**Pasta Model**

**Arquivo 1: Database.php**

**Descrição:**  
Este arquivo define a classe Database, responsável por estabelecer a conexão com o banco de dados e-commerce.

**Classe:**

* Database

**Atributos privados:**

* $database (string): Nome do banco de dados.
* $host (string): Endereço do host onde o banco de dados está hospedado.
* $user (string): Nome de usuário do banco de dados.
* $password (string): Senha para o acesso ao banco de dados.

**Métodos Públicos:**

* getConnection():
  + **Descrição:** Estabelece e retorna a conexão com o banco de dados utilizando mysqli\_connect.
  + **Retorno:** Uma instância de conexão mysqli.

**Arquivo 2: ProductModel.php**

**Descrição:**  
Este arquivo define a classe ProductModel, que lida com operações relacionadas aos produtos, conectando-se à tabela produtos no banco de dados.

**Classe:**

* ProductModel (extends Database)

**Atributos Privados:**

* $mysqli: Instância da conexão com o banco de dados, obtida a partir do método getConnection().

**Métodos Públicos:**

* \_\_construct():
  + **Descrição:** Inicializa uma nova instância de ProductModel e estabelece uma conexão com o banco de dados.
* fetch():
  + **Descrição:** Realiza uma consulta que busca todos os registros da tabela produtos.
  + **Retorno:** Array de todos os registros.
* fetchById($id):
  + **Descrição:** Realiza uma consulta que busca um registro da tabela produtos com base em um id específico.
  + **Parâmetro:** $id (int) — ID do produto a ser buscado.
  + **Retorno:** Array com o registro correspondente ao ID.
* query($sql):
  + **Descrição:** Executa uma consulta SQL diretamente no banco de dados.
  + **Parâmetro:** $sql (string) — Comando SQL a ser executado.
  + **Retorno:** true se a execução foi bem-sucedida.

**Arquivo 3: UserModel.php**

**Descrição:**  
Este arquivo define a classe UserModel, que gerencia as operações relacionadas aos usuários, acessando a tabela usuarios no banco de dados.

**Classe:**

* UserModel (extends Database)

**Atributos Privados:**

* $mysqli: Instância da conexão com o banco de dados, obtida a partir do método getConnection().

**Métodos Públicos:**

* \_\_construct():
  + **Descrição:** Inicializa uma nova instância de UserModel e estabelece uma conexão com o banco de dados.
* fetch():
  + **Descrição:** Realiza uma consulta que busca todos os registros da tabela usuarios.
  + **Retorno:** Array com todos os registros.
* fetchById($id):
  + **Descrição:** Realiza uma consulta que busca um registro da tabela usuarios com base em um id específico.
  + **Parâmetro:** $id (int) — ID do usuário a ser buscado.
  + **Retorno:** Array com o registro correspondente ao ID.
* query($sql):
  + **Descrição:** Executa uma consulta SQL diretamente no banco de dados.
  + **Parâmetro:** $sql (string) — Comando SQL a ser executado.
  + **Retorno:** true se a execução foi bem-sucedida.

**Pasta Controler**

Estes controladores são responsáveis por lidar com diferentes páginas de cadastro no site, implementando a lógica para carregar a view correta para cada tipo de cadastro (usuário, pagamento, produto). Abaixo, cada controlador é descrito detalhadamente.

#### **Arquivo 1**: CadastroController.php

**Descrição**: O CadastroController é responsável por carregar a view de cadastro de usuário. Ele herda de RenderView, uma classe que provê o método loadView para renderização de páginas.

**Método**:

* index(): Esse método é executado quando a rota de cadastro de usuário é chamada. Ele utiliza o método loadView para carregar a view cadastro.

php

Copiar código

<?php

class CadastroController extends RenderView {

public function index() {

$this->loadView('cadastro', []);

}

}

#### **Arquivo 2: CadastroPagController.php**

**Descrição**: O CadastroPagController é responsável por carregar a view de cadastro de pagamentos. Também herda de RenderView e utiliza o método loadView.

