Sistema de E-commerce de Hortifrut

Integrante: Lucas Antonio Salomão Jarek

Integrante: Vinicius da Costa Silva

Integrante: Wallacy Gabriel Alves Bandeira

Documentação do Projeto PHP Acadêmico

Introdução e Objetivos

Este documento apresenta a documentação técnica e funcional de um projeto acadêmico desenvolvido em PHP, com foco em um sistema de e-commerce de hortifruti. O projeto foi concebido para demonstrar a aplicação de conceitos de desenvolvimento web, incluindo arquitetura MVC, manipulação de banco de dados MySQL com PDO, gerenciamento de sessões e cookies, e implementação de operações CRUD (Create, Read, Update, Delete).

O objetivo principal deste projeto é fornecer uma plataforma funcional para a venda de produtos hortifrutigranjeiros online, permitindo que usuários se cadastrem, naveguem por produtos, adicionem itens ao carrinho de compras e finalizem pedidos. Além disso, o sistema inclui funcionalidades administrativas para o gerenciamento de produtos, clientes e funcionários.

Os requisitos para o desenvolvimento deste projeto foram definidos pelo Professor Fábio Kravetz na disciplina de Desenvolvimento de Sistemas, com ênfase na aplicação prática de tecnologias e padrões de codificação. Este documento visa detalhar a implementação desses requisitos, a arquitetura do sistema, os fluxos de trabalho, e as tecnologias empregadas, servindo como um guia completo para compreensão e manutenção do software.

Arquitetura do Sistema

O sistema adota o padrão arquitetural **MVC (Model-View-Controller)**, que promove a separação de responsabilidades, facilitando a organização do código, a manutenção e a escalabilidade. A estrutura do projeto é organizada da seguinte forma:

- **Model**: Responsável pela lógica de negócio e interação com o banco de dados. Os modelos (`app/models/`) encapsulam as operações de persistência e validação de dados para entidades como `Produto`, `Cliente`, `Funcionario`, etc.
- **View**: Responsável pela apresentação dos dados ao usuário. As views (`app/views/`) são arquivos PHP que contêm a estrutura HTML e a lógica de exibição, recebendo dados dos controllers para renderizar a interface do usuário.
- **Controller**: Atua como intermediário entre o Model e a View, processando as requisições do usuário, interagindo com os modelos para obter ou manipular dados, e selecionando a view apropriada para exibir a resposta. Os controllers (`app/controllers/`) gerenciam o fluxo da aplicação.

Tecnologias e Versões

- **Linguagem de Programação**: PHP (versão compatível com as funções `password_hash()` e `password_verify()`, geralmente PHP 5.5+).
- **Banco de Dados**: MySQL (utilizado para persistência de dados, com o esquema definido em `app/database/schema.sql`).

- **Driver de Banco de Dados**: PDO (PHP Data Objects) para uma conexão segura e padronizada com o MySQL.
- **Front-end**: HTML5, CSS3 (estilos customizados em `app/public/assets/css/`), JavaScript (com um arquivo `cart.js` para funcionalidades de carrinho).
- **Servidor Web**: Ambiente LAMP/XAMPP/WAMP (Apache, MySQL, PHP) é o esperado para execução local.

Diagrama de Arquitetura

O diagrama a seguir ilustra a arquitetura MVC do sistema, destacando a interação entre os componentes:

