TheFSM | The Food Safety Market

The Food Safety Market: An SME-powered industrial data platform to boost the competitiveness of European food certification

D2.2 - Data Services

DELIVERABLE NUMBER	D2.2
DELIVERABLE TITLE	Data Services
RESPONSIBLE AUTHOR	Svetla Boytcheva (SIR)

Co-funded by the Horizon 2020
Framework Programme of the European Union

GRANT AGREEMENT N.	871703
PROJECT ACRONYM	TheFSM
PROJECT FULL NAME	The Food Safety Market: An SME-powered industrial data platform to boost the competitiveness of European food certification
STARTING DATE (DUR.)	01/02/2020 (36 months)
ENDING DATE	31/01/2023
PROJECT WEBSITE	www.foodsafetymarket.eu
COORDINATOR	Nikos Manouselis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	nikosm@agroknow.com
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Stefano Bertolo
WORKPACKAGE N. | TITLE	WP2 | Data
WORKPACKAGE LEADER	SIRMA AI
DELIVERABLE N. | TITLE	D 2.2 | Data Services
RESPONSIBLE AUTHOR	Svetla Boytcheva (SIR)
REPLY TO	svetla.boytcheva@ontotext.com
DOCUMENT URL	
DATE OF DELIVERY (CONTRACTUAL)	31 January 2022 (M24)
DATE OF DELIVERY (SUBMITTED)	31 January 2022 (M24)
VERSION | STATUS	2.0 | Final
NATURE	Demonstrator (DEM)
DISSEMINATION LEVEL	Public (P)
AUTHORS (PARTNER)	Svetla Boytcheva (SIR), Plamen Tarkalanov (SIR), Nikola Tulechki (SIR), Pavlin Gyurov (SIR), Nikola Rusinov (SIR), Antoniy Kunchev (SIR)
CONTRIBUTORS	Taniou Pinelopi, Theodorakopoulou Zacharoula (Agroknow)
REVIEWER	Manos Karvounis (Agroknow)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Table of contents	15/11/2020	Svetla Boytcheva (SIR)
0.2	Initial draft version	15/12/2020	Plamen Tarkalanov, Pavlin Gyurov, Nikola Rusinov (SAI)
0.3	Sections contribution, corrections, and observations	21/01/2021	Plamen Tarkalanov, Nikola Tulechki, Pavlin Gyurov, Nikola Rusinov (SAI)
1.0	Document refinement, addition of figures and descriptions	28/01/2021	Svetla Boytcheva (SIR)
1.2	Updates to sections Data Ingestion, Data Curation and Data Publishing Services	10/12/2021	Antoniy Kunchev (SAI)
1.4	Added new section Automated Data Processing Service	9/01/2022	Antoniy Kunchev (SAI)
1.6	Updated ToC, added diagrams and screenshots	24/01/2022	Antoniy Kunchev (SAI)
1.8	Added new section 7 - Risk assesment and prediction services	28/01/2022	Taniou Pinelopi, Theodorakopoulou Zacharoula (Agroknow)
1.9	Internal review	30/01/2022	Manos Karvounis (Agroknow)
2.0	Sections contribution, added descriptions and figures for new services, updated outdated information	31/01/2022	Svetla Boytcheva, Antoniy Kunchev (SAI)

PARTNERS	CONTACT
Agroknow IKE (Agroknow, Greece)	Nikos Manouselis (Agroknow) nikosm@agroknow.com
SIRMA AI EAD (SAI, Bulgaria)	Zlatina Marinova (SAI) zlatina.marinova@ontotext.com
GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS (UBITECH, Greece)	Danai Vergeti (UBITECH) vergetid@ubitech.eu
AGRIVI DOO ZA PROIZVODNJU, TRGOVINU I USLUGE (Agrivi d.o.o., Croatia)	Filip Gerin (Agrivi d.o.o.) filip.gerin@agrivi.com
PROSPEH, POSLOVNE STORITVE IN DIGITALNE RESITVE DOO (PROSPEH DOO, Slovenia)	Martina Poberaj (PROSPEH DOO) martina.poberaj@tracelabs.io
UNIVERSITAT WIEN (UNIVIE, Austria)	Tima Anwana (UNIVIE) tima.anwana@univie.ac.at
STICHTING WAGENINGEN RESEARCH (WFSR, Netherlands)	Yamine Bouzembrak (WFSR) yamine.bouzembrak@wur.nl
TUV- AUSTRIA ELLAS MONOPROSOPI ETAIREIA PERIORISMENIS EUTHYNIS (TUV AU HELLAS, Greece)	Kostas Mavropoulos (TUV AU HELLAS) konstantinos.mavropoulos@tuv.at
TUV AUSTRIA ROMANIA SRL (TUV AU ROMANIA, Romania)	George Gheorghiou (TUV AU Romania) george.gheorghiu@tuv.at
VALORITALIA SOCIETA PER LA CERTIFICAZIONE DELLE QUALITA'E DELLE PRODUZIONI VITIVINICOLE ITALIANE SRL (VALORITALIA, Italy)	Francesca Romero (Valoritalia) francesca.romero@valoritalia.it
TUV AUSTRIA CYPRUS (TUV AU CYPRUS, Cyprus)	Sousanna Charalambidou (TUV AU CYPRUS) sousanna.charalambidou@tuv.at

ACRONYMS LIST

API	Application Programming Interface
CSV	Comma-separated values
HTTP	Hypertext Transfer Protocol
GREL	Google Refine Expression Language
JSON	JavaScript Object Notation
RDF	Resource Description Framework
RDBMS	Relational Data Base Management System
REST	Representational state transfer
SHACL	Shapes Constraint Language
SKOS	Simple Knowledge Organization System Primer
SOML	Semantic Objects Modelling Language
SPARQL	SPARQL Protocol and RDF Query Language
TheFSM	The Food Safety Market
XML	Extensible Markup Language
WKT	Well-known text

EXECUTIVE SUMMARY

This document presents the data services developed within The Food Safety Market (TheFSM) project – data ingestion service; data curation service, data publishing service, and risk assessment and prediction services. There are brief descriptions of the main functionalities for each service, examples illustrated with screenshots and discussion about the main tools and technologies used in the implementation. The developed services are based on the detailed analysis of business, data and technical requirements of all use cases. The best practices and standards are taken into consideration as well. This version of the document describes the services developed during the first 24 months of the project. It will be updated again at M36 to describe the final versions of data services, appropriately fine-tuned according to the TheFSM project needs.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Scope

The Food Safety Market (TheFSM) project aims to build a full stack of data management and processing services in order to equip the FSM Platform with full functionalities for data ingestion; data curation, reconciliation, data publishing, and risk assessment and prediction. This will allow the FSM Platform users to easily tackle the problems that typically arise with heterogeneous data integration and processing.

The main purpose of this document is to provide detailed information about the data services that are developed in order to extend the FSM Platform functionality and make it easily accessible for all suppliers and consumers. The services that are described in the document are: Data Ingestion Service, Data Curation Service, Data Publishing Service, Automated Data Processing Service, Risk Assessment and Prediction Services.

1.2 Services

The main purpose of the Data Ingestion Service is to implement semantically based services that will make uploading, mapping and data transformation easier and user friendly, for different stakeholders.

Data Curation Service implements an advanced storage environment to make life easier for domain experts, when they perform tasks like deduplication, enrichment, mapping, entity matching, cleaning, etc.

The Data Publishing Services provides a data publishing gateway that will enhance the FSM Platform with more sophisticated semantically driven services for data providers and APIs for data consumers.

The Automated Data Processing Service combines functionalities for data ingestion and data curation and performs automated data processing so that end users of the FSM Platform can take advantage of cleaning, semantic enrichment and RDFization of their data, even if they do not posess the necessary expertise and understanding of data engineering processes.

The Risk Assessment and Prediction Services integrate and process appropriate data to train and develop prediction models and algorithms that can be used to estimate and also to predict risks and risk scores at the multiple levels: incidents and emerging threats, product risks, and supplier risks.

1.3 Relation to other deliverables

The data services functionalities are based on the detailed analysis of business, data and technical requirements of all use cases described in D1.1 "Report on Requirements for TheFSM", in compliance with D1.2: "TheFSM Development Roadmap", D2.3 "Report on Data Population" and D3.1 "TheFSM Open Reference Architecture". The best practices, state-of-the-art and standards are taken into consideration as well. Finally, the Risk Assessment and Prediction services power the relevant end-user dashboards of FOODAKAI 2.0 and Food Inspector applications, as reported in D4.1.

1.4 Outputs of the deliverable

Version 2 of this deliverable provides an overview of the current state of the services and their evolution up to M24, from the initial vision, design and implementation. As a result, the reader may acquire a deep view into what each service is responsible, how they work internally, and what their purpose is in the FSM Platform as whole.

Also, the document provides targeted technical aspects and descriptions of the data processing flows, how and why they are realized, appropriate examples and references to the code bases, deployments and relevant documentation.

2. DATA INGESTION SERVICES

2.1 Data Ingestion using OntoRefine

GraphDB[1] OntoRefine is an upgraded version of the open source OpenRefine2 data transformation tool. It allows a quick mapping of any structured data to a locally stored RDF3 schema in GraphDB. The visual interface is optimized to guide users in choosing the right predicates and types, in defining the datatype to RDF mappings, as well as in implementing complex transformation using OpenRefine's GREL4 language. GREL is the Google Refine Expression Language that helps in defining complex transformations of the data.

Data Ingestion Example

The following screen shows a visual mapping of tabular data to RDF. Here, the example data is about restaurants with various properties such as the name (in multiple languages) and location (expressed as WKT literals) mapped using the relations required by the chosen ontology.

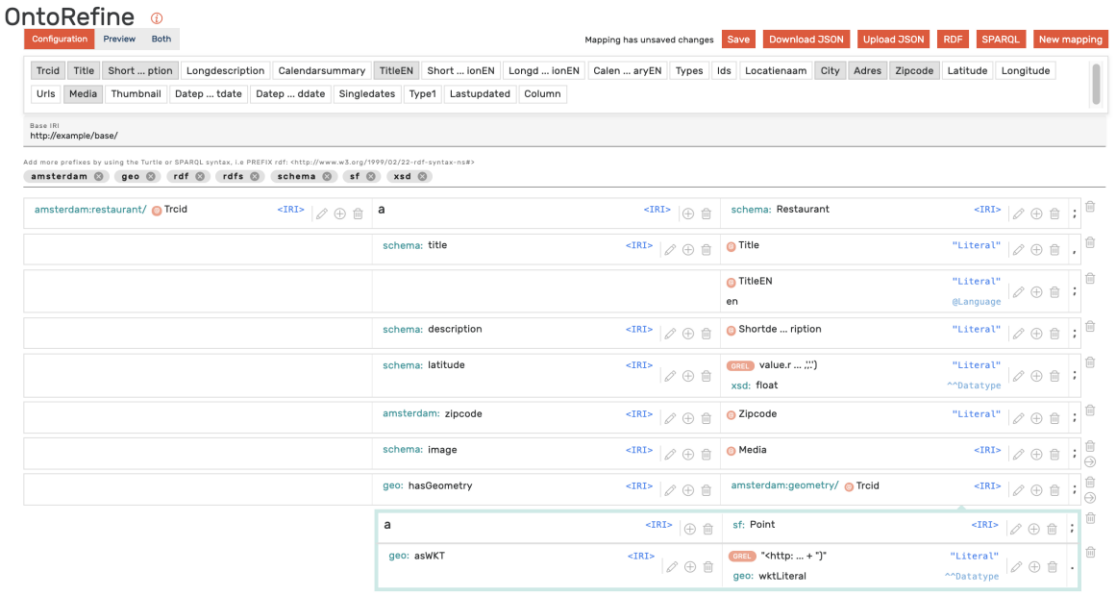

Figure 1: Example for data ingestion – visual mapping of tabular data to RDF

[1] https://www.ontotext.com/products/graphdb/

[2] https://openrefine.org/

[3] https://www.w3.org/RDF/

[4] https://guides.library.illinois.edu/openrefine/grel

This visually defined mapping result in the following RDF:

```
@base <http://example/base/> .

@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

@prefix schema: <http://schema.org/> .

@prefix geo: <http://www.opengis.net/ont/geosparql#> .

@prefix amsterdam: <https://data/amsterdam/nl/resource/> .

@prefix sf: <http://www.opengis.net/ont/sf#> .

@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


<https://data/amsterdam/nl/resource/restaurant/669d7d82-8962-4e88-b2e1-7b8706633aa0>

 a schema:Restaurant;

 schema:title "Smits Noord-Zuid Hollandsch Koffiehuis", "Smits Noord-Zuid Hollandsch Koffiehuis"@en;

 schema:description "Het Smits Koffiehuis ontleent haar ontstaan aan de stoomtram die de verbinding onderhield met Amsterdam naar het noorden van de provincie en is in 1919 gebouwd. Nu is er een restaurant en een koffiebar. Ook is hier een informatiekantoor van Amsterdam Marketing gehuisvest.";

 schema:latitude "0"^^xsd:float;

 amsterdam:zipcode "1012 AB";

 schema:image <https%3A//media.iamsterdam.com/ndtrc/Images/20101122/ec8faec5-5cd5-43d6-b0fa-eb0dab65e278.jpg>;

 geo:hasGeometry <https://data/amsterdam/nl/resource/geometry/669d7d82-8962-4e88-b2e1-7b8706633aa0>;

 amsterdam:uniquelocation _:node1em9j7qmhx179149;

 amsterdam:valuelocation _:669d7d82-8962-4e88-b2e1-7b8706633aa0 .


<https://data/amsterdam/nl/resource/geometry/669d7d82-8962-4e88-b2e1-7b8706633aa0>
```

```
 a sf:Point;

 geo:asWKT "<http://www.opengis.net/def/crs/OGC/1.3/CRS84> POINT (4.9003230
52.3775440)"^^geo:wktLiteral .

_:node1em9j7qmhx179149 amsterdam:address "Stationsplein 10" .

_:669d7d82-8962-4e88-b2e1-7b8706633aa0 amsterdam:city "AMSTERDAM" .
```

The full example of the data and the RDF transformation steps can be found in the official documentation[5] of OntoRefine.

