Conteúdo

[Projeto 1](#_Toc158111714)

[Título e objetivo 1](#_Toc158111715)

[Autor, data, contexto do trabalho 1](#_Toc158111716)

[Descrição do problema 1](#_Toc158111717)

[Ferramentas e versões 1](#_Toc158111718)

[Análise de dados 1](#_Toc158111719)

[Descrição de processos 1](#_Toc158111720)

[Dicionário 1](#_Toc158111721)

[String connection, instâncias, bases de dados 1](#_Toc158111722)

[Lista de tabelas, campos e tipos 1](#_Toc158111723)

[Namespaces 1](#_Toc158111724)

[Classes e métodos 1](#_Toc158111725)

[Outras informações 1](#_Toc158111726)

# Projeto

## Título e objetivo

PHPShop, criação de uma webstore base, adaptável para qualquer tipo de estabelecimento.

## Autor, data, contexto do trabalho

Felippe de Almeida Santana, 30-04-2024, Projeto final programação EFA-0123 IEFP

## Descrição do problema

A aplicação web, desenvolvida em PHP, deve ser altamente adaptável às necessidades específicas do cliente. Com o PHPShop, apenas pequenas modificações são necessárias para customizar o sistema de acordo com as peculiaridades do negócio do cliente. Isso possibilita uma implementação rápida e eficiente, garantindo que o sistema esteja pronto e funcional para uso imediato.

## Ferramentas e versões

PHP 8.3.6

Composer 2.7.1

Bootstrap 5.3

FontAwesome 6.5.2

GitHub

Laravel Valet (Hospeda a página para desenvolvimento)

Visual Studio Code

**Packages:**

PHPMailer - Versão: 6.9.1

# Análise de dados



# Descrição de processos

1. **Registo e Login:**
   * O utilizador acede a página de registo através do AuthenticationController. Ele preenche seus dados e submete o formulário.
   * O controlador valida os dados inseridos e cria uma conta de utilizador.
   * O utilizador pode então fazer login através do mesmo controlador, inserindo suas credenciais.
   * Após a autenticação bem-sucedida, o utilizador é redirecionado de volta para a página inicial ou para a página que estava tentando aceder.
2. **Navegação na Loja:**
   * O utilizador pode navegar pelos produtos da loja acedendo a diferentes páginas gerenciadas pelo MainController e pelo ProductController.
   * O ProductController é responsável por exibir informações detalhadas sobre os produtos, como nome, descrição, preço, etc.
3. **Adição de Produtos ao Carrinho:**
   * Quando o utilizador deseja comprar um produto, ele pode adicioná-lo ao carrinho clicando em um botão específico.
   * Isso desencadeia a adição do produto ao carrinho através do CartController.
4. **Gerenciamento do Carrinho:**
   * O utilizador pode visualizar o conteúdo do carrinho, atualizar quantidades, remover itens ou prosseguir para o checkout.
   * Todas essas ações são gerenciadas pelo CartController.
   * O utilizador só pode proceder ao checkout se estiver logado.
5. **Checkout e Processamento de Pedidos:**
   * Quando o utilizador decide finalizar a compra, ele é direcionado para o processo de checkout, que é controlado pelo OrderController.
   * Aqui, o utilizador insere informações de entrega, escolhe o método de pagamento e revisa o pedido antes de confirmar.
   * Após a confirmação, o OrderController cria um pedido e envia uma confirmação por e-mail.
6. **Perfil do Utilizador e Histórico de Pedidos:**
   * O UserController permite que o utilizador acesse e atualize seu perfil, incluindo informações pessoais e preferências.
   * Além disso, o utilizador pode visualizar seu histórico de pedidos, que é gerenciado pelo mesmo controlador.
7. **Avaliação de Produtos:**
   * O utilizador pode deixar comentários ou avaliações nos produtos através do ReviewController.
8. **Logout:**
   * Quando o utilizador deseja sair da sua conta, ele pode clicar no botão de logout, que é tratado pelo AuthenticationController.

# Dicionário

## String connection, instâncias, bases de dados

MYSQL\_SERVER: 62.28.39.135:3306

MYSQL\_DATABASE: php\_shop

MYSQL\_USER: Raquel

MYSQL\_PASS: Silva1234

## Lista de tabelas, campos e tipos

**Clientes (cliente):** Armazena informações sobre os utilizadores registados.

id INT

nome\_completo VARCHAR (100)

email VARCHAR (50)

senha (criptografada) VARCHAR(255)

morada VARCHAR(100)

cidade VARCHAR(50)

telefone VARCHAR(50)

personal\_url VARCHAR(50)

ativo TINYINT

created\_at DATETIME

updated\_at DATETIME

deleted\_at DATETIME

**Produtos (produto):** Contém informações sobre os produtos.

id INT

nome VARCHAR (100)

descricao VARCHAR (255)

preco DECIMAL(6,2)

quantidade INT

categoria\_id INT

imagem VARCHAR(255)

ativo TINYINT

created\_at DATETIME

updated\_at DATETIME

deleted\_at DATETIME

**Categorias (categoria):** Armazena informações sobre as categorias dos produtos.

id INT

nome VARCHAR (100)

descricao VARCHAR (255)

