# Teste Técnico – PHP Nativo e Laravel (Foco em Migração)

## Contexto

## Você participará de um teste prático (2–3h) para avaliarmos sua senioridade em PHP nativo e Laravel moderno, com foco em migração de sistemas legados (PHP 7.4 procedural) para Laravel 12. O cenário simula uma parte de um sistema legado em PHP nativo que precisa ser migrado para Laravel, aplicando boas práticas e arquitetura limpa. Objetivos de Avaliação

• PHP nativo: leitura, entendimento e refatoração de código legado  
• Laravel moderno (10+): uso de Eloquent, Migrations, FormRequest, Controllers, Services, Resources e Testing  
• Boas práticas: segurança, PSR-12, validações, arquitetura em camadas, transações.  
• Raciocínio de migração: clareza no plano de evolução do legado.

## Entregáveis

1. Refatoração em Laravel → API funcional com endpoints equivalentes ao legado  
2. Plano de migração (máx. 1 página, prático e objetivo)  
3. Testes automatizados (mínimo: sucesso, falha de validação, busca com filtro)  
4. README com instruções de execução

## Forma de Entrega

Repositório Git ou ZIP contendo: migrations, models, controllers, requests, services, testes, README e plano de migração. Incluir também o código legado como referência.

## Observações

Tempo sugerido: até 3h. Não é necessário hospedar. Será valorizada clareza, organização e boas práticas.

## Cenário

Código legado (fornecedor\_legacy.php):

A screen shot of a computer code

AI-generated content may be incorrect.

Estrutura de tabela legado:

CREATE TABLE fornecedores (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 nome VARCHAR(255) NOT NULL,  
 cnpj VARCHAR(14) NOT NULL,  
 email VARCHAR(255) NULL,  
 criado\_em DATETIME NOT NULL  
);  
CREATE UNIQUE INDEX ux\_fornecedores\_cnpj ON fornecedores(cnpj);

## O que deve ser feito em Laravel

• Criar migration e model Fornecedor (usar created\_at/updated\_at; considerar softDeletes)  
• Implementar validações via FormRequest  
• Criar endpoints REST (POST/GET com filtro por nome)  
• Usar Service Layer para regra de negócio (sanitização CNPJ, transações)  
• Usar Resources para formatar resposta JSON  
• Implementar testes Feature (mínimo: sucesso, falha de validação, busca filtrada)  
• Plano de migração em 1 página (dados, campos, validações, estratégia incremental)

## Critérios de Correção

**A. Segurança & Dados**- Prevenção de SQL Injection.  
- Validação e sanitização corretas.  
- Uso de transações quando necessário.

**B. Arquitetura & Código**- Separação de responsabilidades.  
- Aderência a PSR-12.  
- Clareza e legibilidade.

**C. API & Funcional**- Endpoints funcionais.  
- Filtros e respostas corretas.  
- Estrutura RESTful.

**D. Testes**- Cobertura mínima exigida.  
- Casos de falha contemplados.

**E. Migração & Documentação**- Plano realista, aplicável.  
- README claro.

**Bônus**

- Validação customizada para CNPJ.  
- Seeder para dados de teste.  
- Uso de softDeletes.  
- Boa organização de commits.