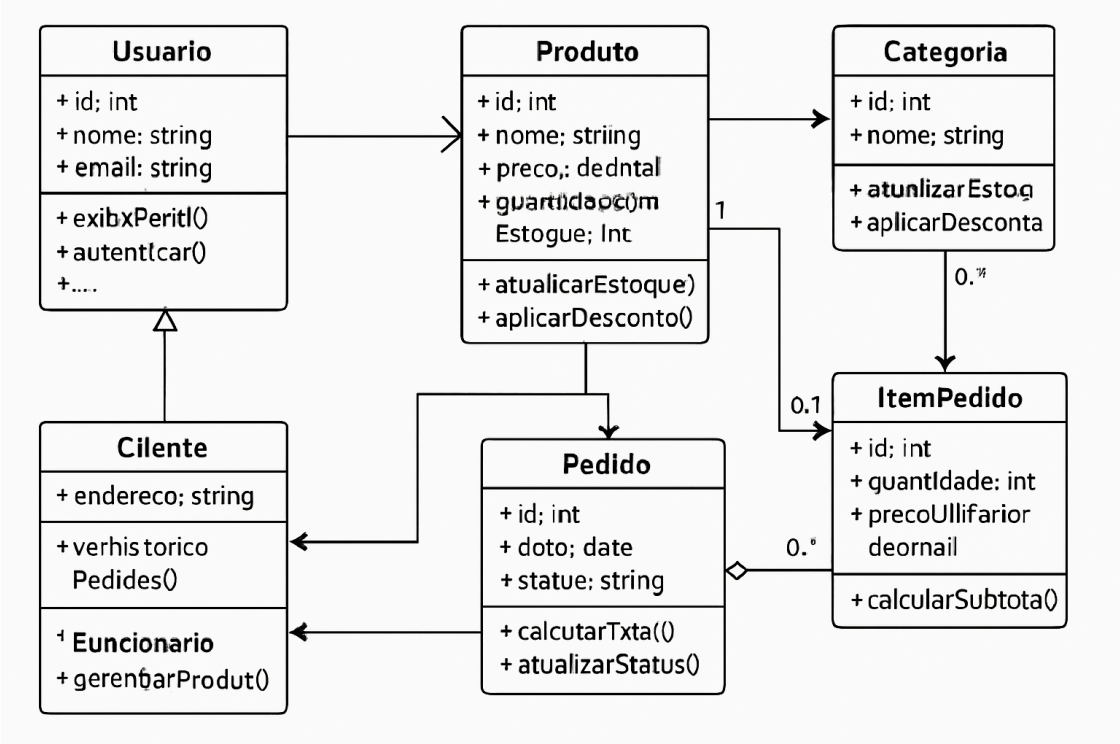

Diagrama de Arquitetura MVC

Requisitos Funcionais e Não Funcionais

Os requisitos do projeto foram extraídos do documento fornecido pelo professor e são detalhados a seguir, juntamente com sua rastreabilidade para as seções correspondentes no código-fonte e neste documento.

Tabela de Rastreabilidade de Requisitos

ID do Requisito	Descrição do Requisito	Local no Código (arquivo/linha/trecho)	Página correspondente no documento gerado

RQ01	CRUD completo (CREATE, READ, UPDATE e DELETE): O sistema deve possuir no mínimo três CRUD’s, permitindo o cadastro, alteração, deleção e listagem de informações relevantes para o projeto.	app/controllers/ProdutoController.php, app/models/ProdutoModels.php, app/views/produto/index.php, create.php, edit.php
RQ02	No mínimo três páginas de navegação abertas (antes de fazer o login), sem contar com a página de login do usuário.	app/views/home/index.php, app/views/home/sobre.php, app/views/home/contato.php
RQ03	Sistema de Login: É necessário criar um sistema de login com um botão para se cadastrar e uma opção de “Lembrar senha”; Utilização das funções password_hash() e password_verify() é desejável.	app/controllers/AuthController.php, app/models/AuthModels.php, app/views/auth/login.php, app/views/auth/register.php, app/lib/Auth.php
RQ04	Sessões e Cookies: É obrigatório o uso de sessões e cookies.	app/public/index.php (session_start()), app/lib/Auth.php

RQ05	MVC (Model, View, Controller): A organização do projeto deve seguir a arquitetura MVC, separando adequadamente a lógica de negócio (Model), a interface do usuário (View) e a camada de controle (Controller) com a implementação da validação dos dados de entrada.	app/controllers/, app/models/, app/views/, app/public/index.php
RQ06	Utilizar estruturas de controle e laços de repetição: Utilize laços de repetição (for, foreach, while ou do-while) e estruturas de controle (if, elseif, else, switch ou match) de maneira apropriada no sistema, propiciando um fluxo lógico e eficiente; Realizar todas as validações via linguagem PHP.	Diversos arquivos .php em controllers e views. Ex: app/controllers/ProdutoController.php, app/views/produto/index.php
RQ07	PDO: Utilize o PDO para realizar as transações no	app/config/database.php, app/lib/Databas

