



TREBALL FINAL DE GRAU



ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Zihan Chen

Titulació: Bachelor's Degree in Computer Engineering

Títol de Treball Final de Grau: Assistant for the Transformation of Data
Tabular to Knowledge Graphs

Director/a: Roberto García González

Presentació

Mes: September

Any: 2024



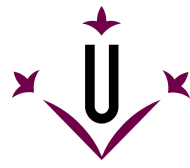
FINAL DEGREE PROJECT

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

Assistant for the Transformation of Tabular Data to Knowledge Graphs

AUTHOR/A: Zihan Chen

Lleida, September 2024



Universitat de Lleida
Escola Politècnica Superior

FINAL DEGREE PROJECT

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

Assistant for the Transformation of Tabular Data to Knowledge Graphs

SUPERVISOR: Roberto García González
AUTHOR: Zihan Chen

Lleida, September 2024

Acknowledgement

I want to express my most sincere gratitude to all the people who have contributed to the completion of this final degree project. Without their support and guidance, this project would not have been possible.

First of all, I would like to thank my advisor, Dr. Roberto García González, for his invaluable support and guidance throughout the entire process of developing this tool. His expertise and knowledge have been fundamental to the proper implementation of the proposed ideas. I am deeply grateful for his patience and advice, which have helped me grow both professionally and personally.

Similarly, I would like to extend my gratitude to Dr. Jorge Chamorro Padial, who provided his assistance at every stage of the project. His skill in web development and willingness to address my questions have been of great help, enabling me to progress in the development of this tool. Thank you for sharing your knowledge and for inspiring me to give my best in every stage of this work.

Contents

List of Figures	10
List of Tables	11
I Prolegomenon	1
1 Introduction	2
1.1 Motivation	2
1.2 Scope and Objectives	2
1.2.1 Scope	3
1.2.2 Objectives	3
1.3 Organisation of the document	3
2 Background	6
2.1 Context	6
2.2 State of the art	6
2.2.1 Semantic Web	6
2.2.2 FAIR Data	7
2.2.3 W3C	8
2.2.4 RDF	8
2.2.5 YARRRML	9
3 Project Management Plan	12
3.1 Development methodology	12
3.2 Technologies	14
3.2.1 Programming languages	15
3.2.2 Frameworks and libraries	15
3.2.3 Development tools	15
II Development	17
4 Iteration 1: First Steps	18
4.1 Requirements	18
4.2 Design	19
4.3 Implementation	19
4.4 Results	19
4.5 Timeline iteration 01	22
5 Iteration 2: Data Mapping	23
5.1 Requirements	23
5.2 Design	23

5.3	Implementation	26
5.4	Results	27
5.5	Timeline iteration 02	29
6	Iteration 3: Accessibility Improvement	30
6.1	Requirements	30
6.2	Design	30
6.3	Implementation	31
6.4	Results	32
6.5	Timeline iteration 03	33
7	Iteration 4: YAML File and Linked Data Generation	34
7.1	Requirements	34
7.2	Design	34
7.3	Implementation	36
7.4	Results	42
7.5	Timeline iteration 04	45
8	Iteration 5: Login and Signup	46
8.1	Requeriments	46
8.2	Design	46
8.3	Implementation	46
8.4	Results	53
8.5	Timeline iteration 05	55
9	Iteration 6: Authentication and RDF Generation	56
9.1	Requirements	56
9.2	Design	56
9.3	Implementation	57
9.4	Results	59
9.5	Timeline for Iteration 06	60
10	Iteration 7: Restructuring and Various Front-End Components	61
10.1	Requirements	61
10.2	Design	61
10.3	Implementation	62
10.4	Results	66
10.5	Timeline for Iteration 07	67
11	Iteration 8: RML Microservices	68
11.1	Requirements	68
11.2	Results	68
11.3	Timeline for Iteration 08	69
12	Iteration 9: Final Results	70
12.1	RDF Generator	70
12.1.1	Deployment	70
12.2	Timeline for Iteration 09	72

12.3	Final Application	72
12.3.1	Final Back-end Class Diagram	77
12.3.2	Example of a Mapping	77
III	Epilogue	93
13	Conclusions	95
13.1	Future Work	95
	Bibliography	98
IV	Annexes	103
	Glossary of terms	105
	Acronyms	106

List of Figures

3.1	Cascade methodology scheme[1]	12
3.2	Agile methodology scheme [2]	13
3.3	Waterfall Scheme vs agile [3]	14
4.1	Front-end snapshots Sprint 01	21
4.2	Timeline with the requirements of iteration 01	22
5.1	XSD datatype format [4]	25
5.2	Front-end snapshots Sprint 01	28
5.3	Timeline with the requirements of iteration 02	29
6.1	Compilation Time	30
6.2	Front-end snapshots Iteration 03	32
6.3	Timeline with the requirements of Iteration 03	33
7.1	Initial class diagram for the back-end	35
7.2	Initial diagram of linked data generation	42
7.3	Timeline with the requirements of Iteration 04	45
8.2	Timeline with the requirements of Iteration 05	55
9.1	MUI Snackbar	60
9.2	Timeline with the requirements of Iteration 06	60
10.1	Mapping details page	66
10.2	Timeline with the requirements of Iteration 07	67
11.1	Timeline with the requirements of Iteration 08	69
12.1	Final API communication schema	70
12.2	Timeline with the requirements of iteración 09	72
12.3	Login page	72
12.4	Signup page	73
12.5	Home page	73
12.6	File upload page	74
12.7	Mapping table	74
12.8	Dropdown for ontology mapping	75
12.9	Table with all user-created and public mappings	75
12.10	Mapping details page	76
12.11	Final back-end class diagram	77
12.12	CSV table imported	78
12.13	Ontology form 01	80
12.14	Ontology form 02	80

List of Tables

12.1 CSV file containing information about the pig and the farmyard	78
---	----

Part I

Prolegomenon

1. Introduction

Below, the motivation for this project and its scope are described. It also includes the organization of the remainder of this project's documentation

1.1. Motivation

In today's digital landscape, access to data and its collection is essential for societal progress. However, much of this data is collected in different formats, by different people, with diverse cultures and languages, etc., which presents a great challenge when reusing it for purposes such as scientific research or within the business environment.

In the context of scientific research, effective collaboration among researchers largely depends on the availability, accessibility, and interoperability of data. These characteristics are crucial for the advancement of knowledge in any domain. However, the exponential growth of data generated in recent decades has made its organization, access, and understanding increasingly difficult, posing a significant challenge to the scientific community.

The need for research data to be more accessible and understandable is indisputable. Despite the significant efforts and resources devoted to data collection and generation, sharing it effectively remains an obstacle, limiting its potential impact and usefulness for the scientific community in general.

This is where the FAIR initiative comes into play, which proposes that data should be Findable, Accessible, Interoperable and Reusable. These principles are designed to maximize the value of research data by ensuring that it is easily found, accessible through standardized communication protocols, interoperable between different systems, and reusable in various contexts.

For this reason, I am excited to contribute to the research community by creating advanced [semantic modelling](#) tools, such as data modeling through ontologies and transformation into [knowledge graphs](#), strengthening the application to handle the inherent complexity of research data and facilitating its analysis and interpretation, all while aligning with the FAIR principles to enhance its impact on the community.

1.2. Scope and Objectives

The scope and objectives detailed below outline the boundaries and goals pursued for the design and implementation of the proposed web tool.

1.2.1 Scope

The scope covers several essential points in developing a complete project, namely the conceptualization, design, and implementation of the web tool aimed at open data researchers. Therefore, the system is designed to offer a platform that allows users to:

- Annotate research data in a structured manner so that researchers can organize the data by detailing and including information on the type of measurements, units used, and measurable variables.
- Import data from open data repositories or local files, thereby enabling interoperability and the reuse of existing data.
- Visualize data in an intuitive and understandable way, facilitating users' interpretation and analysis.
- Model the data structure using [ontologies](#) and transform it into a knowledge graph through mapping tools, thus addressing semantic complexity and promoting the understanding and integration of data among different projects.

1.2.2 Objectives

The objectives presented below focus on creating a robust, functional, and intuitive web tool that meets the needs of researchers in managing and sharing research data, promoting transparency, collaboration, and the advancement of scientific knowledge. The specific objectives for this project are as follows:

- Investigate and understand the [FAIR](#) standards for open data management.
- Develop an intuitive and user-friendly interface that allows users to annotate, import, and visualize data effectively.
- Integrate semantic modeling tools for representing and converting data into ontologies and knowledge graphs through [RML](#) tools [2.2.5](#).
- Intelligent system for detecting the ontology according to the type of data to be modeled.

1.3. Organisation of the document

This document is organized into several parts that guide the reader from the fundamental concepts to the specific technical details of the developed work, concluding with the final contributions and possible future lines of work.

- **Chapter 1: Introduction**

This chapter presents the motivation behind the project, as well as its objectives and scope. It explains the reasons that led to the choice of the topic and describes the goals pursued in developing the work.

- **Chapter 2: Background**

This chapter offers an overview of the context in which the project is framed. It reviews the state of the art related to the transformation of tabular data into knowledge graphs, highlighting the existing technologies and tools that have inspired or influenced the development of this project.

- **Chapter 3: Project Management Plan**

This chapter describes the methodologies and technologies that will guide the development of the project.

- **Chapters 4-12**

In these chapters, the iterations from 1 to 9 are described, with each chapter divided into four sections: analysis, design, implementation, and results.

- **Chapter 13**

In this chapter, the conclusions of the work carried out are presented, highlighting the most significant contributions and reflecting on the lessons learned. Possible improvements and future lines of work are also proposed to continue with the development or expansion of the system.

- **Bibliography**

This section includes bibliographic references used throughout the development of the work, citing both the theoretical sources and the tools and technologies employed.

- **Annexes**

Finally, the appendices present a glossary of terms and acronyms to provide clear definitions of the terms used throughout the document.

2. Background

In this chapter, a detailed review of the relevant background that supports and contextualizes the present work is presented. It explores the current context and the fundamental theories needed to understand the state of the project under discussion.

2.1. Context

In the contemporary world, the exponential growth of technology has triggered the massive generation of data in practically every field, from scientific research to commerce and industry. This increase in both the quantity and diversity of data has brought about a series of challenges related to its management, organization, and efficient access. In the scientific realm, proper management of research data is essential not only to ensure the quality and reproducibility of results, but also to facilitate new research and discoveries.

The challenge lies in the fact that, as the volume of data grows at unprecedented rates, traditional information management tools have become insufficient. Therefore, new solutions are needed that not only allow for the storage and retrieval of large volumes of data, but also do so in a manner that makes the data accessible, interoperable, and reusable across different systems and platforms.

In this context, the development of tools and technologies that support the organization and understanding of data becomes a key factor in advancing scientific research. Such tools must be capable of handling [heterogeneous data](#) and facilitating their integration into broader systems, such as the semantic web, where data can be effectively linked and reused.

2.2. State of the art

2.2.1 Semantic Web

The Semantic Web is an evolution of the World Wide Web that seeks to improve the ability of machines to understand and process information published online. Unlike the traditional web, which is primarily designed for human consumption, the Semantic Web introduces a set of standards and technologies that allow data to be described in a structured and explicitly meaningful way, enabling machines to interpret and use this information more effectively.

The Semantic Web is based on several key concepts and technologies that make it function:

- **Metadata and Structured Meaning:** Metadata are data that describe other data. In the context of the Semantic Web, metadata are used to label information with precise meanings, which allows machines to understand not only the content

of a document but also its context and the relationships between different pieces of information.

- **Ontologies:** An ontology is a formal structure that defines a set of concepts and the relationships among them within a specific domain. In the Semantic Web, ontologies are used to model knowledge and establish a common vocabulary that can be shared and understood by different systems. This facilitates interoperability between applications and the coherent exchange of information.
- **Resource Description Framework (RDF):** RDF is a standard data model used to describe web resources and the relationships among them. RDF represents this information in the form of triples (subject, predicate, object), which allows data and their interrelationships to be expressed in a way that machines can process.
- **SPARQL:** SPARQL is a query language designed to retrieve and manipulate data stored in RDF format. It enables users and applications to perform complex queries on data distributed across the Semantic Web, facilitating rapid and efficient access to relevant information.
- **Linked Data:** Linked Data is a methodology for connecting and publishing structured data on the web so that it can be easily linked and queried by other datasets. By using standards such as RDF, linked data enables the creation of a global network of interconnected data that can be navigated and queried much like the hyperlinks on the traditional web.

2.2.2 FAIR Data

Open data must be defined according to certain principles that enhance the computational ability to find, access, interoperate, and reuse data, thereby minimizing human intervention. These principles enable computational systems to “understand” the data and establish semantic links between them. As both the volume and complexity of data grow, along with the constant influx of new data, it becomes increasingly necessary to depend on computational support for effective management.

To present these rules, it is important to define the entities to which these principles apply: data refers to any type of digital object that can be tied to a series of facts, while metadata consists of the information that describes this data, thereby providing it with specific properties.

The acronym and the principles were defined in the article published in March 2016 in the journal *Scientific Data* [5], in which the terms Findable, Accessible, Interoperable, and Reusable are established.

Encontrable

For data to be easily found by machines, they must include metadata that is readable by algorithms; therefore, these metadata must be assigned a unique identifier that ensures global persistence so that most users can locate them.

Accessible

To ensure the security of retrieved data, a must be added for access. Accordingly, this rule defines that data should be retrievable by its unique identifier using a standardized communication protocol. This protocol must be universal, open, and free.

Interoperable

This rule highlights the importance of integrating data with other data and facilitating their interaction with applications for analysis, storage, and processing. Consequently, metadata or data should use a formal, accessible, and shared language that has extensive application in knowledge representation.

Reutilizable

Finally, the last rule prioritizes the reuse of data. To achieve this, data or metadata must be properly constructed so that they can be replicated and combined in different contexts. It is stipulated that data or metadata should be described in detail with a diversity of precise and relevant attributes.

2.2.3 W3C

The World Wide Web Consortium (W3C) is an international organization that is fundamental in the development and standardization of web technologies. Founded in 1994 by Tim Berners-Lee, the creator of the World Wide Web, the W3C has been dedicated to promoting the growth of the web in an accessible, interoperable, and efficient manner. Its mission is to ensure that the web works for everyone, regardless of their abilities, location, or device.

One of the main functions of the W3C is the development of technical standards. These standards are crucial for ensuring interoperability among different platforms, systems, and devices on the web. Notable among them are HTML (Hypertext Markup Language), which defines the structure of web pages; CSS (Cascading Style Sheets), which is used to describe the presentation of an HTML document; and XML (eXtensible Markup Language), which allows data to be represented in a structured and machine-readable format. The creation of these standards is essential for fostering a cohesive and accessible web environment for all users.

2.2.4 RDF

The Resource Description Framework (RDF) is a standard developed by the World Wide Web Consortium (W3C) designed to describe resources on the web in a way that enables their interoperability and reuse. RDF provides a structured model to represent information about resources, facilitating the integration and exchange of data among different applications and domains.

The RDF model is based on the concept of representing information using triples, which consist of three parts: subject, predicate, and object. The subject is the resource being described, the predicate is the property or relationship established about the subject,

and the object is the value of that property or the related resource. For example, in the triple ‘Elon Musk’ (subject) ‘is the founder of’ (predicate) ‘SpaceX’ (object), a clear relationship is established between the entities. This approach allows RDF to express information in a flexible and semantic manner, facilitating machine understanding and analysis.

One of the most notable features of RDF is its ability to represent data in a decentralized manner. This means that resources can be identified by unique identifiers, such as URIs (Uniform Resource Identifiers), which allows information about a resource to be stored and managed in different locations. This decentralization promotes interoperability and data exchange among various systems and applications, making RDF particularly well-suited for building the Semantic Web.

RDF also facilitates the creation of ontologies, which are sets of concepts and categories within a particular domain, along with the relationships among them. This enables the development of a common vocabulary that different applications can use to describe and understand data consistently. Ontologies, together with RDF, are fundamental for representing knowledge and enabling the inference of new information from existing data.

RDF data can be serialized in various formats, the most common being RDF/XML, Turtle, and N-Triples. These formats allow RDF data to be easily stored, shared, and processed by web applications and data management systems. The choice of serialization format depends on the specific requirements of each application and the environment in which the data are used.

2.2.5 YARRRML

YARRRML (Yet Another RDF Mapping Language) is a language designed to facilitate the conversion of semi-structured data—such as XML, JSON, and HTML—into a structured RDF format. This process aids integration, since RDF, based on a graph model, allows tabular data to be represented hierarchically, much like JSON. Converting data to RDF not only enhances the structure and organization of the information, but also facilitates the explicit representation of its semantics, thereby aligning the data with FAIR principles.

In 2012, the W3C developed R2RML, a mapping language that enables users to create custom rules to transform data stored in relational databases into RDF. However, as the need to handle data in various formats increased, new extensions and languages emerged that expanded R2RML’s capabilities. Among these is RML, or “RDF Mapping Language,” which was designed to extend R2RML by enabling the construction of mappings that integrate data from sources in various formats such as CSV, JSON, and XML. Therefore, RML offers greater flexibility for transforming semi-structured and non-relational data into RDF, adapting to the diversity of data formats present on the web.

Developed as an extension of RML, YARRRML focuses on simplifying the creation of data mappings, making it more accessible for developers working with diverse data sources.

YARRRML makes use of YAML—a more intuitive and user-friendly serialization language—to apply the transformation and compile the mapping into RML. This compilation step converts the YARRRML definitions written in YAML into the corresponding RML instructions, which are then interpreted by tools such as [rmlmapper](#), developed by the same organization [RMLio](#). These tools take the data generated by YARRRML and transform it into RDF.

One of the most notable features of YARRRML is its syntax based on YAML (YAML Ain't Markup Language), which provides a more readable and understandable way to define mappings compared to other formats such as XML or JSON. The clarity and simplicity of YAML syntax allow users to create data mappings more intuitively, facilitating the adoption of YARRRML in projects that require data transformation.

YARRRML enables the definition of various mapping structures, including the specification of data sources and predicates that describe how data should be mapped to RDF. Its ability to handle multiple types of data sources, such as relational databases, CSV files, and [APIs](#), makes YARRRML highly versatile and applicable in a variety of contexts. By defining how each data source should be transformed into RDF triples, YARRRML provides a robust framework for data integration.

3. Project Management Plan

Below, a series of points that will come together to form a complete project plan are described.

3.1. Development methodology

Before starting the project, several methodologies were considered to be used throughout the entire process, simply because the chosen methodology covers the entire project life cycle. Therefore, it is very important to rigorously select one from the entire software development ecosystem so that it can complement the success of the development process.

To choose a project management methodology, both Waterfall and Agile were considered as possibilities. Both help in planning, executing, monitoring, and closing projects, but they differ in their philosophy and way of working.

The Waterfall methodology, or cascade, is based on a series of phases that must be completed one after another before moving to the next.

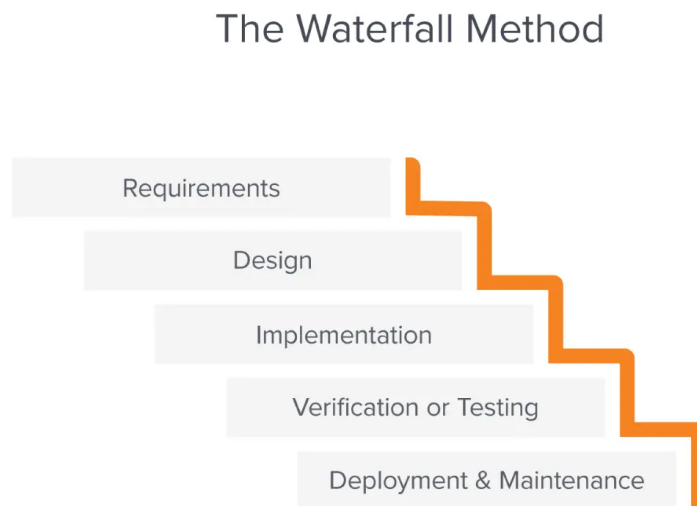


Figure 3.1: Cascade methodology scheme[1]

On the other hand, the Agile methodology is composed of iterations in which the team receives a continuous flow of feedback from the client. Instead of solving the problem just once, shorter phases are executed during the development cycle, which leads to progressive improvement.