**Método**:

* index(): Carrega a view cadastroPag quando a rota de cadastro de pagamento é acessada.

php

Copiar código

<?php

class CadastroPagController extends RenderView {

public function index() {

$this->loadView('cadastroPag', []);

}

}

#### **Arquivo 3: CadastroProdutoController.php**

**Descrição**: O CadastroProdutoController controla o carregamento da view de cadastro de produtos. Assim como os demais, herda de RenderView.

**Método**:

* index(): Responsável por carregar a view cadastroProduto, que exibe o formulário de cadastro de produtos.

php

Copiar código

<?php

class CadastroProdutoController extends RenderView {

public function index() {

$this->loadView('cadastroProduto', []);

}

}

Esses controladores lidam com operações relacionadas ao carrinho, estoque, login, loja, e emissão de recibos, além de gerenciar rotas não encontradas.

#### **Arquivo 4: CarrinhoController.php**

**Descrição**: O CarrinhoController é responsável pelo carregamento da view de carrinho de compras.

**Método**:

* index(): Carrega a view carrinho ao acessar a rota de carrinho.

php

Copiar código

<?php

class CarrinhoController extends RenderView {

public function index() {

$this->loadView('carrinho', []);

}

}

#### **Arquivo 5: EstoqueController.php**

**Descrição**: O EstoqueController gerencia a visualização, criação, atualização e remoção de produtos no estoque.

**Métodos**:

* index(): Carrega a view estoque com dados de produtos.
* create(): Carrega a view cadastroProduto para adicionar novos produtos.
* store(): Recebe dados do formulário e armazena o produto no banco de dados.
* edit($id): Carrega a view atualizarProduto com os dados do produto específico para edição.
* update($id): Atualiza os dados do produto no banco de dados.
* delete($id): Remove o produto do banco de dados.

php

Copiar código

<?php

class EstoqueController extends RenderView {

public function index() {

$bd = new ProductModel();

$this->loadView('estoque', ['produtos' => $bd->fetch()]);

}

public function create() {

$this->loadView('cadastroProduto');

}

public function store() {

$bd = new ProductModel();

$name = $\_POST['name'];

$estoque = $\_POST['estoque'];

$preco = $\_POST['preco'];

$description = $\_POST['description'];

$sql = "INSERT INTO produtos (nome, descricao, preco, quantidade) VALUES ('$name', '$description', '$preco', '$estoque');";

$bd->query($sql);

header('Location: estoque');

}

public function edit($id) {

$bd = new ProductModel();

$this->loadView('atualizarProduto', ['id' => $id[0], 'dados' => $bd->fetchById($id[0])]);

}

public function update($id) {

$bd = new ProductModel();

$name = $\_POST['name'];

$estoque = $\_POST['estoque'];

$preco = $\_POST['preco'];

$description = $\_POST['description'];

$sql = "UPDATE produtos SET nome='$name', descricao='$description', preco='$preco', quantidade='$estoque' WHERE id='$id[0]'";

$bd->query($sql);

header('Location: ../estoque');

}

public function delete($id) {

$bd = new ProductModel();

$sql = "DELETE FROM produtos WHERE id='$id[0]'";

$bd->query($sql);

header('Location: ../estoque');

}

}

#### **Arquivo 6: LoginController.php**

**Descrição**: O LoginController controla o acesso à view de login, gerenciando a autenticação de usuários.

**Método**:

* index(): Carrega a view login quando a rota de login é acessada.

php

Copiar código

<?php

class LoginController extends RenderView {

public function index() {

$this->loadView('login', []);

}

}

#### **Arquivo 7: LojaController.php**

**Descrição**: O LojaController carrega a view da loja virtual onde os produtos estão listados.

**Método**:

* index(): Carrega a view loja.

php

Copiar código

<?php

class LojaController extends RenderView {

public function index() {

$this->loadView('loja', []);

