Imaflora - Instituto de Manejo e Certificação Florestal e Agrícola

Relatório: Processo ETL para Dados de desmatamento do projeto PRODES

Edson Matias dos Santos

Piracicaba, 2025

Sumário

1 Introdução

Este relatório descreve a solução desenvolvida para o desafio de Engenharia de Dados, que consiste em um processo ETL (Extração, Transformação e Carregamento) para dados de desmatamento do bioma Cerrado, disponibilizados pelo TerraBrasilis via GeoServer. Os dados são parte do projeto PRODES, que monitora anualmente o desmatamento na Amazônia e no Cerrado brasileiro. A solução visa garantir a ingestão eficiente, a correção de geometrias geoespaciais e o armazenamento otimizado em um banco PostgreSQL/PostGIS, permitindo consultas espaciais otimizadas.

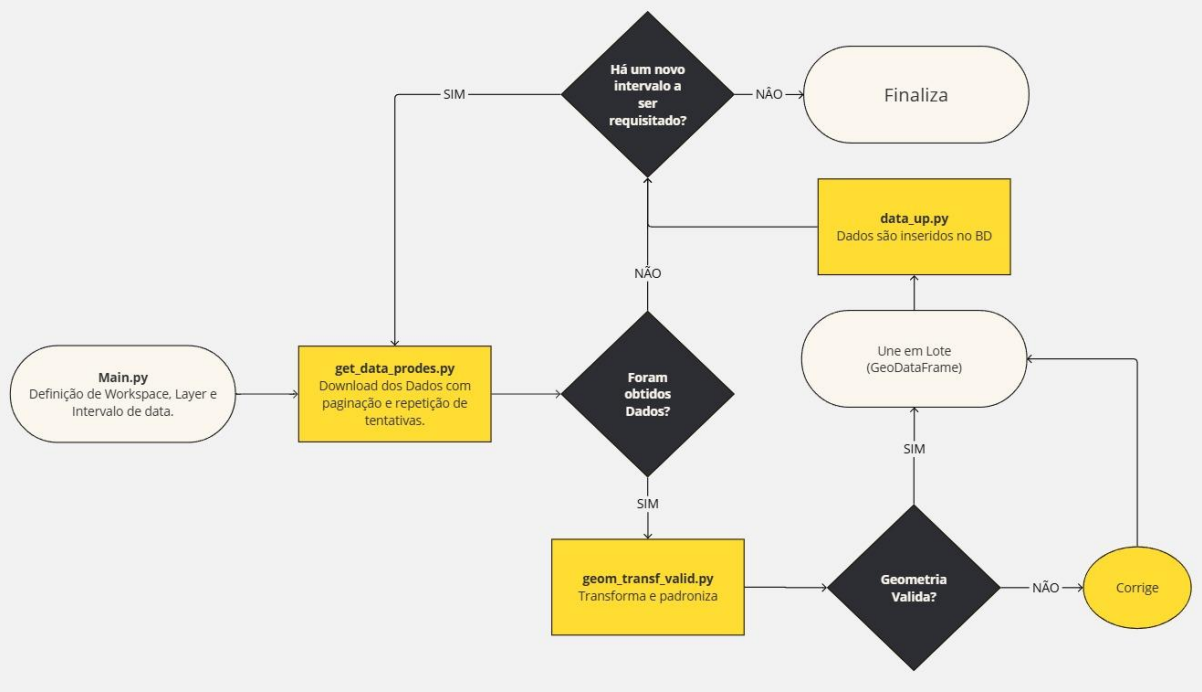
2 Objetivos

- Extrair dados do GeoServer do TerraBrasilis com resiliência a falhas;
- Processar e validar geometrias geoespaciais;
- Armazenar os dados em um banco PostgreSQL com índices e estrutura otimizados;
- Garantir reprodutibilidade e escalabilidade do processo.

3 Metodologia

De maneira geral e resumida, o funcionamento do ETL é explicado no fluxograma abaixo:

Figura 1 – Fluxograma com a representação das interações dos algoritmos



3.1 Download de Dados (get_data_prodes.py)

Abordagem Técnica:

- Integração com WFS: A URL do GeoServer é construída dinamicamente com base no workspace e camada (ex: prodes-cerrado-nb/yearly_deforestation), seguindo o padrão OGC WFS;
- Filtragem por Intervalo de Anos: Uso de CQL (year BETWEEN start AND end) para limitar os dados ao período desejado, reduzindo o volume transferido;
- Paginação Inteligente: Cada requisição retorna até 10.000 registros (page_size). O loop incrementa startIndex até que todas as páginas sejam baixadas.

Resiliência das requisições:

- Tentativas: 10 tentativas com intervalo de 60 segundos para cada requisição falha, podendo ser modificado de acordo com a necessidade;
- Controle de Exceções: Interrupção imediata em caso de falha persistente, evitando loops infinitos.

Otimizações:

- Streaming de Dados: Os dados são acumulados em memória de forma incremental, evitando sobrecarga;
- Compressão de Resposta: O GeoServer retorna dados em JSON de forma a facilitar o tratamento com diferentes bibliotecas, já que atualmente é um formato de dado muito popular.

3.2 Processamento de Dados (geom_transf_valid.py)

Padronização de Colunas:

- Remoção de campos redundantes (ex: main_class);
- Preenchimento de campos ausentes com valores padrão (ex: source = 'cerrado');
- Conversão de tipos (ex: datas para DATE, geometrias para shapely.geometry).

Correção de Geometrias:

- make_valid: Corrige geometrias inválidas (ex: polígonos auto-intersectantes) usando a biblioteca shapely;
- Validação Final: Descarte de geometrias irreparáveis para garantir integridade;
- Consistência de CRS: Todas as geometrias são convertidas para EPSG:4326, padrão muito usada dentro da comunidade que usa dados geoespaciais.

Melhores Práticas:

- Separação de Responsabilidades: Funções dedicadas para conversão (convert_feature_to_gdf) e validação (validate_and_fix_geometry);
- Eficiência em Memória: Processamento por feature, evitando carregar todo o dataset de uma vez.

3. Armazenamento no PostgreSQL (data_up.py e docker-compose.yml)

3.3 Criação do Banco de Dados

Configuração via Docker:

- Uso da imagem postgis/postgis para garantir suporte a geometrias;
- Criação facilitada garantindo portabilidade e a replicação do ambiente desenvolvido além da redução de conflitos com dependências isolando a solução;
- Volume persistente (postgres_data) para evitar perda de dados após reinicializações.

Esquema e Índices:

- Tabela: Foi criado um esquema raw_data e a tabela desmatamento para armazenamento dos dados;
- Integridade: A tabela inclui campos NOT NULL como year, state, e geom, para garantir que esses dados estejam presentes em qualquer inserção;
- Indices: Alguns índices foram criados, para acelerar consultas, em determinados campos que categorizam as informações, além de Índices GIST que aceleram consultas espaciais (ex: ST_Within, ST_Distance);
- O Id da feature foi reaproveitado como chave primária da tabela.

3.3.1 Mecanismo de Ingestão

- Conversão para WKT: Geometrias são convertidas para texto antes do upload, compatível com PostGIS;
- Inserção em Massa: Uso SQLAlchemy para eficiência e com condição de não inserir dados duplicados.

3.3.2 Principais vantagens desta implementação:

- Escalabilidade: A estrutura suporta até milhões de registros sem degradação de performance;
- Consulta Otimizada: Índices reduzem o tempo de busca para filtros comuns.

3.4 Execução: Script Principal (main.py)

Orquestração:

- Divisão em Lotes: Processamento em intervalos de 2 anos (ex: 2000-2001, 2002-2003) para equilibrar carga de memória e garantir mais rastreabilidade nas inserções.

Integração de Módulos:

- Download: Chamada da função get_paginated_geojson para download de acordo com as entradas (filtros) definidos;
- Transformação: É realizado a transformação e validação de forma a deixar o dado pronto para ingestão;
- Carga: Upload único por lote para minimizar transações no banco.

4 Principais dificuldades e desafios encontrados

4.1 Escassez de informações sobre o Dataset

A limitada disponibilidade de informações na internet dificultou a estruturação das requisições necessárias para acessar os dados. Foi necessário investir um tempo considerável em pesquisas, testes e validações para estabelecer um padrão robusto de extração, garantindo que a construção dinâmica da URL e a implementação da paginação fossem realizadas de maneira eficaz.

4.2 Falta de documentação detalhada dos dados

A carência de informações precisas sobre o conteúdo e a estrutura dos dados requisitados impôs desafios adicionais para a criação dos processos de transformação. Essa limitação exigiu a implementação de contornos específicos para tratar e validar os dados, resultando em transformações que visam reduzir redundâncias e garantir um armazenamento otimizado, mesmo com informações incompletas.

4.3 Inconsistências e ausência de padronização

A variabilidade dos dados, com falta de um padrão consistente, demandou uma análise mais profunda para definir algumas regras de conversão e correção. Isso envolveu explorar o dataset para identificar os ajustes necessários, como a padronização de colunas e a correção de geometrias, de maneira a minimizar erros e assegurar a integridade das informações.

4.4 Volume elevado de dados

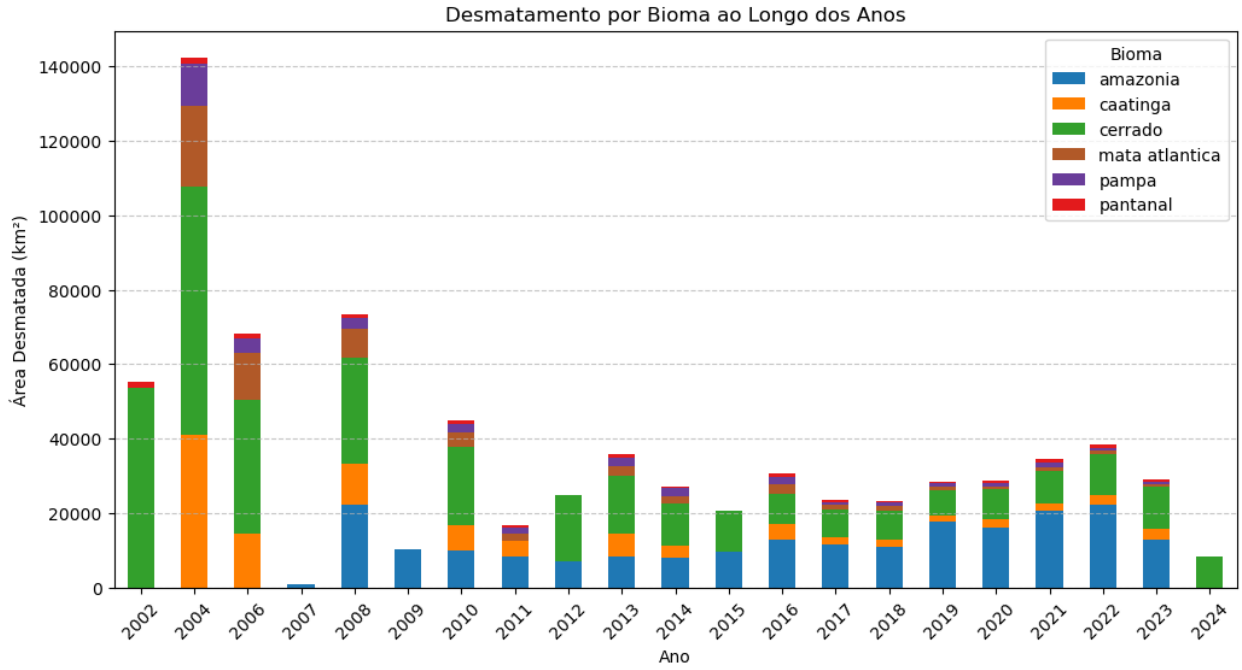
O processamento de uma grande quantidade de registros impôs a necessidade de um equilíbrio constante entre performance e a quantidade de dados processados. Para mitigar esse desafio, foram implementadas estratégias como a paginação inteligente e o processamento em lotes, permitindo a manipulação eficiente dos dados sem comprometer a performance do sistema.

5 Exploração dos Dados

Foi feito alguns exemplos rápidos de exploração desses dados que estão no notebook Data_Exploration.ipynb, como:

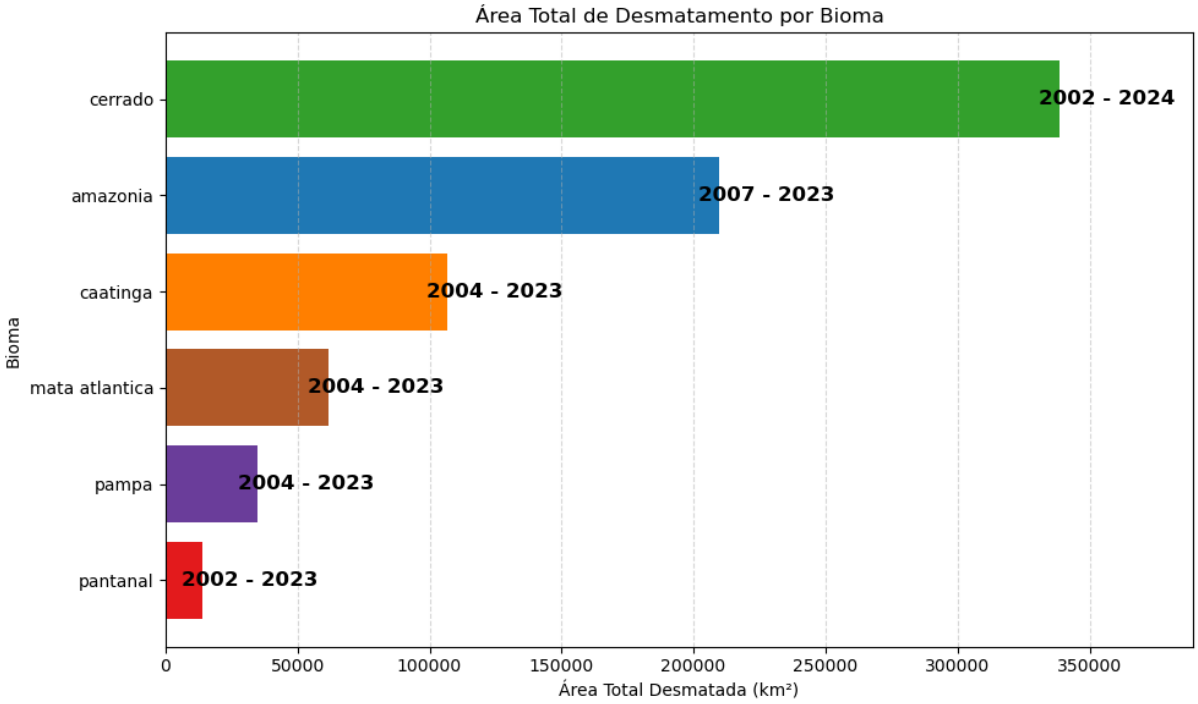
5.1 Desmatamento por bioma em cada ano

Figura 2 – Gráfico de desmatamento por ano.



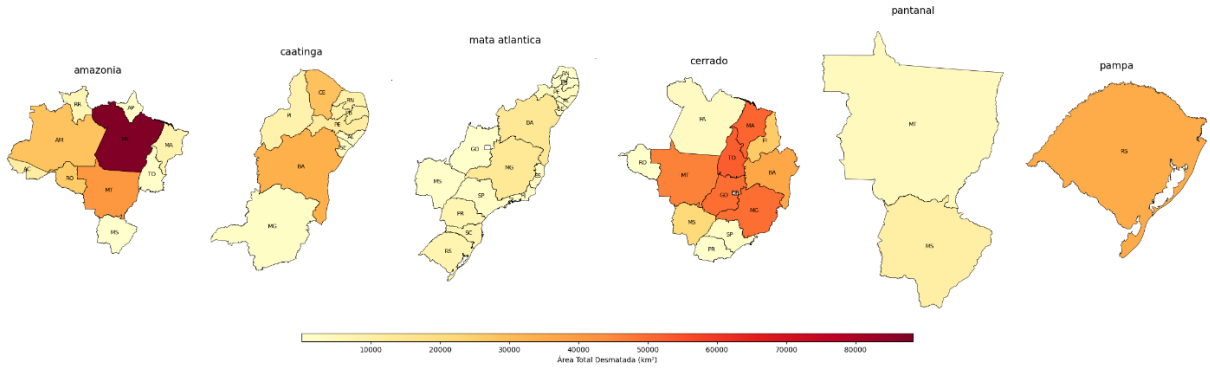
5.2 Desmatamento total por bioma

Figura 3 – Gráfico de desmatamento por bioma.



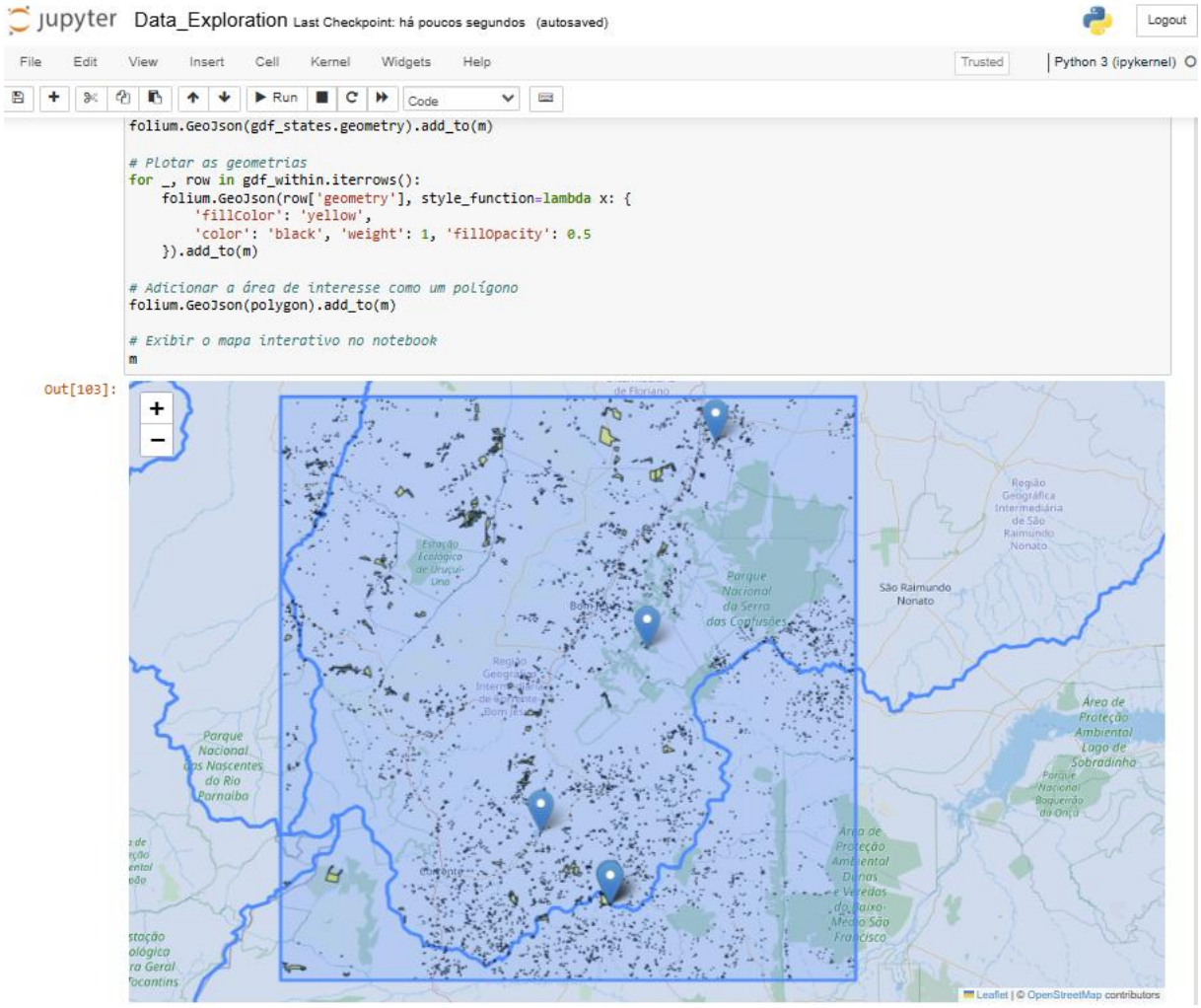
5.3 Desmatamento por estado

Figura 4 – Mapa de desmatamento por estado



5.4 Consulta por geometria e mapa interativo

Figura 5 – Mapa interativo criado a partir de uma consulta com geometria.



6 Conclusões

- Mecanismos de paginação e tentativas garantem robustez contra instabilidades de conexão;
- Geometrias inválidas são corrigidas ou descartadas, assegurando qualidade dos dados;
- Processamento em lotes minimiza uso de memória.

Sugestões de Melhoria:

- Implementar logs detalhado para diagnóstico de erros nas execuções;
- Adicionar testes unitários para validação de geometrias;
- Paralelizar o download e processamento de dados para ganhar ainda mais performance;
- Maior generalização dos algoritmos para suportarem mais features disponibilizadas no dataset.
- Estudar mais a fundo a base de dados, para identificar dados irrelevantes, correções a serem feitas de forma a otimizar ainda mais o armazenamento e garantir a qualidade dos dados.
- O tipo de arquitetura medalhão (Bronze, Silver e Gold), como forma de auxílio a organização num processo de ETL para esse tipo de cenário, pode ser interessante de forma a explorar seu refinamento de acordo com as finalidades de uso mitigando um pipeline de Data Quality.
- Com o Algoritmo generalizado, criar pipeline de ETL completo em soluções de orquestração como Apache Airflow para se obter uma extração recorrente e padronizada dos dados.