



76B9 - ATIVIDADES PRATICAS SUPERVISIONADAS

CONTROLE DE ESTOQUE

Victor Thauan de Andrade - N634CG0

Vitor Bonfadini – F197Al6

Pedro Lorencini Favarão - N549645

João Victor Lorencini Favarão - N5496E2

SÚMARIO

Figura	Página
1.- Objetivo e motivação	4-5
2.– Introdução	5-6
3.- Referencial Teórico	7–9
4.- Plano de desenvolvimento da aplicação (elementos e ferramentas que serão utilizadas)	10–12
5. - Projeto (estrutura e módulos que serão desenvolvidos) do programa.....	13–17
5.1. Model–view–controller (MVC)	13-13
5.2. CRUD	14-14
5.3. Código	14-17
6- Relatório com as linhas de código do programa	18–23
7- Apresentação do programa em funcionamento em um computador, apresentando todas as funcionalidades pedidas e extras.....	24–27
8– Bibliografia	28–28
9- Ficha de Atividades Práticas Supervisionadas	29–30

1. Objetivo e motivação.

Essa atividade tem como objetivo aprofundar os conhecimentos a respeito da matéria para com os alunos do curso e assim, prepará-los psicologicamente e profissionalmente para o mercado de trabalho que os mesmos irão encarar ou já encaram.

Coloca a prova o conhecimento dos alunos no quesito “codificação” em Java com integração com Banco de Dados, tanto ao trabalho em equipe (fator que será comum na rotina dos mesmos). Além desses pontos importantes, os alunos são induzidos a realizar pesquisas nos fóruns da internet (ou outras fontes) para conseguir concluir a atividade em questão, influenciando indiretamente a habilidade de pesquisa deles.

O grupo responsável por esse relatório chegou à decisão de elaborar um programa que funcionaria para a organização de estoque onde o usuário tivesse todo o controle sobre os produtos vendidos, valores, entrada e saída de vendas. E partindo do ponto de que sem uma boa organização de estoque, a ordenação desses registros torna-se de certa forma necessária, evitando futuras frustrações e complicações com os mesmos.

Cidades interioranas e de pequeno porte que comportam alguma venda física, ainda utilizam cadernos para realizar a ação de controle de seus produtos, seja pelo fato de um número controlado de atendimentos diários - tornando a implementação de um sistema direcionado a isso “desnecessária” – ou pelo fato de que as pessoas que trabalham nesses locais serem acostumadas a esta maneira, e quando recebem algum computador para realizar as suas atividades profissionais, optam pelo papel e caneta.

Abaixo temos alguns motivos extras que corroboram para a substituição do papel e caneta por teclado e mouse:

- Torna a organização local cada vez mais difícil;

- À medida que o número de atendimentos aumenta e as atividades se intensificam, manter o controle de todos esses dados se torna uma tarefa cada vez mais complicada;
- A cada nova marcação de consulta ou a cada novo cadastro de venda/produto, papéis são retirados do lugar e precisam ser organizados novamente. E essa é uma atividade constante. O mesmo vale para os dados coletados e gerados pelos setores financeiros e administrativos do local. É preciso espaço físico disponível e muito cuidado para separar e arquivar documentos, recibos, contratos, notas fiscais etc;
- Essa rotina baseada no uso do papel e da caneta compromete a produtividade dos colaboradores, aumenta a ocorrência de falhas e dificulta o gerenciamento das informações e das atividades da clínica;
- Dificulta o armazenamento e o acesso a informações;
- Ainda com um processo de organização bem feito, o acesso a esses dados pode ser complicado e lento, gerando transtornos e até prejuízos;

*Dados retirados de: CBA Tech -

2. Introdução

A gestão de venda deve ser implantada nos estabelecimentos para fazer com que a empresa tenha lucro e prosperidade, o que difere um bar de esquina de uma empresa de medio porte é justamente as ferramentas que utilizam, caneta e papel já ficou ultrapassado, com: controle de produto, historico de venda, valor do produto no fornecedor, valor do produto no varejo, entre outros para controlar, seria necessario muito mais tempo e dedicação para anotar tudo isso no papel. Uma empresa que visa diminuir o gasto indevidos, necessita de uma ferramentas administrativas a cumprir na sua rotina de fiscalização. Entre todas as responsabilidades, a gestão dos produtos é uma das mais importantes, a qual merece atenção especial visto a importância que exerce no atendimento ao cliente.

Do ponto de vista do cliente, é necessario que haja algum metodo de controle, pois ninguem gosta de comprar um produto e quando for retirar, receber a noticia que o mesmo nao possui em estoque, partindo desse ponto, clientes contam exclusivamente com esses serviços para obter atendimento e realizar compras.

Mudando de lado na visão da empresa, o ERP (Enterprise Resource Planning) oferece acessibilidade a informações importantes sobre os clientes a partir do sistema de qualquer departamento. Isso também facilita na hora da atualização desses dados.

Graças ao fácil acesso, os logistas, por exemplo, podem consultar esses registros rapidamente. Esses profissionais valorizam muito o uso do ERP, pois a solução os ajuda a garantir que o diagnóstico seja feito de maneira precisa. Auxiliando também na economia sobre a consulta do valor mais em conta, o aumento que teve, o que está bom para investir no momento e os lucros.

Também é fundamental que o estabelecimento trabalhe com recursos tecnológicos modernos tendo assim uma correlação com as melhorias práticas para administrar os serviços de saúde com as mais diversas complexidades, assim como oferecer boas condições de trabalho a seus profissionais.

Muitos empreendedores relegam a gestão do estoque a um segundo plano por gerirem um negócio de pequeno porte. Esse pensamento faz parte de um mito que deve ser eliminado.

Ter um controle total do que entra e sai da sua empresa evita prejuízos significativos devido a furtos e também aumenta as chances de encontrar oportunidades de negociação com os fornecedores. Ao mesmo tempo, torna mais fácil criar promoções para os clientes, organizar demandas e vender mais. (Conta Azul – Controle de estoque)

Organizar as informações em geral é importante para inovar e aperfeiçoar os processos de gestão no dia a dia do trabalho administrativo de uma entidade. Com o auxílio de tal tecnologia, todas as informações referentes podem ser acessadas de um só lugar, de forma organizada e controlada. Tal tecnologia pode aumentar a produtividade dos profissionais, facilitar o acesso aos serviços disponíveis e reduzir custos administrativos associados à prestação de serviços.

Bases para a elaboração do texto:

- Carin Tom – Guia do controle de estoque para pequenas empresas
- Marcos Leite – O ERP é acessível para Pequenas Empresas?
- Egestor – Controle de estoque: tudo o que você precisa saber

3. Referencial Teórico.

CBAtech – **Por que caneta e papel impedem o desempenho?**
(18/11/2019) -

<https://www.cbatech.com.br/?p=214883>

Acesso em: 05/04/2021

Egestor – **Controle de estoque: tudo o que você precisa saber**
(04/02/2021) -

<https://blog.egestor.com.br/controle-de-estoque/>

Acesso em: 05/04/2021

Conta Azul – **Guia do controle de estoque para pequenas empresas**
(07/08/2018) -

<https://blog.contaazul.com/como-fazer-gestao-de-estoque>

Acesso em: 05/04/2021

Artsoft – **O ERP é acessível para Pequenas Empresas?** -

<https://www.artsoftsistemas.com.br/blog/o-erp-e-acessivel-para-pequenas-empresas/>

Acesso em: 06/04/2021

Devmedia – **Java: Crie uma Conexão com Banco de dados**
(2016) -

<https://www.devmedia.com.br/java-crie-uma-conexao-com-banco-de-dados/5698>

Acesso em: 26/04/2021

Devmedia – **Explorando a Classe ArrayList no Java**
(2012) -

<https://www.devmedia.com.br/explorando-a-classe-arraylist-no-java/24298>

Acesso em: 29/04/2021

Postgres	–	Documentação	-
https://www.postgresql.org/docs/			
Acesso em: 03/05/2021			
Incc	–	Interfaces Gráficas	-
https://www.incc.br/~rogerio/poo/04a%20-%20Programacao_GUI.pdf			
Acesso em: 04/05/2021			
Devmedia	–	Java Swing (2014)	-
https://www.devmedia.com.br/java-swing-conheca-os-componentes-jtextfield-e-jformattedtextfield/30981			
Acesso em: 04/05/2021			
Javatpoint	–	Java String	-
https://www.javatpoint.com/java-string-format			
Acesso em: 10/05/2021			
Treinaweb	–	O que é MVC?	-
https://www.javatpoint.com/java-string-format			
Acesso em: 10/05/2021			
Devmedia	–	Introdução ao Padrão MVC (2013)	-
https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308			
Acesso em: 10/05/2021			
Devmedia	–	Tratando exceções em Java (2012)	-
https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308			
Acesso em: 12/05/2021			

