UNIFEOB
ESCOLA DE NEGÓCIOS

2023

PROJETO DE CONSULTORIA EMPRESARIAL

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

PROJETO INTEGRADO

SISTEMA EMPRESARIAL

SÃO JOÃO DA BOA VISTA, SP

ABRIL 2023

UNIFEOB

CENTRO UNIVERSITÁRIO DA FUNDAÇÃO DE ENSINO OCTÁVIO BASTOS

ESCOLA DE NEGÓCIOS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

GESTÃO DE TECNOLOGIA DA INFORMAÇÃO

PROJETO INTEGRADO

SISTEMA EMPRESARIAL

MÓDULO DESENVOLVIMENTO DESKTOP

Banco de Dados – Prof. Sidney Gitcoff Telles
Programação Orientada a Objeto – Prof. Sidney Gitcoff Telles
Projeto de Desenvolvimento Desktop – Prof. Sidney Gitcoff Telles

Estudantes:

Kallil Henrique da Silva, RA 1012023100441
Matheus Souza Rodrigues, RA 1012023100326
Caio Augusto Francisco,RA 1012023100404
Leonardo Gomes da Silva, RA 1012023100474
Raphael Gomes Penha, RA 1012023100477

SÃO JOÃO DA BOA VISTA, SP
ABRIL, 2023

SUMÁRIO

Lista de ilustrações

1 INTRODUÇÃO

Nos dias atuais, as empresas têm enfrentado cada vez mais desafios em um mercado cada vez mais competitivo. Para se destacar, é necessário adotar estratégias

eficientes que aumentem a produtividade, reduzam custos e melhorem a qualidade dos serviços prestados. Nesse sentido, a automação de vendas tem sido uma solução cada vez mais adotada por empresas de todos os setores, visando otimizar seus processos de vendas e melhorar seus resultados.

A automação de vendas é um conjunto de práticas e tecnologias que visam automatizar os processos de vendas de uma empresa, desde a captação de potenciais consumidores até a fidelização de clientes. Essa automação pode ser alcançada através do uso de softwares de sistemas de gestão de vendas e outras ferramentas disponíveis no mercado.

O objetivo da automação de vendas é aumentar a eficiência da equipe de vendas, reduzir erros, melhorar a qualidade do atendimento ao cliente e aumentar a produtividade, liberando tempo para que os vendedores possam se concentrar em atividades mais estratégicas, como fechar negócios e desenvolver relacionamentos duradouros com os clientes.

No entanto, a adoção de uma solução de automação de vendas não é simples e requer um planejamento cuidadoso para garantir que o sistema escolhido seja adequado às necessidades da empresa e que sua implementação seja bem-sucedida. Este trabalho tem como objetivo explorar o tema da automação de vendas e construir um software para gestão e organização de vendas, abordando sua importância, os benefícios e desafios de sua implementação e apresentando uma metodologia para a seleção, implementação e mensuração dos resultados de uma solução de automação de vendas.

2 DESCRIÇÃO DA EMPRESA

De acordo com Cleyton Izidoro (2016, p.2) “Negócios é intermediação, comércio e até mesmo uma empresa que promove a produção e desenvolvimento de algo e o comercializam”

O autor também ressalta que as empresas são organizações que atuam na produção e comercialização de bens e serviços, desta forma tudo o que consumimos é desenvolvido e negociado.

Somando o que foi descrito anteriormente o projeto teve seu fundamento apreciando tais argumentações. A empresa Hero’s Games de cnpj: 48.996.827/0001-23 com razão social de: 48.996.827 VIVIANE APARECIDA DE SOUZA, situada em: Rua Cefeu 372, Jd.Satélite, São José dos Campos/SP, que apresenta as seguintes atividades: Comercialização de produtos Novos/Usados relacionados a antiguidades e movimentos retrô. Foi estabelecida para a construção do trabalho.

Foi constatado com os proprietários da empresa diversas perturbações em decorrência de dificuldades ao gerenciar vendas e produtos, do mesmo modo que apresentava contrariedades em relação a estoques de produtos.

Segundo Chiavenato (2014, p.13), diz-se a gestão de vendas que

> [...] envolve o planejamento, a organização, a direção e o controle das atividades de vendas, incluindo recrutamento, seleção, treinamento, remuneração, previsão de vendas,definição de cotas e definição de zonas de vendas, na medida em que essas atividades se aplicam diretamente ao pessoal de vendas.

A solução apresentada aos proprietários foi de a construção de um software de gestão empresarial , com o fim de apresentar as seguintes soluções:

- Cadastro de produtos
- Gestão de produtos
- Cadastro de vendas
- Gestão de vendas
- Cadastro de estoque
- Gestão de estoque

De acordo com Chiavenato (2014), a gestão de vendas está dependente á área de marketing e tem como equivalência as seguintes áreas:

- Pesquisa de mercado
- Promoção e propaganda
- Gerência de produtos
- Distribuição e logística

3 PROJETO DE CONSULTORIA EMPRESARIAL

3.1 BANCO DE DADOS

Nessa parte do PI, a equipe precisa fazer todo o mapeamento da necessidade da empresa. Levantar a modelagem, depois o diagrama e por fim criar as tabelas do banco de dados proposto.

Banco de Dados é uma estrutura organizada de dados que armazena informações de forma persistente em um sistema de computador. É uma das tecnologias fundamentais em todos os sistemas de informação modernos, desde aplicativos de desktop até sistemas empresariais complexos e aplicativos web.

Os bancos de dados são usados para armazenar e gerenciar grandes quantidades de informações, tornando a recuperação e a análise de dados mais eficientes. A maioria dos bancos de dados modernos são baseados em um modelo relacional, onde as informações são armazenadas em tabelas, colunas e linhas.

Além disso, existem vários tipos de bancos de dados, como bancos de dados relacionais, bancos de dados orientados a objetos, bancos de dados NoSQL e bancos de dados em memória. Cada tipo tem suas próprias características e benefícios, dependendo das necessidades do projeto.

Para acessar e manipular os dados armazenados em um banco de dados, os desenvolvedores utilizam uma linguagem de consulta, como SQL (Structured Query Language). Essa linguagem permite que os usuários recuperem, atualizem e excluam informações armazenadas no banco de dados.

Em suma, os bancos de dados são essenciais para o armazenamento e gerenciamento de informações em sistemas de informação modernos. Eles oferecem uma maneira eficiente de armazenar e acessar dados, além de permitir que os desenvolvedores criem aplicativos escaláveis e confiáveis.

