Universidade do Minho
Mestrado Integrado em Engenharia Informática
Licenciatura em Ciências da Computação

Programação Orientada aos Objetos

Ano Letivo de 2019/2020

TrazAqui! APP

Grupo30

Gonçalo Pinto Nogueira
A88617

João Nuno Costa Neves
A81366

Marco António Rodrigues Sampaio
A85508

Resumo

O trabalho proposto no âmbito da disciplina de Programação Orientada aos
Objetos (POO), tem por objetivo a criação de uma plataforma em JAVA que permita conjugar as atuais necessidades de entregar encomendas a pessoas que estão confinadas nas suas habituações.

A aplicação proposta deve dar suporte a toda a funcionalidade que permita a existência de voluntários a fazerem entregas de encomendas a um utilizador, empresas a entregar encomendas e lojas que podem aderir a este serviço para que possam disponibilizar encomendas a ser entregues.

A conceção da aplicação segue uma abordagem dentro do paradigma de
Orientação a Objetos e no presente relatório é explicado a arquitetura adotada e mostradas as funcionalidades disponíveis.

Índice

Conteúdo

Índice de Ilustrações

1-Diagrama de Classes

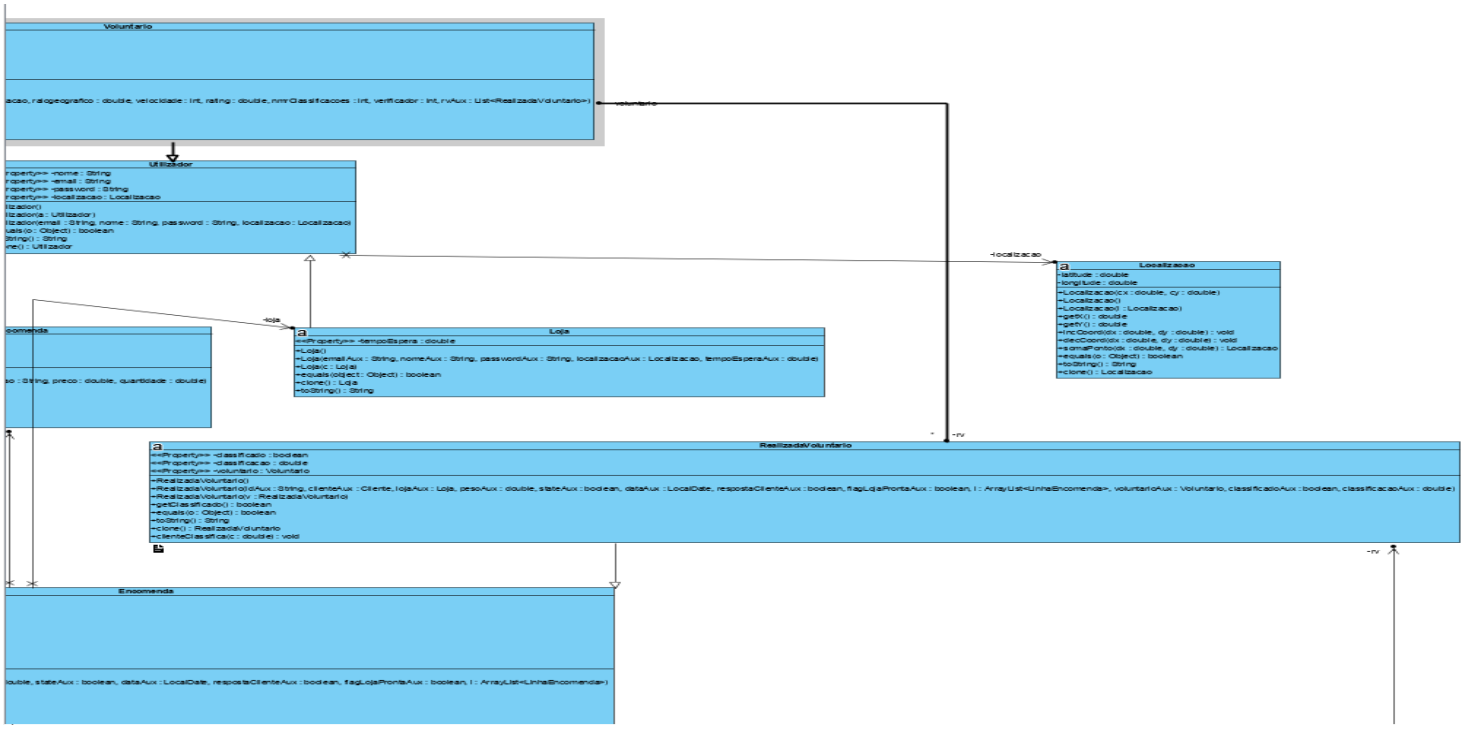

Figura 1 Diagrama de Classes 1

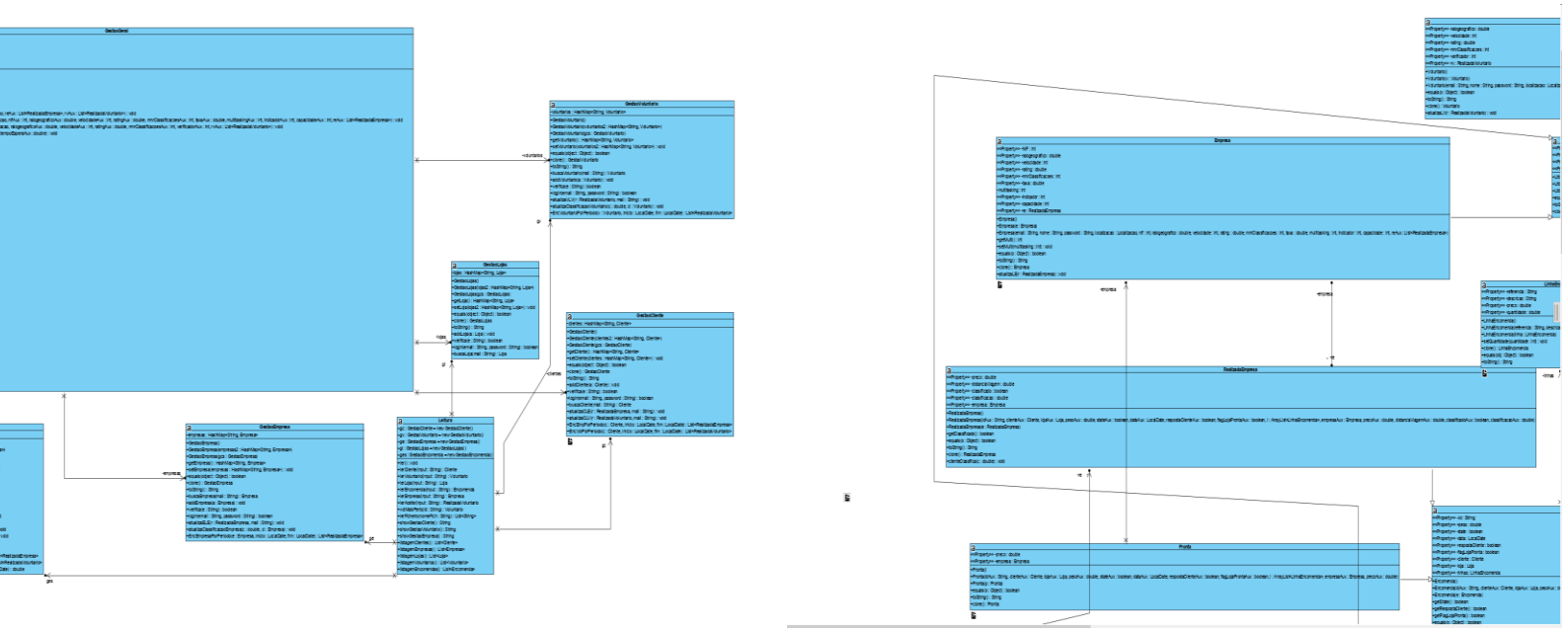

Figura 2 Diagrama de Classes 2 (imagem partida de forma a facilitar análise)

2- Classes

Começamos o projeto por implementar as classes mais básicas (Cliente, Loja, Voluntário, Empresa) com os respetivos atributos e métodos, à medida que o projeto se foi desenvolvendo tivemos de nos adaptar com mais algumas classes (em baixo ilustradas).

A arquitetura implementada para as funcionalidades segue uma estratégia de camadas, isto é, cada entidade acaba por ter uma classe dedicada à sua gestão. Estas classes de gestão são chamadas na GestãoGeral que é chamada na classe Main (TrazAqui!).