Operations Extraction and RDF Mapping

As it was mentioned in the previous section, OntoRefine is based on OpenRefine, whose main purpose is to provide transformation and processing of different datasets of similar type, in the same way. It records the processing steps performed on a dataset and then repeats the same steps, for cleaning, transformation, RDF mapping, etc., for the rest of the datasets.

Every cleaning, transformation or mapping step that is applied to the given dataset is recorded as a specific operation in JSON format. At any time, this JSON could be extracted and reused for a similar dataset to replay the same steps over it. The steps can be seen under the History tab in the OntoRefine, which currently is named "Undo / Redo" as shown in the figure below.

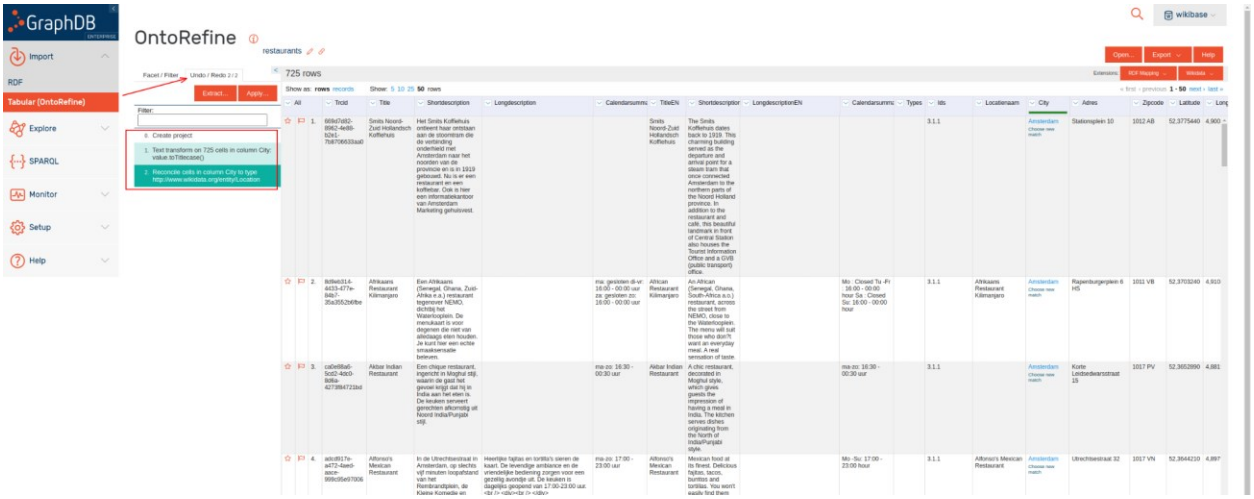

Figure 2: Example of the History tab in OntoRefine

[5] https://graphdb.ontotext.com/documentation/enterprise/loading-data-using-ontorefine.html

Both operations that are applied to the sample dataset are performed on the data in the "City" column. The steps are as follows:

- transforming the values to title case
- reconciling the values against the Location Service

As seen in Figure 2, the steps are recorded and there are several functionalities supporting the process. The important ones are the extraction and application of the steps as JSON, which can be achieved by using the "Extract..." and "Apply" buttons right under the tab name.

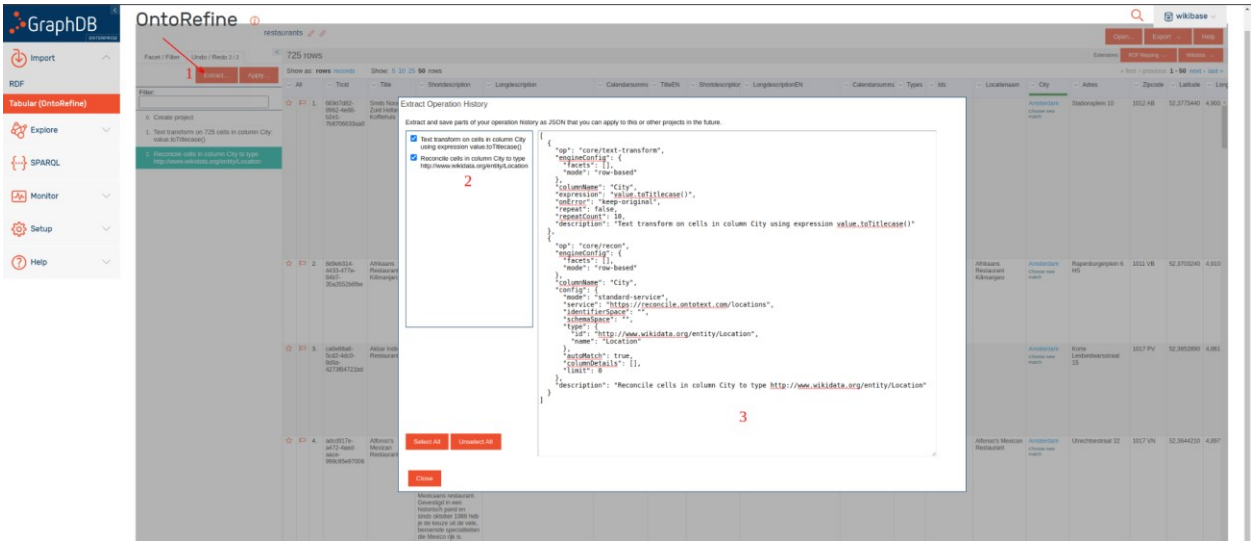

Figure 3: Example operations being extracted from the OntoRefine tool in JSON format

The above figure shows the example of the functionality for extraction of the steps as JSON. When the "Extract..." (1) button is pressed, a new window will appear, which contains a list of the operations (2) that are currently performed over the dataset. The checkbox before the operation names can be used to filter out the operations that should not be included in the JSON.

The actual JSON (3) with the operations representations can be copied and stored in a file or somewhere else, depending on how it is going to be used. This JSON is the most important part that will be used in the automatic pipelines later, where it will be applied to all datasets, which have the same type as the one used to generate the JSON. This gives us the possibility to process large amounts of datasets of the same type quickly and efficiently.

All of the operations that are done over the dataset will be recorded and can be exported, even the RDF mapping, which is created with the help of "RDF mapping" tool. This mapping is required for its representation and storing in the semantic database, which will be done later in the project.

This manual generation of the operations JSON will be done for the different types of datasets that will be uploaded to the FSM Platform. The work will be done by domain experts, where the knowledge of the structure and the model of the data is very important.

3. DATA CURATION SERVICES

3.1 Data Curation using OntoRefine

Besides defining and running transformations, OntoRefine is also used for data cleaning and curation. Multiple functionalities such as faceted exploration of tables and bulk editing. A powerful expression language (GREL) allowing complex string manipulation on individual values, provides easy finding and fixing of errors in the input data.

Data Curation and Cleaning Examples

The following screenshots show some of the most commonly used data cleaning functionalities of OpenRefine. here are some examples:

Facets

Faceted exploration allows to see at a glance all unique values in a column, and to filter based upon them.

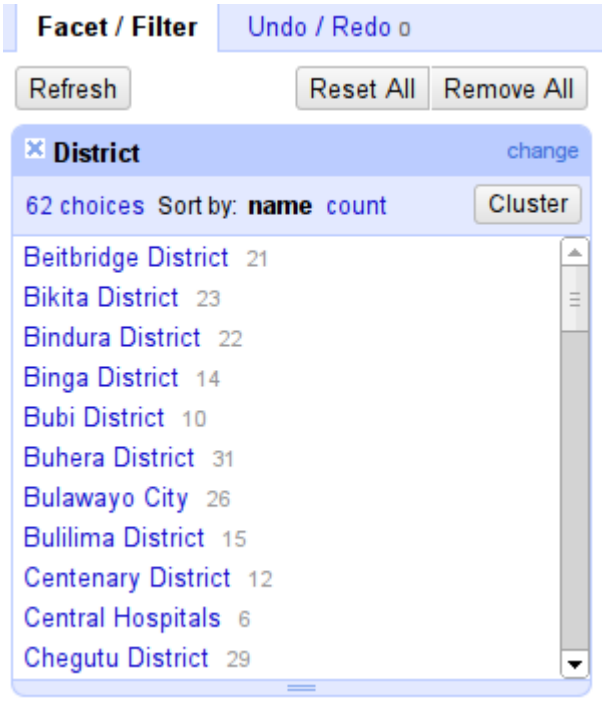

Figure 4: Example for data curation – faceted search

Common String Transformations

Predefined transformations exist for most of the common tasks, such as whitespace trimming, switching data types, case normalisation and others.

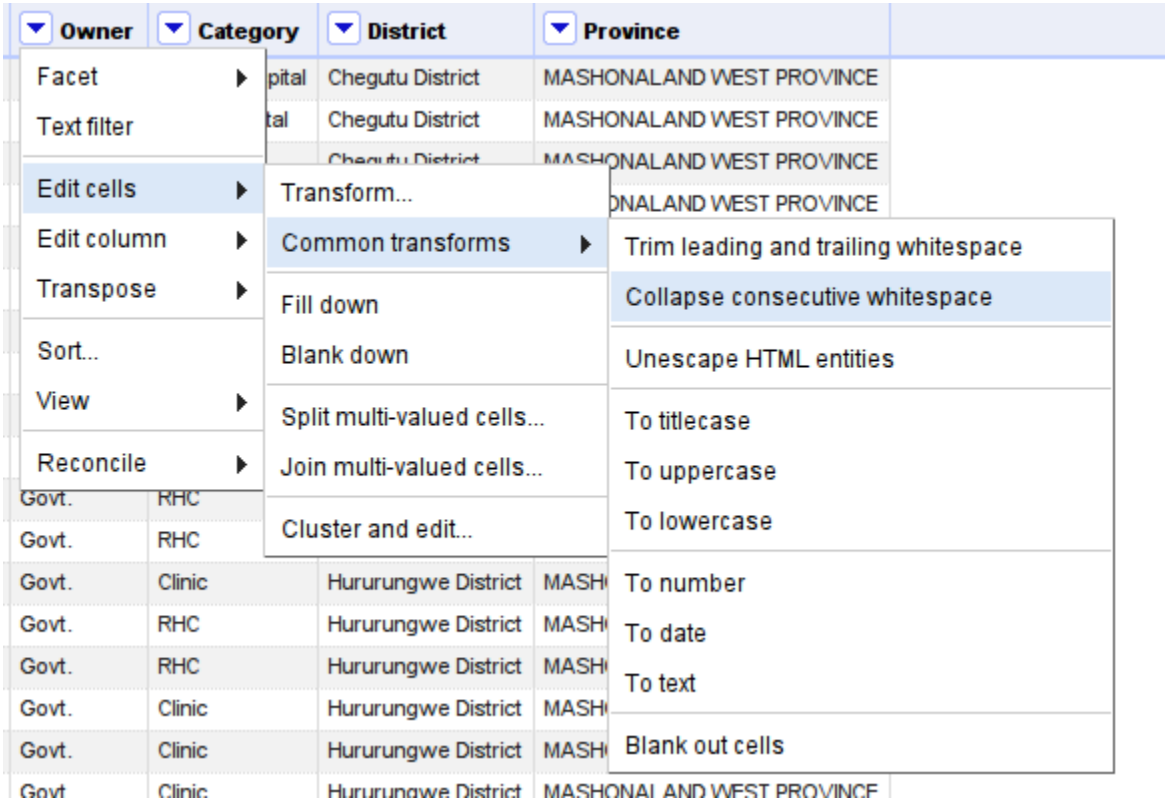

Figure 5: Example for data curation – text transformation

Text Clustering

Text clustering allows folding similar values in a supervised manner thus reducing noise in the data.

