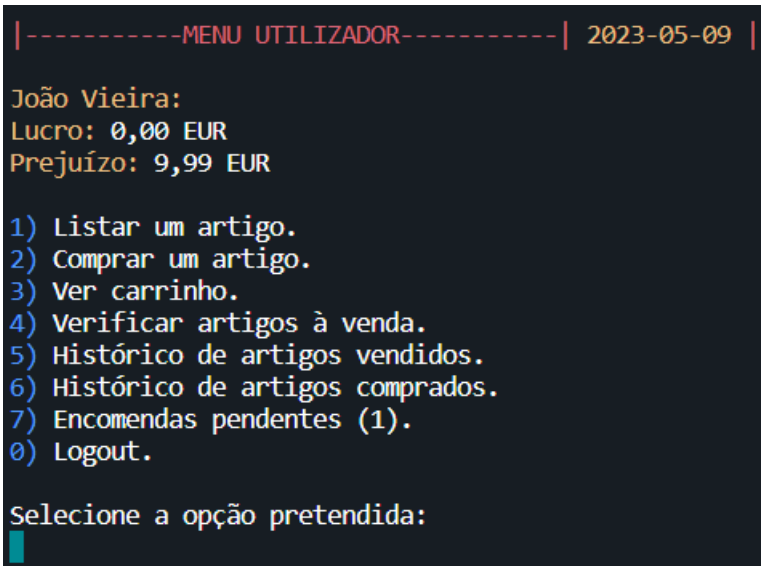
1 - Menu Inicial

Este é o primeiro ecrã apresentado ao utilizador. Indica a data e todas as opções.

O número inserido é enviado para a classe **Controlo** que efetua o determinado processo.

Ao **Iniciar sessão**, ou **Registar nova conta**, é iniciado o **Controlo Utilizador**.

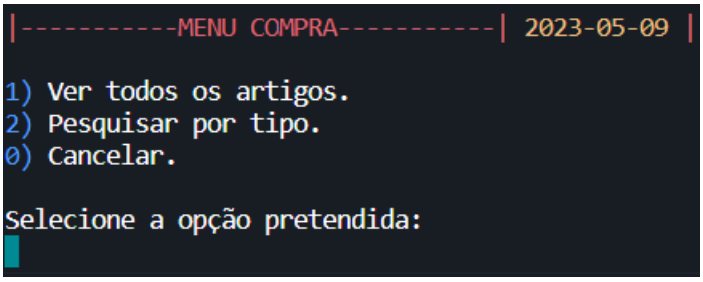
Ao escolher a opção **Ver estatísticas**, é iniciado o **Controlo Estatísticas**.



2 - Menu Utilizador

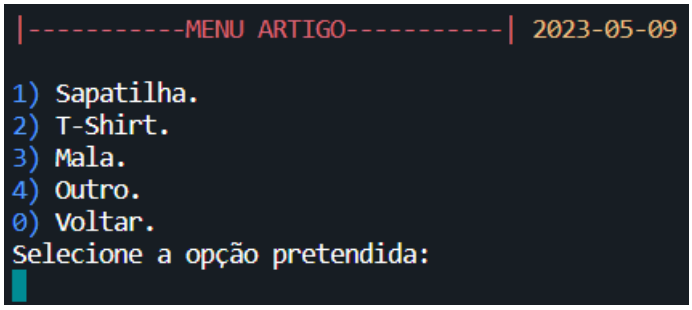
A classe **Controlo Utilizador** controla todos os processos que acontecem neste menu. É apresentada a data atual, tal como o nome do utilizador, e o dinheiro gasto e ganho pelo mesmo.

Se quisermos, por exemplo, comprar um artigo, inserimos "2" no terminal, e acontece o seguinte:



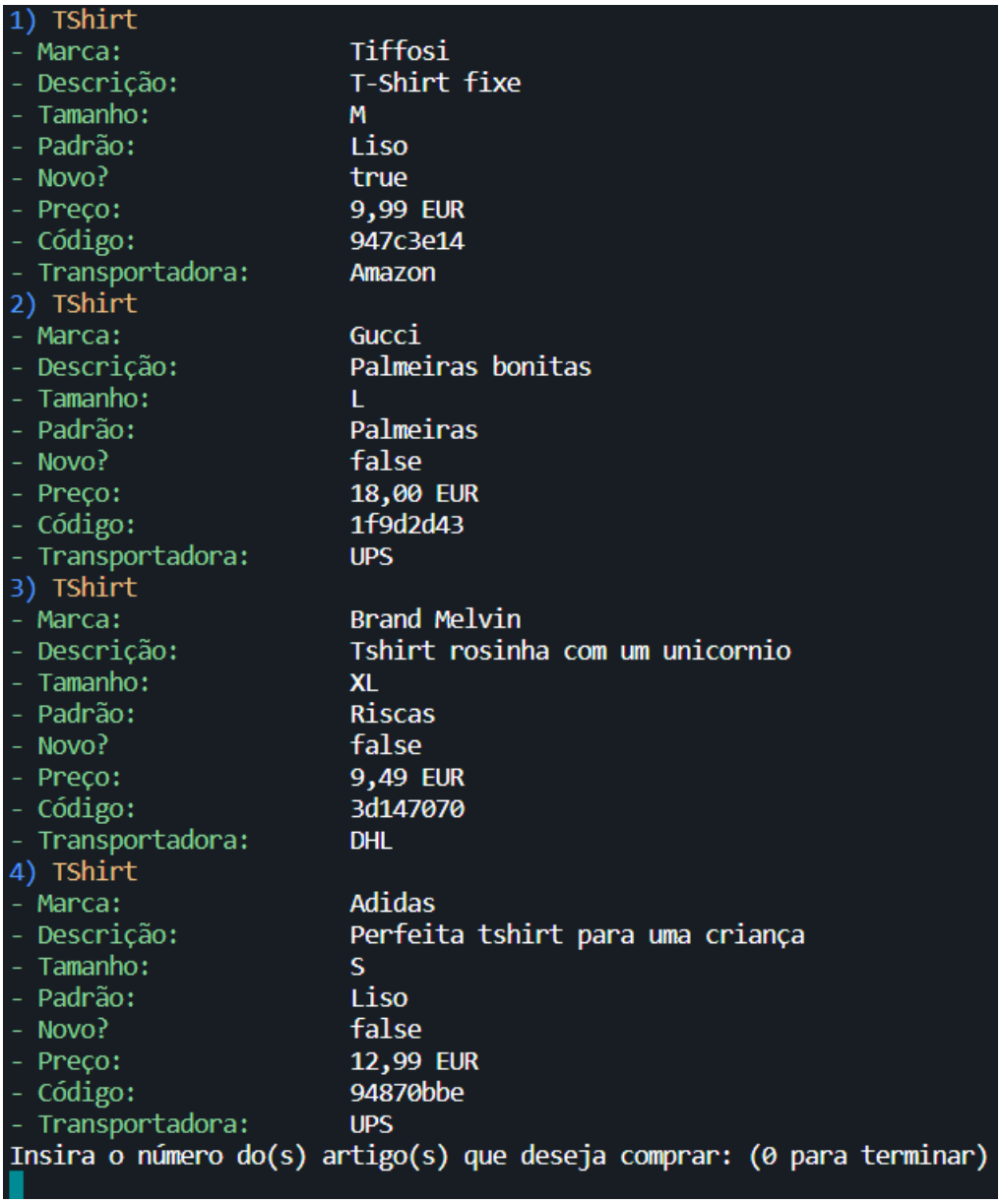
3 - Menu Compra

Novamente, pressionamos "2", para ver os diferentes tipos de artigo:



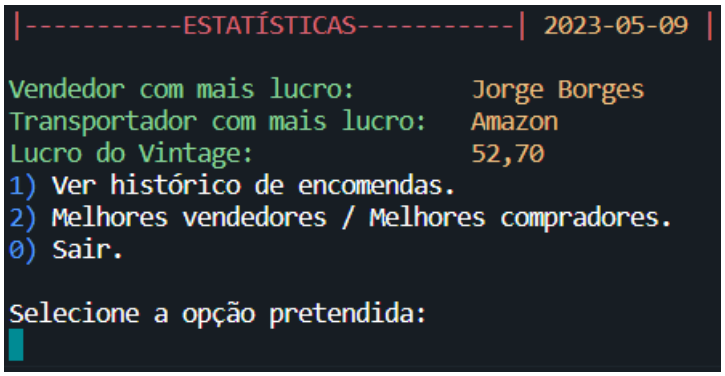
4 - Menu Artigo

E caso estejamos interessados em ver apenas T-Shirts, voltaremos a inserir "2" e ser-nos-á apresentado:



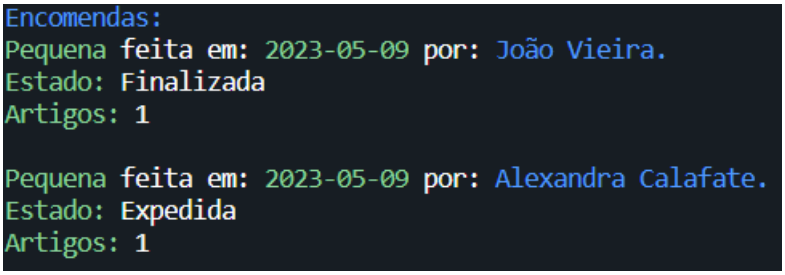
5- Lista de T-Shirts disponíveis

Aqui, podemos escolher vários artigos, pressionando apenas os seus índices e, para terminar, pressionar 0 e voltar ao **Menu Utilizador**.



5 - Menu Estatísticas

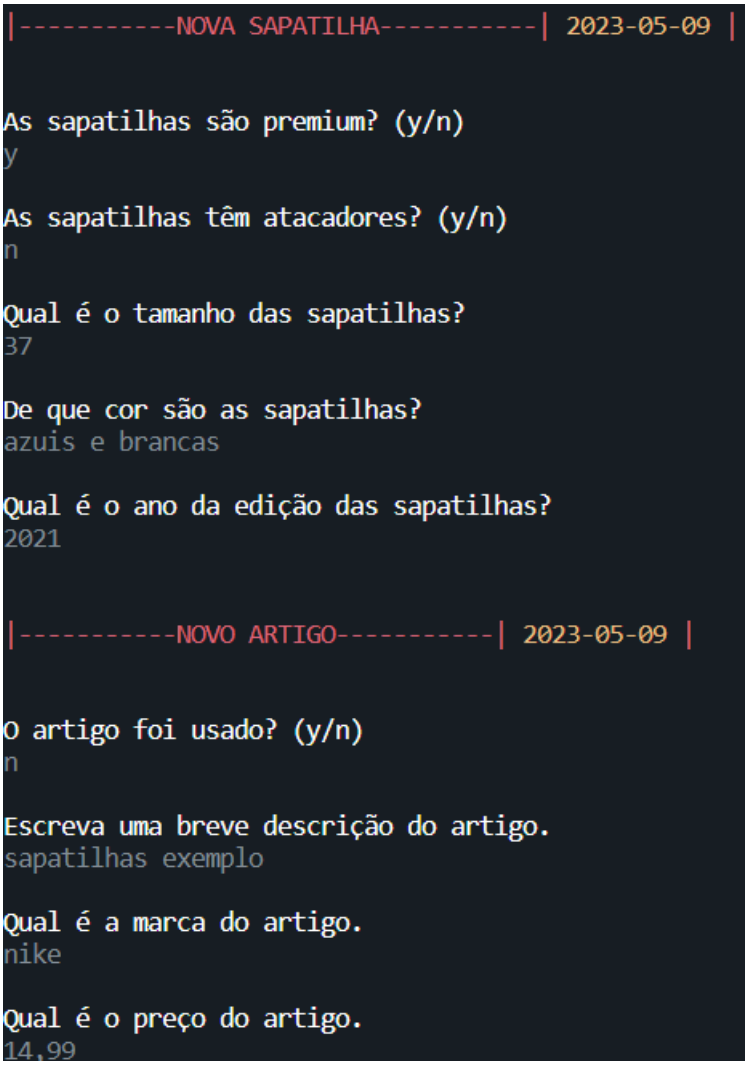
Este é o menu apresentado ao escolhermos a opção 5 no **Menu Inicial**. Aqui é possível ver o histórico de encomendas, ao pressionar "1", que demonstra o seguinte:



6 - Histórico de encomendas

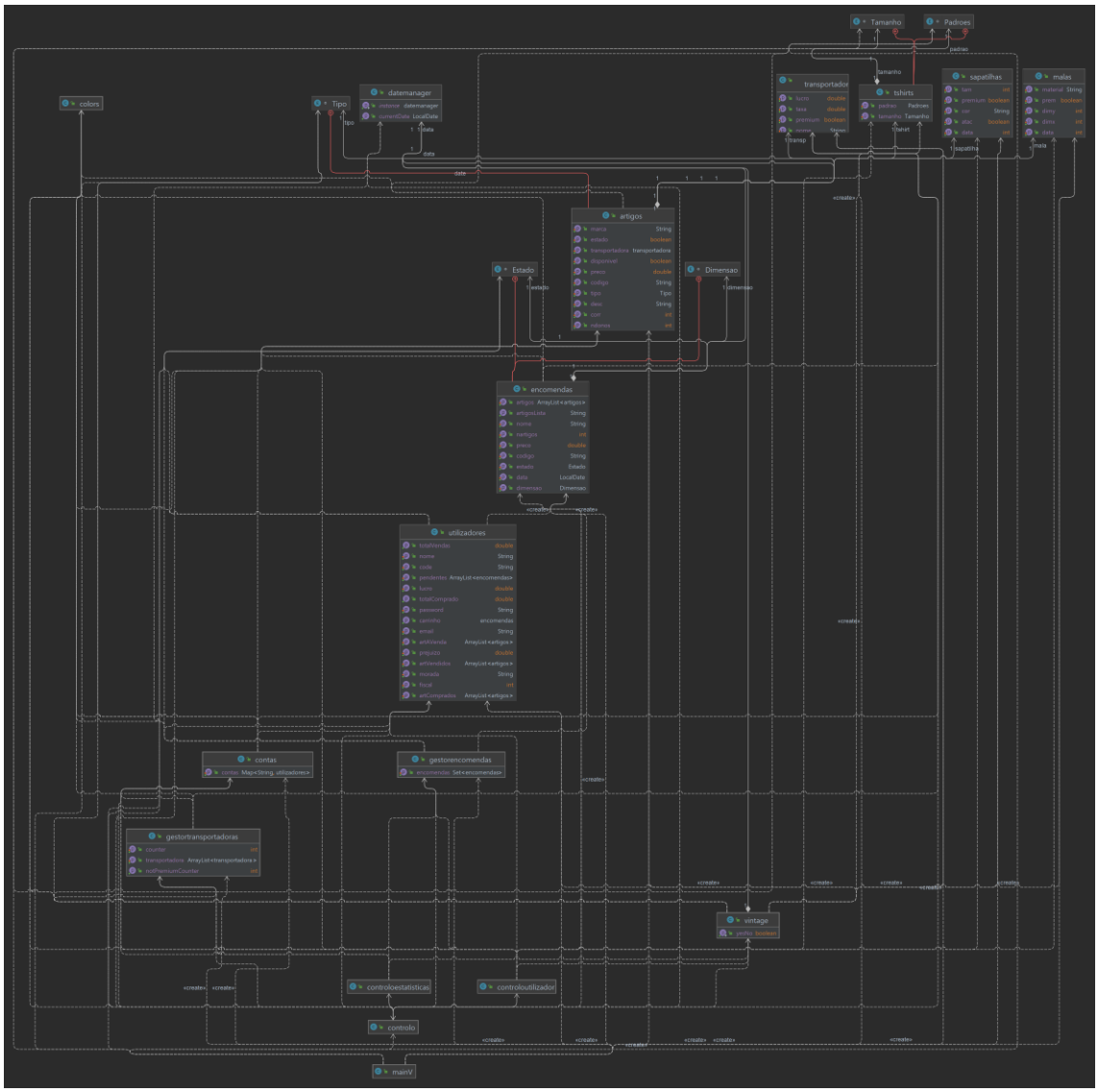
Quando se pretende criar algum objeto novo, sendo ele um artigo, um utilizador ou uma transportadora, são feitas ao utilizador uma quantidade de perguntas para obter a informação necessária e criar esse tal objeto.

No exemplo a seguir, estão representadas uma sequência de perguntas feitas ao utilizador, após ele ter escolhido listar um artigo novo, sendo este uma sapatilha:



7 - Perguntas para criar uma sapatilha

4 Diagrama de Classes



5 Conclusão

A nível geral, acreditamos ter um projeto bem conseguido. Acreditamos que respondemos corretamente aos desafios propostos, exceto a parte da automatização, e conseguimos ultrapassar o maior obstáculo que tivemos, sendo ele a gestão das encomendas e tudo o que a mudança dos seus estados traz.

O projeto despertou vontade a entender mais sobre Java e as suas funcionalidades, e aumentou o nosso conhecimento da linguagem.