

HTTP: 202

http 202 Accepted

Desenvolvimento de Sistemas Distribuídos

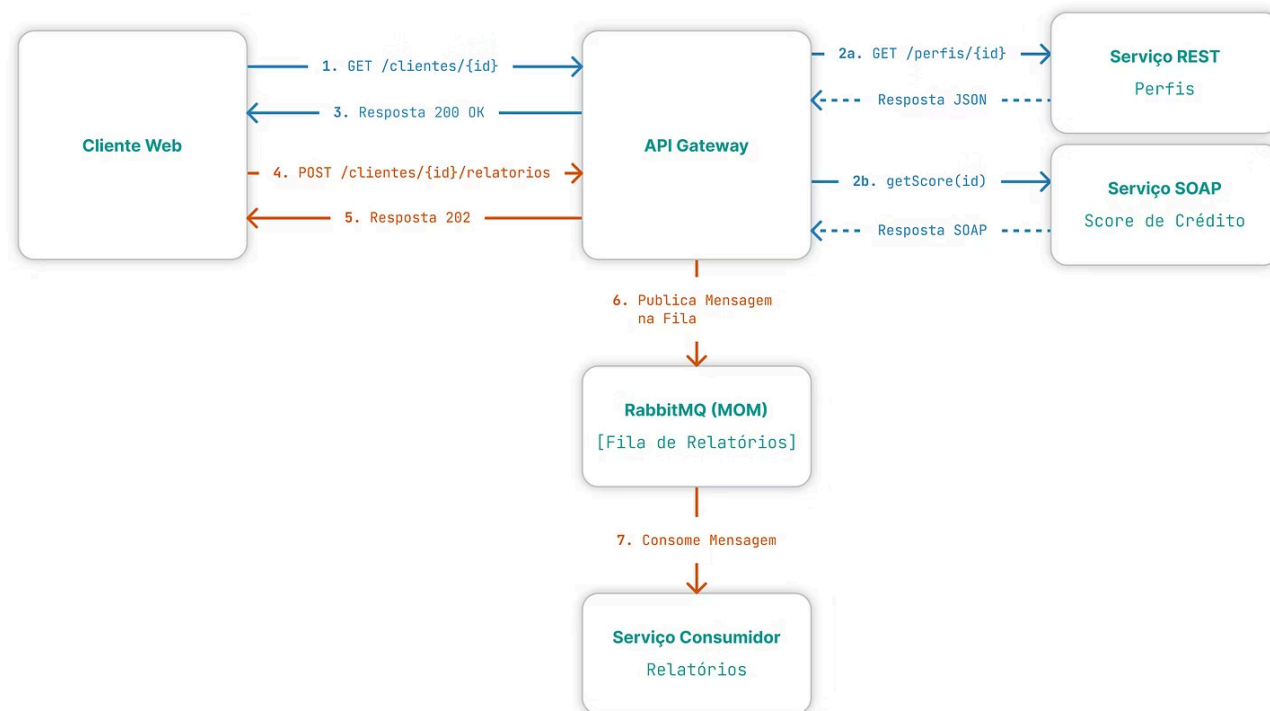
# Arquitetura Híbrida com API Gateway

Integrando Serviços Síncronos (REST/SOAP) e Assíncronos (MOM)



por Maycon Alexsander da Silva

# Arquitetura



## Consulta de Perfil de Cliente

 Buscar Perfil Gerar Relatório

### Dados do Cliente

ID: 123  
Nome: João da Silva  
Email: joao.silva@exemplo.com  
Idade: 34  
Score de Crédito: 450

### Ações Disponíveis

Self Extrato Solicitar Cartao

# Fluxo Síncrono: Consulta Agregada de Dados



## Cliente

Envia **GET** /clientes/{id}



## Gateway

Orquestra chamadas REST e SOAP paralelas



## Serviços

REST (Node.js) e SOAP (Java/JAX-WS)

As respostas são **agregadas** e retornadas em um único payload JSON com links HATEOAS.

# Fluxo Assíncrono: Processamento de Tarefas Demoradas

1

## Cliente

Envia **POST** /clientes/{id}/relatorios

2

## Gateway (Produtor)

Publica mensagem na fila do RabbitMQ e responde com 202 Accepted

3

## Consumidor

Processa a mensagem da fila no seu próprio tempo

### Consulta de Perfil de Cliente

Buscar Perfil

Gerar Relatório

Gateway respondeu (Status 202): Pedido de relatório recebido e sendo processado.

O cliente recebe a confirmação (HTTP 202) **instantaneamente**.