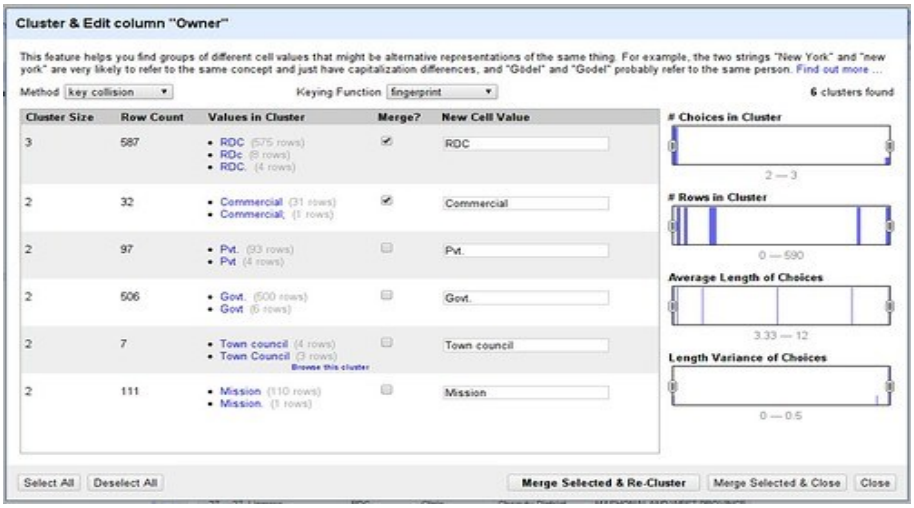

Figure 6: Example for data curation – text clustering

3.2 Reconciliation Service

Besides data cleaning, OntoRefine is also a reconciliation client, following the "Reconciliation API" protocol[6]. Reconciliation or automatic entity matching is a semi-automated process of matching text names to database IDs (keys).

The following screenshot shows the process applied to Entities of type "person". The "itemLabel'' column initially contains names of people as strings. The reconciliation service has searched for these names on a remote endpoint (in this case Wikidata[7]) and proposes several candidates for each string.

[6] https://reconciliation-api.github.io/specs/latest/

[7] https://www.wikidata.org/wiki/Wikidata:Main_Page

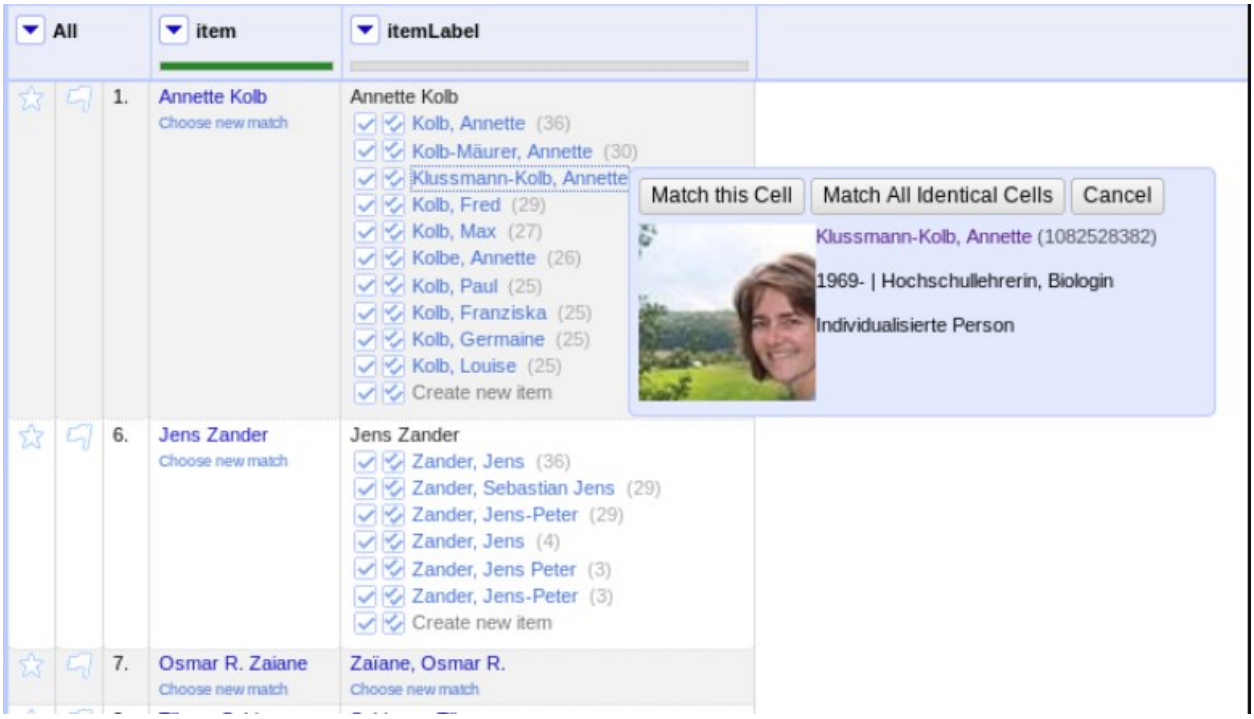

Figure 7: Example for reconciliation service

FSM Reconciliation Service

Besides using readily available public reconciliation endpoints (e.g., Wikidata) we are also developing a framework for setting up and running project specific endpoints corresponding to Food Safety relevant objects. These will be entities such as products, fertilizers, incident categories, and certificate types and families. The service runs on top of the public subset of the semantic data collected and streamlines the process of mapping and submitting new datasets to the FSM.

Version 2 of the Data Services already provides access to several public or specialized reconciliation endpoints. The specialized ones are added directly into the deployment stack of the FSM Platform as additional services that are initialized, when the whole environment is started. The full set of reconciliation endpoints will be provided with the final version of the Data Services at M36.

Public Reconciliation Services

The reconciliation services for locations, people and organizations are provided as public resources and can be used by anyone. All services are based on specific data extracted from Wikidata. The benefit of having these services is in their efficiency and speed, when given information is reconciled against them. Additionally, they are providing better scoring for the

actual match in some cases, when there is an ambiguity. The URL for accessing the services are as follows:

- locations - https://reconcile.ontotext.com/locations
- people - https://reconcile.ontotext.com/people
- organizations - https://reconcile.ontotext.com/organizations

All services will be registered in the OntoRefine tool so that they could be easily used by the clients or data experts during the data setup and automatic dataset upload afterwards. This will be done on deployment level, when the components of the FSM Platform are initialized.

There are some limitations set in place for these services like query timeouts, memory consumption per query, etc., as a means for protection from unintended usage and as a way to guarantee their availability.

Some of the services are even registered in the Reconciliation service test bench8 as we expect all of them to be included soon. This bench provides a table of known publicly available reconciliation services. It provides information about the supported functionalities by the different services such as: suggest, preview or extension of the entities. Any of the services described there could be included as part of the resources that OntoRefine can use for the reconciliation process.

Specialized Reconciliation Services

There is an infrastructure set in place, which allows for different and specific reconciliation services/endpoints to be registered as a part of the FSM Platform. It consists of the following components, which are included in the deployment stack of the Platform:

- GraphDB - used as the main data store, where most of the datasets will be stored and maintained, as required.
- Elasticsearch - used for indexing of the data required for the reconciliation process. It provides easy and quick access to data, plus various aggregations and rankings.
- Conciliator - the service which actually does the reconciliation job, it uses Elasticsearch as data provider for the entity matching.

Currently there is only one reconciliation endpoint that is registered, which is related to food products. The data for it is based on LanguaL dataset, which ensures compliance with the GS1 standards. The dataset is imported in the GraphDB repository, which synchronizes it with the

[8] https://reconciliation-api.github.io/testbench/

Elasticsearch by using specialized GraphDB-Elasticsearch Connector. The connector is also responsible for the Elasticsearch indexes management.

The Conciliator service uses the indexed data as a source for the reconciliation process, which is basically matching of textual values from a dataset field (e.g., column) to specific entities, each with a unique identifier. The service provides its functionality through a specific endpoint, which is described in a configuration file. The file contains JSON describing the path of the endpoint that should be used for this specific data. It also contains mapping describing the properties that should be included for the matching.

When new types of datasets are added to the list supported by the FSM Platform, the corresponding reconciliation endpoints required will be added to the deployed components and OntoRefine tool so that they could be used when users process datasets of these types.

The deployment stack, all required configurations and documentation can be explored in the main Gitlab Repository[9].

[9] https://gitlab.ubitech.eu/thefsm/thefsm-integrated-platform

4. DATA PUBLISHING SERVICES

4.1 GraphQL API

GraphQL[10] is a set of a query-by-example type language allowing users to make queries and server-side tools providing data or executing data mutations in correspondence with the user queries. The queries are passed to the services (API) as plain texts. The responses are JSON structures, following the query structure. The field names and data types are as per preliminary defined data description (also known as schema) and as a rule the field names are quite informative making the queries and results also human readable.

More detailed information about the language, its syntax and semantics, etc. can be found at http://graphql.org.

Why GraphQL

GraphQL is not closely tied to any specific data storage, data source or programming language. It provides the abstraction of data source moving the focus from databases, web services, data types, class inter-relations, indices and any other specifics and details to the data themselves and provides users with an unified interface for dealing with data. The only thing the user has to know is that there are such data classes or concepts with their hierarchy, maybe relations to some other concepts and what are the names of the fields he/she is interested in. Where the data (objects) of such classes are stored and how they can be accessed is up to the system and is generally completely hidden from the users.

Each GraphQL service (also GraphQL endpoint) has a description of data classes (concepts) which it manipulates, known as schema. The schema consists of description of fields, their types and the names of the functions, which can be called in order the field (property) values to be accessed. Additionally, behind the scenes, there are also implementations of these functions in some programming languages, packed as libraries or services and closely connected to the specific data storage and its functionality.

The specific implementation of a GraphQL service (GraphQL endpoint) depends on the storage system of the data which it manipulates, however, the schema is public, and the user can see the available data objects description so that to be able to construct the appropriate GraphQL queries.

[10] https://graphql.org/

Since the real organization of data is hidden, it is completely valid for some predefined queries to be delivered as data classes and data to be accessed by the users in a straightforward manner as if they exist as physical data objects.

This way the GraphQL endpoint becomes an abstraction layer between the data and business logic providing consistency, isolation and standardization.

4.2 Data Federation

The proposed Apollo Federation Service11 as a part of the Data Services module of TheFSM Platform behaves as a GraphQL endpoint.

It allows gathering data from different GraphQL endpoints as per definition in its schema, combining their schemes and providing users with a solid holistic data source.

Specific predefined queries can be defined and added to its combined data scheme so that the users can be supported in their everyday activities without the need of using very complex queries.

In the context of TheFSM Data Services the usage of GraphQL allows development of a standardized way for requesting data and data publishing.

The main GraphQL endpoint is the semantic service which uses the data stored in the GraphDB semantic repository. Its schema is defined in terms of Semantic Objects Modelling Language12 (SOML). The service can be queried using the classes and field names from the schema. It is provided as a web service providing REST API and can be called separately or via the Apollo Federation Service.

4.3 Ontotext Platform Workbench

This component is a user interface and is a kind of playground, where data experts (e.g. data engineers) can explore the semantic schema and view different data classes provided by The FSM platform.

[11] https://www.apollographql.com/docs/federation/

[12] https://platform.ontotext.com/3.0/soml/index.html

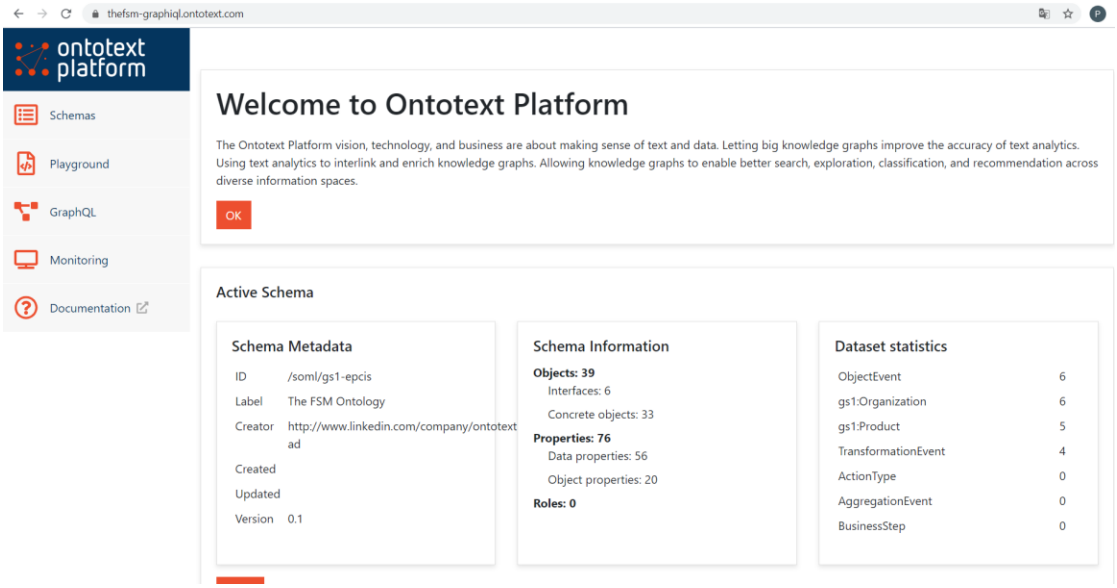

Figure 8: Ontotext Platform – Welcome view

It is designed as an entry point for schema management, monitoring and an interface, where users can try queries and explore their results.

