Programação Funcional

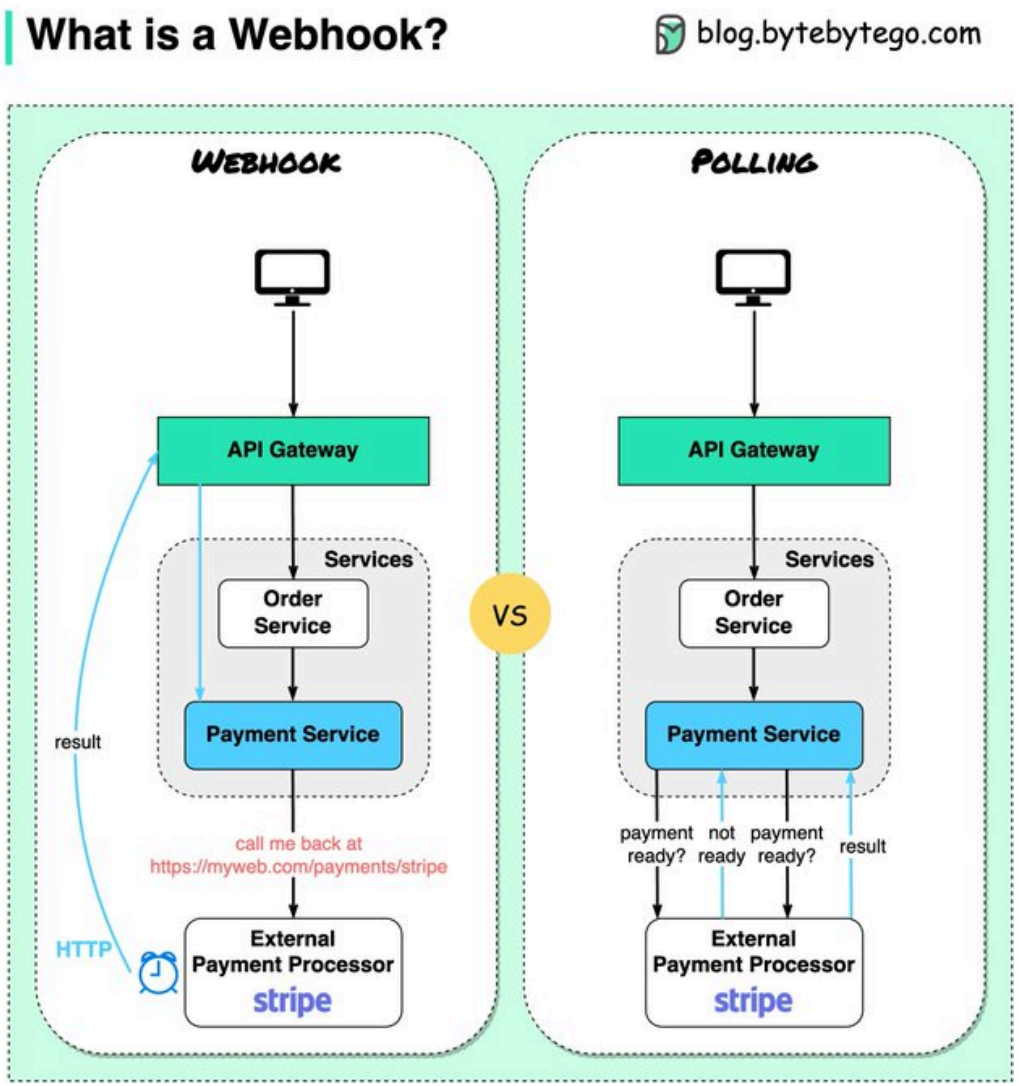
Aula 24

Raul Ikeda
2025-1

Esta Aula

- Project Webhook

Contexto



Uma loja está com todo o sistema de webcommerce pronto e precisa apenas integrar um meio de pagamento. A empresa gostaria de utilizar um gateway específico que trabalha com uma arquitetura chamada no modelo Webhook

Quando um pagamento é confirmado em um gateway (exemplo: Paypal ou MercadoPago), ele envia automaticamente uma requisição POST para uma URL de webhook configurada pela sua aplicação, contendo os dados do pagamento.