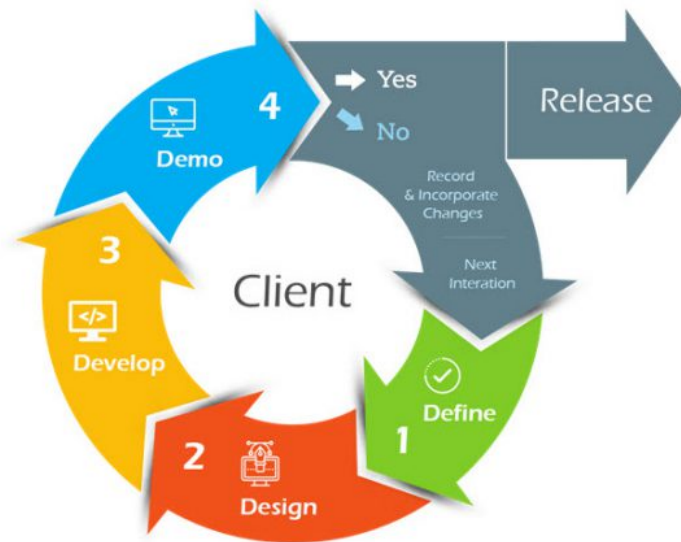


Figure 3.2: Agile methodology scheme [2]

The main differences between the two methodologies, in terms of their way of working, are that Waterfall implies a sequential and structured work process that allows clear objectives to be established from the beginning. In contrast, Agile prioritizes adapting to changes over following a static initial plan, since it focuses on delivering products that can be showcased quickly. Thus, more importance is given to responding to changes promptly rather than meticulously determining requirements and planning.

Furthermore, beyond the work style, they also differ in the type of project. Waterfall is more suited to complex projects, especially those where requirements are clearly defined and documented from the outset, allowing for rigorous control over development. This approach requires a high level of detail in the requirements, with fixed timelines and budgets. In contrast, Agile focuses on interactions with stakeholders, making communication key and involving frequent reviews to facilitate prompt decision-making.

Finally, they differ in client involvement. Waterfall does not require client involvement beyond an early phase for establishing requirements, whereas Agile demands close collaboration with clients, as constant feedback is necessary and they must be actively involved throughout the project.

As a result, the Agile methodology was chosen because its flexibility allows for much better adaptation to the changes and errors that occur throughout the project. This way, even without having all the essential requirements clearly defined from the beginning, the project will continuously benefit from stakeholder feedback to progressively build the complete solution.

Within Agile, there are different types according to the needs of each project, such as Scrum, Kanban, Extreme Programming (XP), Crystal, etc. Among these techniques, a hybrid combination of Scrum and Kanban was chosen, which is an effective strategy to combine the methodical structure of Scrum with the dynamic flexibility of

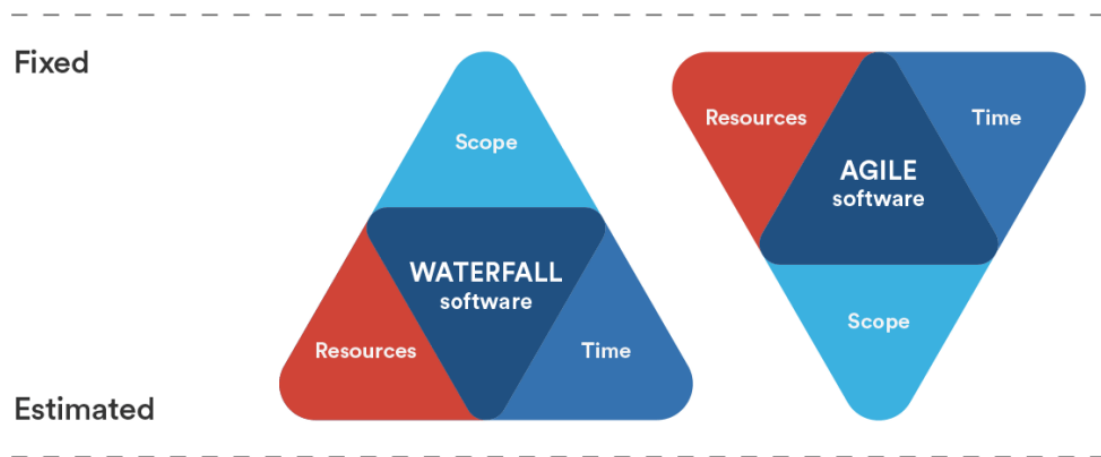


Figure 3.3: Waterfall Scheme vs agile [3]

Kanban.

Kanban offers a visual board for work that allows observation of the entire workflow, while Scrum's iterations and retrospective reviews enable developers to evaluate and adjust performance iteratively, ensuring that work is carried out in well-organized cycles.

This hybrid approach is especially useful for the effective development of new functionalities as well as for continuous maintenance or support. By integrating Scrum and Kanban, the team can plan efficiently, remain organized, and at the same time be agile and responsive to changes, achieving an ideal balance between structure and adaptability.

3.2. Technologies

In this section, the technologies and tools that will be used throughout the project are explained. As with the methodology, these tools will accompany the entire process, and their purpose will be to assist with work tracking, creation of new requirements, front-end and back-end programming, frameworks and libraries, and overall development tools.

For the selection of technologies, the *Developer Roadmap*¹ page recommended by the project advisor was primarily consulted, as it presents roadmaps, guides, and other content that help programmers choose their path and guide them during their learning process.

¹<https://roadmap.sh/>

3.2.1 Programming languages

Front-end

For the front-end, TypeScript was chosen as the high-level programming language. In addition to being a free and open-source project that adds a static typing system to JavaScript, it is the most widely used language by front-end frameworks such as Angular, Vue, React, etc., and it can also be used for back-end development using Node.js. All of this demonstrates the flexibility of TypeScript.

This choice has facilitated code maintainability by imposing a more structured coding framework. With more explicit type annotations, the code has become clearer, making it easier for other developers to understand. It also helps catch type-related errors during compilation. Overall, for a new project, TypeScript can provide a robust foundation that promotes clean, maintainable, and reusable code.

Back-end

The reason for choosing Java for back-end programming is that, in addition to being a language we became very familiar with during our studies, it is statically typed, which provides better performance and stability, as previously mentioned with TypeScript. Moreover, Java also provides better security systems, thanks to the Java Virtual Machine (JVM), which already ensures memory protection and complete ecosystem isolation. Java also contains a large number of libraries, tools, and frameworks available due to its large developer community, which facilitates the development and maintenance of large-scale applications.

3.2.2 Frameworks and libraries

Front-end

From the front-end perspective, the Next.js framework was chosen to support front-end development, as it has unique features such as automatic code splitting, server-side rendering, file-based routing, and Hot Module Replacement (HMR), among others. All these characteristics enable fast and efficient development.

Back-end

On the back-end, the Spring Boot framework was chosen for its ability to simplify the development of Java applications. Spring Boot offers minimal configuration, which facilitates the creation of robust and scalable applications, in addition to integrating security tools, data management, and RESTful services in a simple and efficient manner.

3.2.3 Development tools

Front-end

For front-end development, Visual Studio Code was chosen as the development tool. In addition to being a very popular and free IDE developed by Microsoft, it is

recognized for being fast and lightweight, thanks to its Electron-based architecture, efficient resource management, and its focus on deferred component loading.

Furthermore, it offers excellent support for multiple programming languages, a high degree of customization—allowing free configuration of the interface and keyboard shortcuts—and is supported by a large community of developers who maintain thousands of plugins for the [IDE](#).

Back-end

Similarly, for the previously mentioned back-end development, the IntelliJ Idea Ultimate Edition IDE developed and maintained by JetBrains will be used, as we, being university students, have access to this license free of charge. Moreover, IntelliJ is one of the leading IDEs in the Java development market. It also offers great support for development using the Spring and Spring Boot frameworks. IntelliJ provides features such as intelligent code understanding, code inspection, instantaneous code navigation, and high customizability of configurations.

It also offers integrated tools within the IDE that allow running and testing Spring applications, working with [HTTP](#) requests, and database tools, among many other features, such as support for Kotlin, Spring MVC, Spring Boot, Spring Integration, Spring Security, and Spring Cloud.

Part II

Development

4. Iteration 1: First Steps

This chapter covers the analysis, design, implementation, and results of the project's first sprint, in which we will address the main requirement points, their design and structure, their development, and finally the results achieved after the iteration.

4.1. Requirements

The requirements for this sprint are focused mainly on the presentation layer and the configuration of the logic layer, along with the exploration of different technologies.

After the research process within the project context, the first iteration of the project was launched, focused on investigating various technologies and frameworks available on the market, as well as programming languages and development environments.

Therefore, the proposed requirements for this iteration are:

- Front-end
 - Create a basic structure using the Next.js framework to program with React and TypeScript.
 - Add the Prettier library to standardize the code.
 - Add GitHub Actions to perform build and lint tests.
 - Create the first component to upload CSV files.
 - Modify the repository's Readme.
 - Create a table component for data visualization, adding a component for pagination.
 - Add a dropdown component that allows displaying a list of XSD data types and selecting one.
 - Add a hook so that the dropdown closes when the user interacts outside the component.
 - Develop an initial version to detect XSD formats using Regex (Regular Expressions).
- Back-end
 - Create a fork of the Spring Boot Template provided by the Software Architecture course organization recommended by the project advisor.

4.2. Design

In this section, the design of the application is outlined in accordance with the requirements set forth in the previous section.

For the presentation layer, to build a strong foundational front-end, the Next.js framework was used, as it provides a command that installs all the necessary dependencies and libraries to set up the initial steps of the application along with several other project configurations. Thus, the proposed template is created using the command *npm create-next-app@latest*.

At this stage, the main part of the application will include the folders **components**, **hooks**, **lib**, **ui**. It should be noted that to fully create a component, the logical layer must be developed in the **components** folder and the visual component in the **ui** folder.

On the other hand, for building the logic layer of our back-end application, the Spring Boot template from the Software Architectures course of the Computer Engineering degree at the Universitat de Lleida¹ was utilized to deliver an initial version of the project.

4.3. Implementation

Using the command to generate the Next.js template, all the necessary folders and their components were built. According to the defined requirements, the package **Prettier** was added to format the code. A GitHub Workflow was defined to drive CI/CD (Continuous Integration/Continuous Delivery).

Following the hierarchy of parent and child components, the components involved in this iteration have been built one by one, including the **upload-file component**, **table component**, **pagination component**, and **drop-down component**, all with their respective child components (**ui**) serving as the interface.

4.4. Results

The source code in this iteration is distributed across different folders:

- **components**: This folder contains all the system's logical components.
- **hooks**: In this folder, the React Hooks used in the application are developed.
- **lib**: Various functions are developed here; they are usually extracted separately because they will be reused in multiple places or to simplify the code.
- **ui**: This folder houses the visual components that are the children of the **components** folder.

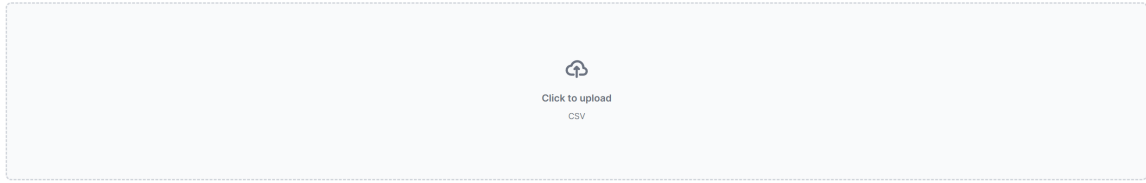
¹Spring Template, <https://github.com/UdL-EPS-SoftArch/spring-template>

In the following, some of the most relevant and important code pieces from this iteration are presented. It should be noted that the source code of the latest commit can be reviewed here, [4c9fc03](#):

Code 1 package.json

```
1 {
2   "name": "tab2kgwiz-client",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "dev": "next dev",
7     "build": "next build",
8     "start": "next start",
9     "lint": "next lint"
10  },
11  "dependencies": {
12    "next": "14.1.0",
13    "react": "^18",
14    "react-dom": "^18"
15  },
16  "devDependencies": {
17    "@types/node": "^20",
18    "@types/react": "^18",
19    "@types/react-dom": "^18",
20    "autoprefixer": "^10.0.1",
21    "eslint": "^8",
22    "eslint-config-next": "14.1.0",
23    "postcss": "^8",
24    "prettier": "3.2.4",
25    "tailwindcss": "^3.3.0",
26    "typescript": "^5"
27  }
28 }
```


Tab2KGWiz



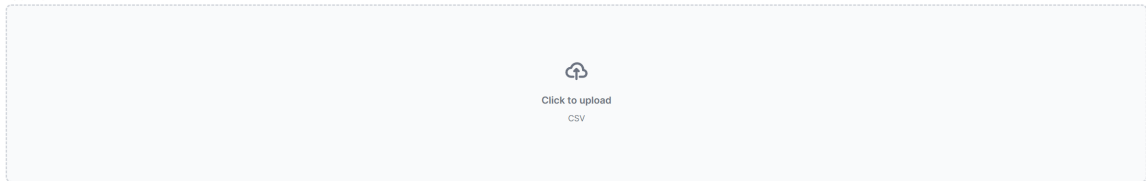
animal_number [integer] ▾	lifenumber [decimal] ▾	responder [integer] ▾	location [integer] ▾	visit_time [string] ▾	duration [integer] ▾	state [integer] ▾	weight [integer] ▾	feed_intake [string] ▾
194481		982091062894481	1	2021-03-16 00:23:12	48	0	0	0
194378		982091062894378	1	2021-03-16 00:27:35	90	0	24000	0
194360		982091062894360	1	2021-03-16 00:39:59	23	0	0	0
294362		982091062894362	2	2021-03-16 01:33:15	21	0	0	0
194414		982091062894414	1	2021-03-16 03:21:33	40	0	20500	0
194414		982091062894414	1	2021-03-16 04:43:34	50	0	20500	0
194414		982091062894414	1	2021-03-16 04:47:29	14	0	20500	0
194414		982091062894414	1	2021-03-16 06:16:52	32	0	20500	0
194378		982091062894378	1	2021-03-16 07:48:40	88	0	24000	0
294501		982091062894501	2	2021-03-16 08:24:45	196	0	0	0

Showing 1 - 10 of 10 results

< 1 2 3 4 5 ... >

(a) First view of the application after importing a CSV

Tab2KGWiz



animal_number [time] ▾	lifenumber [decimal] ▾	responder [integer] ▾	location [integer] ▾	visit_time [time] ▾	duration [integer] ▾	state [integer] ▾	weight [integer] ▾	feed_intake [string] ▾
<input type="radio"/> anyURI		982091062894481	1	2021-03-16 00:23:12	48	0	0	0
<input checked="" type="radio"/> time		982091062894378	1	2021-03-16 00:27:35	90	0	24000	0
<input type="radio"/> dateTime		982091062894360	1	2021-03-16 00:39:59	23	0	0	0
<input type="radio"/> date		982091062894362	2	2021-03-16 01:33:15	21	0	0	0
<input type="radio"/> boolean		982091062894414	1	2021-03-16 03:21:33	40	0	20500	0
<input type="radio"/> integer		982091062894414	1	2021-03-16 04:43:34	50	0	20500	0
<input type="radio"/> decimal		982091062894414	1	2021-03-16 04:47:29	14	0	20500	0
<input type="radio"/> float		982091062894414	1	2021-03-16 06:16:52	32	0	20500	0
<input type="radio"/> double		982091062894414	1	2021-03-16 07:48:40	88	0	24000	0
<input type="radio"/> string		982091062894414	1	2021-03-16 08:24:45	196	0	0	0

Showing 1 - 10 of 10 results

< 1 2 3 4 5 ... >

(b) View after opening the table's drop-down

Figure 4.1: Front-end snapshots Sprint 01

4.5. Timeline iteration 01

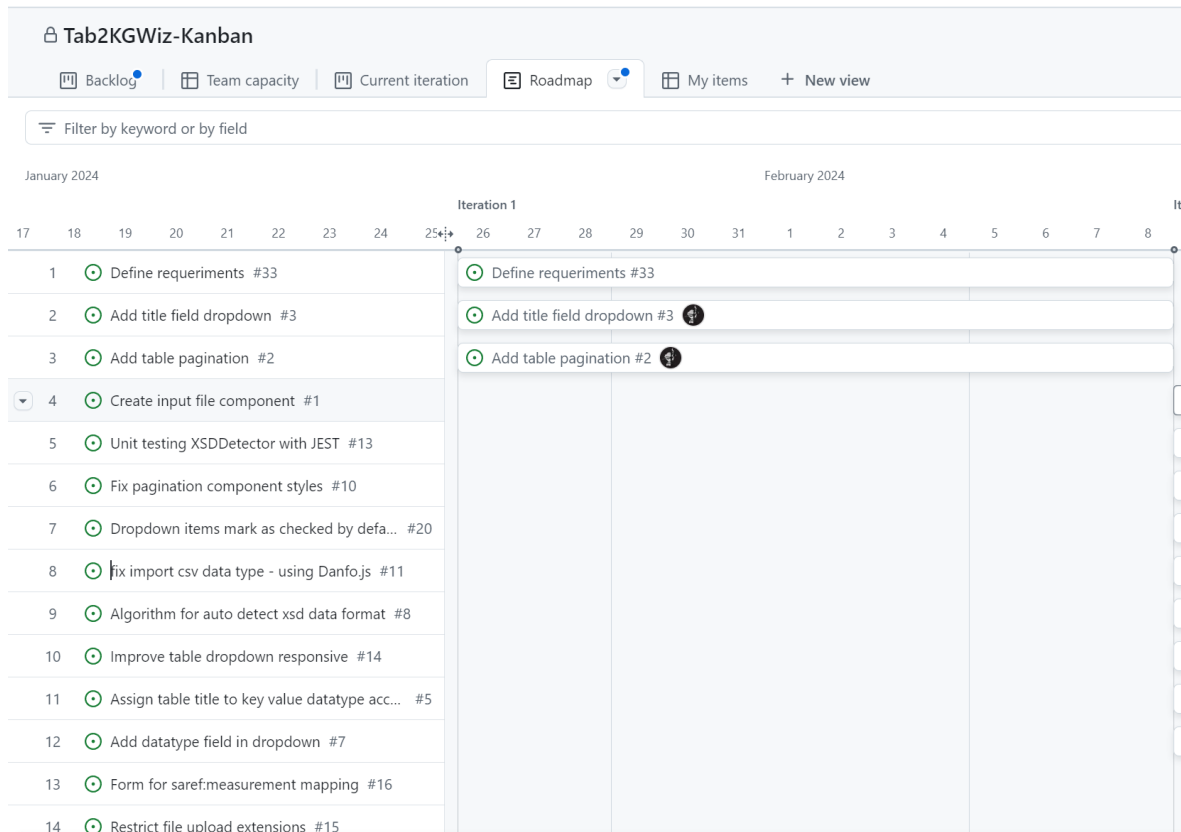


Figure 4.2: Timeline with the requirements of [iteration 01](#)

5. Iteration 2: Data Mapping

5.1. Requirements

After finishing iteration one, new requirements were proposed as the iteration was reviewed, and during iteration two new functionalities were also suggested following Agile principles.

Primarily, in this iteration the focus was on implementing the algorithm to auto-detect the XSD format using Test Driven Development (TDD), after reviewing the different situations or possibilities that may occur depending on the unit or standard of a measurement. For example, the format for hours, minutes, and seconds, or the fact that a measurement may be taken according to the time zone where it is located, which will require handling many cases. Therefore, it is useful to use the TDD technique to implement this functionality.

Therefore, the proposed requirements for this iteration are:

- Front-end
 - Assign the column titles by storing them as key-value pairs, with the column title as the key and the XSD format as the value.
 - Install the Danfo.js library and use it to handle the input CSV data.
 - Correct the styles of the pagination component.
 - The DropDown field should be set by default when the file is loaded.
 - Implement an algorithm to auto-detect the XSD type format.
 - Improve the adaptability of the table's DropDown.
 - Use Test Driven Development (TDD) to implement the auto-detection of the XSD format using unit tests with the JEST framework.

5.2. Design

To install the Danfo.js library, the command `npm install danfojs` was used. Next, to implement the data structure for storing the key-value mapping of the column title with its respective XSD format, the native JavaScript data structure *Map* was used within a hook so that the hook can later be passed to the child component:

Code 2 upload-file.tsx

```
1 ...  
2 const [headerMapping, setHeaderMapping] = React.useState<Map<string, string>>(  
3   new Map(),
```

```
4   );  
5   ...
```

Similarly, to add the default marking functionality for the DropDown, the *disabled* property must be used to mark the format types that were previously detected with the detection algorithm.

The cases to handle are: *10.000,00* is mainly used in non-Anglophone countries and *10,000.00* in Anglophone countries:

- Decimal separator

$\hookrightarrow 10.000 \equiv \text{xsd:integer}$

$\hookrightarrow 10,000 \equiv \text{xsd:integer}$

For date and time data there are many formats that may be represented depending on the time zone. Therefore, these are the formats primarily used in XSD and defined by the World Wide Web Consortium (W3C) in the *Datatypes*¹ standard.