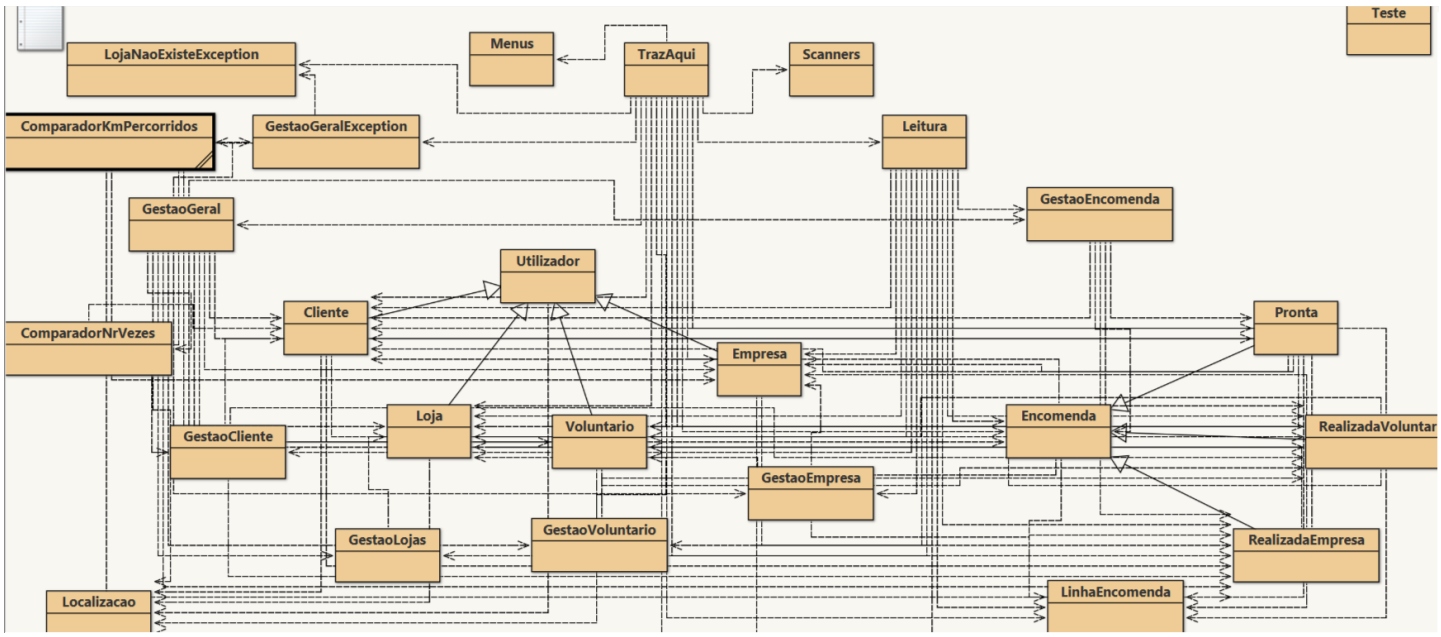

Figura 3 Classes

2.1- Utilizador

Começamos por criar a *Classe Utilizador* que representa quatro tipos distintos de atores do sistema, sendo eles o Cliente, Voluntário, Empresa e Loja,
tendo em conta que possuem atributos comuns, tais como: nome, email password e localização.

Decidiu-se criar Utilizador como Superclasse com os atributos referidos anteriormente e a partir daí as subclasses desta, uma vez que são herdados todos os atributos reutilizando assim o código.

Em relação às subclasses anteriormente abordadas, foram definidas também, as *Classes* de Gestão para cada respetiva subclasse que guardam em HashMap todos os utilizadores inscritos no sistema sendo a "key" o email das respetivas entidades.

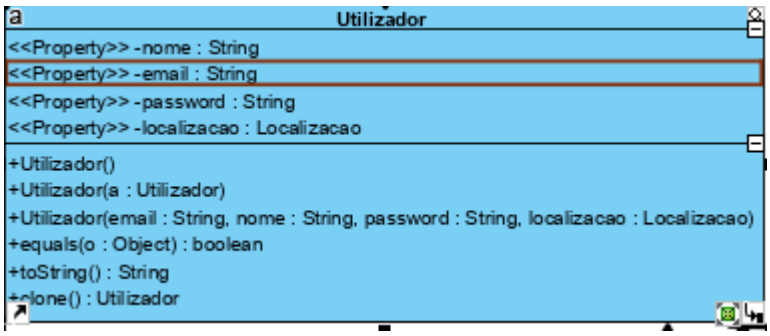

Figura 4 Utilizador

2.2- Cliente

A subclasse Cliente (de Utilizador) define a entidade que realiza os pedidos de encomendas, além dos atributos herdados possui também uma lista de pedidos realizados por voluntários e por empresas respetivamente.

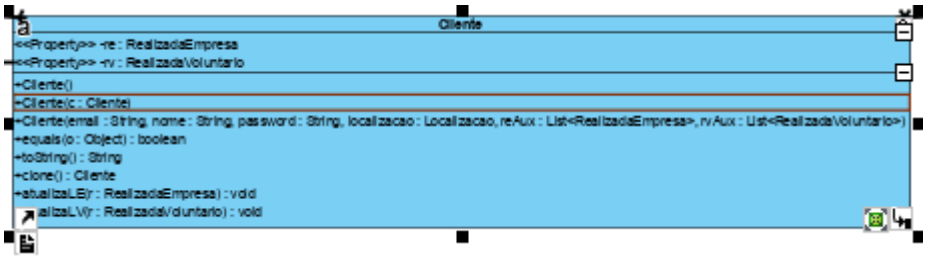

Figura 5 Cliente

2.3- Empresa

A subclasse Empresa (de Utilizador) define a entidade que entrega uma parte dos pedidos solicitados, além dos atributos herdados possui também o NIF, um raio geográfico para limitar as suas entregas, velocidade de entrega, um rating, nº de classificações à empresa, uma taxa, um indicador de multitasking, um indicador, um valor de capacidade de encomendas que consegue entregar ao mesmo tempo, e uma lista de encomendas realizadas pela empresa em questão.