Boson Treinamentos – **O Modelo Entidade-Relacionamento**
(11/10/2020) -

<https://www.bosontreinamentos.com.br/modelagem-de-dados/o-modelo-entidade-relacionamento-introducao/>

Acesso em: 19/05/2021

Oracle – **Documentação: Package** -

<https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html>

Acesso em: 20/05/2021

While True – **Programação em geral** -

https://youtube.com/channel/UCI4mJ2FXeA-RuDwZA0z_MA

Acesso em: 24/05/2021

Code Java – **Programação em geral** -

<https://youtube.com/channel/UC5ZPG2xyYljsUIAXcCutWsQ>

Acesso em: 24/05/2021

Curso em Video – **Programação em geral** -

https://www.cursoemvideo.com/?gclid=EAlaIqobChMlrZTW0r_Z8AIVTAWICR1zwAzxE_AAYASAAEgltf_D_BwE

Acesso em: 24/05/2021

4. Plano de desenvolvimento da aplicação

Antes da implementação do projeto ser iniciada, foi preciso traçar uma linha de raciocínio para que pudesse chegar à conclusão de como estruturar o aplicativo da melhor forma possível, tendo em vista que todos integrantes do grupo tinham sugestões para a estruturação do software, foi necessário fazer uma análise minuciosa para selecionar as ideias mais relevantes para o mesmo. Após ser definida a funcionalidade do aplicativo e os elementos que iriam compô-lo durante seu desenvolvimento, deu-se início ao projeto.

Nesta aplicação podemos ressaltar que foram usados alguns hardwares e softwares para a execução e a conclusão do projeto. Nelas contidas:

- **IDE:** Foi utilizado o IDE NetBeans versão 12.0 LTS. O NetBeans IDE é um ambiente de desenvolvimento integrado gratuito e de código aberto para desenvolvedores de software nas linguagens Java, JavaScript, HTML5, PHP, C/C++, Groovy, Ruby, entre outras. O IDE é executado em muitas plataformas, como Windows, Linux, Solaris e MacOS. Além disso, é usado como base de uma série de software científico de missão crítica em grandes organizações em defesa, aeroespacial, logística e pesquisa, como Boeing, Airbus Defense and Space, NASA e NATO.

A IDE NetBeans auxilia programadores a escrever, compilar, depurar e instalar aplicações, e foi arquitetada em forma de uma estrutura reutilizável que visa simplificar o desenvolvimento e aumentar a produtividade, pois reúne em uma única aplicação todas estas funcionalidades. Totalmente escrita em Java, mas que pode suportar qualquer outra linguagem de programação que desenvolva com Swing, como C, C++, Ruby e PHP. Também suporta linguagens de marcação como XML e HTML.

O NetBeans fornece uma base sólida para a criação de projetos e módulos, possui um grande conjunto de bibliotecas, módulos e APIs (Application Program Interface, um conjunto de rotinas, protocolos e ferramentas para a construção de aplicativos de software) além de uma documentação vasta — inclusive em português — bem organizada. Tais recursos auxiliam o desenvolvedor a escrever seu software de maneira mais rápida.

Alguns dos seus principais recursos são:

- editor de código fonte integrado, rico em recursos para aplicações Web (Servlets e JSP, JSTL, EJBs) e aplicações visuais com Swing que é uma API (Interface de Programação de Aplicativos) Java para interfaces gráficas. A API Swing procura desenhar por contra própria todos os componentes, ao invés de delegar essa tarefa ao sistema operacional, como a maioria das outras APIs de interface gráfica trabalham;
- visualizador de classes integrado ao de interfaces, que gera automaticamente o código dos componentes de forma bem organizada, facilitando assim o entendimento de programadores iniciantes;
- suporte ao Java Enterprise Edition, plataforma de programação de computadores que faz parte da plataforma Java voltada para aplicações multicamadas, baseadas em componentes que são executados em um

servidor de aplicações;

- plugins para UML, Unified Modeling Language, linguagem de modelagem não proprietária de terceira geração, e desenvolvimento remoto em equipes; interface amigável com CVS ou Concurrent Version System (Sistema de Versões Concorrentes) é um sistema de controle de versão que permite que se trabalhe com diversas versões de arquivos organizados em um diretório e localizados local ou remotamente, mantendo-se suas versões antigas e os logs de quem e quando manipulou os arquivos;
- CSS, algumas funcionalidades para editar folhas de estilos como destaques, recursos de autocompletar, análise de código;
- help local e on-line; debug apurado de aplicações e componentes;
- autocompletar avançado; total suporte ao ANT, ferramenta de automatização da construção de programas e TOMCAT, servidor de aplicações Java para web;
- integração de módulos;
- suporte a Database (banco de dados), Data view e Connection wizard que são os módulos embutidos na IDE; geração de Javadoc: a ferramenta permite a geração automática de arquivos javadoc em HTML a partir dos comentários inseridos no código, além de recursos que facilitam a inclusão de comentários no código.
- atalhos para copiar linhas inteiras de código.

- **JDK:** Neste projeto utilizamos o Java SE Development Kit versão 11.0.10 64Bits, acreditando ser a melhor versão para a conclusão do projeto.

O JDK, abreviação para Java Development Kit, é um conjunto de utilitários cuja finalidade é a permissão para criação de jogos e programas para a plataforma Java. Este pacote é disponibilizado pela Oracle, e nele vem todo o ambiente necessário para a criação e execução dos aplicativos java.

O Java JDK é composto pelo compilador e pelas bibliotecas (API's) necessárias para criação de programas em Java e ferramentas úteis para o desenvolvimento e para testes dos programas escritos por esta linguagem de programação. Além disso, uma Máquina Virtual Java é adicionada ao sistema operacional, no caso de ainda não ter uma instalada no computador.

O JDK dispõe de um arquivo executável que faz todo o trabalho de instalação e configuração do ambiente, o que facilita ainda mais a execução de qualquer aplicação Java e criação de novos programas de forma mais simplificada e sem muitos esforços.

- **SGBD:** Para o banco de dados utilizamos o PostgreSQL versão 10 juntamente com a sua interface PgAdmin4, para elaborar e complementar a manipulação e tratativa dos dados necessários.

O PostgreSQL é um dos SGBDs (Sistema Gerenciador de Bancos de Dados) de código aberto mais avançados, contando com recursos como:

- Consultas complexas
 - Chaves estrangeiras
 - Integridade transacional
 - Controle de concorrência multi-versão
 - Suporte ao modelo híbrido objeto-relacional
 - Facilidade de Acesso
 - Gatilhos
 - Visões
 - Linguagem Procedural em várias linguagens (PL/pgSQL, PL/Python, PL/Java, PL/Perl) para Procedimentos armazenados
 - Indexação por texto
 - Estrutura para guardar dados Georreferenciados PostGIS
- **Sistema Operacional:** O SO utilizado junto ao Hardware e aos Softwares foi o Windows 10 Professional 20H2 64Bits. O Windows 10 traz aprimoramentos à interface de linha de comando do sistema, também sendo possível executar um arquivo .class e rodar o programa via prompt de comando.
 - **Hardware:** Parte do desenvolvimento foi feito em um notebook modelo Dell Inspiron 5570, Processador Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz, Memória RAM 8,00 GB, SSD 240G, Placa de Vídeo AMD Radeon 4GB, Mouse sem Fio Logitech.

Por fim pode-se afirmar de forma categórica que o projeto agregou tanto em conhecimento (reforçando o que foi visto no decorrer do semestre) quanto em proporcionar a relação interpessoal para o desenvolvimento de um código em conjunto. Com essa vivência houve o vislumbre do que poderia vir a ser a rotina de

um programador em uma equipe.

As pesquisas feitas aguçaram o senso de equipe, permitindo desenvolver um senso de pesquisa que consiste em analisar, separar e utilizar informações relevantes para desenvolver softwares.

5. Projeto (estrutura e módulos que serão abordados)

Neste projeto utilizamos a metodologia MVC, utilizamos também parcialmente uma espécie de CRUD, digo parcialmente pelo fato de não ter sido implementado um CRUD por completo, onde a opção deletar dados no banco de dados seria uma má prática, e utilizamos uma conexão DAO com o banco de dados.