3.1.1 MER - MODELAGEM ENTIDADE RELACIONAMENTO

O modelo Entidade-Relacionamento (ER) é uma ferramenta importante na modelagem de dados utilizada em sistemas de banco de dados. Ele é amplamente utilizado por desenvolvedores e arquitetos de software para projetar a estrutura de um banco de dados

e estabelecer relações entre as entidades. A importância do modelo ER se deve a várias razões:

Facilidade de entendimento: O modelo ER é uma representação visual simples e intuitiva de como os dados se relacionam entre si. Ele permite que qualquer pessoa que entenda conceitos básicos de banco de dados possa compreender facilmente a estrutura de dados e as relações existentes entre as entidades.

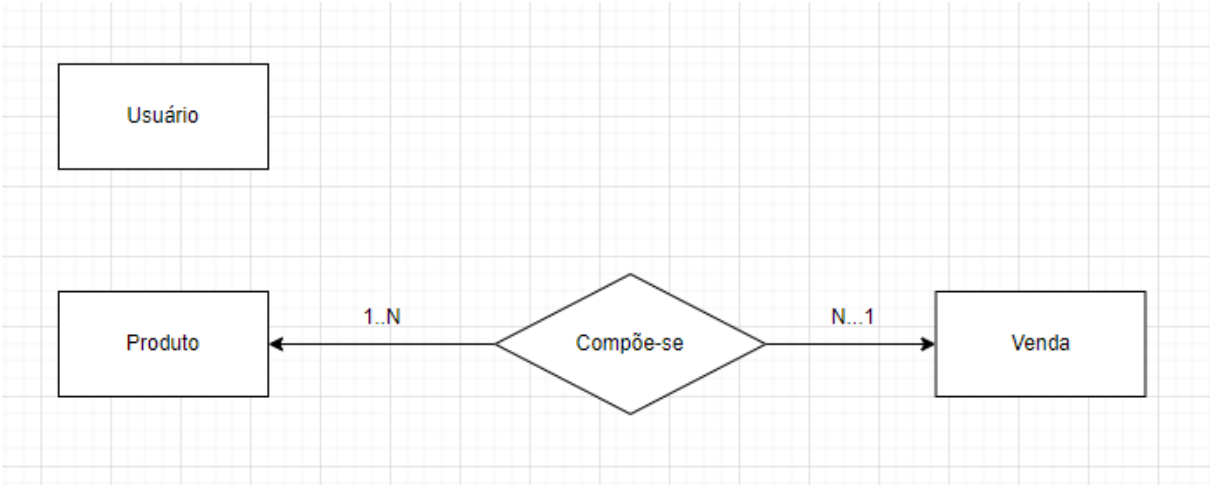
Identificação de problemas de integridade de dados: O modelo ER ajuda a identificar problemas de integridade de dados que podem ocorrer em um banco de dados. Isso inclui a garantia da consistência dos dados, evitando redundância de dados e a eliminação de problemas de atualização de dados.

Mapeamento de dados: O modelo ER ajuda a mapear os dados do mundo real para a estrutura de um banco de dados. Isso ajuda os desenvolvedores a entender como os dados se relacionam e como eles devem ser organizados em uma estrutura de banco de dados.

Comunicação e documentação: O modelo ER é uma ferramenta importante para documentar e comunicar a estrutura de um banco de dados. Isso ajuda a garantir que todos os membros da equipe envolvidos no desenvolvimento de um sistema possam entender a estrutura de dados e as relações entre as entidades.

Redução de tempo e custos: O modelo ER ajuda a reduzir o tempo e os custos envolvidos no desenvolvimento de um sistema. Isso ocorre porque o modelo ER pode identificar problemas de integridade de dados e outros problemas de design de banco de dados antes do início do desenvolvimento. Isso economiza tempo e recursos necessários para corrigir problemas mais tarde.

Figura 1 - Modelagem entidade relacionamento do banco de dados



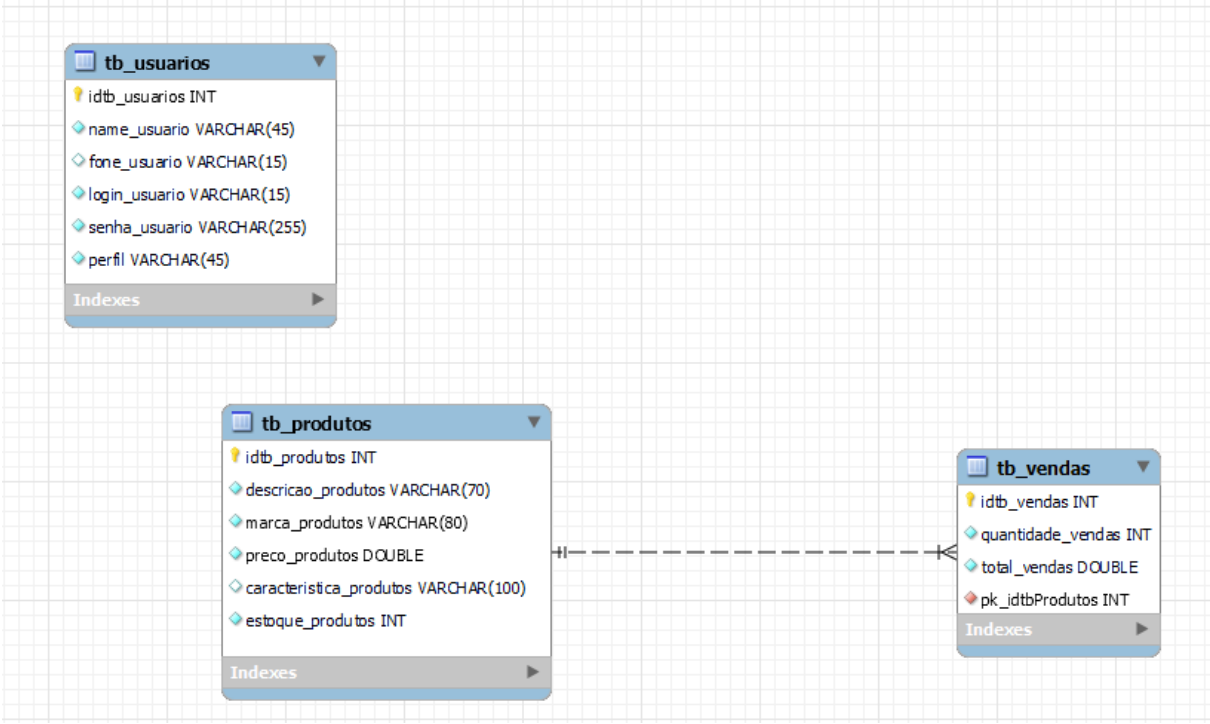
Fonte: Autoria própria

3.1.2 DER - DIAGRAMA ENTIDADE RELACIONAMENTO

O diagrama Entidade-Relacionamento (ER) é uma técnica de modelagem de dados que descreve as entidades e seus relacionamentos em um sistema. Ele é frequentemente usado em sistemas de banco de dados para descrever a estrutura de dados subjacente e para garantir que a informação esteja bem organizada.

