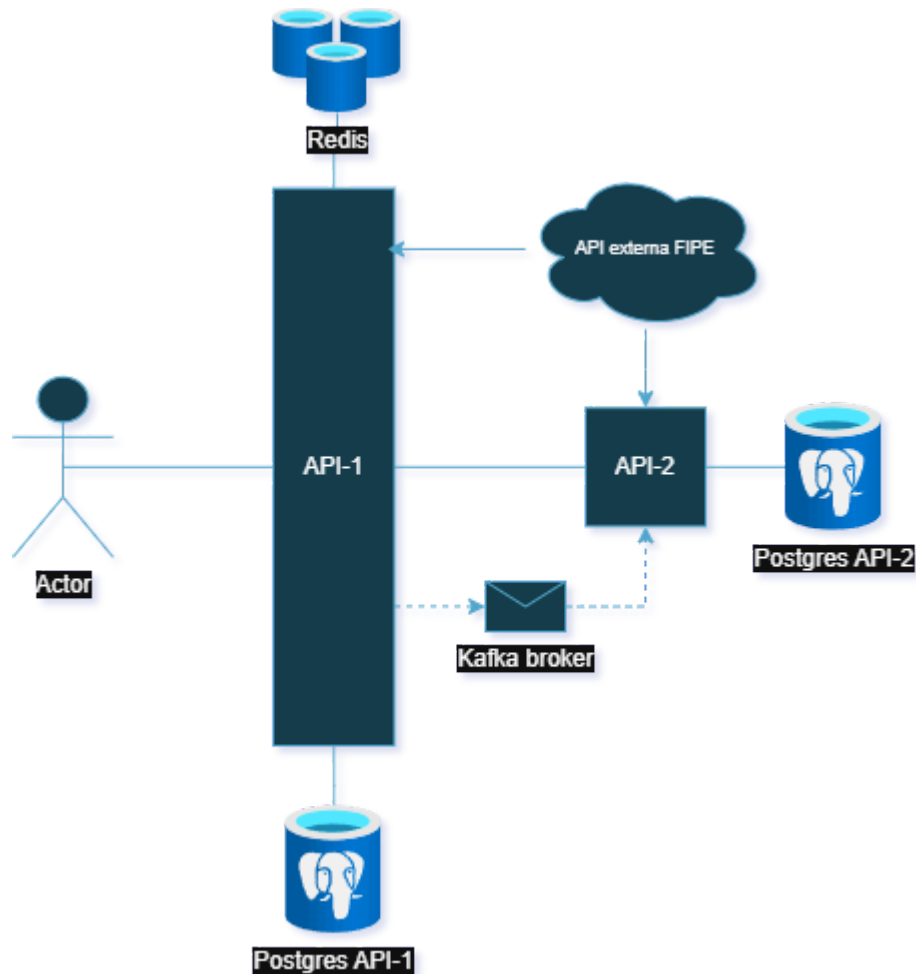


Gateway Fipe

Arquitetura desenvolvida em **Java 21** com **Spring Boot 3.5.8**, utilizando Kafka, Redis, PostgreSQL, comunicação assíncrona e cache distribuído. O sistema é formado por 2 microserviços: um gateway (orchestrator) e um consumer responsável por integrar com a API externa.

Arquitetura do projeto



Pré-requisitos

- Clone o repositório:

```
git clone https://github.com/Alhexx/sciensa-fipe.git
```

Passo a para execução

1. Configuração de um .env:
- Crie um arquivo .env e preencha com, por exemplo:

```
SPRING_DATASOURCE_USERNAME=postgres
SPRING_DATASOURCE_PASSWORD=postgres
SPRING_DATASOURCE_API1_URL=jdbc:postgresql://postgreapi1:5432/api1
SPRING_DATASOURCE_API2_URL=jdbc:postgresql://postgreapi2:5432/api2

SECURITY_JWT_SECRET=secret-de-testes-para-localhostt

EXTERNAL_FIPE_URL=https://parallelum.com.br/fipe/api/v1

EXTERNAL_FIPE_CONSUMER_API_URL=http://fipeconsumer:8081

SPRING_KAFKA_BOOTSTRAP_SERVERS=kafka:9092

REDIS_HOST=redis
REDIS_PORT=6379

APP_KAFKA_TOPICS_BRANDS=fipe-brands

CORS_ALLOWED_ORIGIN=http://fipeorchestrator:8081
```

Isso inicializará:

- Microserviço Gateway (API-1)
- Microserviço Consumer (API-2)
- PostgreSQL (2 bancos distintos)
- Redis
- Kafka
- Toda a comunicação interna via rede Docker

1. Execute o Docker compose para ter a aplicação completa rodando:

```
docker compose up -d --build
```

Apos a inicialização do serviço:

- Acesse <http://localhost:8080/swagger-ui/index.html#/> para visualizar a documentação interativa das rotas.

Funcionamento do Cache

Cache Name	Descrição	TTL
marcas	Lista de marcas de veículos	1h

modelosPorMarca Modelos de uma marca específica (/ {brandNameOrCode}) 1h

O cache é populado nas primeiras requisições e subsequentemente lido diretamente do Redis.

Endpoints

Criação de usuário

POST /users

Corpo:

```
{
  "name": "Fulano",
  "email": "fulano@gmail.com",
  "cpf": "60681101091", # Precisa ser válido
  "password": "string"
}
```

Login de usuário

POST /auth

Corpo:

```
{
  "nome": "fulano",
  "observacoes": "string"
}
```

Exemplo de resposta:

```
{
  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhbGV4YW5kcmVAZ21haWwuY29tIiwiaWF0Ij0i"
}
```

Rotas com Autenticação

Load de Marcas para o Kafka

```
POST /load/initial
```

Comportamento:

- Envia as marcas vindas da API externa de Fipe para o topico do kafka.

Listar marcas

```
GET /api/brands
```

Exemplo de resposta:

```
[
  {
    "nome": "AM Gen",
    "codigo": "4"
  },
  {
    "nome": "ASTON MARTIN",
    "codigo": "189"
  },
  ...
]
```

Listar modelos de uma marca

```
GET /api/models/{brandNameOrCode}
```

Exemplo de resposta:

```
[
  {
    "codigo": "1",
    "nome": "Integra GS 1.8",
    "observacoes": null,
    "marca": "Acura"
  },

```

```
{
  "codigo": "2",
  "nome": "Legend 3.2/3.5",
  "observacoes": null,
  "marca": "Acura"
},
{
  "codigo": "3",
  "nome": "NSX 3.0",
  "observacoes": null,
  "marca": "Acura"
}
]
```

Atualizar um modelo

PATCH /api/models/{code}

Corpo:

```
{
  "nome": "string",
  "observacoes": "string"
}
```

Ambos os campos são opcionais (rota de Patch)

Comportamento:

- Invalida o cache da marca automaticamente:

Observações

- Gateway consta com validação de CPF
- Gateway consta com tratamento de exceções, esperadas e inesperadas
- JWT implementado, token expirando 10 minutos, por padrão
- Arquitetura baseada em camadas inspirada em DDD
- Commits e escrita do código em inglês, mas erros todos em português