The Schemas tab allows users to switch between loaded schemas, to load new ones and in general to manage the used schemas by the workbench.

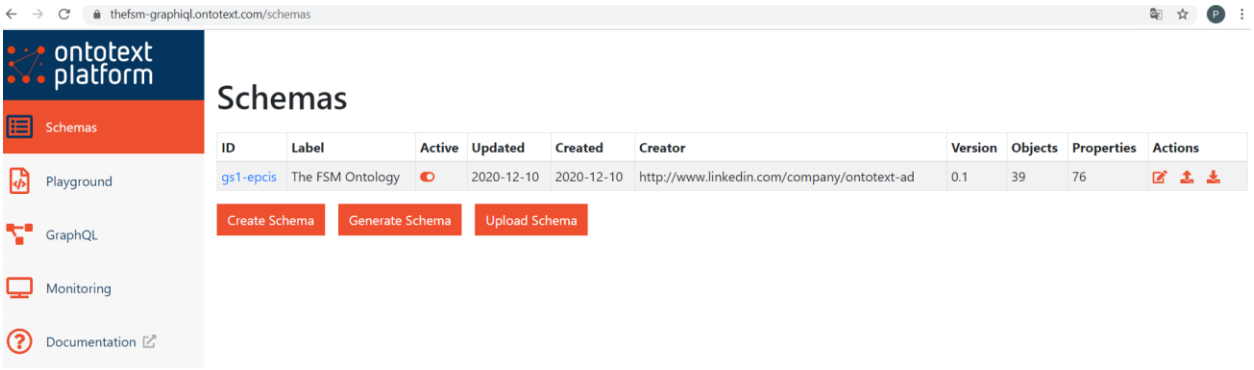

Figure 9: Ontotext Platform - Schemas

The Playground tab provides functionality for schema modification and validation. It acts as a regular text editor for usage with codes.

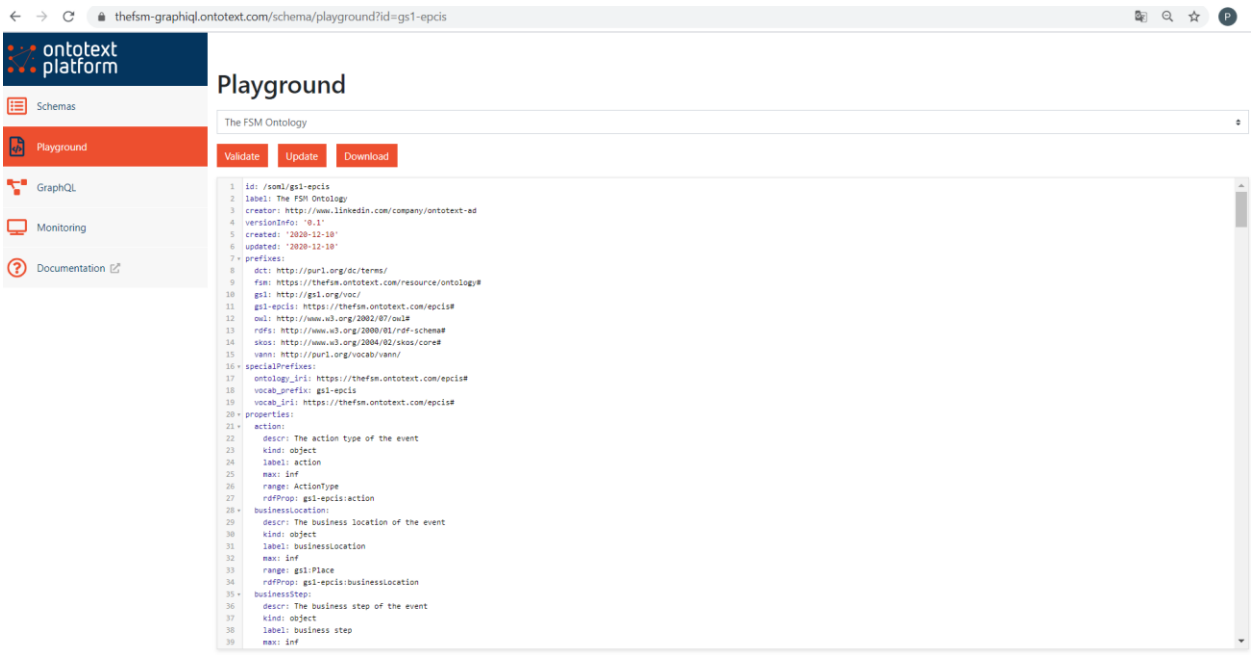

Figure 10: Ontotext Platform - Playground

The Monitoring tab shows the current state of the platform components, A status line is present on the top of the screen just below the screen title. Its background color can be green, yellow or red depending on the state of the platform as a whole. This allows one sight checks - if the colour is green - all is OK and there is no need for further investigation.

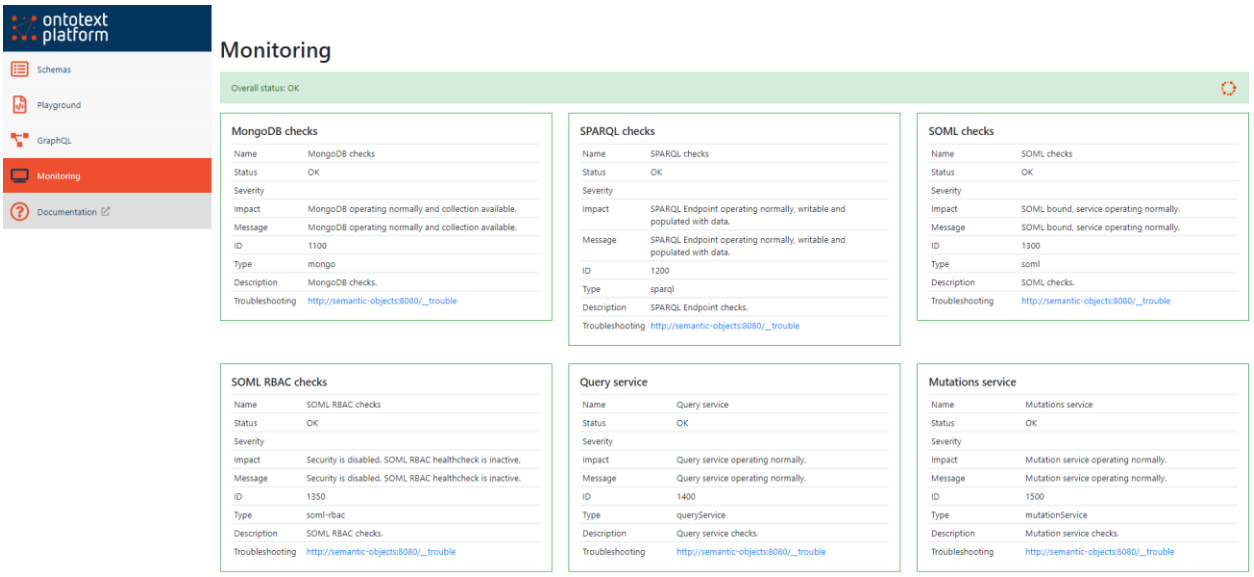

Figure 11: Ontotext Platform - Monitoring

One of the most important features of the platform workbench is in the tab GraphQL. It provides a user interface for writing and execution of GraphQL queries. That is useful for creation and testing of queries which are to be included in the code for querying the services programmatically.

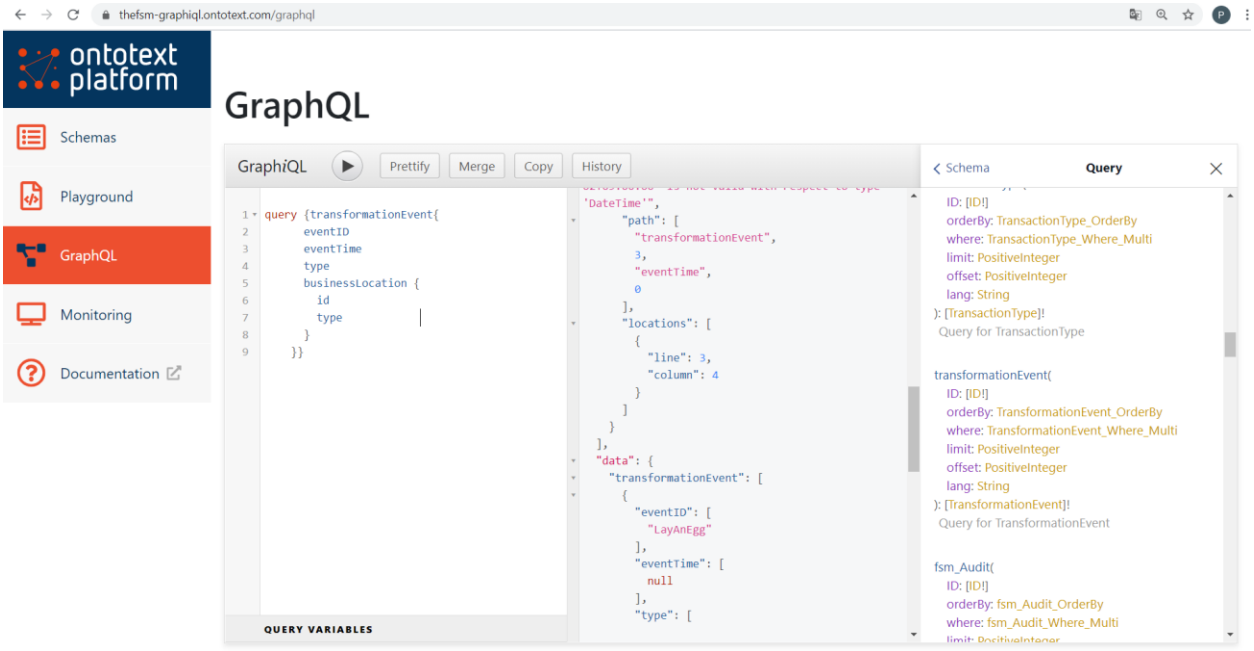

Figure 12: GraphQL

There are three user interface panes. Experts can write GraphQL queries in the left pane, using the data definitions in the rightmost pane. There are also context related hints about data classes and property (field) names. The results of running a query are displayed in the central area.

The query from the above example returns the list of transformation events from the loaded sample data objects in the semantic repository. As can be seen in the illustration, the results are returned in JSON format following the structure of the query.

There is also rich documentation content in the Documentation tab, where in-depth information about the modules of the platform, their usage and interconnections are available.