Uma das principais vantagens do uso do diagrama ER é a sua capacidade de fornecer uma representação visual clara da estrutura de dados do sistema. Isso facilita a compreensão e a comunicação das informações, tanto para os desenvolvedores quanto para os usuários finais. Além disso, o diagrama ER ajuda a identificar erros na estrutura dos dados, o que pode economizar tempo e recursos na fase de desenvolvimento.

Figura 2 - Diagrama entidade relacionamento do banco de dados



Fonte: Autori

3.1.3 FÍSICO

Os estudantes devem criar o banco de dados físico em MySQL, com base nos itens 3.1.1 e 3.1.2. Tirar print dos comandos de criação e das tabelas do banco criado.

Figura 3 - Script construção da tabela de usuários

```
CREATE TABLE IF NOT EXISTS `tb_usuarios` (
  `idtb_usuarios` INT NOT NULL AUTO_INCREMENT,
  `name_usuario` VARCHAR(45) NOT NULL,
  `fone_usuario` VARCHAR(15) NOT NULL,
  `login_usuario` VARCHAR(15) NOT NULL,
  `senha_usuario` VARCHAR(255) NOT NULL,
  `perfil` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`idtb_usuarios`))
ENGINE = InnoDB;
```

Figura 4 - Script construção da tabela de produtos

```
CREATE TABLE IF NOT EXISTS `tb_produtos` (
  `idtb_produtos` INT NOT NULL AUTO_INCREMENT,
  `descricao_produtos` VARCHAR(70) NOT NULL,
  `marca_produtos` VARCHAR(80) NOT NULL,
  `preco_produtos` DOUBLE NOT NULL,
  `caracteristica_produtos` VARCHAR(100) NOT NULL,
  `estoque_produtos` INT NOT NULL,
  PRIMARY KEY (`idtb_produtos`))
ENGINE = InnoDB;
```

Figura 5 - Script construção da tabela de vendas

```
CREATE TABLE IF NOT EXISTS `tb_vendas` (
  `idtb_vendas` INT NOT NULL AUTO_INCREMENT,
  `quantidade_vendas` INT NOT NULL,
  `total_vendas` INT NOT NULL,
  `pk_idtbProdutos` INT NOT NULL,
  PRIMARY KEY (`idtb_vendas`),
  INDEX `fk_tb_vendas_tb_produtos_idx` (`pk_idtbProdutos` ASC),
  CONSTRAINT `fk_tb_vendas_tb_produtos`
    FOREIGN KEY (`pk_idtbProdutos`)
    REFERENCES `tb_produtos` (`idtb_produtos`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

3.2 PROGRAMAÇÃO ORIENTADA A OBJETOS

Nessa parte do PI, a equipe precisa desenvolver o sistema em linguagem Java, utilizando o NetBeans, deverão inserir aqui o diagrama de classe, os códigos e as imagens do sistema.

A programação orientada a objetos (POO) é um paradigma de programação que permite aos desenvolvedores criar softwares usando a abstração de objetos, que são representações de entidades do mundo real ou conceitos abstratos. Na POO, os objetos possuem atributos, que são características que descrevem seu estado, e métodos, que são comportamentos que os objetos podem executar.

A POO é baseada em quatro conceitos fundamentais: encapsulamento, herança, polimorfismo e abstração. O encapsulamento é a capacidade de esconder o estado interno do objeto e expor apenas as interfaces públicas. A herança é a capacidade de criar novas classes a partir de classes existentes, mantendo as características da classe original e adicionando novas funcionalidades. O polimorfismo permite que um objeto possa ser tratado como se fosse de um tipo diferente do seu tipo real. E a abstração é a capacidade de criar classes abstratas que definem comportamentos genéricos que podem ser implementados por classes concretas.

Uma das vantagens da POO é a modularidade do código. A divisão do código em objetos independentes permite que o desenvolvimento e a manutenção do software sejam mais fáceis e menos propensos a erros. Além disso, a POO promove a reutilização do código, uma vez que as classes podem ser usadas em diferentes partes do programa.

Outra vantagem da POO é a facilidade de desenvolvimento em equipe. Os desenvolvedores podem trabalhar em diferentes partes do software, cada um criando seus próprios objetos, sem interferir no trabalho dos outros. Além disso, a POO permite a criação de bibliotecas de objetos reutilizáveis que podem ser compartilhados entre diferentes projetos.

A POO é amplamente utilizada em linguagens de programação modernas, como Java, C + +, Python e Ruby. Essas linguagens possuem recursos avançados de POO que permitem a criação de programas complexos e escaláveis. No entanto, a POO não é adequada para todos os tipos de aplicativos. Em alguns casos, a programação estruturada pode ser mais adequada, especialmente para programas pequenos e simples.

3.2.1 DIAGRAMA DE CLASSES

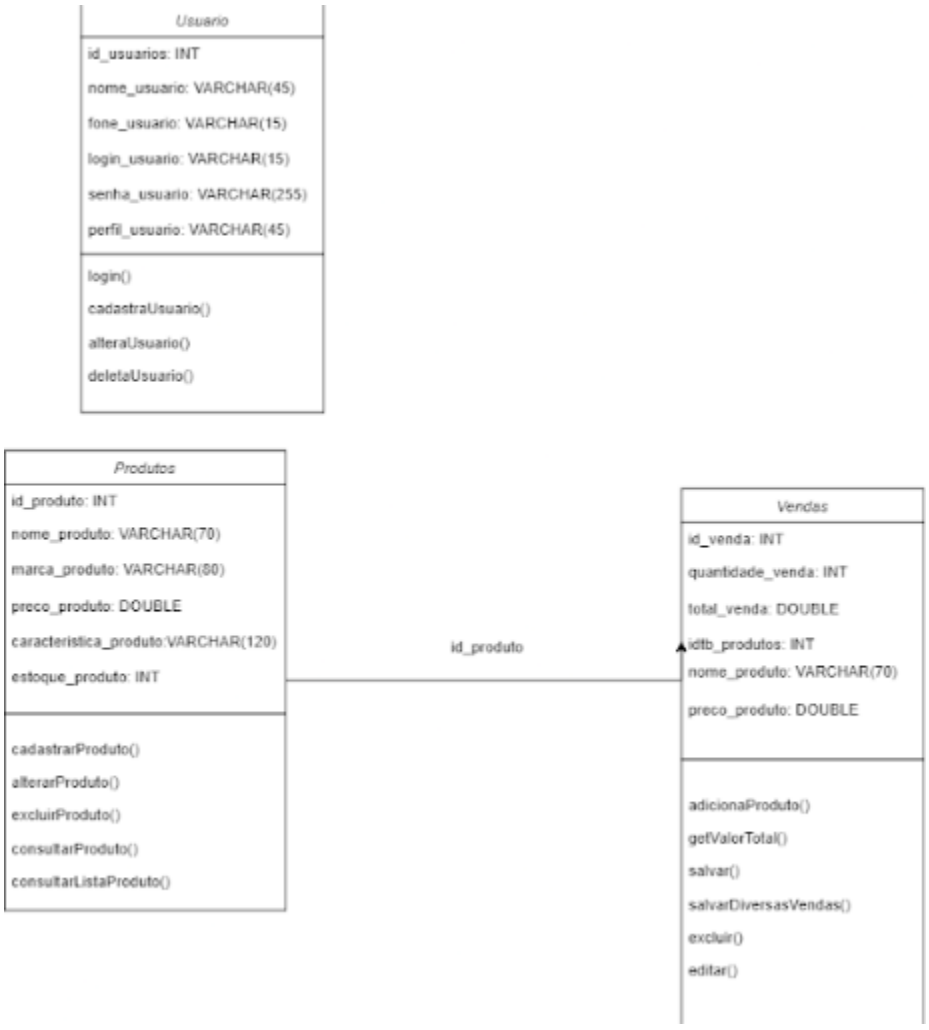
O diagrama de classes é uma ferramenta de modelagem de software que desempenha um papel crítico no desenvolvimento de sistemas orientados a objetos. Ele é uma representação gráfica da estrutura do sistema, incluindo as classes, seus atributos, métodos e relacionamentos.

Uma das principais vantagens do uso do diagrama de classes é a capacidade de visualizar a estrutura do sistema de uma forma intuitiva e clara. Isso permite que os integrantes compreendam rapidamente a arquitetura geral do sistema, identifiquem padrões de design e planejem a implementação das classes e seus relacionamentos. Além disso, o diagrama de classes facilita a comunicação entre os membros da equipe de desenvolvimento, permitindo que eles discutam e compartilhem ideias de forma mais eficiente.

Outra vantagem do diagrama de classes é a sua capacidade de ajudar a prever e evitar possíveis problemas no desenvolvimento do sistema. Ao identificar e mapear os relacionamentos entre as classes e seus métodos, é possível detectar problemas de compatibilidade ou de lógica antecipadamente. Isso economiza tempo e recursos ao evitar a necessidade de refatorar todo o sistema após a descoberta de um problema durante a fase de implementação.

O diagrama de classes também é importante porque pode ser usado para documentar o sistema. À medida que o sistema cresce e se desenvolve, o diagrama de classes pode ser atualizado para refletir as alterações na estrutura. Isso torna mais fácil para os desenvolvedores entenderem a evolução do sistema ao longo do tempo, e também para novos desenvolvedores se familiarizarem com o sistema quando se juntarem à equipe.

Figura 6 - Diagrama de classes do projeto



3.2.2 CÓDIGOS DO SISTEMA

Os estudantes devem desenvolver o sistema e utilizar as classes criadas para o perfeito funcionamento do sistema. Coloquem aqui alguns trechos de códigos do sistema. O trecho do arquivo principal main.java, um arquivo da model com instruções sql e um arquivo da controler que chama essa model.

Figura 7 - Código arquivo model produtos