}

}

#### **Arquivo 8: NotFoundController.php**

**Descrição**: O NotFoundController lida com rotas não encontradas, fornecendo uma mensagem de erro.

**Método**:

* index(): Exibe uma mensagem de "Route Not Found".

php

Copiar código

<?php

class NotFoundController {

public function index() {

echo "Route Not Found";

}

}

#### **Arquivo 9: ReciboController.php**

**Descrição**: O ReciboController carrega a view para emissão de recibos.

**Método**:

* index(): Carrega a view recibo quando a rota de recibo é acessada.

php

Copiar código

<?php

class ReciboController extends RenderView {

public function index() {

$this->loadView('recibo', []);

}

}

Esses detalhes fornecem uma visão geral formal dos controladores, explicando a função de cada arquivo e os métodos principais. Se precisar de mais informações ou explicações adicionais, posso ajudar a expandir.

**Pasta View**

#### **Arquivo 1: cadastroPag.php**

**Descrição**: Essa view é utilizada para o cadastro de informações de pagamento do usuário. O formulário coleta dados essenciais e permite ao usuário escolher entre duas formas de pagamento: boleto e Pix.

**Campos do Formulário**:

* **Nome Completo** (nome): Campo de texto para o nome completo do usuário.
* **CPF** (cpf): Campo de texto para o CPF do usuário.
* **Endereço** (endereco): Campo de texto para o endereço do usuário.
* **CEP** (cep): Campo de texto para o CEP do usuário.
* **Forma de Pagamento** (forma\_pagamento): Opções de pagamento, onde o usuário escolhe entre **Boleto** ou **Pix**.

**Estilos**:

* A view utiliza um estilo leve com uma paleta neutra, trazendo um layout centralizado para melhor usabilidade e legibilidade.

**Ação do Formulário**:

* Os dados são enviados para a URL /processar-pagamento via método POST, para o processamento do pagamento.

#### **Arquivo 2: cadastroPagamento.php**

**Descrição**: Este arquivo contém a estrutura de um formulário de cadastro para informações de pagamento do usuário. Ele permite que o usuário insira seus dados e escolha entre as opções de pagamento por boleto ou Pix. Este arquivo é muito semelhante ao **Arquivo 2**, mas possui um link adicional para a página de **Recibo** após a confirmação do pagamento.

**Campos do Formulário**:

* **Nome Completo** (nome): Campo de texto para inserir o nome completo do usuário.
* **CPF** (cpf): Campo de texto para inserir o CPF do usuário.
* **Endereço** (endereco): Campo de texto para o endereço do usuário.
* **CEP** (cep): Campo de texto para inserir o CEP do usuário.
* **Forma de Pagamento** (forma\_pagamento): Opções de pagamento disponíveis, com seleção entre **Boleto** e **Pix**.

**Botão de Enviar e Link para Recibo**:

* **Confirmar Pagamento**: Um botão que envia o formulário para /processar-pagamento via método POST, para processar o pagamento.
* **Link para Recibo**: Um link adicional que redireciona para a página recibo, onde o usuário pode visualizar e imprimir o recibo após o cadastro do pagamento.

**Estilos**:

* O formulário é centralizado na tela e utiliza um estilo limpo com um esquema de cores neutras, fornecendo uma aparência profissional e facilitando a usabilidade.

#### **Arquivo** **3: carrinho.php**

**Descrição**: Este arquivo exibe uma lista de produtos cadastrados em estoque e permite que o usuário visualize informações de cada produto, como nome, quantidade, preço e descrição. Também oferece opções para editar ou excluir produtos e um botão para adicionar novos itens ao estoque.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Inclui links para ícone (favicon) e bibliotecas de **Bootstrap** para estilização e componentes interativos.
   * Título da página configurado como **Carrinho de Compras | Home**.
2. **Barra de Navegação (<nav>)**:
   * Contém links de navegação para **Home** e **Loja**.
3. **Tabela de Produtos (<table>)**:
   * Uma tabela com colunas para exibir os seguintes detalhes de cada produto:
     + **Produto**: Nome do produto.
     + **Estoque**: Quantidade em estoque.
     + **Preço**: Preço do produto.
     + **Sobre**: Descrição do produto.
     + **Validade**: Atualmente preenchida com valor 'null' como marcador.
   * Para cada produto na tabela, são gerados botões para **Editar** e **Excluir**.
4. **Botão de Cadastro de Produto**:
   * Um botão abaixo da tabela permite adicionar novos produtos, redirecionando para a rota **estoque-create**.

