de-RSE 2025

MLentory

A Machine Learning model registry with natural language queries

Nelson Quiñones
Dietrich Rebholz-Schuhmann
Leyla Jael Castro

de RSE CONFERENCE KARLSRUHE 2025

DFG

de NBI
GERMAN NETWORK FOR BIOINFORMATICS INFRASTRUCTURE

NFDI4 DataScience

ZB MED

Lost in a *sea* of ML models and resources?

Table of content

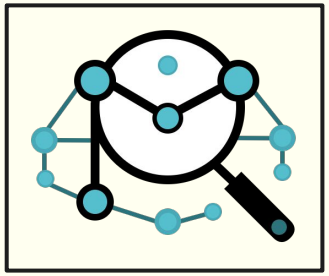
Introducing MLentory

- MLentory is one of the projects of NFDI4DS.
- MLentory **gathers** ML model information from diverse platforms, **harmonizes** this data into a common and standardized format and **shares** this information in a FAIR Digital Object (FDO) registry.

Skyscanner

Google

~

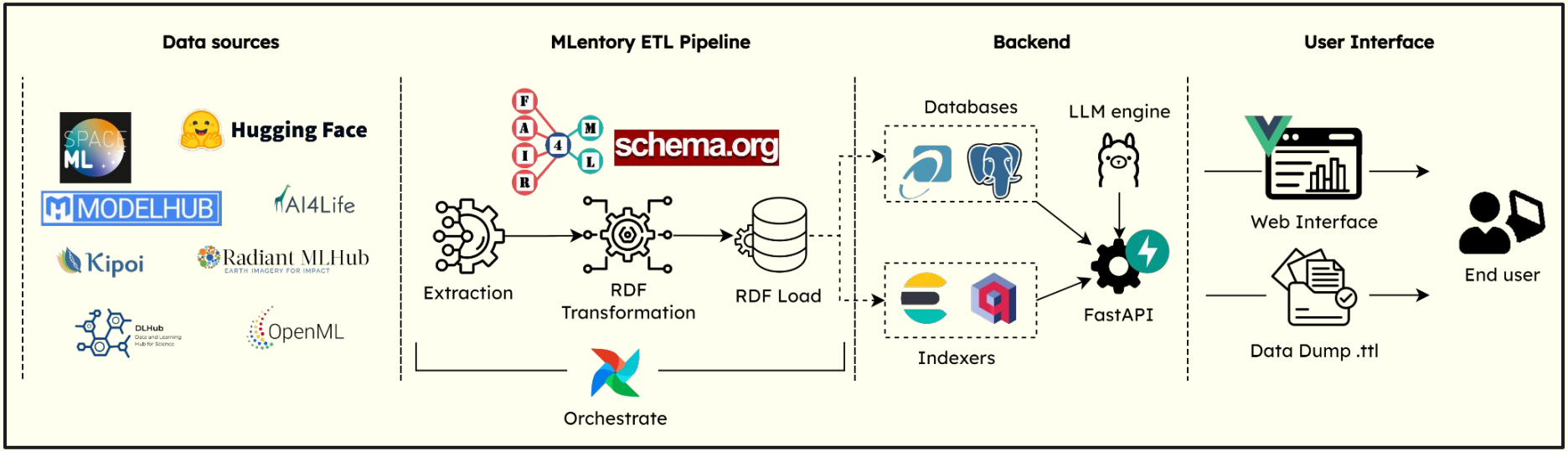


What are we doing differently?

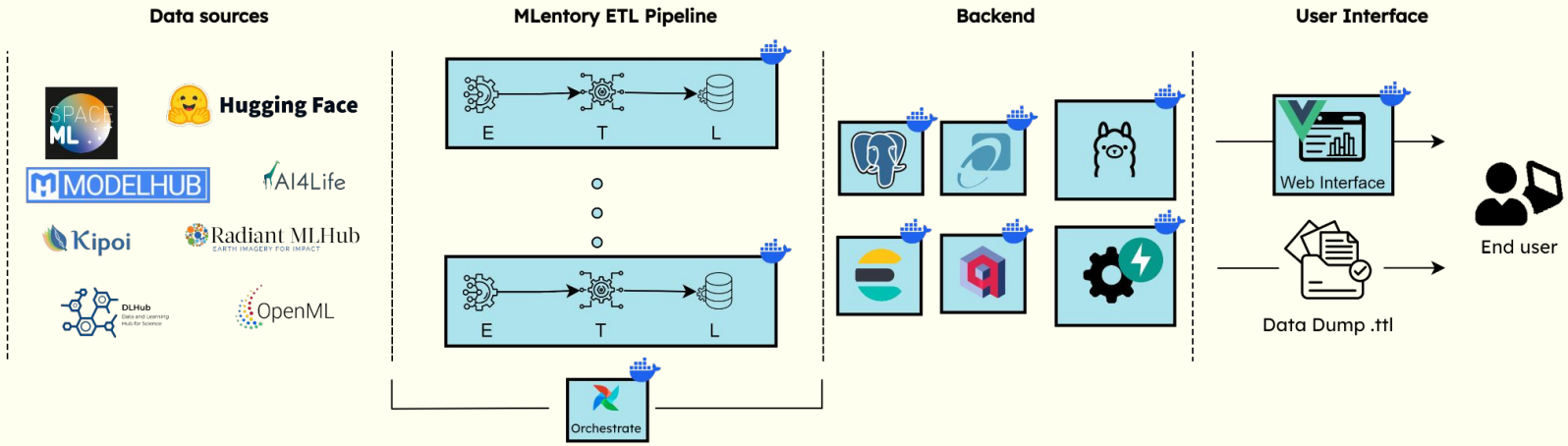
- Collecting resources from different sources.
- Curating a schema for ML models called FAIR4ML.
- Enabling natural text queries to search for resources.
- Keeping track of the history of the metadata.