```java
package model;
/**
*
* @author Matheus
*/
public class ModelProdutos {

    private int idtb_vendas;
    private String nome_produto;
    private String marca_produto;
    private double preco_produtos;
    private String caracteristica_produtos;
    private int estoque_produtos;

    /**
    * Construtor
    */
    public ModelProdutos(){}

    /**
    * seta o valor de idtb_vendas
    * @param pIdtb_vendas
    */
    public void setIdtb_vendas(int pIdtb_vendas){
        this.idtb_vendas = pIdtb_vendas;
    }
    /**
    * @return pk_idtb_vendas
    */
    public int getIdtb_vendas(){
        return this.idtb_vendas;
    }
```

Figura 8 - Código arquivo model vendas

```java
package model;
/**
*
* @author Matheus
*/
public class ModelVendas {

    private int idtb_vendas;
    private int quantidade_vendas;
    private double total_vendas;
    private int idtb_produtos;
    private String nome_produtos;
    private double preco_produtos;
    /**
    * Construtor
    */
    public ModelVendas(){}

    /**
    * seta o valor de idtb_vendas
    * @param pIdtb_vendas
    */
    public void setIdtb_vendas(int pIdtb_vendas){
        this.idtb_vendas = pIdtb_vendas;
    }
    /**
    * @return pk_idtb_vendas
    */
    public int getIdtb_vendas(){
        return this.idtb_vendas;
    }
```

Figura 9 - Código arquivo dao produtos

```
package br.com.PiHerosGames.Dao;

import model.ModelProdutos;
import br.com.PiHerosGames.conexao.ConexaoMySql;
import java.util.ArrayList;
/**
*
* @author Matheus
*/
public class DAOProdutos extends ConexaoMySql {

    /**
    * grava Produtos
    * @param pModelProdutos
    * @return int
    */
    public int salvarProdutosDAO(ModelProdutos pModelProdutos){
        try {
            this.conectar();
            return this.insertSQL(
                "INSERT INTO tb_produtos ("
                    + "descricao_produtos,"
                    + "marca_produtos,"
                    + "preco_produtos,"
                    + "caracteristica_produtos,"
                    + "estoque_produtos"
                + ") VALUES ("
                    + "'" + pModelProdutos.getNome_produto() + "',"
                    + "'" + pModelProdutos.getMarca_produto() + "',"
                    + "'" + pModelProdutos.getPreco_produtos() + "',"
                    + "'" + pModelProdutos.getCaracteristica_produtos() + "',"
                    + "'" + pModelProdutos.getEstoque_produtos() + "'"
```

Figura 10 - Código arquivo dao vendas

```java
package br.com.PiHerosGames.Dao;

import model.ModelVendas;
import model.ModelProdutos;

import br.com.PiHerosGames.conexao.ModuloConexao;
import br.com.PiHerosGames.conexao.ConexaoMySql;
import java.util.ArrayList;
import java.sql.*;

/**
 *
 * @author Matheus
 */
public class DAOVendas extends ConexaoMySql {
    Connection conn;
    PreparedStatement pstm;
    ResultSet rs;


    public boolean salvarDiversasVendasDao(ArrayList<ModelVendas> plistaVendas){
        String sql = "INSERT INTO tb_vendas (quantidade_vendas,total_vendas,pk_idtbProdutos) VALUES(?,?,?)";
        conn = new ModuloConexao().conector();
        try {
            int cont = plistaVendas.size();
            for (int i = 0; i < cont; i++) {
                pstm = conn.prepareStatement( string: sql);
                pstm.setInt( i: 1,  il: plistaVendas.get( index: i).getQuantidade_vendas());
                pstm.setDouble( i: 2, d: plistaVendas.get( index: i).getTotal_vendas());
                pstm.setInt( i: 3,  il: plistaVendas.get( index: i).getIdtb_produtos());
                pstm.execute();
            }
```

Figura 11 - Código controller produtos