#### **Arquivo 4: login.php**

**Descrição**: Este arquivo é uma página de login para criação de uma conta de usuário. Ele exibe um formulário para o usuário inserir seu e-mail e senha, além de opções para "Salvar" as credenciais, recuperar senha e um link para registrar uma nova senha.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Inclui um componente {Head title={title} path={path}} que parece ser parte de uma estrutura de template, com variáveis que seriam passadas dinamicamente.
   * Configura o título e outros metadados para a página.
2. **Corpo Principal (<body>)**:
   * **Título da Página**: A palavra "Login" centralizada na parte superior.
   * **Cartão de Login (<div class="card">)**:
     + Dentro do cartão, exibe a mensagem **"Criar Sua Conta"**.
     + **Formulário de Login**:
       - Campo para **e-mail** e **senha** com ícones associados (envelope para e-mail e cadeado para senha).
       - **Checkbox "Salvar"**: Opção para o usuário marcar e salvar suas credenciais.
       - **Botão "Criar"**: Envia o formulário para criação da conta.
       - Links adicionais para:
         * **Esqueci Minha Senha**: Redireciona para uma página de recuperação de senha.
         * **Registre Nova Senha**: Link para página de registro de nova senha.
   * **Link para Loja**: Ao final, há um link para redirecionar o usuário para a página da loja.
3. **Rodapé de Scripts (<Scripts path={path} />)**:
   * Inclui scripts adicionais da página, referenciados por {path}, para carregar funcionalidades adicionais.

#### **Arquivo 5: recibo.php**

**Descrição**: Este arquivo exibe uma **tela simples de recibo de pagamento**. Ele informa ao usuário que o pagamento foi realizado e fornece um link para retornar à página inicial da loja.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Define a codificação de caracteres como **UTF-8** e configura a viewport para um layout responsivo.
   * O título da página é definido como **"Recibo"**.
2. **Corpo (<body>)**:
   * Um título principal (<h1>) que indica **"Tela de Recibo de Pagamento"**.
   * Um link para retornar à página inicial da loja, com o texto **"Voltar ao início"**.

#### **Arquivo 6: carrinho.php**

**Descrição**: Este arquivo exibe uma lista de produtos cadastrados em estoque e permite que o usuário visualize informações de cada produto, como nome, quantidade, preço e descrição. Também oferece opções para editar ou excluir produtos e um botão para adicionar novos itens ao estoque.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Inclui links para ícone (favicon) e bibliotecas de **Bootstrap** para estilização e componentes interativos.
   * Título da página configurado como **Carrinho de Compras | Home**.
2. **Barra de Navegação (<nav>)**:
   * Contém links de navegação para **Home** e **Loja**.
3. **Tabela de Produtos (<table>)**:
   * Uma tabela com colunas para exibir os seguintes detalhes de cada produto:
     + **Produto**: Nome do produto.
     + **Estoque**: Quantidade em estoque.
     + **Preço**: Preço do produto.
     + **Sobre**: Descrição do produto.
     + **Validade**: Atualmente preenchida com valor 'null' como marcador.
   * Para cada produto na tabela, são gerados botões para **Editar** e **Excluir**.
4. **Botão de Cadastro de Produto**:
   * Um botão abaixo da tabela permite adicionar novos produtos, redirecionando para a rota **estoque-create**.