Figura 6 Empresa

2.4- Voluntário

A subclasse Voluntário (de Utilizador) define a outra entidade encarregada de entregar pedidos solicitados. Para além dos atributos herdados possui também, tal como as empresas, um raio geográfico, velocidade de entrega, rating, um nº de classificações, um verificador e uma lista de encomendas realizadas pelo voluntário em questão.

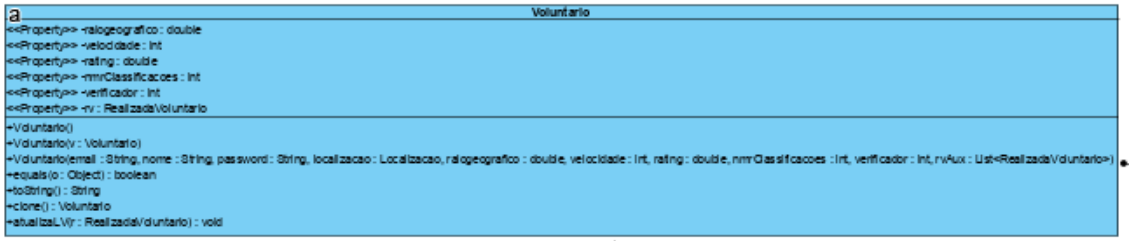

Figura 7 Voluntário

2.5- Loja

A subclasse Loja (de Utilizador) define a entidade com a função de receber os pedidos solicitados pelos clientes. Para além dos atributos herdados possui também um tempo de espera.

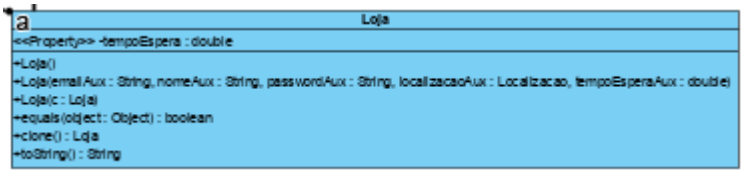

Figura 8 Loja

2.6- Localização

Foi criada a classe Localização, com os atributos latitude e longitude. Esta implementação permite-nos usufruir de métodos importantes para o funcionamento geral da aplicação, para saber por exemplo, a distância entre um Voluntário e uma Loja.

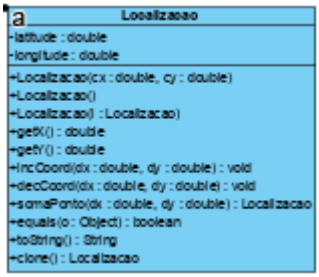

Figura 9 Localização

2.7- Linha de Encomenda

A classe Linha de Encomenda ajuda-nos a declarar os produtos respetivos a uma encomenda e tem como atributos uma referência, uma descrição, um preço e uma quantidade.

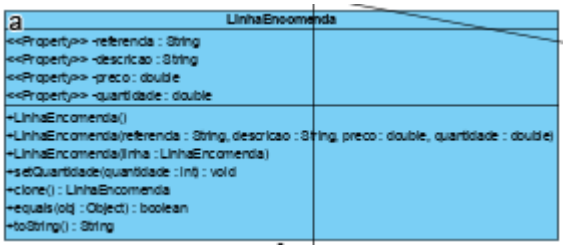

Figura 10 Linha de Encomenda

2.8- Encomenda

A classe Encomenda é uma das principais componentes do projeto. Tem como atributos um identificador, os respetivos cliente e loja, o seu peso, um estado (1 se for encomenda médica, 0 caso contrário), a data do dia em que foi solicitada, um booleano que representa se já foi enviado pedido ao cliente, um booleano que nos indica se a encomenda está pronta a ser entregue e um *ArrayList* com Linhas de Encomenda.

Esta classe é superclasse das classes Pronta (relativa aos pedidos de encomenda das empresas), RealizadaVoluntario (relativa às encomendas realizadas por voluntários), RealizadaEmpresa (relativa às encomendas realizadas por empresas).

Esta classe tem também uma classe de gestão associada.

