**Relatório Técnico e Documentação do Projeto: API Central ERP**

**Versão:** 1.3 **Data:** 18 de Agosto de 2025

**1. Visão Geral e Arquitetura do Projeto**

O objetivo deste projeto é construir o "cérebro" de um sistema de gestão (ERP) híbrido e multi-nicho. A arquitetura foi desenhada para ser escalável, flexível e moderna.

**Tecnologias Base:**

* **Backend:** PHP 8.2+ com o framework Laravel 11.
* **Banco de Dados:** MySQL.

**Princípios da Arquitetura:**

* **API-First (API como Cérebro):** A API Central é o núcleo do sistema, contendo todas as regras de negócio e sendo a única a se comunicar com o banco de dados.
* **Sistema Híbrido:** O sistema é projetado para ter múltiplas interfaces (frontends) consumindo a mesma API: Retaguarda Web, PDV Desktop e App Mobile.
* **Solução Multi-Nicho com Relacionamentos Polimórficos:** A tabela produtos é "abstrata", contendo dados universais. Tabelas de "detalhes" (ex:

detalhes\_item\_mercado) contêm campos específicos de cada nicho, conectadas através de um **Relacionamento Polimórfico** do Laravel.

* **Fluxos de Trabalho Avançados:** Foram desenhados fluxos que espelham operações reais, como um ambiente de "Pré-Lançamento" para notas de compra e um sistema de "Vínculo Inteligente de Produtos" por fornecedor.

**2. A Fundação: Banco de Dados e Migrations**

A estrutura do banco de dados é robusta e utiliza as melhores práticas do MySQL (InnoDB, UTF8MB4). O controle de versão do esquema do banco é feito através do sistema de

**Migrations** do Laravel (em database/migrations/).

**Tabelas Principais Criadas:**

* empresas e usuarios para controle de acesso e multi-nicho.
* produtos, detalhes\_item\_mercado e dados\_fiscais\_produto para o catálogo polimórfico.
* fornecedores e clientes para os cadastros básicos.
* pedidos e itens\_pedido para as vendas.
* compras e itens\_compra para o pré-lançamento de notas.
* produto\_fornecedores para o vínculo automático.
* movimentacoes\_estoque para auditoria de inventário.
* historico\_pedidos para a timeline das vendas.

**3. A Ponte: Models do Eloquent**

Os Models (em

app/Models/) representam as tabelas e definem os relacionamentos entre elas (hasMany, belongsTo, morphTo, hasOne), permitindo uma interação limpa e orientada a objetos com os dados.

**4. A Lógica: Controllers, Requests e Rotas**

A lógica de negócio (em app/Http/Controllers/Api/) segue um padrão consistente e organizado:

* **Requests (Validação):** Classes dedicadas (ex: StoreProdutoMercadoRequest) isolam as regras de validação.
* **Controllers (Lógica):** Orquestram as ações, utilizam os Models e retornam respostas em JSON.

**Transações de Banco de Dados** (DB::transaction) são usadas em operações críticas.

* **Rotas (Pontos de Entrada):** O arquivo routes/api.php define as URLs da API e as conecta aos métodos dos controllers.

**5. Testes e Validação da Qualidade**

Adotamos uma estratégia de

**Test-Driven Development (TDD)** com o framework **Pest**. Para cada funcionalidade da API, uma suíte de testes automatizados foi criada (em

tests/Feature/Api/). Os testes cobrem todos os módulos de CRUD da API e as lógicas complexas de Pedidos e Compras, atuando como uma rede de segurança que garante a qualidade e a estabilidade do sistema.

**6. A Interface: Retaguarda Web (Painel de Gestão)**

Iniciamos a construção da primeira interface de usuário do sistema.

**6.1. Tecnologia:** A Retaguarda Web utiliza o stack padrão do Laravel, com o kit de início **Laravel Breeze**, que provê uma base sólida e segura. As tecnologias são:

* **Views:** Blade
* **Estilização:** Tailwind CSS
* **Interatividade:** Alpine.js

**6.2. Funcionalidades Implementadas:**

* **Autenticação:** Sistema completo e seguro de Login, Registro e Recuperação de Senha, instalado via Breeze.
* **Layout e Navegação:** O layout principal foi estabelecido e o menu de navegação (navigation.blade.php) foi customizado para incluir um **submenu "Cadastros"**, preparando o sistema para um crescimento organizado.
* **Módulo de Entrada de Notas (Início):** Foi criada a primeira página funcional da Retaguarda Web:
  + **Rota:** GET /compras/create registrada em routes/web.php.
  + **Controller:** CompraWebController com o método create que busca os fornecedores no banco.
  + **View:** O formulário inicial em resources/views/compras/create.blade.php foi construído, permitindo ao usuário iniciar o lançamento de uma nova nota.

**7. Próximos Passos e Recomendações**

**Status Atual:** A fundação completa da API está concluída e testada. A base da Retaguarda Web (autenticação, layout e a primeira tela funcional) está implementada.

**Próximos Passos Imediatos (Retaguarda Web):**

1. **Implementar o store do CompraWebController:** Programar a lógica que pega os dados do formulário de "Lançar Nova Nota" e os envia para a nossa própria API (POST /api/compras) que já está pronta e testada.
2. **Desenvolver a Tela de Listagem de Compras (compras.index):** Criar a página que lista todas as pré-notas, mostrando seu status ('Digitação', 'Finalizada', etc.).
3. **Desenvolver a Tela de Conferência (compras.edit):** Esta será a tela mais complexa, a nossa "mesa de cirurgia", onde o usuário irá vincular produtos, cadastrar novos itens, definir o markup e, finalmente, clicar no botão para dar entrada no estoque.

**Passos Futuros:**

* Finalizar a Retaguarda Web com todos os CRUDs e relatórios.
* Desenvolver o PDV Desktop.
* Implementar a lógica para os demais nichos de negócio.