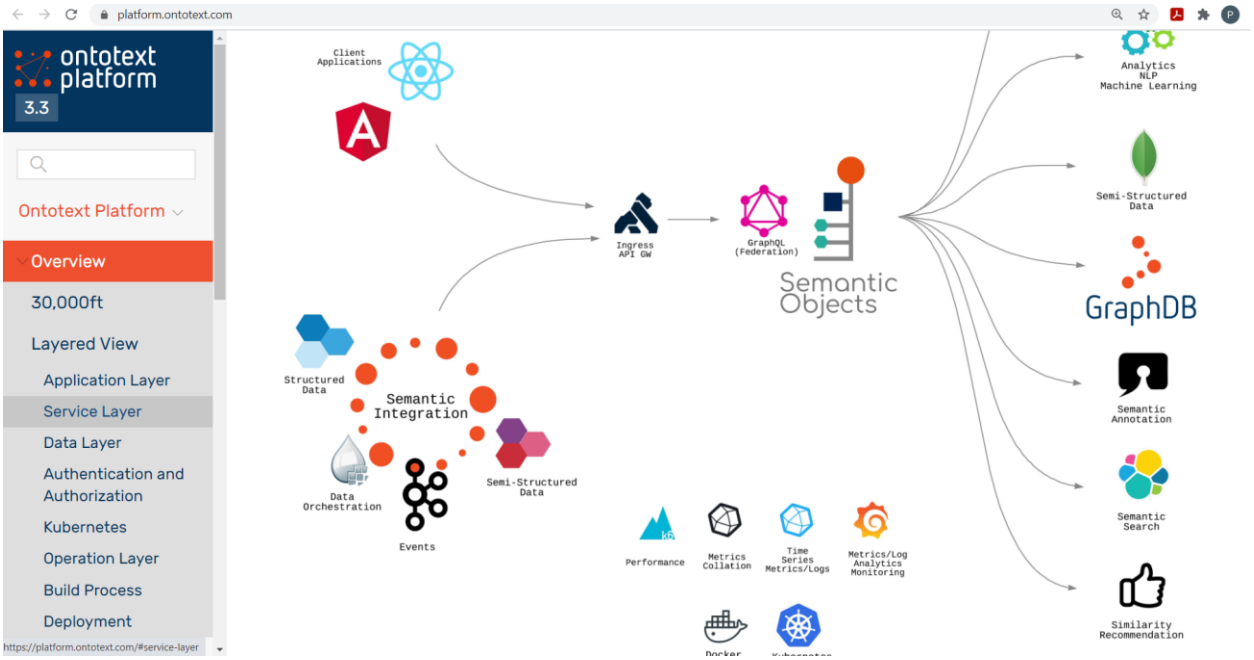

Figure 13: Ontotext Platform - overview

4.4 GraphQL endpoint

The same result can be obtained by an application when sending the following HTTP request to the service.

```
curl -X POST https://thefsm-graphql.ontotext.com/graphql \
        -H "Content-type: application/json"                \
        --data-binary '{"query":                                \
                "query {transformationEvent{                    \
                        eventID                                         \
                        eventTime                                       \
                        type                                            \
                        businessLocation {                      \
                                id                                      \
                                type                                    \
                        }                                               \
                }}"}'                                                   \
```

The above example is provided for the cUrl command line tool. Every developer can construct the corresponding HTTP requests for the program language he uses following the above pattern.

In other words, when a developer tests a desired query using the GraphQL tab of the Platform workbench, the text of the query is placed in the "query" field of the JSON, which the program will send as a body of the http request to the GraphQL endpoint.

Important remark: The current versions of the data services are running on Sirma AI's internal infrastructure, and the server address may be changed for the final implementation of the Data Services (e.g., these might be hosted on AWS infrastructure[13]).

Other usable API calls

curl -X GET https://thefsm-graphql.ontotext.com/soml - returns the schema of the GraphQL endpoint

curl -X GET https://thefsm-graphql.ontotext.com/__gtg - returns "Good to Go" status, e.g. {"gtg":"OK"}

curl -X GET https://thefsm-graphql.ontotext.com/__health returns more detail status information of the Ontotext Platform and its components

4.5 Data Publishing Workflow

The FSM project is data centric. All the user stories include data transfers from one user to another and most of them require data enrichment and disambiguation against Linked Open Data14 sources, datasets from other users and mapping to The FSM data model (described in D2.1).

The Data Services provide unified data access via GraphQL queries, federating among the various data sources.

From the viewpoint of data persistence, there are two major super-types of data:

- **Stored static data** – which are not expected to be changed during the lifecycle of the project and are used frequently. They will be stored in the Secure data storage. All mappings, enrichments, etc. will be performed on the stage of initial data transformation before storing. Such datasets are:

13 https://aws.amazon.com/about-aws/global-infrastructure/#:~:text=The%20AWS%20Global%20Infrastructure%20gives,the%20AWS%20Regions%20and%20AZ's.

14 https://lod-cloud.net/

  - ontologies, vocabularies, etc.
  - public static datasets which do not require frequent updates.
  - static published datasets from specific users.
- **Realtime data** - this data will be processed, enriched, and mapped in real-time when requested by customer queries. Such data types are:
  - Data provided by various applications,
  - LOD,
  - Data with restricted access, which depends on the querying end user credentials. Such data cannot be imported into the system.
  - Published data from users via providing access to their internal systems

Data Services also give a standardized way for publishing data. The users of the Platform can be broadly divided to two major types:

- Data consumers - users who consume data from the platform using provided functionality,
- Data providers - parties that provide their data to be used by the consumers via the platform functionality.

Any dataset published or processed on the FSM Platform must be described and its structure must be added to the platform data schema in order to be accessible by the Semantic Objects service. This means that any single data source requires some effort in order to be integrated with the other part of the platform.

As already mentioned in the case of static data, which will be stored inside the platform, all necessary enrichments, mappings to ontology concepts, disambiguation, etc. are performed at the initial processing time. The data structure is evaluated, and data descriptions are prepared. The specific data schema descriptors are created and added to the semantic schema of the platform. The data itself is processed, mapped and linked to the concepts of the ontology, LOD, etc. The result of the process is a RDF representation of data, subsequently added to the repository.

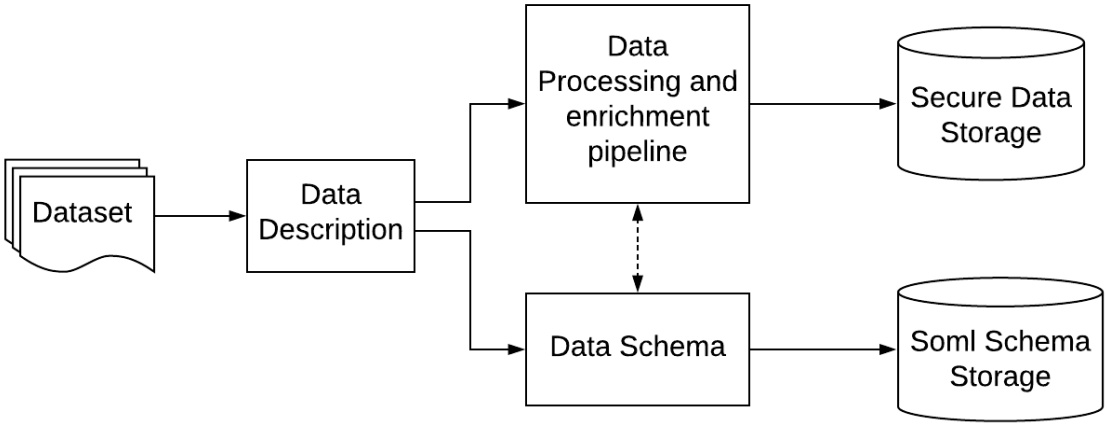

Figure 14: Static data publishing workflow

In case of a real-time data sources, the description of the provided API (if it is not federation-ready GraphQL endpoint) is studied. Data Schema extensions are created and added to the federation service schema. If the provided API is from a GraphQL federation-ready endpoint, only its data description is added to the federation service schema.

However, in the common case a GraphQL endpoint must be implemented as a data source wrapper in order the functionalities for accessing data from the external data source to be presented in a standard GraphQL manner and ready for use by the Apollo Federation service. It is added to the set of microservices running on the infrastructure.

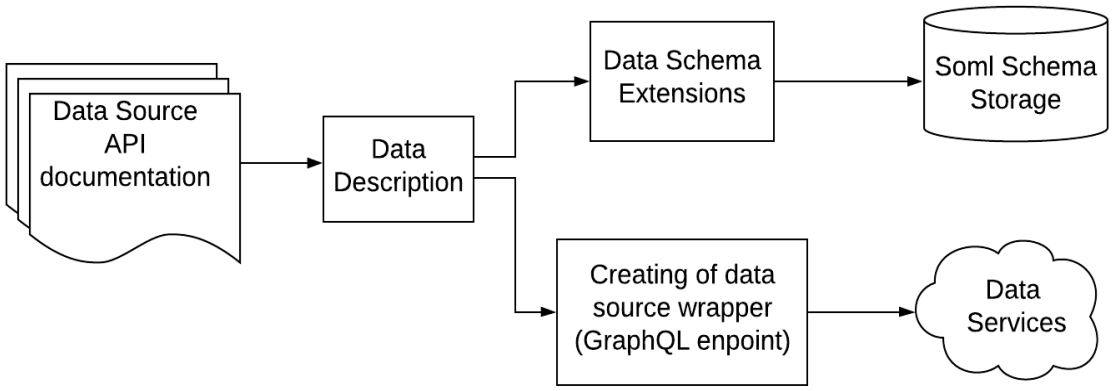

Figure 15: Realtime data publishing workflow

As it can be seen, adding a new data source, especially a data source with dynamic content implies some effort - data analysis, writing schema extensions, and potentially programming - creating the corresponding wrapper microservice or the processing pipeline, accordingly.

As described below, adding a static dataset can be semi-automated, reducing the necessary manual effort to a few hours.

4.6 Dynamic publishing of domain specific ontologies

The principle is as follows. The implementation resides in the project's github[15] . Master vocabulary data resides in a shared Google Sheet. Google Sheets is used as a tool for handling many of the difficulties of remote collaboration:

- It relieves us of the responsibility of monitoring data integrity and data versioning.
- It maintains a complete log of all user actions.
- It's simple to revert to the previous version or assign an error to a user.
- This allows us to be sufficiently agile while ensuring that the final data meets the rigorous consistency required to generate RDF.
- It has a collaboration mechanism with comments and tasks on specific cells allowing fast and efficient remote communication on specific data elements.
- Linking with cells and ranges also facilitates communication.

Google sheet data is consumed over HTTP in a simple CSV tabular format and is converted to RDF using a custom SPARQL query and the TARQL[16] tool. TARQL (or tabular sparql) allows us to define a tabular-to RDF mapping in a SPAQRL query. This allows us to maintain the mappings in a fully declarative way by having the various elements independent of each other and in version control (GitHub).

The generated ontology in RDF is also committed to the GitHub repository and can be directly consumed via HTTP.

4.7 Data validation using RDF Shapes

Another aspect of publishing data through TheFSM Platform is assuring their consistency before they become available to data consumers.

Since the problem of data validation before integration is very common worldwide, a special experts group of W3C was constituted. Its name is RDF Data Shapes Working Group[17.] It created Shapes Constraint Language[18] (SHACL) as a recommendation for description of data integration

[15] https://github.com/OriginTrail/epcis-erm/tree/food-auth/vocab

[16] https://tarql.github.io/

[17] https://www.w3.org/2014/data-shapes/

[18] https://www.w3.org/TR/shacl/

and validity rules (also called shapes). More details about the language and ways of description of constraints can be found here[19].

SHACL validation is incorporated in the latest version of GraphDB. Detailed information about setup and usage is available under the Documentation Tab, Usage section, SHACL validation.

In summary, this functionality provides a tool for validating graphs against a predefined set of rules, in the form of shapes or other constructs, which are loaded in as separate RDF graphs. They are also called shape graphs.

Formally, each shape is an IRI or a blank node "s" that fulfils at least one of the following conditions in the shapes graph:

- "s" type is one of sh:NodeShape or sh:PropertyShape.
- "s" is the subject of a triple that has one of the following predicates: sh:targetNode, sh:targetClass, sh:targetObjectsOf or sh:targetSubjectsOf.
- "s" is the subject of a triple that has a parametric predicate.
- "s" is a value of a shape-expecting, non-list-taking parameter such as sh:node, or a member of a SHACL list that is a value of a shape-expecting and list-taking parameter such as sh:or.

Every SHACL repository contains a named graph with the reserved name http://rdf4j.org/schema/rdf4j#SHACLShapeGraph, where the validation rules are inserted.

Users that intend to use SHACL validation in some repository must specify it on the creation stage. This cannot be added afterwards. If the workbench user interface is used, some additional checkboxes appear. Their detailed description and recommended values are given in the online documentation.

The SHACL rules must be loaded to the http://rdf4j.org/schema/rdf4j#SHACLShapeGraph named graph using one of the following methods:

- Workbench Import - as a file or RDF triples in the immediate window
- Import Statements in SPARQL window
- The GraphDB REST API
- When new data is inserted into the repository and SHACL validation is on, the RDF triples compliance to the rules are evaluated and if there is violation, an exception

[19] https://www.w3.org/TR/shacl/

is thrown. Information about the violation is provided. More detailed information can be logged, depending on the repository settings.

- The supported SHACL constraints are listed in detail in the official documentation20 of GraphDB, also available online under the Documentation tab - Usage->SHACL validation.

4.8 Ontotext Platform and SHACL

In recent versions of the Ontotext Platform a functionality which allows generation of SHACL scheme, based on the defined SOML, is available. This makes the integration of the data validation functionality much easier and more straightforward. More information about how the functionality works, how to configure it, can be found in the Static Validations[21] section in the Ontotext Platform documentation.

