Escola Superior
de Tecnologia
e Gestão
Politécnico de Coimbra

Relatório de Projecto

Programação Aplicada | Programação IV

Avaliação [Periódica]

Autor(es):
Jorge Miguel dos Santos Martins

Data: Abril de 2021

Resumo

Este é um relatório de descrição do tipo de soluções efetuadas pelo estudante em relação à construção do seu projeto. Contrariamente a relatórios passados não serão demonstrados todos os métodos, mas sim serão demonstrados os pontos chave do programa indicando a generalidade dos métodos usados, dando como exemplo dos restantes, pois estes são similares.

Este relatório serve como base de compreensão do projeto idealizado e construído pelo estudante, assim guiando quem o consulte para uma perceção ágil das decisões tomadas e estruturas do trabalho.

Palavras-chave

Programação Aplicada, Projeto 1, Projeto Programação Aplicada, Bases de Dados Relacionais.

Índice

Lista de Figuras

Lista de Tabelas

Não foi encontrada nenhuma entrada do índice de ilustrações.

Lista de Acrónimos

ER	Modelo Entidade-Relacionamento
SoA	State Of the Art
PA	Programação Aplicada
BD	Base de Dados
ID	Identificador/Identidade

1. Introdução

Foi designado o desenvolvimento de um software que tenha a responsabilidade de adicionar utilizadores ao qual o acesso é através de um processo de validação de credenciais. Este software tem como principal objetivo o manuseamento de dados de utilizadores (gestores, clientes, funcionários) e o manuseamento de dados relativos a produtos. Por conseguinte as encomendas é o principal processo para o qual o programa está organizado, sendo o seu objetivo principal através dos seus dados base gerir encomendas e todos os seus estados envolvidos.

Este projeto diferencia-se pela nova temática de acesso e manipulação de dados através de base de dados relacionais.

No presente relatório a abordagem será demonstrar as classes dos objetos manipulados pelo programa, seguidamente serão demonstradas as opções utilizadas para uso de métodos estáticos de interação com o utilizador para a manipulação desses objetos e por fim demonstrar os métodos usados para a ligação com a base de dados.

Alargando a descrição do projeto não será apenas descrito o código e suas explicações, mas também esquemas de relação de classes, estruturas da base de dados, “script” de criação de base de dados.

2. Objetivos e Metodologias

2.1. Ferramentas e Tecnologias

Para este projeto foi utilizado o software PowerDesigner para o desenho da Base de Dados do software a desenvolver. Para a construção da base de dados relacional foi utilizado o programa HeidSql, que foi "host" da base de dados mysql do software.

Relativamente ao desenvolvimento de código Java para o projeto foi utilizado o software Eclipse.

2.2. Planeamento

O planeamento do projeto foi realizado de forma linear, primeiramente aquando o projeto ficou disponível foi visualizado o enunciado e através dos requisitos foi idealizada uma base de dados relacional que de uma forma geral pudesse suportar os dados oriundos do programa escrito em java com esses requisitos. Após a gerar uma base de dados útil e funcional foi escrito o programa em java dividido em fases, primeiramente todo e qualquer requisito a utilizadores (criar, listar, validar), em seguida produtos e por fim encomendas. Os movimentos dos utilizadores (Logs) tal como verificação do tempo de execução foram deixados para o final do programa.

3. Trabalho Desenvolvido

3.1. Requisitos Implementados

Tabela Requisitos							
Requisito	Estado	Requisito	Estado	Requisito	Estado	Requisito	Estado
R1	Completo	R21	Completo	R41	Completo	R61	Completo
R2	Completo	R22	Completo	R42	Completo	R62	Completo
R3	Completo	R23	Completo	R43	Completo	R63	Completo
R4	Completo	R24	Completo	R44	Completo	R64	Completo
R5	Completo	R25	Completo	R45	Completo	R65	Completo
R6	Completo	R26	Completo	R46	Completo	R66	Completo
R7	Completo	R27	Completo	R47	Completo	R67	Completo
R8	Completo	R28	Completo	R48	Completo	R68	Completo
R9	Completo	R29	Completo	R49	Completo	R69	Completo
R10	Completo	R30	Completo	R50	Completo	R70	Completo
R11	Completo	R31	Completo	R51	Completo	R71	Completo
R12	Completo	R32	Completo	R52	Completo	R72	Completo
R13	Completo	R33	Completo	R53	Completo	R73	Completo
R14	Completo	R34	Completo	R54	Completo	R74	Completo
R15	Completo	R35	Completo	R55	Completo	R75	Completo
R16	Completo	R36	Completo	R56	Completo	R76	Completo
R17	Completo	R37	Completo	R57	Completo	R77	Completo
R18	Completo	R38	Completo	R58	Completo	R78	Completo
R19	Completo	R39	Completo	R59	Completo	R79	Completo
R20	Completo	R40	Completo	R60	Completo	R80	Completo

3.2. Classes e *Packages*

Para uma melhor organização e legibilidade do meu código eu o dividi por packages agrupando as classes aos seus objetivos.

Neste ponto de um modo geral são demonstrados os packages e as suas classes ao qual serão usados diagramas de classe. Dividi por package cada representação de diagrama para que seja mais compreensível a organização do código e assim de uma forma ágil se compreender a estrutura do projeto.

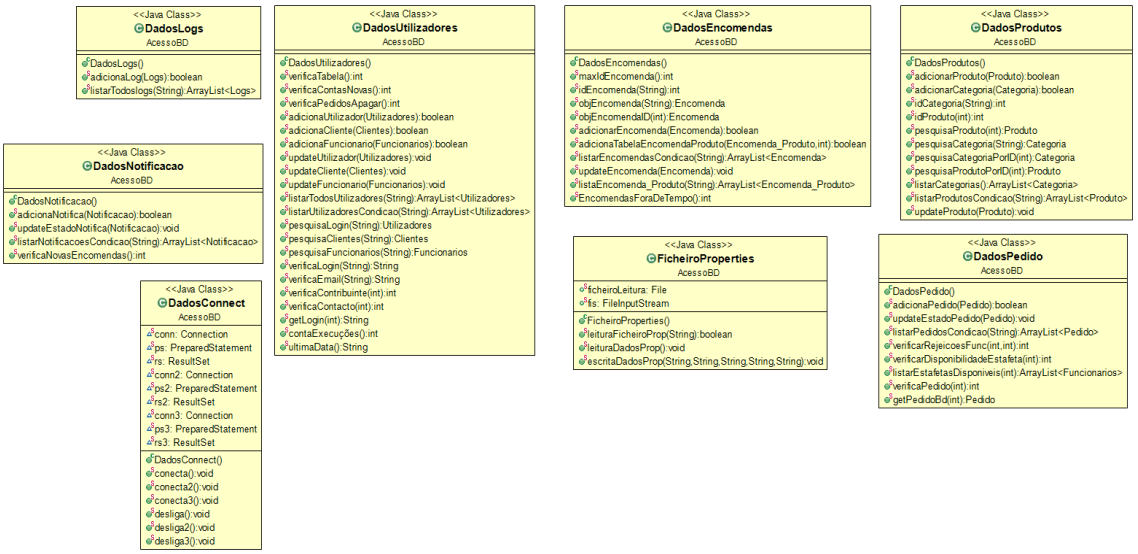

Figura 1 - Diagrama Classes da Package AcessoBD

Esta package contém os métodos responsáveis de acesso à base de dados relacional e o método de leitura e escrita do ficheiro “properties”.

Para uma organização cada classe está através do seu nome a direccionar o seu objetivo na ligação à base de dados relacional.

A classe “DadosConnect” é uma classe criada com variáveis estáticas e métodos estáticos de ligação à base de dados para que se reduza um pouco em código as ligações do software com a base de dados. Já explicando um pouco as variáveis como do tipo “preparedStatement” ou “resultset” são assim reutilizadas várias vezes. Será descrito esta decisão mais à frente no corrente relatório.

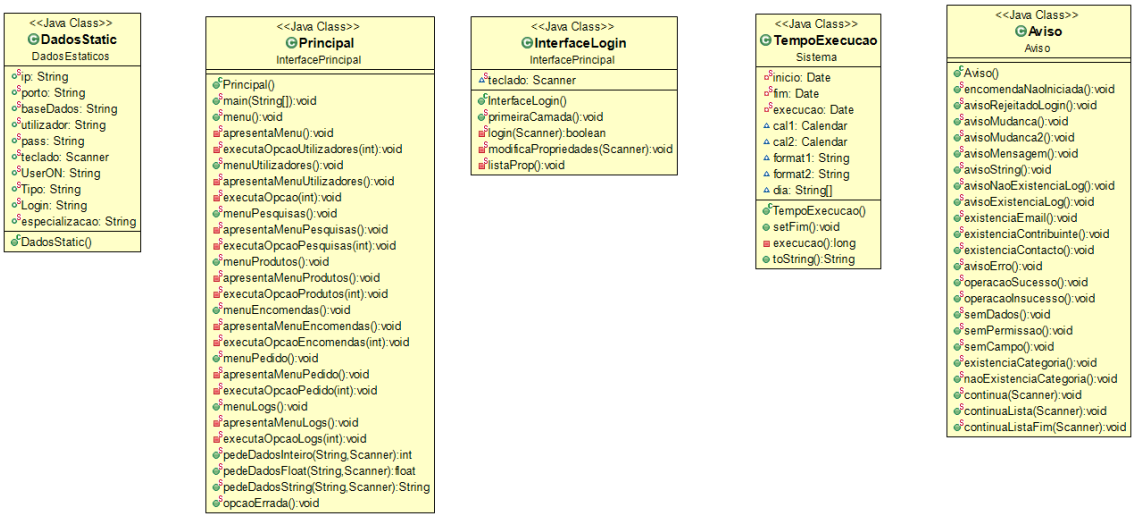

Figura 2 - Diagrama Package Aviso, Interface, Sistema e Dados Estáticos

Devido a estas packages serem mais pequenas em relação a classes, para a sua representação de diagrama de classes foi decidido que fossem em conjunto.

Estas foram separadas por packages para que não só o programador criador do código consiga com facilidade ter uma legibilidade do seu próprio código, mas também para

um elemento exterior ao projeto não encontre desconforto na perceção do que foi programado.

A classe "DadosStatic" é uma classe que contem algumas variáveis estáticas que serão usadas para informações importantes no decorrer do software, tais como as informações de ligação e pedidos à base de dados (exemplo Ip, password, user, base de dados), mas também algumas variáveis que indicam informações acerca do utilizador que se encontra online a utilizar o software.

A classe Principal como o nome indica é onde se encontra o método "main" do software e por conseguinte nesta classe é onde se encontram os métodos que formam os menus do software.

A classe "InterfaceLogin" é considerada a primeira camada do software, aqui é onde é realizado o login por parte do utilizador como também um pedido de registo que se inicia como inactivo no software (apenas o gestor pode mudar isso). Nesta primeira camada o utilizador pode também modificar os dados referentes à ligação da base de dados, cada dado que este modifique o mesmo é gravado no ficheiro Properties que é o responsável por guardar esta informação.

A classe "tempoExecução" é uma classe especialmente criada para o sistema identificar a hora de início e a hora do fim ao encerrar o processo este possa informar o utilizador do tempo decorrido de utilização.

A classe Aviso foi criada com a intenção de conter o mais variados avisos que possam existir no software, assim avisos que se repetem varias vezes são apenas chamados por métodos, deixando o código mais limpo e legível, não tendo o aspeto desorganizado de vários "System.out". Em algumas situações a classe não é usada devido à especificidade do aviso. Esta classe é uma tentativa por parte do estudante para começar a ter o seu código limpo e legível.

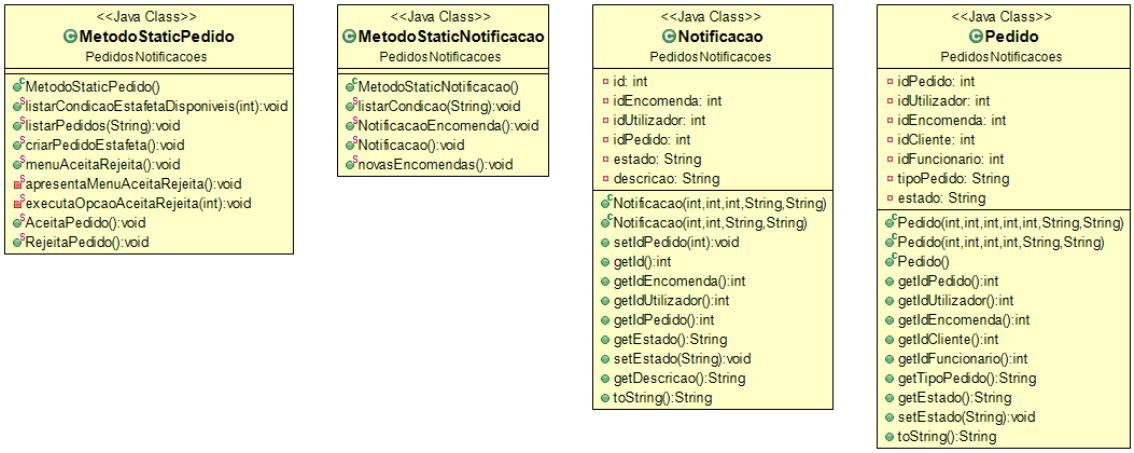

Figura 3 - Package Pedido e Notificações

Na figura 3 pode-se observar a package de Pedidos e Notificações do software, esta como o nome indica contem as classes para criação de objetos relativos às notificações e pedidos tal como os seus métodos estáticos de interação com o utilizador ou que estão mais próximos a essa camada.

