Fatec São José do Rio Preto

Curso Superior de Tecnologia em Informática para Negócios

Projeto de Extensão Comunitária

(Curricularização)

Desenvolvimento de um MÍNIMO PRODUTO VIÁVEL (MPV) de aplicação Web para uma organização

Integrantes do Projeto:

Artur Nonato Beti Pedro

São José do Rio Preto/SP

2º Semestre de 2024

Sumário

1. Características do Projeto de Extensão

Título	Desenvolvimento de um MPV de aplicação Web para uma organização (4º Semestre).
Temática	☐Programas ☒Projetos ☐Cursos e Oficinas ☐Eventos ☐Prestação de Serviços
Descrição	O projeto poderá ser baseado nos resultados de projetos de extensão anteriores. Será realizado o desenvolvimento de um MPV (Mínimo Produto Viável) de uma aplicação Web destinada a organização estudada.
Objetivos	▪ Analisar documentações de Engenharia de Software existentes para o entendimento das melhores oportunidades para a criação da aplicação Web; ▪ Agregar novas funções à base de dados persistente, por meio de *stored procedures*, *triggers*, entre outros recursos possíveis; ▪ Implementar uma planilha eletrônica que permita a análise de dados e informações de negócios da empresa estudada a partir de técnicas e/ou indicadores estatísticos (medidas de tendência central, regressão, probabilidade, inferência entre outras); ▪ Desenvolver um estudo de qualidade total e melhoria contínua da organização estudada a partir dos levantamentos dos processos de negócio e demandas existentes já detectadas previamente, apontando possíveis soluções e melhorias.
Carga horária	▪ Total: **106** horas/aula
Público-alvo	Empresas, ONGs, órgãos públicos e entidades sociais.
Ações/Etapas de execução	1. Programar, testar e efetuar demonstração em ambiente de testes uma aplicação Web; 2. Implementar recursos adicionais na base de dados para uma aplicação Web; 3. Usar métodos estatísticos para geração de dados e informação numa planilha de Excel; 4. Efetuar um estudo de qualidade total para os processos e demandas existentes numa organização.
Entregas	▪ 01 Script de banco de dados, com as instruções SQL e/ou descritivas dos recursos de BD criados; ▪ 01 Código-fonte da aplicação Web projetada em repositório online na internet (Github ou similar); ▪ 01 Documento de planilha eletrônica funcional com tabela aplicando técnicas de Estatística e explicações; ▪ 01 Relatório diagnóstico contendo as principais demandas,

	técnicas e soluções para a melhoria da qualidade dos processos corporativos considerados.
Instrumentos e procedimentos de avaliação	**Aluno** – trabalho em grupo, eficácia na realização das tarefas, entrega digital do resultado das tarefas em relatório padronizado de atividade de extensão. **Projeto** – resultados obtidos, publicação dos resultados em repositório online, envio de resultados para a entidade beneficiada na comunidade ou execução de evento para apresentação dos resultados e integração com representantes da comunidade externa beneficiada.
Componente(s)curricular(es) envolvidos	▪ Administração de Banco de Dados (20 horas/aula); ▪ Linguagens de Programação II (40 horas/aula); ▪ Estatística (24 horas/aula). ▪ Gestão da Qualidade e Ambiental (22 horas).
Formas de evidência	▪ Entrega de relatório de atividade de extensão com os resultados em cada disciplina.

2. Informações da Equipe de Trabalho

Nome do Grupo: SIGAS – Sistema de Gerenciamento Arte Sagrada

Nr.	Nome Completo do(s) Aluno(s)	Contatos	
01	Artur Nonato Beti Pedro	Email:	anonatopedro@gmail.com
		Whatsapp:	17 98823-8239
02		Email:	
		Whatsapp:	
03		Email:	
		Whatsapp:	
04		Email:	
		Whatsapp:	
05		Email:	
		Whatsapp:	

3. Etapa 1 – Programação, teste e demonstração em ambiente de testes de uma aplicação Web

Objetivo: Analisar documentações de Engenharia de Software existentes para o entendimento das melhores oportunidades para a criação da aplicação Web.