5.1 Model–view–controller (MVC)

- O que é MVC?

MVC é nada mais que um padrão de arquitetura de software, separando sua aplicação em 3 camadas. A camada de interação do usuário(view), a camada de manipulação dos dados(model) e a camada de controle(controller).

- **Model**

Sempre que você pensar em manipulação de dados, pense em model. Ele é responsável pela leitura e escrita de dados, e também de suas validações.

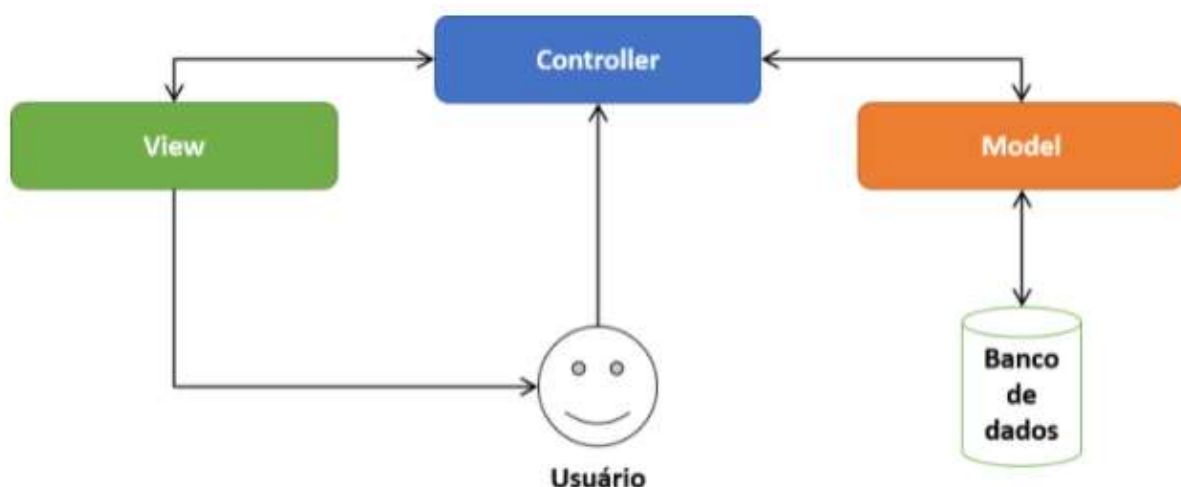
- **View**

Simple: a camada de interação com o usuário. Ela apenas faz a exibição dos dados, sendo ela por meio de um html ou xml.

- **Controller**

O responsável por receber todas as requisições do usuário. Seus métodos chamados actions são responsáveis por uma página, controlando qual model usar e qual view será mostrado ao usuário.

Veja no exemplo abaixo:



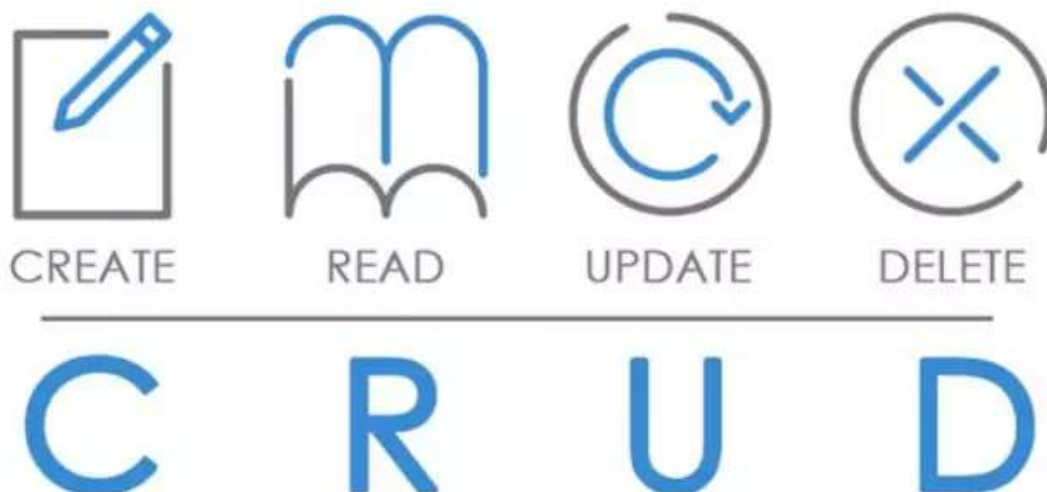
5.2 CRUD

- O que é CRUD?

Em linhas simples, as interfaces CRUD que lhe permitem cadastrar (create), visualizar (read), editar (update) e excluir (delete) registros de um sistema.

- **Create:** Podemos considerar a funcionalidade 'Create' como a mais importante em uma interface CRUD, afinal, sem ela as demais não teriam utilidade. Permite criar e registrar novos registros para então manipula-los.
- **Read:** Ler um registro, ou um conjunto de registros. Permitindo assim poder acessar os dados contidos/salvos em um banco de dados.
- **Update:** Nos permite atualizar um registro que já esta no banco de dados.
- **Delete:** Excluir um registro.

Veja no exemplo abaixo:



5.3 Código

Tendo essas duas metodologias em mente (MVC, CRUD) podemos assim iniciar o resumo do projeto. Logo abaixo explicamos a Classe UsuarioDAO como exemplo, e deixamos de ressaltar as demais classes pois são métodos espelho e tem a mesma função.

O MVC foi utilizado para assim facilitar não só na criação e tempo do desenvolvimento do projeto como também para facilitar o próprio entendimento e manutenções futuras.

O CRUD não foi totalmente implementado a risca o método de deletar um registro em minha opinião seria uma má pratica a ser executada.

5.3.1 DAO: Dando início ao código podemos começar com a pasta “DAO” nela contida a base do projeto responsável por fazer a conexão com o banco de dados.

Segue abaixo classes utilizadas para o desenvolvimento do projeto:

conexao.java: Responsável em efetuar a conexão do java com o banco de dados via URL por autenticação de login e senha.

UsuarioDAO: Nela contida os métodos necessários para a execução do programa.

Insert: Recebe como parâmetro um objeto do tipo usuário assim pegando os dados contido no objeto, podendo então transferir e salvar todos os dados no banco de dados.

pegaUsuario: Recebe um objeto do tipo “usuário” como parâmetro para pegar o nome de usuário contido nele, para assim então retornar e espelhar os atributos do banco de dados e atribui-los no objeto usuário.

validarUsuarioSenha: É um método que retorna um valor booleano e recebe um objeto do tipo “usuario”, para pegar os atributos de usuário e senha deste objeto e verificar se esse parâmetro contém no banco de dados, retornando assim “true” ou “false”.

Update: Método que recebe um objeto do tipo “usuário” aonde pega todos atributos que contém no mesmo e sobrepõem nos valores dos atributos na entidade do banco de dados.

selectAll: Método que nos retorna uma ArrayList do tipo “usuário” onde pega cada tupla da entidade no banco de dados convertendo cada uma delas para um objeto e salvando nesta ArrayList, assim pegando todos usuário que contém no banco de dados.

5.3.2 Model: São as classes que espelham as entidades no banco de dados, tendo nelas contidos seus atributos, com suas getters e setters.

5.3.3 Controller: Utilizamos a classe “FormUsuarioController” nela contida todos os chamados de métodos necessários para cadastrar, atualizar e consultar os usuários.

salvarUsuario: estanciamos os objetos conexão e usuarioDAO para acessarmos e manipular os dados no banco de dados, com os métodos contidos no usuarioDAO.

btnCadastrar: Como o próprio nome diz, é o botão que ao ser acionado salvamos um novo usuário dentro do banco de dados (este método está sendo chamado em um JFrame).

atualizarUsuario: Responsavel por chamar os métodos necessários para atualizar os dados do usuário no banco de dados.

Btnid: Chamamos este método para pegarmos o valor do ID do usuário que queremos atualizar.

preenchertabelaAll: Nela fazemos a manipulação de uma Jtable recebendo assim uma ArrayList como parâmetro para preencher e imprimir todos usuários contidos no banco de dados.

Consultar: Esta responsável por chamar todos os métodos para retornar e imprimir os valores desejados na Jtable.

btnConsultar: Conforme diz o nome, este método pega o valor que digitamos em uma JFrame e transferimos para o método consultar da própria classe.

