Análise do sistema: Loja LTDA

Uma Visão Geral da Arquitetura, Escalabilidade e Funcionalidades de um E-commerce Híbrido (PHP + JS).

Principais Funcionalidades

Experiência do Cliente

Carrinho Híbrido

Gerenciamento de carrinho via `localStorage` (visitante) com sincronização e merge automático com o BD após o login do cliente.

Checkout Completo

Fluxo de usuário desde o carrinho, seleção de endereço (com ViaCEP), pagamento (simulado) e geração de nota fiscal (via jsPDF).

Gestão de Conta

Clientes podem gerenciar múltiplos endereços, ver histórico de pedidos e postar/excluir avaliações (notas + comentários) de produtos.

Painel do Fornecedor

Gerenciamento de Produtos

Fornecedores podem criar e enviar produtos para a loja, controlando estoque, preço e descontos.

Rascunhos Locais

Um sistema de rascunhos baseado em `localStorage` permite que fornecedores salvem o trabalho de cadastro antes de enviar ao BD.

Catálogo Detalhado

O cadastro de produtos inclui múltiplas imagens, categorias (do BD), características dinâmicas e descrições formatadas.

Arquitetura do Sistema

Uma análise da arquitetura híbrida (PHP-SSR e Client-Side JS).

Arquitetura Híbrida

Backend (PHP + MySQL)

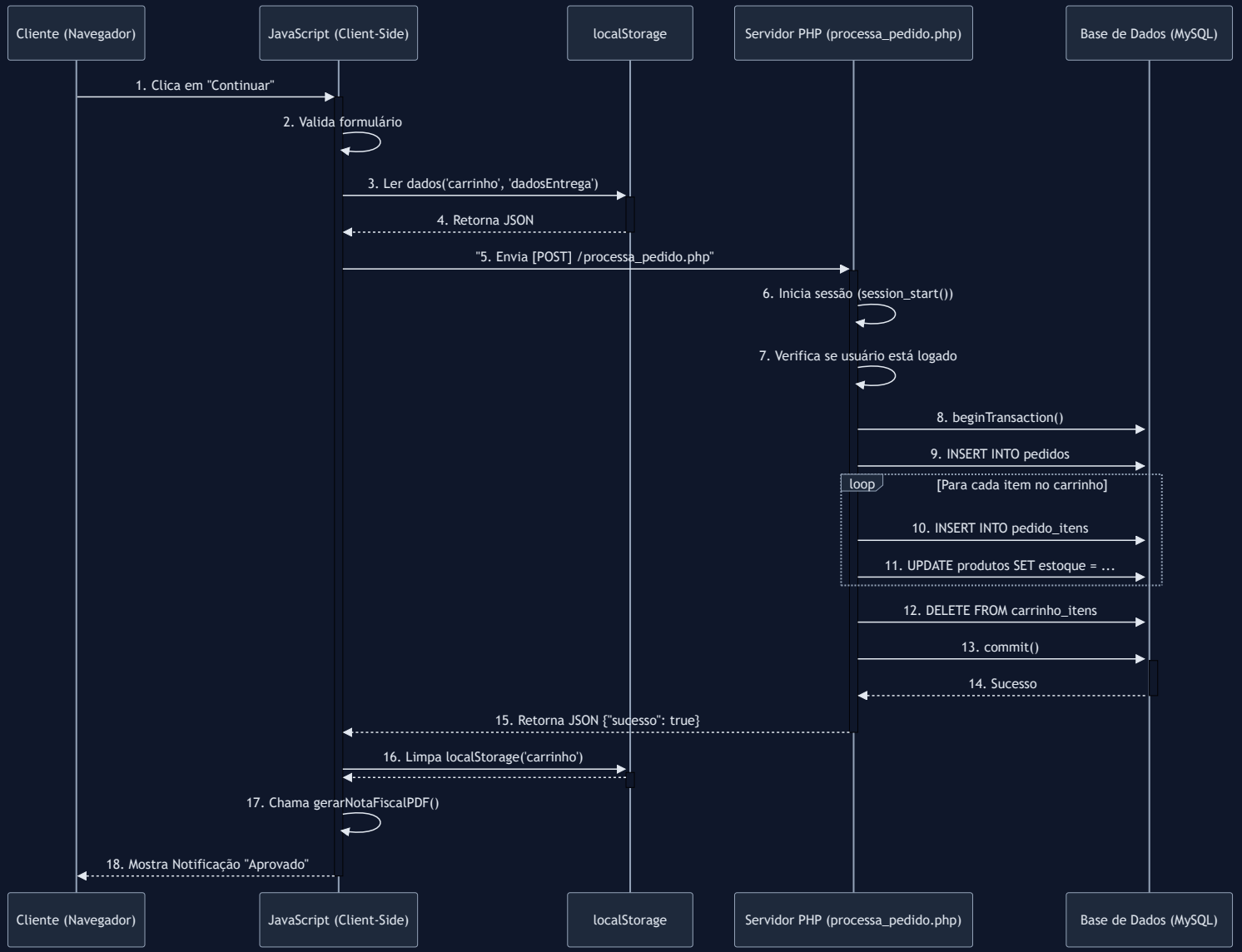
O PHP atua em duas frentes:

1. **SSR (Server-Side Rendering):** Monta e envia páginas HTML prontas (Ex: `index.php`, `tela_produto.php`).
2. **API "Headless":** Scripts em `Banco de dados/` (Ex: `processa_pedido.php`) recebem `fetch` (JSON) do cliente para executar lógica de negócios.

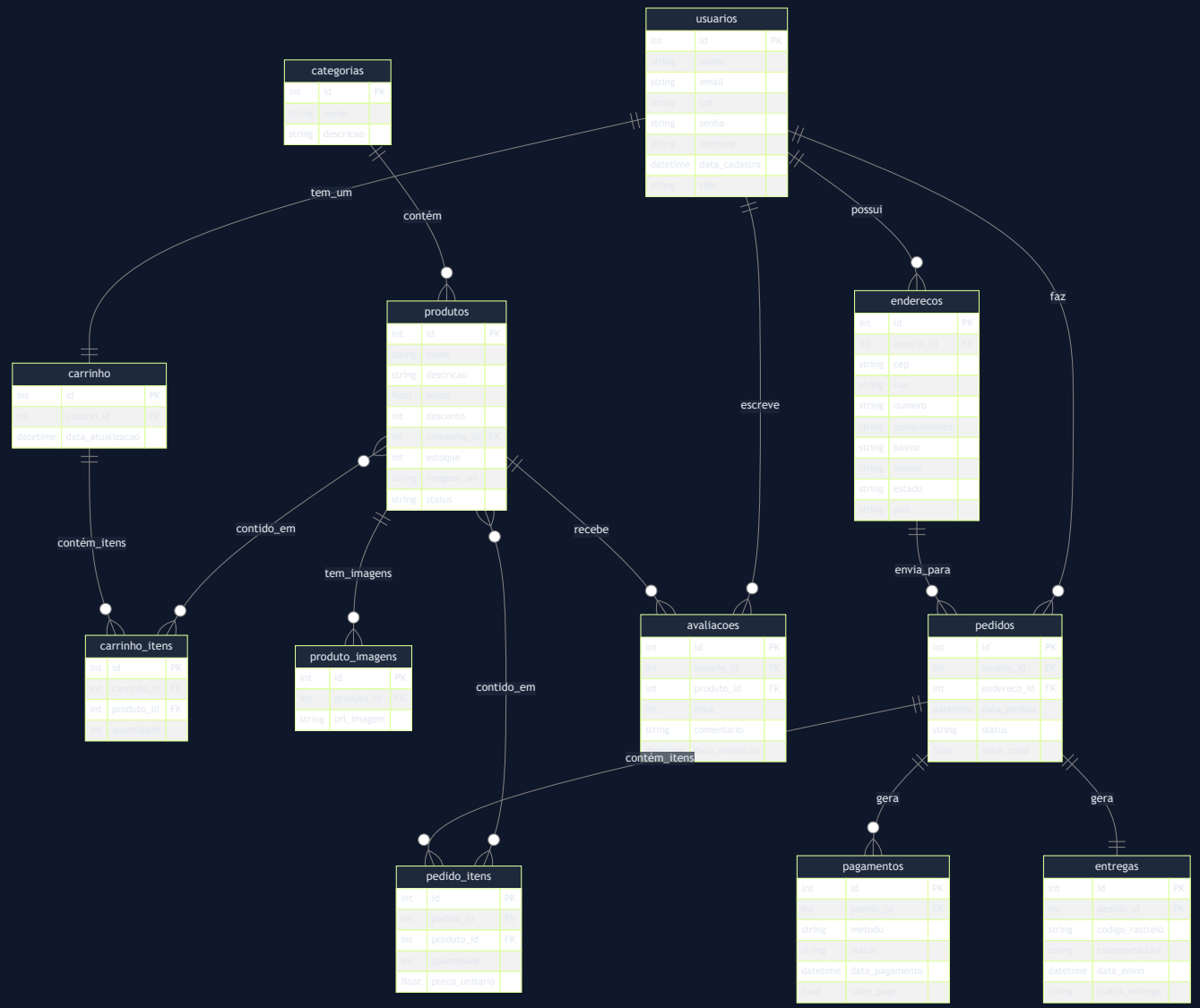
Frontend (JS + `localStorage`)

