

DESAFIO TÉCNICO — Desenvolvedor Full Stack (Node / React)

A Hyper possui um produto B2B que atende múltiplos clientes (multi-tenant), com grande volume de acessos e integrações externas.

Um dos módulos críticos do produto é o **processamento de eventos financeiros**, que precisa ser:

- confiável
- escalável
- observável
- fácil de evoluir

Você foi alocado para desenvolver a **primeira versão funcional** desse módulo.

❖ PROBLEMA

Crie um **serviço de processamento de transações** com as seguintes características:

- **Backend (obrigatório)**

Construa uma API usando **Node.js (NestJS ou Express)** que:

Funcionalidades mínimas:

- Receber transações financeiras via API
- Persistir as transações

- Garantir idempotência considerando concorrência e múltiplas requisições simultâneas.(a mesma transação não pode ser processada duas vezes)
- Disponibilizar um endpoint para consulta de transações

Requisitos técnicos:

- Banco de dados relacional (PostgreSQL, MySQL ou SQLite)
- Uso de TypeScript
- Estrutura organizada (controllers, services, repositories, etc.)
- Validações básicas
- Logs estruturados (mesmo que simples)
- Tratamento de erros consistente

Cenário importante

Considere que esse endpoint pode receber **alto volume de requisições simultâneas**.

Explique no README:

- Onde estaria o gargalo nesta implementação
- Qual seria o primeiro problema real em produção
- Qual solução você priorizaria primeiro e por quê

❖ Frontend (simplificado)

Crie uma interface simples em **React** que:

- Liste as transações
- Permita enviar uma nova transação
- Mostre erros de forma clara

Não estamos avaliando design visual, e sim:

- organização do código
- clareza
- separação de responsabilidades

❖ Arquitetura & Decisões (muito importante)

No README, responda **objetivamente**:

- Por que você organizou o projeto dessa forma?
- Onde você colocaria cache? Quando não colocaria?
- Como garantiria observabilidade em produção?
- Em que cenário você usaria fila/mensageria?
- O que você deixaria como dúvida técnica consciente?

❖ DIFERENCIAL (não obrigatório)

Escolha **apenas um** se quiser:

- Testes automatizados
- Uso de fila (BullMQ, RabbitMQ, etc.)
- Deploy fictício (ex: como faria na AWS)

✗ O QUE NÃO FAZER

- Copiar boilerplate sem explicar
- Criar algo gigantesco
- Usar IA sem entender o que está entregando
- Entregar sem README

Código sem explicação **vale menos** que código simples bem explicado.