FormVendaController: Também utilizando a classe FormVendaController utilizamos alguns métodos conforme abaixo:

executarVenda: Este método é o último método a ser executado depois de efetuar todos os recebimento de produto para assim então o método btnFinalizarVenda dar a execução ao seu proposito.

gerarIDPedido: Neste método ele é gerado um ID com 5 numeros aleatórios não iguais ao banco de dados para assim então salvar no seu atributo id de pedido que é uma primary key considerando o banco de dados.

Preenchertabela: Ele atualiza em tempo real toda e qualquer alteração feita no próprio atributo da classe do tipo ArrayList.

btnImprimirCliente: Esta responsável em consultar no banco de dados e pegar o cliente informado e nos retornar para salvar na Venda.

btnImprimirProduto: Também responsável em consultar no banco de dados e pegar o produto informado e nos retornar para salvar no carrinho da venda.

btnAdicionarCarrinho: Todo código de barras valido que digitamos na interface do front-end pegamos o produto no banco de dados e manipulamos seus dados e adicionamos em uma ArrayList do tipo venda para assim então imprimir em uma Jtable utilizando o método preencher tabela para visualizar todo o carrinho.

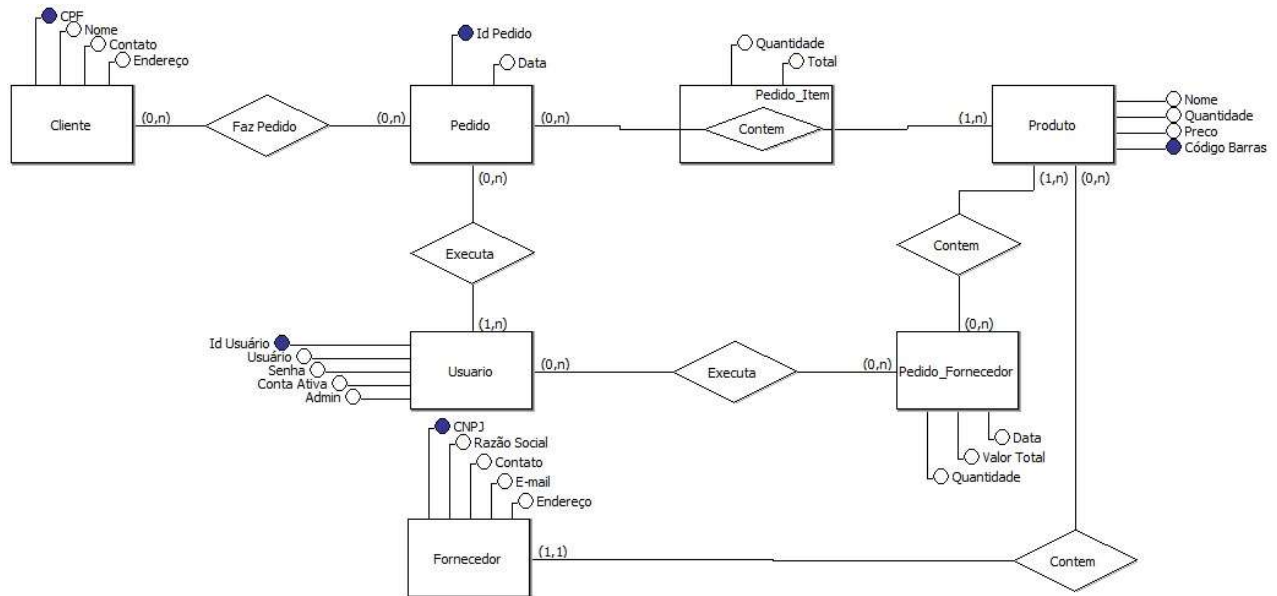
btnExcluirItem: Metodo que chamamos um JOptionPane para introduzir o item do carrinho a ser excluído.

btnFinalizarVenda: Verifica se o carrinho está vazio ou contém algum item para ser finalizado, assim então chamando o método necessário para executa-lo.

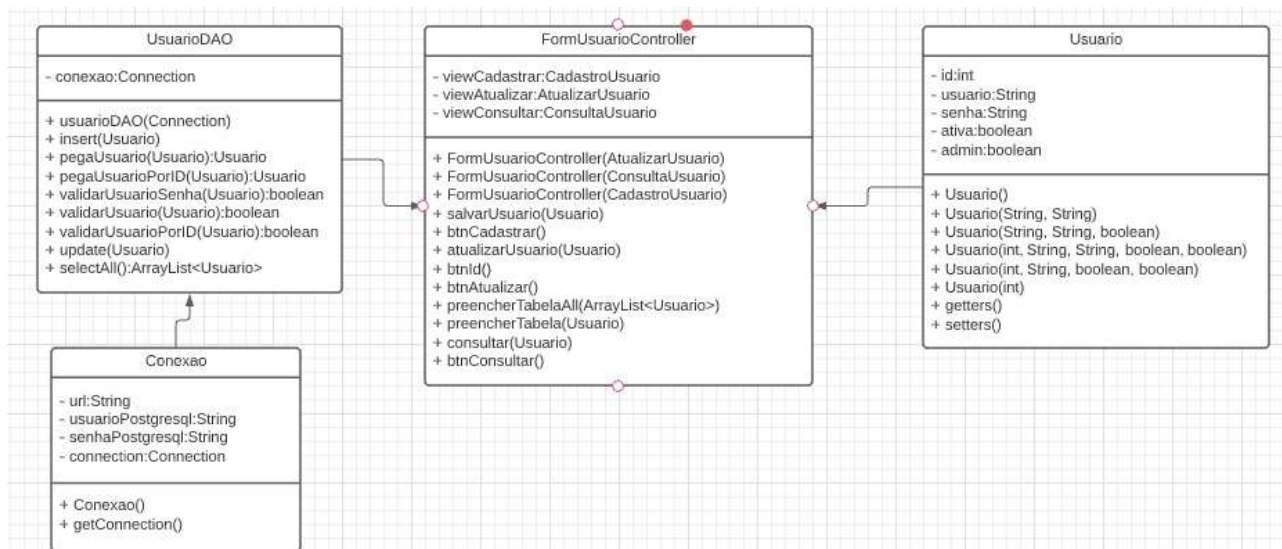
FormMenuUsuarioController: Nela esta contido os botões para dar funcionalidade no Front-end.

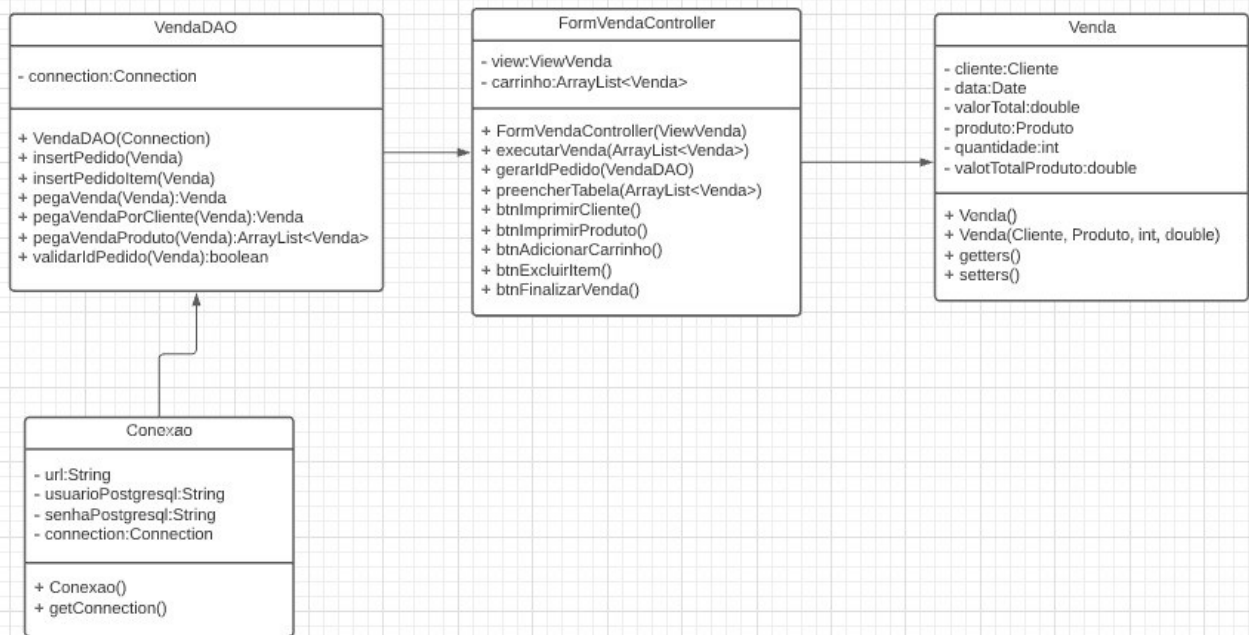
5.3.4 View: Generalizando todas as telas do JFrame utilizadas para interação do usuário com o programa estão localizadas nesta pasta.