A classe Pedido foi desenhada e criada para que pudesse ser usada de forma abrangente tendo em si a possibilidade de conectar o ID de um cliente, utilizador, funcionário. Desta forma esta classe futuramente pode ser usada para transformar o código para que o software se comporte de forma mais organizada, pois algumas soluções que foram criadas não acabaram por usar esta classe (ex. Pedido de remoção de conta apenas muda o estado), a classe mesmo não usando todas as suas possibilidades foi deixada a sua estrutura original com o pensamento numa otimização futura ou decisão de mudança de funcionamento. Inicialmente o pensamento também seria que numa fase em que fosse necessário filtrar informação acerca de pedidos realizados por cliente a gestor ou funcionário a gestor, através desta classe poderia ser fornecida essa informação.

De se observar nesta classe que existem dois construtores diferentes, isto é devido ao momento em que a mesma é criada pelo software não terá uma iniciação com a variável ID de pedido, pois este número será dado pela base de dados, então o segundo construtor é para a criação de um objeto desta classe quando se pede a informação à base de dados, ficando assim o objeto com o ID que a Base de Dados tem acerca dele.

O mesmo sucede com a classe notificação, esta classe foi desenhada para se puder usar numa forma geral para notificar tudo no software, mas em decisão do programador, as notificações gerais (ex. notificações para todos os gestores de novas contas) não são fornecidas através de uma notificação da tabela de base de dados mas sim através de um comando em "sql" quer verifica a contagem dos novos registos de conta. Assim a notificação é usada para o fornecimento de informação direcionada a um utilizador em especifico, sendo que as notificações gerais aparecem sempre que o programa for iniciado (enquanto o utilizador não modificar o estado da ação, como por exemplo aceitar/rejeitar uma nova conta), enquanto que as notificações direcionadas apenas aparecem uma vez a iniciar o programa ficando estas inativas.

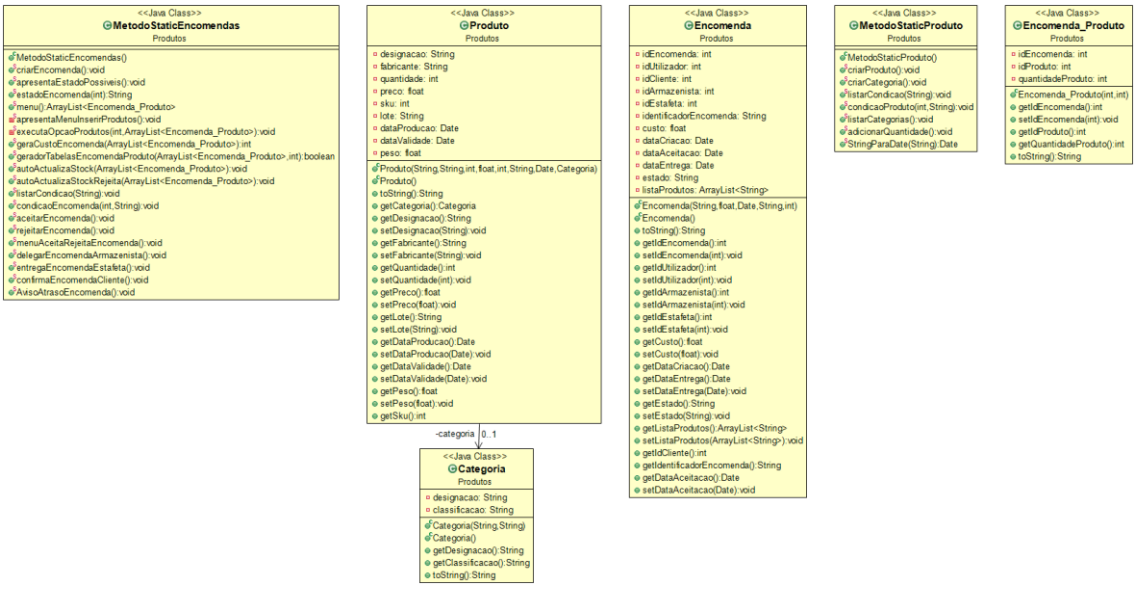

Figura 4 - Package Produtos

A package produtos contem as classes que se envolvem com produtos, por isso as encomendas terem sido também englobadas nesta package assim organizando classes

com propósitos próximos sendo mais prático a um elemento externo na leitura do código se guiar onde poderá encontrar a informação de código que procura.

A classe produto contém algumas variáveis que não são usadas neste momento pelo software, tais como peso e a data de Validade, estavam inseridos nos requisitos como possibilidades, mas não apareciam como chaves de pesquisa dos produtos, relevando não serem dados direcionados aos produtos que o software vai manusear. Entretanto decidiu-se deixar estes dados para um desenvolvimento posterior ao software (Update do mesmo) assim a classe já conter essas informações, o mesmo sucede na sua tabela na base de dados.

Tal como a classe Produto como a classe Encomenda não seguem a mesma logica de construtores diferentes, primeiramente para a classe produto através do sua variável única “SKU” é possível retirar a informação acerca do seu ID da base de dados quando o utilizador precisar, enquanto que na classe Encomenda esta sofre várias alterações no decorrer do seu processo de vida no software, processos que são isolados entre si, criação, aceitação (insere ID de gestor), delegar a funcionário armazenista (insere seu ID) e seguintes processos. Encontrada esta situação em relação a esta classe em vez de criar vários construtores diferentes, decidiu-se que deveria manter a sua iniciação original mas na iniciação do objeto através das informações da base de dados se deveria usar os métodos “set” para que se desse a informação completa ao objeto da classe.

A classe “encomenda_produto” não contém propriamente uma classe de nome método estático focado só em si devido ao facto desta classe ser o elo entre a quantidade de produtos existente em cada encomenda criada.

Esta classe segue a mesma logica que a classe encomenda, pois primeiramente é criada com o ID de produto e a sua quantidade e só após a encomenda é terminada de se criar é que em seguida são geradas as tabelas relativas à informação da quantidade que cada produto constituinte da encomenda tem. Assim o objeto recebe através do método set o ID da encomenda e em seguida é dado à base de dados a informação completa, para iniciar o objeto através da informação da base de dados, primeiramente inicia-se o objeto com o ID de produto e com a quantidade do mesmo e em seguida usa-se o método “set” da classe para que seja dado o ID da encomenda.

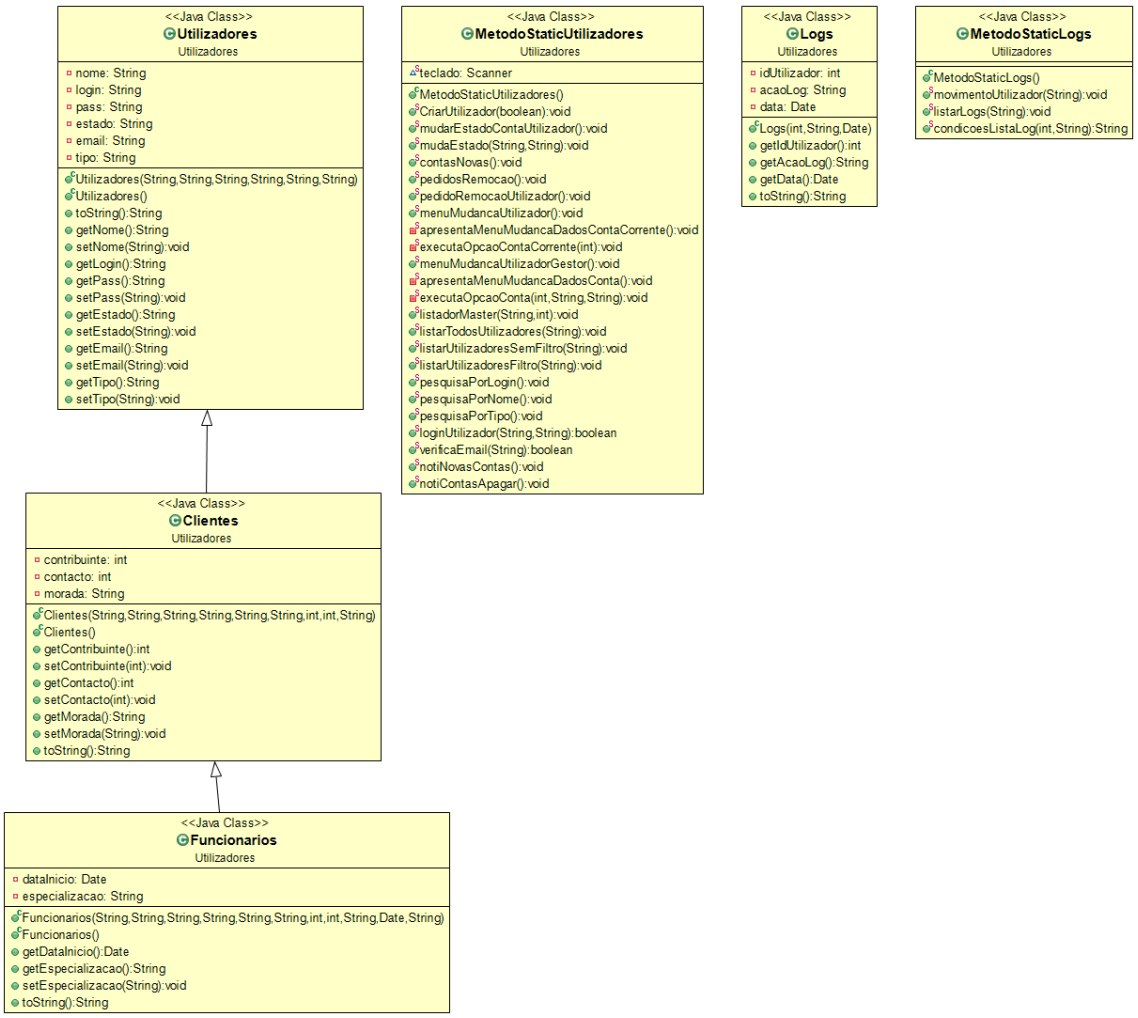

Figura 5 - Package Utilizadores

A package Utilizadores contém as classes responsáveis pelos dados relativos aos utilizadores do software. A classe utilizadora (Pai) é a classe que contém os dados mais comuns e existentes nos três tipos de utilizadores, sendo que a conta de gestor é a que usa o objeto da classe Utilizadores. As suas heranças foram realizadas devido ao facto das outras duas contas conterem algo específico sobre si, diferenciando-se em método de escada, assim Clientes herda de Utilizadores e Funcionários herda de Clientes.

A classe que contem os métodos estáticos de utilizadores tal como já visto anteriormente é a responsável pela interação com o utilizador e também aquela que chamada os métodos (pedidos) à base de dados para que esta forneça a informação desejada pelo utilizador.

A classe “Logs” foi inserida neste package pois esta é responsável por guardar os movimentos do utilizador online na base de dados, identificando onde nos menus o utilizador escolheu as suas opções (ex. entrou no menu Utilizadores), assim visto ser uma classe com direção aos atos do utilizador decidiu-se que seria logico a alocar na package referente aos Utilizadores.

Breve explicação do funcionamento do software

Após a demonstração geral do diagrama de classes do software, será necessário demonstrar alguns exemplos acerca do código em si, elucidando o leitor a compreender melhor o código desenvolvido pelo programador. Assim é de interesse informar que não serão demonstrados todos os métodos realizados, pois existem muitos que são similares no seu funcionamento, são demonstrados os métodos estáticos de interação com o utilizador (criação, listagem, verificação) e os métodos utilizados para ligação na base de dados (insert, update, select).

O software funciona de maneira similar entre os seus métodos, o que difere são as direções para onde os métodos estão a ser usados, por exemplo o mesmo método de listagem é reutilizado para a pesquisa pois a forma de pedido à base de dados apenas difere na condição após o "from" na "query" de "sql" mas os objetos iniciados com as informações do "result set " são os mesmos.

