

# README.md

## Microsserviço de Gestão de Entregas (Reuso de Software)

Este projeto implementa o microsserviço "Serviço Entregas" em Python (Flask) para calcular fretes e rastrear encomendas, integrando-se a uma API Mock para demonstrar a Integração com Reuso de Software.

### 1. Classificação da Integração

Critério	Classificação	Justificativa
Contexto de Negócio	B2B (Business-to-Business)	Integração entre o e-commerce (nosso sistema) e o provedor de serviços logísticos.
Padrão Arquitetural	Microsserviços	A lógica de integração e persistência é isolada no ServiçoEntregas.
Padrão de Comunicação	Síncrona (Requisição/Resposta)	O ServiçoEntregas aguarda a resposta imediata da API (HTTP/REST).

### 2. Requisitos Atendidos e Qualidade de Código

Requisito	Implementação	Qualidade (Código Limpo)
API Consumida	Implementada via LogisticaGateway (usando requests).	Uso de Serviço Separado (Gateway/Adapter).
Tolerância a Falhas	Padrão <b>Retry</b> com Tenacity (3 tentativas em 503).	Tratamento de Exceções customizadas.

<b>Persistência de Dados</b>	Simulação de histórico no EntregaRepository.	Uso de <b>Injeção de Dependência</b> (Service recebe Gateway e Repository).
------------------------------	--	---

### 3. Passos para Execução

#### Pré-requisitos

Instale as dependências: `pip install flask requests tenacity`

#### Passo 1 e 2: Iniciar Servidores (Terminais 1 e 2)

Inicie o Mock (5001) e o Serviço Principal (5000) em janelas separadas.

```
Terminal 1: Mock
python mock_correios_api_app.py

Terminal 2: API Principal
python -m servico_entregas.api
```

### 4. PROVA FINAL DE VALIDAÇÃO

A comprovação funcional de todos os requisitos é feita através do script `teste_manual.py`, que substitui a execução direta do Pytest para uma demonstração mais estável.

#### Passo 3: Executar a Demonstração (Terminal 3)

Execute o script `teste_manual.py` com os servidores (Terminais 1 e 2) rodando:

```
python teste_manual.py
```

#### Análise da Saída (Comprovação da Robustez)

Cenário Testado	Saída Comprovada	Critério Atendido
<b>Cálculo/Rastreo Sucesso</b>	Status <b>200 OK</b>	API Consumida Corretamente.

<b>Falha de Negócio</b> (400 e 500)	Status <b>400</b> e <b>500</b>	Erros de validação e de serviço tratados corretamente pelo API.
<b>FALHA CRÍTICA</b> <b>(RETRY)</b>	Status <b>503</b> com <b>Tempo de Resposta &gt; 12s</b>	O atraso prova que o LogisticaGateway tentou a conexão 3 vezes antes de falhar, cumprindo o requisito de <b>Tolerância a Falhas</b> .