DOCUMENTAÇÃO – FINDSNEAKERS STORE

1. IDENTIFICAÇÃO

Projeto: FindSneakers Store

Universidade Estácio de Sá

Disciplina: Desenvolvimento Web em HTML5, CSS3, JavaScript e PHP

Professor: Lucas Gomes

Data: 24/11/2025

Autores: Daniel José Marques Carvalho, Guilherme David Sena, Moisés Ribeiro de Souza Araújo.

Link: https://github.com/moises1dev/FindSneakers

2. DESCRIÇÃO GERAL DO PROJETO

O projeto FindSneakers Store consiste no desenvolvimento de um sistema web completo de uma loja de tênis, com operações de CRUD para gerenciamento de produtos, autenticação de administradores, cadastro e login de clientes, interface responsiva e integração com banco de dados MySQL. O projeto utiliza Doctrine ORM para acesso a dados e integra o padrão MVC por meio do Zend Framework (Laminas MVC) dentro do mesmo diretório da aplicação.

3. EVOLUÇÃO DO PROJETO

Todas as etapas a seguir foram concluídas até 24 de novembro de 2025.

• Planejamento inicial: definição do tema, público-alvo e escopo.

• Estrutura prévia da documentação e criação do cronograma.

• Desenvolvimento do layout inicial e modelagem do banco de dados.

• Implementação completa do CRUD integrado ao Doctrine ORM.

• Criação e finalização do frontend responsivo (HTML, CSS e JavaScript).

• Implementação do sistema de login e controle de sessão.

• Criação da área administrativa com acesso restrito.

• Login e cadastro de clientes funcionando com persistência em banco.

• Integração do Zend Framework (Laminas MVC) ao projeto já existente.

• Testes finais e correção de erros.

• Revisão e atualização da documentação.

• Preparação da apresentação final e publicação no GitHub.

4. REQUISITOS FUNCIONAIS

RF01 – Autenticação segura de administradores.

RF02 – CRUD completo de produtos (criar, listar, editar e excluir).

RF03 – Login e cadastro de clientes com validação de dados.

RF04 – Controle de acesso à área administrativa.

RF05 – Exibição, busca e filtragem de produtos na interface principal.

5. REQUISITOS NÃO FUNCIONAIS

RNF01 – Interface intuitiva e de fácil uso.

RNF02 – Layout responsivo para diversos dispositivos.

RNF03 – Gerenciamento de sessão local e segurança básica (hash de senhas).

RNF04 – Padrão arquitetural MVC utilizando Zend Framework (Laminas).

RNF05 – Consultas otimizadas utilizando Doctrine ORM.

6. TECNOLOGIAS UTILIZADAS

• Backend: PHP 8.2, Zend Framework / Laminas MVC.

• ORM: Doctrine ORM mapeando a entidade Product.

• Frontend: HTML5, CSS3, JavaScript.

• Banco de Dados: MySQL (banco findsneakers).

• Servidor: Apache (XAMPP).

• Versionamento: Git e GitHub.

• Editor de desenvolvimento: Visual Studio Code.

7. SITUAÇÃO ATUAL DO PROJETO

O sistema encontra-se plenamente funcional, contendo:

• CRUD de produtos integrado ao Doctrine ORM.

• Área administrativa acessível apenas mediante login de administrador.

• Login e cadastro de clientes com validação de credenciais.

• Interface pública com filtros funcionais por marca, gênero e busca textual.

• Sessões separadas para clientes e administradores.

• Integração do Zend Framework (Laminas MVC).

8. ARQUITETURA GERAL DA APLICAÇÃO

A aplicação adota uma arquitetura em camadas, combinando uma interface web dinâmica com APIs em PHP e acesso estruturado ao banco de dados. De forma geral, o sistema é organizado nas seguintes camadas:

- Camada de Apresentação (Frontend):
  Responsável pela interface com o usuário. É composta principalmente pelos arquivos index.html, styles.css e app.js.
  A página principal é carregada uma única vez e o JavaScript atualiza o conteúdo de forma dinâmica (listagem de produtos, modais de login/cadastro, área administrativa), consumindo as APIs via fetch.
  Por esse motivo, a aplicação pode ser caracterizada como uma SPA simplificada (Single Page Application).

- Camada de Lógica de Negócio / API (Backend):
  Implementada em PHP por meio de arquivos como api_products.php, api_login_admin.php, api_login_customer.php e api_register_customer.php.
  Esses scripts recebem requisições HTTP, validam dados de entrada, aplicam regras de negócio (CRUD de produtos, autenticação de admins e clientes, verificação de credenciais) e retornam respostas em formato JSON.
  As APIs seguem um estilo REST/JSON, expondo operações de criação, leitura, atualização e exclusão de recursos de forma organizada.

- Camada de Dados (Persistência):
  Utiliza o banco de dados MySQL (banco findsneakers), com tabelas como products, admins e customers.
  A tabela products é mapeada para a entidade Product por meio do Doctrine ORM, que abstrai o SQL e permite trabalhar com objetos. As demais tabelas utilizam PDO com prepared statements, garantindo segurança básica e prevenção contra SQL Injection.

- Camada de Framework / MVC (Laminas):
  O projeto também integra o Zend Framework / Laminas MVC dentro do mesmo diretório da aplicação.
  Foi configurado um módulo Application com public/index.php atuando como front controller, application.config.php definindo módulos e IndexController + views demonstrando o uso de rotas, controllers e views no padrão MVC.

Essa camada serve como base estrutural para futuras páginas server-side (como relatórios ou painéis adicionais), coexistindo com a camada SPA existente.

Em conjunto, essas camadas permitem separar de forma clara a interface com o usuário, a lógica de negócio e o acesso a dados, facilitando a manutenção, a evolução do sistema e a justificativa técnica do uso de tecnologias como PHP, Doctrine ORM e Zend Framework (Laminas MVC).

9. INTEGRAÇÃO DO ZEND FRAMEWORK (LAMINAS MVC)

O projeto incorpora o Zend Framework / Laminas MVC de forma integrada ao mesmo diretório da aplicação web principal. O objetivo dessa integração é demonstrar o uso de um framework PHP moderno baseado no padrão Modelo–Visão–Controlador (MVC), convivendo com a camada de APIs REST já existente.

A estrutura adicionada ao projeto inclui:

- public/index.php — Front Controller do Laminas, responsável por inicializar o framework e direcionar requisições.
- config/application.config.php — Arquivo que registra módulos habilitados e configurações globais.
- module/Application/ — Módulo base contendo:
    - src/Controller/IndexController.php — Controller inicial.
    - view/application/index/index.phtml — View padrão.
    - config/module.config.php — Definição de rotas, controllers e templates.

Essa integração permite:

- expansão futura com páginas administrativas server-side,
- criação de rotas adicionais usando MVC tradicional,
- coexistência entre SPA + APIs REST e um módulo MVC estruturado,
- demonstração acadêmica do uso de um framework corporativo dentro da mesma solução.

10. DECISÕES TÉCNICAS

• Uso de PHP procedural simples nas APIs para didática.

• Doctrine ORM aplicado à entidade Product.

• Laminas MVC configurado no mesmo projeto, sem projeto separado.

• Organização simplificada para rodar no XAMPP.