3.3. Algoritmos

Como já explicado anteriormente os métodos do programa são muito similares entre si, ao qual neste tema demonstrarei os métodos que interligam o resultado proveniente do método que faz o pedido da base de dados e também aqueles que interagem com o utilizador.

```
public static void CriarUtilizador(boolean primeiroLogin) {
    boolean state;
    String nome;
    String login;
    String password;
    String email;
    String tipo = "Gestor";
    String estado = "espera";
    Integer contacto = 0;
    String especializacao = new String();
    Date dataInicio = new Date();
    String morada = new String();
    Integer contribuinte = 0;
    do{
        login = Principal.pedeDadosString("Insira o seu Login", teclado);
        state = login.contains(";");
        if(state)
            Aviso.avisoMensagem();
        if(login.isEmpty() || login.trim().isEmpty() || login.charAt(0) == ' '){
            state = true;
            Aviso.avisoString();
        }
        if(DadosUtilizadores.verificaLogin(login) == null || DadosUtilizadores.verificaLogin(login).length() > 0) {
            Aviso.avisoExistenciaLog();
            state = true;
        }
    }while(state);
    do{
        nome = Principal.pedeDadosString("Insira o seu nome", teclado);
```

Figura 6 - Criar Utilizador

Esta é a forma usada para através da interação com o utilizador adquirir os dados necessários para a criação de um objeto. Neste caso na figura 6 está representado parte do método estático de criação de um Utilizador, na medida que são pedidos dados estes são verificados e validados.

```
if(!tipo.equalsIgnoreCase("Gestor")) { //nao igual a gestor
    do {
        contribuinte = Principal.pedeDadosInteiro("\nInsira o seu contribuinte\n", teclado);
        if(DadosUtilizadores.verificaContribuinte(contribuinte) == 0) {
            String auxContribuinte = Integer.toString(contribuinte);
            if(auxContribuinte.length() < 9 || contribuinte / 100000000 < 0 || contribuinte / 100000000 > 3) { //divisao int
                state = true;
                System.out.println("Erro na Inserção do Contribuinte, tem de ter 9 digitos e começa por 1, 2 ou 3");
            }else {
                state = false;
            }

        }else {
            state = true;
            Aviso.existenciaContribuinte();
        }
    }while(state);
```

Figura 7 - Método Criar Utilizador Verificar na Base de Dados

Além de se verificar através do código da validade dos dados, também este método verifica para os dados que tendem a ser únicos, se os mesmo já se encontram na base de dados. Para este caso é realizado um pedido à base de dados por parte do método "DadosUtilizadores.verificaContribuinte" que realiza um pedido à base de dados com uma query que devolve informação de um inteiro através da pesquisa da tabela clientes com a condição de este ter este contribuinte.

Se nenhum utilizador existir com este contribuinte o dado devolvido será 0, pois a tabela com apenas uma coluna estará vazia, assim demonstrando que não existe utilizador (cliente ou funcionário) com este contribuinte.

O mesmo tipo de logica é utilizada na inserção e validação de dados em outros campos do software, tal como a inserção de um produto ou mesmo na inserção de uma encomenda. No caso de uma encomenda é verificado se o armazenista existe na delegação por parte do gestor, ou se o identificador do produto dado pelo cliente está correto. Todos estes processos de inserção de dados são similares.

```
public static void mudarEstadoContaUtilizador(){
    String login = Principal.pedeDadosString("Insira o login do utilizador a modificar o seu estado", Dados
    if(login.equals(DadosStatic.UserON)){
        System.out.println("Não pode modificar o seu próprio Estado por Segurança do Sistema!");
    }else{
        Utilizadores utilizador = DadosUtilizadores.pesquisaLogin(login);
        if(utilizador.getLogin() != null) {
            if(utilizador.getLogin().equals(login)){
                int opcao = Principal.pedeDadosInteiro("Insira para que estado quer mudar \n1-Activo \n2-In
                switch(opcao){
                    case 1: mudaEstado(login, "activo"); Aviso.operacaoSucesso(); break;
                    case 2: mudaEstado(login, "inactivo"); Aviso.operacaoSucesso(); break;
                    default: Aviso.operacaoInsucesso(); break;

                }
            }else{
                Aviso.avisoNaoExistenciaLog();
            }
        }else{
            Aviso.avisoNaoExistenciaLog();
        }

    }
}
```

Figura 8 - Método Mudança de estado

Reforçando um pouco o que já foi demonstrado em relação aos métodos que recebem dados por parte dos utilizadores, verifica-se na figura 8 o método de mudança de estado de uma conta por parte de um gestor.

Primeiramente o programa pede ao utilizador que insira os dados de login do utilizador que vai ser modificado o seu estado e em seguida após verificar não é o próprio utilizador, é usado o método de pesquisa pelo login, que realiza um pedido à base de dados para que esta devolva um "result set" com os dados para se iniciar um objeto da classe utilizadores. Deste modo após o objeto igualar ao objeto recebido pelo método que realiza o pedido o mesmo é verificado, primeiramente se não se encontra nulo, ou seja se este não existe na base dados e por segurança dobrada uma verificação se o login coincide com o do objeto.

Em seguida a opção escolhida de mudança do estado envia a "string" com o login e o estado pelo qual ele vai ficar, o que significado que estas duas informações vão ser adicionadas numa query que as irá usar para modificar na tabela de utilizadores, um utilizador com o login igual terá um "update" com esse novo estado.

```
public static void mudaEstado(String aLogin, String aEstado) {
    Utilizadores utilizador = DadosUtilizadores.pesquisaLogin(aLogin);
    utilizador.setEstado(aEstado);
    DadosUtilizadores.updateUtilizador(utilizador);
}
```

Figura 9 - Mudança de estado parte 2

Como referenciado o segundo método de suporte para o "update" ainda é um método estático que irá com as informações pedidas iniciar um objeto com informações oriundas da base de dados e por conseguinte através do método "set" modificar o seu estado. Finalmente realiza um "update" a todo o objeto, pois como será demonstrado mais à frente o método "update" realiza alterações a todas as colunas da tabela, assim este podendo ser utilizado das mais variadas maneiras, poupando métodos desnecessários.

```
public static void listarTodosUtilizadores(String aOrdem) {

    ArrayList <Utilizadores> utilizadores = DadosUtilizadores.listarTodosUtilizadores(aOrdem);

    if(utilizadores != null){
        Iterator <Utilizadores> tabela = utilizadores.iterator();
        String gestor = "";
        String cliente = "";
        String funcionario = "";
        Utilizadores auxiliar;
        int contador = 0;
        while(tabela.hasNext()) {
           auxiliar = tabela.next();
           contador++;
           if(auxiliar.getTipo().equals("Gestor")){
                gestor += "Login: " + auxiliar.getLogin() + " Nome: " + auxiliar.getNome() + " \n";
           }
           if(auxiliar.getTipo().equals("Cliente")){
                cliente += "Login: " + auxiliar.getLogin() + " Nome: " + auxiliar.getNome() + " \n";
           }
           if(auxiliar.getTipo().equals("Funcionario")){
                funcionario += "Login: " + auxiliar.getLogin() + " Nome: " + auxiliar.getNome() + " \n";
           }
           if(contador == 10) {
               listadorMaster(" Gestores \n" + gestor + " \n" + " Clientes \n" + cliente + " \n" + " Funcionarios \n" + funcio
               contador = 0;
               gestor = "";
               cliente = "";
               funcionario = "";
           }

        }
        if(contador > 0) {
            listadorMaster(" Gestores \n" + gestor + " \n" + " Clientes \n" + cliente + " \n" + " Funcionarios \n" + funcionar
        }
```

Figura 10 - Listagem Utilizadores Filtro

```
public static void listarUtilizadoresSemFiltro(String aCondicao) {
    ArrayList <Utilizadores> utilizadores = DadosUtilizadores.listarUtilizadoresCondicao(aCondicao);

    if(utilizadores != null){
        Iterator <Utilizadores> tabela = utilizadores.iterator();
        String envio = "";
        Utilizadores auxiliar;
        int contador = 0;
           while(tabela.hasNext()) {
               auxiliar = tabela.next();
               envio += "Login: " + auxiliar.getLogin() + " Nome: " + auxiliar.getNome() + "Tipo: " + auxiliar.getTipo() + "\n
               contador++;

               if(contador == 10) {
                   listadorMaster(" Utilizadores \n" + envio + "\n", contador);
                   contador = 0;
                   envio = "";
               }
           }
           if(contador > 0) {
               listadorMaster(" Utilizadores \n" + envio + "\n", contador);
           }

    }
}
```

Figura 11 - Listagem Utilizadores Sem Filtro

A figura 10 e 11 demonstram os métodos de listagem usados para que o ArrayList recebido do método que realiza o pedido à base de dados por informações seja listado.

Como requisito apenas em todas as listagens são apenas demonstradas páginas com 10 pesquisas/listagens, sendo que se deve então dividir a listagem. O método com filtro por tipo apenas é utilizado para os utilizadores que estes é que contém três tipos de conta já conhecidos de antemão, no resto todos as classes (produtos, encomendas, pedidos, notificações) usam um método similar ao demonstrado na figura 11, com as diferenças de que é especifico para a sua classe mudando o tipo de “ArrayList” que vai receber tal como os objetos a manipular.

Antes de enviar para o geral “listadorMaster” que apenas lista a “String” que recebe, é realizada uma contagem pelo número de vezes que informação é retirada do “Array” para que sejam listados de 10 em 10, assim condicionado uma paginação de 10 em 10.

Este método também é utilizado para pesquisas avançadas pois estas comportam-se da mesma forma como uma listagem por tipo, apenas tem uma condicionante mas por norma recebem imensos dados, assim reutilizei o mesmo método em vez de realizar métodos diferentes.

```
public static void pesquisaPorNome() {
    String nome = Principal.pedeDadosString("Insira o nome a pesquisar", DadosStatic.teclado);
    listarUtilizadoresFiltro("NOME_UTILIZADOR LIKE '%"+nome+"%'");
    Aviso.continua(DadosStatic.teclado);
}
```

Figura 12 - Pesquisa Avançada Exemplo

Para este caso até é utilizada a listagem por filtro para identificar os utilizadores pelo seu tipo, pesquisando de forma avançada com a condicionante do seu nome ser algo como aquilo que o utilizador forneceu para pesquisa.

```
public static void pesquisaPorLogin() {
    String login = Principal.pedeDadosString("Insira o login a pesquisar", DadosStatic.teclado);
    Utilizadores utilizador = DadosUtilizadores.pesquisaLogin(login);
    if(utilizador.getLogin() != null) {
        if(utilizador.getLogin().equalsIgnoreCase(login)) {
            System.out.println(utilizador);
        }else
            Aviso.semDados();
    }else {
        Aviso.semDados();
    }


    Aviso.continua(DadosStatic.teclado);
}
```

Figura 13 - Caso específico Pesquisa

Para um caso específico de pesquisa pelo qual é de conhecimento que só um dado será fornecido, como por exemplo através do login que é único, então para este caso existe um método mais específico sem o uso da listagem.

Este método de pesquisa por login não é apenas utilizado para pesquisa de fornecimento de dados ao utilizador, mas também para funcionamento interno do software, tal como validação de dados e uso para iniciar um objeto específico de um utilizador para manusear os seus dados (“update” dos seus dados).

```
public static void listadorMaster(String lista, int tamanho) {
    if(lista != null || tamanho > 0)
        System.out.println(lista);
    else
        Aviso.semDados();
    if(tamanho < 10) {
        Aviso.continuaListaFim(teclado);
    }else {
        Aviso.continuaLista(teclado);
    }

}
```

Figura 14 - Método Listador Master

Este método é reutilizado mesmo pelas outras classes, sendo chamado dentro das próprias classes de métodos estáticos de produtos e encomendas para listagem, devido ao facto de não ser necessário alterar nada na sua utilidade, foi logica a escolha da sua reutilização em vez de escrever código redundante.

3.4. Estruturas de Dados

Como já identificado no tema anterior a estrutura mais utilizada para a manipulação de dados é o “ArrayList”, utilizado para a listagem de dados oriundos da base dados, primeiramente são iniciados os objetos e em seguida inseridos no “ArrayList” a ser enviado para o seu método de listagem.

3.5. Armazenamento de Dados

3.5.1. Métodos Java de Acesso à Base de Dados

Os métodos responsáveis pelo armazenamento dos dados como já referenciados encontram-se no package de acesso á base de dados "AcessoBD". Os métodos de acesso á base de dados são entre si similares, pelo qual para cada classe existe o seu método genérico mudando as condições de seleção de dados para o pedido.

O único caso do software pelo qual foi usada uma inserção de dados com a opção "autoCommit" da base dados em manual, foi nos dados relativos a clientes e funcionários, pois tratam-se de heranças e como tal na inserção dos dados na base de dados é necessária a existência de uma segurança se caso alguma exceção ocorra no software, fazendo assim um "rollback" dos dados. Referenciar que na criação dos dados só após obter todos os dados necessários por parte do utilizador e os validar é que os mesmos são pedidos para ser inseridos na base de dados, reduzindo as chances de erros assegurando um bom funcionamento do software.

```
17  */
18 public class DadosConnect {
19
20     static Connection conn = null;
21     static PreparedStatement ps = null;
22     static ResultSet rs = null;
23
```

Figura 15 - Dados Estáticos de Conexão

Para um código mais limpo e com métodos menos extensos, decidi usar variáveis de conexão à base de dados de forma estática, assim esta classe além de conter estar variáveis, também contém os métodos de as ligar e de as fechar.

De referenciar o facto de dentro desta classe existirem três grupos de variáveis estáticas e seus métodos, isto devido ao facto que em alguns pedidos à base de dados, alguns métodos em si mesmos realizam pedidos de validação e assim protege-se de nenhuma das variáveis ser fechada fora do tempo programado de uso.

```
*/
public static void conecta() {
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
        conn = DriverManager.getConnection("jdbc:mysql://"+DadosStatic.ip+DadosStatic.porto+"/"+DadosStatic.baseDados+"?useSSL=
    }catch (SQLException e) {
        System.out.println("!! SQL Exception !!\n"+e);
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
            System.out.println("!! Class Not Found. Unable to load Database Drive !!\n"+e);
    }

}
```

Figura 16 - Método Conecta

Todo o método de acesso à base de dados inicia com o chamar deste método (ou seus similares), este é responsável de iniciar a variável da classe "connection" à base de dados relacional que o software utiliza e aqui é verificado o caminho da mesma. Nesta situação

são usadas as variáveis estáticas que retiram os dados oriundos do ficheiro “properties” para que seja realizada com eficácia a ligação à base de dados.

```java
/**
 * Metodo 1 que desliga a ligação à base de dados de forma segura
 */
public static void desliga() {
    try {
        if(conn != null)
            conn.close();
        if(ps != null)
            ps.close();
        if(rs != null)
            rs.close();
    }catch (SQLException e) {
        System.out.println("!! SQL Exception !!\n"+e);
        e.printStackTrace();
    }
}
```

Figura 17 - Método desliga

Todo o método de acesso à base de dados também contém o chamamento ao método de desligar ou fechar as variáveis de acesso, mas que assim não existam erros posteriores no decorrer do funcionamento do programa e seus pedidos à base de dados.

```java
public static boolean adicionarEncomenda(Encomenda encomenda) {
    DadosConnect.conecta();
    try {

        StringBuffer sqlQuery = new StringBuffer();

        // prepared statement for select
        sqlQuery.append(" INSERT INTO encomenda  (CLI_ID_UTILIZADOR, IDENTIFICADOR_ENCOMENDA, CUSTO_ENCOMENDA, DATACRIACAO_ENC
                + " VALUES(?, ?, ?, ?, ?); ");
        java.sql.Date sqlDate = new java.sql.Date(encomenda.getDataCriacao().getTime());

        DadosConnect.ps = DadosConnect.conn.prepareStatement(sqlQuery.toString());
        DadosConnect.ps.clearParameters();
        DadosConnect.ps.setInt(1, encomenda.getIdCliente());
        DadosConnect.ps.setString(2, encomenda.getIdentificadorEncomenda());
        DadosConnect.ps.setFloat(3, encomenda.getCusto());
        DadosConnect.ps.setDate(4, sqlDate);
        DadosConnect.ps.setString(5, encomenda.getEstado());


        DadosConnect.ps.executeUpdate();



    } catch (SQLException e) {
        System.out.println("!! SQL Exception !!\n"+e);
        e.printStackTrace();
        return false;
    }

    DadosConnect.desliga();
    return true;
}
```

Figura 18 - Método Insert Dados

Aqui está representada a estrutura de um método “insert” utilizado neste software, é usado o “prepared statement” para uma melhor segurança dos dados introduzidos pelo utilizador este método manipula a informação de um objeto que recebe de forma a introduzi-la nos campos corretos relativos à tabela de base de dados.

Cada método de introdução de dados na base de dados relacional segue este modelo aqui representado, diferenciando apenas a tabela que está a ser chamada na “query”

tal como os campos que irão ser introduzidos, também difere o objeto que recebe, sendo o da classe representativa no código dos dados da tabela de Base de Dados, mas em suma toda a arquitetura é a mesma, o mesmo pensamento a mesma logica de resolução.

```
    public static void updateEncomenda(Encomenda encomenda) {
        DadosConnect.conecta();
        try {
              java.sql.Date sqlDate;
              java.sql.Date sqlDate2;

              StringBuffer sqlQuery = new StringBuffer();
              sqlQuery.append(" UPDATE encomenda ");
              sqlQuery.append(" SET ESTADO_ENCOMENDA = ? ,");
              sqlQuery.append(" ID_UTILIZADOR = ? ,");
              sqlQuery.append(" FUN_ID_UTILIZADOR = ? ,");
              sqlQuery.append(" FUN_ID_UTILIZADOR2 = ? ,");
              sqlQuery.append(" DATAACEITACAO_ENCOMENDA = ? ,");
              sqlQuery.append(" DATAENTREGA_ENCOMENDA = ? ");
              sqlQuery.append(" WHERE IDENTIFICADOR_ENCOMENDA = ? ");

              DadosConnect.ps = DadosConnect.conn.prepareStatement(sqlQuery.toString());

              DadosConnect.ps.clearParameters();
              DadosConnect.ps.setString(1, encomenda.getEstado());
              DadosConnect.ps.setInt(2, encomenda.getIdUtilizador());
              if(encomenda.getIdArmazenista() != 0) {
                  DadosConnect.ps.setInt(3, encomenda.getIdArmazenista());
              }else {
                  DadosConnect.ps.setNull(3, Types.INTEGER);
              }
              if(encomenda.getIdEstafeta() != 0) {
                  DadosConnect.ps.setInt(4, encomenda.getIdEstafeta());
              }else {
                  DadosConnect.ps.setNull(4, Types.INTEGER);
```

Figura 19 - Método Update

Pegando no exemplo mais completo de “update” de uma tabela de base de dados, primeiramente comentar que para que seja utilizado nas mais diversas ocasiões para as classes de relativas a utilizadores (e suas heranças), produtos e encomenda o “update” é realizado sempre a todos os dados, mesmo estes se mantendo os mesmos, dando assim uma polivalência ao método de que liga á base de dados e a atualiza.

Existem nos casos das classes de Pedido e Notificação que após estudo da funcionalidade das mesmas, mesmo futuramente não faria sentido modificar dados além do seu estado, pois não foi um requisito pedido mas ser interessante implementar uma classe que apagasse os dados relativos a pedidos e notificações de 10 em 10 dias, mas também poderia ser viável para o utilizador ter esses dados futuramente para percecionar comportamentos dos seus funcionários (visto os pedidos por exemplo apenas serem usados na fase de passagem de encomenda para um estafeta).

Chegando ao ponto inicial acerca de método, começou por se relatar que é um exemplo completo, pois aqui também se encontra uma situação pelo qual os dados desta tabela estão em constante transformação, pois inicialmente a mesma não contem dados acerca do gestor que a aceitou, o armazenista que a preparou, ou estafeta que a entregou, nem as respetivas datas de aceitação e entrega, então para estes casos ao qual é necessário enviar à base de dados um valor nulo, utilizou-se o “setNull” em seguida indica-se o tipo de valor nulo que se dá para qual tipo “Types.tipo”.

```
*/
public static ArrayList <Encomenda> listarEncomendasCondicao(String aCondicao){
    ArrayList <Encomenda> lista = new ArrayList <Encomenda> ();
    DadosConnect.conecta();
    try {

          StringBuffer sqlQuery = new StringBuffer();

          // prepared statement for select
          sqlQuery.append(" SELECT * FROM encomenda ");
          sqlQuery.append(" "+aCondicao+" ; ");
          DadosConnect.ps = DadosConnect.conn.prepareStatement(sqlQuery.toString());
          DadosConnect.ps.clearParameters();

          DadosConnect.rs = DadosConnect.ps.executeQuery();

          if (DadosConnect.rs == null) {
                System.out.println("!! No Record on table !!");
          } else {
              lista = new ArrayList <Encomenda> (); //garantir que a lista fica vazia..
              while (DadosConnect.rs.next()) {
                  lista.add(new Encomenda(DadosConnect.rs.getString("IDENTIFICADOR_ENCOMENDA"), DadosConnect.rs.getFloat('
              }
          }
```

Figura 20 - Lista Dados

Este método é um exemplo dos seus similares na obtenção de vários dados da base de dados numa só opção, sabendo neste caso o método é adaptado para as encomendas então o "ArrayList" será para objetos de "Encomenda" e a primeira parte da "query" já automática será obter todos os dados das colunas da tabela encomenda da base de dados. Então é onde a diferenciação das condições para os diferentes tipos de listagem e pesquisa avançada acontecem é na segunda colagem da "String" que insere a condição vinda de um método estático responsável pela opção.

```
 public static Utilizadores pesquisaLogin(String aLogin) {
     DadosConnect.conecta();
     Utilizadores utilizador = new Utilizadores();
     try {

           StringBuffer sqlQuery = new StringBuffer();

           // prepared statement for select
           sqlQuery.append(" SELECT * FROM utilizadores ");
           sqlQuery.append(" WHERE LOGIN_UTILIZADOR = ? ;");

           DadosConnect.ps = DadosConnect.conn.prepareStatement(sqlQuery.toString());
           DadosConnect.ps.clearParameters();
           DadosConnect.ps.setString(1, aLogin);

           DadosConnect.rs = DadosConnect.ps.executeQuery();

           if (DadosConnect.rs == null) {
                 System.out.println("!! No Record on table !!");
           } else {
               while (DadosConnect.rs.next()) {
                   utilizador = new Utilizadores(DadosConnect.rs.getString("NOME_UTILIZADOR"), DadosConnect.rs.getString("LO
               }
           }
```

Figura 21 - Pesquisa Especifica

Para a obtenção de um dado especifico como se pode observar as mudanças não são radicais apenas é feito algum ajuste, tal como a "query" já está preparada apenas necessita do valor (preparada também para valores inseridos pelo utilizador), assim este pedido devolve apenas um objeto da classe utilizador, mas existe o mesmo método similar em outras classes, para que a manipulação de dados específicos seja feita em objeto pelo software e a base de dados seja apenas responsável por atender a pedidos de atualização e inserção.

```
public static String verificaLogin(String login) {
    DadosConnect.conecta2();
    String envio = "";

    try {

          StringBuffer sqlQuery = new StringBuffer();

          // prepared statement for select
          sqlQuery.append(" SELECT ID_UTILIZADOR FROM utilizadores ");
          sqlQuery.append(" WHERE LOGIN_UTILIZADOR = ? ");

          DadosConnect.ps2 = DadosConnect.conn2.prepareStatement(sqlQuery.toString());
          DadosConnect.ps2.clearParameters();
          DadosConnect.ps2.setString(1, login);

          DadosConnect.rs2 = DadosConnect.ps2.executeQuery();

          if (DadosConnect.rs2 == null) {
                System.out.println("!! No Record on table !!");
          } else
              while (DadosConnect.rs2.next()) {
                  envio = DadosConnect.rs2.getString("ID_UTILIZADOR");
              }

    } catch (SQLException e) {
```

Figura 22 - Verifica existência

Sendo mais pratico verificar a existência através do que a base de dados devolve para um inteiro ou uma “String” em vez do objeto (alguns problemas com nullPointer), estes métodos são utilizados várias vezes para verificar existência de dados, tanto para validações como até contagens. Sendo que por exemplo no caso de inteiros a verificar se o contribuinte existe sabe-se que caso ele exista na base de dados ela devolve o contribuinte caso não exista ao devolver uma tabela vazia este é considerado 0, assim o número 0 representa a sua inexistência na base de dados.

```
public static int EncomendasForaDeTempo() {
    int conta = 0;


    StringBuffer sqlQuery = new StringBuffer();

    DadosConnect.conecta();

    sqlQuery.append(" SELECT COUNT(*) AS \"Contador\" FROM encomenda ");
    sqlQuery.append(" WHERE ID_UTILIZADOR = '"+DadosUtilizadores.verificaLogin(DadosStatic.Login)+"' "
                  + " AND ABS(DATEDIFF(DATAACEITACAO_ENCOMENDA, DATAENTREGA_ENCOMENDA)) > 10 ; ");

    try {
        DadosConnect.ps = DadosConnect.conn.prepareStatement(sqlQuery.toString());
        DadosConnect.ps.clearParameters();
```

Figura 23 - Notificação Geral Gestores

Ao contrario das notificações especificas que contam com uma “query” ao qual a condição identifica na tabela de notificações alguma ainda ativa que esteja com um ID direcionado para a conta que ficou online, aí é demonstrado ao utilizador a notificação e a mesma se torna inativa (mudança do seu estado).

Este exemplo de notificação geral ela tem tendência a persistir também devido à sua importância mas também porque é uma notificação de responsabilidade de uma classe (gestores), pelo qual é utilizado um contador na “query” e verifica se alguma encomenda se encontra nas condições impostas, caso não existam a tabela devolve vazio que como

inteiro é considerado 0, sendo que o seu método estático apenas verifica se o que é devolvido é maior que 0 para ser verdade e notificar a existência de alguma situação com aquelas condições.

Concluindo esta parte do tema, obvio que não existem apenas estes métodos, mas todos os métodos utilizados são bastante similares aos aqui exibidos apenas com alguns ajustes direcionados á sua classe e sua utilidade, penso que seja menos massivo para o leitor a compreensão através deste relatório sobre o funcionamento interno do software e seguidamente ao ler o código com facilidade entender as estruturas usadas.

3.5.2. Diagrama Conceptual

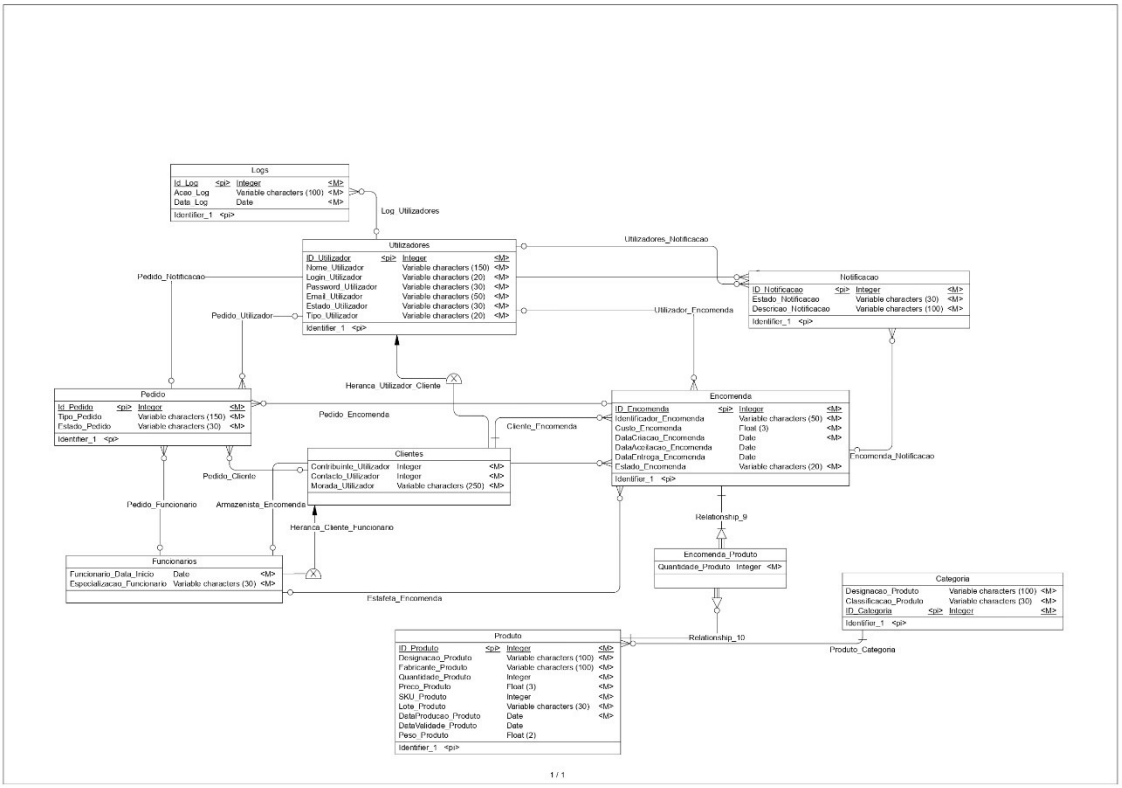

Figura 24 - Diagrama Conceptual

Em anexo é enviado com este relatório o diagrama conceptual do mesmo em formato pdf para uma melhor compreensão do mesmo.

Inicialmente ainda foi pensado em usar em todas as tabelas “n para n” mas devido ao facto depois em código ser mais complexo ter de estar a gerar e manipular várias tabelas, quando logicamente em certos requisitos apenas é necessário uma ligação, foi decidido criar uma boa base para suportar o software e não criar obstáculos ao programar.

A ligação para a tabela encomenda com funcionários poderia ser de “n para n” mas devido ao facto de apenas serem dois funcionários envolvidos criou-se duas ligações distintas, identificando os responsáveis (armazenista e estafeta) e assim apenas foi

necessário a manipulação dos seus ID como chave forasteira, sendo pratico e eficaz no problema encontrado.

Em alguns casos deixei também prioritária a existência de um dado de uma outra tabela (ex Produto com categoria) pois já no código será algo obrigatório então pensei que poderia deixar também na base de dados. Deste modo seguindo apenas a logica que seria aplicada no desenvolvimento do código do software.

Utilizei também que algumas variáveis fossem obrigatórias na inserção de uma linha numa tabela mas optei por não usar na criação a unicidade das mesmas, deixando essa segurança para o código, pois caso tentasse através da base de dados apenas resultaria numa exceção “truncada” e o código iria parar.

3.5.3. Diagrama Físico

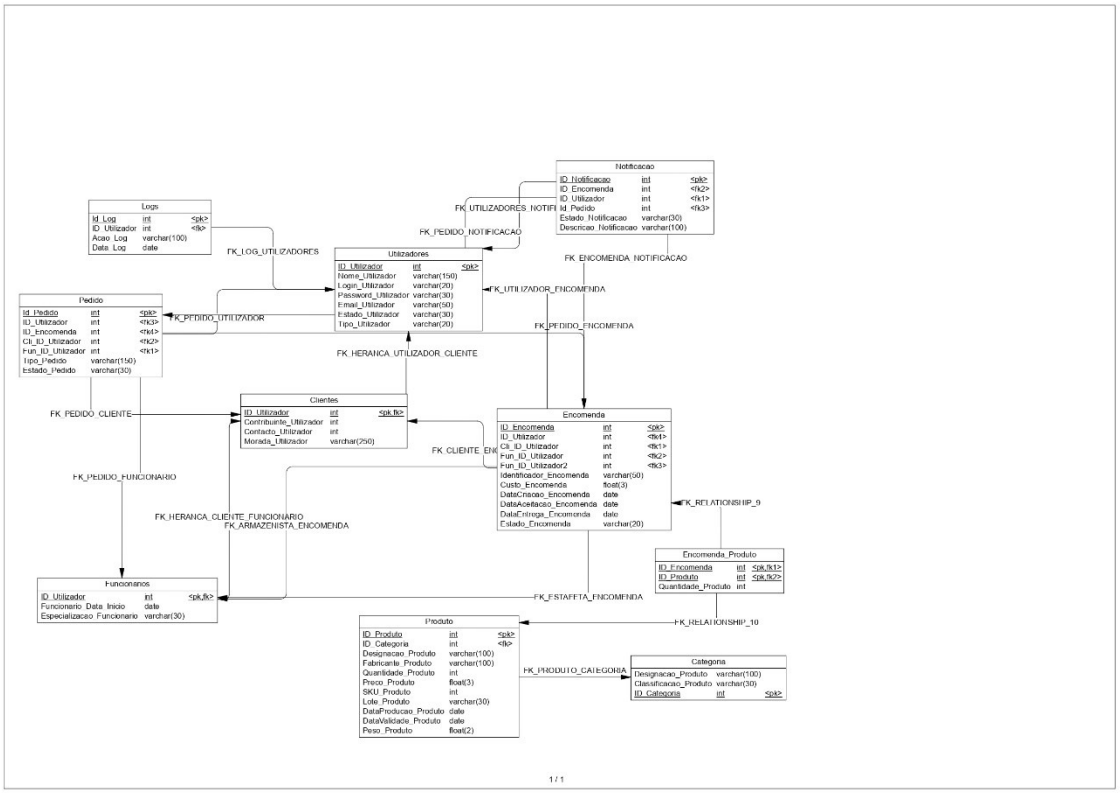

Figura 25 - Diagrama Físico

Observando o diagrama físico é possível perceber os resultados referidos em relação às chaves forasteiras existentes em cada tabela.

Este diagrama também se encontra em anexo para uma melhor legibilidade.

Para não ocupar espaço de forma desnecessária o script de criação da base de dados será enviado em anexo, este foi gerado automaticamente pelo software “power designer”, apenas adicionei o comando de “AUTO_INCREMENT” às chaves primarias para que estas sejam de incrementação automática.

3.5.4. Ficheiro “Properties”

A informação acerca dos dados necessários para que a ligação com a base de dados seja realizada de forma correta, encontra-se num ficheiro “Propertie”. Para este método utiliza-se um método de leitura e escrita de dados como de um documento de texto.

A leitura dos dados é realizada ao iniciar o software, mas na primeira camada do programa existe a possibilidade de o utilizador mudar a ligação à base de dados para uma outra que ele deseja.

```
public static boolean leituraFicheiroProp(String aCaminho) {

    if (aCaminho != null && aCaminho.length() > 0) {
      ficheiroLeitura = new File(aCaminho);
      if (ficheiroLeitura.exists()) {

        try {
          fis = new FileInputStream(ficheiroLeitura);
          return true;
        } catch (IOException ioe) {
          ioe.printStackTrace();
        }
      }
    }
    return false;
  }
```

Figura 26 - Abertura de Ficheiro

```
public static void leituraDadosProp() {
      LeituraFicheiroProp("Properties/dadosAcesso.properties");

      Properties properties = new Properties();
      try {
          properties.load(fis);
      }catch(Exception e) {
          e.printStackTrace();
      }

      DadosStatic.ip = properties.getProperty("ip");
      DadosStatic.porto = properties.getProperty("porto");
      DadosStatic.baseDados = properties.getProperty("baseDados");
      DadosStatic.utilizador = properties.getProperty("utilizador");
      DadosStatic.pass = properties.getProperty("pass");
      try {
          fis.close();
      }catch(Exception e) {
          e.printStackTrace();
      }
  }
```

Figura 27 - Retirar Dados do Ficheiro

Através da figura 26 e 27 pode-se observar o processo de abertura do ficheiro e sua leitura, retirando os valores que são guardados como que num “hash-map” ao qual cada chave devolve um valor.

Como o ficheiro “properties” está agregado à pasta do projeto então apenas foi dado o caminho relativo da mesma ao qual o explorador inicia dentro da pasta projeto.

```
public static void escritaDadosProp(String aIp, String aPorto, String aBaseDados, String aUtilizador, 


    OutputStream os = new FileOutputStream("/Properties/dadosAcesso.properties");

    Properties properties = new Properties();

    properties.setProperty("ip", aIp);
    properties.setProperty("porto", aPorto);
    properties.setProperty("baseDados", aBaseDados);
    properties.setProperty("utilizador", aUtilizador);
    properties.setProperty("pass", aPass);

    properties.store(os, null);

    os.close();
```

Figura 28 - Escrita “Propertie”

Para escrita dos dados é relativamente parecido o processo ao qual abrimos a ligação ao ficheiro (abrir o ficheiro) e em seguida são remetidos os valores para as chaves, neste processo que o ficheiro é reescrito é como se estivesse a inserir novas chaves e novos valores. A logica usada para o “update” em base de dados é usada aqui, qualquer gravação de dados, realiza uma atualização a todos os dados.

```
 * @param teclado
private static void modificaPropriedades(Scanner teclado){
    FicheiroProperties.leituraDadosProp();
    int opcao = 0;
    do{
        listaProp();
        opcao = Principal.pedeDadosInteiro("Escolha opcao a modificar: \n1-Base Dados \n2-Ip \n3-Password \n4-Porto \n5-Utilizad
        switch(opcao){
        case 1: DadosStatic.baseDados = Principal.pedeDadosString("\nInsira nova base de dados: ", teclado); Aviso.operacaoSuces
        case 2: DadosStatic.ip = Principal.pedeDadosString("\n Insira o novo Ip: ", teclado); Aviso.operacaoSucesso(); break;
        case 3: DadosStatic.pass = Principal.pedeDadosString("\n Insira nova password da Base de Dados: ", teclado); Aviso.opera
        case 4: DadosStatic.porto = Principal.pedeDadosString("\n Insira novo porto de ligação: ", teclado); Aviso.operacaoSuces
        case 5: DadosStatic.utilizador = Principal.pedeDadosString("\n Insira novo utilizador da base de dados: ", teclado); Avi
        case 6: try {
                FicheiroProperties.escritaDadosProp(DadosStatic.ip, DadosStatic.porto, DadosStatic.baseDados, DadosStatic.utiliz
                }catch(Exception e) {
                    e.printStackTrace();
                    Aviso.operacaoInsucesso();
                    break;
                }
        case 7: break;
        default: Aviso.avisoErro();

        }
    }while(opcao != 7);
```

Figura 29 - Menu “Properties”

Para que se um utilizador quiser apenas modificar um valor não estar a obrigar a modificar todos os valores, criou-se um menu que cada opção remete a uma ação de mudança a um dos valores de ligação à base de dados sendo que a sexta opção serve de gravação dos dados. Para tornar mais apelativo e informativo ao utilizador listei através do método “listaProp()” os dados atuais para que o utilizador tenha noção quais as palavras chave usadas na ligação à base de dados.

```
private static void listaProp() {
    System.out.println("Base de dados : "+DadosStatic.baseDados+"\n" + "Ip: " +DadosStatic.ip +"\n"+"Password: "+ DadosStatic.p
}
```

Figura 30 - Listar "Properties"

Ainda foi pensado em inserir palavra passe de acesso a esta opção mas devido à natureza do projeto não solicitar o mesmo e este se debruçar de um produto de exercício académico, esta opção acabou por ser abandonada.

3.6. Procedimentos de Teste

```
4-Sair
2
Insira o seu Login
a223
Insira a sua Password
123
Bem-vindo Jorge Martins
Notificação - Existem novas contas registadas para serem aceites!!

Numero de execuções do programa: 24
Data da ultima inicialização do programa: 2021-04-07

1 -  Utilizadores
2 -  Produtos
3 -  Encomendas
4 -  Pedidos ao Estafeta
5 -  Pesquisas
6 -  Logs
7 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
2
1  -  Criar Produto
2  -  Criar Categoria de Produto
3  -  Listar Categorias de Produto
4  -  Listar Produtos ordenados por Designação Ordem Crescente
5  -  Listar Produtos ordenados por Designação Ordem Decrescente
6  -  Listar Produtos por Categoria Ordem Crescente
7  -  Listar Produtos por Categoria Ordem Decrescente
8  -  Pesquisar Produtos por Designacao
9 -   Pesquisar Produtos por categoria
10 -  Pesquisar Produtos abaixo de determinado stock
11 -  Inserir Stock em Produto
12 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
1
Insira a descrição do produto
```

Figura 31 - Exemplo Teste

Para a realização de testes, tentei de certa forma procurar bugs na conceção do software, testando várias vezes com dados inválidos e válidos e observando se o seu comportamento é o esperado e correto. Mesmo no decorrer da escrita deste relatório

ainda algumas arestas estão a ser resolvidas. A bateria de testes foca-se na minha visualização do comportamento do software enquanto são inseridos os mais variados dados e tentativas de aceder indevidamente a opções interditas à conta online no momento.

4. Manual de Utilizador

Para a demonstração deste manual de utilizador serão demonstradas as situações que se poderá deparar neste software, indicando o caminho a percorrer para que as suas ações desejadas sejam assim realizadas.

Como aviso este sistema utiliza uma base de dados "mysql" que deverá estar pré instalada num sistema de base de dados.

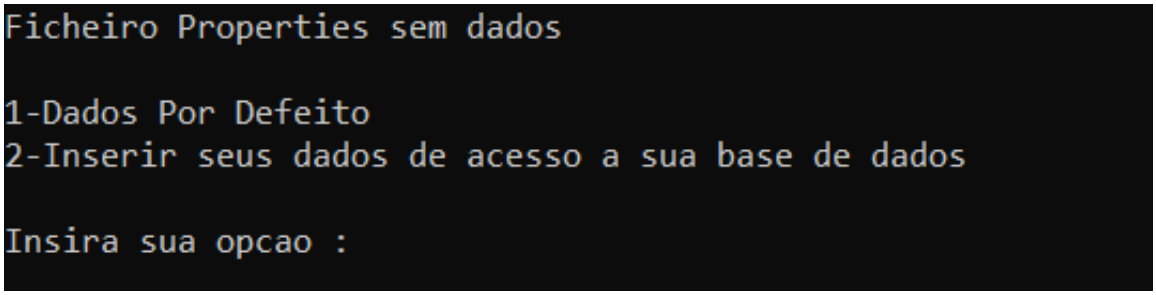

Figura 32 - Sistema Sem Ficheiro Properties

Se o Sistema não tiver um ficheiro "properties", será indicado duas opções antes do seu arranque, se o utilizador quer utilizar os dados por defeito que se incidem no uso de uma base de dados com as mesmas credenciais que o utilizador usou ou se o utilizador quer usar as suas próprias credenciais de base de dados. A segunda é a que se pensa ser de referência para um novo utilizador, assim pode logo no início indicar o nome da base de dados, ip, utilizador, password e porto da sua base de dados e iniciar o sistema.

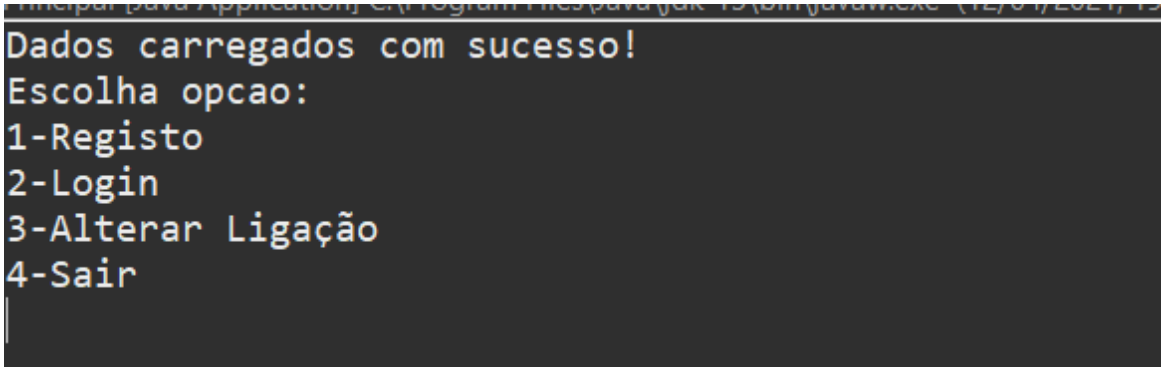

Figura 33 - Sistema Com Ficheiro Properties

Após aquele menu inicial na figura 32 ou se o sistema já tiver um ficheiro "properties" com os seus dados de acesso, é demonstrado o menu de interface login, este tem as a opção de registo de conta, ao qual um utilizador pode fazer o seu registo mas a conta

não ficará logo de imediato ativa. Pode também sair do software para não ficar em loop infino caso não tenha credenciais de acesso e pode alterar a ligação á base de dados, ou seja se o programa trabalhar com duas ou três base de dados diferentes basta o utilizador indicar nesta opção.

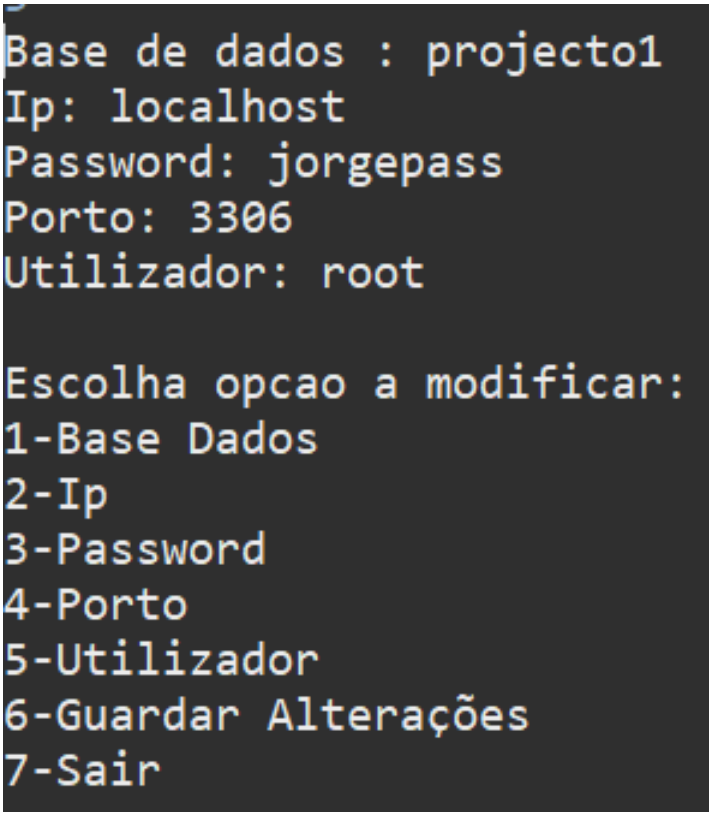

Figura 34 - Alterar Ligações

Após guardar as alterações de ligação poderá voltar ao menu de interface de entrada do sistema.

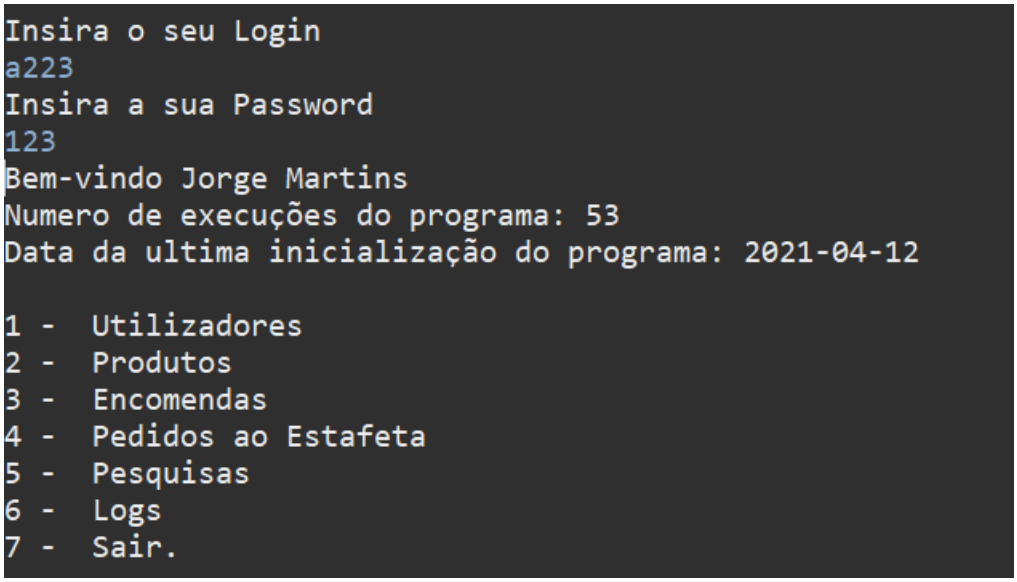

Figura 35 - Login e Menu Principal

Após realização do login o utilizador será redirecionado para o menu principal, neste menu irá receber as notificações logo abaixo do Bem-Vindo, indicando a descrição da mesma, neste caso não existem notificações então não será notificado.

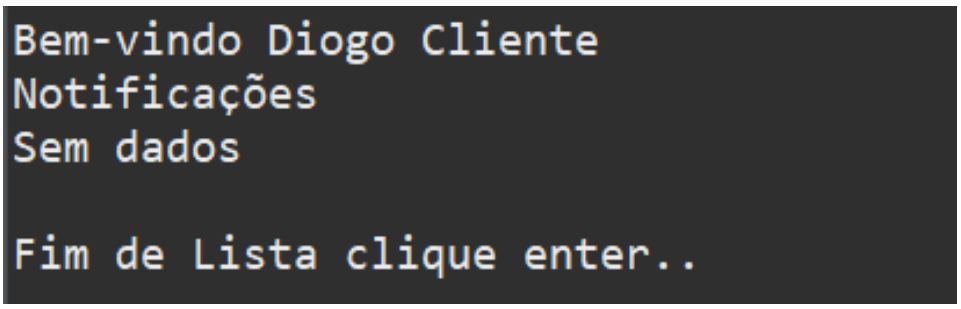

Figura 36 - Exemplo Notificação

Dando um exemplo como cliente a indicar que não existem notificações.

4.1. Menu Utilizadores

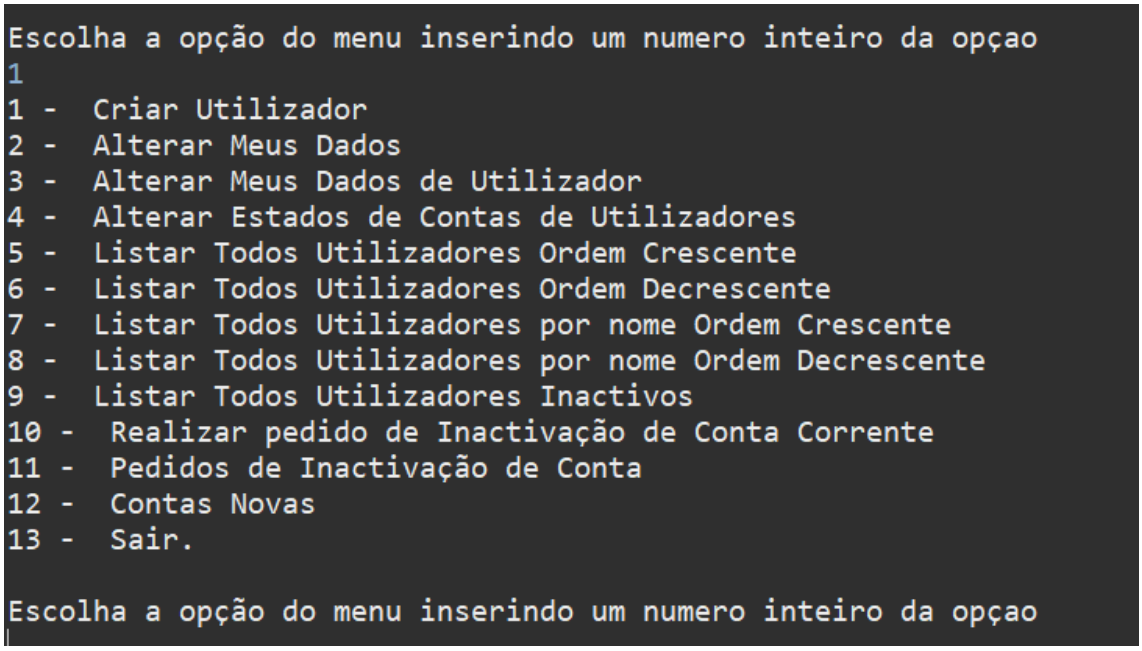

Figura 37 - Menu Utilizadores

Estas são as opções do menu de criar utilizadores, aqui poderá alterar dados, criar utilizadores, listar e verificar pedidos acerca das contas de utilizador.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
2
1 -  Alterar Nome
2 -  Alterar Password
3 -  Alterar email
4 -  Alterar Contribuinte
5 -  Alterar Contacto
6 -  Alterar Morada
7 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
```

Figura 38 - Menu Alterar Dados

Cada dado que alterar será automaticamente gravado na base de dados.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
4
Insira o login do utilizador a modificar o seu estado
c223
Insira para que estado quer mudar
1-Activo
2-Inactivo
```

Figura 39 - Menu Alterar Estado

Após indicar o login pelo qual quer mudar o estado, são demonstradas duas opções de escolha pelo qual o utilizador indica o estado ao qual vai deixar a conta.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
5
 Gestores
Login: a223 Nome: Jorge Martins
Login: a225 Nome: Admin Paulo

 Clientes
Login: c223 Nome: Cliente Mudado
Login: c224 Nome: Diogo Cliente

 Funcionarios
Login: ar223 Nome: Tigas Armas
Login: e223 Nome: Gulias Estafeta
Login: e225 Nome: Greg Estafeta
Login: ar226 Nome: Fisgas Armazenas


Fim de Lista clique enter..
```

Figura 40 – Listagem

Qualquer das opções de listagem irá listar o desejado, neste caso é uma listagem que filtra os utilizadores para que o utilizador perceba os tipos de utilizador.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
11
Sem dados para demonstrar
Fim de Lista clique enter..

Insira o login do utilizador para remover a informação
```

Figura 41 - Pedidos Inativação

Se o gestor ao iniciar perceber que existe notificação acerca de pedidos de inativação de conta poderá na opção 11 verificar as contas que realizaram o pedido para inativação de conta. Serão listadas demonstrando os dados de utilizador para que o gestor insira um login para remover a informação.

Caso o utilizador não queira desde já não remover basta inserir um login não existente que o mesmo sai desta opção (ex clicar enter apenas).

```
Escolha a opção do menu inserindo um numero inteiro da opçao
10
Insira para que estado quer mudar
1- Cancelar Pedido/Manter Activa
2-Realizar Pedido de Remoção
```

Figura 42 - Pedido Inativar

Na opção 10 o utilizador poderá realizar um pedido de inativação, ou cancelar o pedido/manter a sua conta ativa, para caso tenha acontecido algum engano.

4.2. Menu Produtos

```
Escolha a opção do menu inserindo um numero inteiro da opçao
2
1  -  Criar Produto
2  -  Criar Categoria de Produto
3  -  Listar Categorias de Produto
4  -  Listar Produtos ordenados por Designação Ordem Crescente
5  -  Listar Produtos ordenados por Designação Ordem Decrescente
6  -  Listar Produtos por Categoria Ordem Crescente
7  -  Listar Produtos por Categoria Ordem Decrescente
8  -  Pesquisar Produtos por Designacao
9 -   Pesquisar Produtos por categoria
10 -  Pesquisar Produtos abaixo de determinado stock
11 -  Inserir Stock em Produto
12 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
```

Figura 43 - Menu Produtos

Escolhendo no menu principal a opção de produtos o utilizador será direcionado para este submenu que se encontra na figura 43. Através da opção de criar produto poderá inserir os dados do produto que quer criar, mas terá de ter atenção em criar primeiro uma categoria pois um produto necessita obrigatoriamente de uma categoria, pelo qual não será possível criar um sem ela.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
3
 Categorias
Categoria [descricao=Plastico, classificacao=Alta]Categoria [descricao=Metal, classificacao=media]

Fim de Lista clique enter..
```

Figura 44 - Listar Categorias

Para que o utilizador saiba as que já existem, pode sempre utilizar a opção de listar categorias e assim visualizar as já existentes na base de dados.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
4
 Produtos
Designacao: Cabos HDMI Quantidade: 400 SKU: 146016 preco: 12.0
Designacao: Cd Quantidade: 400 SKU: 585623 preco: 1.0
Designacao: Disketes Quantidade: 300 SKU: 799214 preco: 2.0
Designacao: Teclado  Quantidade: 250 SKU: 810306 preco: 30.0
```

Figura 45 - Listagem Produtos

Ao clicar nas opções de listagem os produtos serão demonstrados através das ordens escolhidas, o esquema de visualização é o que se encontra na figura 45.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
8
Insira a Designacao de produto a pesquisar
Cd
 Produtos
Designacao: Cd Quantidade: 400 SKU: 585623 preco: 1.0


Fim de Lista clique enter..
```

Figura 46 - Pesquisa Produto

Para uma pesquisa de produto será pedido ao utilizador a palavra para pesquisar, como no exemplo acima foi o caso da designação de um produto e por conseguinte é demonstrado o resultado da pesquisa.

Neste menu existem as pesquisas, devido ao facto do menu de pesquisas não ficar demasiado extenso com opções tornando confuso para o utilizador.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
11
Indique condigo sku do produto
585623
Stock actual 400

Insira a quantidade a adicionar ao stock
13
```

Figura 47 - Inserir Stock

Para inserir o stock de um produto o mesmo primeiro solicita ao utilizador a escrita do código SKU do produto para que o mesmo seja verificado, em seguida é demonstrado o stock atual do produto e a quantidade a adicionar ao stock já existente.

4.3. Encomendas

```
Escolha a opção do menu inserindo um numero inteiro da opçao
3
1  -  Criar Encomenda
2  -  Listar TODAS Encomendas por Data de Criação Ascendente
3  -  Listar TODAS Encomendas por Data de Criação Descendente
4  -  Listar TODAS Encomendas por Antiguidade de Cliente Ascendente
5  -  Listar TODAS Encomendas por Antiguidade de Cliente Descendente
6  -  Listar TODAS Encomendas não entregue ordenada por Data Ascendente
7  -  Listar TODAS Encomendas não entregues ordenadas por Data Descendente
8  -  Listar as SUAS Encomendas ordenadas por Data Criação Ascendente
9  -  Listar as SUAS Encomendas ordenadas por Data Criação Descendente
10 -  Listar as SUAS Encomendas ordenadas por Identificador Ascendente
11 -  Listar as SUAS Encomendas ordenadas por Identificador Descendente
12 -  Listar Encomendas Iniciadas por Clientes
13 -  Aceitar/Rejeitar Encomendas
14 -  Delegar Encomenda a Armazenista
15 -  Marcar Encomenda como Entregue
16 -  Confirmar entrega de Encomenda
17 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
```

Figura 48 - Menu Encomenda

Através do menu principal, se o utilizador escolher a opção encomendas é direcionado para este submenu. Para criar uma encomenda de referenciar que têm de existir produtos, e apenas o cliente cria uma encomenda, para esta opção apenas serão pedidos os dados para a encomenda.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
2
| Encomendas
Identificador Encomenda: 120210408120656 custo: 4000.0 datacriacao: 2021-03-20 estado: confirmada


Fim de Lista clique enter..
```

Figura 49 - Listar Encomendas

Tal como nos produtos ao clicar nas opções de listagem as mesmas automaticamente serão listadas pela ordem que a opção indica. De referenciar que apenas o gestor pode listar todas as encomendas, sendo que as opções "as suas" são reservadas para as listagens a clientes e funcionários, mas os gestores também as podem listar, listando apenas as encomendas que eles são responsáveis.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
12
Encomendas
Sem dados

Fim de Lista clique enter..
```

Figura 50 - Listar Encomendas Iniciadas

Opção para listar as encomendas que foram iniciadas por clientes, assim o gestor poderá observar e escolher se aceita ou rejeita alguma das encomendas.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
13
Encomendas
Sem dados

Fim de Lista clique enter..

1 - Aceitar uma Encomenda
2 - Rejeitar uma Encomenda
3-Sair

Insira a sua opcao
```

Figura 51 - Menu Aceitar Rejeitar Encomenda

Como já referenciado nesta opção o gestor pode aceitar ou rejeitar uma encomenda, não sendo desde já obrigado, ao que poderá simplesmente sair do menu.

Obs. Rejeitar uma encomenda gera uma notificação direcionada ao cliente que a criou.

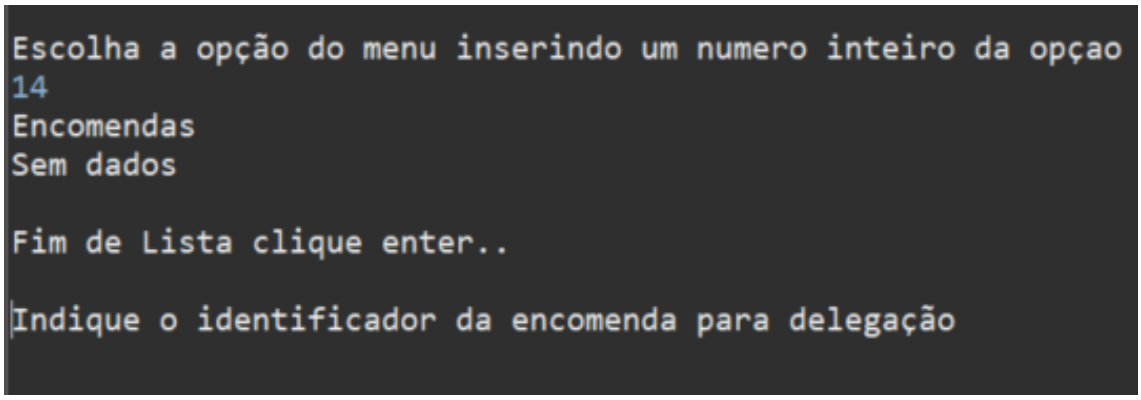

Figura 52 - Delegar encomenda a Armazenista

Nesta opção o gestor poderá delegar a encomenda a um armazenista, neste exemplo não existem encomendas iniciadas, mas será feita uma listagem das encomendas aceites pelo gestor (encontram-se no estado de aceite).

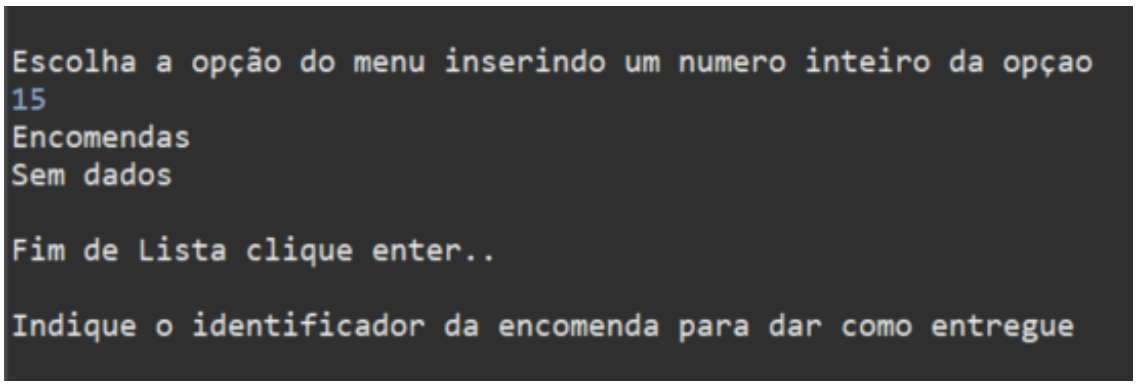

Figura 53 - Entrega de Encomenda

Nesta opção o estafeta indica que já entregou a encomenda, gerando uma notificação ao cliente que em seguida poderá confirmar a entrega da mesma. O processo é parecido ao da delegação do gestor para o armazenista.

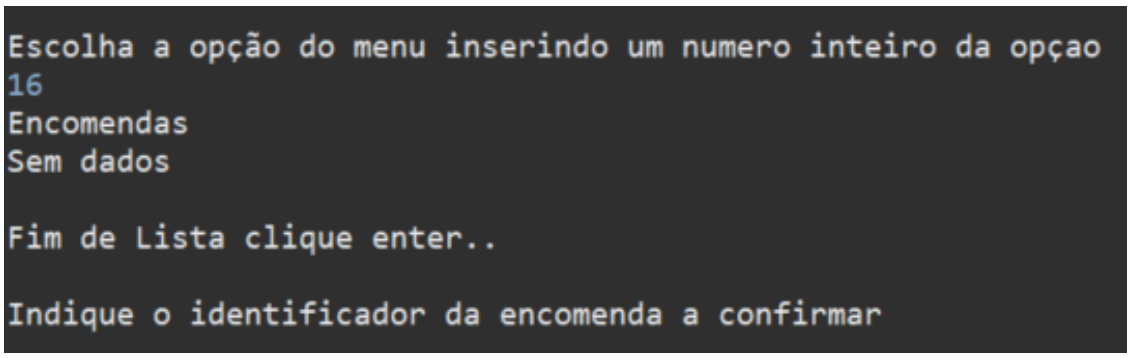

Figura 54 - Confirmar Encomenda

Mantendo a coerência do sistema o modo de funcionar da confirmação é similar aos anteriores.

4.4. Menu Pedidos

```
Escolha a opção do menu inserindo um numero inteiro da opçao
4
1 - Pedido de Entrega a Estafeta
2 -  Aceitar / Rejeitar Pedido
3 -  Sair

Escolha a opção do menu inserindo um numero inteiro da opçao
1
Encomendas
Sem dados

Fim de Lista clique enter..

Insira o identificador da encomenda a delegar
```

Figura 55 - Menu Pedidos

Ao clicar na opção pedidos é mostrado um menu com a primeira opção reservada para o armazenista realizar um pedido ao estafeta e na segunda opção reservada ao estafeta aceitar ou rejeitar um pedido de entrega que lhe seja transmitido. O Funcionamento das duas opções é similar pois é dada a lista de encomendas que estão com o estado e ligadas ao seu utilizador para ação desejada.

4.5. Menu Pesquisas

```
Escolha a opção do menu inserindo um numero inteiro da opçao
5
1  -  Pesquisar Utilizador pelo Login
2  -  Pesquisar Utilizador pelo Nome
3  -  Pesquisar Utilizador pelo Tipo
4  -  Pesquisar Encomendas Por Data de Criação
5  -  Pesquisar Encomendas Por Estado
6  -  Pesquisar Encomendas por Identificador
7  -  Pesquisar Encomendas por cliente
8  -  Pesquisar Encomendas por intervalo de tempo
9  -  Pesquisar SUAS Encomendas por Data de Criação
10 -  Pesquisar SUAS Encomendas por Identificador
11 -  Pesquisar SUAS Encomendas por Estado
12 -  Sair.

Escolha a opção do menu inserindo um numero inteiro da opçao
```

Figura 56 - Menu Pesquisas

Ao escolher o menu de pesquisas são demonstradas as opções restantes de pesquisa, porque existem algumas já no menu de produtos, pelo qual estas se dividem como as listagens. Pesquisas gerais são reservadas para a conta de gestor, enquanto que as pesquisas pessoais são filtradas para a conta online.

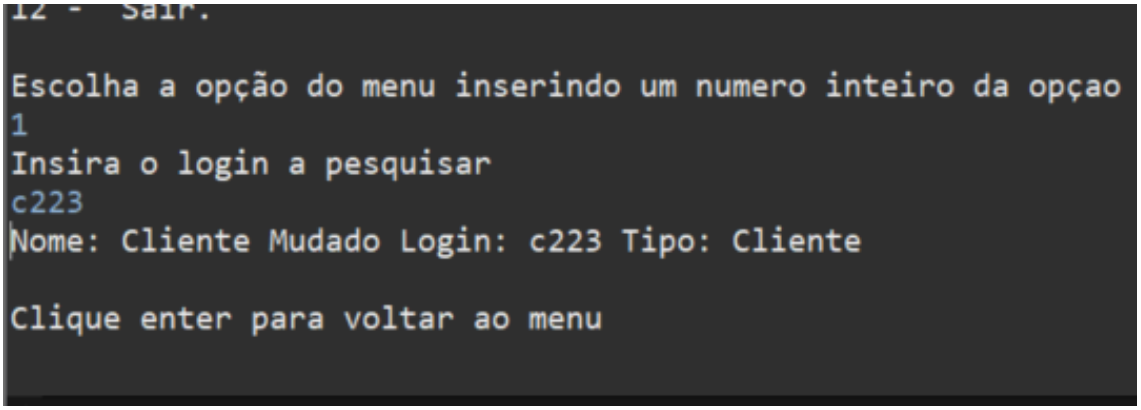

Figura 57 – Pesquisa

Mantendo a coerência do software o método de pesquisa é bastante similar, pedindo os dados ao utilizador para pesquisa e demonstrando esse resultado.

Salientar que pesquisas por nome são pesquisas avançadas, o que resulta numa pesquisa pelo conjunto de caracteres inseridos estarem apenas inseridos dentro de uma "string" (nome de utilizador).

4.6. Menu Logs

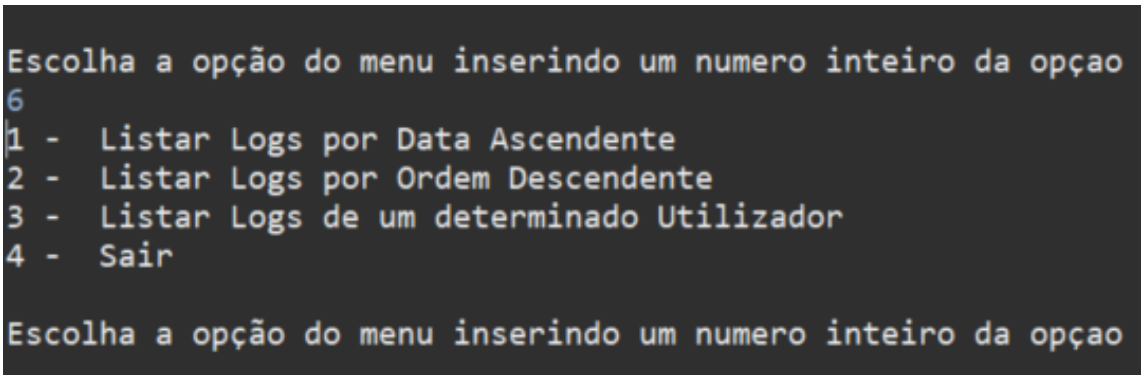

Figura 58 - Menu Logs

Ao escolher a opção de "Logs" do menu principal o utilizador é direcionado para o menu dos mesmos ao qual são listados todos os movimentos por data ascendente ou por ordem descendente. Tendo também uma opção de listar/pesquisar os logs(movimentos) de um determinado utilizador.

```
Escolha a opção do menu inserindo um numero inteiro da opçao
3
Indique o login para pesquisar os logs
a223
 Logs
Login: a223 Acção: Iniciou o Programa Data: 2021-04-06
Login: a223 Acção: Listou Logs asc Data: 2021-04-06
Login: a223 Acção: Entrou No menu logs Data: 2021-04-06
Login: a223 Acção: Entrou no Menu Produtos Data: 2021-04-06
Login: a223 Acção: Listou Logs asc Data: 2021-04-06
Login: a223 Acção: Entrou No menu logs Data: 2021-04-06
Login: a223 Acção: Listou as Categorias Data: 2021-04-06
Login: a223 Acção: Listar Produtos Designacao Crescente Data: 2021-04-06
Login: a223 Acção: Pesquisa Produto Designacao Data: 2021-04-06
Login: a223 Acção: Entrou no Menu Produtos Data: 2021-04-06


Clique enter para visualizar a proxima pagina
```

Figura 59 - Listar Logs

A listagem dos mesmo segue os parâmetros de 10 por página, ao qual todo o software segue essa regra, na figura 59 visualizamos uma pesquisa condicionada de um log de um utilizador em específico. Basta inserir o login do utilizador e a listagem condicionada será transmitida.

5. Conclusões

Este foi um projeto que me fez desenvolver bastante enquanto programador, pois no anterior semestre foram obtidos conhecimentos acerca da linguagem java e de base de dados, mas a sua ligação e conexão foi algo possível neste projeto. Desta forma também me possibilitou aprofundar e consolidar certos conhecimentos ao longo do desenvolvimento do projeto.

Contrariamente a relatórios anteriores decidi não criar um relatório com demasiada informação, pois documentação densa tende a algumas vezes confundir o leitor na perceção do funcionamento e construção do programa, assim foi entendido que os pontos principais acerca do funcionamento do mesmo foram abordados, demonstrando as logicas adotadas.

5.1. Forças

Este é um programa funcional e eficaz para os requisitos desejados, através de uma ligação de base de dados relacional manipula dados relativos a utilizadores, produto e encomendas, gerando notificações aos utilizadores de ações a prevenir ou de atuação imediata.

Tanto para o utilizador com uma interface simples e tentando manter alguma logica de aproximação das ações a realizar por cada menu, a construção do software ao se manter na mesma logica de funcionamento nos seus variados métodos torna o software simples de se compreender.

5.2. Limitações

Este trabalho ou projeto tem algumas limitações e uma encontra-se na sua eficiência, sei que poderia ter alguns dos métodos mais simplificados de forma a reutilizar em mais situações, (caso no order by dos utilizadores) que poderia apenas conter um método que receberia a condição já com o “where” ou com o “order by” tornando o código mais reduzido.

Uma limitação que encontrei foi a falta de tempo, mesmo tendo conseguido realizar o projeto antes do tempo exigido, devido há existência de outros trabalhos para outras disciplinas não foi possível dedicar mais tempo na observação de bugs / erros no software, tal como na simplificação do código de forma a este ser o mais eficiente possível.

5.3. Trabalho Futuro

Para uma melhoria no projeto seria dedicação de mais tempo na resolução de erros que possam surgir de situações atípicas do software que algumas vezes apenas sucedem após várias variáveis estarem reunidas.

Algo que penso que seria de valorizar seria uma adição de alguns métodos, como a existência de um “super” gestor que tivesse apenas acesso aos ficheiros “properties”, ou uma existência de métodos que aplicassem o uso de menus específicos para cada tipo de utilizador.

6. Referências

Página da Internet:

Páginas relativas a manipulação de Datas:

https://www.geeksforgeeks.org/localdate-parse-method-in-java-with-examples/

https://www.guj.com.br/t/como-utilizar-a-funcao-parse-date/72210/2

https://coderanch.com/t/395538/java/import-SimpleDateFormat

https://www.guj.com.br/t/transformar-a-data-em-um-inteiro/48781/16

https://www.javatpoint.com/java-string-to-date

https://commons.apache.org/proper/commons-validator/apidocs/org/apache/commons/validator/routines/DateValidator.html

https://dzone.com/articles/techtip-use-setlenient-method

https://www.convertworld.com/pt/tempo/horas.html

Páginas relativas a expressão regular:

https://pt.stackoverflow.com/questions/1386/express%C3%A3o-regular-para-valida%C3%A7%C3%A3o-de-e-mail

Páginas relativas a transações e mySql:

https://docs.microsoft.com/en-us/sql/t-sql/language-elements/rollback-transaction-transact-sql?view=sql-server-ver15

https://www.heidisql.com/forum.php?t=15612

https://cursos.alura.com.br/forum/topico-statement-fica-em-aberto-92878

https://www.heidisql.com/forum.php?t=10931

https://www.tutorialspoint.com/jdbc/commit-rollback.htm

https://pt.coredump.biz/questions/32739719/what-happens-on-connectionsetautocommit-false

Páginas relativas a ficheiros Properties:

https://www.youtube.com/watch?v=OBnjTszNe3c

https://www.youtube.com/watch?v=Vp4bI36GFCo

https://www.youtube.com/watch?v=w7D5YB2U2jU

7. Anexos

1. Ficheiros Java e Extras
    - Ficheiros da pasta src com suas packages e ficheiros .java
    - Pasta lib com Jar de conector de mysql
    - Pasta com ficheiro Properties com credenciais da base de dados
    - Diagramas de Classes Gerados pela Eclipse Organizados por Package
2. Javadoc
3. Projecto1_BaseDados_Scripts_Diagramas
    - BD_Proj1.sql (Script de criação de Base de Dados)
    - Diagrama Conceptual em Pdf
    - Diagrama Fisico em Pdf
    - Diagramas Fisico e Conceptual do software PowerDesing
4. Projecto1.jar (Executavel .jar)
5. Diagramas de Classes em Imagens