In the following figure, the complete schema of the data types defined by this standard can be observed:

¹<https://www.w3.org/TR/xmlschema-2/>

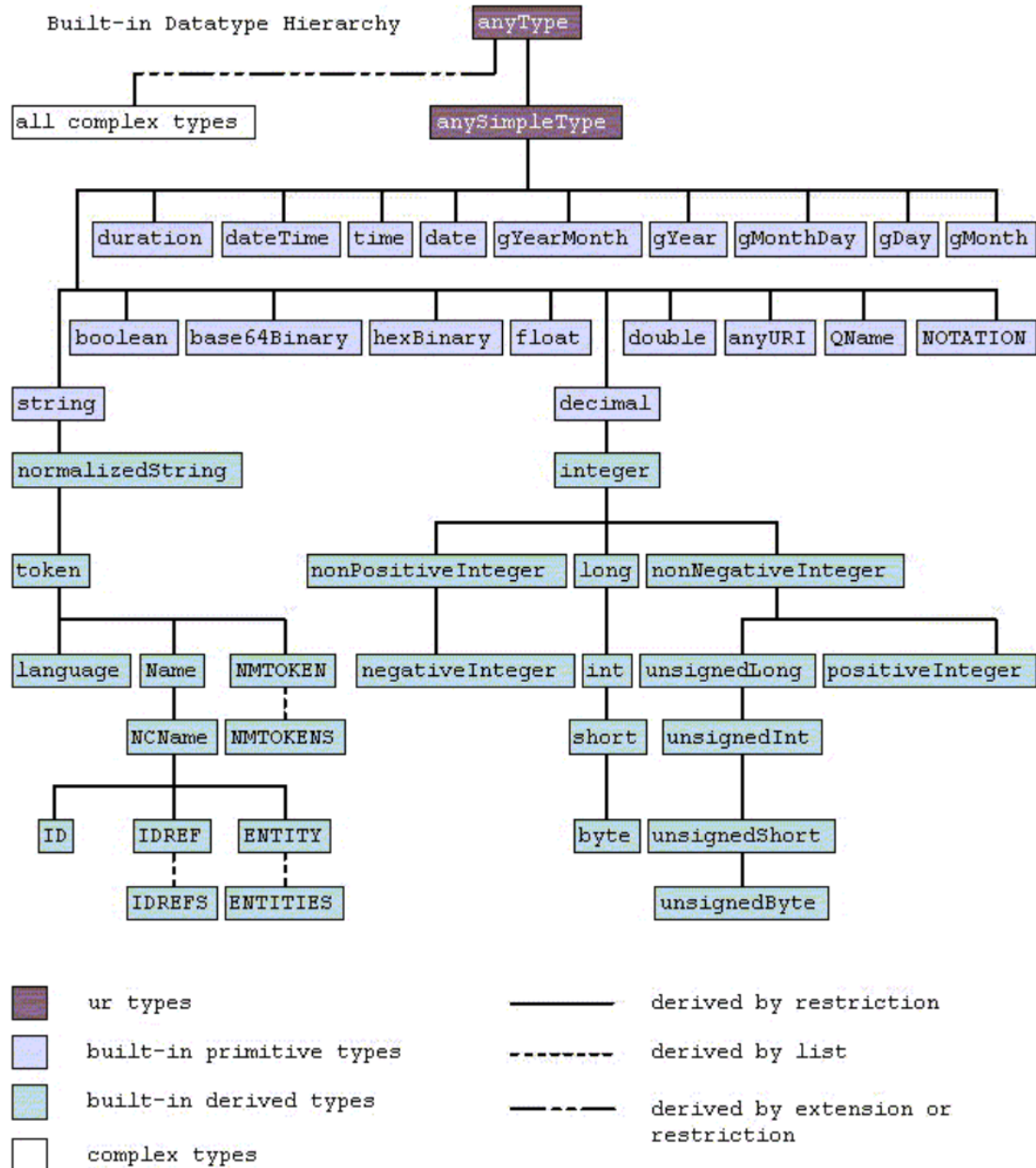


Figure 5.1: XSD datatype format [4]

- Date and Time
 - Date
 - $\hookrightarrow 2002-09-24 \equiv \text{xsd:date}$
 - $\hookrightarrow 2002-09-24Z \equiv \text{xsd:date}$
 - $\hookrightarrow 2002-09-24-06:00 \equiv \text{xsd:date} - \text{time zones}$
 - $\hookrightarrow 2002-09-24+06:00 \equiv \text{xsd:date} - \text{time zones}$
 - Time
 - $\hookrightarrow 09:00:00 \equiv \text{xsd:time}$

- ↪ 09:30:10.5 ≡ xsd:time
- ↪ 09:30:10Z ≡ xsd:time - time zones
- ↪ 09:30:10-06:00 ≡ xsd:time - time zones
- ↪ 09:30:10+06:00 ≡ xsd:time - time zones
- DateTime
 - ↪ 2002-05-30T09:00:00 ≡ xsd:dateTime
 - ↪ 2002-05-30T09:30:10.5 ≡ xsd:dateTime
 - ↪ 2002-05-30T09:30:10Z ≡ xsd:dateTime - time zones
 - ↪ 2002-05-30T09:30:10-06:00 ≡ xsd:dateTime - time zones
 - ↪ 2002-05-30T09:30:10+06:00 ≡ xsd:dateTime - time zones
- Duration
 - ↪ P5Y ≡ xsd:duration
 - ↪ P5Y2M10D ≡ xsd:duration
 - ↪ P5Y2M10DT15H ≡ xsd:duration
 - ↪ PT15H ≡ xsd:duration
 - ↪ -P10D ≡ xsd:duration - negative
- Other formats
 - ↪ DD ≡ xsd:gDay
 - ↪ MM ≡ xsd:gMonth
 - ↪ MM-DD ≡ xsd:gMonthDay
 - ↪ YYYY ≡ xsd:gYear
 - ↪ YYYY-MM ≡ xsd:gYearMonth

This chapter addresses all aspects related to the implementation of the system in code, using a specific technological environment.

5.3. Implementation

Using the command to create the Next.js template, all the necessary folders and components were built. According to the defined requirements, the **Prettier** package was added to format the code. A GitHub Workflow was defined to drive CI/CD (Continuous Integration/Continuous Delivery).

Following the hierarchy of parent and child components, the components involved in this iteration have been built one by one, including the **upload-file component**, **table component**, **pagination component**, and **drop-down component**, all with their respective child components (**ui**) serving as the interface.

5.4. Results

The source code in this iteration is distributed across different folders:

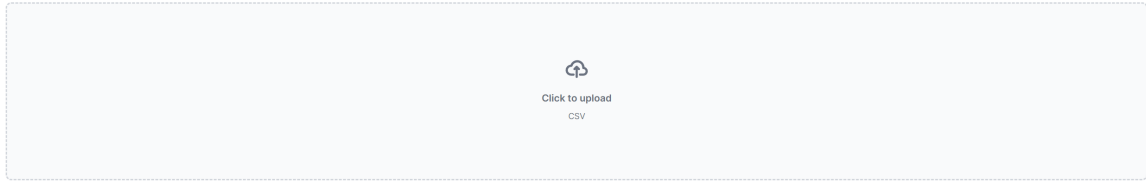
- **components**: This folder contains all the logical components of the system.
- **hooks**: In this folder, the React Hooks used in the application are developed.
- **lib**: Various functions are built here; they are usually extracted separately because they will be reused in multiple places or to simplify the code.
- **ui**: This folder houses the visual components that are children of the **components** folder.

Below are some of the most relevant and important pieces of code from this iteration. It should be noted that the source code of the latest commit can be reviewed here, [4c9fc03](#):

Code 3 package.json

```
1 {
2   "name": "tab2kgwiz-client",
3   "version": "0.1.0",
4   "private": true,
5   "scripts": {
6     "dev": "next dev",
7     "build": "next build",
8     "start": "next start",
9     "lint": "next lint"
10  },
11  "dependencies": {
12    "next": "14.1.0",
13    "react": "^18",
14    "react-dom": "^18"
15  },
16  "devDependencies": {
17    "@types/node": "^20",
18    "@types/react": "^18",
19    "@types/react-dom": "^18",
20    "autoprefixer": "^10.0.1",
21    "eslint": "^8",
22    "eslint-config-next": "14.1.0",
23    "postcss": "^8",
24    "prettier": "3.2.4",
25    "tailwindcss": "^3.3.0",
26    "typescript": "^5"
27  }
28 }
```

Tab2KGWiz



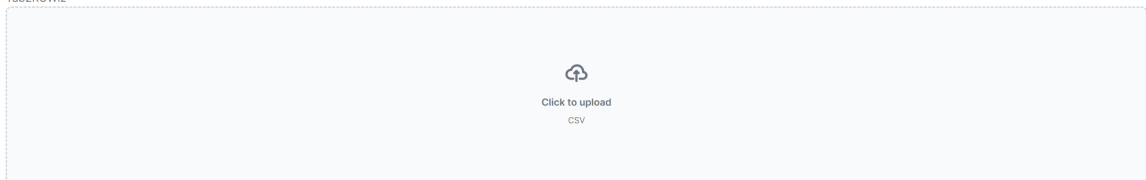
animal_number [integer] ▼	lifenumber [decimal] ▼	responder [integer] ▼	location [integer] ▼	visit_time [string] ▼	duration [integer] ▼	state [integer] ▼	weight [integer] ▼	feed_intake [string] ▼
194481		982091062894481	1	2021-03-16 00:23:12	48	0	0	0
194378		982091062894378	1	2021-03-16 00:27:35	90	0	24000	0
194360		982091062894360	1	2021-03-16 00:39:59	23	0	0	0
294362		982091062894362	2	2021-03-16 01:33:15	21	0	0	0
194414		982091062894414	1	2021-03-16 03:21:33	40	0	20500	0
194414		982091062894414	1	2021-03-16 04:43:34	50	0	20500	0
194414		982091062894414	1	2021-03-16 04:47:29	14	0	20500	0
194414		982091062894414	1	2021-03-16 06:16:52	32	0	20500	0
194378		982091062894378	1	2021-03-16 07:48:40	88	0	24000	0
294501		982091062894501	2	2021-03-16 08:24:45	196	0	0	0

Showing 1 - 10 of 10 results

< 1 2 3 4 5 ... >

(a) First view of the application after importing a CSV

Tab2KGWiz



animal_number [time] ▼	lifenumber [decimal] ▼	responder [integer] ▼	location [integer] ▼	visit_time [time] ▼	duration [integer] ▼	state [integer] ▼	weight [integer] ▼	feed_intake [string] ▼
<input type="radio"/> anyURI		982091062894481	1	2021-03-16 00:23:12	48	0	0	0
<input checked="" type="radio"/> time		982091062894378	1	2021-03-16 00:27:35	90	0	24000	0
<input type="radio"/> dateTime		982091062894360	1	2021-03-16 00:39:59	23	0	0	0
<input type="radio"/> date		982091062894362	2	2021-03-16 01:33:15	21	0	0	0
<input type="radio"/> boolean		982091062894414	1	2021-03-16 03:21:33	40	0	20500	0
<input type="radio"/> integer		982091062894414	1	2021-03-16 04:43:34	50	0	20500	0
<input type="radio"/> decimal		982091062894414	1	2021-03-16 04:47:29	14	0	20500	0
<input type="radio"/> float		982091062894414	1	2021-03-16 06:16:52	32	0	20500	0
<input type="radio"/> double		982091062894414	1	2021-03-16 07:48:40	88	0	24000	0
<input type="radio"/> string		982091062894501	2	2021-03-16 08:24:45	196	0	0	0

Showing 1 - 10 of 10 results

< 1 2 3 4 5 ... >

(b) View after opening the table's drop-down

Figure 5.2: Front-end snapshots Sprint 01

5.5. Timeline iteration 02

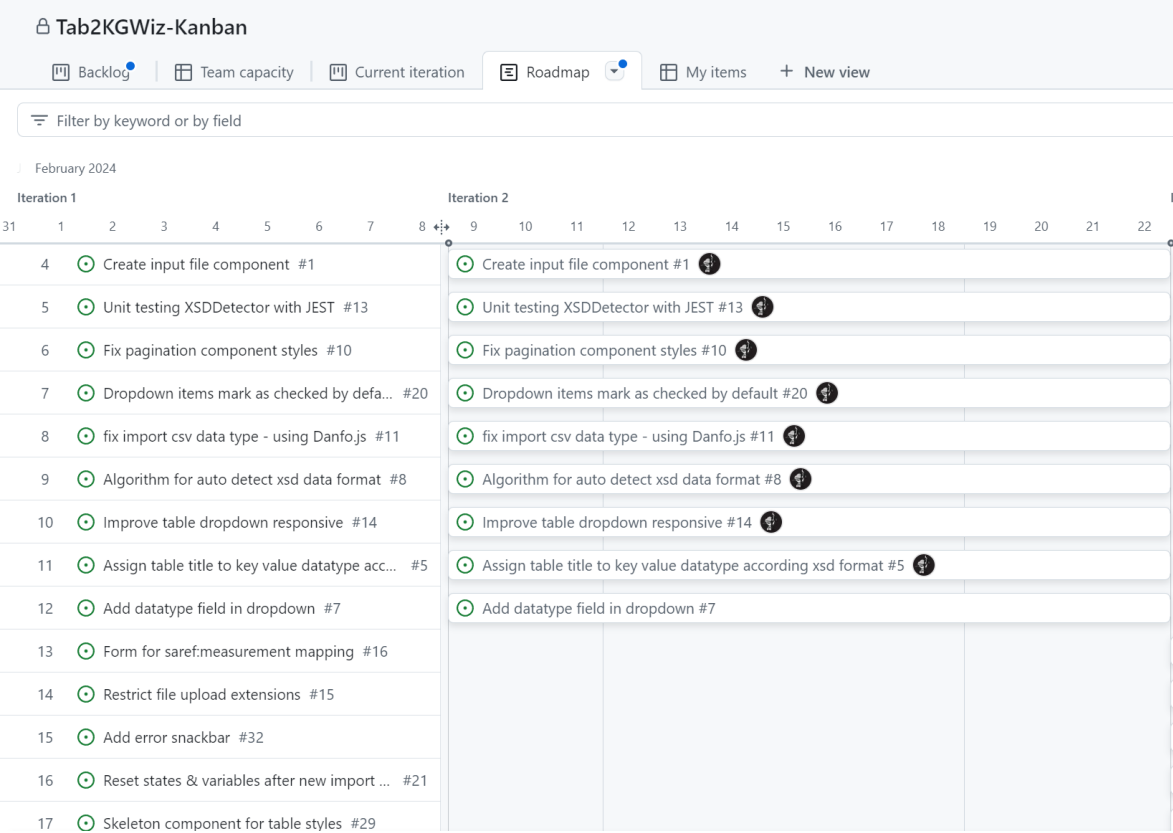


Figure 5.3: Timeline with the requirements of [iteration 02](#)

6. Iteration 3: Accessibility Improvement

6.1. Requirements

In this iteration the focus has been on resolving some issues, bugs, and warnings that were indicated, along with the implementation of a graphical component to better visualize and interpret the program's state.

Therefore, the proposed requirements for this iteration are:

- Front-end
 - A form to map if a field is of the **measurement** type.
 - Restrict the format of the file imports.
 - Add an error Snackbar for file import failures.
 - Reset states and variables after importing a new file.
 - Add a Skeleton component to display a loading placeholder for the table, providing user feedback while waiting.
 - Optimize the page load time.
 - Correct the browser warnings.
 - Fix the bug related to the xsd format type.

6.2. Design

In the previous iteration, it was noticed that there was a performance issue— the initial page was loading very slowly, approximately 8-10 seconds for a homepage that barely contains any resource-intensive visual elements. This can be observed in Figure 6.1.



Figure 6.1: Compilation Time

After investigating the problem, it was observed that the performance degradation was due to the initial load when importing the package *import * as dfd from "danfojs"*; as it

imports all components of the `danfo` package, causing a considerable loss in performance for the application.

Another issue resolved in this iteration is the error in auto-detecting the `xsd` format that was implemented in previous iterations. This error manifested as an inconsistent `xsd` algorithm output after performing several consecutive file imports without reloading the page. It occurred because the variables (hooks) were not reset after each new import, leading to an incoherent replacement of these variables, which resulted in the `xsd` format being misaligned with its corresponding column—even after repeatedly importing the same file.

Perhaps in a future iteration, a caching system could be implemented to store the import data and avoid recalculating values when the same file is reimported.

Subsequently, to improve the user experience, two components were added. First, a *skeleton* component was implemented to display a loading skeleton in the background of the page, simulating content loading to provide feedback to the user while they wait. Second, since the file formats that the user can import have been limited, an error message was added to inform the user that a file with that format cannot be imported.

6.3. Implementation

To resolve the inconsistency problem with the `xsd` format discussed above, a `React useEffect` was defined to monitor when a new file is imported. When the user imports a new file, it detects that the file has changed and resets the content of the variables. The specific modification can be seen in code 4.

Code 4 upload-file.tsx - commit: [d3e9f9e](#)

```
1  ...
2  useEffect(() => {
3    setHeader(undefined);
4    setRow([]);
5    setHeaderMapping(new Map());
6  }, [file]);
7  ...
```

Next, to reduce the initial load time for the user, a *promise* was used to load the `danfo` library only when the user requests it; that is, when the user imports an initial file, the elements of the library are loaded. This significantly improves the user experience. The specific modification can be seen in code 5.

Code 5 upload-file.tsx - commit: [d3e9f9e](#)

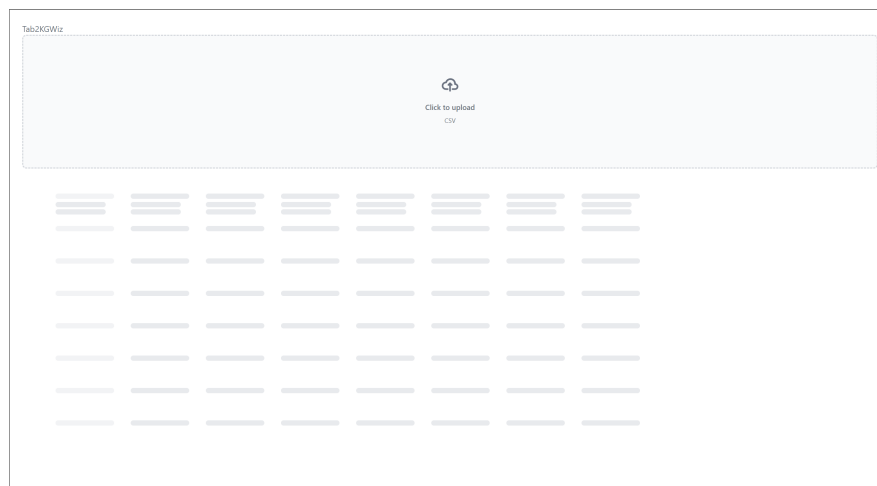
```
1  ...
2  (async () => {
3    const dfd = await import("danfojs");
4    ...
```

```
5 }  
6 ...
```

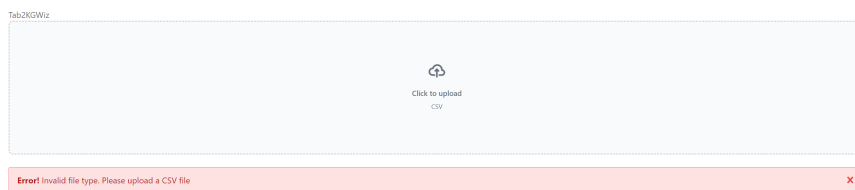
Among several other adjustments, the two important visual components for user feedback have been implemented and the browser warnings have been fixed.

