DOCUMENTAÇÃO FINAL – FINDSNEAKERS STORE

1. IDENTIFICAÇÃO

Projeto: FindSneakers Store

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

Professor: Lucas Gomes

Instituição: Centro Universitário Estácio do Recife — Abdias de Carvalho

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

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

Data: 30/11/2025

2. DESCRIÇÃO GERAL DO PROJETO

O projeto FindSneakers Store consiste no desenvolvimento de um sistema web completo para venda de calçados esportivos (loja de tênis), oferecendo operações de CRUD para gerenciamento de produtos, autenticação de administradores, cadastro e login de clientes, interface responsiva e persistência de dados em MySQL. O sistema utiliza Doctrine ORM para mapeamento objeto-relacional e incorpora o padrão MVC por meio do Laminas (Zend Framework), totalmente integrado ao diretório raiz da aplicação.

3. EVOLUÇÃO DO PROJETO

Todas as etapas previstas no cronograma foram concluídas, incluindo planejamento, modelagem, desenvolvimento do frontend, desenvolvimento do backend, integração com Doctrine ORM, integração com Laminas MVC, testes finais e revisão da documentação.

O projeto encontra-se estável e totalmente funcional.

4. REQUISITOS FUNCIONAIS

RF01 – Autenticação de administradores.

RF02 – CRUD completo de produtos.

RF03 – Cadastro e login de clientes.

RF04 – Controle de sessão para clientes e administradores.

RF05 – Exibição, busca e filtros dinâmicos de produtos.

5. REQUISITOS NÃO FUNCIONAIS

RNF01 – Interface intuitiva e amigável.

RNF02 – Layout responsivo.

RNF03 – Gerenciamento de sessão local e segurança básica.

RNF04 – Arquitetura MVC com Laminas.

RNF05 – Consultas otimizadas com Doctrine ORM.

6. TECNOLOGIAS UTILIZADAS

Backend: PHP 8.2, Laminas MVC, APIs em PHP.

ORM: Doctrine ORM.

Frontend: HTML5, CSS3, JavaScript.

Banco de Dados: MySQL (findsneakers).

Servidor: Apache (XAMPP).

Ambiente de desenvolvimento integrado: Visual Studio Code.

Versionamento: GitHub.

7. SITUAÇÃO ATUAL DO PROJETO

O sistema está totalmente funcional, incluindo:

• CRUD completo de produtos com criação, visualização, edição e exclusão.

• Área administrativa restrita com autenticação.

• Login e cadastro de clientes.

• Filtros por marca, gênero e busca textual.

• Integração total com Doctrine ORM e Laminas MVC.

8. ARQUITETURA GERAL DA APLICAÇÃO

A aplicação adota uma arquitetura em camadas, separando interface, lógica de negócio e persistência de dados.

• Camada de Apresentação (Frontend):

Constituída por index.html, styles.css e app.js. A página principal é carregada uma única vez e atualizações de conteúdo são feitas dinamicamente via JavaScript, caracterizando a aplicação como uma SPA simplificada (Single Page Application).

• Camada de Lógica / API (Backend):

Implementada através de APIs REST/JSON em PHP, responsáveis por CRUD, validações e autenticação. Os endpoints retornam respostas em JSON consumidas pelo frontend.

• Camada de Dados (Persistência):

Utiliza MySQL, com a tabela products mapeada pelo Doctrine ORM. As tabelas admins e customers utilizam PDO com prepared statements.

• Camada de Framework / MVC (Laminas):

Integração com Laminas MVC adiciona estrutura modular, rotas, controllers e views server-side, convivendo com o modelo SPA + REST.

9. INTEGRAÇÃO COM LAMINAS MVC

O Laminas MVC foi integrado ao projeto dentro da pasta principal, permitindo a utilização do padrão Modelo–Visão–Controlador.

Estrutura integrada:

• public/index.php – Front controller do Laminas.

• config/application.config.php – Registro de módulos e configurações.

• module/Application/src/Controller/IndexController.php – Controller inicial.

• module/Application/view/application/index/index.phtml – View padrão.

• module/Application/config/module.config.php – Rotas e mapeamentos.

Essa integração permite expansão modular e futuras páginas server-side, coexistindo com a camada SPA existente.

10. DECISÕES TÉCNICAS

• Uso de APIs em PHP procedural para facilitar integrações com o frontend via fetch().

• Doctrine ORM adotado para abstração de SQL e melhor manutenção da camada de dados.

• Laminas MVC integrado para demonstrar estrutura corporativa de MVC.

• Projeto compatível com PHP 8.2 e configurado via Composer.