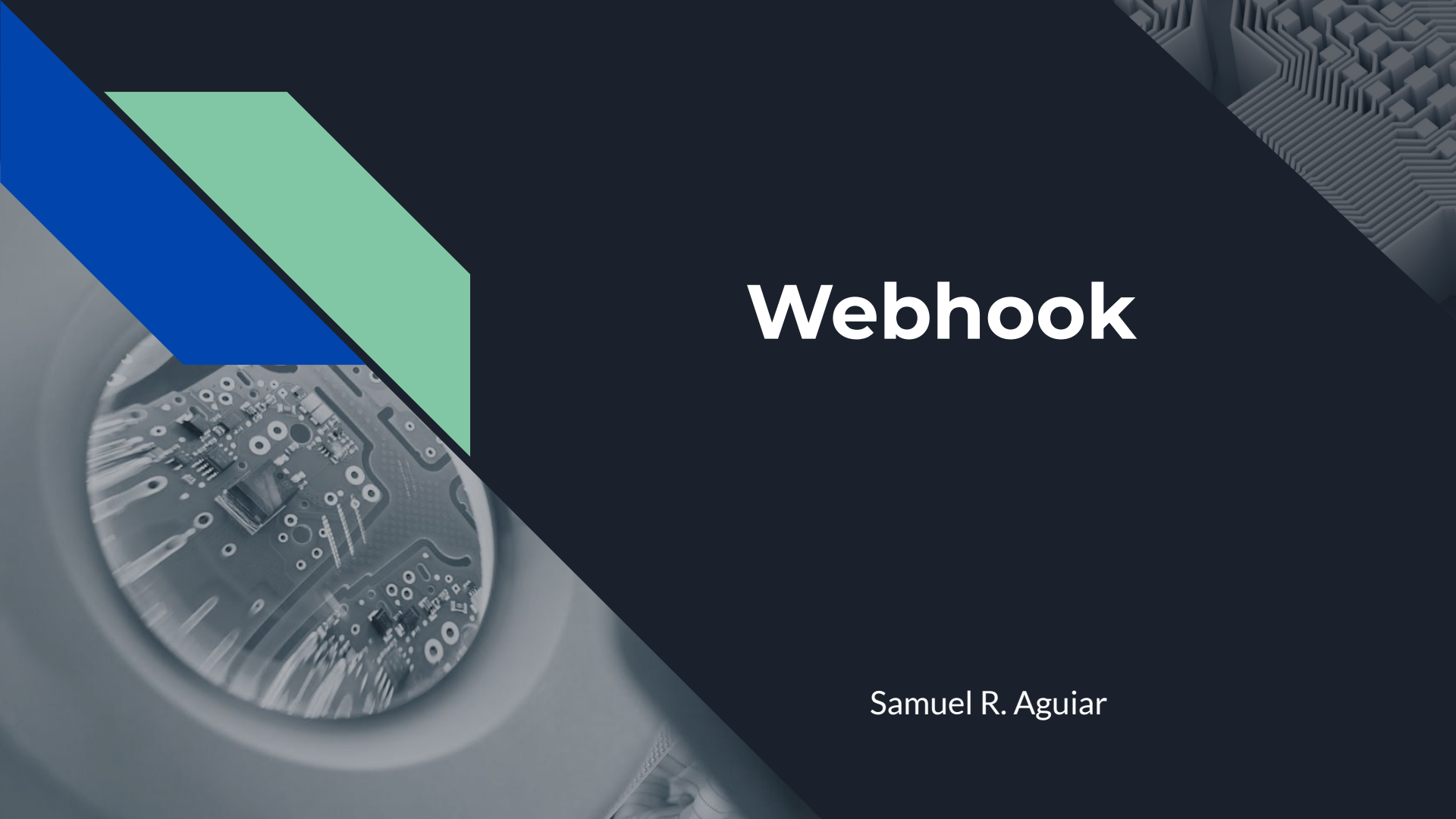

Webhook

Samuel R. Aguiar

Definição

Definição

Imagine um contexto em que temos um aplicativo **A**, que precisa ser notificado automaticamente sempre que um evento específico ocorra em outro aplicativo **B**. **Webhooks** tornam esse tipo de comunicação possível.

Definição

O objetivo do webhook é atuar como uma interface de comunicação externa, notificando um sistema a partir de eventos ocorridos em outra aplicação.

Aplicações

Exemplo 01 - Cenário

Em um contexto prático, suponha que você esteja desenvolvendo um *e-commerce* no qual o pagamento de cada compra é delegado a uma plataforma externa, que já tem integração com diversos sistemas bancários.

Exemplo 01 - Cenário

Como os pagamentos são feitos de forma externa, o sistema do e-commerce precisa obter informações sobre eles de alguma forma.

Exemplo 01 - Cenário

Quando uma compra é feita e um cliente paga, como saber se deu tudo certo? A pergunta precisa ser respondida para continuar com o processo.