6.4. Results

The following presents the visual elements that provide feedback to the user:



(a) Visual skeleton during the table load time



(b) Alert after loading a non-acceptable file format

Figure 6.2: Front-end snapshots Iteration 03

6.5. Timeline iteration 03

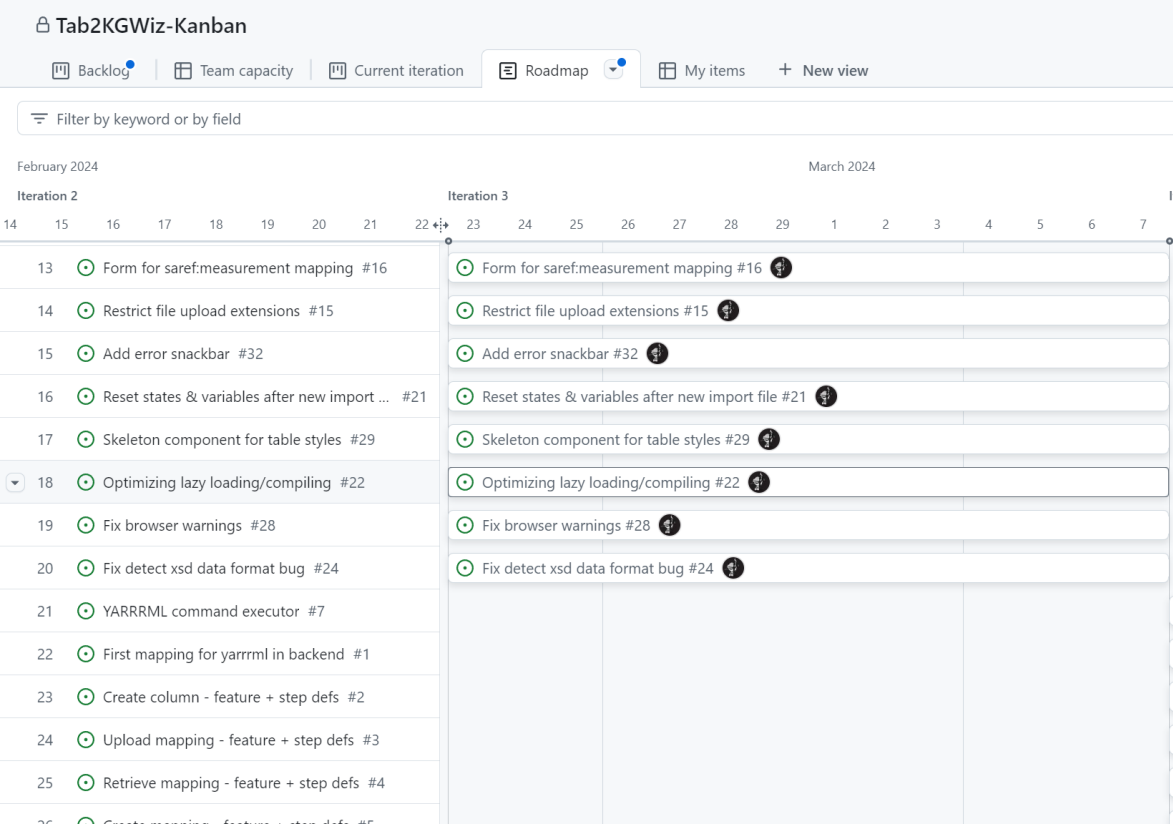


Figure 6.3: Timeline with the requirements of [Iteration 03](#)

7. Iteration 4: YAML File and Linked Data Generation

7.1. Requirements

After implementing a fundamental part of the front-end in previous iterations, in this iteration the focus is on designing and implementing the back-end API. The back-end API will primarily serve the function of receiving the mapping definition from the front-end, transforming it into a YAML file, then using that file to convert it into an RML *rml.ttl* file that will ultimately be used to transform structured files into RDF format.

But before that, following the Spring Boot framework, it is necessary to construct the main classes of the API.

Therefore, the proposed requirements for this iteration are:

- Back-end
 - Create the *mapping* class with its features and step definitions for the CREATE, RETRIEVE, UPDATE/UPLOAD, and DELETE operations.
 - Create the *supplier* class with its features and step definitions for the CREATE, RETRIEVE, UPDATE/UPLOAD, and DELETE operations.
 - Create the *column* class with its features and step definitions for the CREATE, RETRIEVE, UPDATE/UPLOAD, and DELETE operations.
 - Generate an initial YAML file.
 - Execute commands for generating RML and RDF files using the RMLio frameworks.

7.2. Design

From the front-end, the representation of the table with its respective column mappings is already visible; therefore, in this iteration an initial design of the back-end API will be studied using the Spring Boot framework. This involves building the Java domain classes that will represent each table forming the skeleton of the API, and using the annotations provided by Spring Boot to create the service.

Thus, the initial class diagram is reflected in the following figure:

For the initial construction of the API classes, BDD (Behavior-Driven Development) testing methodology has been employed. This approach, a form of behavior-guided development, follows the Agile methodology and will define tests that focus on the user and the system's behavior rather than solely on its functionality. To achieve

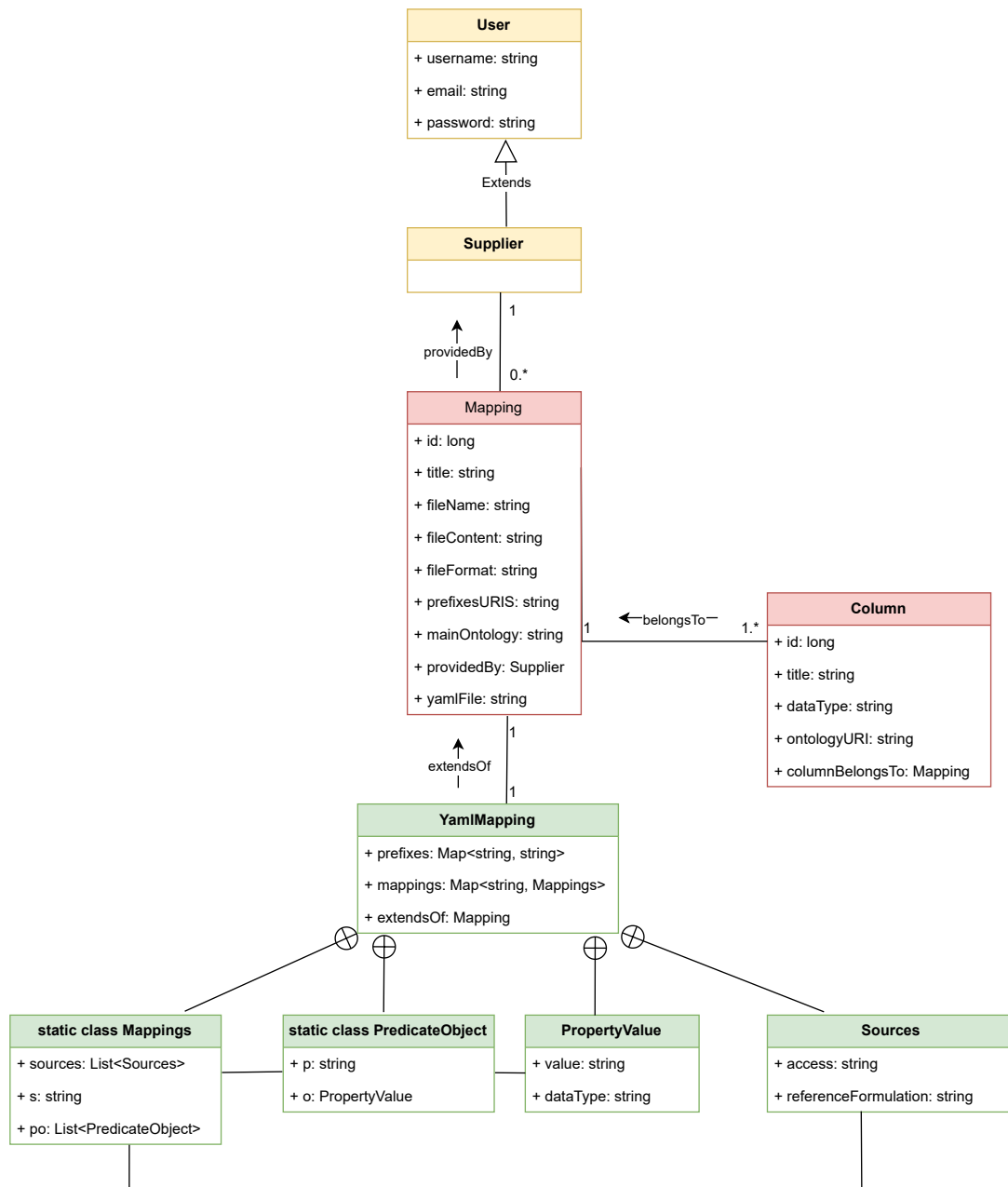


Figure 7.1: Initial class diagram for the back-end

this, the Cucumber library written in Ruby was used, as it is designed to implement BDD.

Then, to execute the RML.io yarrml-parser framework, a file with the YAML extension is needed. To more easily define the YAML instruction file compatible with YARRRML from the back-end code, the Jackson library is used to convert Java classes into YAML format. To create the class that will be called `YamlMapping`, it extends the `Mapping` class and therefore will contain the attributes that represent each field. If a field contains other fields, a nested class is used to create an instance of that attribute.

7.3. Implementation

Following the class diagram outlined in the design phase, the entities will be implemented.

First, a data repository (JPA repository) is built for the Mapping domain. This repository will essentially contain the APIs for the basic CRUD (Create, Retrieve, Update, Delete) operations, as well as APIs for pagination and sorting. This technology is used to reduce the amount of code needed for database access since Spring Data JPA already provides the methods to perform these accesses.

Code 6 MappingRepository.java - commit: [c1f1f27](#)

```
1  ...
2  @RepositoryRestResource
3  public interface MappingRepository extends PagingAndSortingRepository<Mapping, Long>
4  ↪ {
5      List<Mapping> findByTitle(@Param("title") String title);
6      @Override
7      public Iterable<Mapping> findAll(Sort sort);
8
9      @Override
10     public Page<Mapping> findAll(Pageable pageable);
11
12     public Optional<Mapping> findById(Long id);
13 }
```

Next, to define the controllers, the endpoints needed to access the Mapping resource are implemented. By using its ID, all the attributes of the Mapping class can be retrieved, as shown in part of the following code:

Code 7 MappingController.java - commit: [c1f1f27](#)

```
1  ...
2  @RepositoryRestController
3  public class MappingController {
4      private final MappingRepository mappingRepository;
5
6      @RequestMapping(value = "/mappings/{id}", method = RequestMethod.GET)
7      public @ResponseBody PersistentEntityResource
8      ↪ getMapping(PersistentEntityResourceAssembler resourceAssembler,
9                  @PathVariable Long id) {
10         ...
11     }
12 }
```

Furthermore, since a Mapping is created by a user who, in the context of the application, is referred to as Supplier (one of the first user roles in the application), this Supplier will need the ability to register:

Code 8 RegisterSupplier.feature - commit: [c1f1f27](#)

```
1 ...
2 Scenario: Register provider with invalid email
3     Given I'm not logged in
4     When I register a new supplier with username "supplier", email "invalidEmail" and
5     ↪ password "password"
6     Then The response code is 400
7     And The error message is "must be a well-formed email address"
8     And It has not found a supplier with username "supplier"
9 ...
```

Similarly, for creating a Mapping:

Code 9 CreateMapping.feature - commit: [c1f1f27](#)

```
1 ...
2 Feature: Create a Mapping of the Data
3     In order to create a mapping of the data
4     As a provider
5     I want to create a mapping of the data
6
7     Scenario: Create a mapping of the data
8         Given There is a registered supplier with username "supplier", email
9         ↪ "supplier@sample.app" and password "password"
10        And I login as "supplier" with password "password"
11        When I create a new mapping with name "Sample Mapping"
12        Then The response code is 201
13 ...
```

Moreover, to facilitate testing, a default supplier user has been defined exclusively for testing purposes:

Code 10 AuthenticationConfig.java - commit: [c1f1f27](#)

```
1 ...
2 public AuthenticationConfig(BasicUserDetailsService basicUserDetailsService,
3     ↪ UserRepository userRepository,
4     SupplierRepository supplierRepository) {
5     ...
6     // Sample User
7     if (!userRepository.existsById("demo")) {
8         User user = new User();
9         user.setEmail("demo@sample.app");
10        user.setUsername("demo");
11        user.setPassword(defaultPassword);
12        user.encodePassword();
13        userRepository.save(user);
14    }
15 ...
```

Once the features simulating the user's interaction with the system have been defined,

the steps for each definition line created in the features must be implemented. For example:

Code 11 CreateMappingStepDefs.java - commit: [c1f1f27](#)

```
1  ...
2  public class CreateMappingStepDefs {
3      ...
4      @When("I create a new mapping with name {string}")
5      public void iCreateANewMappingWithTitleAndDescription(String name) throws
6          ↳ Throwable {
7          Mapping mapping = new Mapping();
8          mapping.setTitle(name);
9
10         stepDefs.result = stepDefs.mockMvc.perform(
11             post("/mappings")
12             .contentType(MediaType.APPLICATION_JSON_UTF8)
13             .content(stepDefs.mapper.writeValueAsString(mapping))
14             .accept(MediaType.APPLICATION_JSON)
15             .with(AuthenticationStepDefs.authenticate()))
16             .andDo(print());
17     }
18 }
```

Next, to create the YAML instruction file, the structure of the YAML file is defined. The indentation of the code reveals that this file will contain two major structures: the first corresponding to *prefixes*, which itself contains a structure of prefix and content (i.e., a Map<String, String> called *prefixes*), and another Map<String, Mappings> called *mappings*. In the latter, the key (of type *string*) will contain the name of the mapping—in this case *animalWeight*—and its value will be an instance of the *Mappings* class. Within *animalWeight*, it is observed that it contains three additional attributes: *sources*, *s*, and *po*. The *Sources* class will contain the attributes *access* and *referenceFormulation*. The *PredicateObject* class will also be defined with two attributes: *p* and *o*.

Therefore, YamlMapping will be structured as follows:

Code 12 YamlMapping.java - commit: [c1f1f27](#)

```
1  ...
2  @JsonInclude(JsonInclude.Include.NON_NULL)
3  public class YamlMapping extends Mapping {
4
5      @JsonProperty
6      private Map<String, String> prefixes;
7      @JsonProperty
8      private Map<String, Mappings> mappings;
9
10     @OneToOne
11     @JsonIdentityReference(alwaysAsId = true)
12     private Mapping extendsOf;
```

```

13
14     public void setPrefixes(Map<String, String> prefixes) {
15         this.prefixes = prefixes;
16     }
17
18     public void setMappings(Map<String, Mappings> mappings) {
19         this.mappings = mappings;
20     }
21
22     public void setExtendsOf(Mapping extendsOf) {
23         this.extendsOf = extendsOf;
24     }
25
26     @JsonInclude(JsonInclude.Include.NON_NULL)
27     public static class Mappings {
28         @JsonProperty
29         private List<Sources> sources;
30
31         @JsonProperty
32         private String s;
33         @JsonProperty
34         private List<PredicateObject> po;
35
36
37         public void setSources(List<Sources> sources) {
38             this.sources = sources;
39         }
40
41         public void setS(String s) {
42             this.s = s;
43         }
44
45         public void setPo(List<PredicateObject> po) {
46             this.po = po;
47         }
48     }
49
50
51     @JsonInclude(JsonInclude.Include.NON_NULL)
52     public static class PredicateObject {
53         @JsonProperty
54         private String p;
55         @JsonProperty
56         private PropertyValue o;
57
58         public void setP(String p) {
59             this.p = p;
60         }
61
62         public void setO(PropertyValue o) {
63             this.o = o;
64         }
65
66     }
67

```

```

68     @JsonInclude(JsonInclude.Include.NON_NULL)
69     public static class PropertyValue {
70         @JsonProperty
71         private String value;
72         @JsonProperty
73         private String datatype;
74
75         public PropertyValue(String value, String datatype) {
76             this.value = value;
77             this.datatype = datatype;
78         }
79
80         public PropertyValue(String value) {
81             this.value = value;
82         }
83     }
84
85     @JsonInclude(JsonInclude.Include.NON_NULL)
86     public static class Sources {
87         @JsonProperty
88         private String access;
89         @JsonProperty
90         private String referenceFormulation;
91
92         public void setAccess(String access) {
93             this.access = access;
94         }
95
96         public void setReferenceFormulation(String referenceFormulation) {
97             this.referenceFormulation = referenceFormulation;
98         }
99     }
100 }

```

Code 13 rulesPork.yml - commit: [c1f1f27](#)

```

1 prefixes:
2   ex: http://www.example.com/
3   e: http://myontology.com/
4
5 mappings:
6   animalWeight:
7     sources:
8     - access: "CEP-2021-S1-WEIGHT.csv"
9       referenceFormulation: csv
10    - ["CEP-2021-S1-WEIGHT.csv~csv"]
11   s: ex:$(Animal ID)
12   po:
13   - [a, schema:Pork]
14   - p: schema:date
15     o:
16       value: $(Date)
17   - p: ex:weight
18     o:

```

```
19         value: $(Weight)
20         datatype: xsd:integer
```

[view original^a](#)

Once the `YamlMapping` class is implemented, the next step is to write the YAML file. To generate the YAML file by transforming all the content of the `YamlMapping` class into YAML, the `YAMLFactory` from the Jackson package is used. In summary, the code will look as follows:

Code 14 `WriteYamlStepDefs.java` - commit: [c1f1f27](#)

```
1  ...
2  public class WriteYamlStepDefs {
3      ...
4      public void generateTheYamlFile() throws IOException {
5          ...
6          YAMLFactory yamlFactory = new YAMLFactory();
7          yamlFactory.configure(YAMLGenerator.Feature.MINIMIZE_QUOTES, true);
8          yamlFactory.configure(YAMLGenerator.Feature.WRITE_DOC_START_MARKER, false);
9
10         ObjectMapper mapper = new ObjectMapper(yamlFactory);
11         mapper.writeValue(new File("src/main/resources/pork.yaml"), yamlMapping);
12     }
13 }
14 ...
```

Next, once the YAML file has been obtained, the `RMLio` frameworks must be called to transform it into Turtle format using `yarrml-parser`, and then pass it to the `rmlmapper` framework to transform it into RDF format.

Therefore, the `ExternalCommandExecutor` class has been created to execute the file conversion processes by locating the previously generated YAML file in its directory.

Code 15 `ExternalCommandExecutor.java` - commit: [c1f1f27](#)

```
1  ...
2  public void executeYARRRMLParser(String inputFileName, String outputFileName)
3      ↪ throws IOException, InterruptedException {
4      String inputPath =
5          ↪ "/mnt/c/Users/Zihan/Desktop/TFG/tab2kgwiz-api/src/main/resources/" +
6          ↪ inputFileName;
7      String outputPath =
8          ↪ "/mnt/c/Users/Zihan/Desktop/TFG/tab2kgwiz-api/src/main/resources/" +
9          ↪ outputFileName;
10
11      ProcessBuilder builder = new ProcessBuilder("ws1",
12          ↪ "/home/zihan/.nvm/versions/node/v21.5.0/bin/yarrml-parser", "-i",
13          ↪ inputPath, "-o", outputPath);
```

^aIn some PDF readers it doesn't work, in others you have to double click

```

9      Process process = builder.start();
10     int exitCode = process.waitFor();
11
12     if (exitCode != 0) {
13         throw new RuntimeException("YARRRML parser failed. Exit code: " +
14             ↪ exitCode);
15     }
16 }
17
18 public void executeRMLMapper(String outputFileName) throws IOException,
19 ↪ InterruptedException {
20     String outputPath =
21         ↪ "/mnt/c/Users/Zihan/Desktop/TFG/tab2kgwiz-api/src/main/resources/" +
22         ↪ outputFileName;
23     ProcessBuilder builder = new ProcessBuilder("java", "-jar",
24         ↪ "rmlmapper-6.5.1-r371-all.jar", "-m", outputPath);
25
26     Process process = builder.start();
27     int exitCode = process.waitFor();
28
29     if (exitCode != 0) {
30         throw new RuntimeException("RML mapper failed. Exit code: " + exitCode);
31     }
32 }
33 ...

```

7.4. Results

As a result of this iteration, the basic BDD-defined requirements for a user have been met, along with CRUD operations for each entity class, the generation of the YAML file using the Jackson library, and finally the local execution of the framework commands to transform the YAML file into TTL format and, subsequently, from TTL into linked data.

