1. Estrutura Geral do Projeto

O projeto MortyStock está organizado de maneira que reflete uma aplicação web típica baseada em PHP. A estrutura básica do projeto inclui diretórios para arquivos PHP, JavaScript, CSS, e bibliotecas externas, além de um banco de dados para armazenar informações. Aqui está uma visão geral:

- **controllers/**: Este diretório contém arquivos PHP que lidam com a lógica de negócios e interações com o banco de dados. Exemplos incluem banco.php para conexões com o banco e scripts como cadastrar_produto.php, editar_produto.php, e excluir_produto.php que lidam com operações CRUD.
- **assets/**: Inclui recursos estáticos como arquivos CSS, JavaScript personalizados, imagens, etc. O estilo visual da aplicação é definido aqui.
- **vendor/**: Contém dependências externas gerenciadas por um gerenciador de pacotes como Composer. Bibliotecas de terceiros, como phpdotenv para gerenciamento de variáveis de ambiente, são encontradas aqui.
- **index.php**: O ponto de entrada principal da aplicação, que provavelmente define a estrutura básica da página e inclui outros componentes conforme necessário.
- **pages/**: Este diretório provavelmente contém páginas específicas do aplicativo, como home.php, que lida com a interface do usuário para visualização e manipulação de produtos.

2. Configuração Inicial

O projeto parece usar o Composer para gerenciar dependências. Aqui estão alguns aspectos de configuração importantes:

- **Banco de Dados**: A configuração do banco de dados é gerida pelo arquivo banco.php, que provavelmente contém a lógica para conectar-se ao banco de dados usando PDO (PHP Data Objects). Este arquivo é incluído em outros scripts para acessar o banco de dados.
- **Variáveis de Ambiente**: O uso de phpdotenv sugere que configurações sensíveis, como credenciais de banco de dados, estão sendo gerenciadas em um arquivo .env. Isso é uma boa prática para manter as credenciais seguras e fora do código-fonte.

3. Funcionalidades Específicas

O projeto MortyStock possui funcionalidades-chave de CRUD (Create, Read, Update, Delete) para gerenciar produtos. Aqui está uma descrição de cada uma:

- **Cadastro de Produto** (cadastrar_produto.php): Este arquivo recebe dados do formulário de cadastro, valida os campos e insere um novo registro no banco de dados. Ele retorna uma resposta em JSON indicando sucesso ou falha.
- **Edição de Produto** (editar_produto.php e atualizar_produto.php): O arquivo editar_produto.php busca informações de um produto específico para preenchê-las no modal de edição. atualizar_produto.php lida com a atualização dos dados do produto no banco de dados após as edições.
- **Exclusão de Produto** (excluir_produto.php): Este script remove um produto específico do banco de dados com base nos IDs fornecidos.
- **Listagem de Produtos** (listar_produtos.php): Gera a tabela de produtos para exibição no front-end, preenchendo cada linha com os dados dos produtos e adicionando botões para editar e excluir.

4. Integração com o Banco de Dados

Os arquivos PHP usam PDO para interagir com o banco de dados. Isso oferece uma interface segura e flexível para executar operações SQL. Exemplos incluem o uso de prepare() e execute() para evitar SQL injection.

5. Uso de JavaScript e AJAX

O projeto faz uso extensivo de JavaScript para criar uma experiência de usuário interativa:

- **AJAX**: Chamadas AJAX são usadas para enviar e receber dados dos scripts PHP sem recarregar a página. Por exemplo, ao editar ou excluir um produto, a página não precisa ser recarregada para refletir essas mudanças.
- **Manipulação do DOM**: JavaScript, juntamente com jQuery, é usado para manipular elementos do DOM, como modais, e preencher ou limpar campos de formulário.
- **DataTables**: A biblioteca DataTables é usada para exibir produtos em uma tabela dinâmica, com recursos como paginação, ordenação e pesquisa.

6. Estilo e Layout

- **Bootstrap**: O framework Bootstrap é usado para fornecer uma base responsiva e estilizar componentes como botões, tabelas e modais.

- **CSS Personalizado**: Arquivos CSS em assets/css/ provavelmente contêm estilos personalizados que se sobrepõem ao Bootstrap ou fornecem estilos específicos não incluídos por padrão.

7. Outras Considerações

- **Segurança**: A sanitização e validação de entrada são implementadas para prevenir ataques de injeção de SQL e outros tipos de ataques baseados em entrada. Isso é visto no uso de funções como filter_var() e declarações preparadas.
- **Melhorias Futuras**: Implementações adicionais de segurança, como CSRF (Cross-Site Request Forgery) tokens, poderiam ser consideradas. Além disso, a otimização de consultas e uso de indexação no banco de dados pode melhorar a eficiência.