```java
package controller;

import model.ModelProdutos;
import br.com.PiHerosGames.Dao.DAOProdutos;
import java.util.ArrayList;

/**
*
* @author Matheus
*/
public class ControllerProdutos {

    private DAOProdutos daoProdutos = new DAOProdutos();

    /**
    * grava Produtos
    * @param pModelProdutos
    * @return int
    */
    public int salvarProdutosController(ModelProdutos pModelProdutos){
        return this.daoProdutos.salvarProdutosDAO(pModelProdutos);
    }

    /**
    * recupera Produtos
    * @param pIdtb_vendas
    * @return ModelProdutos
    */
    public ModelProdutos getProdutosController(int pIdtb_vendas){
        return this.daoProdutos.getProdutosDAO(pIdtb_vendas);
    }
```

Figura 12 - Código controller vendas

```
package controller;

import model.ModelVendas;
import br.com.PiHerosGames.Dao.DAOVendas;
import java.util.ArrayList;

/**
*
* @author Matheus
*/
public class ControllerVendas {

    private DAOVendas daoVendas = new DAOVendas();

    /**
    * grava Vendas
    * @param plistaVendas
    * @return int
    */
    public boolean salvarDiversasVendasDao(ArrayList<ModelVendas> plistaVendas){
        return this.daoVendas.salvarDiversasVendasDao(plistaVendas);
    }

     public ArrayList<ModelVendas> getListaVendasLeftJoinController(){
         return this.daoVendas.getListaVendasLeftJoin();
     }
    /**
    * recupera Vendas
    * @param pIdtb_vendas
    * @return ModelVendas
    */
    public ModelVendas getVendasController(int pIdtb_vendas){
```

Figura 13 - Código conexão com banco de dados

```
private boolean status = false;
private String mensagem = "";   //variavel que vai informar o status da conexao
private Connection con = null;  //variavel para conexao
private Statement statement;
private ResultSet resultSet;

private String servidor = "localhost";
private String nomeDoBanco = "projetointegrado";
private String usuario = "root";
private String senha = "";

public ConexaoMySql(){}

public ConexaoMySql(String pServidor, String pNomeDoBanco, String pUsuario, String pSenha){
    this.servidor = pServidor;
    this.nomeDoBanco = pNomeDoBanco;
    this.usuario = pUsuario;
    this.senha = pSenha;
}

/**
 * Abre uma conexao com o banco
 * @return Connection
 */
public Connection conectar(){
    try {
        //Driver do PostgreSQL
        Class.forName( className: "com.mysql.jdbc.Driver").newInstance();

        //local do banco, nome do banco, usuario e senha
        String url = "jdbc:mysql://" + servidor + "/" + nomeDoBanco;
```

Figura 14 - Código método de login

```
public void Logar() {
    String sqlConsulta = "SELECT * FROM tb_usuarios WHERE login_usuario=? AND senha_usuario=?";
    try {
        // Preparam a consulta ao banco
        pst = conexao.prepareStatement( string: sqlConsulta);
        pst.setString( i: 1,  string: txt_UsuarioTxt.getText());
        pst.setString( i: 2,  string: txt_Senha.getText());
        //Executa a query
        rs = pst.executeQuery();
        // Estrutura para verificar se existem dados
        if (rs.next()) {
            PaginaInicial principal = new PaginaInicial();
            // Pega nome do usuario
            String perfil = rs.getString( i: 2);
            //Exibe nome do usuario
            PaginaInicial.lbl_usuario.setText( text: perfil);
            principal.setVisible( b: true);
            this.dispose();
            conexao.close();
        } else {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Usuário e/ou senha inválido(s)");
        }