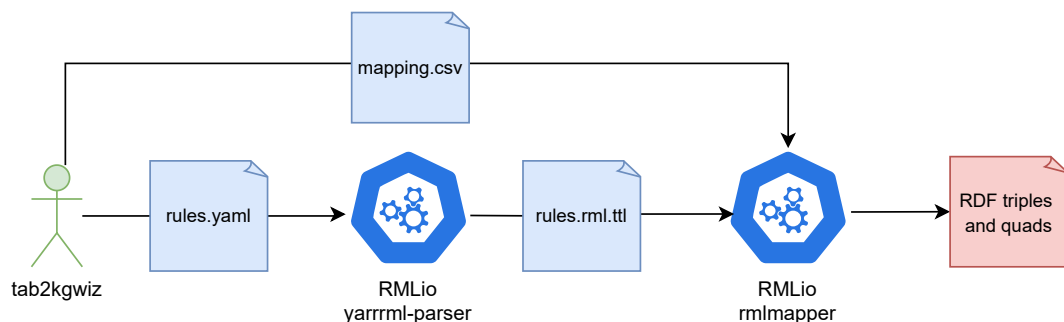


Figure 7.2: Initial diagram of linked data generation

Code 16 Archivo *rules.rml.ttl*

```

1 @prefix rr: <http://www.w3.org/ns/r2rml#>.
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

```

```

3 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
4 @prefix fnml: <http://semweb.mmlab.be/ns/fnml#>.
5 @prefix fno: <https://w3id.org/function/ontology#>.
6 @prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.
7 @prefix void: <http://rdfs.org/ns/void#>.
8 @prefix dc: <http://purl.org/dc/terms/>.
9 @prefix foaf: <http://xmlns.com/foaf/0.1/>.
10 @prefix rml: <http://semweb.mmlab.be/ns/rml#>.
11 @prefix ql: <http://semweb.mmlab.be/ns/ql#>.
12 @prefix : <http://mapping.example.com/>.
13 @prefix ex: <http://www.example.com/>.
14 @prefix e: <http://myontology.com/>.
15
16 :rules_000 a void:Dataset;
17     void:exampleResource :map_animalWeight_000.
18 :map_animalWeight_000 rml:logicalSource :source_000.
19 :source_000 a rml:LogicalSource;
20     rml:source "CEP-2021-S1-WEIGHT.csv";
21     rml:referenceFormulation ql:CSV.
22 :map_animalWeight_000 a rr:TriplesMap;
23     rdfs:label "animalWeight".
24 :s_000 a rr:SubjectMap.
25 :map_animalWeight_000 rr:subjectMap :s_000.
26 :s_000 rr:template "http://www.example.com/{Animal ID}".
27 :pom_000 a rr:PredicateObjectMap.
28 :map_animalWeight_000 rr:predicateObjectMap :pom_000.
29 :pm_000 a rr:PredicateMap.
30 :pom_000 rr:predicateMap :pm_000.
31 :pm_000 rr:constant rdf:type.
32 :pom_000 rr:objectMap :om_000.
33 :om_000 a rr:ObjectMap;
34     rr:constant <http://schema.org/Pork>;
35     rr:termType rr:IRI.
36 :pom_001 a rr:PredicateObjectMap.
37 :map_animalWeight_000 rr:predicateObjectMap :pom_001.
38 :pm_001 a rr:PredicateMap.
39 :pom_001 rr:predicateMap :pm_001.
40 :pm_001 rr:constant <http://schema.org/date>.
41 :pom_001 rr:objectMap :om_001.
42 :om_001 a rr:ObjectMap;
43     rml:reference "Date";
44     rr:termType rr:Literal.
45 :pom_002 a rr:PredicateObjectMap.
46 :map_animalWeight_000 rr:predicateObjectMap :pom_002.
47 :pm_002 a rr:PredicateMap.
48 :pom_002 rr:predicateMap :pm_002.
49 :pm_002 rr:constant ex:weight.
50 :pom_002 rr:objectMap :om_002.
51 :om_002 a rr:ObjectMap;
52     rml:reference "Weight";
53     rr:termType rr:Literal;
54     rr:datatype <http://www.w3.org/2001/XMLSchema#integer>.
55 :rules_000 void:exampleResource :map_animalWeight_001.
56 :map_animalWeight_001 rml:logicalSource :source_001.
57 :source_001 a rml:LogicalSource;
58     rml:source "CEP-2021-S1-WEIGHT.csv";

```

```

59     rml:referenceFormulation ql:CSV.
60 :map_animalWeight_001 a rr:TriplesMap;
61     rdfs:label "animalWeight".
62 :s_001 a rr:SubjectMap.
63 :map_animalWeight_001 rr:subjectMap :s_001.
64 :s_001 rr:template "http://www.example.com/{Animal ID}".
65 :pom_003 a rr:PredicateObjectMap.
66 :map_animalWeight_001 rr:predicateObjectMap :pom_003.
67 :pm_003 a rr:PredicateMap.
68 :pom_003 rr:predicateMap :pm_003.
69 :pm_003 rr:constant rdf:type.
70 :pom_003 rr:objectMap :om_003.
71 :om_003 a rr:ObjectMap;
72     rr:constant <http://schema.org/Pork>;
73     rr:termType rr:IRI.
74 :pom_004 a rr:PredicateObjectMap.
75 :map_animalWeight_001 rr:predicateObjectMap :pom_004.
76 :pm_004 a rr:PredicateMap.
77 :pom_004 rr:predicateMap :pm_004.
78 :pm_004 rr:constant <http://schema.org/date>.
79 :pom_004 rr:objectMap :om_004.
80 :om_004 a rr:ObjectMap;
81     rml:reference "Date";
82     rr:termType rr:Literal.
83 :pom_005 a rr:PredicateObjectMap.
84 :map_animalWeight_001 rr:predicateObjectMap :pom_005.
85 :pm_005 a rr:PredicateMap.
86 :pom_005 rr:predicateMap :pm_005.
87 :pm_005 rr:constant ex:weight.
88 :pom_005 rr:objectMap :om_005.
89 :om_005 a rr:ObjectMap;
90     rml:reference "Weight";
91     rr:termType rr:Literal;
92     rr:datatype <http://www.w3.org/2001/XMLSchema#integer>.

```

[view original^a](#)

^aIn some PDF readers it doesn't work, in others you have to double click

7.5. Timeline iteration 04

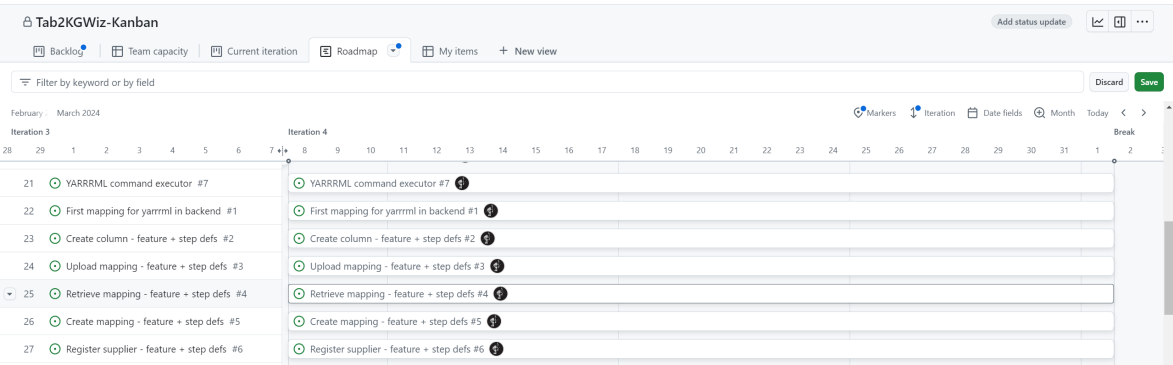


Figure 7.3: Timeline with the requirements of Iteration 04

8. Iteration 5: Login and Signup

8.1. Requeriments

After implementing the basic structure of the *supplier*, *mapping*, and *column* entities, Cucumber tests have been established to verify their functionality. Next, an authentication system for the *supplier* role will be implemented along with modifications to the execution of the RDF conversion commands.

Therefore, the proposed requirements for this iteration are:

- Back-end
 - Authentication for *Supplier*
 - YAML Prefix Mapper
 - Generation of YAML file and RDF conversion endpoints
 - Sequential execution of *rmlmapper* and *yarrml-parser*
- Front-end
 - Pages for signin and signup
 - Signin and signup middleware
 - Alert component

8.2. Design

In this iteration, the primary focus will be establishing authentication for *Supplier* using Token-Based Authentication. This security technique is used to verify identity and, unlike cookie- or session-based authentication, it improves flexibility, security, and the scalability of applications.

8.3. Implementation

To implement the token-based authentication system, the Spring Boot application has been configured using an authentication filter. This filter intercepts every HTTP request to verify if the JWT token is valid and, if so, sets the authentication in Spring's security context. Thus, this filter will be used to authenticate HTTP requests, verify the token's validity, extract user information, and establish authentication.

Code 17 AuthTokenFilter.java - commit: [8b09d14](#)

```
1 ...
2 @Override
3     protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
    ↪ response, FilterChain filterChain)
4         throws ServletException, IOException {
5         try {
6             String jwt = parseJwt(request);
7             if (jwt != null && jwtUtil.validateJwtToken(jwt)) {
8                 String username = jwtUtil.getUserNameFromJwtToken(jwt);
9                 UserDetails userDetails =
    ↪ userDetailsService.loadUserByUsername(username);
10                UsernamePasswordAuthenticationToken authentication = new
    ↪ UsernamePasswordAuthenticationToken(userDetails, null,
11                    userDetails.getAuthorities());
12                authentication.setDetails(new
    ↪ WebAuthenticationDetailsSource().buildDetails(request));
13                SecurityContextHolder.getContext().setAuthentication(authentication);
14            }
15        } catch (Exception e) {
16            log.error("Cannot set user authentication: {}", e);
17        }
18        filterChain.doFilter(request, response);
19    }
20 ...
```

The data structure (DTO) used to package the information from a server JWT response is shown below:

Code 18 JwtResponse.java - commit: [8b09d14](#)

```
1 ...
2 @Data
3 @NoArgsConstructor
4 public class JwtResponse {
5     private String token;
6     private String type = "Bearer";
7     private String id;
8     private String username;
9     private List<String> roles;
10 }
11 ...
```

The JWT utility class, which handles operations such as generating, extracting, and validating JWT tokens, is defined as follows:

Code 19 JwtUtil.java - commit: [8b09d14](#)

```
1 ...
2 @Component
3 @Log4j2
```

```

4 public class JwtUtil {
5     @Value("${tab2kgwiz.app.jwtSecret}")
6     private String jwtSecret;
7
8     @Value("${tab2kgwiz.app.jwtExpirationMs}")
9     private int jwtExpirationMs;
10
11     public String generateJwtToken(Authentication authentication) {
12         BasicUserDetailsImpl userPrincipal = (BasicUserDetailsImpl)
13             ↪ authentication.getPrincipal();
14         return
15             ↪ Jwts.builder().setSubject((userPrincipal.getUsername())).setIssuedAt(new
16             ↪ Date())
17             .setExpiration(new Date((new Date()).getTime() +
18             ↪ jwtExpirationMs)).signWith(SignatureAlgorithm.HS512, jwtSecret)
19             .compact();
20     }
21
22     public String getUserNamFromJwtToken(String token) {
23         return Jwts.parser().setSigningKey(jwtSecret).
24             parseClaimsJws(token).getBody().getSubject();
25     }
26
27     public boolean validateJwtToken(String authToken) {
28         try {
29             Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
30             return true;
31         } catch (SignatureException e) {
32             log.error("Invalid JWT signature: {}", e.getMessage());
33         } catch (MalformedJwtException e) {
34             log.error("Invalid JWT token: {}", e.getMessage());
35         } catch (ExpiredJwtException e) {
36             log.error("JWT token is expired: {}", e.getMessage());
37         } catch (UnsupportedJwtException e) {
38             log.error("JWT token is unsupported: {}", e.getMessage());
39         } catch (IllegalArgumentException e) {
40             log.error("JWT claims string is empty: {}", e.getMessage());
41         }
42         return false;
43     }
44 }
45 ...

```

To enable client registration and login, two endpoints have been created: one for signing in and one for signing up.

Code 20 SignInRequest.java - commit: [8b09d14](#)

```

1 @Data
2 public class SignInRequest {
3     private String username;
4     private String password;
5 }

```

Code 21 SignUpRequest.java - commit: [8b09d14](#)

```
1 @Data
2 public class SignUpRequest {
3     private String username;
4     private String email;
5     private String password;
6 }
```

Code 22 LoginController.java - commit: [8b09d14](#)

```
1 @PostMapping("/signin")
2 public ResponseEntity<?> login(@RequestBody SignInRequest signInRequest) {
3     Authentication authentication = authenticationManager.authenticate(new
4         ↪ UsernamePasswordAuthenticationToken(
5         signInRequest.getUsername(), signInRequest.getPassword()));
6     SecurityContextHolder.getContext().setAuthentication(authentication);
7
8     String jwt = jwtUtil.generateJwtToken(authentication);
9
10    BasicUserDetailsImpl userDetails = (BasicUserDetailsImpl)
11    ↪ authentication.getPrincipal();
12
13    List<String> roles = userDetails.getAuthorities().stream()
14    ↪ .map(item -> item.getAuthority())
15    ↪ .collect(Collectors.toList());
16    JwtResponse res = new JwtResponse();
17    res.setToken(jwt);
18    res.setId(userDetails.getId());
19    res.setUsername(userDetails.getUsername());
20    res.setRoles(roles);
21    return ResponseEntity.ok(res);
22 }
23
24 @PostMapping("/signup")
25 public ResponseEntity<String> signup(@RequestBody SignUpRequest signUpRequest) {
26     if (supplierRepository.existsById(signUpRequest.getUsername())) {
27         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("username is
28         ↪ already taken");
29     }
30     if (supplierRepository.existsByEmail(signUpRequest.getEmail())) {
31         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("email is
32         ↪ already taken");
33     }
34
35     String hashedPassword = passwordEncoder.encode(signUpRequest.getPassword());
36
37     Supplier supplier = new Supplier();
38     supplier.setUsername(signUpRequest.getUsername());
39     supplier.setEmail(signUpRequest.getEmail());
40     supplier.setPassword(hashedPassword);
41     supplierRepository.save(supplier);
42     return ResponseEntity.ok("Supplier registered success");
43 }
```

To generate the YAML file, an endpoint has been created that will receive the name of the mapping to be generated and execute the generator. In addition, another endpoint has been created for RDF generation; unlike the previous one, this does not receive any parameters and simply executes the converter using the locally generated YAML file.

Code 23 YamlController.java - commit: [8b09d14](#)

```

1  ...
2  @PostMapping("/yaml/generate")
3  public ResponseEntity<?> generateYaml(@RequestBody String mappingName) {
4      Authentication authentication =
5          ↪ SecurityContextHolder.getContext().getAuthentication();
6      if (authentication instanceof AnonymousAuthenticationToken) {
7          throw new NotAuthorizedException();
8      }
9      try {
10         yamlGenerator.generateYaml(mappingRepository, columnRepository,
11             ↪ mappingName);
12         return ResponseEntity.ok().body("Yaml file generated successfully");
13     } catch (IOException e) {
14         return ResponseEntity.badRequest().body("Error generating the Yaml file");
15     }
16
17     @PostMapping("/yaml/yarrmlmapper")
18     public ResponseEntity<?> yarrmlMapper() {
19         Authentication authentication =
20             ↪ SecurityContextHolder.getContext().getAuthentication();
21         if (authentication instanceof AnonymousAuthenticationToken) {
22             throw new NotAuthorizedException();
23         }
24         ExternalCommandExecutor executor = new ExternalCommandExecutor();
25         executor.executeYARRRMLParser();
26         return ResponseEntity.ok().body("YARRRML parser executed successfully");
27     }
28     ...

```

To obtain prefixes and the URI of a resource, the *PrefixCCMap* has been created. This code was extracted from the [rhizomerAPI](#) tool.

Code 24 PrefixCCMap.java - commit: [8b09d14](#)

```

1  ...
2  @Service
3  public class PrefixCCMap extends PrefixMapStd {
4      ...
5      public String prefixCCReverseLookup(String uri) {

```

```

6      switch (uri) {
7          case "http://www.w3.org/1999/02/22-rdf-syntax-ns#": return "rdf";
8          case "http://www.w3.org/2000/01/rdf-schema#": return "rdfs";
9          case "http://www.w3.org/2002/07/owl#": return "owl";
10         case "http://www.w3.org/2001/XMLSchema#": return "xsd";
11         case "http://xmlns.com/foaf/0.1/": return "foaf";
12         default: {
13             RestTemplate restTemplate = new RestTemplate();
14             try {
15                 String response = restTemplate.getForObject
16                 ("http://prefix.cc/reverse?uri={uri}&format={format}",
17                  ↪ String.class, uri, "txt");
18                 String[] pair = response.split("\\s");
19                 if (pair.length == 2)
20                     return pair[0];
21             } catch (RestClientException e) {
22                 logger.info("Prefix for URI {} not found in http://prefix.cc",
23                             ↪ uri);
24             }
25         }
26     }
27     return null;
28 }

```

To allow for the sequential execution of the frameworks for RDF generation, two process executors have been created.

Code 25 ExternalCommandExecutor.java - commit: [8b09d14](#)

```

1  ...
2  public class ExternalCommandExecutor {
3      public void executeYARRRMLParser() {
4          ProcessBuilder builder1 = new ProcessBuilder("docker", "run", "--rm", "-v",
5              ↪ "...", "rmlio/yarrml-parser:latest", "-i",
6              ↪ "/data/mappings.yarrml.yml");
7          // Redirect standard output (STDOUT) to a file
8          builder1.redirectOutput(new File(...));
9          try {
10             Process process = builder1.start();
11             process.waitFor(); // Wait for the container to finish
12             int exitCode = process.exitValue();
13
14             if (exitCode == 0) {
15                 ProcessBuilder builder2 = new ProcessBuilder("docker", "run", "--rm",
16                     ↪ "-v",
17                     ↪ "...",
18                     ↪ "-m", "rules.rml.ttl");
19
20                 // Redirect standard output (STDOUT) to a file
21                 builder2.redirectOutput(new File(
22                     ↪ "..."));

```

```

21         try {
22             Process process2 = builder2.start();
23             process2.waitFor(); // Wait for the container to finish
24             ...
25         }
26     }
27 }
28 }
29 }
30 ...

```

Below are fragments of the code for the login functionality and for the middleware that manages content access for authenticated users only.

Code 26 page.tsx - commit: [0a89ad5](#)

```

1     ...
2     export default function SignIn() {
3         ...
4         const [formData, setFormData] = useState({
5             username: "",
6             password: "",
7         });
8
9         const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
10             const { name, value } = event.target;
11             setFormData({ ...formData, [name]: value });
12         };
13
14         const handleSubmit = async () => {
15             const res = await fetch(`http://localhost:8080/signin`, {
16                 method: "POST",
17                 body: JSON.stringify(formData),
18                 headers: {
19                     "Content-Type": "application/json",
20                 },
21             });
22             if (res.ok) {
23                 const json = await res.json();
24                 localStorage.setItem("token", json.token);
25                 router.push("/home");
26             } else {
27                 alert("Bad credentials");
28             }
29         };
30         ...
31     }
32     ...

```



Code 27 middleware.ts - commit: [0a89ad5](#)

```
1  ...
2  export function middleware(request: NextRequest) {
3      const accessToken = request.cookies.get("accessToken")?.value;
4
5      if (request.nextUrl.pathname === "/home" && !accessToken) {
6          return NextResponse.redirect(new URL("/signin", request.url));
7      }
8
9      return NextResponse.next();
10 }
```

8.4. Results


As a result of this iteration, the endpoints for registration and login have been successfully implemented on the back-end, along with the corresponding forms on the front-end, achieving a JWT token-based authentication system. In the back-end, authentication is configured using an `AuthTokenFilter` that intercepts each HTTP request to verify the JWT token's validity and set the authentication in Spring's security context if the token is valid.


On the front-end, signin and signup forms have been implemented. These forms allow users to enter their credentials and, upon submission, communicate with the back-end endpoints. If authentication is successful, the received JWT token is stored in the browser's local storage and the user is redirected to the home page. Additionally, a middleware has been developed to manage access to restricted content by redirecting unauthenticated users to the signin page.



Sign In


Sign Up

 Username

 Password


Sign In


(a) Initial version of signin




Sign In

Sign Up

 Username

 Email address

 Password

Sign Up

[Already have an account?](#)

(b) Initial version of signup

8.5. Timeline iteration 05

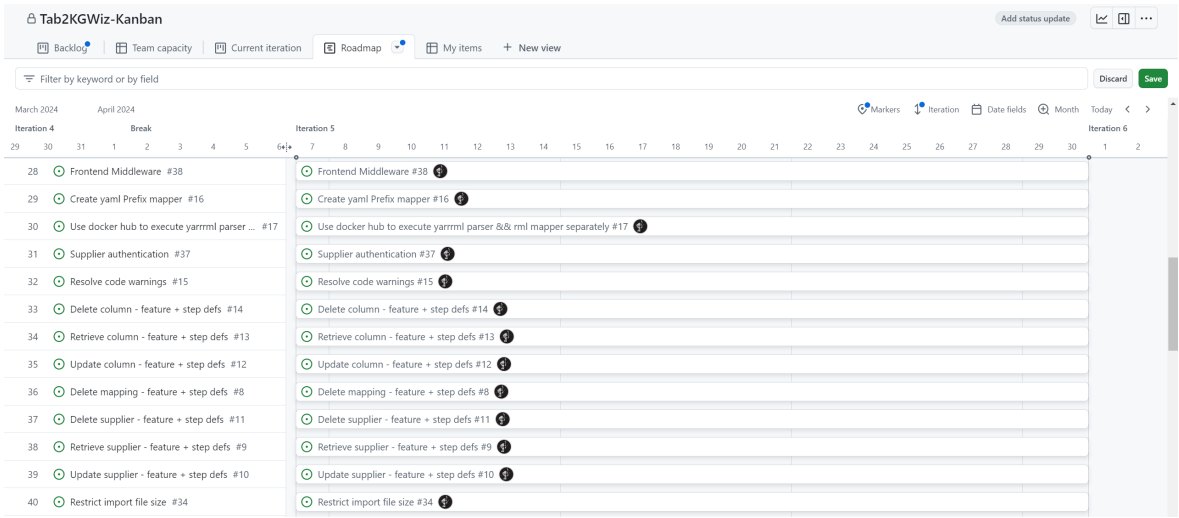


Figure 8.2: Timeline with the requirements of Iteration 05

9. Iteration 6: Authentication and RDF Generation

9.1. Requirements

Once the basic structure of the *supplier*, *mapping*, and *column* entities was implemented, Cucumber tests were established to verify their functionality. Next, an authentication system for the *supplier* role and modifications in the execution of the RDF conversion commands will be implemented.