8. Ferramentas Utilizadas

Linguagens e Tecnologias:

- **PHP**: Linguagem de script do lado do servidor usada para desenvolvimento web. É utilizada para gerenciar a lógica do backend, comunicação com o banco de dados, e operações de CRUD (Create, Read, Update, Delete).
- **JavaScript (com jQuery)**: Linguagem de script do lado do cliente usada para manipulação do DOM, interatividade, e requisições AJAX. jQuery é uma biblioteca JavaScript que simplifica a manipulação de elementos HTML e facilita as requisições AJAX.
- **HTML/CSS**: Linguagens de marcação e estilo usadas para estruturar e estilizar as páginas da web. HTML define a estrutura dos elementos da página, enquanto CSS cuida do estilo visual.
- **SQL**: Linguagem de consulta estruturada usada para interagir com o banco de dados. Usada em consultas para buscar, inserir, atualizar e excluir dados.

Frameworks e Bibliotecas:

- **Bootstrap**: Framework CSS para desenvolvimento de interfaces responsivas e móveis. Facilita a criação de layouts consistentes e componentes estilizados como botões, tabelas e modais.
- **DataTables**: Plugin jQuery que facilita a criação de tabelas dinâmicas com funcionalidades como paginação, busca e ordenação.

- **Font Awesome**: Biblioteca de ícones usada para adicionar ícones gráficos de forma fácil e consistente nos botões e outros elementos da interface.
- **phpdotenv**: Biblioteca PHP usada para gerenciar variáveis de ambiente, permitindo armazenar informações sensíveis como credenciais de banco de dados fora do código-fonte.

Ferramentas de Desenvolvimento:

- **Composer**: Gerenciador de dependências para PHP. Utilizado para instalar e gerenciar bibliotecas PHP, como phpdotenv.
- **Git**: Sistema de controle de versão para rastrear alterações no código e colaborar com outros desenvolvedores.

2. Tipos de Programação Utilizados

- **Programação Procedural**: A maior parte do código PHP segue um estilo procedural, executando instruções de forma sequencial. Isso é típico em scripts PHP que realizam tarefas específicas, como conectar-se ao banco de dados ou processar um formulário.
- **Programação Orientada a Eventos**: O JavaScript no projeto reage a eventos, como cliques de botão, para executar funções específicas. Isso inclui abrir modais, enviar requisições AJAX, e atualizar elementos da página dinamicamente.
- **Programação Assíncrona**: Uso de AJAX para enviar e receber dados de forma assíncrona sem recarregar a página, melhorando a experiência do usuário.

3. Descrição de Arquivos

Arquivos de Backend (PHP)

- **banco.php**: Arquivo responsável pela conexão com o banco de dados usando PDO (PHP Data Objects). Este arquivo contém as credenciais de conexão e métodos para abrir e fechar a conexão.
- **cadastrar_produto.php**: Script que recebe dados do formulário de cadastro de produtos, valida os campos e insere um novo registro no banco de dados. Retorna uma resposta JSON indicando sucesso ou falha.
- **editar_produto.php**: Script que busca os detalhes de um produto específico a partir de seu ID para preenchê-los no formulário de edição. Retorna os dados do produto em formato JSON.

- **atualizar_produto.php**: Recebe os dados do produto editado, sanitiza e valida os campos, e atualiza o registro correspondente no banco de dados. Retorna uma resposta JSON para indicar o resultado da operação.
- **excluir_produto.php**: Script que recebe o ID de um produto e o remove do banco de dados. Retorna uma resposta JSON indicando se a exclusão foi bem-sucedida.
- **listar_produtos.php**: Gera uma lista de produtos para ser exibida em uma tabela no frontend. Inclui lógica para adicionar botões de edição e exclusão para cada produto.

Arquivos de Frontend (HTML, JavaScript, CSS)

- **home.php**: Página principal do sistema que exibe a lista de produtos em uma tabela. Inclui botões para cadastrar novos produtos e abre modais para editar ou excluir produtos existentes.
- **customs.js**: Script JavaScript que gerencia a lógica de frontend, incluindo eventos de clique para botões de editar e excluir, requisições AJAX para interagir com scripts PHP, e manipulação do DOM para atualizar o conteúdo da página.
- **app.css**: Arquivo CSS que contém estilos personalizados aplicados aos elementos HTML para ajustar a aparência além do que o Bootstrap oferece.
- **pagina.css**: Pode conter estilos adicionais ou específicos para determinadas páginas do projeto, ajudando a personalizar a interface do usuário.

Dependências e Configurações

- **composer.json**: Arquivo de configuração para o Composer que define as dependências do projeto. Inclui bibliotecas necessárias como phpdotenv.
- **.env**: Arquivo de configuração (não incluído por padrão) que armazena variáveis de ambiente, como credenciais de banco de dados, para manter as informações sensíveis seguras.