3.1 Interfaces gráficas desenvolvidas

3.2 Classes, atributos e métodos (pacote *model* com *annotations*)

```
package br.com.sigas.entities;

import java.time.LocalDate;
import java.time.LocalDateTime;
import jakarta.persistence.*;

@Entity
@Inheritance(strategy = InheritanceType.JOINED)
@DiscriminatorColumn(name = "tipo_pessoa", discriminatorType =
DiscriminatorType.STRING)
@Table(name = "pessoas")
public class Pessoas {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id_pessoa;

    @Column(name = "tipo_pessoa", insertable = false, updatable =
false)
    private String tipo_pessoa;

    @Column(name = "nome_pessoa", nullable = false, length = 100)
    private String nome_pessoa;

    @Column(name = "email_pessoa", length = 60)
    private String email_pessoa;

    @Column(name = "endereco_pessoa", length = 200)
    private String endereco_pessoa;

    @Column(name = "tel1_pessoa", nullable = false, length = 13)
    private String tel1_pessoa;
```

```
    @Column(name = "tel2_pessoa", length = 13)
    private String tel2_pessoa;

    @Column(name = "is_active", nullable = false)
    private Boolean isActive = true;

    @Column(name = "data_criacao", updatable = false)
    private LocalDateTime dataCriacao;

    @Column(name = "data_modificacao")
    private LocalDateTime dataModificacao;

    @Column(name = "data_nascimento", nullable = false)
    private LocalDate dataNascimento;

    @Column(name = "idade", insertable = false, updatable =
false)
    private Integer idade; // A idade será gerada automaticamente
pelo banco de dados

    @PrePersist
    protected void onCreate() {
        dataCriacao = LocalDateTime.now();
        dataModificacao = LocalDateTime.now();
    }

    @PreUpdate
    protected void onUpdate() {
        dataModificacao = LocalDateTime.now();
    }

    public Pessoas() {
    }

    public Pessoas(long id_pessoa, String tipo_pessoa, String
nome_pessoa, String email_pessoa, String endereco_pessoa,
            String tel1_pessoa, String tel2_pessoa, Boolean
isActive, LocalDateTime dataCriacao,
            LocalDateTime dataModificacao, LocalDate
dataNascimento, Integer idade) {
        this.id_pessoa = id_pessoa;
```

```
        this.tipo_pessoa = tipo_pessoa;
        this.nome_pessoa = nome_pessoa;
        this.email_pessoa = email_pessoa;
        this.endereco_pessoa = endereco_pessoa;
        this.tel1_pessoa = tel1_pessoa;
        this.tel2_pessoa = tel2_pessoa;
        this.isActive = isActive;
        this.dataCriacao = dataCriacao;
        this.dataModificacao = dataModificacao;
        this.dataNascimento = dataNascimento;
        this.idade = idade;
    }

    // Getters e Setters
    public long getId_pessoa() {
        return id_pessoa;
    }

    public void setId_pessoa(long id_pessoa) {
        this.id_pessoa = id_pessoa;
    }

    public String getTipo_pessoa() {
        return tipo_pessoa;
    }

    public void setTipo_pessoa(String tipo_pessoa) {
        this.tipo_pessoa = tipo_pessoa;
    }

    public String getNome_pessoa() {
        return nome_pessoa;
    }

    public void setNome_pessoa(String nome_pessoa) {
        this.nome_pessoa = nome_pessoa;
    }

    public String getEmail_pessoa() {
        return email_pessoa;
    }
```

```
    public void setEmail_pessoa(String email_pessoa) {
        this.email_pessoa = email_pessoa;
    }

    public String getEndereco_pessoa() {
        return endereco_pessoa;
    }

    public void setEndereco_pessoa(String endereco_pessoa) {
        this.endereco_pessoa = endereco_pessoa;
    }

    public String getTel1_pessoa() {
        return tel1_pessoa;
    }

    public void setTel1_pessoa(String tel1_pessoa) {
        this.tel1_pessoa = tel1_pessoa;
    }

    public String getTel2_pessoa() {
        return tel2_pessoa;
    }

    public void setTel2_pessoa(String tel2_pessoa) {
        this.tel2_pessoa = tel2_pessoa;
    }

    public Boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(Boolean isActive) {
        this.isActive = isActive;
    }

    public LocalDateTime getDataCriacao() {
        return dataCriacao;
    }

    public void setDataCriacao(LocalDateTime dataCriacao) {
        this.dataCriacao = dataCriacao;
```

```
    }

    public LocalDateTime getDataModificacao() {
        return dataModificacao;
    }

    public void setDataModificacao(LocalDateTime dataModificacao)
{
        this.dataModificacao = dataModificacao;
    }

    public LocalDate getDataNascimento() {
        return dataNascimento;
    }

    public void setDataNascimento(LocalDate dataNascimento) {
        this.dataNascimento = dataNascimento;
    }

    public Integer getIdade() {
        return idade;
    }
}
```

```
package br.com.sigas.entities;

import java.time.LocalDate;
import java.time.LocalDateTime;

import com.fasterxml.jackson.annotation.JsonBackReference;

import jakarta.persistence.*;

@Entity
@DiscriminatorValue("F")
public class PessoasFisicas extends Pessoas {

    @Column(name = "cpf", nullable = false, length = 14, unique =
true)
    private String cpf;
```

```
    @ManyToOne
    @JoinColumn(name = "id_pessoa", referencedColumnName =
"id_pessoa", insertable = false, updatable = false)
    @JsonBackReference
    private Pessoas pessoa;

    public PessoasFisicas() {
    }



    public PessoasFisicas(long id_pessoa, String tipo_pessoa,
String nome_pessoa, String email_pessoa,
            String endereco_pessoa, String tel1_pessoa, String
tel2_pessoa, Boolean isActive, LocalDateTime dataCriacao,
            LocalDateTime dataModificacao, LocalDate
dataNascimento, Integer idade, String cpf, Pessoas pessoa) {
        super(id_pessoa, tipo_pessoa, nome_pessoa, email_pessoa,
endereco_pessoa, tel1_pessoa, tel2_pessoa, isActive,
                dataCriacao, dataModificacao, dataNascimento,
idade);
        this.cpf = cpf;
        this.pessoa = pessoa;
    }



    public PessoasFisicas(String cpf, Pessoas pessoa) {
        this.cpf = cpf;
        this.pessoa = pessoa;
    }



    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }
```

```
    public Pessoas getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoas pessoa) {
        this.pessoa = pessoa;
    }

}
```

```
package br.com.sigas.entities;

import java.time.LocalDate;
import java.time.LocalDateTime;

import com.fasterxml.jackson.annotation.JsonBackReference;

import jakarta.persistence.*;

@Entity
@DiscriminatorValue("J")
@Table(name = "pessoas_juridicas")
public class PessoasJuridicas extends Pessoas {

    @Column(name = "cnpj", nullable = false, length = 18, unique
= true)
    private String cnpj;

    @Column(name = "razao_social", nullable = false, length =
100)
    private String razao_social;

    @ManyToOne
    @JoinColumn(name = "id_pessoa", referencedColumnName =
"id_pessoa", insertable = false, updatable = false)
    @JsonBackReference
    private Pessoas pessoa;

    public PessoasJuridicas() {
    }
```

```
    public PessoasJuridicas(long id_pessoa, String tipo_pessoa,
String nome_pessoa, String email_pessoa,
            String endereco_pessoa, String tel1_pessoa, String
tel2_pessoa, Boolean isActive, LocalDateTime dataCriacao,
            LocalDateTime dataModificacao, LocalDate
dataNascimento, Integer idade, String cnpj, String razao_social,
            Pessoas pessoa) {
        super(id_pessoa, tipo_pessoa, nome_pessoa, email_pessoa,
endereco_pessoa, tel1_pessoa, tel2_pessoa, isActive,
                dataCriacao, dataModificacao, dataNascimento,
idade);
        this.cnpj = cnpj;
        this.razao_social = razao_social;
        this.pessoa = pessoa;
    }



    public PessoasJuridicas(String cnpj, String razao_social,
Pessoas pessoa) {
        this.cnpj = cnpj;
        this.razao_social = razao_social;
        this.pessoa = pessoa;
    }



    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }

    public String getRazao_social() {
        return razao_social;
    }
```

```
    public void setRazao_social(String razao_social) {
        this.razao_social = razao_social;
    }

    public Pessoas getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoas pessoa) {
        this.pessoa = pessoa;
    }

}
```

```
package br.com.sigas.entities;

import java.time.LocalDateTime;
import java.util.List;

import jakarta.persistence.*;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity
@Table(name = "categorias")
public class Categorias {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id_categoria;

    @Column(name = "nome_categoria", nullable = false, length =
100)
    private String nome_categoria;

    @Column(name = "isActive", nullable = false)
    private boolean isActive;

    @OneToMany(mappedBy = "categoria")
```

```
    @JsonIgnoreProperties("categoria")
    private List<Produtos> produtos;

    @Column(name = "data_criacao", nullable = false, updatable =
false)
    private LocalDateTime dataCriacao;

    @Column(name = "data_modificacao")
    private LocalDateTime dataModificacao;

    @PrePersist
    protected void onCreate() {
        dataCriacao = LocalDateTime.now();
        dataModificacao = LocalDateTime.now();
    }

    @PreUpdate
    protected void onUpdate() {
        dataModificacao = LocalDateTime.now();
    }

    public Categorias() {
    }

    public Categorias(Long id_categoria, String nome_categoria,
boolean isActive, List<Produtos> produtos,
            LocalDateTime dataCriacao, LocalDateTime
dataModificacao) {
        this.id_categoria = id_categoria;
        this.nome_categoria = nome_categoria;
        this.isActive = isActive;
        this.produtos = produtos;
        this.dataCriacao = dataCriacao;
        this.dataModificacao = dataModificacao;
    }

