Desafio Técnico — GoBots:

1. Objetivo

Construir duas APIs simples que se comunicam via webhook para simular um fluxo de marketplace:

1. **Marketplace API:** Gerencia pedidos e dispara eventos para webhooks cadastrados.
2. **Receiver API:** Recebe eventos, enriquece os dados consultando a Marketplace e persiste a informação.

O foco é um fluxo *end-to-end* funcional, com infraestrutura configurada via Docker e documentação para validação.

---

2. Escopo Obrigatório (MVP)

A. Marketplace API

Esta API é responsável pela gestão do ciclo de vida do pedido e pelo disparo de notificações.

Endpoints

Verbo	Rota	Descrição
POST	/orders	Cria um pedido. O pedido deve nascer com status CREATED.
GET	/orders/:id	Retorna os dados completos de um pedido pelo ID.
PATCH	/orders/:id/status	Atualiza o status de um pedido. Deve permitir a transição para PAID.
POST	/webhooks	Cadastra webhooks por loja (Store ID).

Payload de Cadastro de Webhook

JSON

```
None
{
  "storeIds": ["store_001", "store_002"],
  "callbackUrl": "http://receiver-api:port/events"
}
```

Eventos e Disparos
A Marketplace deve disparar webhooks nas seguintes situações:

- order.created: Imediatamente após criar o pedido.
- order.paid: Ao atualizar o status para PAID.

Regra de Negócio: O evento deve ser enviado **apenas** para webhooks cadastrados que contenham a storeId do pedido.

Payload Mínimo do Evento (Webhook):

JSON

```
None
{
  "event": "order.paid",
  "orderId": "order_123",
  "storeId": "store_001",
  "timestamp": "2025-10-27T10:00:00Z"
}
```

B. Receiver API

Esta API simula um sistema parceiro (ERP, Hub, etc.) que recebe notificações.

Endpoints

Verbo	Rota	Descrição
POST	/events	Endpoint que receberá o POST disparado pela Marketplace.

Comportamento Esperado

Ao receber um evento na rota /events:

1. Ler o orderId do payload recebido.
2. Consultar os detalhes do pedido na Marketplace API (GET /orders/:id).
3. Salvar no banco de dados um registro contendo:
    - O evento original recebido.
    - O *snapshot* (dados completos) do pedido retornado pela Marketplace.

---

3. Infraestrutura e Execução

O projeto deve ser "containerizado" para facilitar a execução e avaliação.

Requisitos Docker

- **Dockerfiles:** Um para cada API.
- **Docker Compose:** Um arquivo docker-compose.yml na raiz do projeto que orquestre:
    1. Marketplace API.
    2. Receiver API.
    3. Banco de Dados (Livre escolha: Postgres, Mongo, MySQL, Redis, etc.).

Requisitos de Rede

- O comando docker-compose up --build deve subir todo o ambiente sem necessidade de passos manuais extras.
- A **Receiver API** deve conseguir acessar a **Marketplace API** via hostname interno da rede do Docker.

---

4. Entregáveis

O repositório do projeto deve conter:

1. Código fonte das duas APIs.
2. Dois Dockerfiles.
3. Arquivo docker-compose.yml.
4. Arquivo README.md.

Conteúdo do README

O documento deve ser curto, direto e reproduzível, contendo:

- **Como subir o projeto:** O comando exato (ex: docker-compose up).
- **Como cadastrar um webhook:** Exemplo de curl ou JSON.
- **Como criar um pedido:** Exemplo de chamada HTTP.
- **Como validar o resultado:** Instruções de como verificar se o Receiver salvou os dados (comando SQL, logs do container, endpoint de listagem, etc.).

---

5. Escopo Opcional (Diferenciais)

Não é obrigatório, mas conta pontos extras:

- Novos eventos: order.shipped, order.completed, order.canceled.
- Validação de máquina de estados (ex: não permitir ir de CREATED direto para SHIPPED).
- Mecanismo de **Retry/Backoff** no envio do webhook em caso de falha.
- **Idempotência** no Receiver (evitar salvar o mesmo evento duas vezes).
- Healthcheck configurado no Docker Compose.
- Testes unitários ou de integração.

---

6. Critérios de Avaliação

Critério	**O que será avaliado**
Funcionalidade	Fluxo completo (Criação → Disparo → Enriquecimento → Persistência). Respostas HTTP corretas.
Qualidade de Código	Organização em módulos/camadas, nomenclatura limpa e tratamento de erros.
Integração	A comunicação HTTP entre os contêineres Docker funciona corretamente.
Docker	O ambiente sobe com um único comando e a rede está configurada.
Documentação	Facilidade em rodar e testar o projeto seguindo o README.

Nota: A solução deve ser simples, funcional e bem documentada. Uma entrega pequena e completa (MVP funcionando) vale mais do que um escopo grande incompleto ou com bugs.