Therefore, the proposed requirements for this iteration are:

- **Back-end**
 - Reuse the generated YAML file if it contains the same CSV headers.
 - Endpoint to generate RDF.
- **Front-end**
 - Configure the alert system so that it persists when the user navigates to another page.
 - Test the endpoint for generating linked data (RDF).
 - App Bar component.
 - Prevent double rendering of requests to the back-end.

9.2. Design

For the implementation of YAML file reuse, if the user has already generated a mapping, then if the user requests to generate the YAML file again, the previously generated file will be reused.

Accordingly, the implementation of the endpoint to generate RDF will consist of a Spring REST controller that packages the necessary content—that is, the previously generated YAML file and the CSV content—both as attachments for a POST request to the *generateLinkedData* route.

In order for the alert system to persist on the page, the React context provider will be used to share data between components without having to pass arguments or props manually through multiple levels of the component tree. Therefore, a *Snack-bar Provider* will be created at the root of the application, which will wrap all child components so that this alert component remains visible across all pages.

To consume the back-end endpoint for generating RDF, a POST request will be made to the endpoint `http://localhost:8080/mappings/${mappingId}/generate` using the *axios*

framework, which will allow for a successful request and response.

A problem was encountered during development: React, by default, double-renders all components (a behavior introduced to help detect potential bugs) in development mode. This means that API calls could be made twice because the *Strict Mode* is enabled. To prevent duplicate requests, data fetching will be performed using SWR—a library for caching data that avoids duplicating network requests by first checking the cache and only making a new request if the data is outdated.

9.3. Implementation

When the CSV header remains the same, the previously generated YAML file is reused.

Code 28 MappingController.java - commit: [28bd38e](#)

```
1 ...
2 Iterable<Mapping> mappings = mappingRepository.findAll();
3     for (Mapping m : mappings) {
4         // If the csv header is the same, we assume that the yaml file is the same
5         if (m.getYamlFile() != null &&
6             m.getFileContent()
7             .split("\n")[0].equals(mapping.getFileContent().split("\n")[0])) {
8             mapping.setYamlFile(m.getYamlFile());
9         }
10    }
11 ...
```

Endpoint to generate linked data.

Code 29 YamlController.java - commit: [28bd38e](#)

```
1 ...
2 @PostMapping("/generateLinkedData")
3     public ResponseEntity<?> generateLinkedData(@RequestParam("yamlFile")
4         ↳ MultipartFile yamlFile,
5         @RequestParam("csvFile") MultipartFile
6         ↳ csvFile) throws IOException {
7     byte[] linkedData = ExternalCommandExecutor.generateLinkedData(yamlFile,
8         ↳ csvFile);
9     return ResponseEntity.ok()
10        .contentType(MediaType.APPLICATION_JSON)
11        .header("Content-Disposition", "attachment; filename=linked-data.txt")
12        .body(linkedData);
13 }
```

To create a context provider for the Snackbar component, which is built using the MUI (Material UI) library, the context provider is established where it belongs.

Code 30 snackbar-provider.tsx - commit: [f8603e3](#)

```
1 ...
2 type SnackbarContextActions = {
3   showSnackBar: (text: string, typeColor: AlertColor) => void;
4 };
5
6 const SnackbarContext = createContext({} as SnackbarContextActions);
7
8 interface SnackbarContextProviderProps {
9   children: React.ReactNode;
10 }
11
12 const SnackbarProvider: React.FC<SnackbarContextProviderProps> = ({
13   children,
14 }) => {
15   const [open, setOpen] = React.useState<boolean>(false);
16   const [message, setMessage] = React.useState<string>("");
17   const [typeColor, setTypeColor] = React.useState<AlertColor>("info");
18
19   const showSnackBar = (text: string, color: AlertColor) => {
20     setMessage(text);
21     setTypeColor(color);
22     setOpen(true);
23   };
24
25   const handleClose = () => {
26     setOpen(false);
27     setTypeColor("info");
28   };
29
30   return (
31     <SnackbarContext.Provider value={{ showSnackBar }}>
32       <Snackbar
33         open={open}
34         autoHideDuration={6000}
35         anchorOrigin={{ vertical: "bottom", horizontal: "right" }}
36         onClose={handleClose}
37       >
38         <Alert onClose={handleClose} severity={typeColor}>
39           {message}
40         </Alert>
41       </Snackbar>
42     </SnackbarContext.Provider>
43   );
44 };
45 ...
46
```

Code 31 layout.tsx - commit: [f8603e3](#)

```
1 ...
2 export default function RootLayout({
```

```

3   children,
4   }: Readonly<{
5     children: React.ReactNode;
6   }> {
7     return (
8       <html lang="en">
9         <SnackBarProvider>
10          <body className={inter.className}>{children}</body>
11        </SnackBarProvider>
12      </html>
13    );
14  }

```

Below is a small piece of code demonstrating how SWR is used.

Code 32 page.tsx - commit: [f8603e3](#)

```

1
2  const useCreateMappingSWR = (
3    ...
4  ) => {
5    const shouldFetch = !!file; // Only fetch if file available
6
7    const { data, error } = useSWR(
8      shouldFetch ? `` : null,
9      async () => {
10        ...
11      },
12      {
13        revalidateOnFocus: false, // Avoid unnecessary refetches on focus
14      },
15    );
16
17    return { data, error };
18  };

```

9.4. Results

Reusing the YAML file when the CSV headers match has significantly optimized the RDF generation process. Note that this is considered an initial improvement; later, when the user is given the option to map the prefixes for the columns, it will be necessary not only to check the header but also to verify whether the mappings are identical. In that case, it will be necessary to decide if it is worth iterating over each mapping in search of duplicates or simply generating a new YAML file.

On the other hand, configuring the persistent alert system using a context provider for the Material UI SnackBar component has improved user interaction by ensuring that notifications remain visible regardless of navigation between pages. Additionally, the implementation of the SWR hook to handle data requests has prevented duplicate rendering during development.

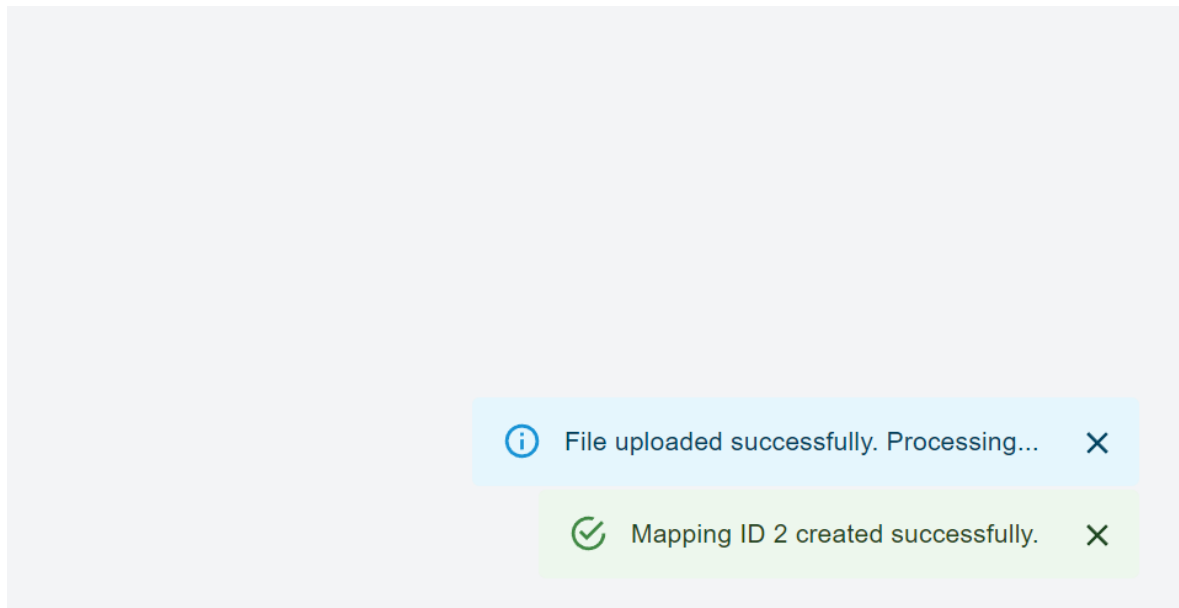


Figure 9.1: MUI Snackbar

9.5. Timeline for Iteration 06

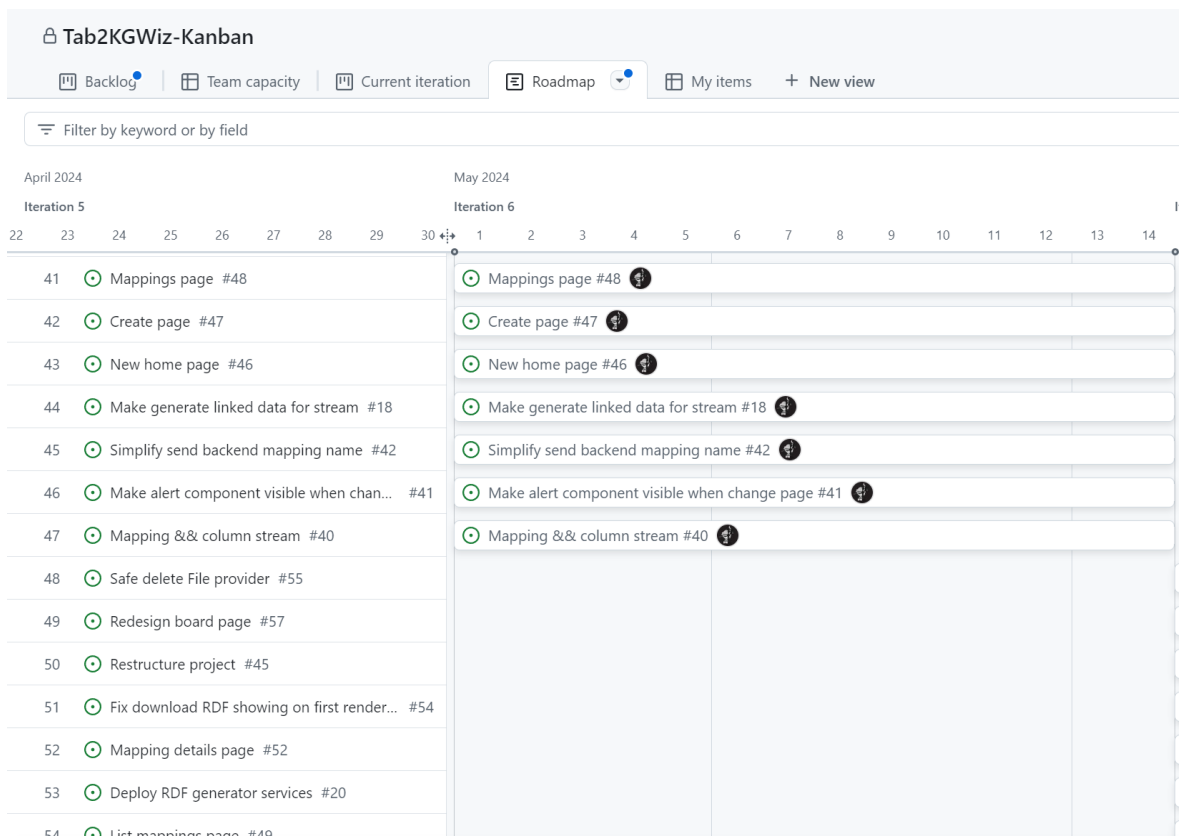


Figure 9.2: Timeline with the requirements of Iteration 06

10. Iteration 7: Restructuring and Various Front-End Components

10.1. Requirements

In this iteration, the focus is primarily on restructuring the project—redesigning the board page, the mapping details page, as well as various other front-end modifications—and deploying the RDF generator service described in Chapter [12.1.1](#).

Therefore, the proposed requirements for this iteration are:

- **Back-end**
 - Fix the infinite loop error
 - Retrieve the list of mapping IDs
- **Front-end**
 - Board mapping component
 - Mapping details page
 - Project restructuring

10.2. Design

In Spring Boot, JPA (Java Persistence API) annotations can be defined to specify the relationships between entities and control JSON serialization.

Code 33 Mapping.java

```
1 @Entity
2 @Data
3 @EqualsAndHashCode(callSuper = true)
4 public class Mapping extends UriEntity<Long> {
5     ...
6     @OneToMany(fetch = FetchType.LAZY, mappedBy = "columnBelongsTo", cascade =
7         ↪ CascadeType.ALL)
8     private List<cat.udl.eps.softarch.demo.domain.Column> columns;
9     ...
}
```

Code 34 Column.java

```
1 @Entity
2 @Data
3 @EqualsAndHashCode(callSuper = true)
4 public class Column extends UriEntity<Long> {
5     ...
6     @ManyToOne(cascade = CascadeType.ALL)
7     @JsonIdentityReference(alwaysAsId = true)
8     private Mapping columnBelongsTo;
9     ...
10 }
```

Note that, on one hand, the *OneToMany* annotation indicates a one-to-many relationship from Mapping to Column, and on the other hand, the *ManyToOne* annotation relates Column to Mapping. Since this is a bidirectional relationship, it can cause an infinite loop during JSON serialization because when serializing a Mapping entity, it includes the list of Columns, which in turn include a back-reference to the Mapping entity. One solution is to override the *toString* method, excluding the *columnBelongsTo* attribute from serialization to avoid the error.

To implement the details and board pages, Material UI (MUI) components will be used. This library provides a wide range of predefined and customizable components, such as [Stack](#), [Box](#), [Container](#), [Card](#), [Typography](#), [Button](#) y [Grid](#).

10.3. Implementation

Implement the *toString* method to avoid infinite loops.

Code 35 Mapping.java - commit: [9282ad8](#)

```
1 @Entity
2 @Data
3 @EqualsAndHashCode(callSuper = true)
4 public class Mapping extends UriEntity<Long> {
5     ...
6     @Override
7     public String toString() {
8         return "Mapping{" +
9             "id=" + id +
10             ", title='" + title + '\'' +
11             ", fileName='" + fileName + '\'' +
12             ", fileContent='" + fileContent + '\'' +
13             ", fileFormat='" + fileFormat + '\'' +
14             ", prefixesURIS='" + prefixesURIS + '\'' +
15             ", mainOntology='" + mainOntology + '\'' +
16             ", providedBy=" + providedBy +
17             ", yamlFile='" + yamlFile + '\'' +
18             '}';
19     }
20     ...
}
```

21 }

Code 36 Column.java - commit: [9282ad8](#)

```
1 @Entity
2 @Data
3 @EqualsAndHashCode(callSuper = true)
4 public class Column extends UriEntity<Long> {
5     ...
6     @Override
7     public String toString() {
8         return "Column{" +
9             "id=" + id +
10            ", title='" + title + '\'' +
11            ", dataType='" + dataType + '\'' +
12            ", ontologyURI='" + ontologyURI + '\'' +
13            ", ontologyType='" + ontologyType + '\'' +
14            '}';
15     }
16     ...
17 }
```

The first version of the *UserBoard* component was built by constructing a basic table that lists all user-created mappings ordered by ID.

Code 37 page.tsx (Board) - commit: [5f8bfd7](#)

```
1 const UserBoard: React.FC<Props> = (props): JSX.Element => {
2     ...
3     return (
4         <>
5             <Container maxWidth="xl">
6                 <Box
7                     display="flex"
8                     sx={{
9                         height: "90vh",
10                        bgcolor: "#f3f4f6",
11                        marginTop: "2vh",
12                        borderRadius: "10px",
13                        boxShadow: 1,
14                    }}
15                 >
16                     <Stack spacing={2} sx={{ marginTop: "2vh", marginLeft: "2vh" }}>
17                         <h2 className="mb-2 text-lg font-semibold text-gray-900 dark:text-white">
18                             Mapping ID
19                         </h2>
20                         <ul className="max-w-md space-y-1 text-gray-500 list-disc list-inside
21                             ⇨ dark:text-gray-400">
22                             {mappingsIds.length === 0 ? (
23                                 <li>There are no mappings available.</li>
24                             ) : (
25                                 <>
```

```

25         {mappingsIds.map((mappingId) => (
26             <li key={mappingId}>
27                 <Link href={ ... }>
28                     <a>Mapping ID: {mappingId}</a>
29                 </Link>
30             </li>
31         ))}
32     </>
33 })
34 </ul>
35 </Stack>
36 </Box>
37 </Container>
38 </>
39 );
40 ...
41 }

```

Code 38 page.tsx (Details) - commit: 5f8bfd7

```

1  const MappingDetailsPage: React.FC<{
2    params: { mappingsId: string };
3  }> = ({ params }): JSX.Element => {
4    ...
5    return (
6      <>
7        <>
8          <Typography
9            variant="h5"
10           sx={{
11             marginTop: "3vh",
12             marginLeft: "10vh",
13             marginRight: "5vh",
14             color: "#000000",
15           }}
16         >
17           Mapping Details
18         </Typography>
19         <br />
20         <Stack
21           direction="row"
22           spacing={2}
23           sx={{ ... }}
24         >
25           >
26           <Card sx={{ width: 345, height: 800 }}>
27             <CardHeader title="Columns" />
28             <CardContent>
29               <Typography variant="body2" color="text.secondary">
30                 Show list of columns in the mapping.
31               </Typography>
32             </CardContent>
33
34             <Button

```

```

35     size="small"
36     variant="outlined"
37     startIcon={<EditNoteOutlinedIcon />}
38     onClick={handleEdit}
39     sx={{
40       marginLeft: 2,
41     }}
42   >
43     <span>Edit</span>
44   </Button>
45
46   <List
47     sx={{
48       width: "100%",
49       maxWidth: 400,
50       bgcolor: "background.paper",
51       position: "relative",
52       overflow: "auto",
53       maxHeight: 700,
54       "& ul": { padding: 0 },
55     }}
56   >
57     {data?.columns.map(({ id, title, dataType, ontologyURI }) => (
58       ...
59     ))}
60   </List>
61 </Card>
62
63 <Grid container>
64   <Grid item xs={12} marginBottom={1}>
65     <Card sx={{ maxWidth: 1300, height: 390 }}>
66       <CardHeader title="Mapping details" />
67       <CardContent>
68         <Typography variant="body2" color="text.secondary">
69           Indicate the details of the mapping, including the generated
70           Yaml File and uploaded CSV File.
71         </Typography>
72       </CardContent>
73       <Stack spacing={2}>
74         ...
75       </Stack>
76       <Stack spacing={2}>
77         ...
78       </Stack>
79       </Stack>
80     </Card>
81   </Grid>
82   <Grid item xs={12}>
83     <Card sx={{ maxWidth: 1300, height: 400 }}>
84       <CardHeader title="RDF" />
85       <CardContent>
86         <Typography variant="body2" color="text.secondary"></Typography>
87       </CardContent>
88       <CardActions disableSpacing>
89         <IconButton aria-label="share">

```

```

90         <FavoriteIcon />
91       </IconButton>
92     </CardActions>
93   </Card>
94 </Grid>
95 </Grid>
96 </Stack>
97 );
98 ...
99 }

```

10.4. Results

On the back-end, the infinite loop error caused by the bidirectional relationship between the *Mapping* and *Column* entities was resolved.

On the front-end, the first version of the *UserBoard* component was developed, which presents a basic table listing all user-created mappings ordered by ID. Later, a second version was implemented using the complete set of MUI components to create a table incorporating all key mapping characteristics.

Additionally, a details page was implemented that provides more detailed information about a specific mapping, including accessible data such as the uploaded CSV file, the generated YAML file, a list of columns with their ontologies, and an option to generate RDF.