5.3.5 Modelo ER:



5.3.6 Diagrama de Classe:





- Não foram utilizadas todas as classes pelo simples fato de todas as outras classes terem os mesmos métodos.

6. Relatório com as linhas de código.

6.1 Classe conexão com o Banco de Dados:

```

1 package dao;
2 import java.sql.*;
3
4 public class Conexao {
5     private String url;
6     private String usuarioPostgresql;
7     private String senhaPostgresql;
8     private Connection connection;
9
10    public Conexao() {
11        this.url = "jdbc:postgresql://localhost:5432/Controle_de_Estoque";
12        this.usuarioPostgresql = "postgres";
13        this.senhaPostgresql = "123456";
14    }
15
16    public Connection getConnection() throws SQLException {
17        connection = DriverManager.getConnection(url, usuarioPostgresql, senhaPostgresql);
18        return connection;
19    }
20
21 }
  
```

6.2 Classe UsuarioDAO:

```

1 package dao;
2 import java.sql.*;
3 import java.util.ArrayList;
4 import model.Usuario;
5
6 public class UsuarioDAO{
7     private final Connection connection;
8
9     public UsuarioDAO(Connection connection) {
10         this.connection = connection;
11     }
12
13
14     public void insert(Usuario usuario) throws SQLException{
15
16         String sql = "INSERT INTO tb_usuario(usuario, senha, conta_ativa, admin)" +
17             "VALUES (?, ?, ?, ?)";
18         PreparedStatement statement = connection.prepareStatement(sql);
19         statement.setString(1, usuario.getUsuario());
20         statement.setString(2, usuario.getSenha());
21         statement.setBoolean(3, usuario.isAtiva());
22         statement.setBoolean(4, usuario.isAdmin());
23         int rows = statement.executeUpdate();
24         if(rows > 0){
25             System.out.println("A new user has been inserted.");
26         }
27     }
28
29     public Usuario pegaUsuario(Usuario usuario) throws SQLException{
30         String sql = "SELECT * FROM tb_usuario WHERE usuario = ?";
31         PreparedStatement statement = connection.prepareStatement(sql);
32         statement.setString(1, usuario.getUsuario());
33         ResultSet result = statement.executeQuery();
34
35         while(result.next()){
36             int id_usuarioBD = result.getInt("id_usuario");
37             String usuarioBD = result.getString("usuario");
38             String senhaBD = result.getString("senha");
39             boolean conta_ativa = result.getBoolean("conta_ativa");
40             boolean admin = result.getBoolean("admin");
41
42             usuario.setId(id_usuarioBD);
43             usuario.setUsuario(usuarioBD);
44             usuario.setSenha(senhaBD);
45             usuario.setAtiva(conta_ativa);
46             usuario.setAdmin(admin);
47         }
48
49         return usuario;
50     }
51
52     public Usuario pegaUsuarioPorID(Usuario usuario) throws SQLException{
53         String sql = "SELECT * FROM tb_usuario WHERE id_usuario = ?";
54         PreparedStatement statement = connection.prepareStatement(sql);
55         statement.setInt(1, usuario.getId());
56         ResultSet result = statement.executeQuery();
57
58         while(result.next()){
59             String usuarioBD = result.getString("usuario");
60             String senhaBD = result.getString("senha");
61             boolean conta_ativa = result.getBoolean("conta_ativa");
62             boolean admin = result.getBoolean("admin");
63             usuario.setUsuario(usuarioBD);
64             usuario.setSenha(senhaBD);
65             usuario.setAtiva(conta_ativa);
66             usuario.setAdmin(admin);
67         }
68
69         return usuario;
70     }
71
72     public boolean validarUsuarioSenha(Usuario usuario) throws SQLException {
73         String sql = "SELECT * FROM tb_usuario WHERE usuario = ? and senha = ?";
74         PreparedStatement statement = connection.prepareStatement(sql);
75         statement.setString(1, usuario.getUsuario());
76         statement.setString(2, usuario.getSenha());
77         ResultSet result = statement.executeQuery();
78
79         while(result.next()){
80             String usuarioBD = result.getString("usuario");
81             String senhaBD = result.getString("senha");
82             boolean conta_ativa = result.getBoolean("conta_ativa");
83             boolean admin = result.getBoolean("admin");
84
85             System.out.println("Usuário: "+usuarioBD+" - Senha: "+senhaBD+" - Conta Ativa: "+conta_ativa+" - Admin: "+admin);
86             if(usuarioBD != "" && senhaBD != ""){
87                 return true;
88             }
89         }
90
91         return false;
92     }
93
94     public boolean validarUsuario(Usuario usuario) throws SQLException {
95         String sql = "SELECT * FROM tb_usuario WHERE usuario = ?";
96         PreparedStatement statement = connection.prepareStatement(sql);
97         statement.setString(1, usuario.getUsuario());
98         ResultSet result = statement.executeQuery();
99         while(result.next()){
100             String usuarioBD = result.getString("usuario");

```

```

102         String usuarioBD = result.getString("usuario");
103
104         if(usuarioBD != ""){
105             return true;
106         }
107     }
108     return false;
109 }
110
111 public boolean validarUsuarioPorId(Usuario usuario) throws SQLException {
112     String sql = "SELECT * FROM tb_usuario WHERE id_usuario = ?";
113     PreparedStatement statement = connection.prepareStatement(sql);
114     statement.setInt(1, usuario.getId());
115     ResultSet result = statement.executeQuery();
116     while(result.next()){
117         String idBD = result.getString("id_usuario");
118
119         if(idBD != ""){
120             return true;
121         }
122     }
123     return false;
124 }
125
126 public void update(Usuario usuario) throws SQLException {
127     String sql = "UPDATE tb_usuario SET senha = ?, conta_ativa = ?, admin = ? WHERE id_usuario = ?";
128     PreparedStatement statement = connection.prepareStatement(sql);
129     statement.setString(1, usuario.getSenha());
130     statement.setBoolean(2, usuario.isAtiva());
131     statement.setBoolean(3, usuario.isAdmin());
132     statement.setInt(4, usuario.getId());
133     statement.execute();
134 }
135
136 public ArrayList<Usuario> selectAll () throws SQLException{
137     ArrayList<Usuario> usuarios = new ArrayList<Usuario>();
138
139     String sql = "SELECT * FROM tb_usuario";
140
141     PreparedStatement statement = connection.prepareStatement(sql);
142     ResultSet result = statement.executeQuery();
143
144     while(result.next()){
145         int id_usuarioBD = result.getInt("id_usuario");
146         String usuarioBD = result.getString("usuario");
147         String senhaBD = result.getString("senha");
148         boolean conta_ativaBD = result.getBoolean("conta_ativa");
149         boolean adminBD = result.getBoolean("admin");
150
151         Usuario usuario = new Usuario(id_usuarioBD, usuarioBD, senhaBD, conta_ativaBD, adminBD);
152         usuarios.add(usuario);
153     }
154     return usuarios;
155 }
156 }
157