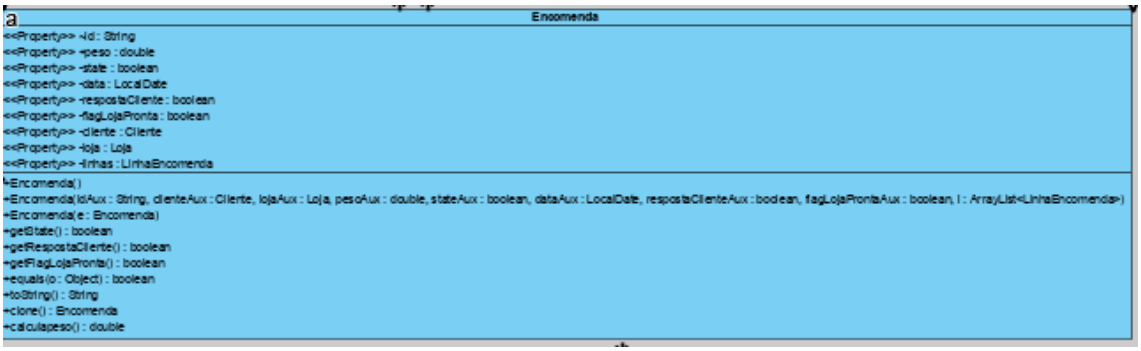

Figura 11 Encomenda

2.9- Gestão Geral

A classe GestaoGeral, foi criada, com o intuito de implementar os métodos mais gerais de inserção, gestão e pesquisa, trabalhando sobre as outras classes de gestão nomeadamente: Gestão Clientes, Gestão Empresas, Gestão Voluntários, Gestão Lojas e Gestão Encomendas.

Esta classe possui as funcionalidades às quais a aplicação tem de responder:

- Registar um qualquer utilizador;
- Permitir Logins dos utilizadores;
- Adicionar listas de entidades (a partir de um ficheiro logs);
- Listar todas as entidades gravadas no sistema;
- Listar as encomendas de uma loja que estão ou não prontas a ser entregues;
- Sinalizar a entrega de uma encomenda por parte de um voluntário ou uma empresa;
- Aceitar por parte de uma empresa o transporte de uma encomenda;
- Indicar por parte de uma loja que uma encomenda já está pronta para ser transportada;
- Indicar quantas pessoas estão em fila de espera;
- Apresentar ao cliente os pedidos a que tem de dar uma resposta;
- Apresentar ao cliente os pedidos que pode classificar;
- Classificar empresas e voluntários por parte dos clientes;
- Permitir ao cliente aceitar ou recusar pedidos;
- Calcular a distância entre localizações;
- Apresentar encomendas num determinado espaço de tempo;
- Solicitar encomendas;
- Apresentar a faturação de uma empresa num determinado espaço de tempo;
- Apresentar o top 10 clientes com mais encomendas;

- Apresentar o top 10 empresas com mais kms percorridos;
- Gravar um estado;
- Carregar um estado;

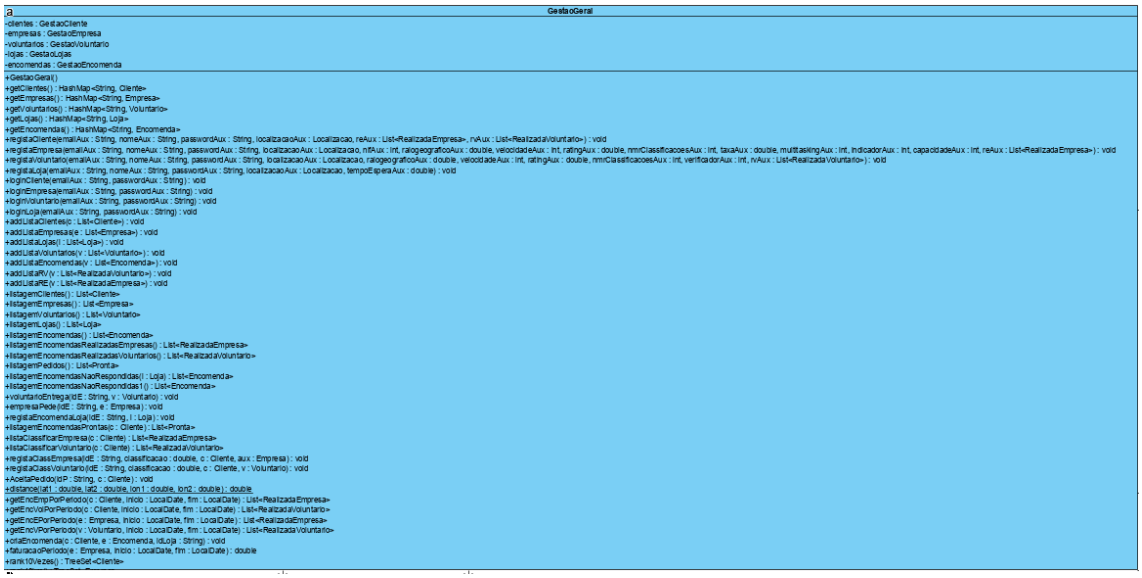

Figura 12 Gestão Geral

2.10- GestaoGeralException

A classe “GestaoGeralException” foi criada de forma a providenciar tratamento
específico aquando de um erro em relação à GestãoGeral. Assim, esta classe trata do
lançamento das variadíssimas exceções que a aplicação trata.