O que é um Webhook

Webhook é um mecanismo de comunicação entre sistemas que permite que um servidor notifique outro automaticamente quando um determinado evento ocorre, sem que o segundo precise perguntar continuamente (evitando polling).

Tecnicamente, Webhook é uma chamada HTTP feita por um sistema para um endpoint previamente configurado em outro sistema, disparada por um evento específico.

Características principais:

- Baseado em eventos (event-driven)
- Comunicação assíncrona
- Usa requisições HTTP (geralmente POST)
- O sistema receptor deve estar pronto para receber e processar a requisição

Como Funciona?

- O Cliente faz uma requisição para iniciar um processo (ex: pagamento) e é redirecionado para a página do Gateway de Pagamento.
- O Cliente realiza o pagamento normalmente na forma como desejar (PIX, cartão, etc).
- Após o pagamento, o site redireciona de volta para a página da loja.
- Ao mesmo tempo, o Gateway envia uma requisição no Webhook informando do evento de pagamento.
- O webhook é encarregado de confirmar e registrar o pagamento.
- O front então é atualizado com o resultado do processamento do Webhook.

Quais são os pontos de atenção

- O webhook precisa ser seguro (HTTPS), rápido e confiável
- Precisa verificar se a mensagem veio de uma fonte confiável
- Precisa lidar com falhas de envio e duplicações
- Uso de filas ou retry mechanisms é comum

Restrições do Problema

- Requisições HTTP
- Comunicação assíncrona
- Serialização/Deserialização (JSON)
- Lidar com falhas de rede
- Garantir idempotência no webhook

Por que usar Programação Funcional?

- Sem efeitos colaterais: facilita testar e depurar
- Imutabilidade: menos chance de bugs por concorrência
- Funções puras: isolamento claro de lógica
- Composição: fácil criar pipelines de transformação

Estrutura Esperada do Projeto

- Um servidor HTTP (ou HTTPS) que recebe uma mensagem POST com um payload:

```
{ "event": "payment_success",
  "transaction_id": "abc123",
  "amount": 49.90,
  "currency": "BRL",
  "timestamp": "2025-05-11T16:00:00Z" }
```

- O serviço deve conferir se o payload está correto.
- O serviço deve verificar se o pagamento é realmente correto.
- O serviço precisa verificar a unicidade do pagamento

Detalhes Operacionais

- Se uma transação estiver ok, deve-se retornar 200 e fazer um request em uma url de confirmação
- Se uma transação não estiver ok, não deve retornar 400
- Se alguma informação estiver errada (ex: valor), deve-se cancelar a transação fazendo um request
- Se alguma informação faltante (exceto transaction_id), deve-se cancelar a transação fazendo um request
- Se o token estiver errado, é uma transação falsa e deve-se ignorá-la

Requisitos do Projeto

- Criar um serviço HTTP que expõe uma rota para o verbo POST
- Tem que passar no teste mínimo fornecido
- Individual
- Pode ser feita em qualquer linguagem funcional
- Montar um README perfeito: descritivo do projeto, indicação de como instalar/rodar
- Data da entrega: 10/Jun/2025 às 23:59 via GitHub

Rubrica

- I: Se não houver entrega ou for irrelevante
- D: Se o projeto estiver incompleto
- C: O projeto passou no teste mínimo

Itens opcionais (+1/2 conceito para cada):

- O serviço deve verificar a integridade do payload
- O serviço deve implementar algum mecanismo de veracidade da transação
- O serviço deve cancelar a transação em caso de divergência
- O serviço deve confirmar a transação em caso de sucesso
- O serviço deve persistir a transação em um BD
- Implementar um serviço HTTPS

Atraso: desconto de 1 conceito

Testando o projeto

Há um arquivo Python (test_webhook.py) no Blackboard para realizar os testes, incluindo os itens opcionais. Ele não captura gravar em BD

Ele usa um payload padrão:

```
{ "event": "payment_success",
  "transaction_id": "abc123",
  "amount": 49.90,
  "currency": "BRL",
  "timestamp": "2025-05-11T16:00:00Z" }
```

Mas é possível também passar outros valores na execução.

Requirements: `fastapi`, `uvicorn` e `requests`

Next Class

- Interviews Simulation