```

6.3 Classe FormUsuarioController:

```

1 package controller;
2
3 import dao.*;
4 import java.util.logging.*;
5 import model.*;
6 import java.sql.*;
7 import java.util.ArrayList;
8 import javax.swing.JOptionPane;
9 import javax.swing.table.DefaultTableModel;
10 import view.*;
11
12 public class FormUsuarioController {
13     private CadastroUsuario viewCadastro;
14     private AtualizarUsuario viewAtualizar;
15     private ConsultaUsuario viewConsultar;
16
17     public FormUsuarioController(CadastroUsuario view) {
18         this.viewCadastro = view;
19     }
20
21     public FormUsuarioController(AtualizarUsuario viewAtualizar) {
22         this.viewAtualizar = viewAtualizar;
23     }
24
25     public FormUsuarioController(ConsultaUsuario viewConsultar) {
26         this.viewConsultar = viewConsultar;
27     }
28
29 // ----- CADASTRAR USUARIO -----
30 public void salvarUsuario(Usuario usuarioCadastro){
31     try {
32         Connection conexao = new Conexao().getConnection();
33         UsuarioDAO usuarioDAO = new UsuarioDAO(conexao);
34
35         boolean existeUsuario = usuarioDAO.validarUsuario(usuarioCadastro);
36
37         if(existeUsuario){
38             JOptionPane.showMessageDialog(null, "Usuário já existente!");
39         }else{
40             usuarioDAO.insert(usuarioCadastro);
41             JOptionPane.showMessageDialog(null, "Usuário salvo com sucesso!");
42             viewCadastro.setVisible(false);
43         }
44     } catch (SQLException ex) {
45         Logger.getLogger(CadastroUsuario.class.getName()).log(Level.SEVERE, null, ex);
46     }
47 }
48
49 public void btnCadastro(){
50     String user = viewCadastro.getCadastroUser().getText().trim();
51     String password = viewCadastro.getCadastroPassword().getText().trim();
52     boolean isAdmin = viewCadastro.getCheckAdmin().isSelected();
53
54     Usuario usuarioCadastro = new Usuario(user, password, isAdmin);
55
56     if(user.length() >= 1){
57         if(password.length() >= 6){
58             this.salvarUsuario(usuarioCadastro);
59         }
60     }
61 }

```

```

59         this.salvarUsuario(usuarioCadastro);
60     }else{
61         JOptionPane.showMessageDialog(null, "Digite uma senha com 6 caracteres ou mais!");
62     }
63     }else{
64         JOptionPane.showMessageDialog(null, "Usuário inválido!");
65     }
66 }
67
68 // ----- ATUALIZAR USUARIO -----
69 public void atualizarUsuario(Usuario usuarioAtualizar){
70     try {
71         Connection conexao = new Conexao().getConnection();
72         UsuarioDAO usuarioDAO = new UsuarioDAO(conexao);
73
74         boolean existeUsuario = usuarioDAO.validarUsuarioPorId(usuarioAtualizar);
75
76         if(!existeUsuario){
77             JOptionPane.showMessageDialog(null, "Usuário não encontrado!");
78         }else{
79             usuarioDAO.update(usuarioAtualizar);
80             JOptionPane.showMessageDialog(null, "Usuário atualizado com sucesso!");
81         }
82     }
83     } catch (SQLException ex) {
84         Logger.getLogger(FormUsuarioController.class.getName()).log(Level.SEVERE, null, ex);
85     }
86 }
87
88 public void btnId(){
89     String idString = viewAtualizar.getTxtId().getText();
90
91     int id = 0;
92     if(idString.length() < 1){
93     }else{
94         id = Integer.parseInt(idString);
95     }
96
97     try {
98         Connection conexao = new Conexao().getConnection();
99         UsuarioDAO usuarioDAO = new UsuarioDAO(conexao);
100
101         Usuario usuario = new Usuario(id);
102         usuario = usuarioDAO.pegarUsuarioPorID(usuario);
103
104         viewAtualizar.setTxtUsuario(usuario.getUsuario());
105         viewAtualizar.setTxtSenha(usuario.getSenha());
106         viewAtualizar.setCheckAtiva(usuario.isAtiva());
107         viewAtualizar.setCheckAdmin(usuario.isAdmin());
108
109         viewAtualizar.setUser(usuario.getUsuario());
110     } catch (SQLException ex) {
111         Logger.getLogger(FormUsuarioController.class.getName()).log(Level.SEVERE, null, ex);
112     }
113 }
114
115 }

```



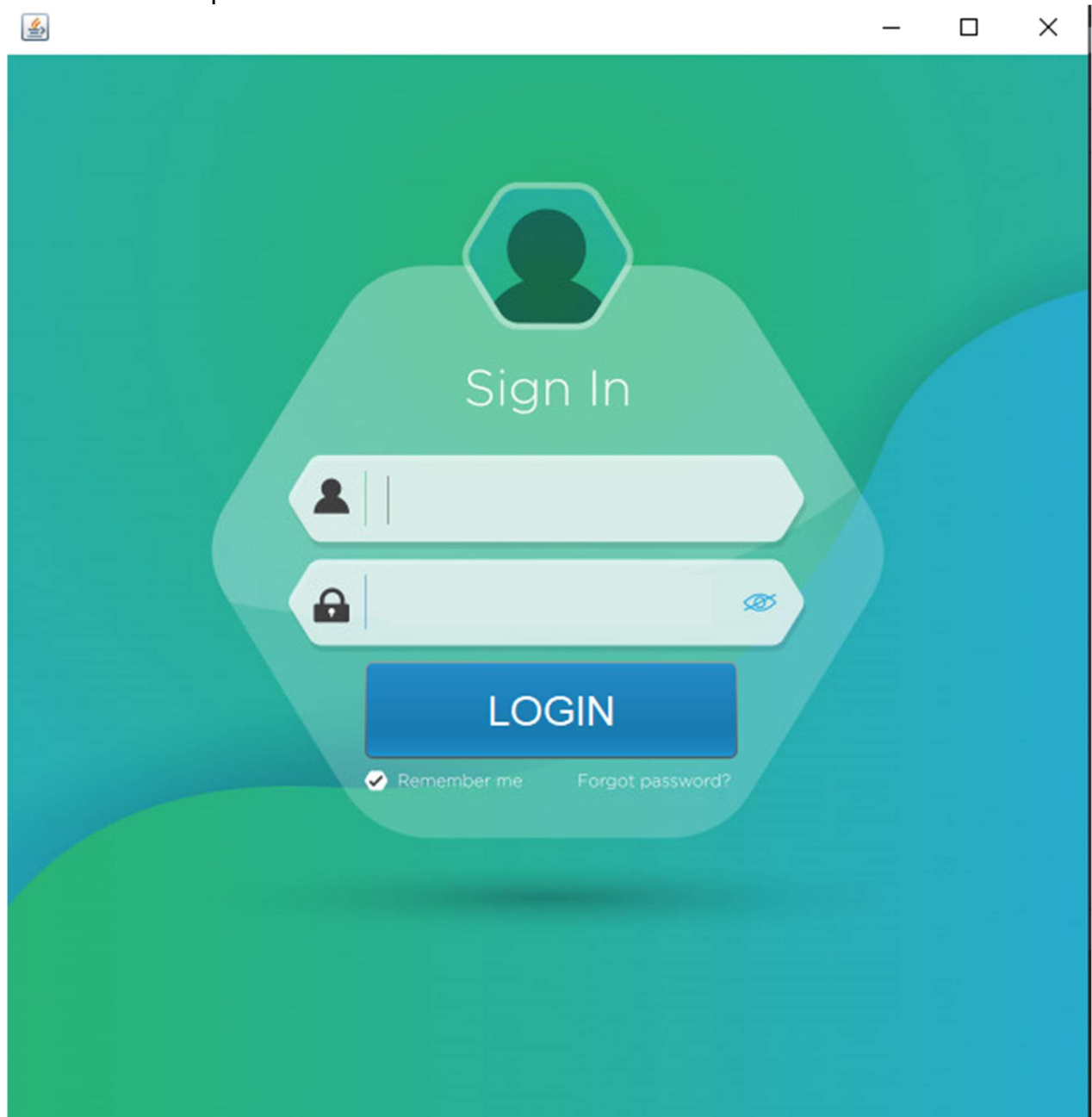
```

117 public void btnAtualizar(){
118     String idString = viewAtualizar.getId().getText();
119
120     String senha = viewAtualizar.getTxtSenha().getText().trim();
121     boolean conta_ativa = viewAtualizar.getCheckAtiva().isSelected();
122     boolean admin = viewAtualizar.getCheckAdmin().isSelected();
123
124     if(idString.length() < 1){
125         JOptionPane.showMessageDialog(null, "Digite um ID!");
126     }else if(senha.length() < 6){
127         JOptionPane.showMessageDialog(null, "Digite uma senha com 6 caracteres ou mais!");
128     }else{
129         int id = Integer.parseInt(idString);
130         Usuario usuarioAtualizar = new Usuario(id, senha, conta_ativa, admin);
131         this.atualizarUsuario(usuarioAtualizar);
132         viewAtualizar.setVisible(false);
133     }
134 }
135
136
137
138 // ----- CONSULTAR USUARIO -----
139
140 public void preencherTabelaAll(ArrayList<Usuario> usuarios){
141     DefaultTableModel tableModel = (DefaultTableModel) viewConsultar.getTableConsultar().getModel();
142     tableModel.setNumRows(0);
143
144     for (Usuario usuario : usuarios) {
145         tableModel.addRow(new Object[]{
146             usuario.getId(),
147             usuario.getUsuario(),
148             usuario.getSenha(),
149             usuario.isAtiva(),
150             usuario.isAdmin()
151         });
152     }
153 }
154
155 public void preencherTabela(Usuario usuario){
156     DefaultTableModel tableModel = (DefaultTableModel) viewConsultar.getTableConsultar().getModel();
157     tableModel.setNumRows(0);
158
159     tableModel.addRow(new Object[]{
160         usuario.getId(),
161         usuario.getUsuario(),
162         usuario.getSenha(),
163         usuario.isAtiva(),
164         usuario.isAdmin()
165     });
166 }
167
168 public void consultar(Usuario usuario){
169     try {
170         Connection conexao = new Conexao().getConnection();
171         UsuarioDAO usuarioDAO = new UsuarioDAO(conexao);
172
173         if(usuario.getUsuario().length() > 0){
174             usuario = usuarioDAO.pegarUsuario(usuario);
175             this.preencherTabela(usuario);
176         }else{
177             this.preencherTabelaAll(usuarioDAO.selectAll());
178         }
179     } catch (SQLException ex) {
180         Logger.getLogger(FormUsuarioController.class.getName()).log(Level.SEVERE, null, ex);
181     }
182 }
183
184
185 public void btnConsultar(){
186     String usuarioString = viewConsultar.getTxtConsultar().getText();
187
188     Usuario usuario = new Usuario();
189     usuario.setUsuario(usuarioString);
190
191     this.consultar(usuario);
192 }
193

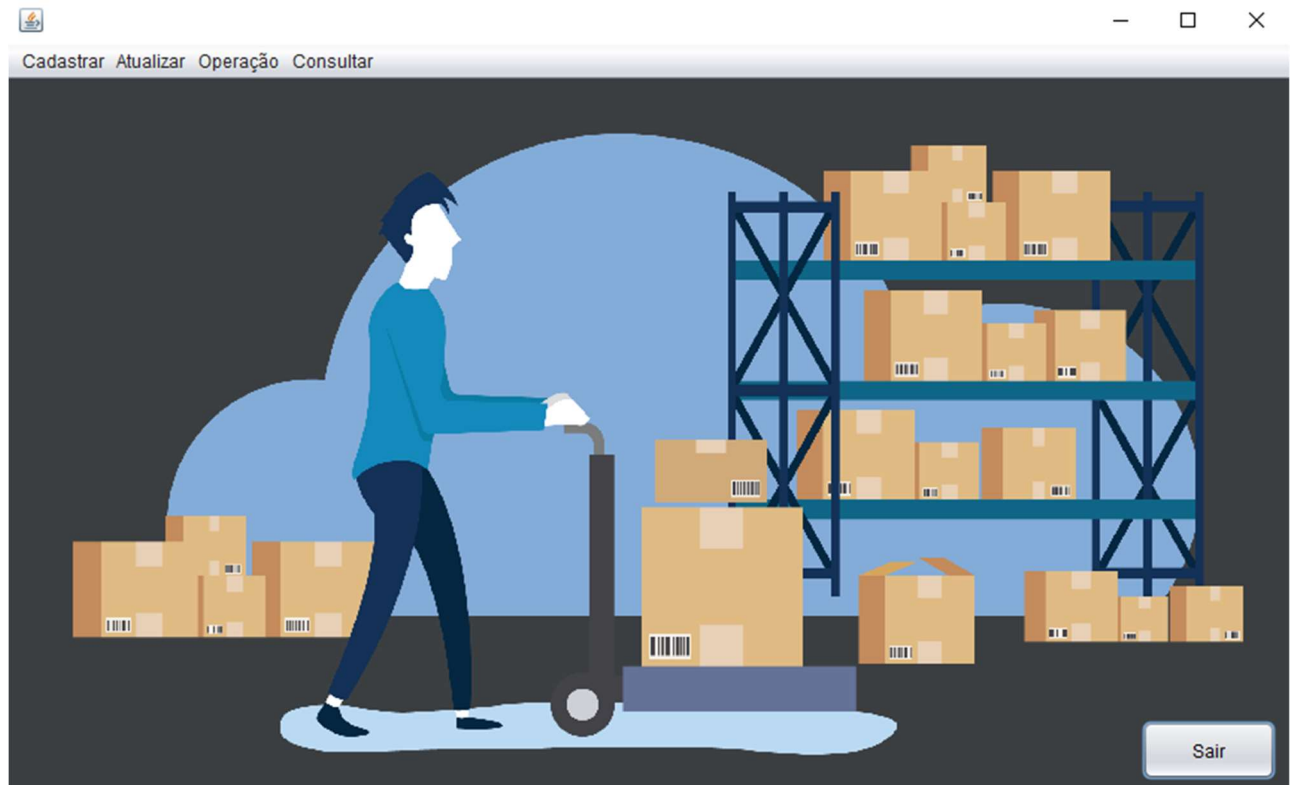
```

