Pontos Positivos

- Documentação muito bem feita e de fácil entendimento
- Projeto subiu e funcionou de primeira, sem ajustes necessários que não fossem nas configurações de ambiente
- Utilização de docker-compose
- Uso de testes unitários
- Boa utilização de enum para os códigos de erro
- Implementação interessante de flag ("nocache") para forçar a busca na api arquivei ao invés do banco

Pontos Negativos

- Não foi realizado o que foi proposto no desafio. O ponto 2 do desafio explicita que as notas consultadas na api devem ser todas salvas no banco até o fim dos registros. O projeto entregue apenas lista as notas em um endpoint e persiste apenas a nota buscada pelo outro endpoint
- Falta de padronização de codificação do PHP ferindo diversos princípios da PSR-1 e PSR-12: indentação incorreta, espaçamento entre chaves incorreto, sem padronização no nome das variáveis (algumas snake_case, outras camelCase, etc...) entre outros. Referência: https://www.php-fig.org/psr/
- Lógica apenas no controller, não foi criada camada de negócio ou abstração para casos de usos
- Credenciais de autenticação (header) inutilizadas: "DESAFIO_ID" e "DESAFIO_KEY"
- Não abstraiu funções nem utilizou outros princípios de código limpo
- Nao fez uso de logs
- Retornos inválidos no código (vide linha 34 do arquivo (app/Http/Controllers/Api/NfController.php que é inacessível)
- Falta de padronização do idioma utilizado no código: uso de português misturado ao inglês
- No controller houve falta de declaraçao de variáveis privadas ($Nfe) utilizadas enquanto outras variáveis declaradas nunca foram utilizadas ($Produto)

Pontos a Melhorar

- Ser mais cuidadoso com a apresentação do código. A falta de padronização absolutamente visível pecou bastante.
- É opcional, porém interessante, utilizar um makefile com comandos bash para facilitar o setup do projeto para outros desenvolvedores
- Usar menos arrays e mais objetos. Nfe foi tratada o tempo todo como array quando poderia ser uma classe onde seus atributos privados representassem os dados do documento
- Além do valor da nota também foi salvo o xml no banco (não solicitado). Na arquivei trabalhamos com um imenso volume de dados (bancos contendo até 20 milhões de documentos) e a persistência dos dados de xml causa um volume de dados desnecessário e impacta performance
- Novamente descuido com nomes: a tabela criada no banco se chama nfes (correta) porém sua migração tem nome "CreateNvesTable".

- Escolher apenas um idioma pra codificar. Como o framework tem o inglês por padrão sugiro também codar sua aplicação/api em inglês (comentários, nomes de rotas, funções, mensagens de erro). Se necessário apresentação do português para o cliente (por decisão de negócios), utilizar arquivos/chaves de tradução
- Separar a camada de negócio: criar casos de usos e estruturas desacopladas do framework. Seguindo os princípios de arquitetura em camadas o seu controller deve ter a responsabilidade de chamar as camadas mais internas da aplicação e não realizar a regra de negócio por si só
- Revisar o próprio código antes de realizar entregas. Volto a enfatizar que a apresentação pareceu descuidada.