Alexandre Roberto - 1520357

bianca pereira de lima lourenço - 2301749

Vinícius Julio - 2500392

**Henrique Ferreira Santos - 2502307** 

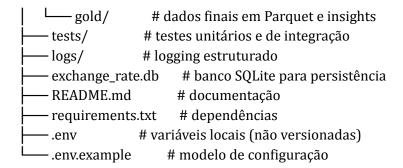
Silvio Cezar Teles - 2502503

# 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:



# 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-pipelne-llm>pytest -v

platform win32 -- Python 3.10.2, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\Ale Roberto\Documents\GitHub\exchange-rate-pipelne-llm\

venv\Scripts\python.exe
cachedir: .pytest_cache

rootdir: C:\Users\Ale Roberto\Documents\GitHub\exchange-rate-pipelne-llm

plugins: anyio-4.11.0
collected 7 items

tests/test_ingest.py::test_fetch_exchange_rates PASSED
[ 14%]

tests/test_lim_insights.py::test_full_pipeline PASSED
[ 28%]

tests/test_load.py::test_save_tlm_insights PASSED
[ 42%]

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%]
```

# **Dashboard**