#### Arquivo 7: login.php

**Descrição**: Este arquivo é uma página de login para criação de uma conta de usuário. Ele exibe um formulário para o usuário inserir seu e-mail e senha, além de opções para "Salvar" as credenciais, recuperar senha e um link para registrar uma nova senha.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Inclui um componente {Head title={title} path={path}} que parece ser parte de uma estrutura de template, com variáveis que seriam passadas dinamicamente.
   * Configura o título e outros metadados para a página.
2. **Corpo Principal (<body>)**:
   * **Título da Página**: A palavra "Login" centralizada na parte superior.
   * **Cartão de Login (<div class="card">)**:
     + Dentro do cartão, exibe a mensagem **"Criar Sua Conta"**.
     + **Formulário de Login**:
       - Campo para **e-mail** e **senha** com ícones associados (envelope para e-mail e cadeado para senha).
       - **Checkbox "Salvar"**: Opção para o usuário marcar e salvar suas credenciais.
       - **Botão "Criar"**: Envia o formulário para criação da conta.
       - Links adicionais para:
         * **Esqueci Minha Senha**: Redireciona para uma página de recuperação de senha.
         * **Registre Nova Senha**: Link para página de registro de nova senha.
   * **Link para Loja**: Ao final, há um link para redirecionar o usuário para a página da loja.
3. **Rodapé de Scripts (<Scripts path={path} />)**:
   * Inclui scripts adicionais da página, referenciados por {path}, para carregar funcionalidades adicionais.

#### **Arquivo 8: home.php ou index.php**

**Descrição**: Este arquivo define a **página inicial** da loja online, onde os produtos disponíveis são exibidos em formato de **cards** com imagem, nome, descrição, preço e um botão para adicionar ao carrinho. Ele contém um layout responsivo, com navegação para outras áreas, incluindo **login**, **estoque** (para administradores) e o **carrinho de compras**.

**Estrutura e Componentes**:

1. **Cabeçalho (<header>)**:
   * Inclui o título principal da página, **"Loja Online"**.
   * Links de navegação:
     + **Sign in**: Para a página de login.
     + **Admin**: Para o estoque, acessível para o administrador.
     + **Carrinho de Compras**: Exibido como um botão destacado, no canto superior direito, para facilitar o acesso ao carrinho.
2. **Seção Principal (<div class="container">)**:
   * **Título de Seção**: "Produtos Disponíveis", informando ao usuário que os produtos podem ser visualizados e adicionados ao carrinho.
   * **Grade de Produtos (<div class="product-grid">)**:
     + Cada produto é exibido em um **card** (<div class="product-card">), que inclui:
       - **Imagem**: Exibida com uma URL de espaço reservado.
       - **Nome do Produto**: Nome fictício, por exemplo, **"Produto 1"**.
       - **Descrição**: Breve descrição.
       - **Preço**: Exibido em negrito.
       - **Botão "Adicionar ao Carrinho"**: Para permitir que o usuário adicione o produto ao carrinho.
3. **Estilos (<style>)**:
   * Configurações gerais de estilo, como fonte, margens e cores, criando um layout simples e direto.
   * **Header**: Em fundo escuro com texto claro, onde links de navegação se destacam.
   * **Cards de Produtos**: Cada card possui borda, padding e cores de fundo para o botão, que muda de cor quando o mouse passa sobre ele.
   * **Botão de Carrinho**: Em azul e posicionado no canto superior direito, destacando-se no layout.

#### **Arquivo 9: recibo.php**

**Descrição**: Este arquivo exibe uma **tela simples de recibo de pagamento**. Ele informa ao usuário que o pagamento foi realizado e fornece um link para retornar à página inicial da loja.

**Estrutura e Componentes**:

1. **Cabeçalho (<head>)**:
   * Define a codificação de caracteres como **UTF-8** e configura a viewport para um layout responsivo.
   * O título da página é definido como **"Recibo"**.
2. **Corpo (<body>)**:
   * Um título principal (<h1>) que indica **"Tela de Recibo de Pagamento"**.
   * Um link para retornar à página inicial da loja, com o texto **"Voltar ao início"**.

.