Você pode programar o e-commerce para **enviar requisições** periódicas ao sistema de pagamento solicitando atualizações (processo também conhecido como *polling*).

Exemplo 01 - Cenário

Ou ainda, no caminho inverso, você pode usar um **webhook** para receber uma notificação do sistema de pagamentos assim que alguma resposta estiver disponível, sem ter essa informação tenha que ser solicitada pelo e-commerce.

Exemplo 01 - Cenário

Um webhook cria uma comunicação entre seu negócio e a plataforma de pagamentos.

Exemplo 01 - Cenário

Integrado ao seu projeto, ele funcionará como **callback HTTP**, permitindo que a cada pagamento realizado, o e-commerce seja notificado para prosseguir com os seus processos de vendas, eliminando a necessidade de consultas constantes.

Vantagens e Desvantagens

Vantagens

Vantagens

- **Eliminação de consultas frequentes entre APIs, por eles serem ativados a partir de eventos específicos.**

Dessa forma, não precisamos realizar consultas frequentes para verificar, por exemplo, mudanças de status (*polling*), já que o sistema é notificado automaticamente.

Desvantagens

Desvantagens

- **Necessidade de implementação de medidas robustas de segurança entre eles e os sistemas** para garantir a **confiabilidade** das conexões e dos dados enviados e recebidos.

Desvantagens

- Além disso, existe uma **complexidade para lidar com erros e bugs gerados durante a callback do webhook,** sendo necessário dispor de recursos para monitorar a detecção de falhas.

Exemplo : Prático

Exemplo: Prático

-> Entre na sua instância da AWS via terminal

```
sudo ssh  admin@ip-ou-dominio  -i  /caminho/chave/aws/chave.pem
```

-> instale as bibliotecas:

```
pip install fastapi uvicorn
```

-> Crie o arquivo do seu código:

EX:

```
nano teste.py
```

-

Exemplo: Prático

-> Cole esse código no arquivo e salve:

```
from fastapi import FastAPI, Request

app = FastAPI()

@app.post("/webhook")
async def github_webhook(request: Request):
    # Pega os dados do GitHub
    data = await request.json()
    event_type = request.headers.get("X-GitHub-Event")  # Tipo do evento

    print("\n--- EVENTO DO GITHUB ---")
    print(f"Tipo de evento: {event_type}")
    print(f"Repositório: {data.get('repository', {}).get('full_name')}")
```

Exemplo: Prático

```
# Detalhes específicos por tipo de evento
    if event_type == "push":
        print(f"Commit ID: {data.get('after')}")
        print(f"Branch: {data.get('ref').split('/')[-1]}")
        print(f"Autor: {data.get('pusher', {}).get('name')}")

    elif event_type == "pull_request":
        pr_action = data.get("action")  # opened, closed, merged
        pr_title = data.get("pull_request", {}).get("title")
        print(f"Ação do PR: {pr_action}")
        print(f"Título do PR: {pr_title}")

    elif event_type == "issues":
        issue_action = data.get("action")  # opened, closed
        issue_title = data.get("issue", {}).get("title")
        print(f"Ação da Issue: {issue_action}")
        print(f"Título da Issue: {issue_title}")
```

Exemplo: Prático

```
    else:
        print("Outro tipo de evento. Dados completos:")
        print(data)

    return {"status": "ok", "event": event_type}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8080)
```

Exemplo: Prático

-> Execute no terminal da AWS:

```
python3 teste.py
```

No GitHub:

1. Vá em **Settings > Webhooks > Add webhook**
2. Configure:
   - **Payload URL**: http://SEU_IP:8080/webhook
   - **Content type**: application/json
   - **Events**: Seleciona os que você quer (ou "Send me everything")

Exemplo: Prático

-> Faça alguma alteração no repositório

Ex: commit um novo arquivo dentro do repositório

-> Vai no terminal da AWS onde o servidor vai está rodando

O retorno esperado é esse:

Exemplo: Prático

-> O retorno esperado é esse:

```
admin@ip-172-31-26-138:~$ ls
__pycache__  noip-duc_3.3.0         server.py  venv     webhook.py
main.py      noip-duc_3.3.0.tar.gz  teste.py   webhook
admin@ip-172-31-26-138:~$ python3 teste.py
INFO:     Started server process [651]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
--- EVENTO DO GITHUB ---
Tipo de evento: push
Repositório: DevSamuelAguiar/teste2-webhook
Commit ID: d612b155f24f9cfb023e5bd704016cf4a0288cf5
Branch: desenvolvimento
Autor: DevSamuelAguiar
INFO:     140.82.115.80:57156 - "POST /webhook HTTP/1.1" 200 OK
```

teste2-webhook

Referências

Referências

https://www.alura.com.br/artigos/webhooks?srsltid=AfmBOoqOFoW-b_RFBVrhxI1fJt2MxGhT_F8XYIkR1G11aT66WtmujPCa#vantagens-e-desvantagens