🚀 JIRA Data Engineering Pipeline – Medallion Architecture

Este projeto implementa um pipeline de Engenharia de Dados robusto para processar dados do JIRA, utilizando a Arquitetura Medallion para transformar dados brutos em inteligência de negócio sobre SLA (Service Level Agreement).

BR Versão em Português

📋 Objetivo

O objetivo deste projeto é automatizar a ingestão e o processamento de chamados do JIRA para calcular o tempo de resolução em **horas úteis**, desconsiderando finais de semana e feriados nacionais.

🏗️ Arquitetura do Pipeline

O projeto segue a **Arquitetura Medallion**, garantindo organização e rastreabilidade:

- **Camada Bronze**: Ingestão de arquivos JSON brutos do Azure Blob Storage utilizando autenticação via Service Principal.
- **Camada Silver**: Limpeza e normalização dos dados. Extração de campos aninhados (`assignee`, `timestamps`) e conversão para o formato **Parquet** visando performance e preservação de metadados.
- **Camada Gold**: Aplicação de regras de negócio. Cálculo de SLA baseado na prioridade (High: 24h, Medium: 72h, Low: 120h), integrando com a **BrasilAPI** para identificação de feriados.
- **Data Quality**: Auditoria automática de integridade, volumetria e validação de nulos ao final do processo.

🛠️ Tecnologias e Boas Práticas

- **Python 3.x** e **Pandas** para manipulação de dados.
- **PyArrow**: Engine de escrita Parquet estável para ambiente Windows.
- **Segurança**: Uso de variáveis de ambiente (`.env`) e proteção de dados sensíveis via `.gitignore`.
- **Modularização**: Código dividido em módulos específicos para cada etapa do processo.

📊 Evidências de Execução e Qualidade

O pipeline conta com um orquestrador central que valida cada etapa. Em execuções de teste, o sistema processou com sucesso:

- **Funil de Dados**: Ingestão de 1000 registros ➡ 990 registros válidos ➡ 804 chamados finalizados para análise de SLA.
- **Auditoria**: Validação de 100% das regras de prioridade e integridade cronológica (Resolução > Criação).

🚀 Como Executar

1. Clone o repositório.
2. Instale as dependências: `pip install -r requirements.txt`.
3. Configure o arquivo `.env` na raiz do projeto com suas credenciais do Azure.
4. Execute o orquestrador: `python main.py`.

---

us English Version

📋 Objective

This project automates the ingestion and processing of JIRA tickets to calculate resolution time in **business hours**, excluding weekends and national holidays.

🏗 Pipeline Architecture

The project follows the **Medallion Architecture**, ensuring organization and traceability:

- **Bronze Layer**: Raw ingestion of JSON files from Azure Blob Storage using Service Principal authentication.
- **Silver Layer**: Data cleaning and normalization. Extraction of nested fields (`assignee`, `timestamps`) and conversion to **Parquet** format for performance and metadata preservation.
- **Gold Layer**: Application of business rules. SLA calculation based on priority (High: 24h, Medium: 72h, Low: 120h), integrated with **BrasilAPI** for holiday identification.
- **Data Quality**: Automated auditing of integrity, volume, and null values at the end of the pipeline.

📊 Execution Evidence & Quality

The pipeline includes a central orchestrator that validates each stage. During test runs, the system successfully processed:

- **Data Funnel**: 1000 raw records ➡ 990 valid records ➡ 804 finalized tickets for SLA analysis.
- **Auditing**: 100% validation of priority rules and chronological integrity (Resolution > Creation).

🗁 Estrutura de Pastas / Project Structure

```
project-root/
├── main.py                # Orquestrador Central / Main Orchestrator
├── .env                   # Credenciais (Não versionado) / Credentials (Not
versioned)
├── src/
│   ├── bronze/            # Ingestão / Ingestion
│   ├── silver/            # Transformação / Transformation
│   ├── gold/              # Regras de Negócio / Business Rules
│   ├── sla_calculation.py  # Motor de Cálculo / Calculation Engine
│   └── validate_pipeline.py # Auditoria de Dados / Data Auditing
└── data/                  # Armazenamento Local / Local Storage (Git Ignored)
```