How does MLentory work?

Overview


Data sources
MLentory ETL Pipeline
Backend
User Interface
Hugging Face
MODELHUB
AI4Life
Kipoi
Radiant MLHub
DLHub
OpenML
schema.org
Extraction
RDF Transformation
RDF Load
Orchestrate
Databases
LLM engine
FastAPI
Indexers
Web Interface
Data Dump .ttl
End user

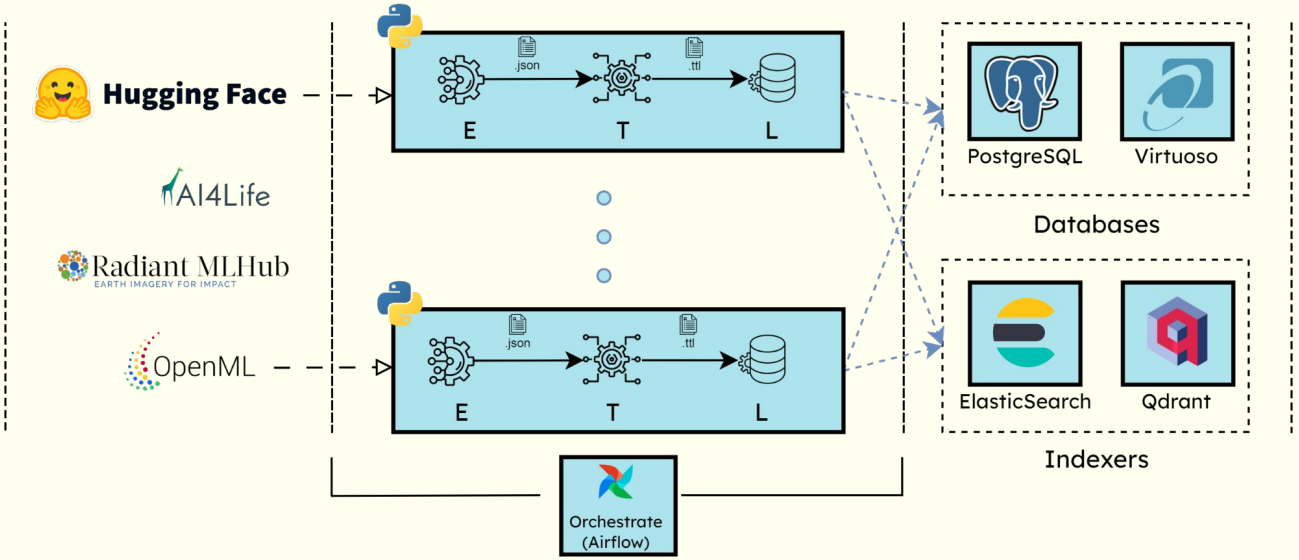
Overview - Docker



- Rapid Prototyping and Experimentation
- Reproducibility and Collaboration
- Simplified Deployment and Scalability

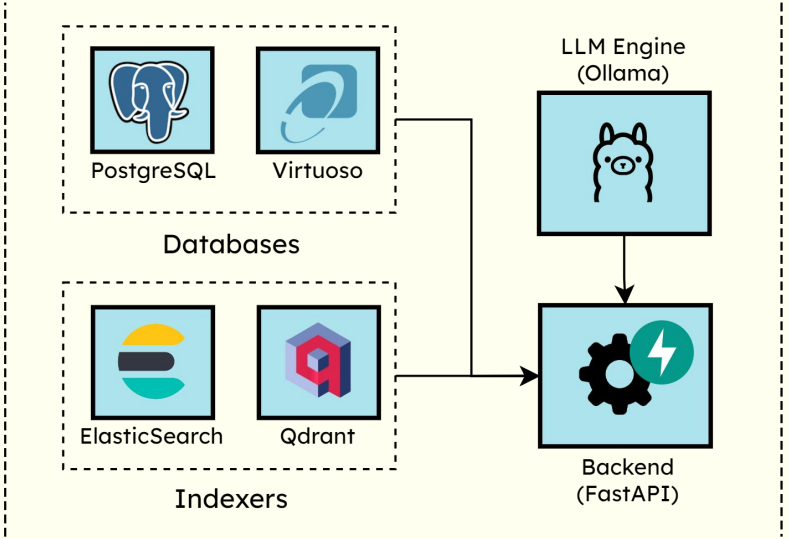
ETL Pipeline

- The **Extractor**, **Transform** and **Load** components are Python packages made for each platform.
- The **scheduler** is a Python script that uses Apache **Airflow** to trigger events.



Backend

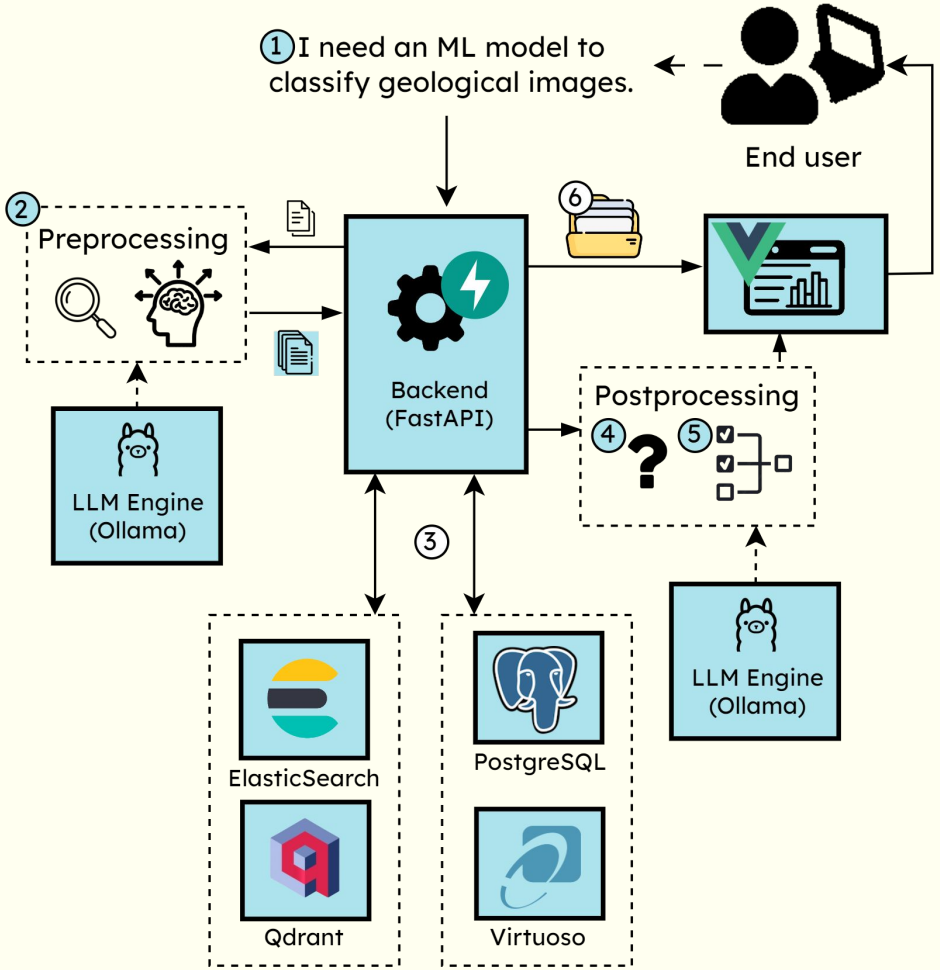
- A **Virtuoso** database to keep the newest version of the metadata in RDF format.
- A **PostgreSQL** database to keep the history of all the extracted metadata.
- An **Elasticsearch** and **Qdrant** components to index data from the Databases.
- An **Ollama** instance to empower natural language interactions with the MLentory resources.
- An API written with **FastAPI** to serve all MLentory's services.



LLMs: Searching

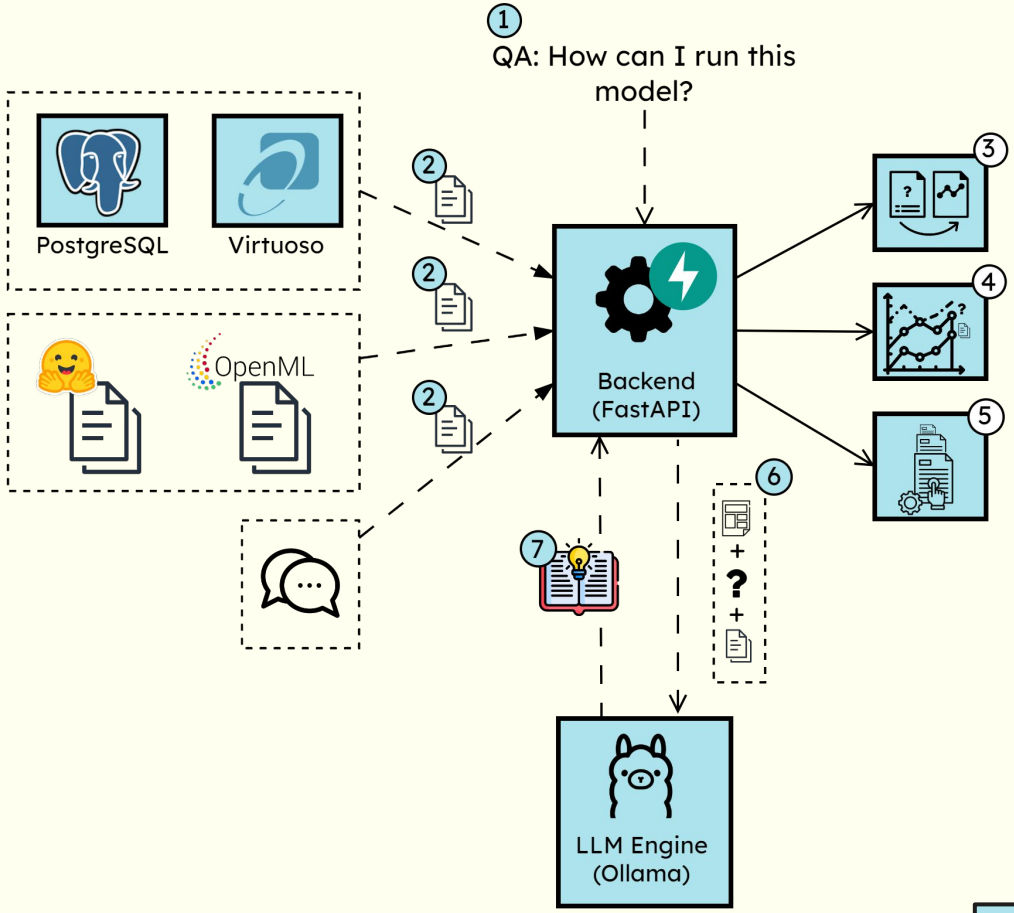
1. Receive a query.
2. Identify key elements and expand on them.
3. Use the new query to extract information.
4. Generate questions for the user.
5. Recommend filters to the user.
6. Serve collected data.

1 ...



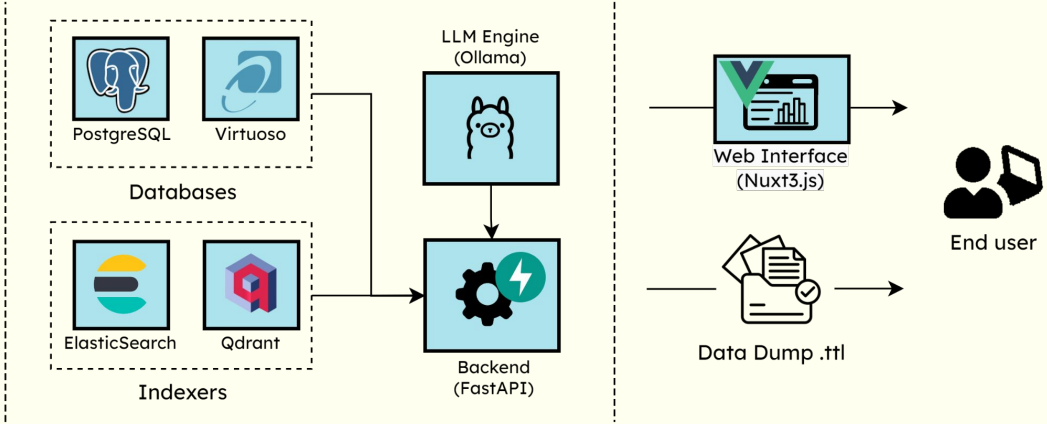
LLMs: Chatting

1. Receive a question.
2. Gather all possible relevant resources.
3. Generate embeddings.
4. Compare resources and target question.
5. Select a subset.
6. Compose the query.
7. Get a response



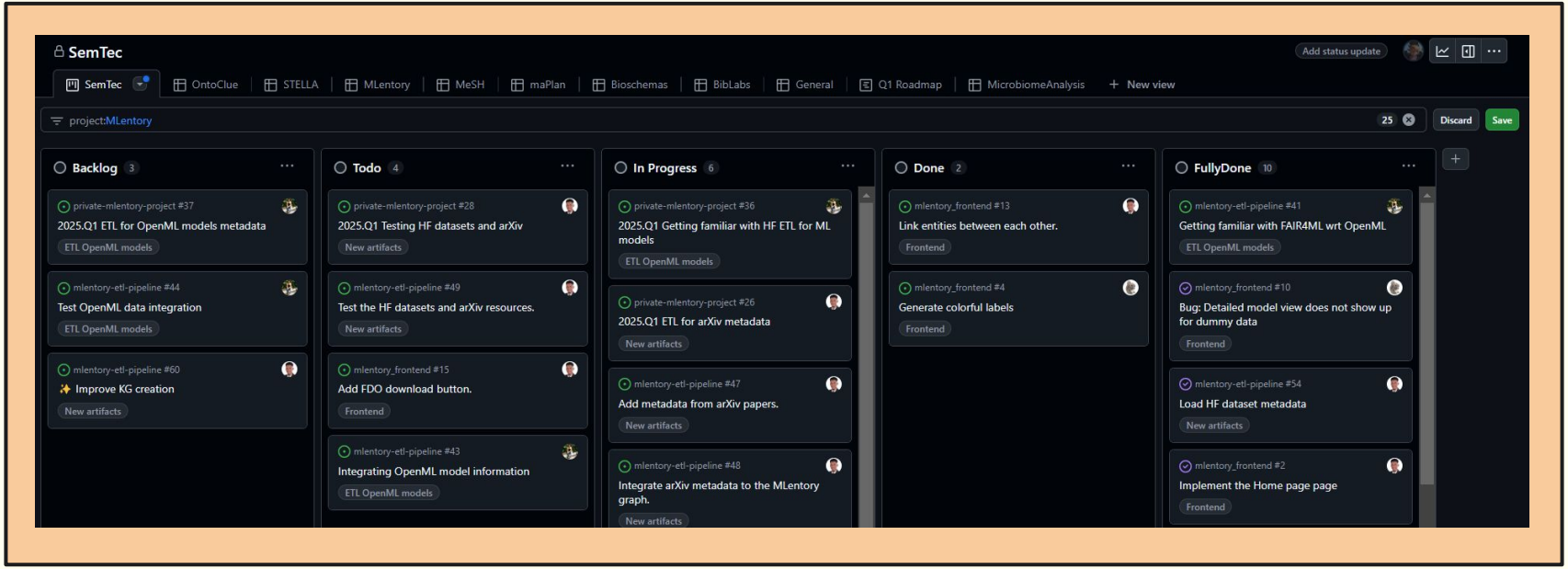
User interface

- A Web Interface built using **Nuxt3.js**
- An API endpoint serving the latest version of the graph in a .ttl file

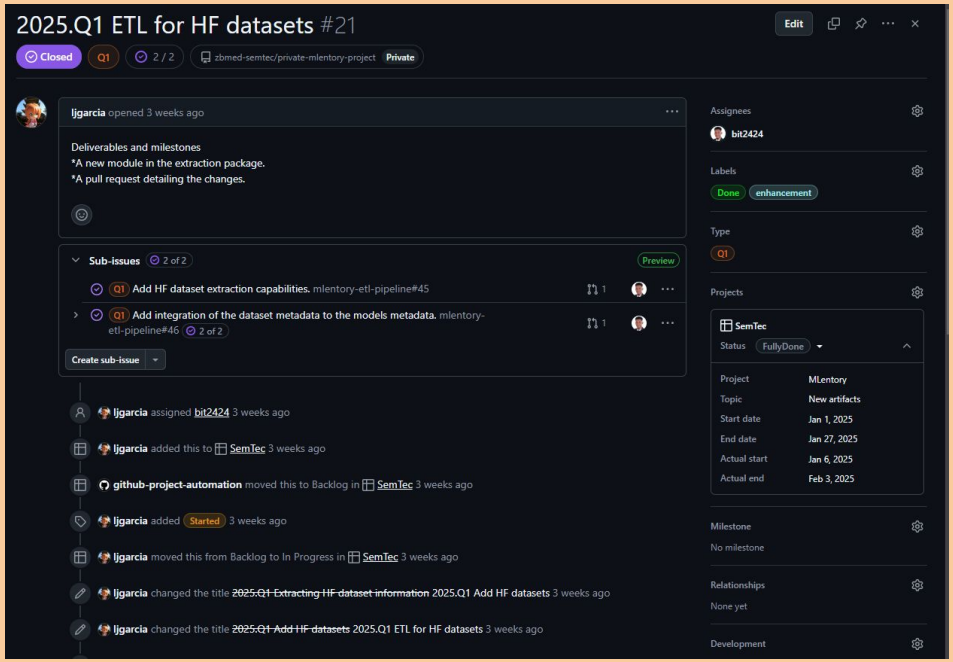


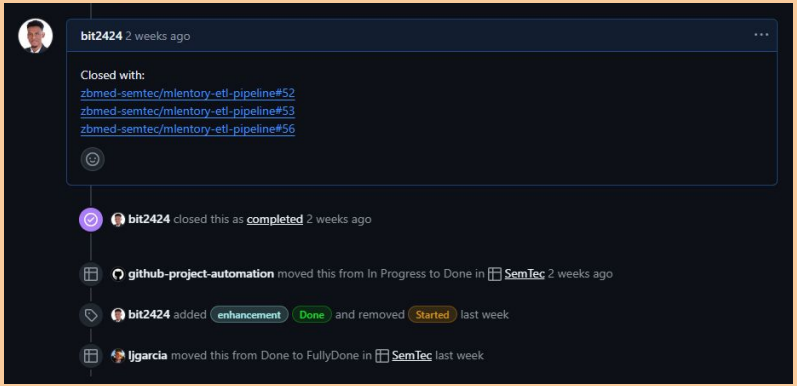
How are we building MLentory?

Development framework

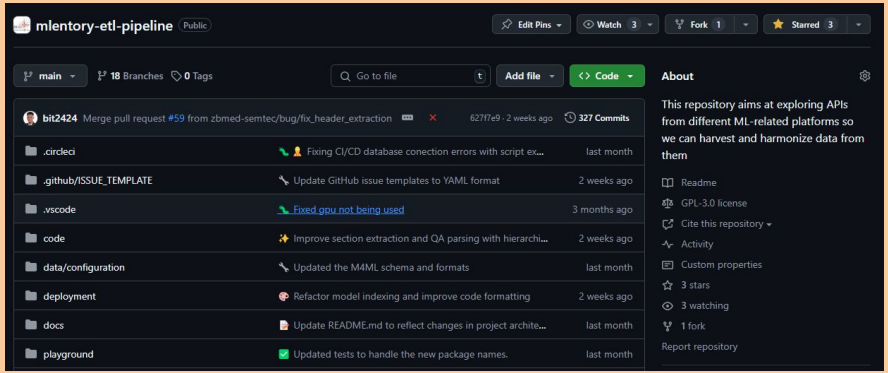


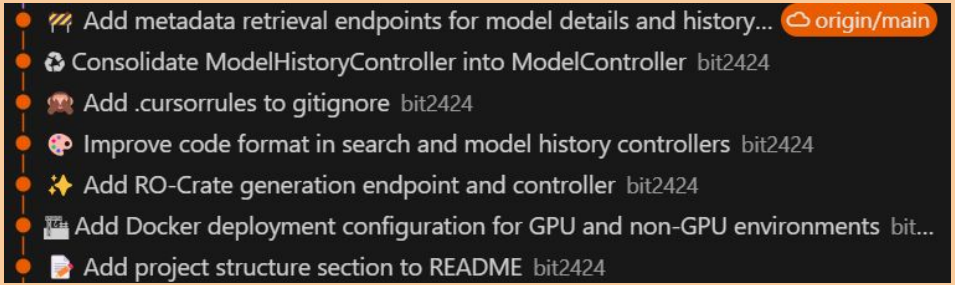
Development framework

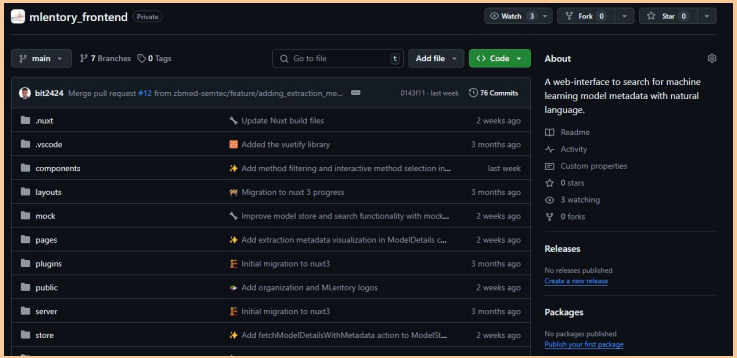




Version Control





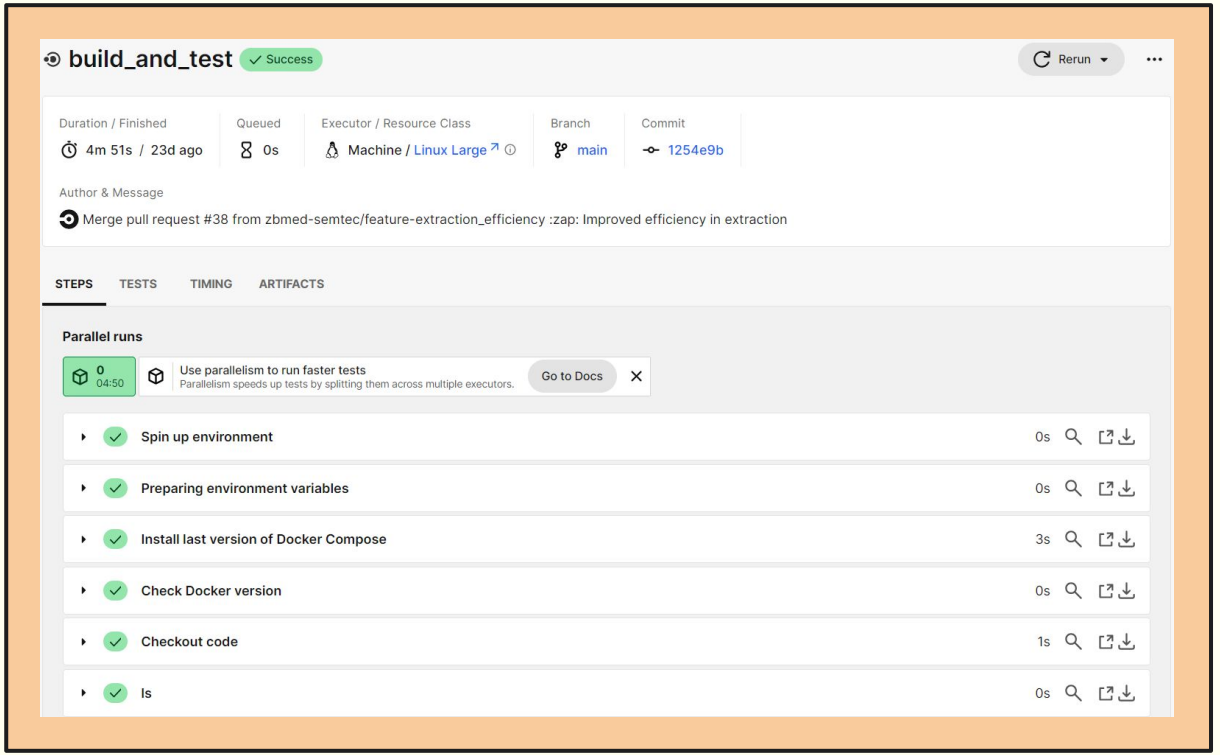


Code Quality

circleci ~ GitHub Actions

Black

pytest



Documentation

Inline Code Documentation

```python
def query(self, sql: str, params: Dict[str, Any] = None) -> pd.DataFrame:
    """
    Execute a SQL query and return results as DataFrame.

    Args:
        sql (str): SQL query to execute
        params (Dict[str, Any], optional): Query parameters

    Returns:
        pd.DataFrame: Query results as DataFrame
    """
```

```python
def get_recent_models_metadata(
    self, limit: int, latest_modification: datetime, threads: int = 4
) -> pd.DataFrame:
    """
    Retrieve recent models metadata from HuggingFace API.

    Args:
        limit (int): Maximum number of models to fetch.
        latest_modification (datetime): The latest modification date to filter models.
        threads (int): Number of threads for parallel processing. Defaults to 4.

    Returns:
        pd.DataFrame: DataFrame containing model metadata
    """
```

README.md in relevant folders

MLentory Extraction/Transformation/Loader (ETL) Pipeline

MLentory is centered around information on ML models, how to harmonize that data, and how to make it available and searchable on an FDO (FAIR Digital Object) registry.

Purpose

To build a system that extracts ML (Machine Learning) model information from different platforms, normalizes that data in a common format, stores it, and shares it in a FDO registry to facilitate IR (Information Retrieval) and comparison/recommendation systems.

This TDD (Technical Design Document) will help new contributors understand and old ones remember what decisions were made on the system's design, the motivation behind them, and their impact. The document focuses on the design of the ETL pipeline to collect, transform, and store extracted information.

Run The Project

There are different things you can execute in this project.

- The first one is the whole ETL pipeline, which is the main component of the project. See instructions here: ETL Pipeline
- The second one is the test component, which is the component that tests the ETL pipeline. See instructions here: Test Component

Background

This project is part of the NFDI4DataScience initiative, a German Consortium whose vision is to support all steps of the complex and interdisciplinary research data lifecycle, including collecting/creating, processing, analyzing, publishing, archiving, and reusing resources in Data Science and Artificial Intelligence.

TDD with the big picture.

MLentory Technical Design Document

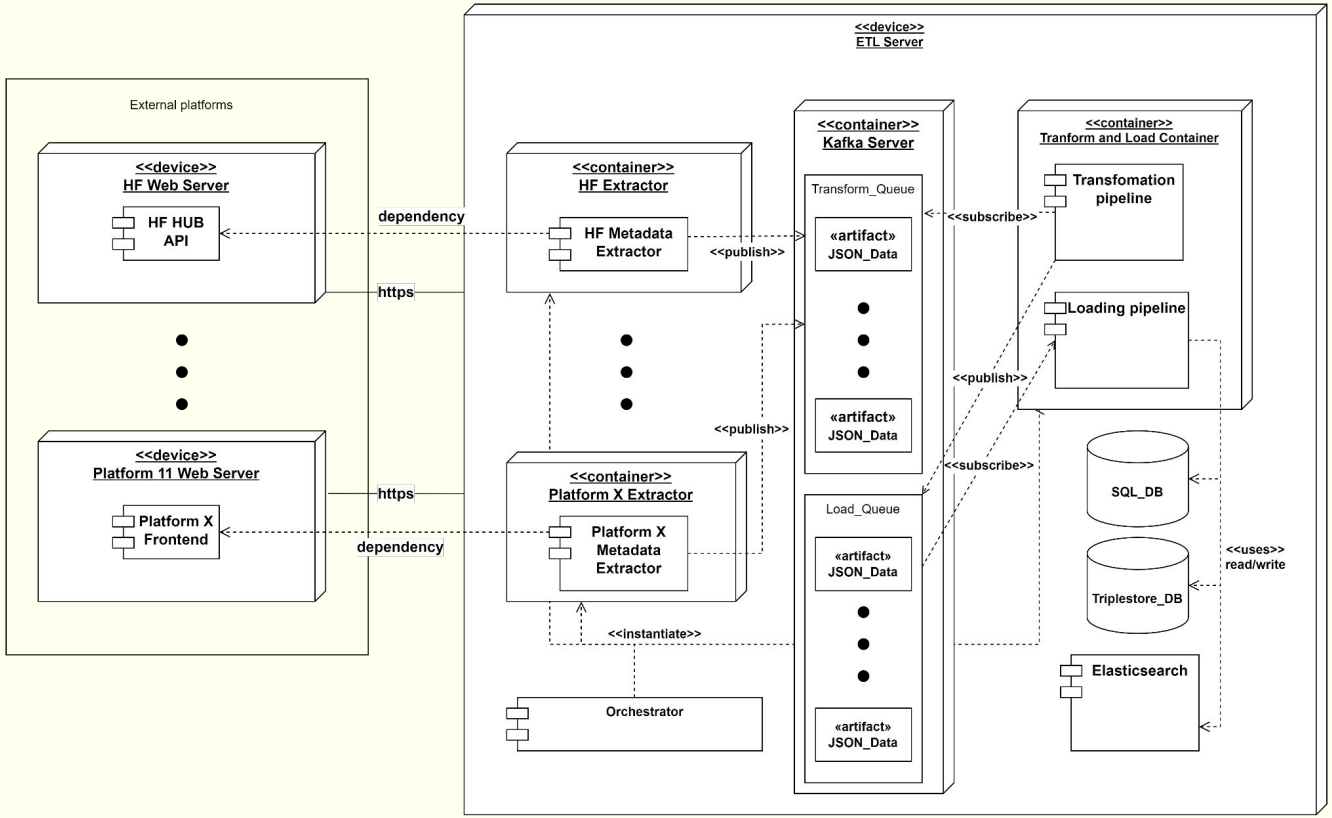
Created by Nelson David Quiñones Virgen and Leyla Jael Castro.

Table of Contents

Demo

Lessons Learned:

- You wont need it
- KISS
- Create an MVP



Next steps

- Make the chat affect GUI elements.
- Improve our unstructured text extraction:
  - Test more LLMs.
  - Improve context selection.
  - Explore better prompts.
- Add more resources and platforms.
- Deploying LLMs in production.
- Share our tool and get more feedback!

Thanks for listening!