[20] https://graphdb.ontotext.com/documentation/9.5/enterprise/shacl-validation.html

[21] https://platform.ontotext.com/semantic-objects/semantic-objects.html#static-validators

5. AUTOMATED DATA PROCESSING SERVICE

The Automated Data Processing Service combines in it the data ingestion and data curation functionality and automates data processing so that end-users of the FSM Platform can easily and without prior data engineering expertise perform cleaning, semantic enrichment and RDFization of their data. This service is the main component that will serve as a communication relay between all services involved in data processing (GraphDB, OntoRefine, Elasticsearch, Conciliator service) and the FSM Platform.

The main idea behind the service is to manage the entire process of cleaning, transforming and semantically enriching the data provided to the FSM Platform with minimal input from the users. This minimizes the requirements to the users to have knowledge about all required tools and semantic technologies that are used for the process.

Ideally, the customers of the FSM Platform will not have a direct contact with the service, because the service itself will be a part of the backend, which serves the intuitive and user-friendly interface. Visual interfaces will guide users to the things that have to be provided and the steps to be followed in order to process their data correctly.

Currently we successfully managed to limit the required input from the user to only two things, one being the actual dataset and the other the type of the dataset for the process to know how to handle the data and which transformation script should be applied to it.

5.1 Data Processing Pipelines

The functionalities, which the service provides are basically processing pipelines that are using the different components to build up an automated closed process, which gets a dataset and few additional parameters and produces a semantically enriched dataset.

In order to automate the data ingestion and data curation processes, the required part for the pipelines is the file with the operations that should be applied to the dataset, which is described in the sections about the data ingestion and curation services. As mentioned, the files will be prepared by a data specialist for each type of dataset that the FSM Platform should support. The scripts will be stored so that they could be reused, when a specific type of dataset is uploaded in the Platform.

The pipelines are exposed by the service via REST endpoints, which makes them flexible in their usage, because they can be embedded in user interfaces or called by different systems and applications.

The process that is automated by the pipelines, given that the client uploads the dataset and transformation script, is:

1. Creation of a new project in the OntoRefine tool. This step is required in order to clean, transform, reconcile, etc., the dataset.
2. Applying the transformation script that was prepared for this specific type of dataset. It may include deduplication, re-formatting or re-wording of a specific value in the data, cleaning of unnecessary data, reconciliation of specific text values in order to map them to specific identifiers or to extract additional information about them, RDF mapping and so on.
3. Exporting the new data in a specific format that will be useful to the clients.
4. (Optionally) Import the RDF data into GraphDB using the provided mapping in the transformation script.

Most of the steps are executed in OntoRefine through HTTP calls using a specific library created for that purpose, called ontorefine-client22. It is a public library, which provides quick access to the functionalities of the refine tool without the need of an user interface and it can connect to any OntoRefine tool on any GraphDB instance. The library itself is intended to provide a middleware between the OntoRefine tool and any other system or application that needs to use some kind of functionality that the tool provides.

There are two pipelines that are currently available in the service. They are classified by the format of the data that they return:

- CSV Pipeline - as its name suggests, it returns files in CSV format. It can accept as input CSV or JSON files.
- JSON-LD Pipeline - returns files in JSON-LD format and also can consume CSV or JSON files.

More technical information about the pipelines is provided in the Gitlab integrated platform repository[23] managed by our partners from UBITECH, and particularly in the README24 file in the semantic-enrichment sub-directory.

[22] https://github.com/Ontotext-AD/ontorefine-client

[23] https://gitlab.ubitech.eu/thefsm/thefsm-integrated-platform

[24] https://gitlab.ubitech.eu/thefsm/thefsm-integrated-platform/-/blob/master/semantic-enrichment/README.md

5.2 Simple Pipeline Example

Below is presented a simple example of the CSV pipeline with a sample of dataset and script. The dataset that is going to be used is as follows:

ID	Name	Type	Location	Description
1	European anchovy	Fish	Black Sea	The European anchovy (Engraulis encrasicolus) is a forage fish somewhat related to the herring.
2	European eel	Fish	Black Sea	The European eel (Anguilla anguilla)[2] is a species of eel, a snake-like, catadromous fish.

The process in the pipeline will use the following transformation script to remove the description as it is not needed and extract some useful information from the other columns by reconciling the values against some of the Wikidata services.

```
[
  {
    "op": "core/column-removal",
    "columnName": "Description",
    "description": "Remove column Description"
  },
  {
    "op": "core/recon",
    "engineConfig": {
      "facets": [],
      "mode": "row-based"
    },
    "columnName": "Location",
```

```
  "config": {
    "mode": "standard-service",
    "service": "https://wikidata.reconci.link/en/api",
    "identifierSpace": "http://www.wikidata.org/entity/",
    "schemaSpace": "http://www.wikidata.org/prop/direct/",
    "type": {
      "id": "Q986177",
      "name": "mediterranean sea"
    },
    "autoMatch": true,
    "columnDetails": [],
    "limit": 0
  },
  "description": "Reconcile cells in column Location to type Q986177"
},
{
  "op": "core/extend-reconciled-data",
  "engineConfig": {
    "facets": [],
    "mode": "row-based"
  },
  "baseColumnName": "Location",
  "endpoint": "https://wikidata.reconci.link/en/api",
  "identifierSpace": "http://www.wikidata.org/entity/",
  "schemaSpace": "http://www.wikidata.org/prop/direct/",
  "extension": {
    "properties": [
      {
        "id": "P205",
        "name": "basin country",
        "settings": {
          "limit": "0",
          "rank": "best",
          "references": "any",
          "count": "on"
        }
      },
      {
        "id": "P2046",
        "name": "area"
      }
    ]
```

```
  },
  "columnInsertIndex": 4,
  "description": "Extend data at index 4 based on column Location"
 }
]
```

After the above transformation script is applied, the output result from the pipeline will be as follows:

ID	Name	Type	Location	basin country	area
1	European anchovy	Fish	Black Sea	6	436402
2	European eel	Fish	Black Sea	6	436402

The figure below shows the execution of the pipeline via HTTP call. The environment is started locally for example purposes. Here we pass the dataset and the transformation script as parameters to the POST request.

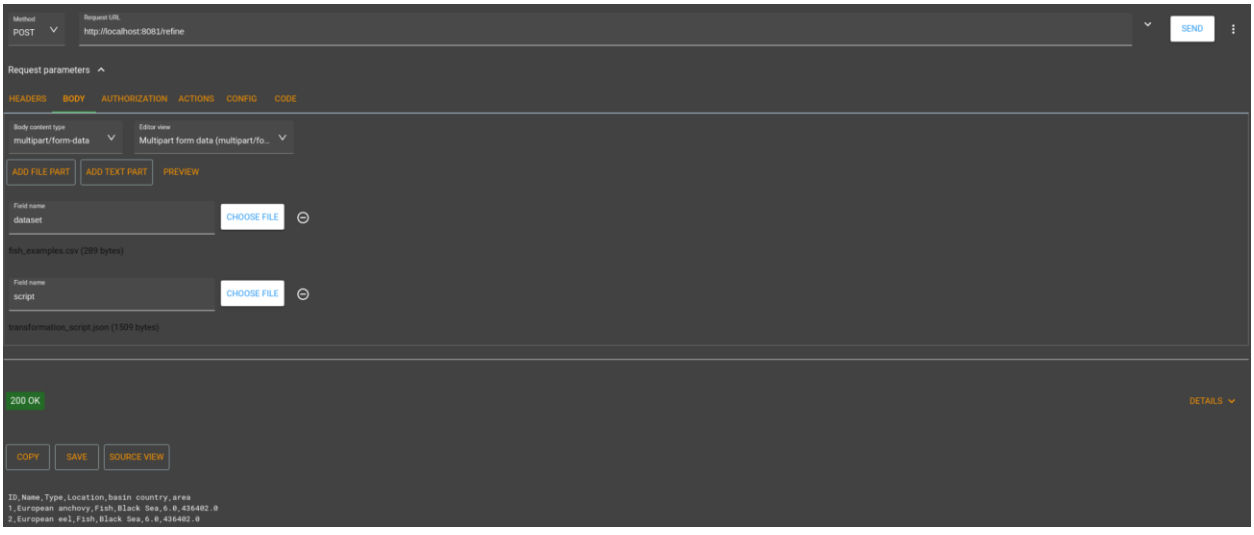

Figure 16: CSV Pipeline executed via Rest Client

To visualize the process and what is going on behind the scenes we are using a simple sequence diagram for the process.

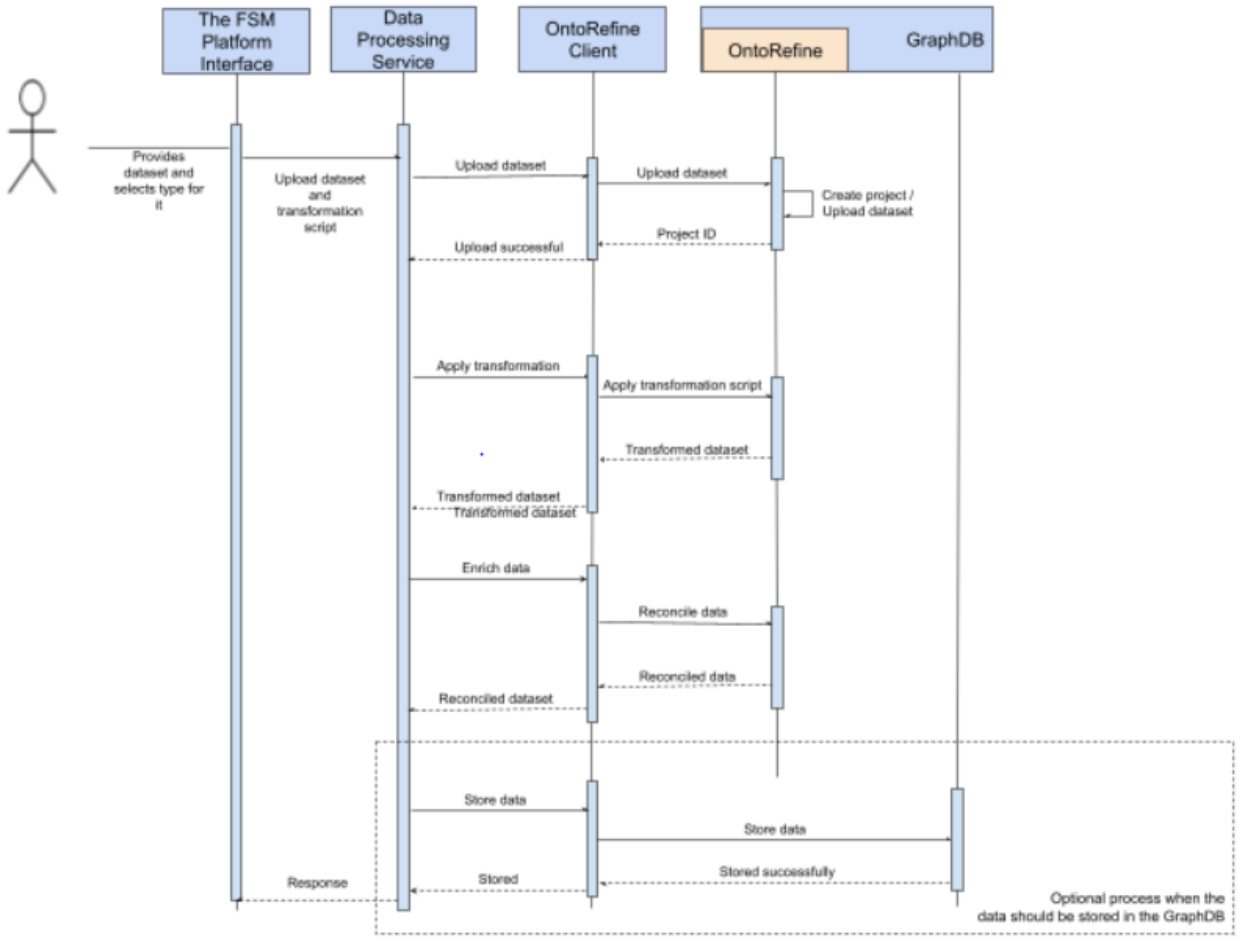

Figure 17: Simple sequence diagram of the CSV pipeline process

6. RISK ASSESSMENT AND PREDICTION SERVICES

In this section, we describe the services that Agroknow for the deployment of a platform that can collect, process, combine, enrich, and analyze heterogeneous, multilingual data using a wide range of data management, big data & AI technologies. On top of this process, the Risk Assessment and Prediction services process these data to offer assessment and prediction on the incident, product, and supplier levels.

6.1 Motivation for Risk Assessment and Predictions for Food Safety

Today, although we have good food safety standards, food safety systems, certifications and risk assessment strategies, recalls and border rejections are increasing: the number of recalls and border rejections reported by Authorities in 2019 was doubled compared to 2014.