7. Apresentação do programa em funcionamento em um computador, apresentando todas as funcionalidades pedidas e extras.

7.1 **Login Usuário:** Logo abaixo temos a tela de login para o usuário, e também serve para administrador.



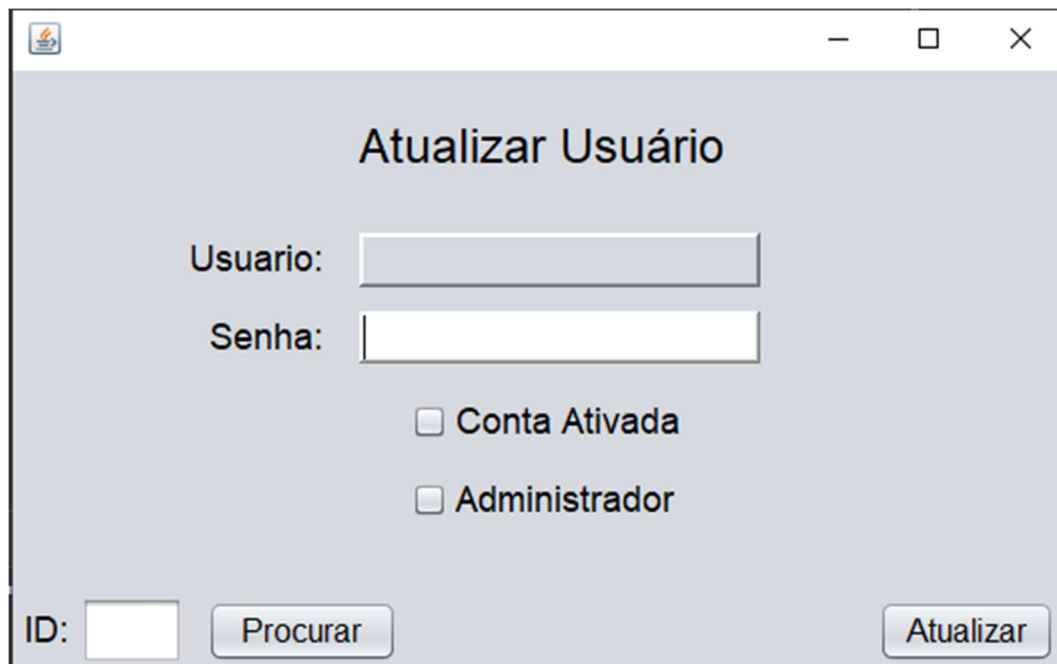
7.2 Menu Usuário: Seguindo o exemplo abaixo temos a tela de Menu, aonde contém as opções de Cadastrar, Atualizar, Operação, Consultar, e botão sair para efetuar o logoff do programa.



7.3 Cadastrar usuário: A tela abaixo é aonde cadastramos um novo usuário, exigindo um usuário e senha, se for o caso também tem a opção para dar permissão de administrador.

A screenshot of a 'Cadastro Usuário' (User Registration) form. The form has a light gray background. At the top, the title 'Cadastro Usuário' is centered. Below the title, there are two labels: 'Usuário:' and 'Senha:'. Each label is followed by a white rectangular input field. Below the 'Senha:' input field, there is a light gray button with the text 'Salvar'. At the bottom right of the form, there is a checkbox followed by the text 'Admin'.

7.4 Atualizar usuário: Neste exemplo temos a tela de atualizar um usuário, aonde a busca do usuário é feita pelo ID (o ID é gerado por ordem de criação de usuários seguindo a sequência numérica) nesta tela temos a opção de desativar um usuário, atualizar senha e/ou dar permissão de administrador.