    } catch (Exception e) {
        JOptionPane.showMessageDialog( parentComponent: null,  message: e);
    }
}
```

3.2.3 IMAGENS DO SISTEMA

Figura 15 - Tela de login

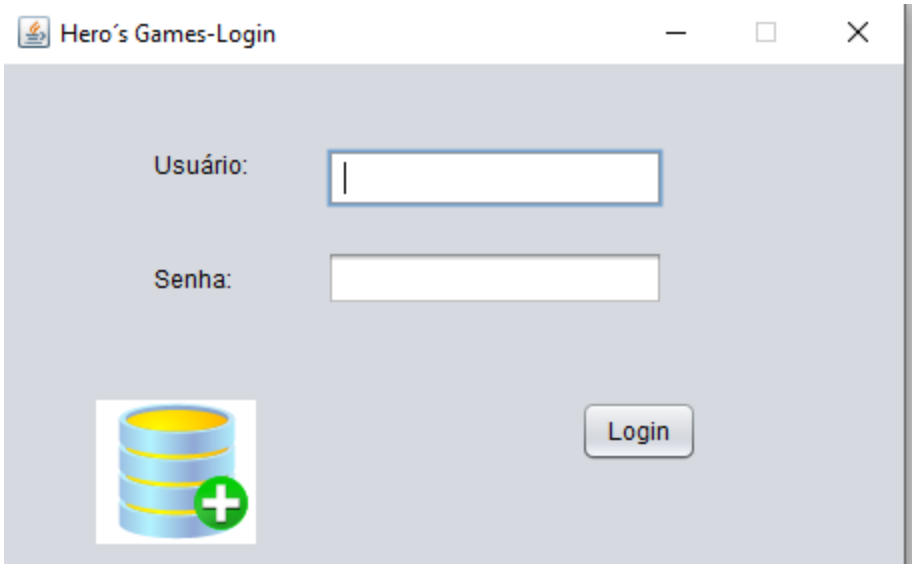


Figura 16 - Tela principal

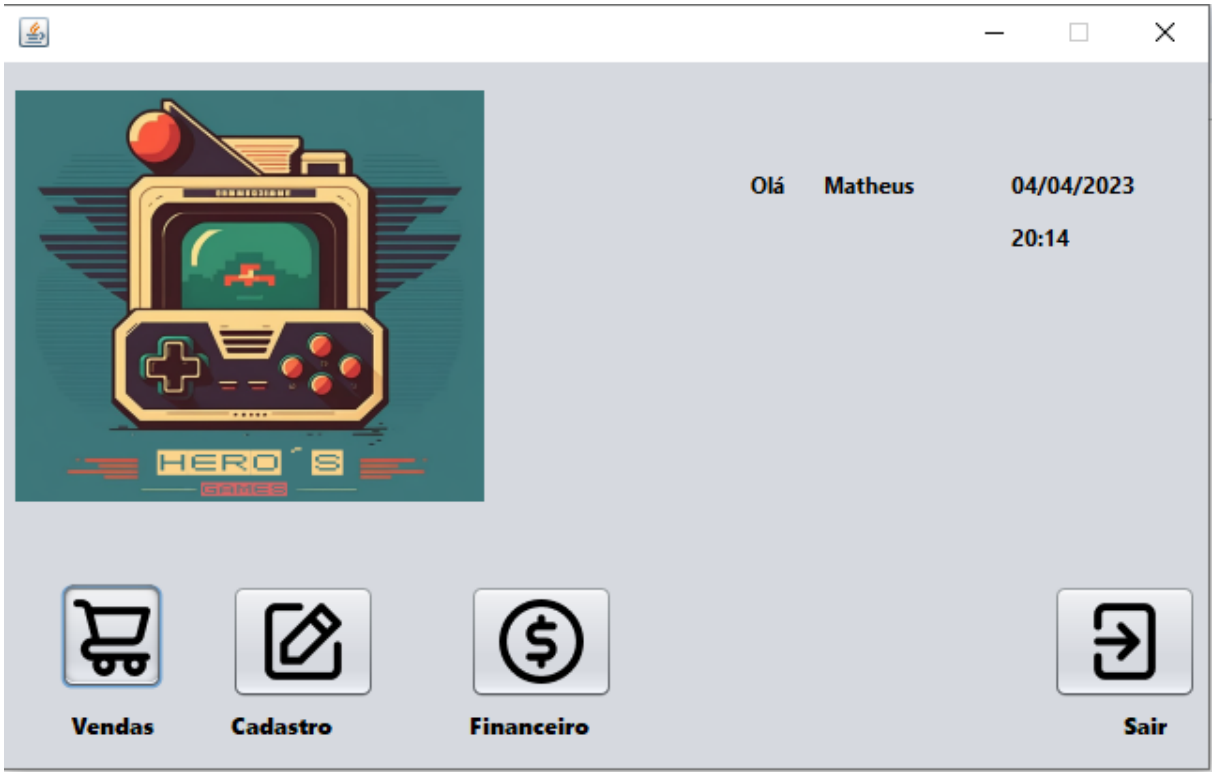


Figura 17 - Tela de cadastro de produtos

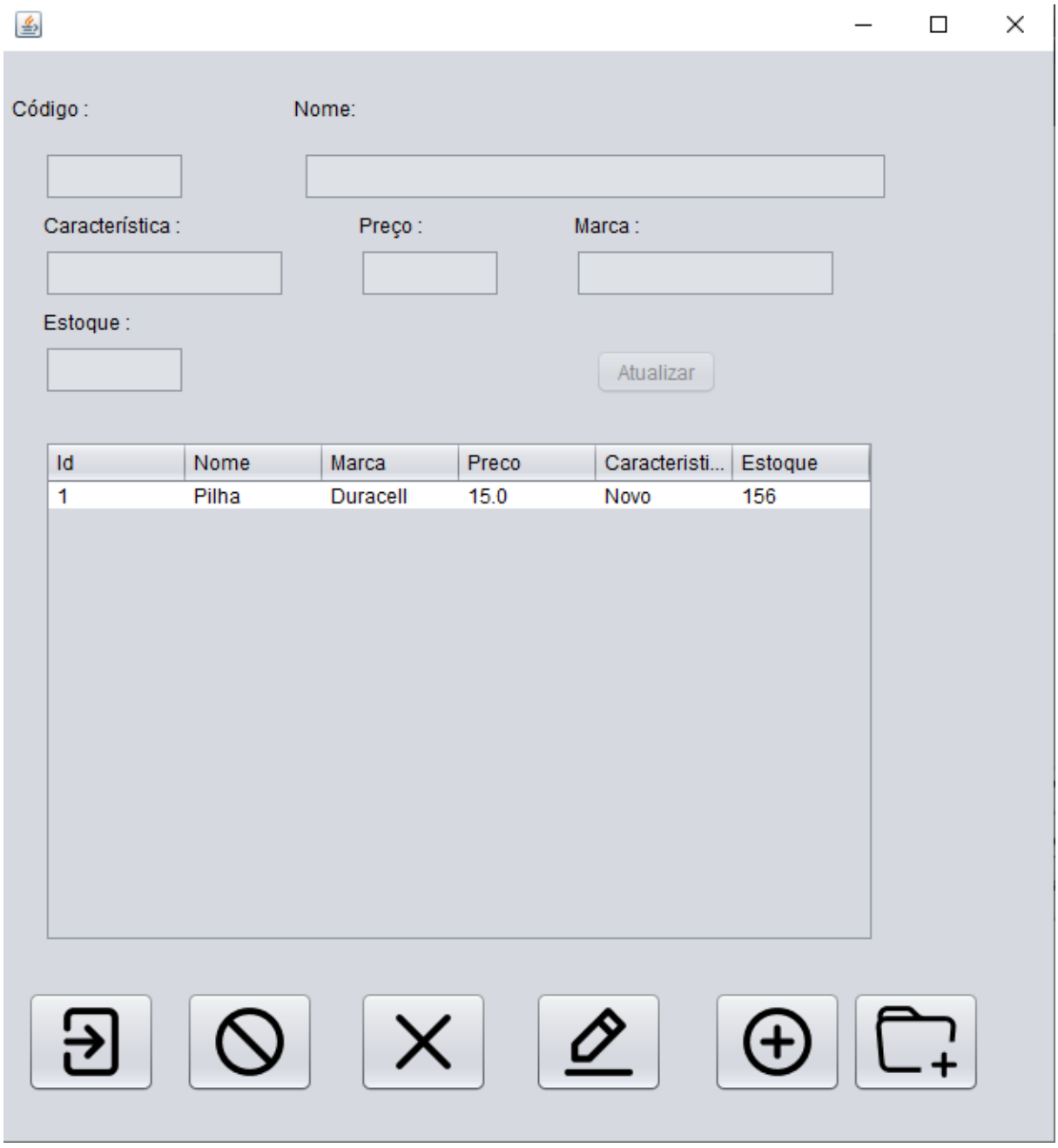


Figura 18 - Tela de vendas

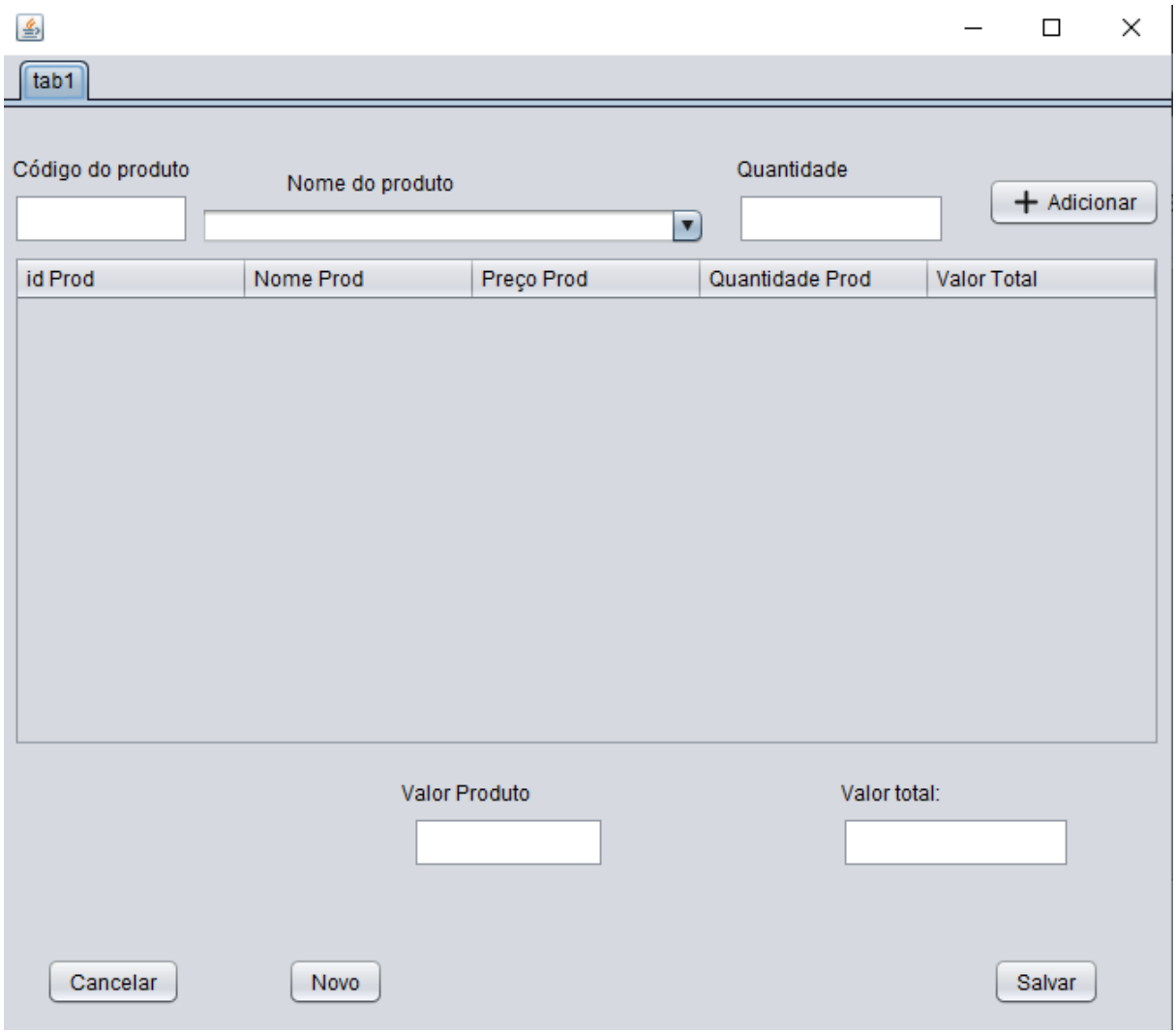
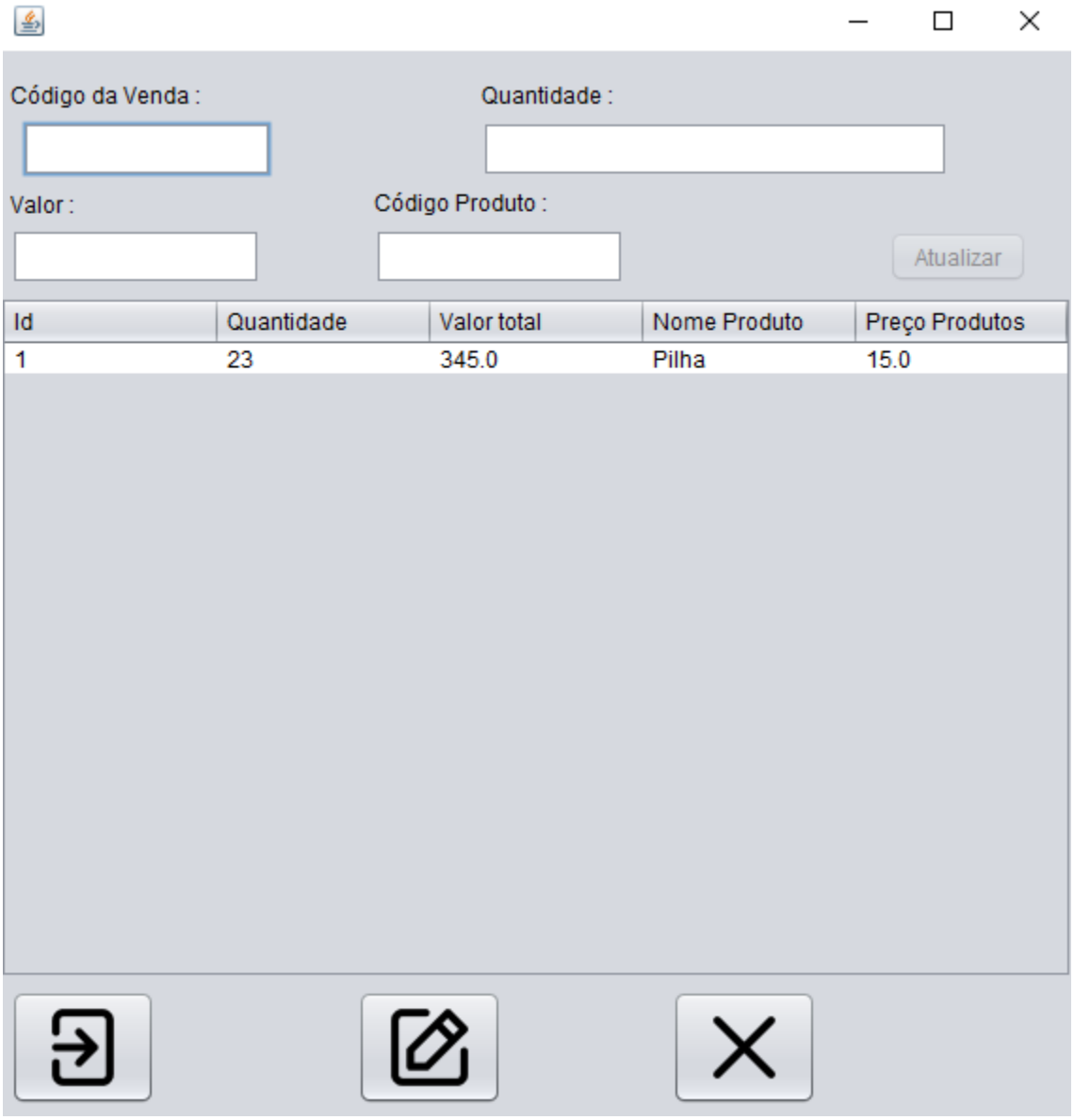


Figura 19 - Tela de financeiro/consulta de vendas



3.3 CONTEÚDO DA FORMAÇÃO PARA A VIDA: CRIANDO O NOVO

A Formação para a Vida é um dos eixos do Projeto Pedagógico de Formação por Competências da UNIFEOB.

Esta parte do Projeto Integrado está diretamente relacionada com a extensão universitária, ou seja, o objetivo é que seja aplicável e que tenha real utilidade para a sociedade, de um modo geral.

3.3.1 CRIANDO O NOVO

O tema Criando o novo proporcionado pela instituição, foi de extrema importância para a construção do projeto, assegurando diversas orientações para um olhar mais prudente aos temas apresentados pelos integrantes.

- **Tópico 1:** Design Thinking nos estudos e na profissão

O projeto se relaciona com o design thinking de amplas maneiras, portanto a principal delas é o uso da criatividade no desenvolvimento da aplicação. A criatividade esteve presente em todos os aspectos de construção, desde a construção do logotipo até as nomeações de variáveis.

- **Tópico 2:** Há mil maneiras de pensar

Este tópico tem relação com o projeto pois não foi denominado um "Lider", que ditou o que cada um dos integrantes teria que fazer ou não, os integrantes do grupo se juntaram em reuniões diárias e debateram suas ideias, até conquistar um senso comum.

- **Tópico 3:** Criando asas

Neste tópico foi realizado diversos questionamentos ao usuário, para detectar quais eram seus pontos em relação ao projeto. Pois sempre é necessário a idealização do design centrado ao usuário.

- **Tópico 4:** Com vocês: O duplo diamante!

Como todos os outros tópicos este foi de suma importância para a construção do projeto, principalmente a parte prática