    public Long getId_categoria() {
        return id_categoria;
    }

    public void setId_categoria(Long id_categoria) {
        this.id_categoria = id_categoria;
```

```
    }

    public String getNome_categoria() {
        return nome_categoria;
    }

    public void setNome_categoria(String nome_categoria) {
        this.nome_categoria = nome_categoria;
    }

    public boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(boolean isActive) {
        this.isActive = isActive;
    }

    public List<Produtos> getProdutos() {
        return produtos;
    }

    public void setProdutos(List<Produtos> produtos) {
        this.produtos = produtos;
    }

    public LocalDateTime getDataCriacao() {
        return dataCriacao;
    }

    public void setDataCriacao(LocalDateTime dataCriacao) {
        this.dataCriacao = dataCriacao;
    }

    public LocalDateTime getDataModificacao() {
        return dataModificacao;
    }

    public void setDataModificacao(LocalDateTime dataModificacao)
{
        this.dataModificacao = dataModificacao;
    }
```

```
}
```

```
package br.com.sigas.entities;

import java.math.BigDecimal;
import java.time.LocalDateTime;

import com.fasterxml.jackson.annotation.JsonBackReference;

import jakarta.persistence.*;

@Entity
@Table(name = "Produtos")
public class Produtos {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id_produto;

    @Column(name = "nome_produto", nullable = false, length = 50)
    private String nome_produto;

    @Column(name = "descricao", nullable = false, length = 200)
    private String descricao;

    @Column(name = "unidade", nullable = false, length = 10)
    private String unidade;

    @Column(name = "preco_unidade", nullable = false, precision =
10, scale = 2)
    private BigDecimal preco_unidade;

    @Column(name = "qtd_estoque", nullable = false)
    private Integer qtd_estoque;

    @Column(name = "is_active", nullable = false)
    private Boolean isActive = true;

    @Column(name = "data_criacao", updatable = false)
```

```
    private LocalDateTime dataCriacao;

    @Column(name = "data_modificacao")
    private LocalDateTime dataModificacao;

    @ManyToOne
    @JoinColumn(name = "id_categoria", nullable = false)
    @JsonBackReference
    private Categorias categoria;

    @PrePersist
    protected void onCreate() {
        dataCriacao = LocalDateTime.now();
        dataModificacao = LocalDateTime.now();
    }

    @PreUpdate
    protected void onUpdate() {
        dataModificacao = LocalDateTime.now();
    }

    public Produtos() {
    }

    public Produtos(Long id_produto, String nome_produto, String
descricao, String unidade, BigDecimal preco_unidade,
            Integer qtd_estoque, Boolean isActive, LocalDateTime
dataCriacao, LocalDateTime dataModificacao,
            Categorias categoria) {
        this.id_produto = id_produto;
        this.nome_produto = nome_produto;
        this.descricao = descricao;
        this.unidade = unidade;
        this.preco_unidade = preco_unidade;
        this.qtd_estoque = qtd_estoque;
        this.isActive = isActive;
        this.dataCriacao = dataCriacao;
        this.dataModificacao = dataModificacao;
        this.categoria = categoria;
    }

    public Long getId_produto() {
```

```
        return id_produto;
    }

    public void setId_produto(Long id_produto) {
        this.id_produto = id_produto;
    }

    public String getNome_produto() {
        return nome_produto;
    }

    public void setNome_produto(String nome_produto) {
        this.nome_produto = nome_produto;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public String getUnidade() {
        return unidade;
    }

    public void setUnidade(String unidade) {
        this.unidade = unidade;
    }

    public BigDecimal getPreco_unidade() {
        return preco_unidade;
    }

    public void setPreco_unidade(BigDecimal preco_unidade) {
        this.preco_unidade = preco_unidade;
    }

    public Integer getQtd_estoque() {
        return qtd_estoque;
    }
```

```
    public void setQtd_estoque(Integer qtd_estoque) {
        this.qtd_estoque = qtd_estoque;
    }

    public Boolean getIsActive() {
        return isActive;
    }

    public void setIsActive(Boolean isActive) {
        this.isActive = isActive;
    }

    public LocalDateTime getDataCriacao() {
        return dataCriacao;
    }

    public void setDataCriacao(LocalDateTime dataCriacao) {
        this.dataCriacao = dataCriacao;
    }

    public LocalDateTime getDataModificacao() {
        return dataModificacao;
    }

    public void setDataModificacao(LocalDateTime dataModificacao)
{
        this.dataModificacao = dataModificacao;
    }

    public Categorias getCategoria() {
        return categoria;
    }

    public void setCategoria(Categorias categoria) {
        this.categoria = categoria;
    }

}
```

```java
package br.com.sigas.entities;

import java.time.LocalDate;
import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonManagedReference;

import jakarta.persistence.*;

@Entity
@Table(name = "operacoes")
public class Operacoes {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator
= "operacoes_seq")
    @SequenceGenerator(name = "operacoes_seq", sequenceName =
"operacoes_sequence", allocationSize = 1)
    private Integer id_operacao;

    @ManyToOne
    @JoinColumn(name = "id_pessoa", nullable = false)
    private Pessoas pessoa;

    @Column(name = "tipo_operacao", nullable = false)
    private Character tipo_operacao;

    @Column(name = "data_operacao", nullable = false)
    private LocalDate data_operacao;

    @OneToMany(mappedBy = "operacao", cascade = CascadeType.ALL,
orphanRemoval = true)
    @JsonIgnoreProperties("operacao")
    @JsonManagedReference
    private List<ItensOperacao> itens_operacao;

    public Operacoes() {
    }

    public Operacoes(Integer id_operacao, Pessoas pessoa,
Character tipo_operacao, LocalDate data_operacao,
```

```
            List<ItensOperacao> itens_operacao) {
        this.id_operacao = id_operacao;
        this.pessoa = pessoa;
        this.tipo_operacao = tipo_operacao;
        this.data_operacao = data_operacao;
        this.itens_operacao = itens_operacao;
    }

    public Integer getId_operacao() {
        return id_operacao;
    }

    public void setId_operacao(Integer id_operacao) {
        this.id_operacao = id_operacao;
    }

    public Pessoas getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoas pessoa) {
        this.pessoa = pessoa;
    }

    public Character getTipo_operacao() {
        return tipo_operacao;
    }

    public void setTipo_operacao(Character tipo_operacao) {
        this.tipo_operacao = tipo_operacao;
    }

    public LocalDate getData_operacao() {
        return data_operacao;
    }

    public void setData_operacao(LocalDate data_operacao) {
        this.data_operacao = data_operacao;
    }

    public List<ItensOperacao> getItens_operacao() {
        return itens_operacao;
```

```
    }

    public void setItens_operacao(List<ItensOperacao>
itens_operacao) {
        this.itens_operacao = itens_operacao;
    }

}
```

```
package br.com.sigas.entities;

import java.math.BigDecimal;

import com.fasterxml.jackson.annotation.JsonBackReference;

import jakarta.persistence.*;

@Entity
@Table(name = "itens_operacao")
public class ItensOperacao {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id_item_operacao;

    @ManyToOne
    @JoinColumn(name = "id_operacao", nullable = false)
    @JsonBackReference
    private Operacoes operacao;

    @ManyToOne
    @JoinColumn(name = "id_produto", nullable = false)
    private Produtos produto;

    @Column(name = "quantidade", nullable = false)
    private Integer quantidade;

    @Column(name = "preco_unitario", nullable = false)
    private BigDecimal preco_unitario;

    @Column(name = "valor_total", precision = 10, scale = 2)
```

```
    private BigDecimal valor_total;

    @PrePersist
    @PreUpdate
    public void calcularValorTotal() {
        if (quantidade != null && preco_unitario != null) {
            this.valor_total = preco_unitario.multiply(new
BigDecimal(quantidade));
        }
    }

    public ItensOperacao() {
    }

    public ItensOperacao(Long id_item_operacao, Operacoes
operacao, Produtos produto, Integer quantidade,
            BigDecimal preco_unitario, BigDecimal valor_total) {
        this.id_item_operacao = id_item_operacao;
        this.operacao = operacao;
        this.produto = produto;
        this.quantidade = quantidade;
        this.preco_unitario = preco_unitario;
        this.valor_total = valor_total;
    }

    public Long getId_item_operacao() {
        return id_item_operacao;
    }

    public void setId_item_operacao(Long id_item_operacao) {
        this.id_item_operacao = id_item_operacao;
    }

    public Operacoes getOperacao() {
        return operacao;
    }

    public void setOperacao(Operacoes operacao) {
        this.operacao = operacao;
    }

    public Produtos getProduto() {
```

```
        return produto;
    }

    public void setProduto(Produtos produto) {
        this.produto = produto;
    }

    public Integer getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(Integer quantidade) {
        this.quantidade = quantidade;
    }

    public BigDecimal getPreco_unitario() {
        return preco_unitario;
    }

    public void setPreco_unitario(BigDecimal preco_unitario) {
        this.preco_unitario = preco_unitario;
    }

    public BigDecimal getValor_total() {
        return valor_total;
    }

    public void setValor_total(BigDecimal valor_total) {
        this.valor_total = valor_total;
    }

}
```

3.3 Link do projeto no github (atualizar o READ.me)

https://github.com/anonatopedro/Projeto-Extens-o-Curricularizado-

3.4 Vídeo demonstrativo (link do Google Drive ou YouTube)

https://drive.google.com/drive/folders/13wgcpG86CfUS5P_MLLHJwNf-8-NFMXJI?usp=sharing

3.5 Diagrama do banco de dados

3.6 Injeção de dados (Script do banco de dados) contendo no mínimo 10 inserts em cada tabela

```
INSERT INTO categorias (nome_categoria, is_active, data_criacao)
VALUES
('Shampoo Artesanal', 1, GETDATE()),
('Condicionador', 1, GETDATE()),
('Sabonete Artesanal Líquido', 1, GETDATE()),
('Sabonete Artesanal em Barra', 1, GETDATE()),
('Rapé Artesanal', 1, GETDATE()),
('Santo Cruzeiro', 1, GETDATE()),
('Guia', 1, GETDATE()),
('Difusor de Ambiente', 1, GETDATE()),
('Creme Hidratante', 1, GETDATE()),
('Mistura de Ervas (Jurema)', 1, GETDATE()),
('Terço', 1, GETDATE()),
('Kuripe', 1, GETDATE()),
('Tepi', 1, GETDATE());
```

```
INSERT INTO produtos (nome_produto, descricao, unidade, preco_unidade, qtd_estoque,
is_active, data_criacao, id_categoria)
VALUES
('Shampoo Artesanal Maracujá', 'Shampoo artesanal de maracujá', 'unidade', 25.50, 50, 1,
GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Shampoo
Artesanal')),
('Shampoo Artesanal Banana', 'Shampoo artesanal de banana', 'unidade', 26.00, 50, 1,
GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Shampoo
Artesanal')),
('Shampoo Artesanal Alecrim', 'Shampoo artesanal de alecrim', 'unidade', 24.90, 50, 1,
GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Shampoo
Artesanal'));
```

```
INSERT INTO produtos (nome_produto, descricao, unidade, preco_unidade, qtd_estoque,
is_active, data_criacao, id_categoria)
VALUES
('Condicionador Manteiga de Karité', 'Condicionador com manteiga de karité', 'unidade',
30.00, 40, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria =
'Condicionador')),
('Condicionador Olibano', 'Condicionador com óleo essencial de olibano', 'unidade',
32.00, 40, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria =
'Condicionador')),
('Condicionador Samaúma', 'Condicionador de Samaúma', 'unidade', 28.50, 40, 1, GETDATE(),
(SELECT id_categoria FROM categorias WHERE nome_categoria = 'Condicionador'));
```

```
INSERT INTO produtos (nome_produto, descricao, unidade, preco_unidade, qtd_estoque,
is_active, data_criacao, id_categoria)
VALUES
('Sabonete Líquido Alecrim', 'Sabonete artesanal líquido de alecrim', 'unidade', 18.00,
60, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Sabonete
Artesanal Líquido')),
('Sabonete Líquido Arruda com Sal Grosso', 'Sabonete artesanal líquido de arruda com sal
grosso', 'unidade', 20.00, 60, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE
nome_categoria = 'Sabonete Artesanal Líquido')),
('Sabonete Líquido Lavanda', 'Sabonete artesanal líquido de lavanda', 'unidade', 19.50,
60, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Sabonete
Artesanal Líquido')),
('Sabonete Líquido Camomila', 'Sabonete artesanal líquido de camomila', 'unidade', 19.00,
60, 1, GETDATE(), (SELECT id_categoria FROM categorias WHERE nome_categoria = 'Sabonete
Artesanal Líquido'));
```

```sql
INSERT INTO pessoas (tipo_pessoa, nome_pessoa, email_pessoa, endereco_pessoa,
tel1_pessoa, tel2_pessoa, data_nascimento)
VALUES
('F', 'Carlos Souza', 'carlos.souza@example.com', 'Av. Paulista, 101', '(12) 91000-1001',
'(12) 92000-1001', '1981-02-12'),
('F', 'Fernanda Oliveira', 'fernanda.oliveira@example.com', 'Rua Bela Vista, 102', '(13)
91000-1002', '(13) 92000-1002', '1982-03-13'),
('F', 'João Almeida', 'joão.almeida@example.com', 'Rua Nova, 103', '(14) 91000-1003',
'(14) 92000-1003', '1983-04-14'),
('F', 'Mariana Santos', 'mariana.santos@example.com', 'Av. Central, 104', '(15) 91000-
1004', '(15) 92000-1004', '1984-05-15'),
('F', 'Rodrigo Batista', 'rodrigo.batista@example.com', 'Rua das Palmeiras, 105', '(16)
91000-1005', '(16) 92000-1005', '1985-06-16'),
('F', 'Tatiana Castro', 'tatiana.castro@example.com', 'Rua Aurora, 106', '(17) 91000-
1006', '(17) 92000-1006', '1986-07-17'),
('F', 'Gabriel Ferreira', 'gabriel.ferreira@example.com', 'Av. do Sol, 107', '(18) 91000-
1007', '(18) 92000-1007', '1987-08-18'),
('F', 'Luciana Pereira', 'luciana.pereira@example.com', 'Rua Verde, 108', '(19) 91000-
1008', '(19) 92000-1008', '1988-09-19'),
('F', 'Bruno Lima', 'bruno.lima@example.com', 'Av. Azul, 109', '(20) 91000-1009', '(20)
92000-1009', '1989-01-11'),
('F', 'Juliana Alves', 'juliana.alves@example.com', 'Rua dos Pinhais, 110', '(11) 91000-
1010', '(11) 92000-1010', '1980-02-12'),
('F', 'Pedro Henrique', 'pedro.henrique@example.com', 'Av. Independência, 111', '(12)
91000-1011', '(12) 92000-1011', '1981-03-13'),
('F', 'Bianca Monteiro', 'bianca.monteiro@example.com', 'Rua Nova Esperança, 112', '(13)
91000-1012', '(13) 92000-1012', '1982-04-14'),
('F', 'Ricardo Lopes', 'ricardo.lopes@example.com', 'Rua da Alegria, 113', '(14) 91000-
1013', '(14) 92000-1013', '1983-05-15'),
('F', 'Patrícia Mendes', 'patrícia.mendes@example.com', 'Av. Brasil, 114', '(15) 91000-
1014', '(15) 92000-1014', '1984-06-16');



INSERT INTO pessoas_fisicas (id_pessoa, cpf)
VALUES
(1, '001.002.003-07'),
(2, '002.004.006-14'),
(3, '003.006.009-21'),
(4, '004.008.012-28'),
(5, '005.010.015-35'),
(6, '006.012.018-42'),
(7, '007.014.021-49'),
(8, '008.016.024-56'),
(9, '009.018.027-63'),
(10, '010.020.030-70'),
(11, '011.022.033-77'),
(12, '012.024.036-84'),
(13, '013.026.039-91'),
(14, '014.028.042-98'),
(15, '015.030.045-05'),
(16, '016.032.048-12'),
(17, '017.034.051-19');



INSERT INTO pessoas (tipo_pessoa, nome_pessoa, email_pessoa, endereco_pessoa,
tel1_pessoa, tel2_pessoa, data_nascimento)
VALUES
('J', 'Alessandro Nunes', 'alessandro.nunes@grupoexcel.com', 'Av. dos Bandeirantes, 123',
'(11) 91001-1001', '(11) 92001-1001', '1982-02-14'),
```

```
('J', 'Camila Ribeiro', 'camila.ribeiro@futuroverde.com', 'Rua das Árvores, 456', '(21)
91002-2002', '(21) 92002-2002', '1987-06-08'),
('J', 'Renato Lima', 'renato.lima@logexpress.com', 'Av. Industrial, 789', '(31) 91003-
3003', '(31) 92003-3003', '1990-11-30'),
('J', 'Jéssica Souza', 'jessica.souza@mundonovo.com', 'Rua Nova Esperança, 101', '(41)
91004-4004', '(41) 92004-4004', '1985-09-15'),
('J', 'Pedro Henrique', 'pedro.henrique@megaeng.com', 'Av. das Indústrias, 222', '(51)
91005-5005', '(51) 92005-5005', '1992-01-01'),
('J', 'Letícia Alves', 'leticia.alves@vidamais.com', 'Rua Bem Estar, 333', '(61) 91006-
6006', '(61) 92006-6006', '1983-08-22'),
('J', 'Vinícius Carvalho', 'vinicius.carvalho@techworld.com', 'Av. do Conhecimento, 444',
'(71) 91007-7007', '(71) 92007-7007', '1980-12-12'),
('J', 'Patrícia Mendes', 'patricia.mendes@beautyshop.com', 'Rua da Beleza, 555', '(81)
91008-8008', '(81) 92008-8008', '1988-05-09'),
('J', 'Lucas Barros', 'lucas.barros@novaconstrucao.com', 'Av. Nova, 666', '(91) 91009-
9009', '(91) 92009-9009', '1981-03-03'),
('J', 'Mariana Oliveira', 'mariana.oliveira@ecoplan.com', 'Rua Sustentável, 777', '(31)
91010-0010', '(31) 92010-0010', '1989-07-19');
```

```
INSERT INTO pessoas_juridicas (id_pessoa, cnpj, razao_social)
VALUES
(151, '01.123.456/0001-11', 'Grupo Excel Contabilidade LTDA'),
(152, '01.234.567/0002-22', 'Futuro Verde Sustentabilidade ME'),
(153, '01.345.678/0003-33', 'LogExpress Transporte e Logística LTDA'),
(154, '01.456.789/0004-44', 'Mundo Novo Comércio de Produtos LTDA'),
(155, '01.567.890/0005-55', 'MegaEng Engenharia e Projetos SA'),
(156, '01.678.901/0006-66', 'Vida Mais Plano de Saúde LTDA'),
(157, '01.789.012/0007-77', 'TechWorld Soluções em TI EIRELI'),
(158, '01.890.123/0008-88', 'BeautyShop Cosméticos LTDA'),
(159, '01.901.234/0009-99', 'Nova Construção Engenharia LTDA'),
(160, '01.012.345/0010-10', 'EcoPlan Consultoria Ambiental SA');
```

```
INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (1, 'V', '2024-11-29', 0); -- id_pessoa = 1

DECLARE @id_operacao_1 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES
(@id_operacao_1, 1, 2, 25.50), -- Shampoo Artesanal Maracujá
(@id_operacao_1, 3, 1, 18.00), -- Sabonete Líquido Alecrim
(@id_operacao_1, 5, 1, 15.00); -- Rapé Samaúma

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (2, 'V', '2024-11-28', 0); -- id_pessoa = 2

DECLARE @id_operacao_2 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES
(@id_operacao_2, 2, 1, 26.00), -- Shampoo Artesanal Banana
(@id_operacao_2, 4, 2, 20.00), -- Sabonete Líquido Arruda com Sal Grosso
(@id_operacao_2, 6, 3, 16.00); -- Rapé Sansara

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (3, 'V', '2024-11-27', 0); -- id_pessoa = 3

DECLARE @id_operacao_3 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
```

```
VALUES
(@id_operacao_3, 7, 2, 17.00), -- Rapé 3 Ervas
(@id_operacao_3, 8, 1, 14.50), -- Rapé Cumaru
(@id_operacao_3, 9, 1, 16.50); -- Rapé Mulateiro

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (4, 'V', '2024-11-09', 0);
DECLARE @id_operacao_4 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_4, 3, 5, 14.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_4, 4, 2, 16.00);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_4, 2, 4, 13.00);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (5, 'V', '2024-11-30', 0);
DECLARE @id_operacao_5 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_5, 2, 3, 13.00);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_5, 4, 4, 16.00);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_5, 5, 3, 17.50);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (6, 'V', '2024-11-09', 0);
DECLARE @id_operacao_6 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_6, 1, 3, 11.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_6, 5, 2, 17.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_6, 2, 5, 13.00);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (7, 'V', '2024-11-29', 0);
DECLARE @id_operacao_7 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_7, 8, 4, 22.00);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_7, 5, 2, 17.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_7, 4, 3, 16.00);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (8, 'V', '2024-11-11', 0);
DECLARE @id_operacao_8 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_8, 6, 4, 19.00);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
```

```
VALUES (@id_operacao_8, 9, 2, 23.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_8, 5, 1, 17.50);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (9, 'V', '2024-11-18', 0);
DECLARE @id_operacao_9 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_9, 3, 5, 14.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_9, 7, 2, 20.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_9, 6, 5, 19.00);

INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao, valor_total)
VALUES (10, 'V', '2024-11-21', 0);
DECLARE @id_operacao_10 INT = SCOPE_IDENTITY();

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_10, 7, 5, 20.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_10, 1, 4, 11.50);

INSERT INTO itens_operacao (id_operacao, id_produto, quantidade, preco_unitario)
VALUES (@id_operacao_10, 4, 4, 16.00);
```

4. Etapa 2 – Implementação de recursos adicionais na base de dados para uma aplicação Web

Objetivo: Agregar novas funções à base de dados persistente, por meio de *stored procedures*, *triggers*, entre outros recursos possíveis.

4.1 Criação das *stored procedure* para receber os dados do sistema e realizar a operação desejada de *insert, update ou delete*.

```
CREATE PROCEDURE inserir_categoria
    @nome_categoria NVARCHAR(100)
AS
BEGIN
    INSERT INTO categorias (nome_categoria, is_active, data_criacao)
    VALUES (@nome_categoria, 1, GETDATE());
END;


CREATE PROCEDURE atualizar_categoria
    @id_categoria INT,
    @nome_categoria NVARCHAR(100)
AS
BEGIN
    UPDATE categorias
    SET nome_categoria = @nome_categoria,
        data_modificacao = GETDATE()
    WHERE id_categoria = @id_categoria;
END;

CREATE PROCEDURE deletar_categoria
    @id_categoria INT
AS
BEGIN
    UPDATE categorias
    SET is_active = 0,
        data_modificacao = GETDATE()
    WHERE id_categoria = @id_categoria;
END;


CREATE PROCEDURE inserir_produto
    @nome_produto NVARCHAR(50),
    @descricao NVARCHAR(MAX),
    @unidade NVARCHAR(10),
    @preco_unidade DECIMAL(10, 2),
    @qtd_estoque INT,
    @id_categoria INT
AS
BEGIN
    INSERT INTO produtos (nome_produto, descricao, unidade, preco_unidade, qtd_estoque,
id_categoria, is_active, data_criacao)
    VALUES (@nome_produto, @descricao, @unidade, @preco_unidade, @qtd_estoque,
@id_categoria, 1, GETDATE());
END;


CREATE PROCEDURE atualizar_produto
    @id_produto INT,
    @nome_produto NVARCHAR(50),
```

```
    @descricao NVARCHAR(MAX),
    @unidade NVARCHAR(10),
    @preco_unidade DECIMAL(10, 2),
    @qtd_estoque INT
AS
BEGIN
    UPDATE produtos
    SET nome_produto = @nome_produto,
        descricao = @descricao,
        unidade = @unidade,
        preco_unidade = @preco_unidade,
        qtd_estoque = @qtd_estoque,
        data_modificacao = GETDATE()
    WHERE id_produto = @id_produto;
END;


CREATE PROCEDURE deletar_produto
    @id_produto INT
AS
BEGIN
    UPDATE produtos
    SET is_active = 0,
        data_modificacao = GETDATE()
    WHERE id_produto = @id_produto;
END;


CREATE PROCEDURE inserir_operacao
    @id_pessoa INT,
    @tipo_operacao CHAR(1),
    @data_operacao DATE
AS
BEGIN
    INSERT INTO operacoes (id_pessoa, tipo_operacao, data_operacao)
    VALUES (@id_pessoa, @tipo_operacao, @data_operacao);
END;

CREATE PROCEDURE atualizar_operacao
    @id_operacao INT,
    @id_pessoa INT,
    @tipo_operacao CHAR(1),
    @data_operacao DATE
AS
BEGIN
    UPDATE operacoes
    SET id_pessoa = @id_pessoa,
        tipo_operacao = @tipo_operacao,
        data_operacao = @data_operacao
    WHERE id_operacao = @id_operacao;
END;


CREATE PROCEDURE deletar_operacao
    @id_operacao INT
AS
BEGIN
    DELETE FROM operacoes
    WHERE id_operacao = @id_operacao;
END;

CREATE PROCEDURE inserir_pessoa_fisica
    @tipo_pessoa CHAR(1),
    @nome_pessoa NVARCHAR(100),
    @email_pessoa NVARCHAR(60),
```

```
    @endereco_pessoa NVARCHAR(200),
    @tel1_pessoa NVARCHAR(13),
    @tel2_pessoa NVARCHAR(13),
    @data_nascimento DATE,
    @cpf NVARCHAR(14)
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        DECLARE @id_pessoa INT;

        INSERT INTO pessoas (
            tipo_pessoa, nome_pessoa, email_pessoa, endereco_pessoa, tel1_pessoa,
tel2_pessoa, data_nascimento, is_active, data_criacao
        )
        VALUES (
            @tipo_pessoa, @nome_pessoa, @email_pessoa, @endereco_pessoa, @tel1_pessoa,
@tel2_pessoa, @data_nascimento, 1, GETDATE()
        );

        SET @id_pessoa = SCOPE_IDENTITY();

        INSERT INTO pessoas_fisicas (
            id_pessoa, cpf
        )
        VALUES (
            @id_pessoa, @cpf
        );

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;


CREATE PROCEDURE atualizar_pessoa_fisica
    @id_pessoa INT,
    @cpf NVARCHAR(14),
    @nome NVARCHAR(100),
    @email NVARCHAR(60),
    @endereco NVARCHAR(200),
    @tel1 NVARCHAR(13),
    @tel2 NVARCHAR(13),
    @data_nascimento DATE
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        UPDATE pessoas_fisicas
        SET cpf = @cpf
        WHERE id_pessoa = @id_pessoa;

        UPDATE pessoas
        SET
            nome_pessoa = @nome, email_pessoa = @email, endereco_pessoa = @endereco,
tel1_pessoa = @tel1, tel2_pessoa = @tel2,
            data_nascimento = @data_nascimento, data_modificacao = GETDATE()
        WHERE id_pessoa = @id_pessoa;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
```

```
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;



CREATE PROCEDURE deletar_pessoa_fisica
    @id_pessoa INT
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        UPDATE pessoas
        SET is_active = 0,
            data_modificacao = GETDATE()
        WHERE id_pessoa = @id_pessoa;

        IF NOT EXISTS (SELECT 1 FROM pessoas_fisicas WHERE id_pessoa = @id_pessoa)
        BEGIN
            THROW 50000, 'Pessoa Física não encontrada.', 1;
        END

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;


CREATE PROCEDURE inserir_pessoa_juridica
    @tipo_pessoa CHAR(1),
    @nome_pessoa NVARCHAR(100),
    @email_pessoa NVARCHAR(60),
    @endereco_pessoa NVARCHAR(200),
    @tel1_pessoa NVARCHAR(13),
    @tel2_pessoa NVARCHAR(13),
    @data_nascimento DATE,
    @cnpj NVARCHAR(18),
    @razao_social NVARCHAR(100)
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        DECLARE @id_pessoa INT;

        -- Insere na tabela "pessoas"
        INSERT INTO pessoas (
            tipo_pessoa, nome_pessoa, email_pessoa, endereco_pessoa, tel1_pessoa,
tel2_pessoa, data_nascimento,is_active, data_criacao
        )
        VALUES (
            @tipo_pessoa, @nome_pessoa, @email_pessoa, @endereco_pessoa, @tel1_pessoa,
@tel2_pessoa, @data_nascimento, 1, GETDATE()
        );

        SET @id_pessoa = SCOPE_IDENTITY();

        -- Insere na tabela "pessoas_juridicas"
        INSERT INTO pessoas_juridicas (
            id_pessoa, cnpj, razao_social
        )
```

```
        VALUES (
            @id_pessoa, @cnpj, @razao_social
        );

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;


CREATE PROCEDURE atualizar_pessoa_juridica
    @id_pessoa INT,
    @nome_pessoa NVARCHAR(100),
    @email_pessoa NVARCHAR(60),
    @endereco_pessoa NVARCHAR(200),
    @tel1_pessoa NVARCHAR(13),
    @tel2_pessoa NVARCHAR(13),
    @data_nascimento DATE,
    @cnpj NVARCHAR(18),
    @razao_social NVARCHAR(100)
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Atualiza a tabela "pessoas"
        UPDATE pessoas
        SET
            nome_pessoa = @nome_pessoa,
            email_pessoa = @email_pessoa,
            endereco_pessoa = @endereco_pessoa,
            tel1_pessoa = @tel1_pessoa,
            tel2_pessoa = @tel2_pessoa,
            data_nascimento = @data_nascimento,
            data_modificacao = GETDATE()
        WHERE id_pessoa = @id_pessoa;

        -- Atualiza a tabela "pessoas_juridicas"
        UPDATE pessoas_juridicas
        SET
            cnpj = @cnpj,
            razao_social = @razao_social
        WHERE id_pessoa = @id_pessoa;

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;


select * from pessoas

CREATE PROCEDURE deletar_pessoa_juridica
    @id_pessoa INT
AS
BEGIN
    BEGIN TRANSACTION;
    BEGIN TRY
        -- Atualiza a tabela "pessoas" para marcar como inativa
        UPDATE pessoas
```

```
        SET is_active = 0,
            data_modificacao = GETDATE()
        WHERE id_pessoa = @id_pessoa;

        -- Verifica se a pessoa jurídica existe
        IF NOT EXISTS (SELECT 1 FROM pessoas_juridicas WHERE id_pessoa = @id_pessoa)
        BEGIN
            THROW 50000, 'Pessoa Jurídica não encontrada.', 1;
        END

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION;
        THROW;
    END CATCH
END;
```

4.2 Criar as *views* e *functions* para realizar as consultas necessárias de forma que o sistema receba os dados e possa exibi-los aos usuários.

```
CREATE VIEW vw_operacoes_com_valor_total AS
SELECT
    o.id_operacao,
    o.id_pessoa,
    o.tipo_operacao,
    o.data_operacao,
    SUM(io.valor_total) AS valor_total
FROM
    operacoes o
LEFT JOIN
    itens_operacao io ON o.id_operacao = io.id_operacao
GROUP BY
    o.id_operacao, o.id_pessoa, o.tipo_operacao, o.data_operacao;


CREATE VIEW vw_estoque_atual AS
SELECT
    P.id_produto,
    P.nome_produto,
    P.qtd_estoque,
    P.unidade,
    P.preco_unidade
FROM Produtos P
WHERE P.qtd_estoque > 0;


select * from v

CREATE VIEW vw_produtos_estoque_baixo AS
SELECT
    p.id_produto,
    p.nome_produto,
    p.qtd_estoque,
    p.unidade,
    p.preco_unidade
FROM produtos p
WHERE p.qtd_estoque < 10;
```

```sql
CREATE VIEW vw_operacoes_periodo AS
SELECT
    o.id_operacao,
    o.tipo_operacao,
    o.data_operacao,
    p.nome_pessoa AS pessoa_nome,
    p.tipo_pessoa
FROM operacoes o
JOIN pessoas p ON o.id_pessoa = p.id_pessoa;


CREATE VIEW vw_resumo_financeiro AS
SELECT
    o.tipo_operacao,
    SUM(io.valor_total) AS total_valor,
    COUNT(o.id_operacao) AS total_operacoes,
    o.data_operacao
FROM operacoes o
JOIN itens_operacao io ON o.id_operacao = io.id_operacao
GROUP BY o.tipo_operacao, o.data_operacao;


CREATE VIEW vw_pessoas_fisicas AS
SELECT
    pf.id_pessoa,
    p.nome_pessoa,
    p.email_pessoa,
    pf.cpf,
    p.tel1_pessoa,
    p.tel2_pessoa,
    p.endereco_pessoa
FROM pessoas p
JOIN pessoas_fisicas pf ON p.id_pessoa = pf.id_pessoa;


CREATE VIEW vw_pessoas_juridicas AS
SELECT
    pj.id_pessoa,
    p.nome_pessoa,
    p.email_pessoa,
    pj.cnpj,
    pj.razao_social,
    p.tel1_pessoa,
    p.tel2_pessoa,
    p.endereco_pessoa
FROM pessoas p
JOIN pessoas_juridicas pj ON p.id_pessoa = pj.id_pessoa;


CREATE VIEW vw_valor_gasto_por_categoria AS
SELECT
    c.id_categoria,
    c.nome_categoria,
    SUM(io.quantidade * io.preco_unitario) AS valor_total_gasto
FROM
    itens_operacao io
INNER JOIN
    produtos p ON io.id_produto = p.id_produto
INNER JOIN
    categorias c ON p.id_categoria = c.id_categoria
INNER JOIN
    operacoes o ON io.id_operacao = o.id_operacao
WHERE
    o.tipo_operacao = 'V' -- Apenas operações de compra
```

```sql
GROUP BY
    c.id_categoria, c.nome_categoria;



CREATE VIEW vw_quantidade_produtos_por_categoria AS
SELECT
    c.id_categoria,
    c.nome_categoria,
    COUNT(p.id_produto) AS quantidade_produtos
FROM
    categorias c
LEFT JOIN
    produtos p ON c.id_categoria = p.id_categoria
GROUP BY
    c.id_categoria, c.nome_categoria;


CREATE FUNCTION calcular_total_vendas(@inicio DATE, @fim DATE)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @total DECIMAL(10, 2);
    SELECT @total = SUM(io.valor_total)
    FROM operacoes o
    JOIN itens_operacao io ON o.id_operacao = io.id_operacao
    WHERE o.tipo_operacao = 'V' AND o.data_operacao BETWEEN @inicio AND @fim;
    RETURN @total;
END;


CREATE FUNCTION calcular_total_compras(@inicio DATE, @fim DATE)
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @total DECIMAL(10, 2);
    SELECT @total = SUM(io.valor_total)
    FROM operacoes o
    JOIN itens_operacao io ON o.id_operacao = io.id_operacao
    WHERE o.tipo_operacao = 'C' AND o.data_operacao BETWEEN @inicio AND @fim;
    RETURN @total;
END;


CREATE FUNCTION produtos_mais_vendidos(@inicio DATE, @fim DATE)
RETURNS TABLE
AS
RETURN (
    SELECT
        p.id_produto,
        p.nome_produto,
        SUM(io.quantidade) AS total_vendido
    FROM itens_operacao io
    JOIN produtos p ON io.id_produto = p.id_produto
    JOIN operacoes o ON io.id_operacao = o.id_operacao
    WHERE o.tipo_operacao = 'V' AND o.data_operacao BETWEEN @inicio AND @fim
    GROUP BY p.id_produto, p.nome_produto
);
```

4.3 Automatizar algumas transações no banco de dados usando *triggers*.

```
CREATE TRIGGER trg_atualizar_estoque
ON itens_operacao
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @id_produto BIGINT, @quantidade INT, @tipo_operacao CHAR(1);

    SELECT
        @id_produto = i.id_produto,
        @quantidade = i.quantidade,
        @tipo_operacao = o.tipo_operacao
    FROM inserted i
    INNER JOIN operacoes o ON i.id_operacao = o.id_operacao;

    IF @tipo_operacao = 'C' -- Compra
    BEGIN
        UPDATE produtos
        SET qtd_estoque = qtd_estoque + @quantidade
        WHERE id_produto = @id_produto;
    END
    ELSE IF @tipo_operacao = 'V' -- Venda
    BEGIN
        UPDATE produtos
        SET qtd_estoque = qtd_estoque - @quantidade
        WHERE id_produto = @id_produto;
    END
END;


CREATE TRIGGER trg_atualizar_valor_total
ON itens_operacao
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    UPDATE o
    SET o.valor_total = (
        SELECT ISNULL(SUM(io.quantidade * io.preco_unitario), 0)
        FROM itens_operacao io
        WHERE io.id_operacao = o.id_operacao
    )
    FROM operacoes o
    WHERE o.id_operacao IN (
        SELECT DISTINCT id_operacao FROM inserted
        UNION
        SELECT DISTINCT id_operacao FROM deleted
    );
END;

CREATE TRIGGER trg_garantir_estoque_positivo
ON itens_operacao
AFTER INSERT, UPDATE
AS
BEGIN
    DECLARE @id_produto BIGINT, @quantidade INT, @tipo_operacao CHAR(1);

    SELECT
        @id_produto = i.id_produto,
        @quantidade = i.quantidade,
        @tipo_operacao = o.tipo_operacao
    FROM inserted i
    INNER JOIN operacoes o ON i.id_operacao = o.id_operacao;

    IF @tipo_operacao = 'V' -- Venda
```

```
    BEGIN
        IF EXISTS (
            SELECT 1
            FROM produtos
            WHERE id_produto = @id_produto AND qtd_estoque < @quantidade
        )
        BEGIN
            RAISERROR ('Estoque insuficiente para a venda.', 16, 1);
            ROLLBACK TRANSACTION;
        END
    END
END;


CREATE TRIGGER trg_atualizar_data_modificacao_categorias
ON categorias
AFTER UPDATE
AS
BEGIN
    UPDATE categorias
    SET data_modificacao = GETDATE()
    WHERE id_categoria IN (SELECT id_categoria FROM inserted);
END;


CREATE TRIGGER trg_atualizar_data_modificacao_produtos
ON produtos
AFTER UPDATE
AS
BEGIN
    UPDATE produtos
    SET data_modificacao = GETDATE()
    WHERE id_produto IN (SELECT id_produto FROM inserted);
END;


CREATE TRIGGER trg_atualizar_data_modificacao_pessoas
ON pessoas
AFTER UPDATE
AS
BEGIN
    UPDATE pessoas
    SET data_modificacao = GETDATE()
    WHERE id_pessoa IN (SELECT id_pessoa FROM inserted);
END;


CREATE TRIGGER trg_log_alteracoes_categorias
ON categorias
AFTER UPDATE
AS
BEGIN
    DECLARE @id_categoria INT, @nome_categoria_old NVARCHAR(100), @nome_categoria_new
NVARCHAR(100);

    SELECT @id_categoria = i.id_categoria,
           @nome_categoria_old = d.nome_categoria,
           @nome_categoria_new = i.nome_categoria
    FROM inserted i
    INNER JOIN deleted d ON i.id_categoria = d.id_categoria;

    IF @nome_categoria_old <> @nome_categoria_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
```

```
        VALUES ('categorias', @id_categoria, 'nome_categoria', @nome_categoria_old,
@nome_categoria_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_produtos
ON produtos
AFTER UPDATE
AS
BEGIN
    DECLARE @id_produto INT, @nome_produto_old NVARCHAR(50), @nome_produto_new
NVARCHAR(50);

    SELECT @id_produto = i.id_produto,
           @nome_produto_old = d.nome_produto,
           @nome_produto_new = i.nome_produto
    FROM inserted i
    INNER JOIN deleted d ON i.id_produto = d.id_produto;

    IF @nome_produto_old <> @nome_produto_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('produtos', @id_produto, 'nome_produto', @nome_produto_old,
@nome_produto_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_pessoas
ON pessoas
AFTER UPDATE
AS
BEGIN
    DECLARE @id_pessoa INT, @nome_pessoa_old NVARCHAR(100), @nome_pessoa_new
NVARCHAR(100);

    SELECT @id_pessoa = i.id_pessoa,
           @nome_pessoa_old = d.nome_pessoa,
           @nome_pessoa_new = i.nome_pessoa
    FROM inserted i
    INNER JOIN deleted d ON i.id_pessoa = d.id_pessoa;

    IF @nome_pessoa_old <> @nome_pessoa_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('pessoas', @id_pessoa, 'nome_pessoa', @nome_pessoa_old,
@nome_pessoa_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_pessoas_fisicas
ON pessoas_fisicas
AFTER UPDATE
AS
BEGIN
    DECLARE @id_pessoa INT, @cpf_old NVARCHAR(14), @cpf_new NVARCHAR(14);

    SELECT @id_pessoa = i.id_pessoa,
           @cpf_old = d.cpf,
           @cpf_new = i.cpf
    FROM inserted i
```

```
    INNER JOIN deleted d ON i.id_pessoa = d.id_pessoa;

    IF @cpf_old <> @cpf_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('pessoas_fisicas', @id_pessoa, 'cpf', @cpf_old, @cpf_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_pessoas_juridicas
ON pessoas_juridicas
AFTER UPDATE
AS
BEGIN
    DECLARE @id_pessoa INT, @cnpj_old NVARCHAR(18), @cnpj_new NVARCHAR(18);

    SELECT @id_pessoa = i.id_pessoa,
           @cnpj_old = d.cnpj,
           @cnpj_new = i.cnpj
    FROM inserted i
    INNER JOIN deleted d ON i.id_pessoa = d.id_pessoa;

    IF @cnpj_old <> @cnpj_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('pessoas_juridicas', @id_pessoa, 'cnpj', @cnpj_old, @cnpj_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_operacoes
ON operacoes
AFTER UPDATE
AS
BEGIN
    DECLARE @id_operacao INT, @tipo_operacao_old CHAR(1), @tipo_operacao_new CHAR(1);

    SELECT @id_operacao = i.id_operacao,
           @tipo_operacao_old = d.tipo_operacao,
           @tipo_operacao_new = i.tipo_operacao
    FROM inserted i
    INNER JOIN deleted d ON i.id_operacao = d.id_operacao;

    IF @tipo_operacao_old <> @tipo_operacao_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('operacoes', @id_operacao, 'tipo_operacao', @tipo_operacao_old,
@tipo_operacao_new);
    END
END;


CREATE TRIGGER trg_log_alteracoes_itens_operacao
ON itens_operacao
AFTER UPDATE
AS
BEGIN
    DECLARE @id_item_operacao INT, @quantidade_old INT, @quantidade_new INT;

    SELECT @id_item_operacao = i.id_item_operacao,
           @quantidade_old = d.quantidade,
```

```
            @quantidade_new = i.quantidade
    FROM inserted i
    INNER JOIN deleted d ON i.id_item_operacao = d.id_item_operacao;

    IF @quantidade_old <> @quantidade_new
    BEGIN
        INSERT INTO log_alteracoes (tabela_alterada, id_registro, campo_alterado,
valor_antigo, valor_novo)
        VALUES ('itens_operacao', @id_item_operacao, 'quantidade', CAST(@quantidade_old
AS NVARCHAR(MAX)), CAST(@quantidade_new AS NVARCHAR(MAX)));
    END
END;
```

4.4 Configurar o banco de dados para criar perfil de usuário e suas permissões para aumentar a segurança ao acesso aos dados, melhorando a segurança do sistema.

```
CREATE LOGIN Admin WITH PASSWORD = 'Admin2024';
CREATE LOGIN Operacional1 WITH PASSWORD = 'Operacional2024';

CREATE USER Admin FOR LOGIN Admin;
CREATE USER Operacional1 FOR LOGIN Operacional1;

CREATE ROLE Role_Administrador;
CREATE ROLE Role_Operacional;

GRANT SELECT, INSERT, UPDATE, DELETE ON pessoas TO Role_Administrador;
GRANT SELECT, INSERT, UPDATE, DELETE ON produtos TO Role_Administrador;
GRANT SELECT, INSERT, UPDATE, DELETE ON categorias TO Role_Administrador;
GRANT SELECT, INSERT, UPDATE, DELETE ON operacoes TO Role_Administrador;
GRANT SELECT, INSERT, UPDATE, DELETE ON itens_operacao TO Role_Administrador;
GRANT EXECUTE ON SCHEMA::dbo TO Role_Administrador;



-- Para teste de Usuário (Operador1)
-- Deletar (inativar) um registro em pessoas
UPDATE pessoas SET is_active = 0 WHERE id_pessoa = 1;
Update pessoas set is_active = 1 where id_pessoa = 1;

select * from pessoas
```

4.5 Automatizar *backup´s* periódicos para o banco de dados do sistema, garantindo que os dados estarão armazenados e seguros, mas possam ser recuperados, caso seja necessário.

5. Etapa 3 – Aplicação de métodos estatísticos para geração de dados e informações numa planilha de Excel

Objetivo: Implementar uma planilha eletrônica que permita a análise de dados e informações de negócios da empresa estudada a partir de técnicas e/ou indicadores estatísticos (medidas de tendência central, regressão, probabilidade, inferência entre outras).

5.1. Tabelas e Gráficos: ferramentas que ajudam a visualizar a distribuição e a relação entre dados. Dentre as principais representações temos:

- Gráficos de linha: usados para mostrar tendências ao longo do tempo.
- Gráficos de colunas/barras: utilizados para comparar valores entre diferentes categorias
- Gráficos de setores (ou pizza): representam a composição de um todo, representados em partes proporcionais.

5.2. Distribuição de frequências e a representação gráfica de uma distribuição de frequências (histogramas)

5.3. Medidas de Posição: fornecem medidas que ajudam na caracterização e comportamento dos elementos de uma série ou um conjunto de observações. Tais medidas incluem a Média Aritmética Simples, a Mediana e a Moda, que indicam a localização central dos dados.

5.4. Medidas de Variabilidade ou Dispersão: parâmetros estatísticos que quantificam a variação dos valores de um conjunto de dados. A finalidade dessas medidas é avaliar o grau de concentração ou afastamento entre os valores, ou seja, o quanto eles estão dispersos em relação à Média Aritmética Simples. Dentre as principais medidas estão a amplitude total, o desvio-padrão, a variância e o coeficiente de variação de Pearson.

5.5. Análise de Regressão (regressão linear): técnica que estima as relações entre variáveis dependentes e independentes para prever resultados

5.6. Gráficos de Dispersão: Utilizados para exibir a relação entre duas variáveis, plotando pontos em um gráfico. Correlação (R-quadrado): medida que indica a força e a direção da relação (linear) entre duas variáveis.

6. Etapa 4 – Estudo da qualidade total dos processos e demandas existentes numa organização

Objetivo: Desenvolver um estudo de qualidade total e melhoria contínua da organização estudada a partir dos levantamentos dos processos de negócio e demandas existentes já detectadas previamente, apontando possíveis soluções e melhorias.

6.1 Apresentar o negócio, a Missão, a Visão, os valores e os meios de comunicá-los para os colaboradores da empresa.

Negócio:
A Arte Sagrada, representada pelo projeto SIGAS, é uma loja de produtos naturais e artigos religiosos, especializada em oferecer higiene natural e conexão com o Sagrado através de terços e rapés medicinais indígenas. Seu foco é atender clientes e fornecedores com eficiência, garantindo uma gestão ágil de estoques e operações.

Missão:
Proporcionar acesso a produtos naturais e artefatos religiosos de qualidade, promovendo bem-estar, conexão, sustentabilidade e respeito à natureza e às tradições indígenas.

Visão:
Ser reconhecida como referência nacional na comercialização de produtos naturais e religiosos, fortalecendo práticas de despertar da consciência, higiene sustentável e expandindo seu impacto social e ambiental.

Valores:

1. Sustentabilidade e respeito ao meio ambiente.
2. Ética e transparência nos negócios.
3. Valorização da cultura indígena e religiosa.
4. Compromisso com a qualidade e a satisfação do cliente.
5. Promoção do bem-estar e da espiritualidade.

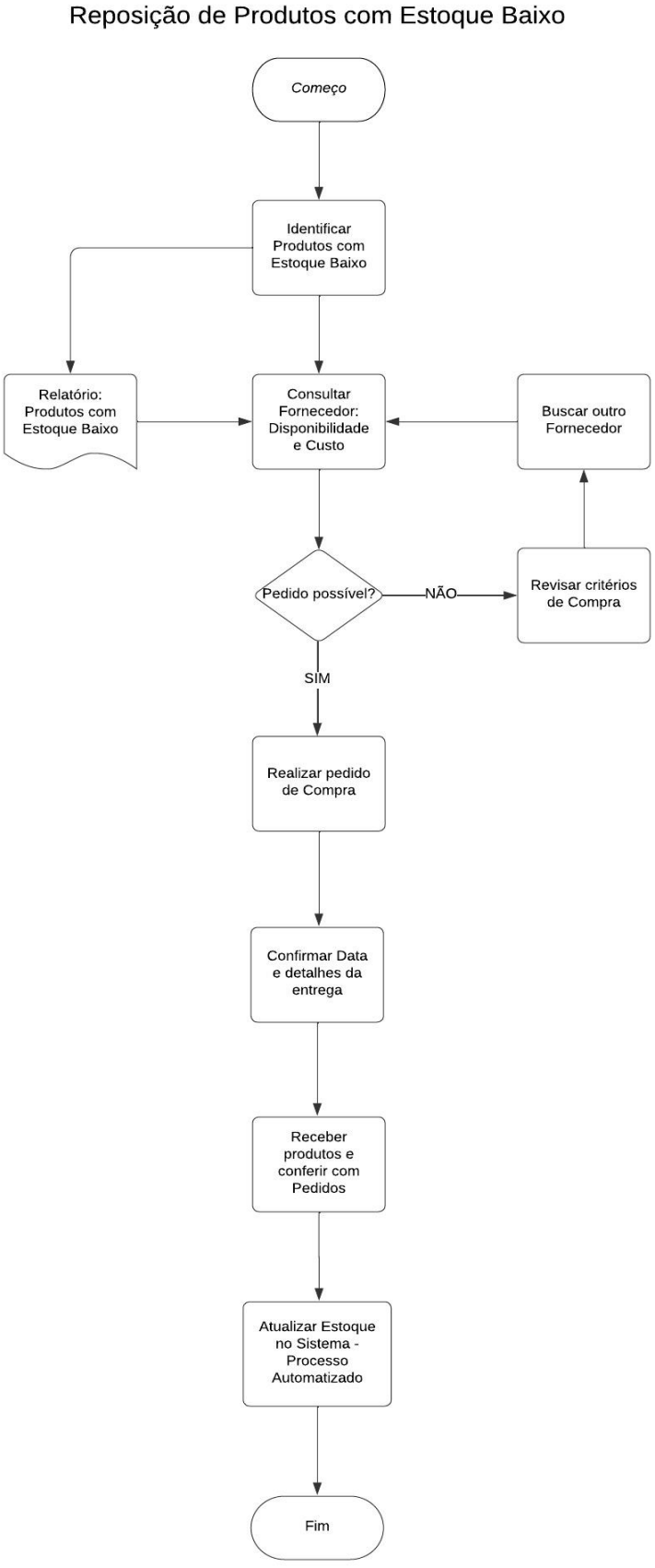
Meios de comunicação dos valores:

- **Reuniões regulares:** Encontros semanais para discutir o desempenho e reforçar os valores da empresa.
- **Manuais e treinamentos:** Materiais que explicam a missão, visão e valores, além de capacitar os colaboradores.
- **Quadros de avisos e comunicados internos:** Divulgar os valores e boas práticas em locais visíveis na empresa.
- **Campanhas motivacionais:** Atividades e incentivos que integrem os colaboradores em torno dos objetivos organizacionais.

6.2 Citar um processo que é acompanhado e o(s) indicador(es) utilizado(s), descrevendo a unidade de medida.

PROCESSO	**INDICADOR**	**UNIDADE DE MEDIDA**
Gerenciamento de Estoques	Produtos com estoque baixo	Produtos com estoque baixo
	Tempo de reposição de produtos	Dias úteis
Operações de venda	Taxa de vendas finalizadas	Percentual (%)
	Satisfação do cliente	Índice de satisfação (1-5)

6.3 Elaborar o fluxograma de um processo, considerando p modelo da Ficha de Diagnóstico/Levantamento de áreas oferecido.



6.4 Sugestão de melhoria no processo, considerando as ferramentas da qualidade – justificar a melhoria através da ferramenta utilizada para propor a mudança.

Problema identificado:
A atualização manual do estoque após o recebimento é lenta e propensa a erros.

Ferramenta da qualidade aplicada:
Diagrama de Ishikawa (Causa e Efeito) foi utilizado para identificar os fatores que contribuem para atrasos na atualização de estoques, como falhas humanas, falta de integração entre sistemas e demora na conferência.

Solução proposta:
Automatizar a atualização de estoque através de um sistema integrado que:
Use códigos de barras para registrar a entrada de produtos.
Atualize automaticamente os dados no sistema SIGAS.
Emita alertas para discrepâncias detectadas.

Justificativa:
Essa solução reduzirá o tempo gasto na atualização, minimizará erros humanos e melhorará a precisão das informações do estoque, contribuindo para maior eficiência operacional e satisfação do cliente.

7. Conclusão

O projeto SIGAS (Sistema de Gerenciamento Arte Sagrada) proporcionou uma experiência prática e enriquecedora, que contribuiu para o aprendizado técnico e o desenvolvimento de uma solução eficiente para a gestão de informações. Durante a execução, enfrentei desafios relacionados à estruturação do banco de dados, integração de funcionalidades e ajustes nos processos, o que exigiu criatividade e aprimoramento de habilidades técnicas.

O SIGAS mostrou-se uma solução funcional, permitindo o gerenciamento de pessoas, produtos e operações de forma organizada. Funcionalidades geração de relatórios e automação de processos agregam confiabilidade e praticidade ao sistema. Essas características tornam o SIGAS um exemplo real de como ferramentas tecnológicas podem atender a demandas específicas.

Apesar das dificuldades, como erros técnicos e restrições de tempo, o projeto trouxe importantes contribuições acadêmicas e práticas. Ele reforçou a relevância de um planejamento adequado e boa organização no desenvolvimento de sistemas.

Por fim, o SIGAS vai além de um produto técnico, representando uma experiência formativa essencial para futuros desafios profissionais e acadêmicos. Ele serve como base para aprimoramentos e inspiração, demonstrando a aplicabilidade de soluções tecnológicas para resolver problemas de gestão e otimizar processos.