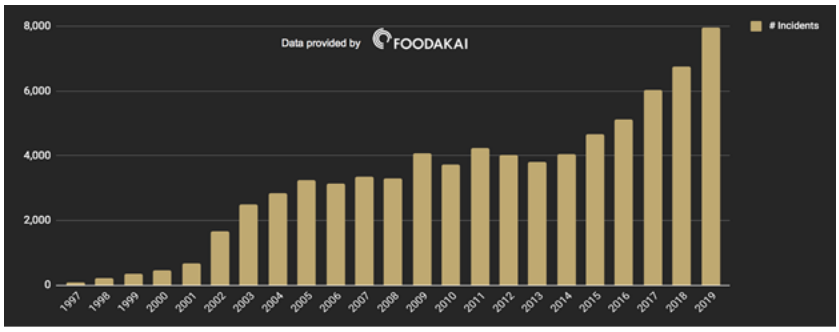

Figure 18: Number of recalls and border rejections reported by National Authorities

To this we need to add the COVID-19 situation, which created a bottleneck to the food industry. Under this new situation, the industry has to deal with a disturbed supply chain, companies need to immediately change suppliers, they are looking for shorter supply chains, less budget is available and due to the travel restrictions experts cannot conduct audits. This situation is moving us to exploring the possibilities for new risk assessment approaches that can utilize all the available data in the supply chain.

The Global Food Safety Initiative (GFSI) standards and regulations like Food Safety Modernization Act (FSMA) require detailed risk assessment of hazards for ingredients, finished products, processes and suppliers. According to the GFSI standards and FSMA, generic risk assessment for product categories and suppliers is no longer sufficient. The standards and new regulation require assessing risk for suppliers, for facilities and for all the materials used in each facility.

In addition to that, consumer trends for less processing and new ingredients and the increased complexity and globalization of the supply chain calls for a risk assessment approach that

- is using accurate data about reported issues in the global and company's supply chain
- is not static and is updated frequently because microbial, chemical, allergen and fraud risks are continuously evolving
- applies predictive models of pathogens growth, toxins and chemicals contaminants
- considers consumption and population data
- considers the processing steps applied to materials (cooking, partitioning, packaging, storing)
- concludes to recommendations on how to mitigate risks

Risk assessment is one of the most important tools to prevent recalls. However, today risk assessment is too often approached as a checklist compliance task. Not utilizing appropriate technology, data and good strategies to the fullest extent is poor risk management practice. It causes resources to be misdirected towards redundant activities or towards addressing events that are unlikely to occur. The risk assessment approaches followed in the industry are based on historical incidents and cannot really help experts to identify emerging and increasing risks.

The assessment and management of risk is challenging as it needs to consider all the stages of the supply chain and to use the data that is produced at each stage, namely production, incoming ingredients, processing & manufacturing, storage, distribution and services.

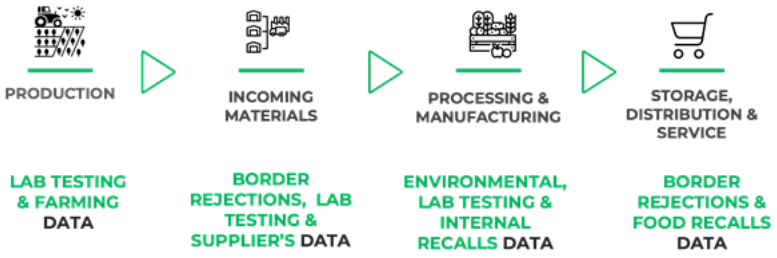

Figure 19: Stages of the supply chain and corresponding data that can be used for risk assessment

At the production level, laboratory testing data of the harvested agricultural products and farming data (cultivation practices, soil type, fertilizers, plant protection products applied) are generated and can be very helpful for the risk assessment in order to identify at an early stage the exceedances of chemicals and high levels of toxins. At the incoming materials level, data about border rejections, laboratory testing results and supplier's information including certificates, audits results, are generated. Data from this level is very helpful to identify risks for the ingredients and materials before getting them in the plant. At the processing level we have a lot of data including laboratory test results, environmental data and internal recalls data. At the storage,

distribution and service level we have data about border rejections, controls on the market and recalls. We should also add to the picture data that is highly relevant for all the stages such as commodity pricing data, country risk indicators and foodborne disease outbreaks.

From all these stages of the supply chain, a wealth of data and knowledge is generated about risks in foods due to better monitoring systems, testing technologies and research advances. An important challenge is how to integrate all this data in order to use it to assess and predict risk. The data is highly scattered, heterogenous, multilingual and disconnected. To integrate all this data a significant amount of time and specialized effort is needed, which lead to the creation of the Big Data Platform described in subsequent chapters.

Conclusively, the food industry needs to understand and predict food safety risk in order to switch from reaction to prevention. This requires combining new analytical tools, new methods and food safety data to enable a timely and proactive risk assessment approach.

6.2 Big Data Platform

The first and very important step towards a technology enhanced risk assessment is the collection and processing of all the scattered and heterogeneous data that are generated every single minute in our supply chain. To address this challenge, we have developed a big data processing workflow and a relevant Big Data Platform. The main goal of big data processing workflow is to take advantage of the computers' power in dealing with big data and combine it with the knowledge and experience of the food safety experts9. The development of the Big Data Platform for the food safety sector focused at targeting the needs of the food safety industry using Big Data processing, text mining, semantic and Artificial Intelligence technologies. The platform is responsible for collecting, processing, indexing and publishing heterogeneous food and agriculture data from a large variety of data sources. The platform was designed using microservice architecture, with different technology components handling different aspects of the data lifecycle. Automatically processed and enriched data are reviewed by the experts to enable high quality and harmonized data.

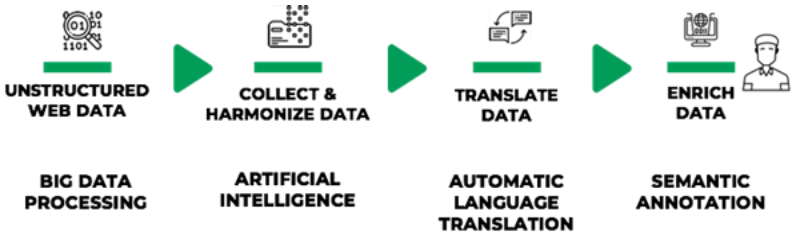

Figure 20: Data processing workflow of the big data platform

The data platform stores and processes millions of food safety data records that can be used for risk assessment and prediction. The data types that can be used for risk assessment and prediction are presented in the following table.

Data type	Number of records (as of March 2021)
Food recalls	78.466
Border rejections	333.842
Inspections	227.603
Laboratory testing results	102.187.114
Suppliers	702.578
Prices	415.670
Country risk and corruption indicators	3.519
News items	68.971
Maximum Residue Level Limit	141.594
Product brands	28.081
Sensor data	12.437.743

In the remainder of this section, we briefly look at the internals of the Big Data Platform, before delving deeper into the Risk Assessment and Prediction services.

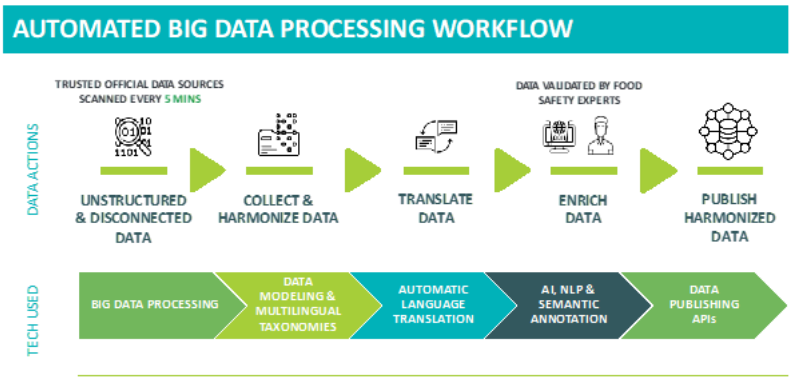

Figure 21: Big Data Platform Workflow

We apply big data technologies for collecting, processing and storing highly heterogeneous data. We use text mining and AI technologies to enrich and link the different data types. This big data and AI engine is supervised by data, technology and most important food safety experts to ensure data of high accuracy and quality.

The overall platform consists of various distinct layers each responsible for a different task. More specifically these layers are as follows:

- **Crawling** layer, responsible for the collection of raw data
- **Raw Data Persistence** layer, responsible for storing the collected/unprocessed raw data
- **Transformer** layer, responsible for transforming the raw data to the internal schema and assigns a unique identifier
- **Enrichment** layer, responsible with enriching the transformed data with important properties (e.g., product/hazard/origin)
- **Enrichment Data** layer, where all the tools/frameworks/technologies utilized by the enrichment layer reside
- **Curation** layer, the internal CMS used to manually annotate food safety data records
- **Data Persistence** layer, where all the production data records reside; utilized by our application layer & internal ETL workflows
- **Intelligence** layer, responsible for the calculation of all the predictive capabilities of our Data Platform & the respective risk assessment
- **Data API** layer, responsible with making accessible to external applications our data through JSON endpoints
- **NLP/Analytics** layer, utilized by our Data API to enhance its free text search capabilities,
- **Monitoring** layer, responsible with ensuring the uptime of all of Data Platform's storage engines and API endpoints,
- **Orchestration** layer, handling the execution of all our ETL workflows.

The data stack is implemented in a microservice architecture with various API endpoints and storage engines.

6.3 Architecture Overview

This section presents a brief overview of all layers, and then we delve deeper on the Intelligence and NLP/Analytics layers, which power important end-user dashboards in the FOODAKAI and

Food Inspector applications, with the aim of assessing and predicting food safety risks. The architecture of the Big Data & AI platform is presented in the following diagram.

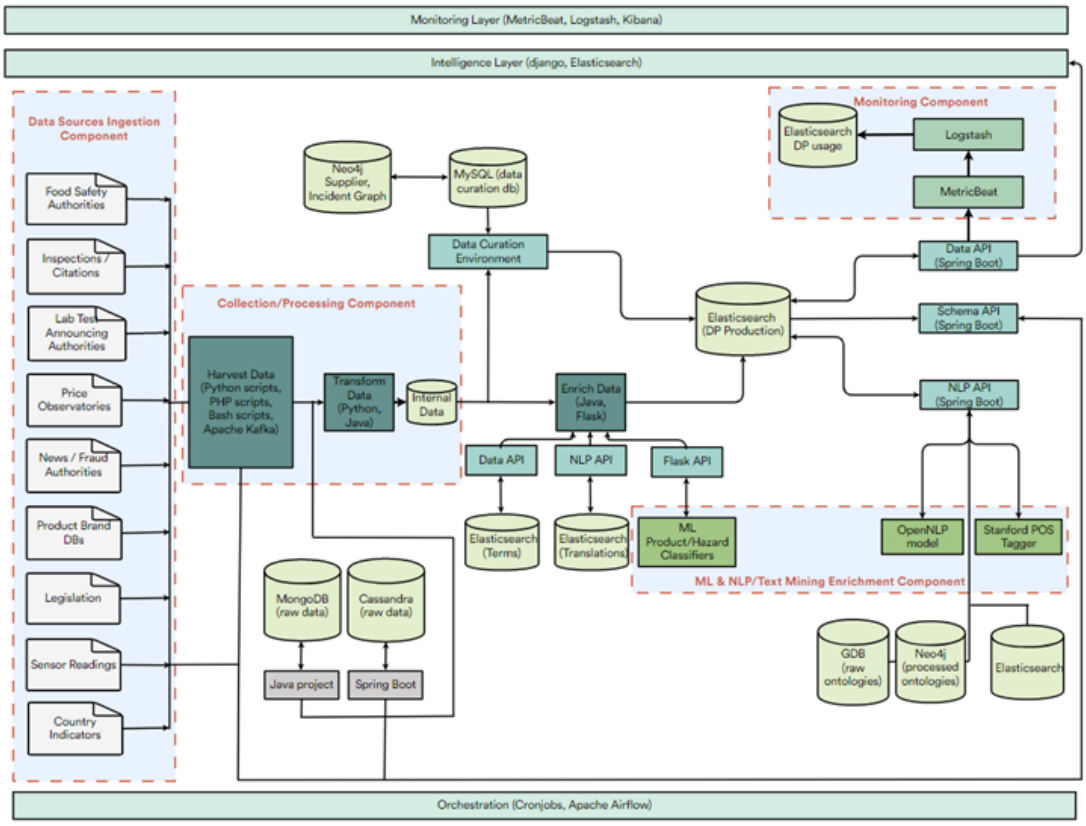

Figure 22: Architectural Diagram of Big Data Platform

Data Ingestion Components

Data ingestion is the first step in the Big Data Platform for building data pipelines and one of the most challenging tasks in Big Data processing. Big Data Ingestion involves connecting to several data sources, extracting the data, and detecting the changed data. It's about moving data from where it originated, into a system where it can be stored, processed and analysed. Furthermore, these several sources exist in different formats such as: Images, OLTP data from RDBMS, CSV and JSON files, etc. Therefore, a common challenge faced at this first phase is to ingest data at a reasonable speed and further process it efficiently so that data can be properly analysed to improve business decisions.

