Laboratório de Banco de Dados Avançado

Elementrix

LMNTRX Shoes

Alice Queiróz de Oliveira RGM: 38689880

Victor Augusto Santos da Silva RGM: 35206225

Ryan Oliveira Campos RGM: 38603993

Gustavo Henrique Perez Aguiar RGM: 38583194 Jessica Pinheiro Ferreira RGM: 32938985

SÃO PAULO

2024

**1. Planejamento e Iniciação**

1. **Definição de Objetivos:**

Criar um site de e-commerce especializado na venda de tênis, com funcionalidades de gerenciamento de produtos, carrinho de compras, pagamento online e um sistema de banco de dados para armazenar informações dos produtos e usuários

1. **Identificação de Stakeholders:**

* Usuários Finais: Clientes interessados na compra de tênis, distribuídos em diversas faixas etárias e de gênero.
* Equipe Administrativa: Inclui o gerente, atendentes comerciais, e o dono da empresa.
* Equipe Técnica: Desenvolvedores e equipe de suporte responsável pela manutenção e atualização do sistema.

1. **Escopo do Projeto:**

**Escopo Funcional:**

* Desenvolvimento de uma página de listagem de produtos com opções de filtragem e ordenação.
* Implementação de um sistema de busca com funcionalidades de autocompletar e busca avançada.
* Criação de páginas detalhadas para cada produto, permitindo avaliações e comentários de clientes.
* Implementação de um carrinho de compras dinâmico e um sistema seguro de login/registro de usuários.
* Integração com gateways de pagamento para processar transações de maneira segura.
* Desenvolvimento de um sistema de gerenciamento de inventário e painel de administração para controle de produtos, usuários e pedidos.

**Escopo Não Funcional:**

* Garantir que o site tenha alta disponibilidade (80%) e suporte para até 10.000 usuários simultâneos.
* Implementar medidas de segurança robustas, incluindo criptografia e proteção contra-ataques como SQL Injection e XSS.
* Assegurar a compatibilidade com os principais navegadores e a responsividade do site em dispositivos móveis.
* Manter o código modular e bem documentado para facilitar a manutenção futura.

**2. Levantamento de Análise de Requisitos**

1. **Requisitos Funcionais:**

* Página de Listagem de Produtos:

- Filtragem: Permitir que os usuários filtrem produtos por marca, tamanho, preço, categoria, e outras características relevantes.

- Ordenação: Oferecer opções para ordenar produtos por preço (crescente/decrescente), popularidade, lançamentos etc.

* Paginação:

Exibir um número limitado de produtos por página com navegação entre as páginas.

* Sistema de Busca:

- Pesquisa por Palavra-chave: Permitir que os usuários busquem produtos usando palavras-chave.

- Autocompletar: Sugerir produtos ou categorias enquanto o usuário digita na barra de busca.

* Busca Avançada:

Oferecer uma busca avançada que permita combinar múltiplos critérios (marca, faixa de preço, categoria).

* Páginas de Detalhes do Produto:

Informações Detalhadas: Exibir informações completas sobre o produto, incluindo descrição, especificações técnicas, imagens, e avaliações de clientes.

* Opções de Compra:

- Permitir a seleção de quantidade, tamanho, cor, e outras variantes do produto.

- Avaliações e Comentários: Exibir e permitir que os usuários deixem avaliações e comentários sobre o produto.

* Carrinho de Compras:

- Adição e Remoção de Produtos: Permitir que os usuários adicionem ou removam produtos do carrinho.

* Resumo do Pedido:

Exibir um resumo do pedido com detalhes como subtotal, impostos, e custos de envio.

* Atualização em Tempo Real:

Atualizar o total do carrinho automaticamente conforme os produtos são adicionados ou removidos.

* Sistema de Login/Registro de Usuário:

- Registro de Usuário: Permitir que novos usuários se registrem usando e-mail ou redes sociais.

- Login: Autenticação de usuários registrados.

- Recuperação de Senha: Fornece um sistema de recuperação de senha via e-mail.

* Perfil de Usuário:

Permitir que os usuários visualizem e editem suas informações pessoais e histórico de pedidos.

* Processamento de Pagamento Seguro:

- Integração com Gateways de Pagamento: Suportar múltiplas formas de pagamento (cartão de crédito, débito, pix etc.).

- Criptografia de Dados: Garantir que todas as transações sejam realizadas de forma segura, utilizando criptografia.

- Confirmação de Pedido: Enviar confirmação do pedido por e-mail após a conclusão do pagamento.

* Sistema de Gerenciamento de Inventário:

- Controle de Estoque: Atualizar automaticamente o estoque de produtos conforme as vendas são realizadas.

- Notificações de Estoque: Notificar o administrador sobre níveis baixos de estoque.

* Gerenciamento de Fornecedores:

Manter registros de fornecedores e prazos de entrega.

* Painel de Administração:

- Gerenciamento de Produtos: Adicionar, editar, e remover produtos.

- Gerenciamento de Usuários: Gerenciar contas de usuários, incluindo a possibilidade de bloquear ou excluir contas.

- Gerenciamento de Pedidos: Visualizar, processar, e atualizar o status dos pedidos.

- Relatórios: Gerar relatórios sobre vendas, estoque, e comportamento do usuário.

1. **Stakeholder:**

Clientes interessados em calçados (Público-alvo: 3-12; 13-17; 18-34; 35-54, 54+, Feminino e Masculino).

Administradores do site (Atendente comercial, Gerente, Dono, Desenvolvedor).

1. **Requisitos Não Funcionais**

* Desempenho:

- Tempo de Resposta: As páginas devem carregar em menos de 3 segundos.

- Escalabilidade: O sistema deve suportar até 10.000 usuários simultâneos sem degradação de desempenho.

* Segurança:

- Autenticação e Autorização: Implementar autenticação segura para usuários e administradores, com diferentes níveis de acesso.

- Proteção contra-ataques: Proteger o sistema contra-ataques como SQL Injection, Cross-Site Scripting (XSS), e Cross-Site Request Forgery (CSRF).

* Backup de Dados:

Realizar backups automáticos dos dados diariamente.

* Usabilidade:

- Interface Amigável: A interface deve ser intuitiva e fácil de usar para todos os tipos de usuários.

* Compatibilidade:

Cross-Browser: O sistema deve ser compatível com os principais navegadores (Chrome, Firefox, Safari, Edge).

* Responsividade:

O site deve ser responsivo e funcionar bem em dispositivos móveis, tablets e desktops.

* Manutenibilidade:

- Código Modular: O código deve ser modular e bem documentado para facilitar futuras manutenções e atualizações.

- Testes Automatizados: Implementar testes automatizados para garantir a qualidade do código.

* Disponibilidade:

O sistema deve estar disponível 80% do tempo.

* Recuperação de Falhas:

Em caso de falhas, o sistema deve ser capaz de se recuperar rapidamente sem perda de dados.

* Legalidade:

- Conformidade com GDPR:

- O sistema deve estar em conformidade com as regulamentações de proteção de dados (GDPR ou equivalente).

1. **Análise de Requisitos:**
2. **Priorização:**

Identificar quais funcionalidades são críticas para o lançamento inicial e quais podem ser implementadas em fases futuras.

1. **Desafios e Soluções:**

* Escalabilidade:

Implementação de uma arquitetura que suporte o crescimento futuro.

* Segurança**:**

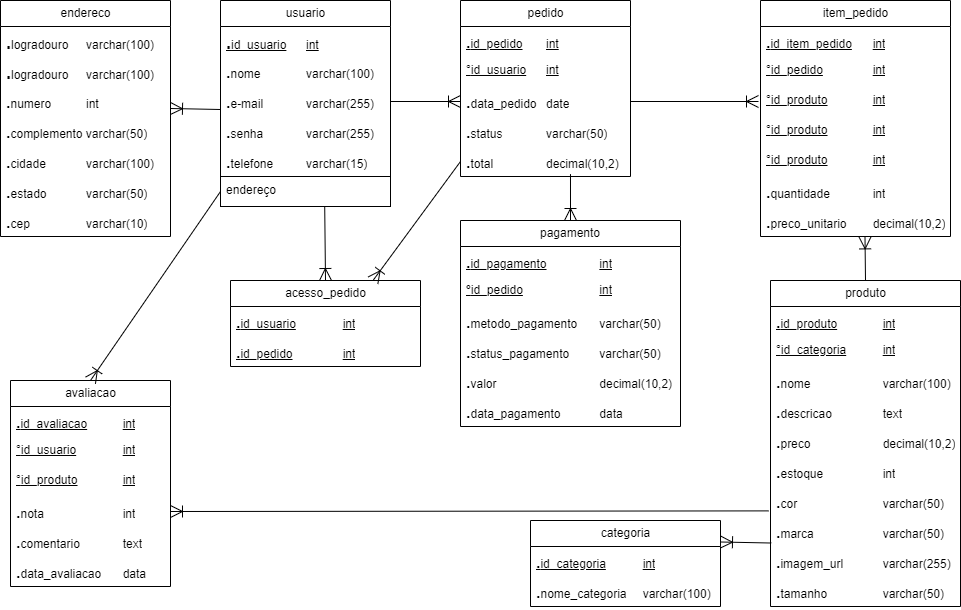
Utilização de boas práticas de segurança desde o início do desenvolvimento.

* Desempenho**:**

Otimizações para garantir tempos de resposta rápidos, mesmo sob alta carga.

**3. Modelagem Conceitual**

1. **Diagrama de Entidade-Relacionamento (ER):**

****

1. **Diagrama

   Descrição gerada automaticamenteDiagrama Entidade Relacionamento (DER):**
2. **Dicionário de Dados:**

* Usuário

- id\_usuario: int Primary Key, Auto Increment, not null

- nome: varchar(100) not null

- e-mail: varchar(255) Unique, not null

- senha: varchar(255) not null

- telefone: varchar(15) not null

- admin; vendedor; cliente role enum not null

Endereço (relacionado ao usuário):

- logradouro: varchar(100) not null

- numero: int not null

- complemento: varchar(50)

- cidade: varchar(100) not null

- estado: varchar(50) not null

- cep: varchar(10) not null

* Pedido

- id\_pedido: int Primary Key, Auto Increment, not null

- id\_usuario: int Foreign Key -> Usuario.id\_usuario, not null

- data\_pedido: date not null

- status: varchar(50) not null

- total: decimal(10,2) not null

* Pagamento

- id\_pagamento: int Primary Key, Auto Increment, not null

- id\_pedido: int Foreign Key -> Pedido.id\_pedido, not null

- metodo\_pagamento: varchar(50) not null

- status\_pagamento: varchar(50) not null

- valor: decimal(10,2) not null

- data\_pagamento: date not null

* Item Pedido

- id\_item\_pedido: int Primary Key, Auto Increment, not null

- id\_pedido: int Foreign Key -> Pedido.id\_pedido, not null

- id\_produto: int Foreign Key -> Produto.id\_produto, not null

- quantidade: int not null

- preco\_unitario: decimal(10,2) not null

* Produto

- id\_produto: int Primary Key, Auto Increment, not null

- id\_categoria: int Foreign Key -> Categoria.id\_categoria, not null

- nome: varchar(100) not null

- descricao: text not null

- preco: decimal(10,2) not null

- estoque: int not null

- cor: varchar(50) not null

- marca: varchar(50) not null

- imagem\_url: varchar(255) not null

- tamanho: varchar(50) not null

* Categoria

id\_categoria: int Primary Key, Auto Increment, not null

nome\_categoria: varchar(100) not null

* Avaliação

id\_avaliacao: int Primary Key, Auto Increment, not null

id\_usuario: int Foreign Key -> Usuario.id\_usuario, not null

id\_produto: int Foreign Key -> Produto.id\_produto, not null

nota: int not null

comentario: text

data\_avaliacao: date not null

* Acesso Pedido

id\_pedido: int not null,

id\_usuario: int not null,

foreign key: (id\_pedido) references pedido(id\_pedido),

foreign key: (id\_usuario) references usuario(id\_usuario),

primary key: (id\_pedido, id\_usuario)

* Relacionamentos:

- **Usuário** - **Pedido**: Um usuário pode fazer vários pedidos, mas um pedido só pode ter um usuário (**N:1**).

- **Pedido** - **Pagamento**: Um pedido tem apenas um pagamento e um pagamento pode ter apenas um pagamento (**1:1**).

- **Pedido** - **Item Pedido**: Um pedido pode ter vários itens e um item pode ter vários pedidos (**M:N**).

- **Produto** - **Item Pedido**: Um produto pode estar presente em vários itens de pedido e um item pedido pode ter vários produtos (**M:N**).

- **Produto** - **Categoria**: Um produto pertence a várias categorias, é uma categoria pode ter vários produtos (**M:N**).

- **Usuário** - **Avaliação**: Um usuário pode fazer várias avaliações, mas uma avaliação pode ter apenas um usuário (**M:1**).

- **Produto** - **Avaliação**: Um produto pode receber várias avaliações, mas uma avaliação pode ter apenas um produto (**M:1**).

- **Acesso Pedido** - **Pedido** e **Usuário**: Um usuário pode ter acesso a vários pedidos através do acesso pedido, mas um pedido pode ter acesso a somente um usuário (**M:1**)

**4. Modelagem Lógica**

1. **Modelo Lógico de Dados:**

create database LMNTRX;

use LMNTRX;

create table usuario (

id\_usuario int not null auto\_increment primary key,

nome varchar(100) not null,

email varchar(255) unique not null,

senha varchar(255) not null,

telefone varchar(15) not null,

role ENUM('admin', 'vendedor', 'cliente') NOT NULL

);

create table endereco (

id\_endereco int primary key auto\_increment not null,

id\_usuario int not null,

logradouro varchar(100) not null,

numero int not null,

complemento varchar(50),

cidade varchar(100) not null,

estado varchar(50) not null,

cep varchar(10) not null,

foreign key (id\_usuario) references usuario(id\_usuario)

);

create table categoria (

id\_categoria int primary key auto\_increment not null,

nome\_categoria varchar(100) not null

);

create table produto (

id\_produto int primary key auto\_increment not null,

id\_categoria int not null,

nome varchar(100) not null,

descricao text not null,

preco decimal(10, 2) not null,

estoque int not null,

cor varchar(50) not null,

marca varchar(50) not null,

imagem\_url varchar(255) not null,

tamanho varchar(50) not null,

foreign key (id\_categoria) references categoria(id\_categoria)

);

create table pedido (

id\_pedido int not null auto\_increment primary key,

id\_usuario int not null,

data\_pedido date not null,

status varchar(50) not null,

total decimal(10, 2) not null,

foreign key (id\_usuario) references usuario(id\_usuario)

);

create table pagamento (

id\_pagamento int primary key auto\_increment not null,

id\_pedido int not null,

metodo\_pagamento varchar(50) not null,

status\_pagamento varchar(50) not null,

valor decimal(10, 2) not null,

data\_pagamento date not null,

foreign key (id\_pedido) references pedido(id\_pedido)

);

create table item\_pedido (

id\_item\_pedido int primary key auto\_increment not null,

id\_pedido int not null,

id\_produto int not null,

quantidade int not null,

preco\_unitario decimal(10, 2) not null,

foreign key (id\_pedido) references pedido(id\_pedido),

foreign key (id\_produto) references produto(id\_produto)

);

create table avaliacao (

id\_avaliacao int primary key auto\_increment not null,

id\_usuario int not null,

id\_produto int not null,

nota int not null,

comentario text,

data\_avaliacao date not null,

foreign key (id\_usuario) references usuario(id\_usuario),

foreign key (id\_produto) references produto(id\_produto)

);

create table acesso\_pedido (

id\_pedido int not null,

id\_usuario int not null,

foreign key (id\_pedido) references pedido(id\_pedido),

foreign key (id\_usuario) references usuario(id\_usuario),

primary key (id\_pedido, id\_usuario)

);

1. **Normalização:**

**1. Análise das Tabelas:**

* pedido, endereco, usuario, produto, pagamento, item\_pedido e avaliacao: Todas as tabelas estão normalizadas, com dependências diretas em suas chaves primárias.

**3. Modelagem Física**

1. **Modelo Físico de Dados:**
2. **Script de Criação:**

create table usuario (

id\_usuario int not null auto\_increment primary key,

nome varchar(100) not null,

email varchar(255) unique not null,

senha varchar(255) not null,

telefone varchar(15) not null,

role ENUM('admin', 'vendedor', 'cliente') NOT NULL

);

create table endereco (

id\_endereco int primary key auto\_increment not null,

id\_usuario int not null,

logradouro varchar(100) not null,

numero int not null,

complemento varchar(50),

cidade varchar(100) not null,

estado varchar(50) not null,

cep varchar(10) not null,

foreign key (id\_usuario) references usuario(id\_usuario)

);

create table categoria (

id\_categoria int primary key auto\_increment not null,

nome\_categoria varchar(100) not null

);

create table produto (

id\_produto int primary key auto\_increment not null,

id\_categoria int not null,

nome varchar(100) not null,

descricao text not null,

preco decimal(10, 2) not null,

estoque int not null,

cor varchar(50) not null,

marca varchar(50) not null,

imagem\_url varchar(255) not null,

tamanho varchar(50) not null,

foreign key (id\_categoria) references categoria(id\_categoria)

);

create table pedido (

id\_pedido int not null auto\_increment primary key,

id\_usuario int not null,

data\_pedido date not null,

status varchar(50) not null,

total decimal(10, 2) not null,

foreign key (id\_usuario) references usuario(id\_usuario)

);

create table pagamento (

id\_pagamento int primary key auto\_increment not null,

id\_pedido int not null,

metodo\_pagamento varchar(50) not null,

status\_pagamento varchar(50) not null,

valor decimal(10, 2) not null,

data\_pagamento date not null,

foreign key (id\_pedido) references pedido(id\_pedido)

);

create table item\_pedido (

id\_item\_pedido int primary key auto\_increment not null,

id\_pedido int not null,

id\_produto int not null,

quantidade int not null,

preco\_unitario decimal(10, 2) not null,

foreign key (id\_pedido) references pedido(id\_pedido),

foreign key (id\_produto) references produto(id\_produto)

);

create table avaliacao (

id\_avaliacao int primary key auto\_increment not null,

id\_usuario int not null,

id\_produto int not null,

nota int not null,

comentario text,

data\_avaliacao date not null,

foreign key (id\_usuario) references usuario(id\_usuario),

foreign key (id\_produto) references produto(id\_produto)

);

create table acesso\_pedido (

id\_pedido int not null,

id\_usuario int not null,

foreign key (id\_pedido) references pedido(id\_pedido),

foreign key (id\_usuario) references usuario(id\_usuario),

primary key (id\_pedido, id\_usuario)

);

* Armazenamento de Dados:

O SGBD escolhido para este modelo é o MySQL , que utiliza o mecanismo de armazenamento InnoDB por padrão, garantindo suporte a transações e integridade referencial entre as tabelas.

Tipos de dados como DECIMAL, VARCHARe TEXTforam escolhidos de acordo com a natureza dos dados (numéricos ou de texto) para otimização do armazenamento e consulta.

1. **População de Dados:**

insert into usuario (nome, email, senha, telefone, role)

values ('admin', 'admin@gmail.com', 'senhaadmin', '1234567890', 'admin'), ('vendedor', 'vendedor@gmail.com', 'senhavendedor', '0987654321', 'vendedor'), ('cliente', 'cliente@gmail.com', 'senhacliente', '1122334455', 'cliente');

insert into categoria (nome\_categoria)

values ('automobilismo'), ('aventura'), ('basquete'), ('caminhada'), ('corrida'), ('esporte-de-quadra'), ('streetwar'), ('tennis e squash'), ('ofertas');

insert into produto (nome, marca, descricao, preco, estoque, id\_categoria, cor, imagem\_url, tamanho)

values ('tênis qix skate retrô am preto cinza branco', 'qix', 'tênis de skate retrô em cores clássicas', 239.99, 20, (select id\_categoria from categoria where nome\_categoria = 'streetwar'), 'preto', 'url\_da\_imagem', '42'), ('tênis puma mapf1 neo cat - branco', 'puma', 'tênis inspirado no automobilismo', 499.90, 15, (select id\_categoria from categoria where nome\_categoria = 'automobilismo'), 'branco', 'url\_da\_imagem', '41'), ('tênis under armour basquete spawn 3 masculino - vermelho+branco', 'under armour', 'tênis de basquete de alta performance', 237.49, 10, (select id\_categoria from categoria where nome\_categoria = 'basquete'), 'vermelho+branco', 'url\_da\_imagem', '43');

insert into pedido (id\_usuario, data\_pedido, status, total)

values (2, curdate(), 'pendente', 499.99);

insert into acesso\_pedido (id\_pedido, id\_usuario) select p.id\_pedido, 1 from pedido p where p.id\_usuario != 1;

insert into acesso\_pedido (id\_pedido, id\_usuario) select p.id\_pedido, 2 from pedido p where p.id\_usuario = 2;

1. **Gerenciamento de Permissões:**

create table usuario (

id\_usuario int not null auto\_increment primary key,

nome varchar(100) not null,

email varchar(255) unique not null,

senha varchar(255) not null,

telefone varchar(15) not null,

role enum('admin', 'vendedor', 'cliente') not null

);

create table acesso\_pedido (

id\_pedido int not null,

id\_usuario int not null,

foreign key (id\_pedido) references pedido(id\_pedido),

foreign key (id\_usuario) references usuario(id\_usuario),

primary key (id\_pedido, id\_usuario)

);

insert into acesso\_pedido (id\_pedido, id\_usuario)

select p.id\_pedido, 1 from pedido p where p.id\_usuario != 1;

insert into acesso\_pedido (id\_pedido, id\_usuario)

select p.id\_pedido, 2 from pedido p where p.id\_usuario = 2;