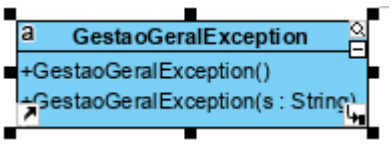

Figura 13 Exception 1

2.11- LojaNaoExisteException

A classe “LojaNaoExisteException” foi criada para lançar uma um aviso quando é solicitada uma loja inexistente.

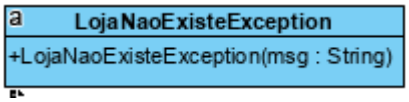

Figura 14 Exception 2

2.12- Comparador

Sendo um dos requisitos a listagem dos 10 clientes que mais usam o sistema em número de vezes e em quilómetros, foi necessária a implementação de dois comparadores, sendo eles *ComparadorKmPercorridos* e *ComparadorNrVezes.*

Estes dois possibilitam a criação de Collections do tipo Tree, que são ordenadas pelo

número de quilómetros e pelo número de alugueres.

Figura 15 Comparators

2.13- Menu

Para a apresentação do menu, foi criada a classe *Menu,* onde se encontram todos os métodos necessários para a iteração entre o ator e o sistema.

Figura 16 Menus

2.14- Leitura e Scanners

A classe Leitura é onde se trata da leitura de dados que posteriormente são guardados nas classes de gestão.

A classe Scanners é onde se trata os parâmetros passados de forma a que a aplicação tenha interação com o utilizador.

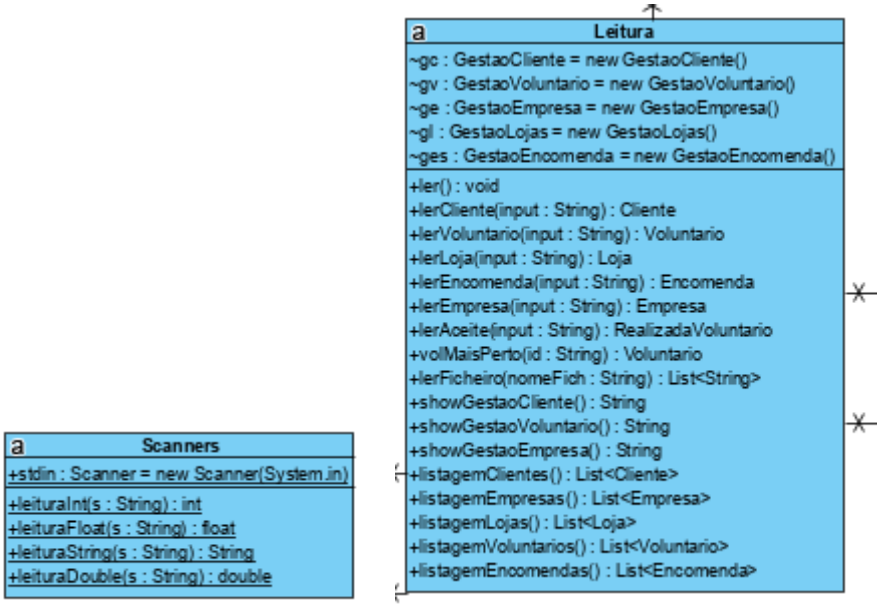

Figura 17 Ferramentas de Leitura e scan

2.15- TrazAqui!

A classe “TrazAqui!” contém o método main, é responsável por fazer correr toda a aplicação de modo simples, eficaz e interativo para que possa ser utilizado pelos utilizadores.

Figura 18 Main

2.16 Decisões

Depois de observadas as classes principais do projeto, podemos então salientar as principais decisões tomadas no projeto. O grupo decidiu utilizar as vantagens que a herança traz e por isso optamos por ter utilizador e encomenda como as superclasses dos conjuntos mais importantes do projeto e a partir disso tirar proveito das funcionalidades que esta característica da programação por objetos possui.

Ao longo do projeto, utilizamos as hashMaps com muita frequência, pois esta coleção é bastante vantajosa para guardarmos toda a informação de forma organizada e de fácil manuseamento tirando proveito dos métodos já implementados.

Para além disto, optamos por dividir o projeto em 2 componentes principais, a parte do model referente a todas as classes e métodos que lidam com a informação e manuseamento do sistema e na classe TrazAqui o controller e a view .