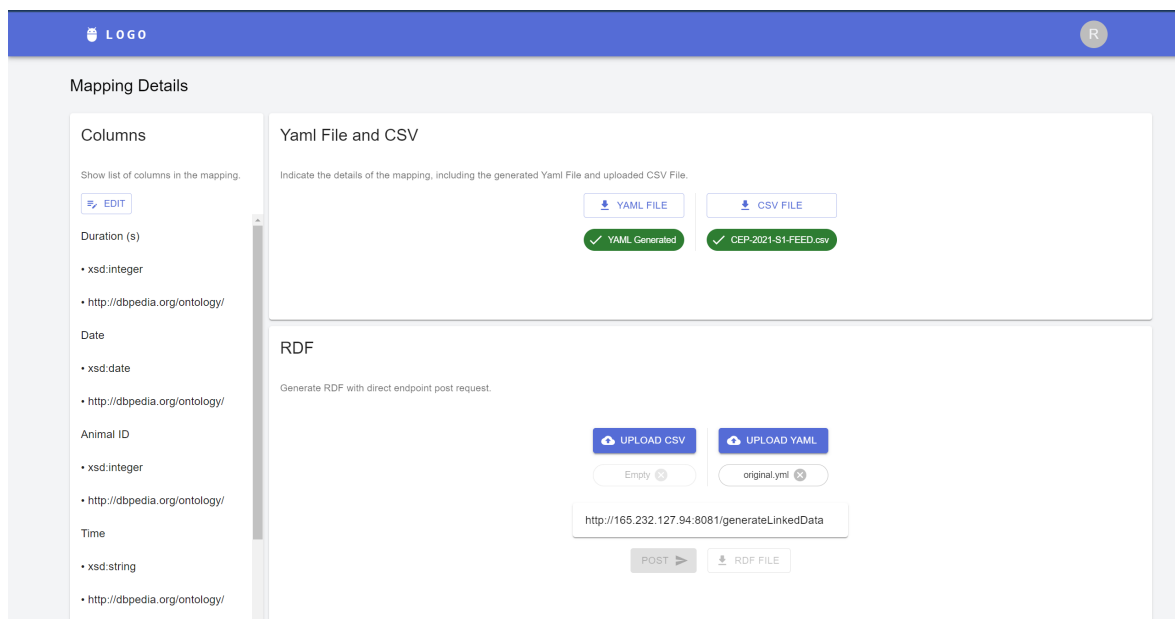


Figure 10.1: Mapping details page

10.5. Timeline for Iteration 07

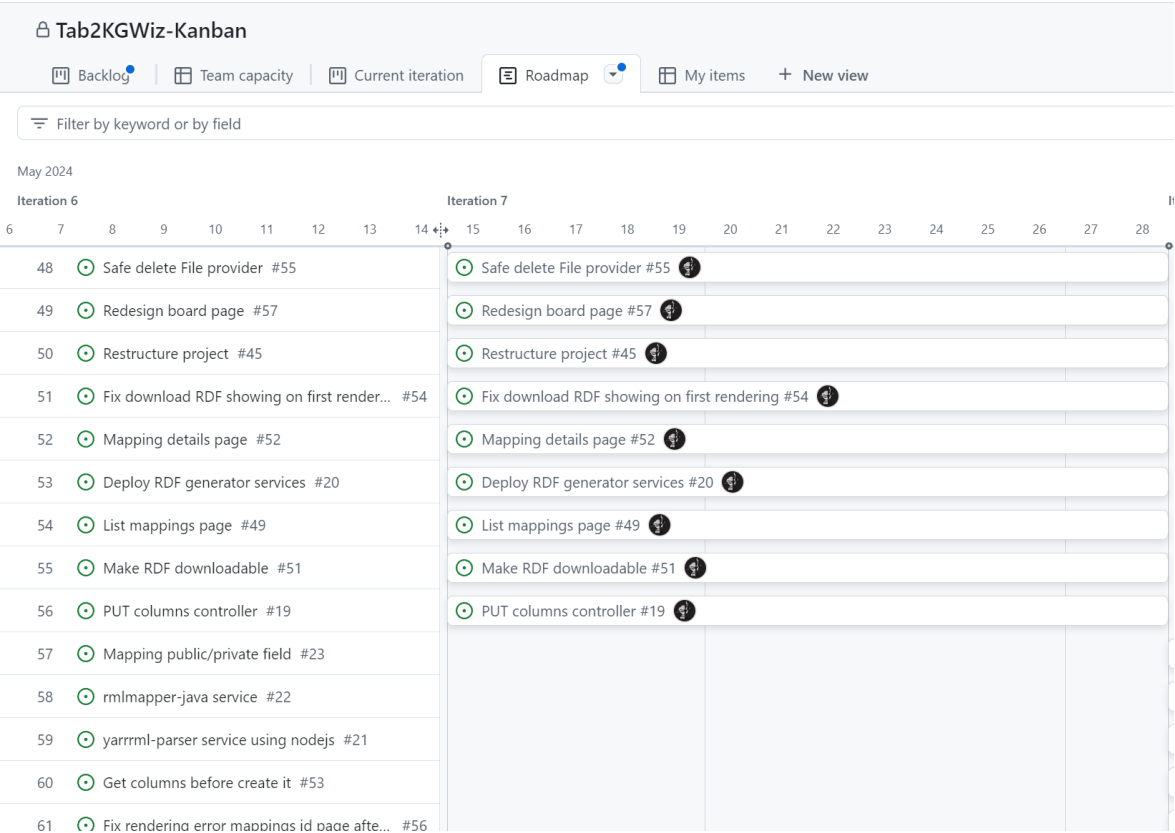


Figure 10.2: Timeline with the requirements of [Iteration 07](#)

11. Iteration 8: RML Microservices

11.1. Requirements

In this iteration, the focus is on creating the services to convert YAML to TTL and TTL to RDF using node.js for yarrrml-parser and rmlmapper-java with [rmlmapper-webapi-js](#) created by RMLio in order to split the two services.

Therefore, the proposed requirements for this iteration are:

- **Back-end**
 - Public or private mappings
 - rmlmapper-java service
 - yarrrml-parser service
- **Front-end**
 - Retrieve columns before creating them

11.2. Results

Now, users can choose to define their mappings as public or private ([8545259](#)). This functionality provides them with greater control over the accessibility and security of their data, allowing them to protect sensitive information or share knowledge as necessary.

Furthermore, the services [rmlmapper-java](#) and [yarrrml-parser](#) can be found in their respective repositories.

11.3. Timeline for Iteration 08

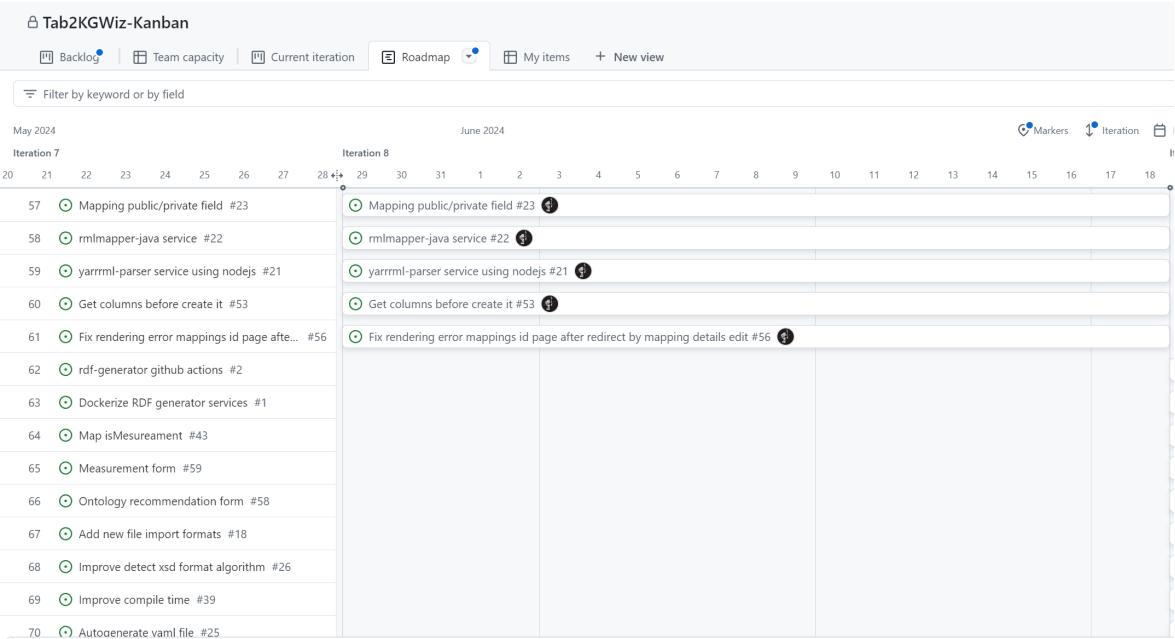


Figure 11.1: Timeline with the requirements of [Iteration 08](#)

12. Iteration 9: Final Results

12.1. RDF Generator

Once the columns that the user wishes to assign an ontology to have been modeled using ontologies, the tool's back-end will transform the modeled entities into YAML language, which will then be sent to the APIs to generate linked data. The schema for this process is shown in the following figure:

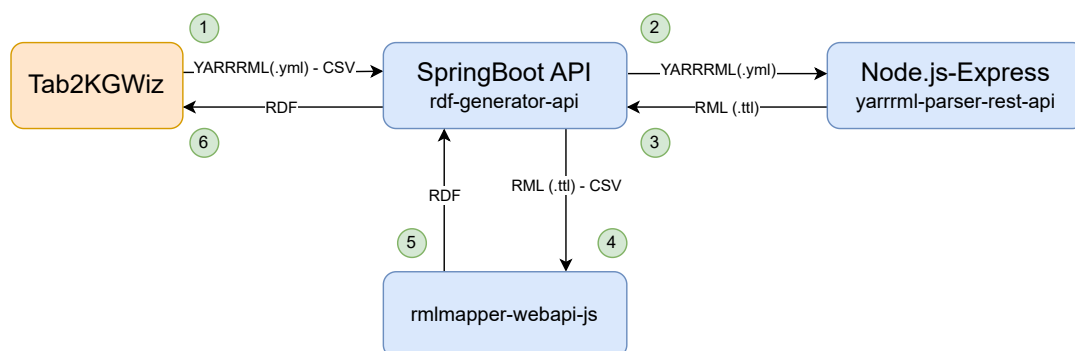


Figure 12.1: Final API communication schema

The application `rdf-generator-api` is a Spring Boot API that makes a request to the `yarrml-parser-rest-api`, which returns the data transformed into Turtle (Terse RDF Triple Language). This output is then sent to `rmlmapper-webapi-js` to convert it into RDF (Resource Description Framework).

12.1.1 Deployment

The API was initially deployed on DigitalOcean servers, a provider of virtual private servers. Through the GitHub student account, 200 dollars were obtained to freely create machines hosted on external servers. Taking advantage of this opportunity for a first deployment, an Ubuntu machine was created on one of the DigitalOcean servers, and the service was started with the `systemctl start` command, which launches the Spring Boot API.

Since this service needs to make requests to both `yarrml-parser-rest-api` and `rmlmapper-webapi-js`, two separate machines were created for them as well.

As a total of three machines was initially required, which was an unnecessary cost, the solution was optimized by dockerizing all three APIs. Consequently, the images (`zihancr/rdfgenerator`, `zihancr/yarrml`, and `zihancr/rmlmapper`) were pushed to DockerHub. Then, using `docker compose`, the entire service was deployed on the Ubuntu machine. Now, all three services run on a single machine.

Code 39 docker-compose.yml

```
1 services:
2   yarrml:
3     image: "zihancr/yarrml"
4     restart: always
5
6   rmlmapper:
7     image: "zihancr/rmlmapper"
8     depends_on:
9       - yarrml
10    restart: always
11
12   rdfgenerator:
13     image: "zihancr/rdfgenerator"
14     depends_on:
15       - yarrml
16       - rmlmapper
17     restart: always
18     ports:
19       - "8081:8081"
```

Once the services have been dockerized, the APIs and the application can run on any operating system. Kubernetes was then used to deploy the service on a Kubernetes cluster on one of UDL's servers.

12.2. Timeline for Iteration 09

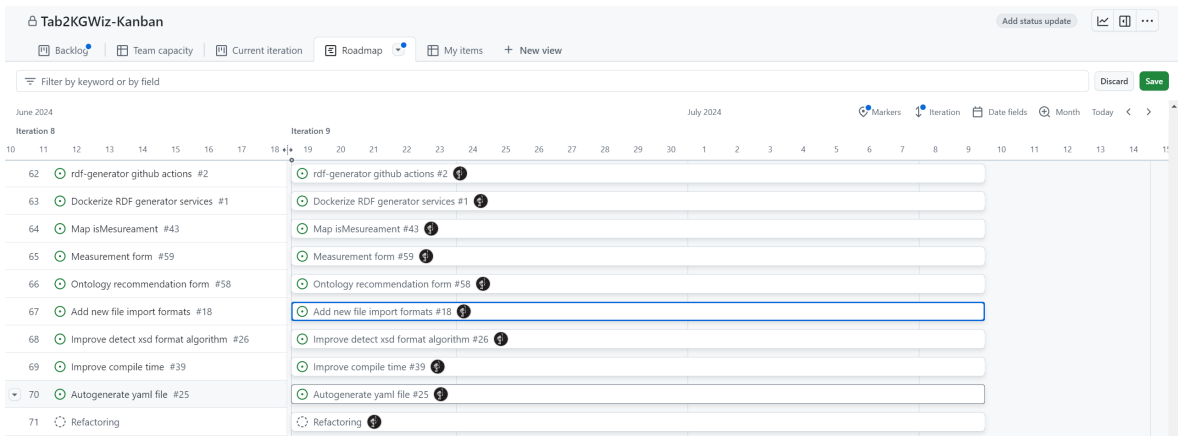


Figure 12.2: Timeline with the requirements of *iteración 09*

12.3. Final Application

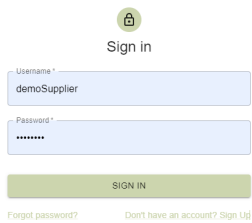
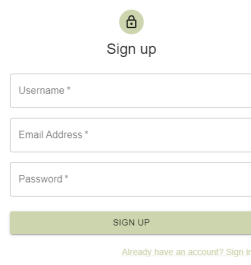


Figure 12.3: Login page



A sign-up form with a green header bar. The header contains a lock icon and the text "Sign up". Below the header are three input fields: "Username *", "Email Address *", and "Password *". A green "SIGN UP" button is at the bottom. Below the button is a link: "Already have an account? Sign in".

Figure 12.4: Signup page



Two green buttons stacked vertically. The top button has a plus icon and the text "CREATE NEW MAPPING". The bottom button has a grid icon and the text "BOARD".

Figure 12.5: Home page

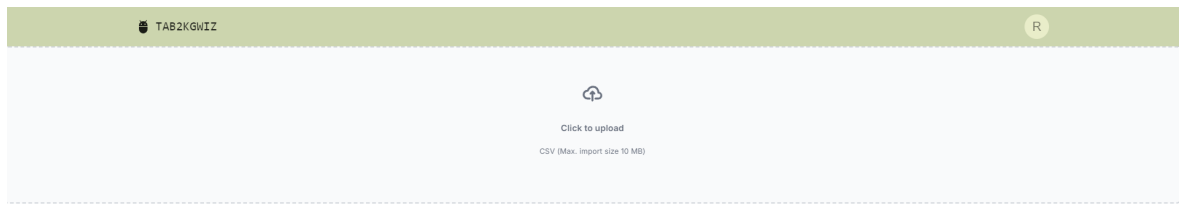


Figure 12.6: File upload page

Pen Number	Animal ID	Date	Time	Duration (s)	Feed (g)	Animal Weight (g)
4	982091062894196	2021-03-16	01:08	42	0	16500
4	982091062894196	2021-03-16	01:09	45	1	16300
4	982091062894196	2021-03-16	13:05	70	0	15300
4	982091062894196	2021-03-17	10:44	5	0	14400
4	982091062894196	2021-03-17	10:45	9	6	14500
4	982091062894196	2021-03-17	10:45	50	14	14400
4	982091062894196	2021-03-17	10:46	2	0	0
4	982091062894196	2021-03-17	10:46	22	0	16900
4	982091062894196	2021-03-17	10:47	25	0	14700
4	982091062894196	2021-03-17	10:48	340	0	20800

Figure 12.7: Mapping table

Ontology mapping form

For each column, select the column type and search for the ontology

Pen Number

Column Type: XSD Data Type:

Animal ID

Column Type: XSD Data Type: Type of entity:

Related to:

Date

Column Type: XSD Data Type: What is it measuring?:

Has unit: Has timestamp: Is measurement of:

Measurement made by:

Time

Column Type: XSD Data Type:

Duration (s)

Column Type: XSD Data Type:

CLOSE

Figure 12.8: Dropdown for ontology mapping

TAB2KGWIZ R

Mappings List

<input type="checkbox"/> Mapping ID ↑	Title	File name	Created by	Availability
<input type="checkbox"/> 1	CEP-2021-S1-FEED.csv	CEP-2021-S1-FEED.csv	demoSupplier	Private
<input type="checkbox"/> 2	CEP-2021-S1-WEIGHT.csv	CEP-2021-S1-WEIGHT.csv	demoSupplier	Private
<input type="checkbox"/> 3	20231215_TreePhysiology_Shestakova_climatedatasetv1.csv	20231215_TreePhysiology_Shestakova_climatedatasetv1.csv	demoSupplier	Public

Rows per page: 5 1-3 of 3 < >

☐ Dense padding

Figure 12.9: Table with all user-created and public mappings

TAB2KGWIZ

R

Details

EDIT

Mapping info

ID: 1

Title: CEP-2021-S1-FEED.csv

File format: csv

Created by: demoSupplier

Availability: Private

Yaml File and CSV

Indicate the details of the mapping, including the generated Yaml File and uploaded CSV File.

YAML FILE

CSV FILE

YAML Not Generated

CEP-2021-S1-FEED.csv

Columns

Show list of columns in the mapping.

Pen Number

Animal ID

Date

Time

Duration (s)

Feed (g)

RDF

Generate RDF with direct endpoint post request.