	banco de dados MySQL.	e.php, app/models/
RQ08	HTML: Não é permitido o uso de Tags depreciadas no HTML;	Todos os arquivos .php em app/views/
RQ09	CSS: Utilize o CSS, em todas as páginas, para deixar seu projeto com uma interface mais amigável para o usuário.	app/public/assets/css/
RQ10	Javascript: A utilização de Javascript é permitida desde que não ultrapasse 30% do código implementado; API's não são permitidas.	app/public/assets/js/cart.js
RQ11	Banco de dados – Tabelas: Avaliação da qualidade das tabelas criadas no banco de dados; Avaliação das querys criadas e utilizadas no projeto.	app/database/schema.sql, app/models/
RQ12	Modularização: É desejável que o código seja implementado de forma modularizada.	app/controllers/, app/models/, app/lib/
RQ13	Documentação: Arquivo PDF	Este documento

	com a explicação e comentários pertinentes relativos a páginas e/ou funções que sejam essenciais para o funcionamento do código.
RQ14	Funcionamento: Código funcionando de maneira adequada e atendendo os requisitos mínimos descritos.
RQ15	Defesa: A defesa equivale a 80% da nota do projeto.

Diagramas

Diagrama de Entidade-Relacionamento (ERD)

O diagrama ERD a seguir representa a estrutura do banco de dados do sistema, mostrando as entidades e seus relacionamentos:

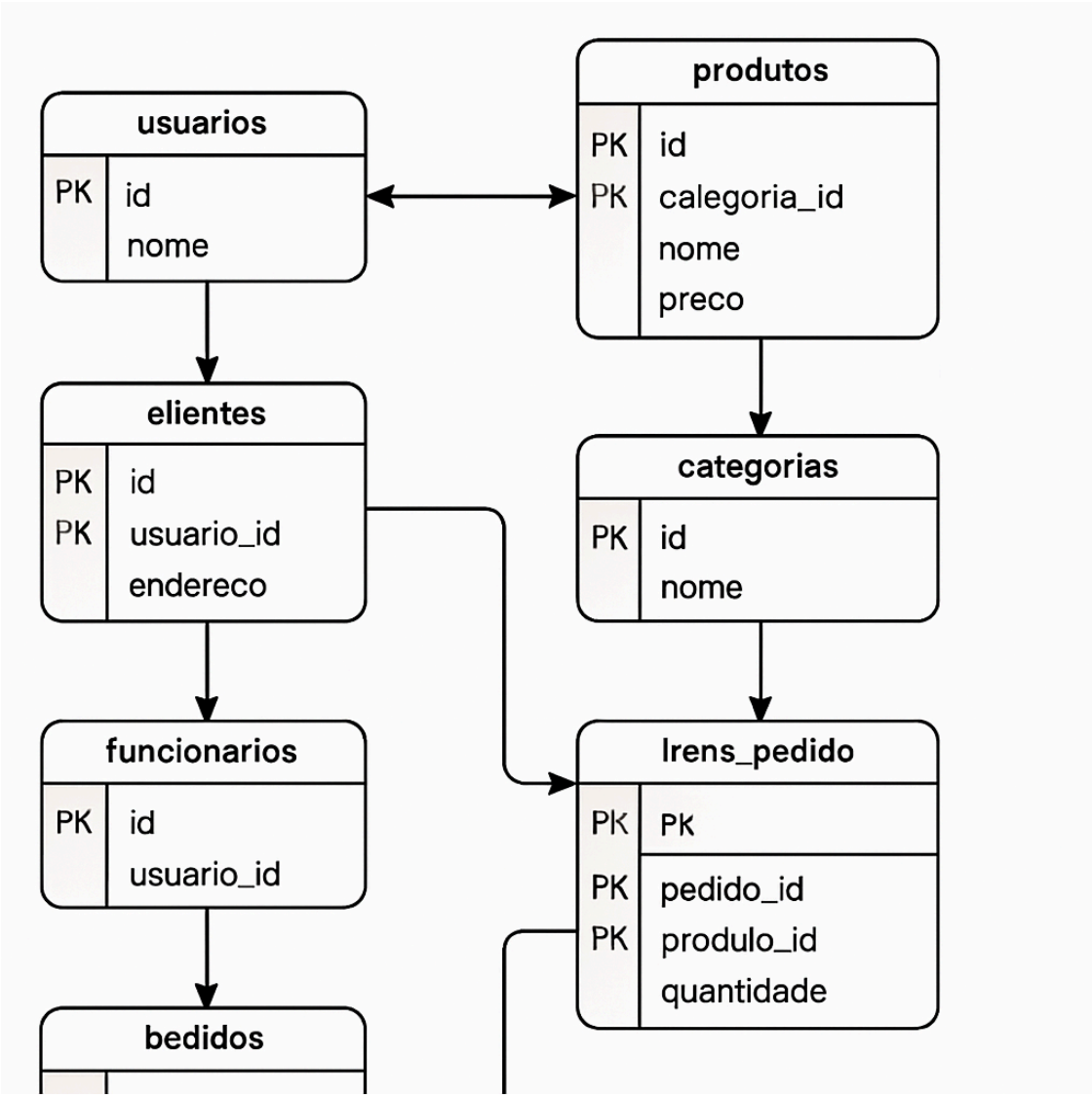

Diagrama ERD

Diagrama de Classes UML

O diagrama de classes UML a seguir ilustra as principais classes do sistema, seus atributos, métodos e os relacionamentos entre elas:

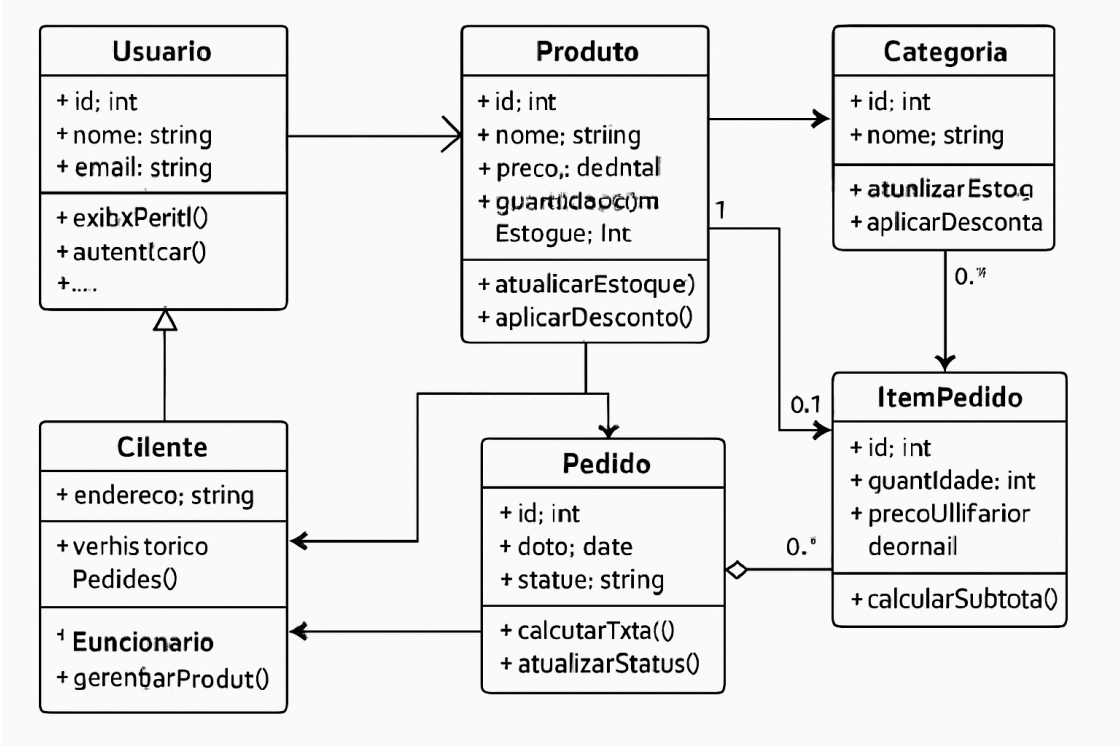

Diagrama de Classes UML

Manual do Usuário/Instalação

Pré-requisitos

Para executar o sistema localmente, são necessários os seguintes componentes:

- Servidor Web (Apache, Nginx, etc.)
- PHP (versão 5.5 ou superior)
- MySQL (ou MariaDB)
- Composer (para gerenciamento de dependências PHP, se aplicável)

Recomenda-se a utilização de um ambiente de desenvolvimento integrado como XAMPP, WAMP ou MAMP, que já incluem Apache, MySQL e PHP.