The data ingestion layer includes a set of crawling and scraping scripts for collecting information from the data sources. For each data source a different script was developed. These scripts vary in form. We utilize Scrapy, Python scripts, a custom Java project, as well as bash scripts to help in the collection of the tracked data sources. Regardless of their type, these collections scripts take

as input a starting URL and some rules and store the matching documents in the file system as output. Cron jobs are used to check every few mins if new notifications have been published on the web site of the agency/authority. For storing the fetched data, a NoSQL database, namely MongoDB is used.

Data Collection Components

In the Big Data Platform, each data source, depending on its format, is collected in a different way and might have a different form as mentioned in the above ingestion component. The raw data is harvested and then transformed using python and java code into an internal data model (schema). The main goal is for all the collected different types of data to have the same data structure.

Storage Components

The storage layer deals with the long-term storage and management of data handled by the platform. Its purpose is to consistently and reliably make the data available to the processing layer. The layer incorporates schema-less persistence technologies that do not pose processing overheads either when storing the data or retrieving them. Therefore, the storing and retrieving complexity is minimized.

Data Processing Components

These components are used throughout our data platform for our data to travel and communicate with other components.

It uses Flask, which is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program. As part of the Data platform, the Flask framework is used as a wrapper access layer on top of the Machine and Deep learning models trained for the classification of food recalls.

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source. As part of the Data Platform, Django is used for the development of incident prediction and risk assessment APIs. On the one hand, food incidents data is used to train a deep

learning prediction model to predict food incidents in the future. On the other hand, the main goal of the risk assessment module is to help Food Safety Experts to identify the ingredients with unacceptable hazard risk.

Monitoring Components

This component provides insights and visibility into the health and status of Data Platform's data clusters by tracking specific metrics in real time and sending alerts or notifications when readings exceed or fall below the set thresholds. Data collected from monitoring our data clusters can be stored, analyzed, and displayed in business intelligence and analytics dashboards and reports. The following software tools were used to monitor the health of the platform.

- Logstash is an open source data collection engine with real-time pipelining capabilities. Data flows through a Logstash pipeline in three stages: the input stage, the filter stage, and the output stage. Logstash can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all your data for diverse advanced downstream analytics and visualization use cases.
- Kibana is an open-source data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support. Also, it provides tight integration with Elasticsearch, which makes Kibana the default choice for visualizing data stored in Elasticsearch.
- Finally, we employ MetricBeat, i.e., the proposed tool for monitoring the scalability of Big Data Platform, over the Elastic stack to monitor and report our chosen metrics.

6.4 Intelligence Layer

The Data Platform includes an intelligence layer that is responsible for implementing the risk assessment and risk prediction algorithms using machine learning and deep learning methods. Risk assessment is one of the most important tools to prevent recalls. The assessment and management of risk is challenging as it needs to consider all the stages of the supply chain and to use the data that is produced at each stage, namely production, incoming ingredients, processing & manufacturing, storage, distribution and services. As an example, our services were applied on fruits and vegetables and the results are presented. Food risk predictions could be generated so that Food Safety and Quality Assurance teams can be prepared to handle the

situation before a major product recall happens and causes damage to consumer health, and company brand's trust and financials.

Risk assessment module and service

For the implementation of Risk Estimation we used a mathematical model that is based on the frequency of the incidents and the severity of the identified hazards in the incidents. The risk is estimated for all the possible product (ingredient, raw material) hazard pairs. Considering that the data platform includes more than 12.000 product (ingredients and materials) and more than 4.300 hazards the risk estimation should be conducted for a large number of pairs.

More specifically, the following risk estimation formula was used for the estimation of the risk based on global food safety incidents data that are collected and processed by the data platform

$Risk = Incident\ Severity \times Hazard\ Severity \times Probability$Risk = Incident Severity × Hazard Severity × Probability

where:

- **Incident Severity**: is the severity based on the type of the incident (e.g., food recall, food alert, border rejection). A table with the severity classification values mapped to incident types is presented later in this document.
- **Hazard Severity**: is the severity of the hazard based on the health impact. The hazard severity is classified as: 1, 2, 3 and follows the classification defined by organisations such as FDA. e.g listeria spp=3, pesticide=2. Detailed values for the hazard severity are presented later in this document.
- **Probability**: is the probability of the specific hazard to occur for the specific ingredient. The estimation of the probability is based on the frequency of the incidents that the FOODAKAI collects and processes every day.

In the following table we present the classification used for the incident severity. The classification of incidents is based on the types of incidents used in official data sources such as the Rapid Alert System for Food and Feed.

Incident Severity	Value
Food Alert/ Recall	5
Border rejection	4
Information	3

Information for attention	2
Information for follow up	1

For the estimation of the incident severity for each specific ingredient in a category we are using the mean value. For instance, if for aflatoxin and for pistachio there are 100 alerts, 120 border rejections and 10 information for attention the Severity = (100*5+120*4+10*2)/230 = 4.34 (the closest integer is used, i.e., 4).

In the following table we present the classification of the hazard severity for the most important hazard categories. In some cases, there are hazards that have different classification values than their category. For instance, in a category like Biological there are hazards such as coliforms and moulds that have a classification of 2 because the health impact is lower. On the other hand, there may be cases that a specific hazard may have higher classification value than its category.

Hazard Category	Value
Allergens	3
Biological	3
Chemical	2
Foreign bodies	2
Organoleptic	1
Fraud	2
Food contact materials	2

The classification that is used in FOODAKAI is based on the FDA's classification of hazards depicted in the following table.

Hazard Severity Classification	
3	This is a health hazard situation where there is a reasonable probability that the use of the product will cause serious, adverse health consequences or death.
2	This is a health hazard situation where there is a remote probability of adverse health consequences from the use of the product.

1	This is a situation where the use of the product will not cause adverse health consequences.

The probability can be estimated using 2 different approaches.

Approach 1: The first approach is estimated as the fraction of the number of incidents for the specific ingredient and hazard to the total number of incidents for the specific ingredient.

$$P = \frac{Number\ of\ incidents\ of\ a\ hazard\ for\ the\ specific\ ingredient}{Total\ number\ of\ incidents\ for\ the\ specific\ ingredient}$$

Approach 2: The second approach focuses on the probability a hazard for a specific ingredient to occur in the ingredient category that the ingredient belongs (e.g., a hazard to occur for okra that belongs to fruits and vegetables). It is estimated as the fraction of the # incidents for the specific product and hazard to the total number of the incidents for the specific category of the ingredient and the specific hazard.

$$P = \frac{Number\ of\ incidents\ of\ a\ hazard\ for\ the\ specific\ ingredient}{Total\ number\ of\ incidents\ of\ hazard\ for\ the\ specific\ ingredient\ category}$$

The ranking of the probability is done based on the values in the following table.

Probability	Rank	
> 75%	5	Frequent
75-50%	4	Possible
50%-25%	3	Occasional
10-25%	2	Small possibility
<10%	1	Unlikely

The estimated risk can be classified as Low, Medium and High as follows:

1- 15 = I - Low

16 - 24 = II - Low

25 - 48 = III - Medium

49 - 75 = IV - High

The potential values of the risk for the risk dimensions, namely hazard severity, incident severity and probability, are presented in the following matrix for a specific slice in which Hazard Severity equals to 3.

Hazard Severity=3					
	Probability				
Incident Severity	(1) Unlikely	(2) Small possibility	(3) Occasional	(4) Possible	(5) Frequent
I (Low)	I (3)	I (6)	I (9)	I (12)	I (15)
II	I (6)	I (12)	II (18)	II (24)	III (30)
III	I (9)	II (18)	III (27)	III (36)	III (45)
IV	I (12)	II (24)	III (36)	III (48)	IV (60)
V (High)	I (15)	III (30)	III (45)	IV (60)	IV (75)

Figure 23: Example of a risk matrix

Finally, the risk assessment service can be used in the following ways:

- a request to the Risk API to calculate the risk for a specific ingredient, origin and hazard. This option can be used every time that we need a risk estimation for a specific time point.
- a request to Risk API to generate a batch of risk time series. This option can be used to estimation the risk for a specific time period e.g., evolution of the risk during the last ten years

Risk Prediction module and service

A robust methodology is needed to generate predictions that are tailor-made to the supply chain of a company that is using a specific set of ingredients. The first step of such methodology (see **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.**) is to prepare the datasets that include data specific to a sector (instead of generic). The second step is to select the AI algorithm that can be applied to this data and fit well to the problem. Different parameters for the AI model need to be explored in terms of performance. The variations of the model need to be trained and tested. Several Artificial Intelligence algorithms may be applied to the data of food recalls and border rejections, both machine learning and deep learning algorithms. In this study we selected an algorithm that can work well in a time series prediction problem.

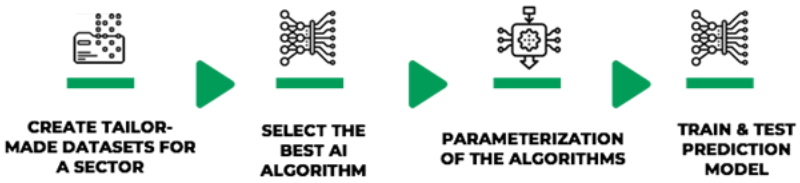

Figure 24: Risk prediction methodology

More specifically, food incident data of the last 30 years is used to train a deep learning prediction model to predict food incidents in the future. The incidents' dataset becomes available through the Data API of the data platform. To that direction, a request to the Data API of the workflow presented in the previous section is sent with the product (ingredient or material) for which we want to predict the incidents of the next 12 months. We can build prediction models for specific hazards and specific regions by filtering the result set with the country of interest and the hazard.

For the implementation of the prediction models that are based on the historical food safety incidents Prophet is used (Sean, et al 2018). Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fitted with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

The validation of a forecasting model like the one that was used in this study, can be achieved by holding some data back from the model, such as the last 12 months. Then, fitting the model on the first portion of the data, using it to make predictions on the held-pack portion, and calculating an error measure, such as the mean absolute error across the forecasts. E.g., a simulated out-of-sample forecast. The score gives an estimate of how well we might expect the model to perform on average when making an out-of-sample forecast. This can be done with the samples data by creating a new dataset for training with the last 12 months removed.

Generating Monthly and Weekly Global Incidents Predictions

Finally, Agroknow has also developed a software which provides monthly and weekly global incidents predictions. More specifically, we developed the following:

- Software that deploys monthly incident forecasting models based on Prophet. It deploys a trained & tested operational algorithm that is predicting upcoming monthly incidents for each ingredient, product and hazard category. The project therefore updates >250 incident prediction models (one per ingredient/product/hazard). The operational versions of these models feed with fresh predictions the Global Predictions Dashboard of FOODAKAI.

- Software that deploys a trained & tested operational algorithm that is predicting the upcoming weekly incidents for each ingredient, product and hazard category. The project therefore updates >250 incident prediction models (one per ingredient/product/hazard). Operational versions of these models are currently under integration within FOODAKAI.

The technologies used for the implementation of the weekly and monthly predictions are

- **Database**: MySQL
- **Web Server**: django embedded (wsgi)
- **Frameworks**: django, prophet, neural-prophet, scitkit-learn + keras

Intelligence API

The prediction engine includes a software API which is responsible for publishing and making searchable

a) all the risk estimation results by the operational risk assessment model
b) all prediction results produced by the operational models for monthly and weekly predictions

The Intelligence API powers FOODAKAI and Food Inspector dashboards that use risk assessment and predictions. Through TheFSM API Gateway, it can be also used to serve risk estimations and predictions to 3rd party platforms.

7. CONCLUSIONS AND NEXT STEPS

In this document we presented the FSM data services: data ingestion service; data curation service data publishing service, and risk assessment and prediction services. The initial version was based on state-of-the-art tools and services for data management and data integration, as presented in version 1 of this deliverable. The analysis and prioritization of the functional and non-functional requirements served as a basis in the FSM data services development. An enhanced version of these services is delivered in M24 and is presented in the current version (2.0) of the document, including the development of an automated data processing pipeline and a technical and scientific analysis of the risk assessment and prediction services. The implementation of the data services will continue in the last year of the project, based on the updated business, data and technical requirements, and the final versions will be described in version 3.0 of D2.2.