

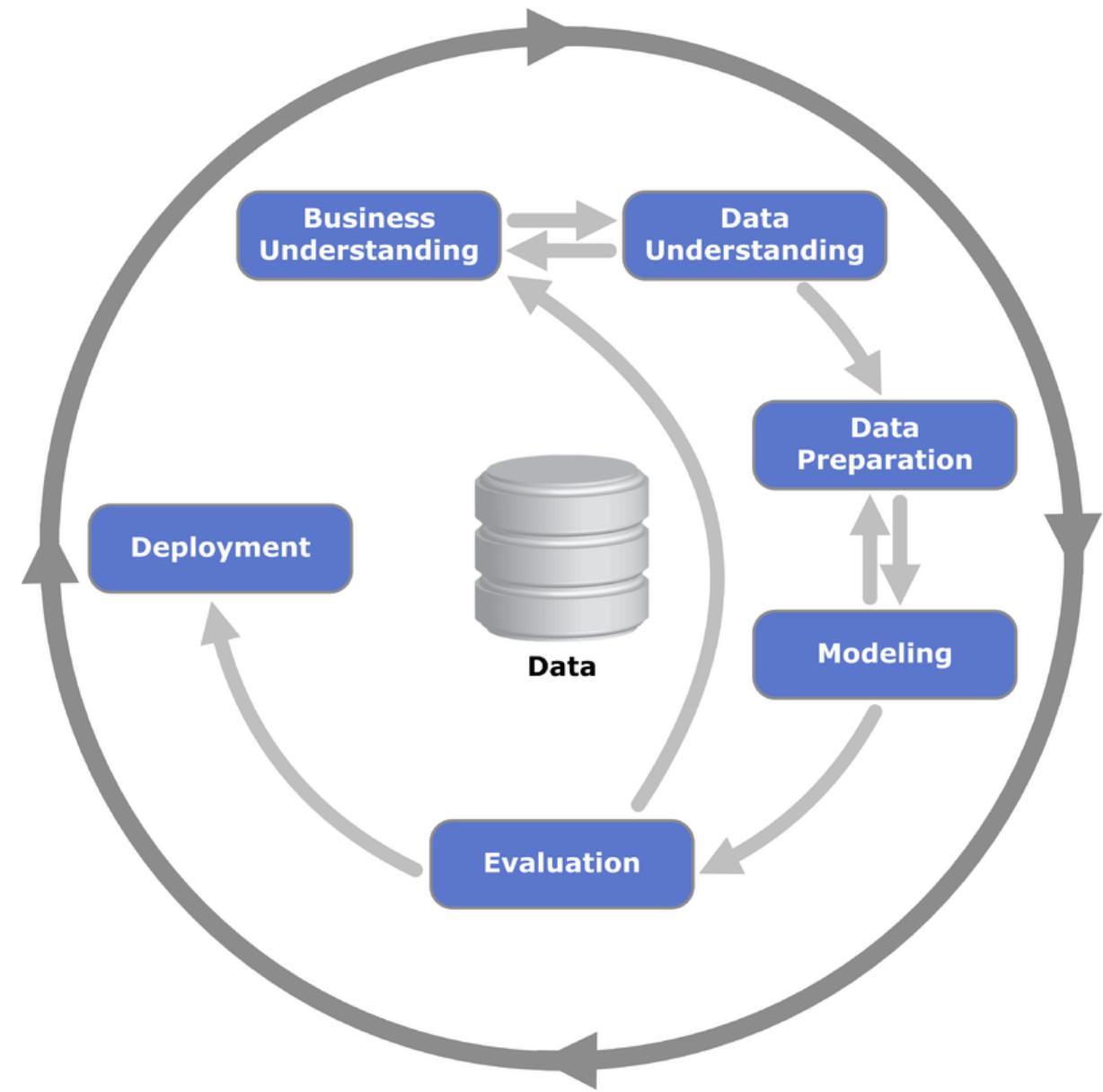
Engenharia de Machine Learning

2025-03-12

Modelo CRISP-DM

Cross Industry Standard Process for Data Mining (1996)

- Entendimento de negócio
- Entendimento do dado
- Preparação dos dados
- Modelagem
- Avaliação
- Deployment

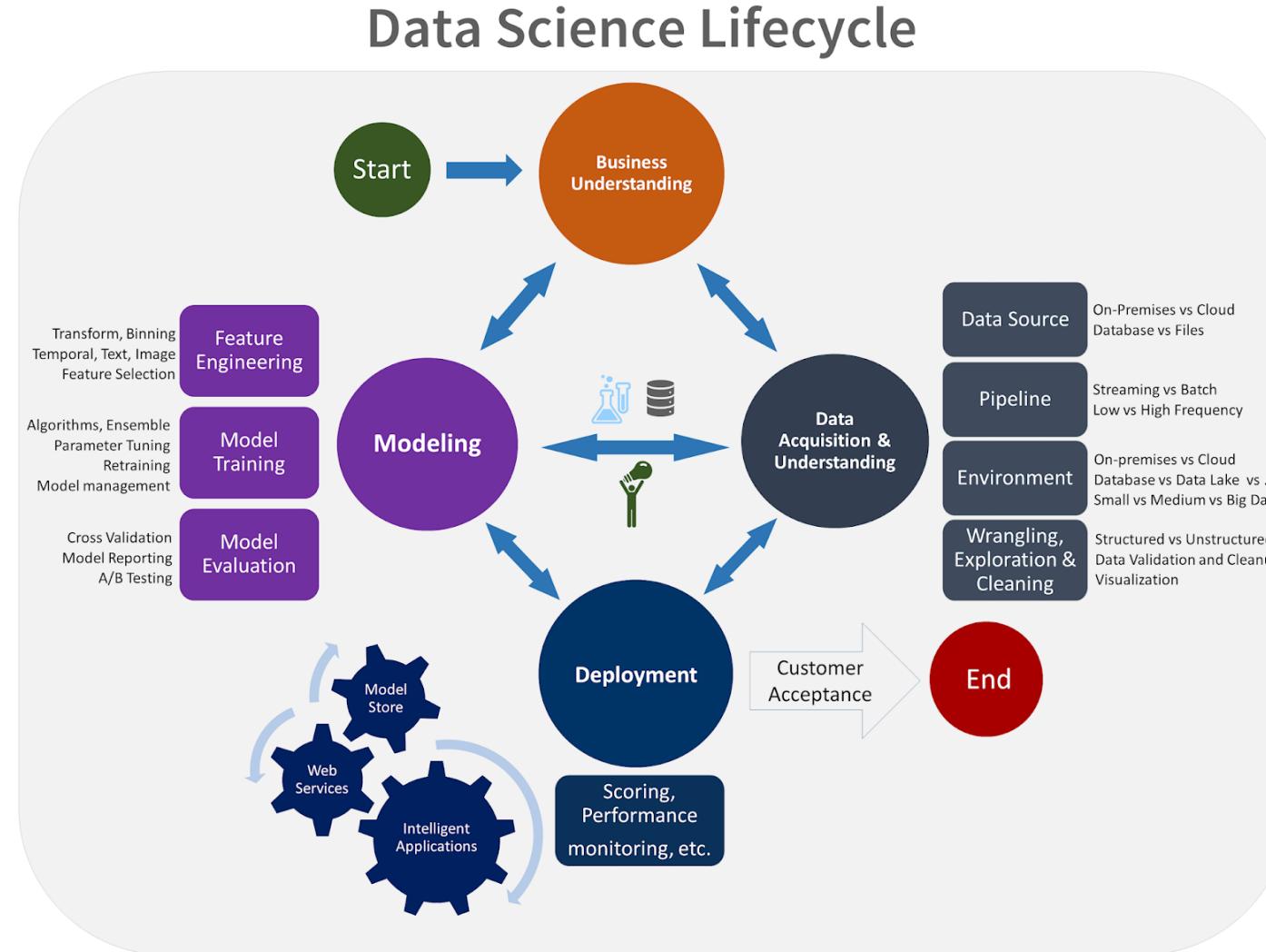


Team Data Science Process (Microsoft)

Metodologia ágil e integrada para o desenvolvimento de projetos de data science em equipe. Define um ciclo de vida do projeto de maneira muito semelhante a outros processos como o CRISP-DM, KDD, SEMMA e ASUM-DM (IBM).

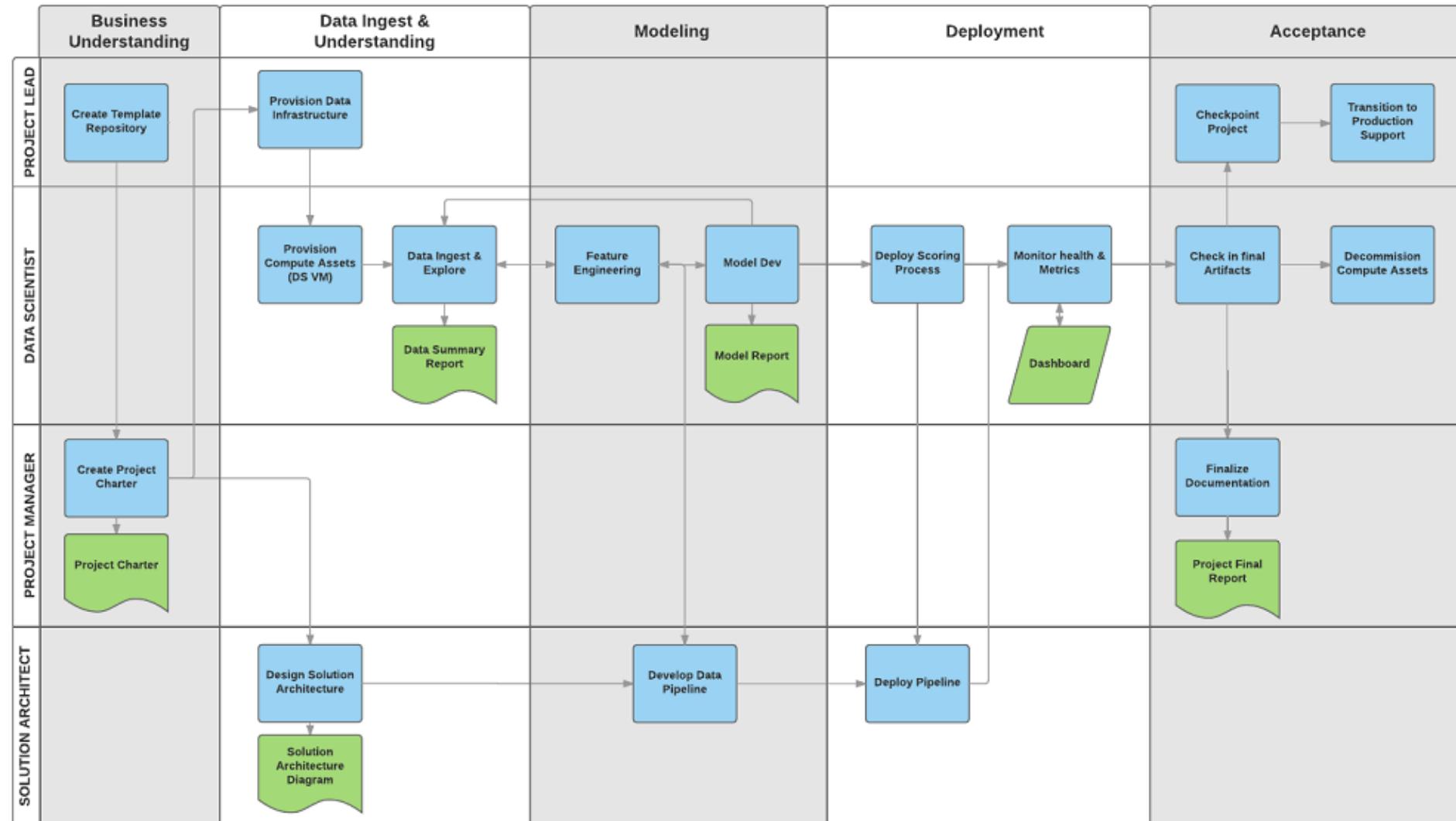
- Definição do ciclo de vida de data science
- Estrutura padrão de projetos
- Infraestrutura e recursos para projetos de data science
- Ferramentas e outras utilidades para a execução do projeto
- Processos para melhorias contínuas (dev/ops, ML/ops)

Ciclo TDSP

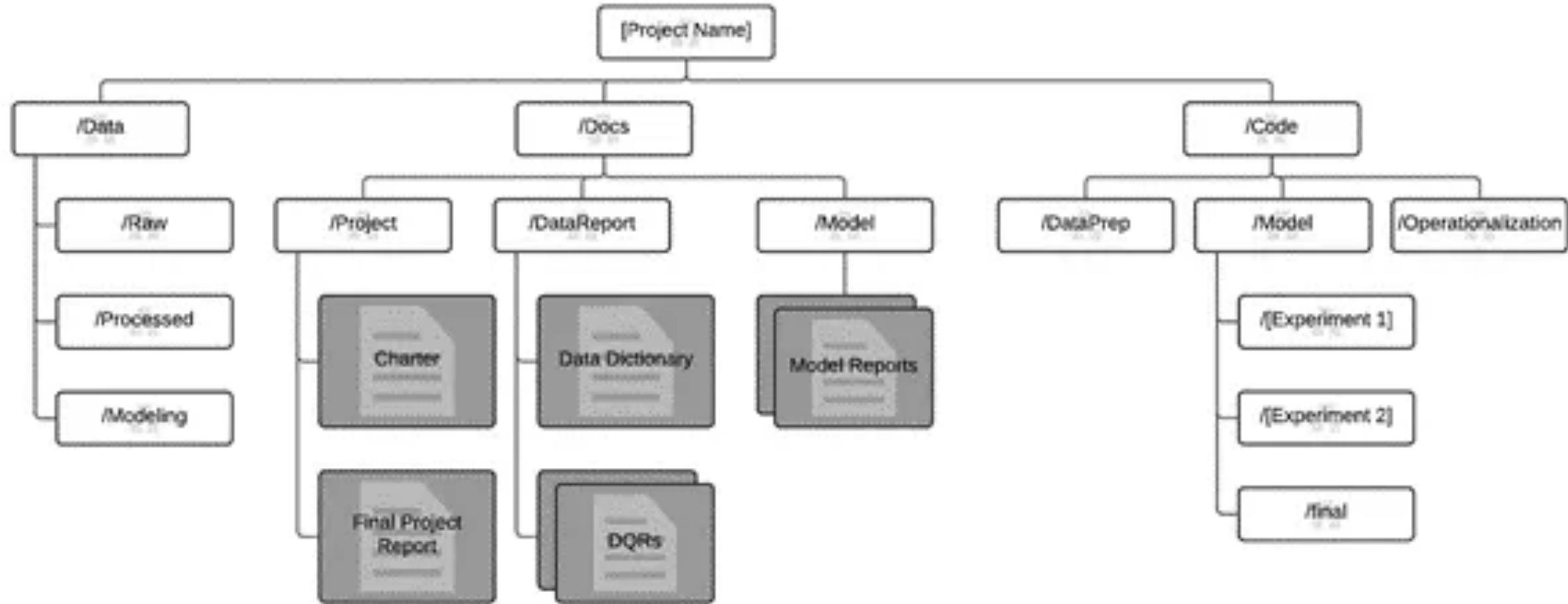


Fonte: Microsoft

Artefatos e Tarefas TDSP



Estrutura de um projeto TDSP



Dois documentos principais: **Charter** e **Exit Report**

Ferramentas MLOps que Usaremos

Kedro

- Estruturação do código de análise
- Foco em padronização e reproduzibilidade

MLFlow

- Rastreamento de experimentos
- Rastreamento de métricas, plots e logs
- Registro e versionamento de modelos
- Servir modelos

Por que Kedro?

Padronização da Forma de Programar

- Uma série de decisões de organização e execução pré-tomadas
- Diminui flexibilidade¹
- Diminui a entropia do código

Promove Modularização

- Pensamento em conjuntos de dados e transformações entre eles
- Maior facilidade de reaproveitar código entre diferentes análises

Promove boas práticas de Eng. de Software

- Testes Automáticos
- Ferramentas de deploy e execução automáticas

Código pronto para produção

- A mesma estrutura de códigos usados para testes locais é a que será usada na execução em larga escala do processamento
- Integra com os principais orquestradores

¹ Nesse caso, a diminuição de flexibilidade é uma vantagem.

Estrutura de um projeto Kedro

Todo projeto Kedro deve possuir:

- Arquivo README.md com descrição do projeto
- Código de produção ([src/](#))
- Código de análises *ad-hoc* ([notebooks/](#))

Opcionalmente, pode possuir:

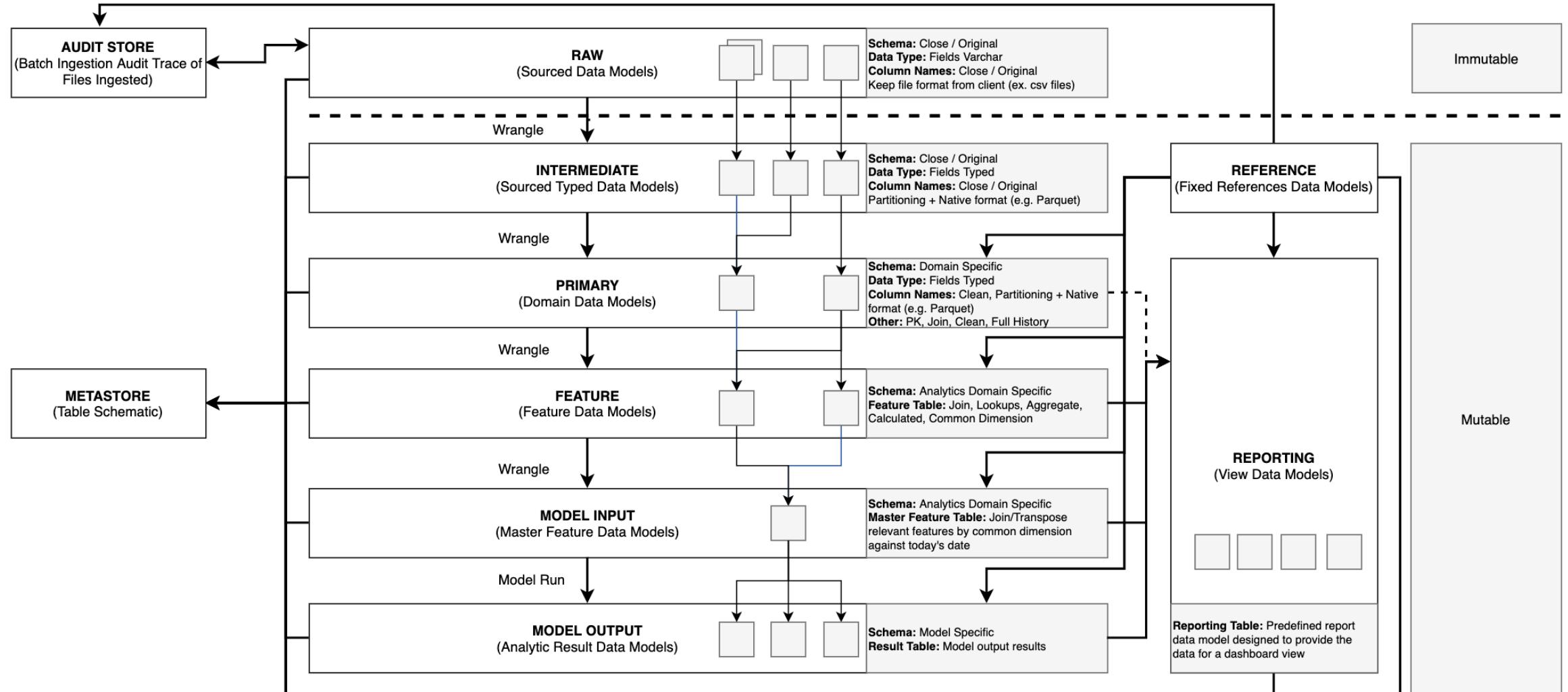
- Documentação ([docs/](#))
- Dados locais ([data/](#))
- Testes automáticos ([tests/](#))

Exemplo de estrutura:

```
project-dir
├── conf
├── data
├── docs
├── notebooks
├── src
├── tests
├── .gitignore
├── pyproject.toml
└── README.md
└── requirements.txt
```

Fonte: [Kedro Concepts](#)

Kedro: Layered Thinking in Data Engineering



Fonte: Joel Schwarzmann. [The importance of Layered Thinking in Data Engineering](#)

Principais conceitos do Kedro

Node

- Abstração de uma função pura do Python
- Recebe entradas e gera saídas

Pipeline

- Define a ordem na qual os nós devem ser executados, e a dependência entre eles

Catálogo de Dados

- Registro de todos as fontes e arquivos para leitura e escrita de dados
- Promove a separação de responsabilidade entre códigos de análise e manipulação de arquivos
- Promove o pensamento como camadas de dados, que é uma categorização da função que um arquivo de dados desempenha na análise

Formas de Aquisição de Dados

Arquivos de Dados ou Bancos de Dados

Forma mais simples, arquivos já disponíveis em local acessível.

Application Programming Interfaces (APIs)

Serviços disponibilizados por terceiros que pode servir como fontes de dados. O mais comum são APIs estilo REST, respondendo em formato JSON. Podem ser necessárias algumas etapas:

- Autenticação (possivelmente condicionada a contratação de um plano)
- Diversos padrões de Comunicação

Web Scraping

Extrair dados publicamente acessíveis na Web, mas que não foram inicialmente disponibilizado para acesso programático.

Bibliotecas Úteis:

- Requests
- BeautifulSoup
- Newpaper3k
- Scrapy

Exemplo: Dados Abertos da Câmara

- API RESTful
- Bem documentado em [OpenAPI](#)
- Pode dar resultados em XML ou JSON

Para recuperar a lista de deputados, basta executar uma chamada GET em

<https://dadosabertos.camara.leg.br/api/v2/deputados>

Exemplo de código

```
1 import requests
2 import pandas as pd
3
4 resp = requests.get("https://dadosabertos.camara.leg.br/api/v2/deputados")
5 data = resp.json()
6
7 df_data = pd.DataFrame(data["dados"])
```

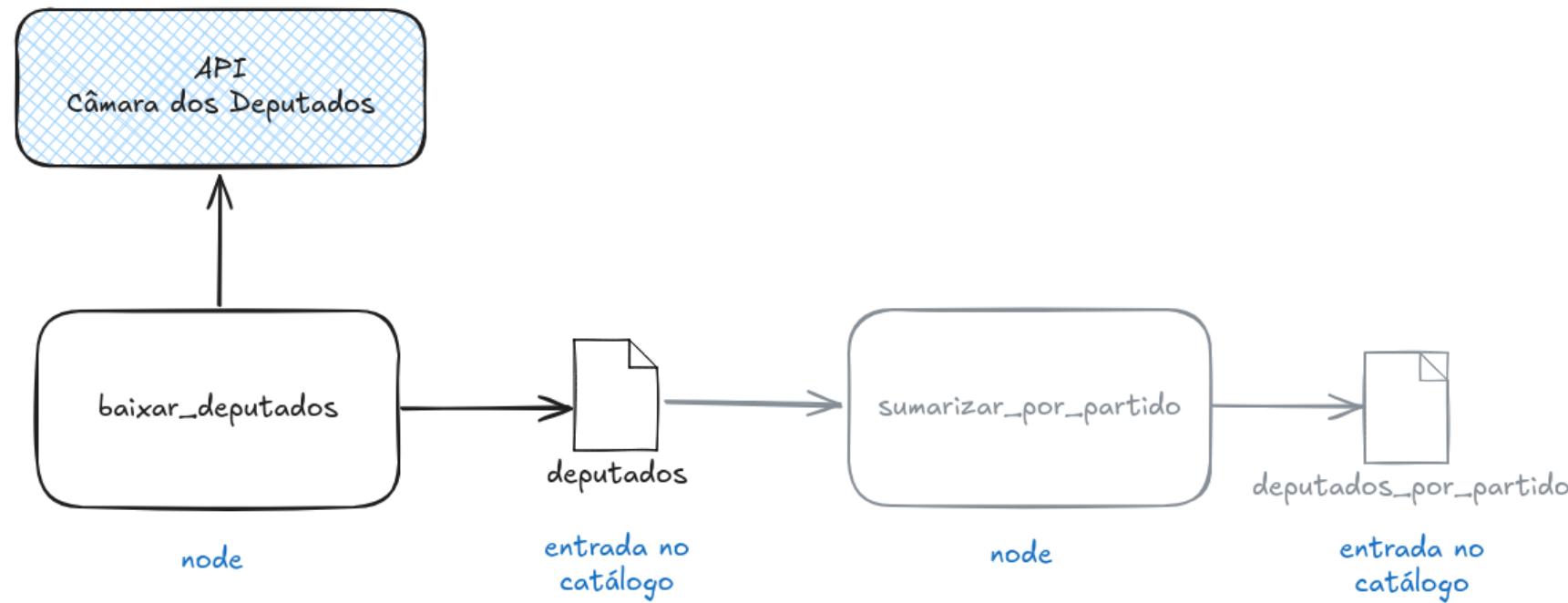
Voltamos 21:18

Exemplo: Exploração sobre Deputados

Objetivo:

- Baixar listagem de deputados
- Sumarizar quantidade de deputados por partido

Diagrama do pipeline:



Primeiros passos

Instalar o Kedro:

```
1 pip install kedro
```

Criar um novo projeto Kedro:

```
1 kedro new
```

- Nome do projeto: [deputados](#)
- Ferramentas: escolher [4, 5, 7](#) para ter docs, dados e [kedro-viz](#)
- Opte por não criar um pipeline de exemplo agora

Editar o arquivo [requirements.txt](#), se necessário e instalar as demais dependências:

```
1 pip install -r requirements.txt
```

Criação do pipeline:

```
1 cd deputados  
2 kedro pipeline create exploracao_deputados
```

Veja que agora há diretórios para o pipeline criado:

```
1 $ tree src  
2 src  
3   └── deputados  
4     ├── __init__.py  
5     ├── __main__.py  
6     ├── pipeline_registry.py  
7     └── pipelines  
8       ├── exploracao_deputados  
9         ├── __init__.py  
10        ├── nodes.py  
11        └── pipeline.py  
12          └── __init__.py  
13   └── settings.py
```

Implementação

Live Coding:

- Testes no Jupyter Notebook
- Configuração do Catálogo de Dados
- Criação dos nós
- Criação do pipeline