



Pipeline de Cotações Cambiais com Python + LLM

Alexandre Roberto - 1520357

bianca pereira de lima lourenço - 2301749

Vinícius Julio - 2500392

Henrique Ferreira Santos - 2502307

Silvio Cezar Teles - 2502503

1. Definição e objetivos

O projeto tem como objetivo construir um pipeline completo de ETL (Extract, Transform, Load) para ingestão e processamento de cotações cambiais, aplicando técnicas de normalização e enriquecimento com insights de um LLM (Large Language Model). A entrega final inclui não apenas a parte técnica (código, testes e automação), mas também os artefatos executivos (documentação e apresentação para diretoria).

2. Estrutura inicial

A arquitetura do projeto foi organizada da seguinte forma:

```
exchange-rate-pipeline-llm/
├── src/                # código-fonte principal
│   ├── ingest.py       # captura dados da API
│   ├── transform.py    # normalização e limpeza
│   ├── load.py         # exporta para Parquet e DB
│   ├── llm_insights.py # persistência dos insights do LLM
│   └── __init__.py
├── data/
│   ├── raw/            # dados brutos da API
│   └── silver/         # dados normalizados
```

```

├── gold/          # dados finais em Parquet e insights
├── tests/         # testes unitários e de integração
├── logs/          # logging estruturado
├── exchange_rate.db # banco SQLite para persistência
├── README.md      # documentação
├── requirements.txt # dependências
├── .env           # variáveis locais (não versionadas)
└── .env.example   # modelo de configuração

```

3. Implementações técnicas

Ingestão dos dados (/data/raw)

- Dados brutos salvos em JSON/CSV
- Uso de requests com timeout e tratamento de erros HTTP
- Retry exponencial usando tenacity
- Salvamento do JSON bruto em /data/raw com nome padronizado YYYY-MM-DD.json e sufixo _HHMMSS se houver múltiplas coletas no mesmo dia
- Inclusão de metadados no JSON (timestamp, status HTTP, URL consultada)
- Logging estruturado em JSON (INFO/ERROR) com run_id e arquivo
- Escrita atômica (salvar em tmp e depois renomear)

Transformação (/data/silver)

- Normalização de colunas e tipos
- Remoção de taxas inválidas
- Leitura do JSON bruto correspondente à data
- Normalizar em DataFrame com colunas obrigatórias (base_currency, target_currency, rate, retrieved_at, date)
- Validação: nenhuma taxa nula, zero ou negativa; base_currency correto
- Conversão de tipos (rate como float/Decimal, arredondamento 6 casas)
- Remoção de duplicatas (target_currency + retrieved_at)
- Arquivo de rejeitados em /data/raw/rejects com motivo
- Logging detalhado de erros
- Escrita em /data/silver/YYYY-MM-DD.parquet (engine pyarrow, compressão snappy, index=False)

Carga (/data/gold)

- Salvamento em Parquet e SQLite
- Agregar arquivos /data/silver e gerar /data/gold/YYYY-MM-DD.parquet
- Inserir metadados (pipeline_version, run_id, run_timestamp)
- Opcional: carregar em banco via SQLAlchemy, if_exists='append', chunksize=1000
- Garantir índice único (date + base_currency + target_currency) para evitar duplicatas

Enriquecimento

- Calcular métricas: pct_change, volatilidade, top movers
- Criar resumo compacto (JSON ou tabela com top 5 moedas)

- Chamar LLM (OpenAI API) com template de prompt
- Logging do prompt e resposta em /logs/llm_prompts.log

Testes e Logging (/tests, /logs)

- Unitários: ingest, transform, load
- Garantir que pytest roda sem falhas
- Logging estruturado: eventos registrados em logs
- Cobrem ingestão, transformação e carga
-

Integração com LLM

- Persistência de insights em JSON no /data/gold

Variáveis de ambiente

- Criado .env.example com placeholders

5. Resumo final

- Pipeline ETL concluído.
- Integração com LLM implementada.
- Testes automatizados todos passando.
- Boas práticas aplicadas (.env.example, logging estruturado).

6. Imagens

Testes:

```
(venv) C:\Users\Ale Roberto\Documents\GitHub\exchange-rate-pipeline-llm>pytest -v
===== test session starts =====
platform win32 -- Python 3.10.2, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\Ale Roberto\Documents\GitHub\exchange-rate-pipeline-llm\
venv\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Ale Roberto\Documents\GitHub\exchange-rate-pipeline-llm
plugins: anyio-4.11.0
collected 7 items

tests/test_ingest.py::test_fetch_exchange_rates PASSED [ 14%]
tests/test_integration.py::test_full_pipeline PASSED [ 28%]
tests/test_llm_insights.py::test_save_llm_insights PASSED [ 42%]
tests/test_load.py::test_save_to_parquet PASSED [ 57%]
tests/test_load.py::test_save_to_sqlite PASSED [ 71%]
tests/test_transform.py::test_transform_file PASSED [ 85%]
tests/test_transform_metrics.py::test_transform_file PASSED [100%]

===== 7 passed in 1.78s =====
```

Dashboard