O JavaScript no cliente gerencia o estado (carrinho local, rascunhos), controla a UI (mapas, carrosséis) e consome as APIs do PHP e de terceiros (ViaCEP).

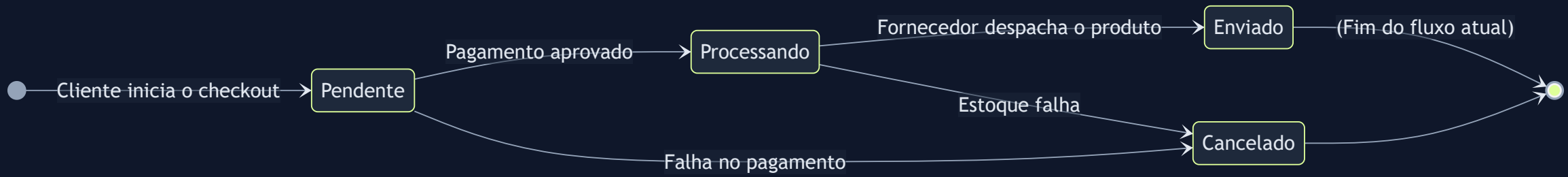
Arquitetura em Ação: Fluxo de Checkout



Arquitetura de Dados: Schema do BD



Comportamento: Ciclo de Vida do Pedido



Estratégia de Testes

Recomendações de Testes

- **Testes Unitários (PHP):** Usar PHPUnit para isolar e testar a lógica de negócios em scripts críticos (Ex: `processa_pedido.php`). Garantir que o cálculo de total e a atualização de estoque estejam corretos.
- **Testes Unitários (JS):** Usar Jest para testar a lógica do cliente (Ex: A função de "merge" do carrinho em `tela_carrinho.php`).
- **Testes de Integração:** Testar a comunicação entre o `fetch` do JS e a API do PHP, garantindo que os contratos de JSON (entrada e saída) estão corretos.
- **Testes E2E (End-to-End):** Usar ferramentas como Cypress para simular o fluxo completo do usuário: "Adicionar ao carrinho", "Fazer Login", "Finalizar Compra".

Análise de Escalabilidade

Avaliação da capacidade do sistema de crescer e lidar com mais tráfego.

Escalabilidade: Prós e Contras

Pontos Fortes

PHP "Shared-Nothing": O PHP é horizontalmente escalável. O uso de CDNs para bibliotecas (jsPDF, Leaflet) alivia a carga do servidor.

Schema do BD Otimizado: O uso correto de índices e chaves estrangeiras (`.sql`) garante consultas rápidas.

Gargalos

Gerenciamento de Sessão: `session_start()` em arquivos locais impede a escalabilidade horizontal. A solução é usar um DB de sessão (Ex: Redis).

Consultas Pesadas: `ORDER BY RAND()` e `AVG()` na home são gargalos que precisam de cache.

Principais Pontos de Melhora

- **Segurança (Crítico):** Corrigir a vulnerabilidade de "Adulteração de Preço". O back-end (`processa_pedido.php`) deve buscar os preços no BD, ignorando os preços enviados pelo cliente.
- **Validação de Uploads:** Implementar verificação de MIME type em `processa_novo_produto.php` para prevenir upload de arquivos maliciosos.
- **Manutenibilidade (DRY):** Substituir o HTML repetido do cabeçalho e rodapé por `require 'templates/header.php';` para centralizar a manutenção.
- **Centralização do Estado:** Migrar a lógica de "Rascunhos" do `localStorage` (volátil) para o BD (com status='rascunho'), garantindo que o usuário não perca seu trabalho.

Obrigado