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS  MEMORY  XRTOS
@mayconalex → /workspaces/DSO-2025.1-Tarefa04/banco-integrado-api/servico-relatorios-consumer (main) $ node consumer.js
Iniciando consumidor de relatórios...
Conectado ao RabbitMQ com sucesso!
[*] Aguardando por pedidos de relatório na fila: "fila_de_relatorios". Para sair, pressione CTRL+C

[4:06:35 PM] Recebido pedido: {"clienteId":"123","tipoRelatorio":"extrato_anual","solicitadoEm":"2025-08-08T16:06:35.560Z"}
Processando o relatório... (isso levará 5 segundos)
[4:06:40 PM] Relatório para {"clienteId":"123","tipoRelatorio":"extrato_anual","solicitadoEm":"2025-08-08T16:06:35.560Z"} foi gerado com sucesso!
```

O Consumidor **retira** a mensagem da fila e **executa a tarefa demorada**.

# Ecosystema Tecnológico



## Frontend

HTML5, JavaScript (Fetch API)



## Gateway/Backend

Node.js (Express, Axios, amqplib)



## Serviço Legado

Java (JAX-WS, Maven)



## Mensageria

RabbitMQ (via Docker)



## Documentação

OpenAPI 3.0 (Swagger)

## Interoperabilidade SOAP:

✓ O contrato **WSDL** gerado pelo serviço Java foi consumido por um cliente **Python (Zeep)**.

PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO **TERMINAL** PORTAS 20 MEMORY XRTOS

```
• @mayconalex → /workspaces/DSD-2025.1-Tarefa04/banco-integrado-api/teste-cliente-soap (main) $ python test_client.py
Tentando conectar ao WSDL em: https://verbose-meme-gr7w774vqjpf55-8080.app.github.dev/score-service?wsdl
Conexão com o WSDL bem-sucedida!
```

```
Consultando score para o cliente ID: 123...
Resultado -> Score: 750
```

```
Consultando score para o cliente ID: 999...
Resultado -> Score: 450
```

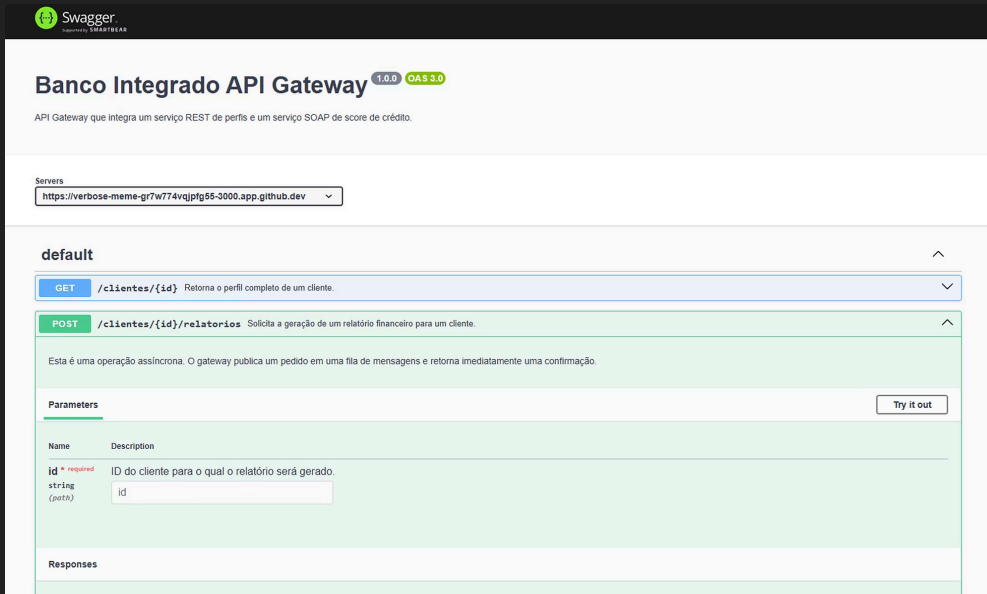
+ v ... | [ ] x

- Consumer servico-relatorios...
- SOAP score-soap-service
- REST servico-perfis-rest
- API Gateway gateway
- Cliente Web cliente-web
- bash teste-cliente-soap

# Validação da API e Documentação

**Objetivo:** Apresentar os artefatos que comprovam a funcionalidade e a qualidade da API.

## Documentação da API:



Interface do **Swagger UI**, mostrando os dois endpoints (GET e POST) com suas descrições. A API é autodocumentada e interativa.

## Resposta HATEOAS:

```
{
  "id": 123,
  "nome": "João Silva",
  "email": "joao.silva@exemplo.com",
  "idade": 34,
  "scoreCredito": 750,
  "_links": {
    "self": {
      "href": "/clientes/123"
    },
    "extrato": {
      "href": "/clientes/123/extrato"
    },
    "solicitar_cartao": {
      "href": "/clientes/123/cartoes"
    }
  }
}
```

Trecho do JSON retornado pelo endpoint GET, destacando o bloco `_links`. Isso permite a descoberta dinâmica de ações pela API.

# Hands-On !

mayconalex/**DSD-2025.1-Tarefa04**



1

Contributor



0

Issues



0

Stars



0

Forks



GitHub



GitHub – mayconalex/DSD-2025.1-Tarefa04

Contribute to mayconalex/DSD-2025.1-Tarefa04 development by creating an account on GitHub.

**Acessar repositório**