Instalação Local

1. **Clone o repositório do projeto:**

   ```
   git clone <URL_DO_REPOSITORIO>
   ```

   (Substituir `<URL_DO_REPOSITORIO>` pelo link real do repositório do projeto)

2. **Configure o servidor web:** Mova o conteúdo do diretório `app/public` para o diretório raiz do seu servidor web (ex: `htdocs` no XAMPP).

3. **Crie o banco de dados:** Acesse o MySQL (via phpMyAdmin ou linha de comando) e crie um banco de dados com o nome `hortifruti_bd` (conforme `app/config/database.php`).

4. **Importe o esquema do banco de dados:** Importe o arquivo `app/database/schema.sql` para o banco de dados recém-criado.

5. **Importe os dados iniciais (opcional):** Importe o arquivo `app/database/seeds.sql` para popular o banco de dados com dados de exemplo.

6. **Ajuste as configurações de conexão (se necessário):** Verifique o arquivo `app/config/database.php` e ajuste as credenciais de acesso ao banco de dados (`$user`, `$pass`) se forem diferentes de `root` e senha em branco.

7. **Acesse o sistema:** Abra seu navegador e acesse `http://localhost/` (ou o endereço configurado para o seu servidor web).

Acesso com Usuário Admin

Após a instalação e importação dos dados iniciais (`seeds.sql`), um usuário administrador padrão estará disponível para acesso. As credenciais são:

- **Usuário**: `admin@example.com` (ou outro e-mail definido no `seeds.sql`)
- **Senha**: `admin123` (ou outra senha definida no `seeds.sql`)

Recomenda-se alterar a senha padrão após o primeiro login por questões de segurança.

Testes

Os testes foram realizados manualmente para verificar a funcionalidade de cada módulo do sistema. Os procedimentos de teste incluíram:

- **Testes de CRUD**: Verificação das operações de criação, leitura, atualização e exclusão para Produtos, Clientes e Funcionários.
- **Testes de Autenticação**: Validação do fluxo de registro, login e logout, incluindo casos de sucesso e falha.
- **Testes de Navegação**: Verificação da acessibilidade e funcionalidade das páginas abertas e restritas.
- **Testes de Carrinho de Compras**: Adição, remoção e atualização de itens no carrinho, e simulação do processo de checkout.
- **Testes de Validação**: Verificação das validações de entrada de dados em formulários.

Não foram implementados testes automatizados (unitários, de integração) neste projeto. A validação foi realizada através de testes de caixa preta, simulando o comportamento do usuário final.

Conclusão

O desenvolvimento deste projeto acadêmico proporcionou uma valiosa experiência na aplicação prática de conceitos de engenharia de software e desenvolvimento web com PHP. A adoção da arquitetura MVC demonstrou ser eficaz na organização do código e na separação de responsabilidades, resultando em um sistema mais modular e fácil de manter. A utilização de PDO para a interação com o banco de dados garantiu maior segurança e flexibilidade nas operações.

Entre as lições aprendidas, destaca-se a importância do planejamento da arquitetura e da modelagem de dados antes da codificação, o que contribui significativamente para a robustez e escalabilidade do sistema. A experiência com o gerenciamento de sessões e cookies, bem como a implementação de funcionalidades CRUD, reforçou a compreensão sobre o ciclo de vida de uma aplicação web.

Para futuras melhorias, sugere-se a implementação de testes automatizados para garantir a qualidade do código, aprimoramento da interface do usuário com frameworks CSS modernos, e a exploração de bibliotecas JavaScript para uma experiência mais interativa. A integração com APIs de pagamento reais e a implementação de um sistema de relatórios mais robusto também seriam adições valiosas.