O diagrama proporcionou um olhar mais resoluto para o tema do projeto, seguindo seus princípios e processos de design thinking, assim assegurando ao projeto um início, meio e fim.

3.3.2 ESTUDANTES NA PRÁTICA

Foi designado pelos integrantes do grupo a realização de uma imagem motivacional para manifestar uma empatia, e ajudar as pessoas a lidar com dificuldades em relação a mudanças abruptas em seu cotidiano.

A imagem constata o que foi descrito anteriormente, as correntes devem ser rompidas para uma melhor forma de viver.

O principal objetivo da imagem é expressar que a pessoa observar a imagem deve superar as adversidades e alcançar seus sonhos.

4 CONCLUSÃO

Em conclusão, a automação de vendas é uma estratégia importante para empresas que desejam aumentar sua produtividade, reduzir custos e melhorar a qualidade do atendimento ao cliente. A implementação de uma solução de automação de vendas envolve etapas como a identificação dos processos que podem ser automatizados, a escolha da solução adequada para as necessidades da empresa, a capacitação da equipe de vendas e a mensuração dos resultados obtidos.

Embora a implementação da automação de vendas possa apresentar alguns desafios, como a necessidade de integração com outros sistemas e a capacitação da equipe de vendas, os benefícios da automação de vendas superam esses desafios. Ao adotar uma solução de automação de vendas, as empresas podem melhorar a eficiência dos processos de vendas, reduzir custos, aumentar a produtividade da equipe de vendas e melhorar a qualidade do atendimento ao cliente.

Em resumo, a automação de vendas é uma estratégia fundamental para empresas que buscam se destacar em um mercado competitivo e aumentar seus resultados. É importante que as empresas considerem cuidadosamente a implementação da automação de vendas e sigam uma metodologia cuidadosa para garantir o sucesso do projeto. Com a implementação bem-sucedida da automação de vendas, as empresas podem alcançar seus objetivos de vendas e melhorar sua posição no mercado.

O projeto desenvolvido foi apresentado aos proprietários do comércio, assim obtendo aprovação e utilização do mesmo, entretanto é possível situar algumas melhorias que pode ser implementadas adiante, como:

- Integração de um código de barras para acelerar o processo de venda
- Um sistema que possa gerar o código de barras a partir do código do produto
- Impressão do código
- Interface PDV
- Atalhos para facilitar processo de vendas.

Todavia é possível concluir que o projeto foi altamente capacitado em seus testes, cumprindo seus compromissos com os usuários.

REFERÊNCIAS

CHIAVENATO, I. Gestão de vendas: uma abordagem introdutória: transformando o profissional de vendas em um gestor. 3. ed. Barueri: Manole, 2014

CLEYTON IZIDORO, Administração de vendas. Person,2016

ANEXOS

Link para projeto no github, para melhor compreensão de como funciona o software:

https://github.com/MatheusSouzaRodrigues235/Projeto-Integrado-Ads-1Trimestre/tree/master

RELATÓRIO FINAL DAS ATIVIDADES DE EXTENSÃO

1. IDENTIDADE DA ATIVIDADE
RELATÓRIO:
CURSO: Análise e Desenvolvimento de Sistemas e Gestão de Tecnologia da Informação
MÓDULO: Desenvolvimento Desktop
PROFESSOR RESPONSÁVEL: Sidney Gitcoff Telles
ESTUDANTE:
PERÍODO DE REALIZAÇÃO: 02/2023 a 04/2023

2. DESENVOLVIMENTO
Contextualização
Desafio
Cronograma das Ações
Síntese das Ações
a. Aspectos positivos
b. Dificuldades encontradas
c. Resultados atingidos

d. Sugestões / Outras observações

3. EQUIPE DOS ESTUDANTES NO PROJETO

RA	NOME
RA	NOME
RA	NOME
RA	NOME
RA	NOME
RA	NOME

Curso de Análise e Desenvolvimento de Sistemas e Gestão de Tecnologia da Informação

Módulo Desenvolvimento Desktop

Cronograma de Validação - Projeto Integrado

Unidade Estudo	**Participação no Projeto**	**ata da Validação**
Banco de Dados	Construção do banco de dados com MER, DER E Físico.	9/03
Programação Orientada a Objetos	Desenvolvimento as telas e da parte lógica do sistema que conectará com o banco de dados	0/03
Descrição do Projeto: criar um sistema, um módulo reduzido, que seja utilizado em qualquer estabelecimento comercial ou empresarial. Esse sistema deverá contemplar atividades básicas da empresa, como controle de produtos, entrada e saída, controle de vendas, módulos menores que possam ser criados e executados neste trimestre.		