Atualizar Usuário

Usuario:

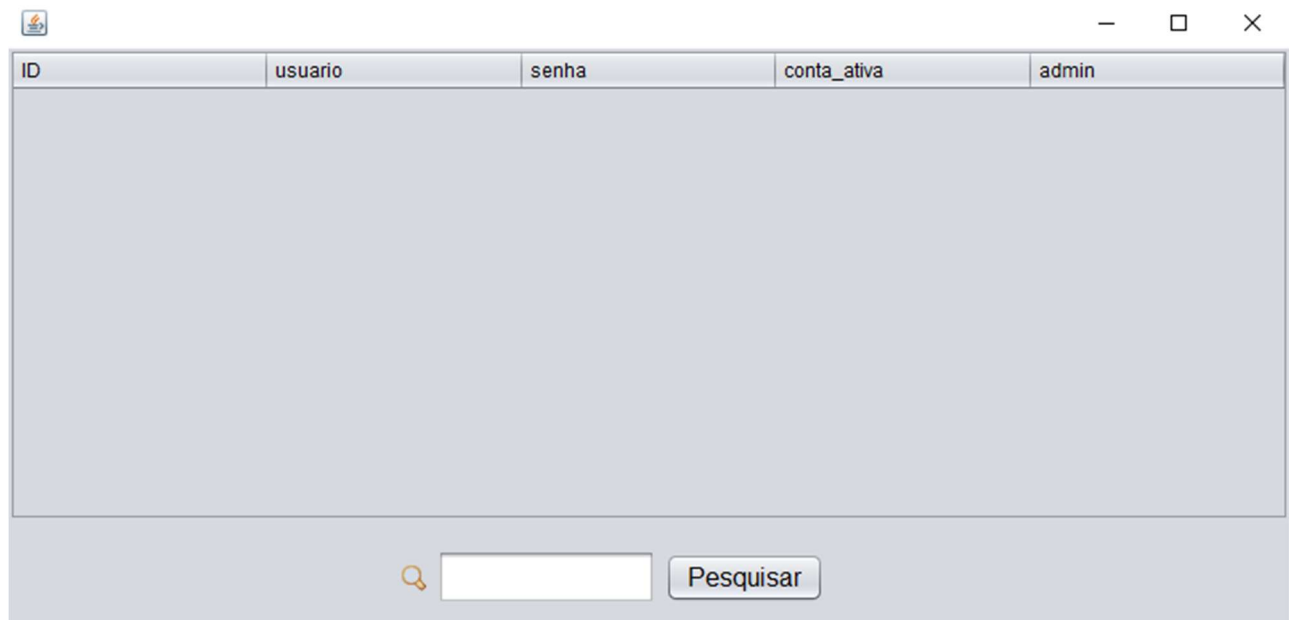
Senha:

☐ Conta Ativada

☐ Administrador


ID:

7.5 Consultar usuário: Neste exemplo abaixo temos a tela de consulta de usuário, onde pode se fazer uma consulta geral apenas deixando o campo em branco e clicando em Pesquisar, ou consultar um nome de usuário específico.



ID	usuario	senha	conta_ativa	admin
----	---------	-------	-------------	-------

7.6 Venda: Na tela abaixo temos a tela de venda, aonde consultamos o cliente pelo CPF ou CNPJ cadastrado, consultamos o produto pelo Código do Produto, escolhemos a quantidade e adicionamos ao carrinho, após confirmação do valor finalizamos a Venda.

- □ ×

Cliente


CPF

Código Produto

Nome Produto

Valor Unit.

Quantidade



Carrinho

Item	Produto	Quantidade	Total

Total: R\$ 0,00

Finalizar Venda

Excluir Item

7.7 Bibliografia.

- Tecnologia para negócio:
<https://www.cbatech.com.br/?p=214883>
- NetBeans IDE:
<https://pt.wikipedia.org/wiki/NetBeans>
- Java JDK:
<https://www.oracle.com/br/java/technologies/javase-jdk11-downloads.html>
<https://www.devmedia.com.br/introducao-ao-java-jdk/28896>
- PostgreSQL:
<https://pt.wikipedia.org/wiki/PostgreSQL>
- Sistema Operacional:
[https://pt.wikipedia.org/wiki/Windows_10#Recursos do Sistema](https://pt.wikipedia.org/wiki/Windows_10#Recursos_do_Sistema)
- Especificações do Hardware:
<https://bityli.com/vm8Ds>
- Model-View-Controller:
<https://tableless.com.br/mvc-afinal-e-o-que/>
<https://www.treinaweb.com.br/blog/o-que-e-mvc>
- CRUD:
<https://devporai.com.br/o-que-e-crud-e-porque-voce-deveria-aprender-a-criar-um/>

7.8 Ficha de Atividades Práticas Supervisionadas.



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Pedro Lorencini Favarão TURMA: CC3Q18 RA: N549645
CURSO: Ciencia da Computação CAMPUS: Ribeirão Preto SEMESTRE: 3º TURNO: Noturno
CÓDIGO DA ATIVIDADE: 7689 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
01/05/2021	Auxílio código de login de usuário.	5	Pedro Lorencini Favarão		
01/05/2021	Estudo modelagem de dados.	10	Pedro Lorencini Favarão		
10/05/2021	Integração com postgres.	20	Pedro Lorencini Favarão		
15/05/2021	Correção de dados.	25	Pedro Lorencini Favarão		
18/05/2021	Integrado view com controller.	20	Pedro Lorencini Favarão		
23/05/2021	Verificação de integração com postgres.		Pedro Lorencini Favarão		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80 Horas

AValiação: _____
Aprovado ou Reprovado

NOTA: _____

DATA: 26 / 05 / 2021

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: João Victor Lorencini Favarão TURMA: CC3Q18 RA: N549E2
CURSO: Ciencia da Computação CAMPUS: Ribeirão Preto SEMESTRE: 3º TURNO: Noturno
CÓDIGO DA ATIVIDADE: 7689 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
01/05/2021	Montagem view menu de login.	5	João Victor Lorencini Favarão		
06/05/2021	Montagem view main.	10	João Victor Lorencini Favarão		
14/05/2021	Montagem view cadastro.	20	João Victor Lorencini Favarão		
17/05/2021	Montagem view consulta.	25	João Victor Lorencini Favarão		
21/05/2021	Verificação de código.	20	João Victor Lorencini Favarão		
22/05/2021	Integrado view com controller.		João Victor Lorencini Favarão		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80 Horas

AValiação: _____
Aprovado ou Reprovado

NOTA: _____

DATA: 26 / 05 / 2021

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Vitor Bonfadini TURMA: CC3P18 RA: F197A16
CURSO: Ciência da Computação CAMPUS: Ribeirão Preto SEMESTRE: 3º TURNO: Noturno
CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
20/04/2021	Estudo mercado de aplicativo.	15	Vitor Bonfadini		
01/05/2021	Montagem exemplo de relatório.	10	Vitor Bonfadini		
17/05/2021	Auxílio Objetivo do projeto.	17	Vitor Bonfadini		
18/05/2021	Correção de código.	15	Vitor Bonfadini		
18/05/2021	Montagem de relatório com linhas do código.	5	Vitor Bonfadini		
20/05/2021	Montagem parte teórica do projeto.	10	Vitor Bonfadini		
25/05/2021	Auxílio na montagem final do relatório.	10	Vitor Bonfadini		

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 82 Horas

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: 26 / 05 / 2021

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Victor Thauan de Andrade TURMA: CC3Q18 RA: N634CG0
CURSO: Ciência da Computação CAMPUS: Ribeirão Preto SEMESTRE: 3º TURNO: Noturno
CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS (1)	ASSINATURA DO PROFESSOR
20/04/2021	Estudo mercado de aplicativo.	5	Victor Thauan de Andrade		
03/05/2021	Montagem do objetivo do trabalho.	10	Victor Thauan de Andrade		
10/05/2021	Atualizado objetivo e motivação.	20	Victor Thauan de Andrade		
18/05/2021	Montagem de introdução.	25	Victor Thauan de Andrade		
18/05/2021	Auxílio na parte view consulta.	20	Victor Thauan de Andrade		
23/05/2021	Verificação e integração com postgres.		Victor Thauan de Andrade		
25/05/2021	Montagem final do relatório.				

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: 80 Horas

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: 26 / 05 / 2021

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO