universidade de aveiro

Development of a scalable architecture to extract metadata from distributed medical databases

André Silva Pedrosa

Master's Dissertation in Informatics Engineering

Supervisor: José Luis Guimarães Oliveira

Context

Researchers want to perform studies

EHR Databases

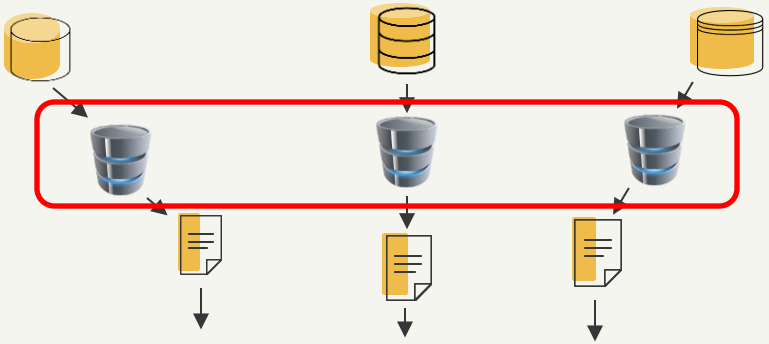
No direct access to data

Context

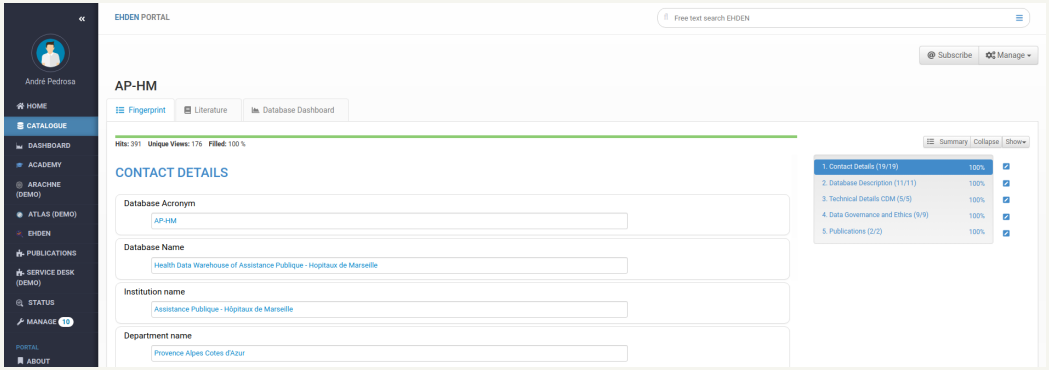
Metadata

Researchers can find databases

Context



EHDEN
EUROPEAN HEALTH DATA & EVIDENCE NETWORK



Objectives

Provide a platform capable of holding and displaying metadata in an intuitive and user-friendly way

Develop or find a tool that extracts metadata from a database

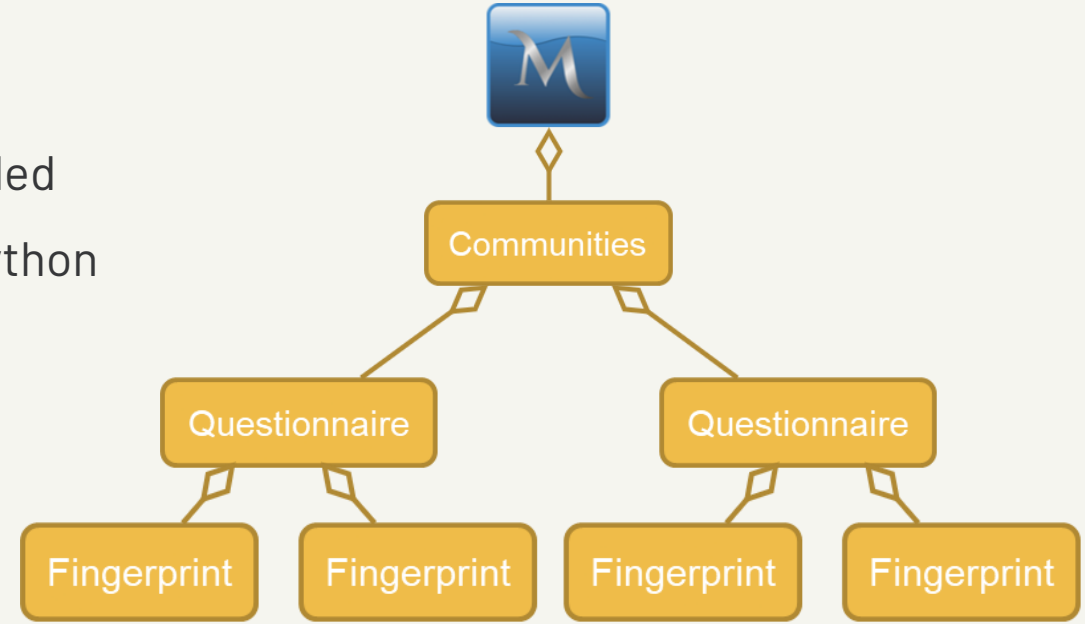
Design a system capable of sending data to the platform, to keep it up-to-date

Background

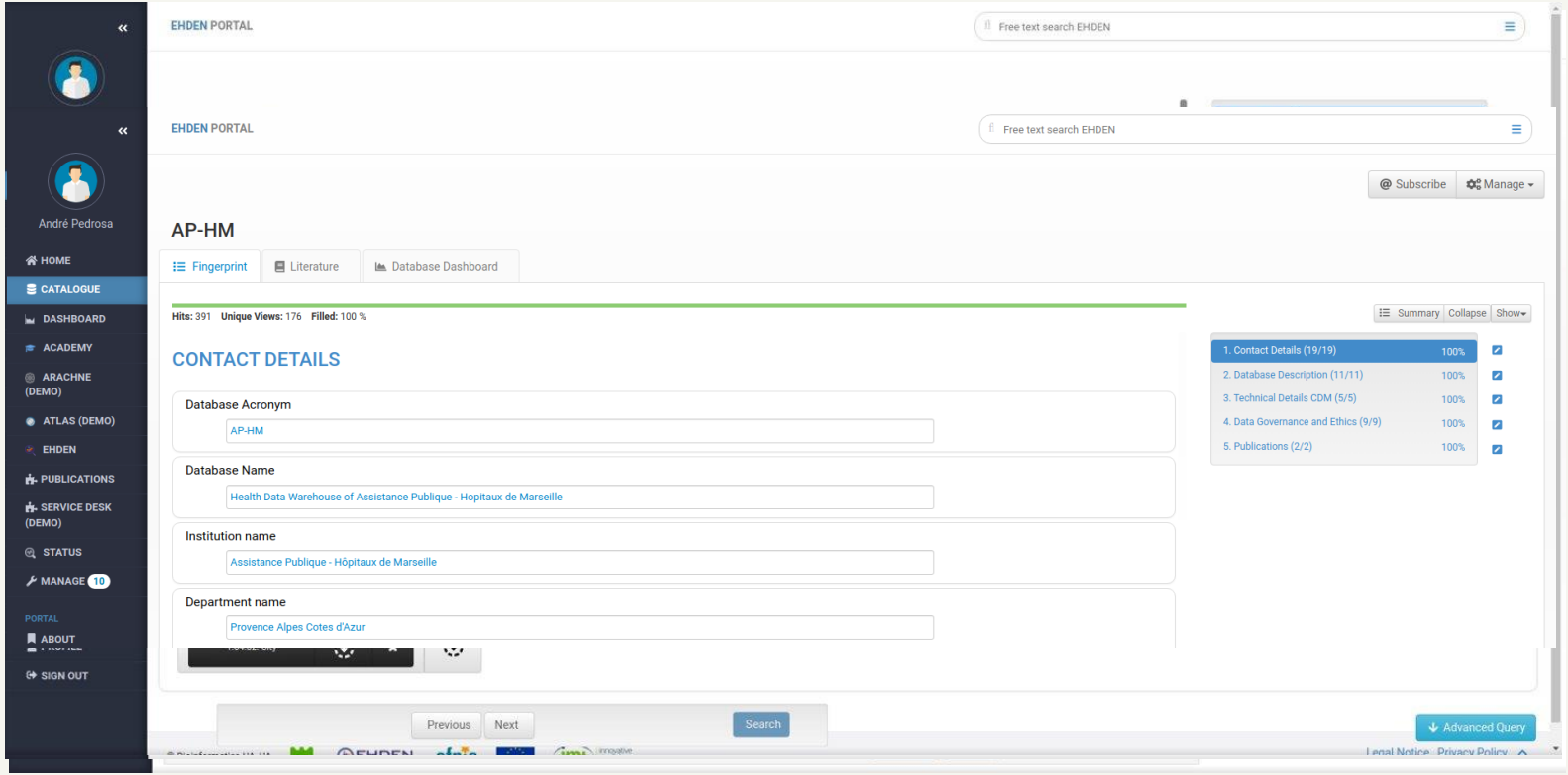
Tool Name	Open Source	Visualization/Interaction		Extraction	Network
		Data protection	FAIR		
eGenVar [1]	✓ [4]	✓ (Users + Permissions)	✓	✗	✗
MONTRA [2]	✓ [5]	✓ (Role based)	✓	✗	✗
REDCap [10]	✗	✓ (Role based)	✓	✗	✗
Data Sphere [13]	✗	✓ (Authorized Users Only)	✗	✗	✗
MOLGENIS [15]	✓ [6]	✓ (Role based)	✓	✗	✗
Cafe Variome [18]	✗	✓ (Role based)	✓	✗	✓
Mica & Opal [19]	✓ [7]	✗	✓	✗	✗
BioSharing [20]	✓ [8]	✗	✓	✗	✗
Dataverse [22]	✓ [9]	✓ (Role Based)	✓	✗	✗
NADA [23]	✓ [10]	✓ (Access Request)	✗	✗	✗
ACHILLES [25]	✓ [11]	✗		✓	✗
DataMed [27]	✓ [12]	✗		✓	✗
Xtract [30]	✗	✗		✓	✓
Skluma [31]	✓ [13]	✗		✓	✗
GAAIN [32]	✗	✗		✗	✓
PopMedNet [33]	✗	✗		✗	✓
EHR4CR [34]	✗	✗		✗	✓
NextGen Connect [35]	✓ [14]	✗		✗	✓

Metadata Visualization: MONTRA

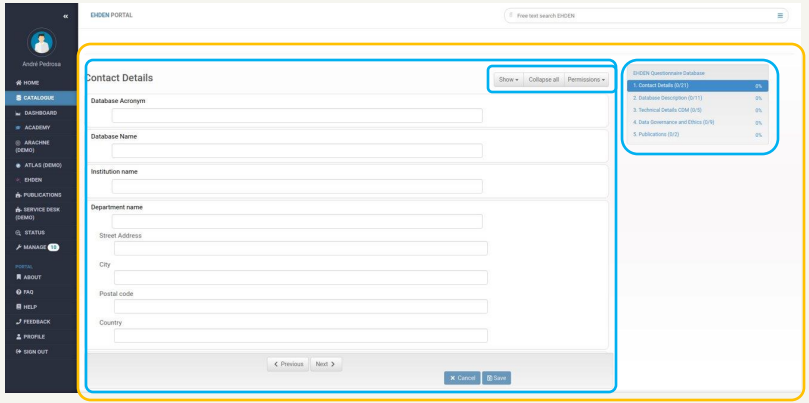
- Database-Centric design
- FAIR access to data provided
- Implemented in Django/Python
- Key Concepts
  - Communities
  - Questionnaires
  - Fingerprints

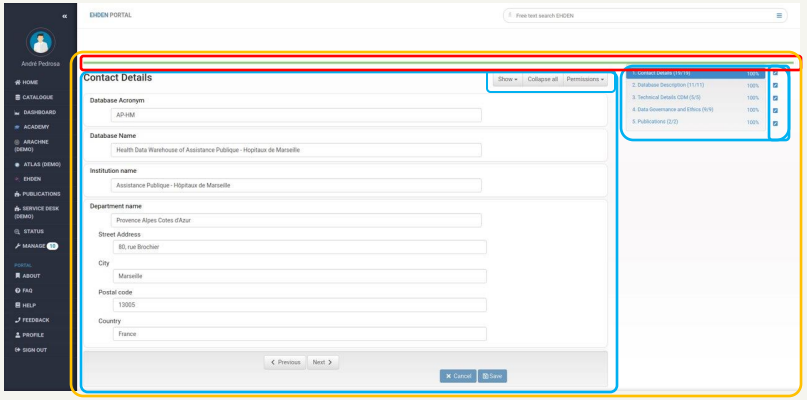


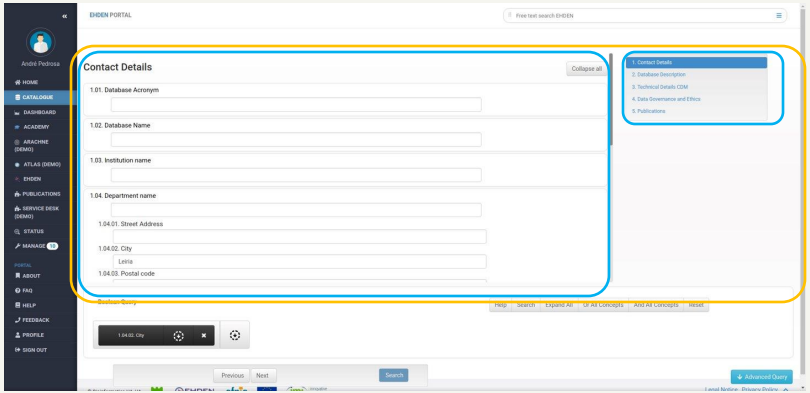
Fingerprint User Interface

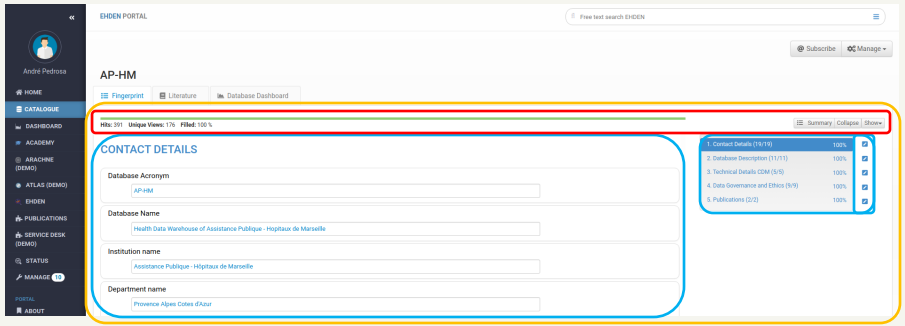


Fingerprint User Interface



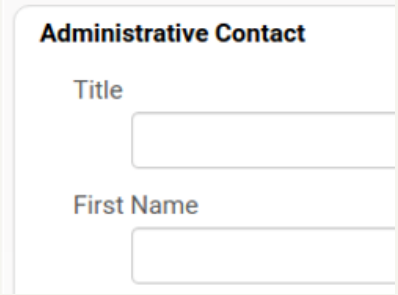






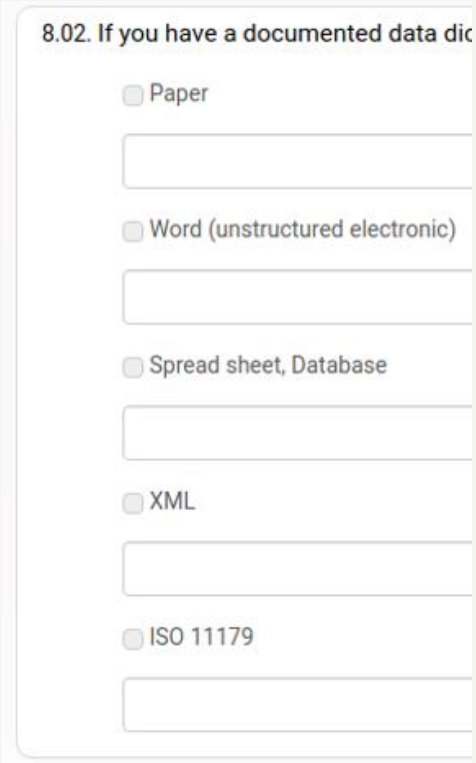
Fingerprint Schema

Type	Text/Question	Level/Number
Category	**Administrative Contact**	h1
Question	Title	h2
Question	First Name	h2



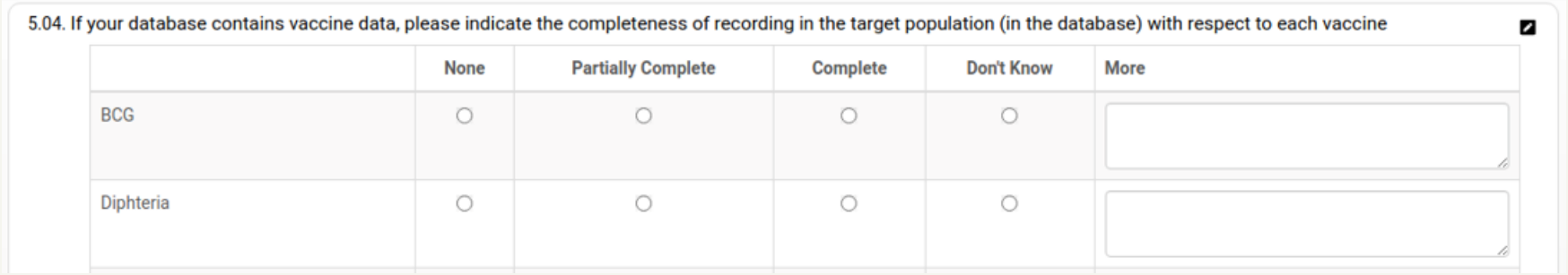
Fingerprint Schema

Type	Text/Question	Data type	Value list
Question	If you have a documented data dictionary, is the data dictionary a document (paper or electronic) or structured (spread sheet, database, XML, ISO 11179 etc.)	choice-multiple	Paper{...}|Word (unstructured electronic){...}|Spread sheet, Database{...}|XML{...}|ISO 11179{...}



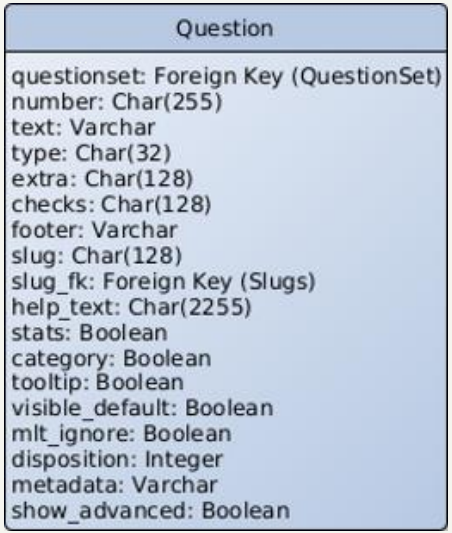
Fingerprint Schema

Type	Text/Question	Data type	Value list
Question	If your database contains vaccine data, please indicate the completeness of recording in the target population (in the database) with respect to each vaccine	choice-tabular	None|Partially Complete|Complete|Don't Know\\BCG|Diphteria|Haemophilus influenzae|Hepatitis A|Hepatitis B|HPV|Influenza|Measles|Meningococcal|Mumps|Pertussis|Pneumococcal|Poliomyelitis|Rabies|Rotavirus|Rubella|Shingles|Tetanus|Tick born encephalitis|Typhoid|Varicella\\choice



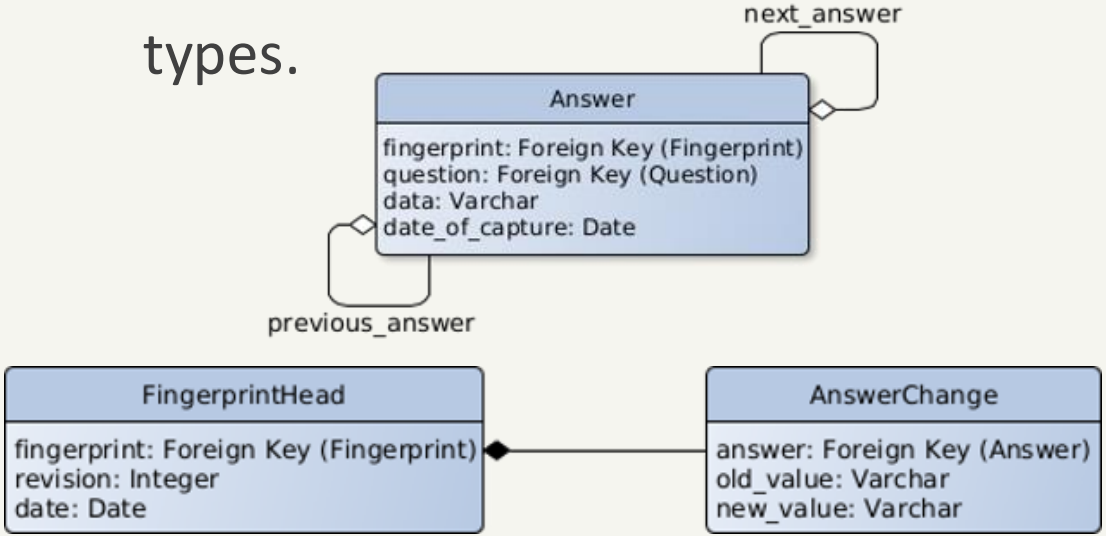
Data Models

- All answers' data is stored as text;
- Only 4 models to store information about a questionnaire:
  - Questionnaire, QuestionSet, Question and Choice;
  - Question model with a high number of fields to store information associated with different question types;



Data Models

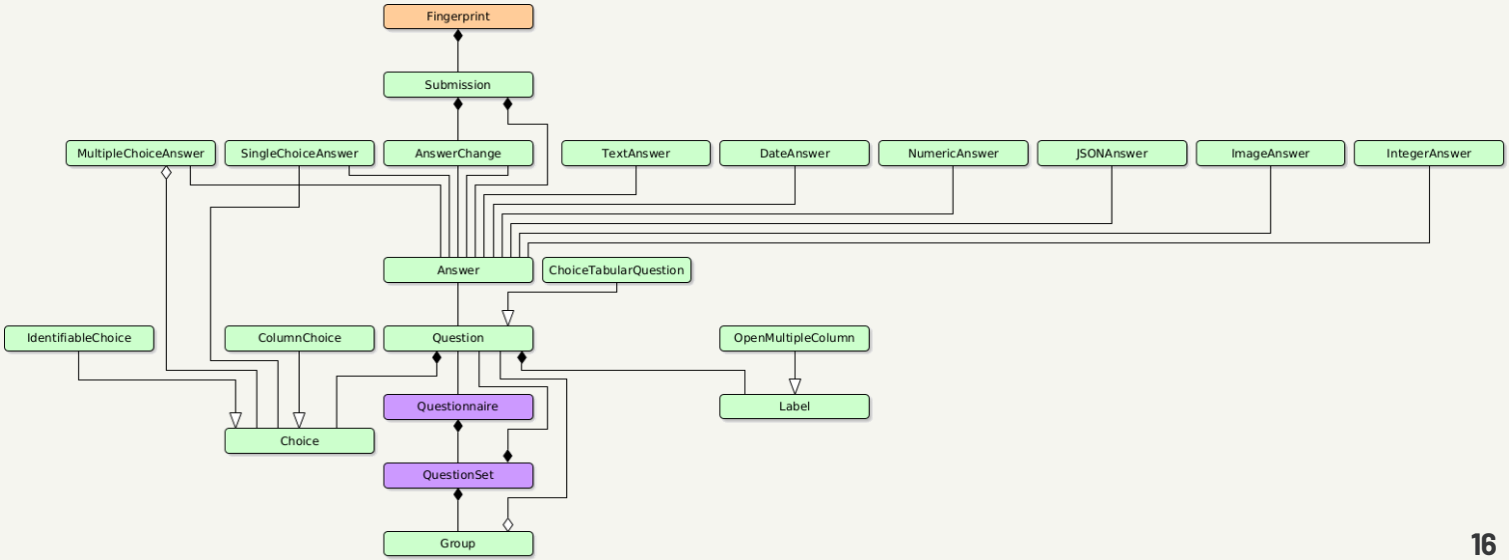
- Other models, related to fingerprints, also had duplicated information and used incorrect data types.



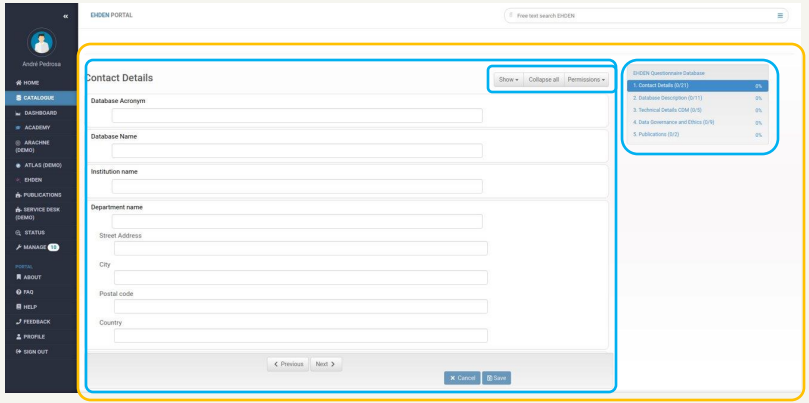
Questionnaire
name: Char(128)
short_description: Char(128)
long_description: Char(512)
logo: Image
redirect_url: Char(128)
slug: Char(128)
disable: Char(128)
in_preview: Boolean
preview_fingerprint: Foreign Key(Fingerprint)

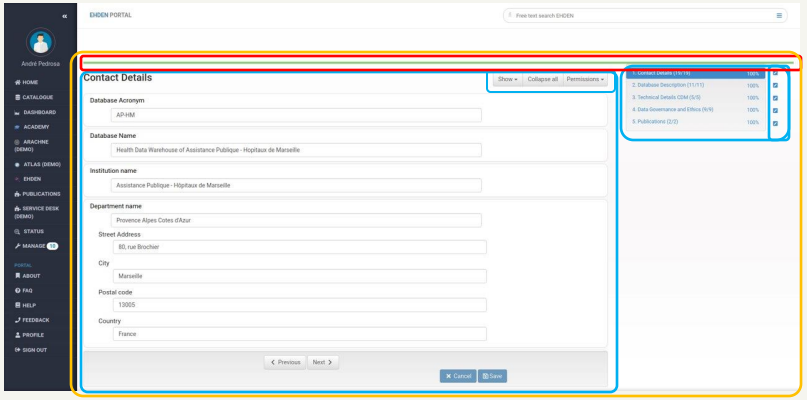
Refactoring: Data Models

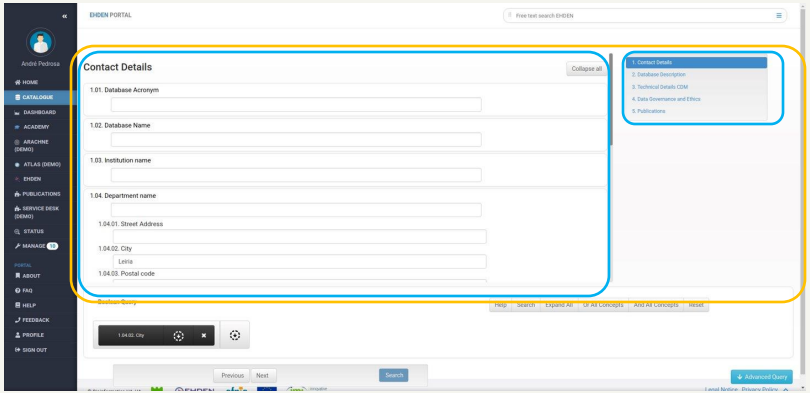
- Other models, related to fingerprints, also had duplicated information and used incorrect data types.

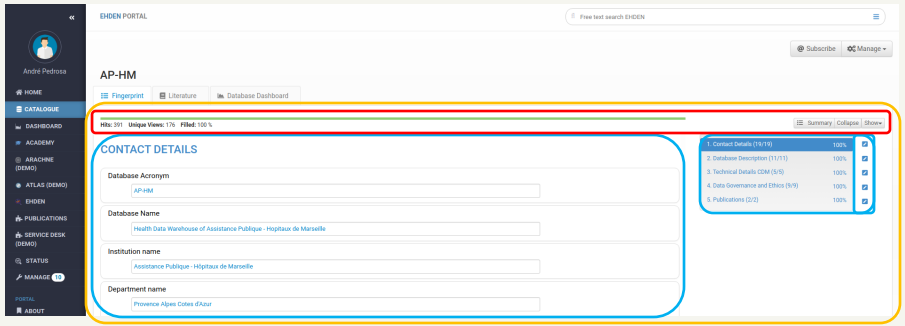


Refactoring: Fingerprint Views









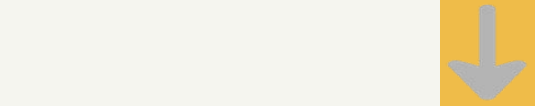
Refactoring: Fingerprint Data Manipulation

- Move the data validation to the backend;
- Make use of Django's built-in data validation system;

Refactoring: Fingerprint Schema

- Fixed clutter problems
- Removed unused columns

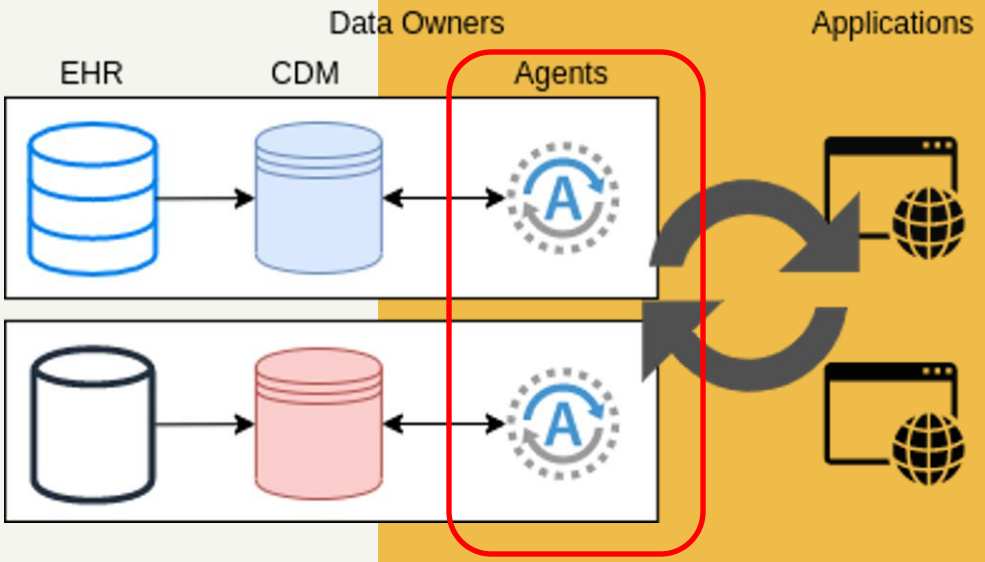
Type	Text/Question	Data type	Value list
Question	If your database contains vaccine data, please indicate the completeness of recording in the target population (in the database) with respect to each vaccine	choice-tabular	None|Partially Complete|Complete|Don't Know\\BCG|Diphteria|Haemophilus influenzae|Hepatitis A|Hepatitis B|HPV|Influenza|Measles|Meningococcal|Mumps|Pertussis|Pneumococcal|Poliomyelitis|Rabies|Rotavirus|Rubella|Shingles|Tetanus|Tick born encephalitis|Typhoid|Varicella\\choice



Type	Text/Question	Data type
Question	If your database contains vaccine data, please indicate the completeness of recording in the target population (in the database) with respect to each vaccine	choice tabular single
TabularChoice	None	
TabularChoice	Partially Complete	
TabularChoice	Complete	
TabularChoice	Don't Know	
TabularRow	BCG	
TabularRow	Diphteria	
TabularRow	Haemophilus influenzae	
TabularRow	Hepatitis A	
TabularRow	Hepatitis B	
TabularRow	HPV	

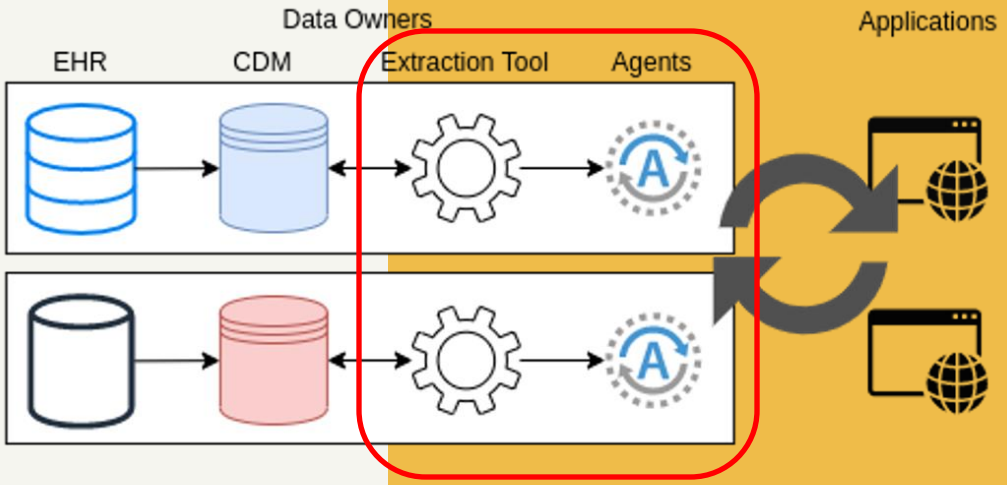
Metadata Extraction & Update

- Agents:
  - Run on data owner's deployment environment;
  - Extracts and sends metadata to the applications;
- Data owners might not want/be able to provide direct access to data;



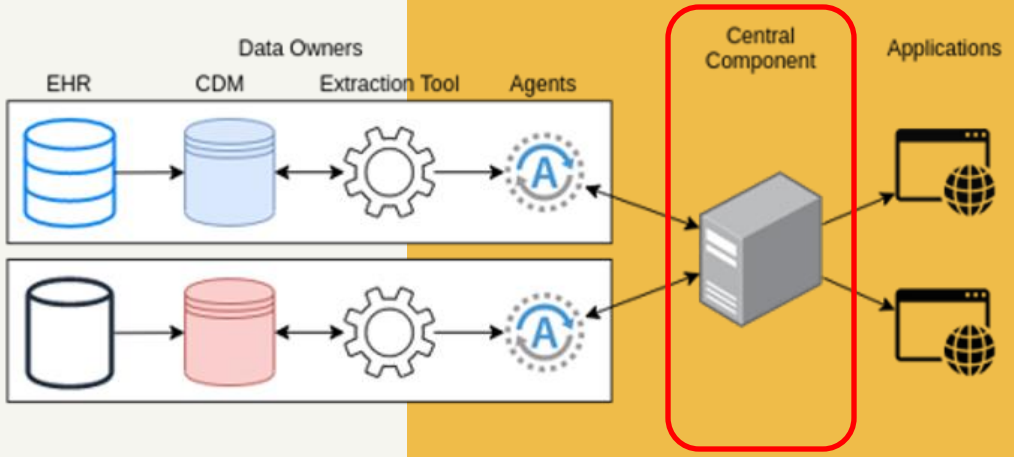
Metadata Extraction & Update

- Agent is now only in charge of sending metadata to applications;
- Data owners take care of the extraction process;
- How data gets to the applications?
  - Peer-to-peer architecture?



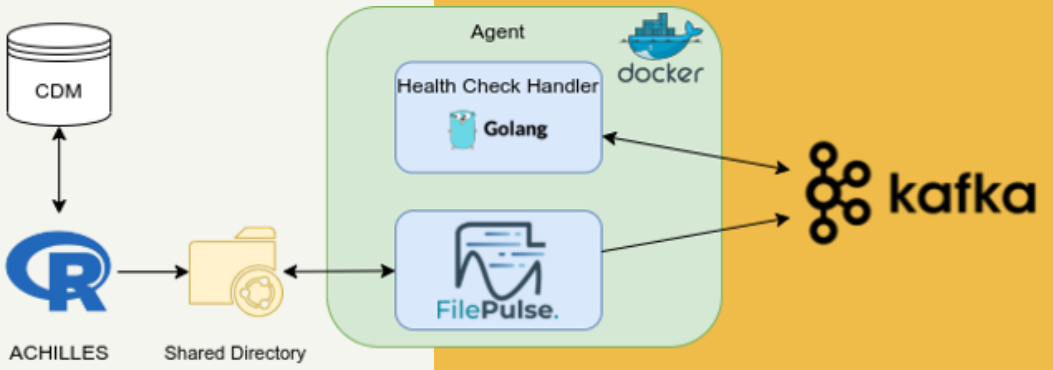
Metadata Extraction & Update

- A central component receives the data from the agents and sends it to the applications;



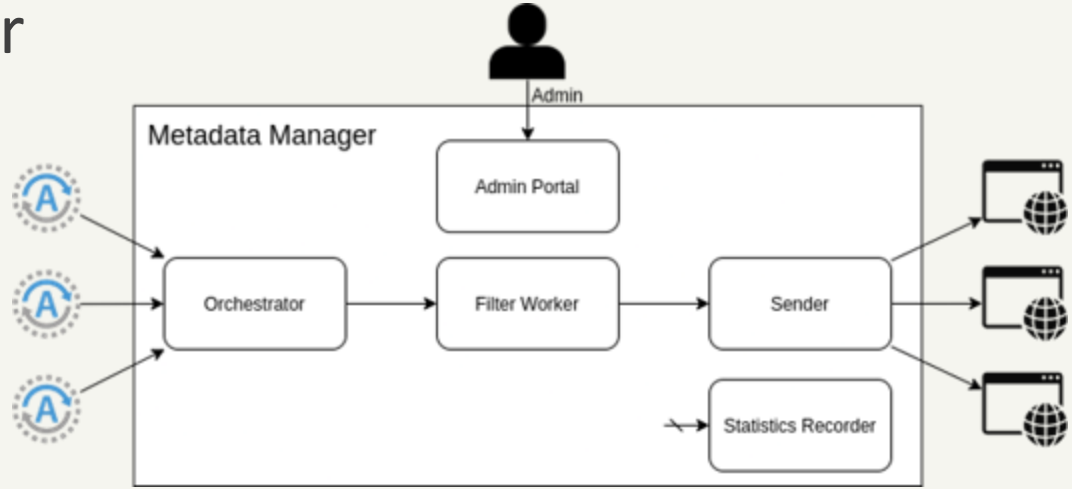
Metadata Extraction

- ACHILLES
- Asynchronous messaging systems:
  - Kafka
- FilePulse Source connector



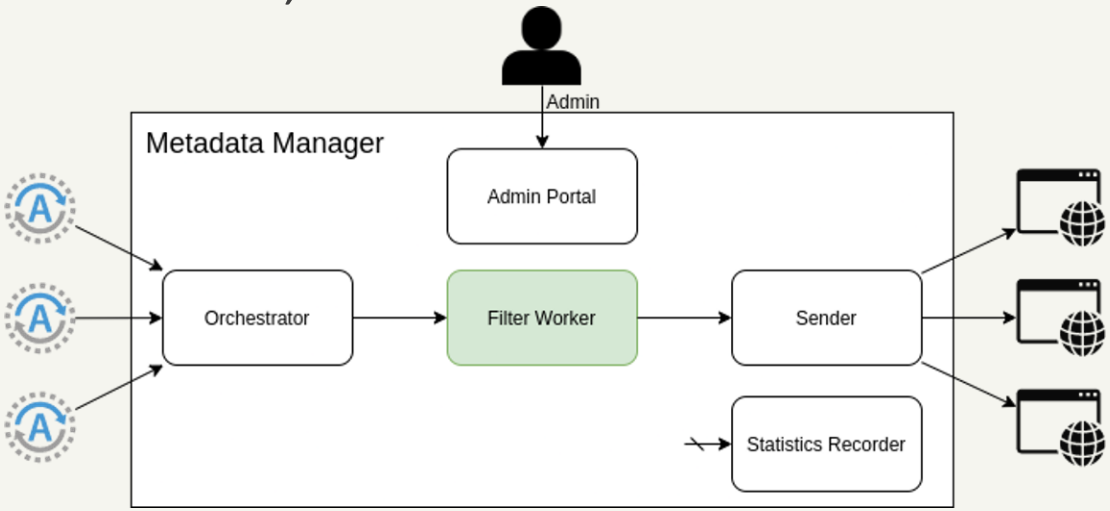
Metadata Update

- Data is in Kafka, now what?
- Kafka Sink connectors do not allow customization;
- Kafka deals with data as unbounded data flow;
- Metadata Manager
  - 5 components



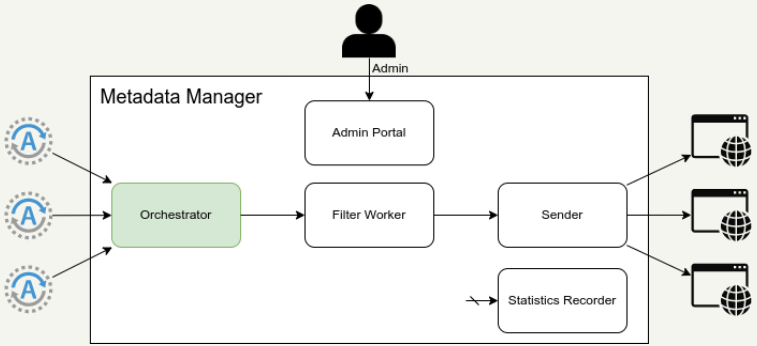
Filter Worker

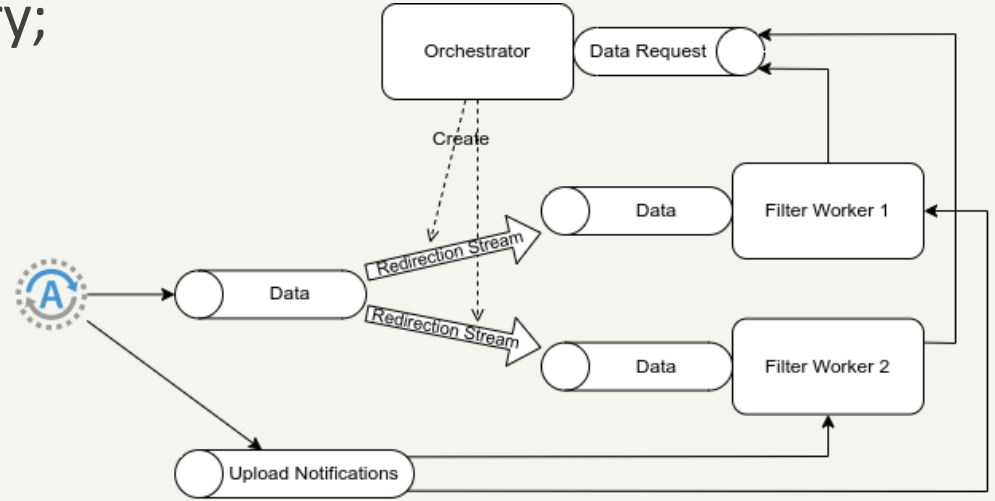
- Application might not require the entire data received from the databases;
- Several filters at the same time;
- Implemented in Go;
- Can be scaled out.



Orchestrator

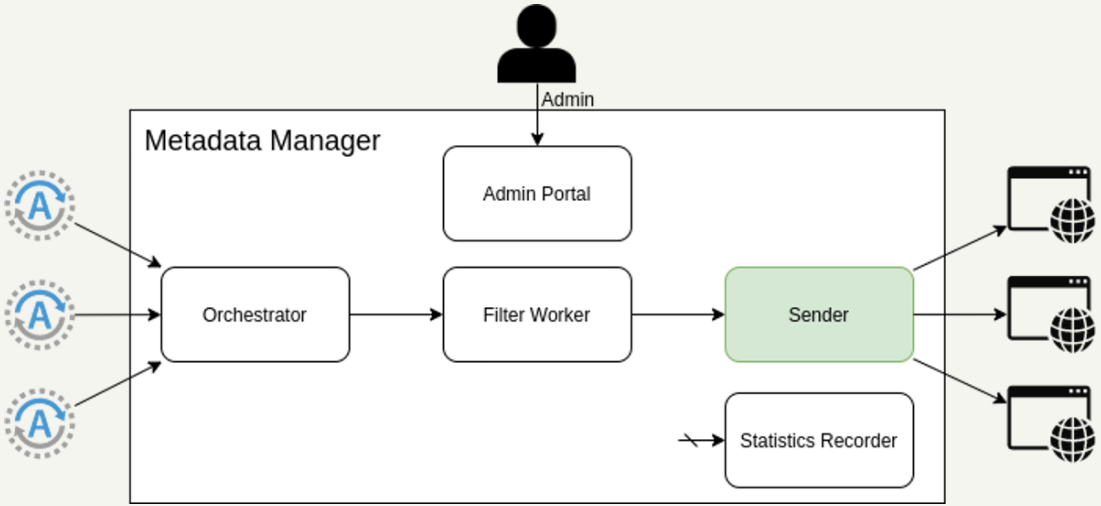
- Distributes the data received from the databases across the existing Filter Worker instances;
- Uses Kafka Streams library;
- Implemented in Java;
- Can be scaled out;





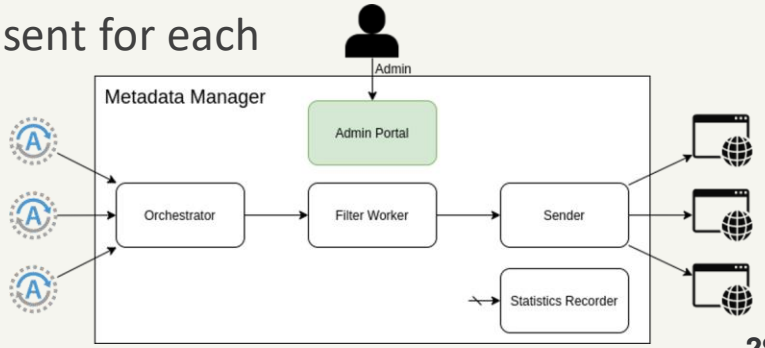
Sender

- Sends data to the applications in form of a HTTP request;
- Allows customization of the several request properties (method, URL, body, ...);



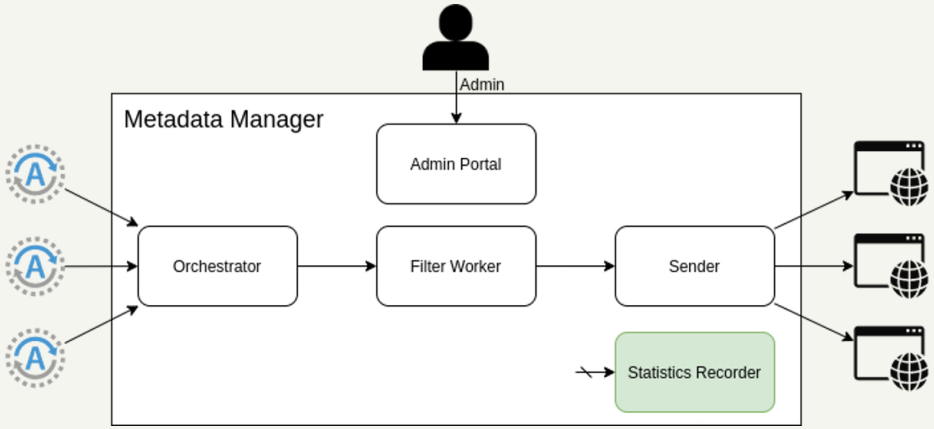
Admin Portal

- Offers an interface to manage the whole Metadata Manager;
- Used to:
    - register new databases;
    - group databases in communities;
    - check statistics to get feedback on data flowing on the system;
    - specify the format template of data to be sent for each application.
- Composed by two components:
    - Frontend - built using React;
    - Backend API - built using Django.



Statistics Recorder

- Stores statistics of the system;
- Implemented as a Kafka Sink Connectors.

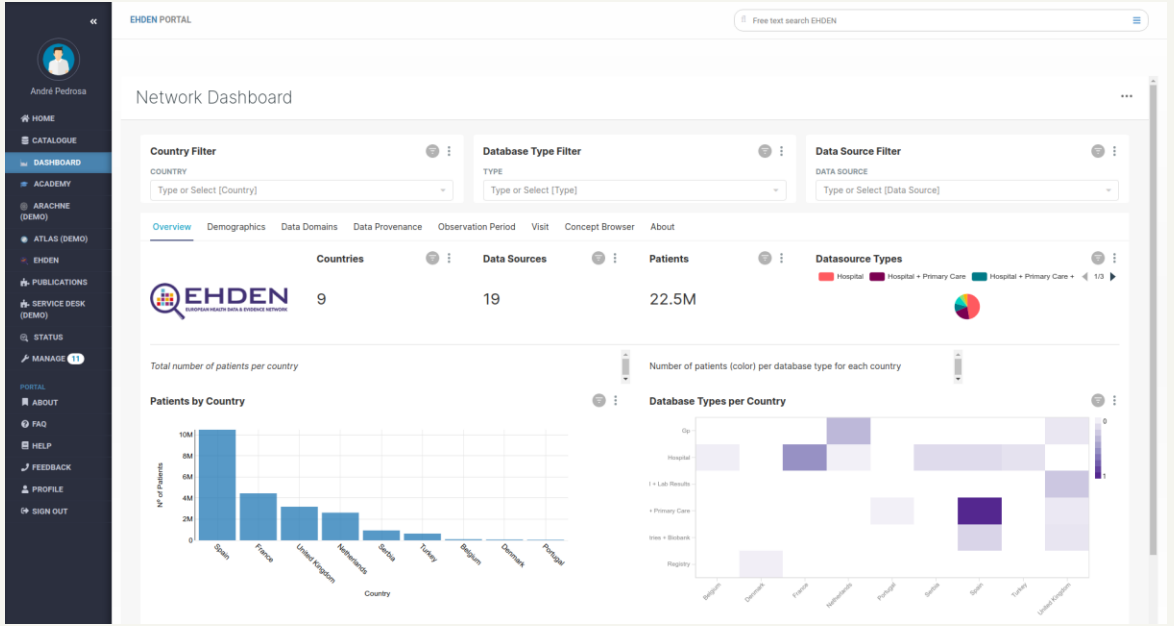


Results

- **Montra Refactoring** - Improvements were done to a fully-fledged tool for metadata visualization;
    - Keep a compatible was a challenge during the development process;
    - Pull request waiting for review with over 150 commits with change on about 200 files.
- **Data Profiling** - A tool able to extract data from databases conforming to the OMOP CDM was proposed:
    - Data owner's data privacy concerns were considered;
    - A System capable of gathering and sending extracted data to the applications was developed
    - Simple components with well-defined objectives;

Conclusions

- EHDEN WP4
- GitHub
- Dashboard



Any Questions?