**Pedidos (pedido):** Regista informações sobre os pedidos feitos pelos utilizadores.

num\_pedido INT

data\_pedido DATETIME

morada \_entrega VARCHAR(100)

observacoes VARCHAR(255)

cliente\_id INT

metodo\_pagamento\_id INT

estado\_pagamento\_id INT

estado\_entrega\_id INT

cod\_rastreamento VARCHAR(50)

ativo TINYINT

created\_at DATETIME

updated\_at DATETIME

deleted\_at DATETIME

**Estados de Entrega (estado\_entrega):** Armazena informações sobre o estado do pedido. (processando, enviado, entregue, não entregue, etc.).

id INT

nome VARCHAR (100)

descrição VARCHAR (255)

**Estados de Pagamento (estado\_pagamento):** Armazena informações sobre o estado do pagamento. (pendente, aprovado, rejeitado, etc.).

id INT

nome VARCHAR (100)

descrição VARCHAR (255)

**Itens do Pedido (item\_pedido):** Uma tabela de junção que relaciona pedidos a produtos.

num\_pedido INT

item\_id INT

quantidade INT

preco DECIMAL (6,2) (Devo guardar o preço no momento em que a venda foi feita)

**Carrinho de Compras (carrinho):** Armazena temporariamente os itens selecionados pelo utilizador antes de finalizar a compra.

item\_id INT

cliente\_id INT

quantidade INT

data\_adicao DATETIME

preco DECIMAL (6,2) (Serve para calcular o total do carrinho e manter o preço no qual foi adicionado ao carrinho)

session\_id VARCHAR(255) (Armazena o id da sessão atual em carrinhos anónimos)

**Comentários e Avaliações (review):** Armazena avaliações dos produtos.

id INT

cliente\_id INT

produto\_id INT

avaliação TINYINT (Nota do produto de 1 a 5)

texto VARCHAR (255)

data\_adicao DATETIME

**Histórico de Pedidos (historico\_pedido):** Serve para rastrear o histórico de pedidos dos utilizadores, mantém o registo de todos os pedidos anteriores.

num\_pedido INT

cliente\_id INT

**Métodos de Pagamento (metodo\_pagamento):** Armazena informações sobre os métodos de pagamento disponíveis.

id INT

tipo VARCHAR (50)

*OBS: Pode ser necessário adicionar mais campos para guardar informações adicionais referentes a tipos de pagamentos específicos, será analisado mais a frente no desenvolvimento.*

## Namespaces

core/classes

core/controllers

core/models

core/views

## Classes e métodos

**Controllers:**

**MainController:** Lida com a lógica das páginas comuns, como a página inicial.

voidindex(); // Invoca a view da página inicial.

**AuthenticationController:** Responsável por lidar com o registo, login, logout e todas as funcionalidades relacionadas à autenticação de utilizadores.

voidlogin(); // Invoca a view ‘login’ e os métodos responsáveis por validar se utilizador está registado.

void logout(); // Remove o utilizador da sessão atual.

void register(); // Invoca a view ‘register’, onde se inserem os dados de utilizador, e o método responsável por criar o utilizador, após fazer as verificações dos campos.

void confirm\_email(); // Invoca o método de validação de email a partir da URL que foi enviada para o email do utilizador.

void forgot\_password(); // Invoca a view ‘forgot-password’, onde se insere o email para recuperação, e os métodos responsáveis por enviar um email de recuperação, após verificar o email inserido.

void recovery(); // Invoca a view ‘recovery’, onde se insere a nova senha, e os métodos responsáveis por fazer a troca da senha, após realizar as verificações dos campos.

**ProductController:** Gerencia as operações relacionadas aos produtos da loja, como exibição, adição, edição e remoção de produtos.

**CartController:** Lida com a adição, remoção e atualização de itens no carrinho de compras, além de calcular totais e processar pedidos.

**OrderController:** Responsável por gerenciar o processo de checkout, incluindo a criação de pedidos e envio de confirmações de pedidos por e-mail.

**UserController**: Além do controle de autenticação, é necessário um controlador separado para gerenciar informações do utilizador, como perfil, preferências e histórico de pedidos.

**PaymentController**: Caso o cliente deseje que sua webstore aceite pagamentos online, será necessário um controlador para integrar com gateways de pagamento e processar transações de forma segura.

**ReviewController**: Permite os utilizadores deixar comentários ou avaliações nos produtos, este controlador gerencia essas interações.

**AdminController**: Um controlador dedicado para operações administrativas, como gerenciamento de utilizadores, produtos, pedidos e outras configurações do sistema.

**Views:**

**Main**:

index

**Authentication:**

login

account\_created

recovery

register

confirm\_email

forgot\_password

**Product:**

**Cart:**

cart,

**Order:**

**User:**

**Payment:**

**Review:**

**Admin:**

**Models:**

**Customer**:

bool is\_email\_in\_use(); // Verifica se o email está em uso.

bool is\_active(); // Verifica se o utilizador está ativo.

bool validate\_email(); // Cuida do processo de verificação do email.

bool create\_user(); // Insere um novo utilizador na base de dados.

bool associate\_purl(); // Associa um novo personal\_url ao cadastro do cliente, para verificações de email.

bool change\_password(); // Realiza o processo de mudança de senha do utilizador.

## Outras informações