Resumo Técnico Completo - Desafio 3: Physical Store

1. Objetivo

Criar uma API em NestJS que retorna lojas físicas com opções de entrega baseadas em um CEP.

Decidir entre entrega local (PDV) ou via Correios (LOJA) com base na distância.

2. Endpoints Obrigatórios

- GET /stores
- GET /stores/:id
- GET /stores/state/:uf
- GET /stores/cep/:cep

3. Entidade Store

Campos: storeID, storeName, type (PDV/LOJA), shippingTimeInDays, localização, etc.

4. Responses Esperadas

Response 1: Para listagem e busca por ID/estado

```
{
  "stores": [...],
  "limit": 1,
  "offset": 1,
  "total": 100
}
```

Response 2: Para busca por CEP, inclui pins do mapa

Resumo Técnico Completo - Desafio 3: Physical Store

{
  "stores": [...],
  "pins": [...],
  "limit": 1,
  "offset": 1,
  "total": 100
}

5. Lógica de Entrega

PDV (50km): Motoboy da loja, valor fixo R$ 15, prazo = shippingTimeInDays + configuração.

LOJA (> 50km): Via Correios (Sedex/PAC), prazo = retorno dos Correios + shippingTimeInDays.

6. APIs Externas Utilizadas

- Google Maps Distance Matrix API (distância)
- ViaCEP (dados de endereço via CEP)
- Correios (prazo/preço para Sedex e PAC)

7. Modelagem Adicional

StoreDeliveryConfig: Configuração de prazo extra por loja e tipo.

DeliveryCalculation: Registro temporário de cálculos de frete com TTL.

Delivery: Entregas confirmadas.

8. Rotas Recomendadas

Resumo Técnico Completo - Desafio 3: Physical Store

POST /delivery/calculate - calcula entrega e salva temporariamente.

POST /delivery/confirm/:calculationId - confirma e salva entrega no banco.

9. Boas Práticas e Requisitos Técnicos

- NestJS com princípios SOLID
- Testes unitários com mocks
- Swagger documentando todos os endpoints
- Uso de cache para otimizar reuso de cálculos de frete