3-Ilustração das Funcionalidades da Aplicação

Quando abrimos a aplicação deparamo-nos com o seguinte menu:

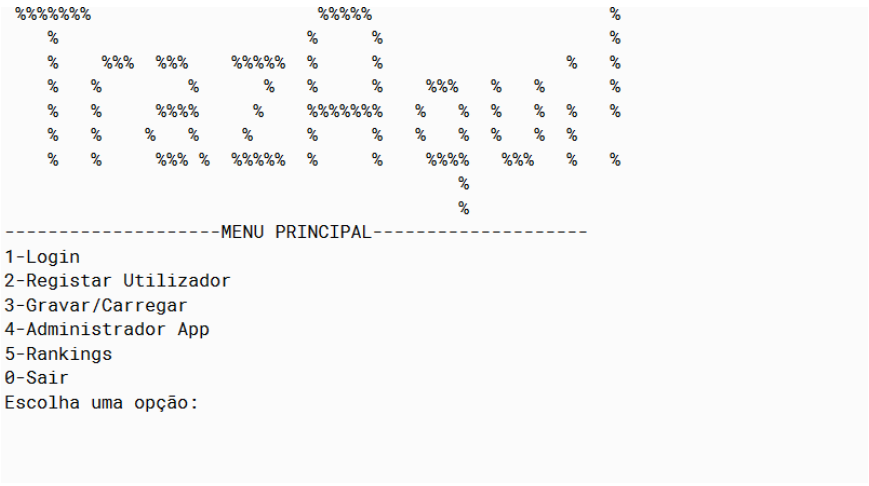

Figura 19 Front end

Caso o utilizador selecione a opção 1 é levado ao menu de login:

```
Escolha uma opção:
1
--------------------MENU LOGIN--------------------
1-Login Cliente
2-Login Loja
3-Login Empresa
4-Login Voluntario
0-Sair
Escolha uma opção
```

Figura 20 Menu Login

De seguida o utilizador poderá fazer o seu respetivo login, tendo em conta a entidade que representa. No exemplo do cliente, se fosse efetuado um login com dados válidos o utilizador seria remetido para o menu Cliente:

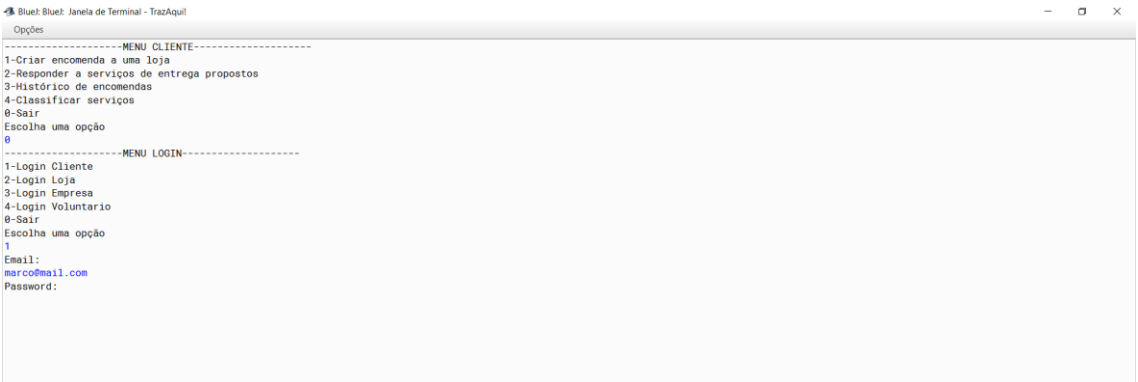

Figura 21 Login

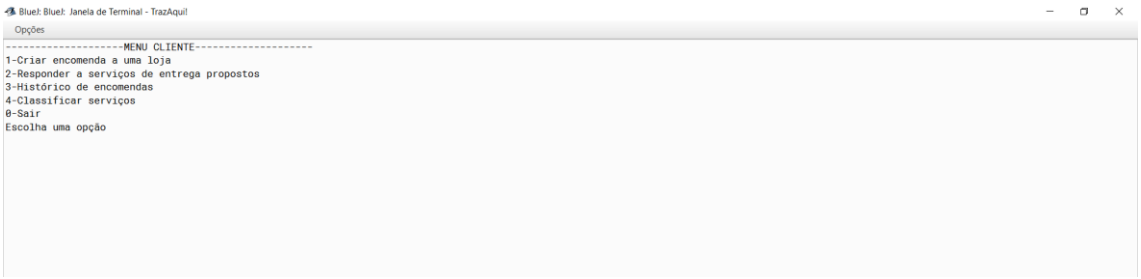

Figura 22 Ferramentas de Encomenda

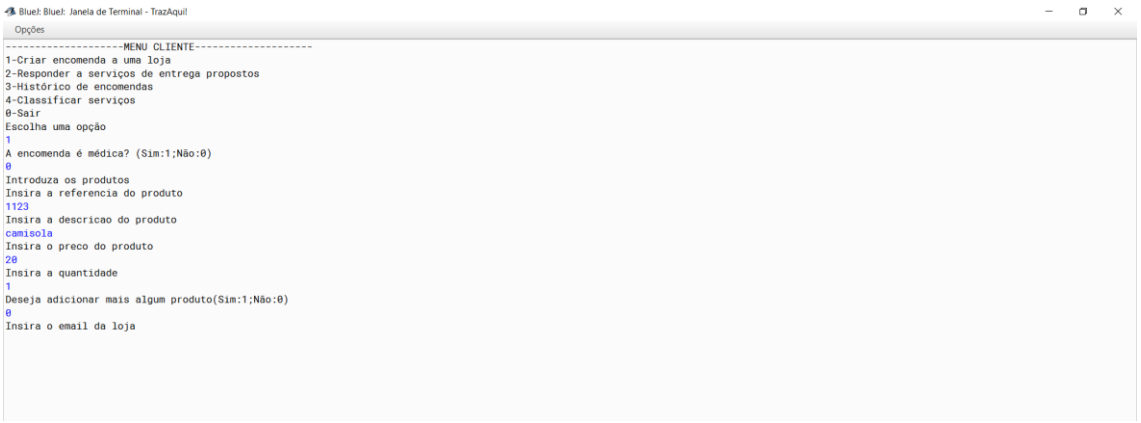

Figura 23 Ferramenta de Encomenda

São apresentadas então todas as opções disponíveis ao cliente, caso o cliente queira realizar uma encomenda vai “dialogar” com a app:

Posteriormente ao login por parte de uma loja é apresentado o seguinte menu:

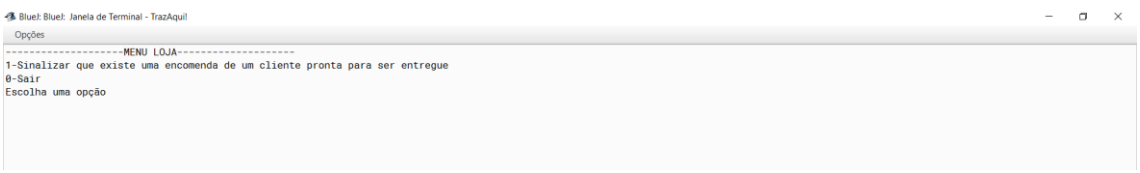

Figura 24 Menu a partir de uma loja

Depois de selecionada a opção 1 são apresentadas as encomendas pendentes a uma resposta:

Figura 25 Encomendas pendentes

No caso das empresas, quando efetuado o login são apresentadas as seguintes opções:

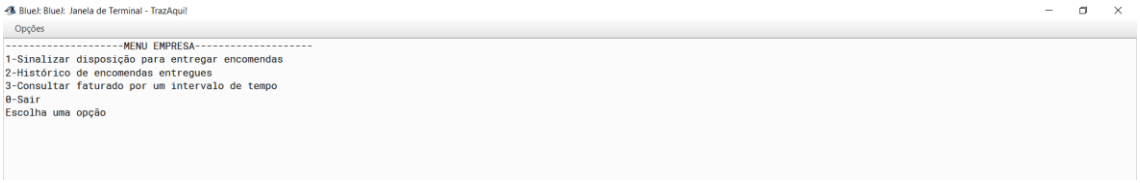

Figura 26 Opções Seleção Empresa

Depois de sinalizada a disposição para entregar uma encomenda o cliente tem de decidir se aceita ou não, voltando então ao cliente e selecionando a opção 2:

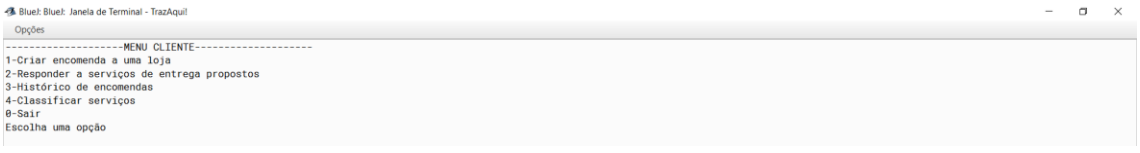

Figura 27 Menu Criação de Encomendas

4-Conclusão

Para concluir é de salientar o desafio que é este projeto mas também a importância que tem para a compreensão daquilo que foi lecionado ao longo do semestre na cadeira de Programação Orientada aos Objetos.

Com este projeto evoluímos não só como programadores mas também a nível de raciocínio e ganhando também experiência na linguagem de programação Java.

Durante a realização do projeto fomo-nos apercebendo de que algumas das soluções que tínhamos encontrado não eram as mais corretas e fomos assim alterando código e adptando-nos sempre à situação. No entanto nem tudo ficou perfeito, também pela sobrecarga de trabalho que fomos tendo ao longo deste semestre. Podíamos ter simplificado nas classes de gestão dos utlizadores e na forma em como abordamos as encomendas. Fomo-nos mantendo um pouco também à frente daquilo que era lecionado nas aulas práticas com medo de que o tempo posteriormente ficasse escasso e isso levou-nos a complicar algumas partes do projeto.

Em suma, podemos dizer que a nossa prestação foi satisfatória, tendo alcançado a maior parte dos objetivos propostos e implementadas as funcionalidades solicitadas.