UPLOAD CSV

UPLOAD YAML

Empty

Empty

http://104.248.240.80:8081/generateLinkedData

POST

RDF FILE

Figure 12.10: Mapping details page

12.3.1 Final Back-end Class Diagram

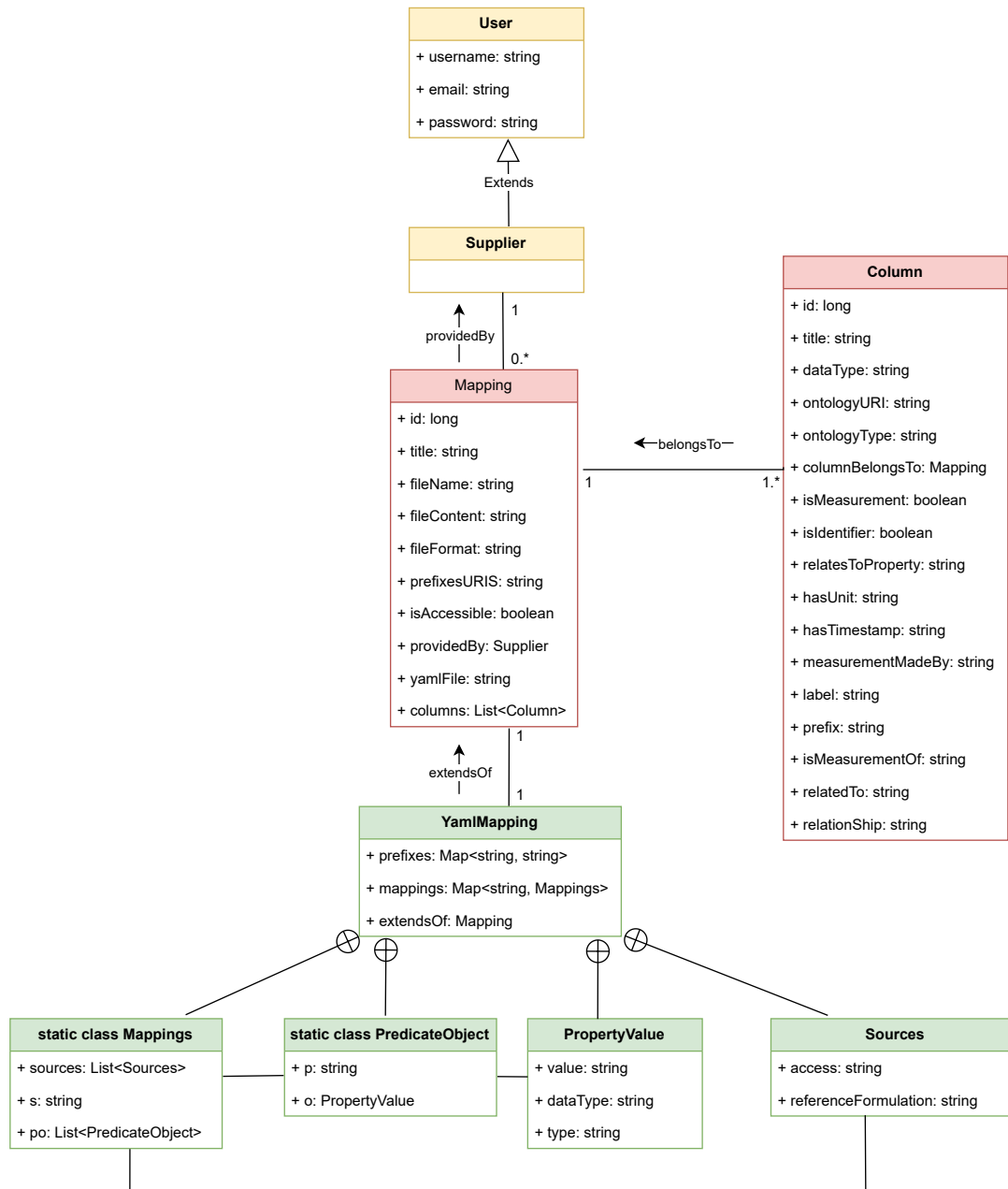


Figure 12.11: Final back-end class diagram

12.3.2 Example of a Mapping

A mapping is to be constructed using data from a pig farm. This mapping will measure the number of pens where the pig is located, the animal's ID, the date, the duration during which the animal is feeding, and the animal's weight at that moment. The following table is provided:

Table 12.1

CSV file containing information about the pig and the farmyard

Pen Number	Animal ID	Date	Time	Duration (s)	Feed (g)	Animal Weight (g)
4	982091062894196	2021-03-16	01:08	42	0	16500
4	982091062894196	2021-03-16	01:09	45	1	16300
4	982091062894196	2021-03-16	13:05	70	0	15300
4	982091062894196	2021-03-17	10:44	5	0	14400
4	982091062894196	2021-03-17	10:45	9	6	14500
4	982091062894196	2021-03-17	10:45	50	14	14400
4	982091062894196	2021-03-17	10:46	2	0	0
4	982091062894196	2021-03-17	10:46	22	0	16900
4	982091062894196	2021-03-17	10:47	25	0	14700
4	982091062894196	2021-03-17	10:48	340	0	20800

Thus, the CSV file is uploaded to the application, which displays the following table:

Pen Number	Animal ID	Date	Time	Duration (s)	Feed (g)	Animal Weight (g)
4	982091062894196	2021-03-16	01:08	42	0	16500
4	982091062894196	2021-03-16	01:09	45	1	16300
4	982091062894196	2021-03-16	13:05	70	0	15300
4	982091062894196	2021-03-17	10:44	5	0	14400
4	982091062894196	2021-03-17	10:45	9	6	14500
4	982091062894196	2021-03-17	10:45	50	14	14400
4	982091062894196	2021-03-17	10:46	2	0	0
4	982091062894196	2021-03-17	10:46	22	0	16900
4	982091062894196	2021-03-17	10:47	25	0	14700
4	982091062894196	2021-03-17	10:48	340	0	20800

Showing 1 - 10 of 10 results

SAVE GENERATE RDF RDF FILE

Figure 12.12: CSV table imported

Once the file is uploaded, the ontologies are mapped. At this point, the user must provide the corresponding information via a form.

- **Pen Number**
 - **Column Type:** Identifier
 - **XSD Data Type:** integer (generated automatically)
 - **Type of entity:** gn:S.CRRL (pen(s))
 - **Related to:** Not mapped
- **Animal ID**
 - **Column Type:** Identifier

- **XSD Data Type:** integer (generated automatically)
- **Type of entity:** dbo:animal (animal)
- **Related to:** Pen Number
- **Relationship:** gn:locatedIn (located in)
- **Feed (g)**
 - **Column Type:** Measurement
 - **XSD Data Type:** integer (generated automatically)
 - **What is it measuring?:** sioc:feed (feed)
 - **Has unit:** unit:GM (Gram)
 - **Has timestamp:** Date
 - **Is measurement of:** Animal ID
 - **Measurement made by:** sensor
- **Animal Weight (g)**
 - **Column Type:** Measurement
 - **XSD Data Type:** integer (generated automatically)
 - **What is it measuring?:** schema:weight (weight)
 - **Has unit:** unit:GM (Gram)
 - **Has timestamp:** Date
 - **Is measurement of:** Animal ID
 - **Measurement made by:** sensor

Thus, the information is filled out as follows:

Ontology mapping form

For each column, select the column type and search for the ontology

Pen Number		
Column Type Identifier	XSD Data Type integer	Type of entity gn:S.CRRL (corral(s))
Related to Not mapped		
Animal ID		
Column Type Identifier	XSD Data Type integer	Type of entity dbo:animal (animal)
Related to Pen Number	Relationship gn:locatedIn (located in)	
Date		
Column Type Not mapped	XSD Data Type date	
Time		
Column Type Not mapped	XSD Data Type string	
Duration (s)		
Column Type Not mapped	XSD Data Type integer	

Figure 12.13: Ontology form 01

Feed (g)		
Column Type Measurement	XSD Data Type integer	What is it measuring? sioc:feed (feed)
Has unit unit:GM (Gram)	Has timestamp Date	Is measurement of Animal ID
Measurement made by sensor		
Animal Weight (g)		
Column Type Measurement	XSD Data Type integer	What is it measuring? schema:weight (weight)
Has unit unit:GM (Gram)	Has timestamp Date	Is measurement of Animal ID
Measurement made by sensor		

CLOSE

Figure 12.14: Ontology form 02

The YAML instructions generated according to the ontology mapping are as follows:

Code 40 exampleFinal.yml

```

1 prefixes:
2   schema: http://schema.org/
3   dbo: http://dbpedia.org/ontology/
4   unit: http://qudt.org/vocab/unit/
5   gs1: https://gs1.org/voc/
6   saref: https://saref.etsi.org/core/
7   gn: http://www.geonames.org/ontology#
8   xsd: http://www.w3.org/2001/XMLSchema#
9   s4agri: https://saref.etsi.org/saref4agri/
10  rdfs: http://www.w3.org/2000/01/rdf-schema#
11  base: https://tab2kgwiz.udl.cat/
12  sioc: http://rdfs.org/sioc/ns#
13 mappings:
```

```

14 animalweightg:
15   sources:
16     - access: mappings.csv
17       referenceFormulation: csv
18   po:
19     - p: rdf:type
20       o:
21         value: saref:Measurement
22         type: iri
23     - p: rdfs:label
24       o:
25         value: weight
26     - p: saref:relatesToProperty
27       o:
28         value: schema:weight
29         type: iri
30     - p: saref:hasValue
31       o:
32         value: $(Animal Weight \ (g\))
33         datatype: xsd:integer
34     - p: saref:hasUnit
35       o:
36         value: unit:GM
37         type: iri
38     - p: saref:hasTimestamp
39       o:
40         value: $(Date)
41         datatype: xsd:date
42     - p: saref:measurementMadeBy
43       o:
44         value: base:sensor
45         type: iri
46     - p: saref:isMeasurementOf
47       o:
48         value: base:$(Animal ID)
49         type: iri
50 feedg:
51   sources:
52     - access: mappings.csv
53       referenceFormulation: csv
54   po:
55     - p: rdf:type
56       o:
57         value: saref:Measurement
58         type: iri
59     - p: rdfs:label
60       o:
61         value: feed
62     - p: saref:relatesToProperty
63       o:
64         value: sioc:feed
65         type: iri
66     - p: saref:hasValue
67       o:
68         value: $(Feed \ (g\))
69         datatype: xsd:integer

```

```

70     - p: saref:hasUnit
71       o:
72         value: unit:GM
73         type: iri
74     - p: saref:hasTimestamp
75       o:
76         value: $(Date)
77         datatype: xsd:date
78     - p: saref:measurementMadeBy
79       o:
80         value: base:sensor
81         type: iri
82     - p: saref:isMeasurementOf
83       o:
84         value: base:$(Animal ID)
85         type: iri
86   animalid:
87     sources:
88     - access: mappings.csv
89       referenceFormulation: csv
90     s: base:$(Animal ID)
91     po:
92     - p: rdf:type
93       o:
94         value: dbo:animal/animal
95         type: iri
96     - p: rdfs:label
97       o:
98         value: $(Animal ID)
99     - p: s4agri:isLocatedIn
100      o:
101        value: base:$(Pen Number)
102        type: iri
103   pennumber:
104     sources:
105     - access: mappings.csv
106       referenceFormulation: csv
107     s: base:$(Pen Number)
108     po:
109     - p: rdf:type
110       o:
111         value: gn:S.CRRL/corral(s)
112         type: iri
113     - p: rdfs:label
114       o:
115         value: $(Pen Number)

```

[view original^a](#)

And the generated RDF is:

^aIn some PDF readers it doesn't work, in others you have to double click

Code 41 rdf.txt

```
1  _:0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
2  _:0 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
3  _:0 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
4  _:0 <https://saref.etsi.org/core/hasValue>
   ↪ "16500"^^<http://www.w3.org/2001/XMLSchema#integer>.
5  _:0 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
6  _:0 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.
7  _:0 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
8  _:0 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
9  _:1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
10 _:1 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
11 _:1 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
12 _:1 <https://saref.etsi.org/core/hasValue>
   ↪ "16300"^^<http://www.w3.org/2001/XMLSchema#integer>.
13 _:1 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
14 _:1 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.
15 _:1 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
16 _:1 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
17 _:2 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
18 _:2 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
19 _:2 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
20 _:2 <https://saref.etsi.org/core/hasValue>
   ↪ "15300"^^<http://www.w3.org/2001/XMLSchema#integer>.
21 _:2 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
22 _:2 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.
23 _:2 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
24 _:2 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
25 _:3 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
26 _:3 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
27 _:3 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
28 _:3 <https://saref.etsi.org/core/hasValue>
   ↪ "14400"^^<http://www.w3.org/2001/XMLSchema#integer>.
29 _:3 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
30 _:3 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
31 _:3 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
32 _:3 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
33 _:4 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
```

```

34 _:4 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
35 _:4 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
36 _:4 <https://saref.etsi.org/core/hasValue>
   ↪ "14500"^^<http://www.w3.org/2001/XMLSchema#integer>.
37 _:4 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
38 _:4 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
39 _:4 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
40 _:4 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
41 _:5 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
42 _:5 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
43 _:5 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
44 _:5 <https://saref.etsi.org/core/hasValue>
   ↪ "14400"^^<http://www.w3.org/2001/XMLSchema#integer>.
45 _:5 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
46 _:5 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
47 _:5 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
48 _:5 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
49 _:6 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
50 _:6 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
51 _:6 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
52 _:6 <https://saref.etsi.org/core/hasValue>
   ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
53 _:6 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
54 _:6 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
55 _:6 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
56 _:6 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
57 _:7 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
58 _:7 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
59 _:7 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
60 _:7 <https://saref.etsi.org/core/hasValue>
   ↪ "16900"^^<http://www.w3.org/2001/XMLSchema#integer>.
61 _:7 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
62 _:7 <https://saref.etsi.org/core/hasTimestamp>
   ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
63 _:7 <https://saref.etsi.org/core/measurementMadeBy>
   ↪ <https://tab2kgwiz.udl.cat/sensor>.
64 _:7 <https://saref.etsi.org/core/isMeasurementOf>
   ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
65 _:8 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
   ↪ <https://saref.etsi.org/core/Measurement>.
66 _:8 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
67 _:8 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
68 _:8 <https://saref.etsi.org/core/hasValue>
   ↪ "14700"^^<http://www.w3.org/2001/XMLSchema#integer>.

```



```

69 _:8 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
70 _:8 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
71 _:8 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
72 _:8 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
73 _:9 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
74 _:9 <http://www.w3.org/2000/01/rdf-schema#label> "weight".
75 _:9 <https://saref.etsi.org/core/relatesToProperty> <http://schema.org/weight>.
76 _:9 <https://saref.etsi.org/core/hasValue>
    ↪ "20800"^^<http://www.w3.org/2001/XMLSchema#integer>.
77 _:9 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
78 _:9 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
79 _:9 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
80 _:9 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
81 _:10 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
82 _:10 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
83 _:10 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
84 _:10 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
85 _:10 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
86 _:10 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.
87 _:10 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
88 _:10 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
89 _:11 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
90 _:11 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
91 _:11 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
92 _:11 <https://saref.etsi.org/core/hasValue>
    ↪ "1"^^<http://www.w3.org/2001/XMLSchema#integer>.
93 _:11 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
94 _:11 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.
95 _:11 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
96 _:11 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
97 _:12 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
98 _:12 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
99 _:12 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
100 _:12 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
101 _:12 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
102 _:12 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-16"^^<http://www.w3.org/2001/XMLSchema#date>.

```

```

103 _:12 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
104 _:12 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
105 _:13 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
106 _:13 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
107 _:13 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
108 _:13 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
109 _:13 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
110 _:13 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
111 _:13 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
112 _:13 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
113 _:14 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
114 _:14 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
115 _:14 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
116 _:14 <https://saref.etsi.org/core/hasValue>
    ↪ "6"^^<http://www.w3.org/2001/XMLSchema#integer>.
117 _:14 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
118 _:14 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
119 _:14 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
120 _:14 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
121 _:15 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
122 _:15 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
123 _:15 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
124 _:15 <https://saref.etsi.org/core/hasValue>
    ↪ "14"^^<http://www.w3.org/2001/XMLSchema#integer>.
125 _:15 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
126 _:15 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
127 _:15 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
128 _:15 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
129 _:16 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
130 _:16 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
131 _:16 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
132 _:16 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
133 _:16 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
134 _:16 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
135 _:16 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
136 _:16 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.

```

```

137 _:17 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
138 _:17 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
139 _:17 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
140 _:17 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
141 _:17 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
142 _:17 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
143 _:17 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
144 _:17 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
145 _:18 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
146 _:18 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
147 _:18 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
148 _:18 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
149 _:18 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
150 _:18 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
151 _:18 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
152 _:18 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
153 _:19 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <https://saref.etsi.org/core/Measurement>.
154 _:19 <http://www.w3.org/2000/01/rdf-schema#label> "feed".
155 _:19 <https://saref.etsi.org/core/relatesToProperty> <http://rdfs.org/sioc/ns#feed>.
156 _:19 <https://saref.etsi.org/core/hasValue>
    ↪ "0"^^<http://www.w3.org/2001/XMLSchema#integer>.
157 _:19 <https://saref.etsi.org/core/hasUnit> <http://qudt.org/vocab/unit/GM>.
158 _:19 <https://saref.etsi.org/core/hasTimestamp>
    ↪ "2021-03-17"^^<http://www.w3.org/2001/XMLSchema#date>.
159 _:19 <https://saref.etsi.org/core/measurementMadeBy>
    ↪ <https://tab2kgwiz.udl.cat/sensor>.
160 _:19 <https://saref.etsi.org/core/isMeasurementOf>
    ↪ <https://tab2kgwiz.udl.cat/982091062894196>.
161 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <http://dbpedia.org/ontology/animal/animal>.
162 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
163 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
164 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <http://dbpedia.org/ontology/animal/animal>.
165 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
166 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
167 <https://tab2kgwiz.udl.cat/982091062894196>
    ↪ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↪ <http://dbpedia.org/ontology/animal/animal>.

```

```

168 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
169 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
170 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
171 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
172 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
173 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
174 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
175 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
176 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
177 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
178 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
179 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
180 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
181 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
182 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
183 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
184 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
185 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
186 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
187 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
188 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://dbpedia.org/ontology/animal/animal>.
189 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <http://www.w3.org/2000/01/rdf-schema#label> "982091062894196".
190 <https://tab2kgwiz.udl.cat/982091062894196>
    ↳ <https://saref.etsi.org/saref4agri/isLocatedIn> <https://tab2kgwiz.udl.cat/4>.
191 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
192 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".

```

```

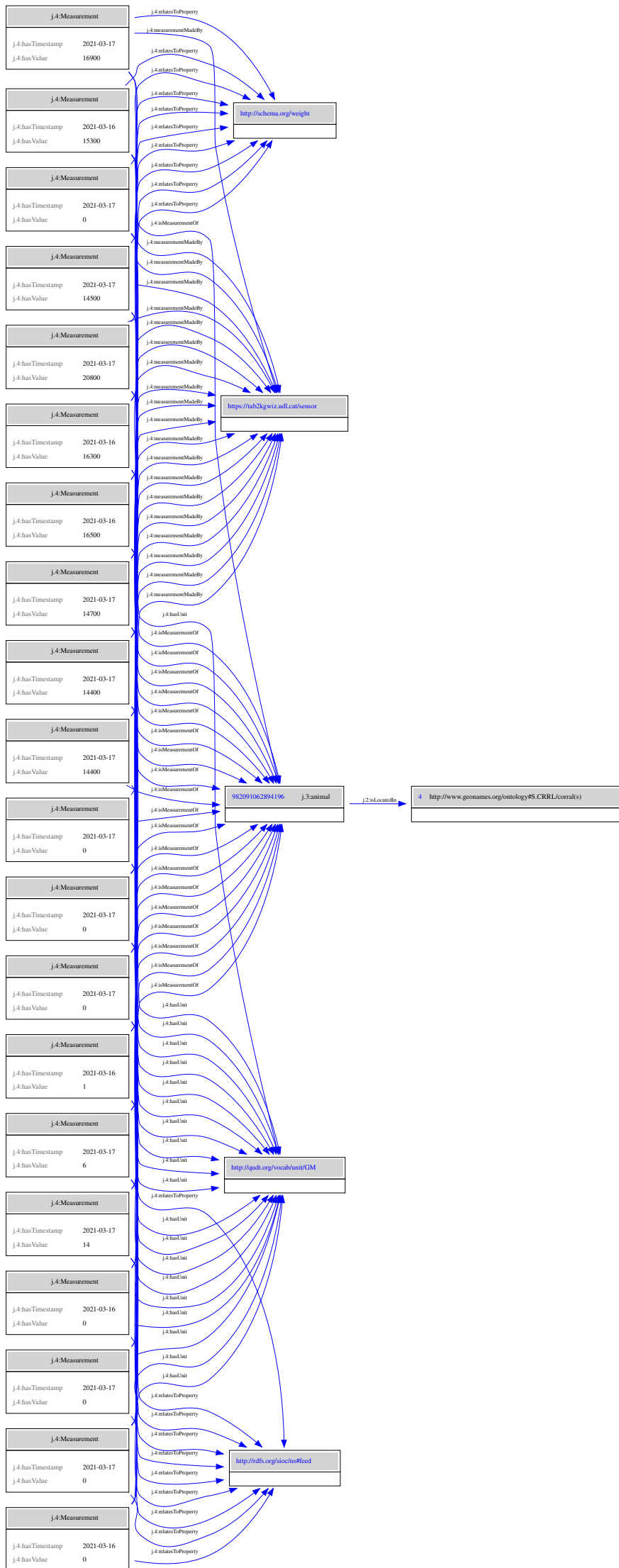
193 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
194 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
195 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
196 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
197 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
198 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
199 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
200 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
201 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
202 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
203 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
204 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
205 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
206 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
207 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
208 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".
209 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    ↳ <http://www.geonames.org/ontology#S.CRRL/corral(s)>.
210 <https://tab2kgwiz.udl.cat/4> <http://www.w3.org/2000/01/rdf-schema#label> "4".

```

[view original^a](#)

Using the RDF-to-SVG conversion tool ([rdf2svg](#)), the following graph is obtained:

^aIn some PDF readers it doesn't work, in others you have to double click



Part III

Epilogue

This part includes conclusions, future work and a bibliography.

13. Conclusions

This project has successfully met the proposed objectives, demonstrating its effectiveness in automating the process of mapping tabular data to a graph model with ontologies for generating linked data. Throughout the development, the technical and conceptual challenges inherent to transforming tabular data into an appropriate semantic format were effectively addressed. The resulting tool not only facilitates this process but also aligns with the Semantic Web standards, significantly contributing to data interoperability in scientific and technological environments.

One of the key aspects of the application's development was its iterative approach, which allowed for the progressive improvement of the system's functionality and usability. Each iteration focused on refining essential components such as data mapping, the user interface, and the generation of YAML and RDF files, resulting in a final tool that not only meets technical requirements but is also accessible and easy to use for end users. It is important to note that although only systematic tests with real users have been conducted, a future task will be to evaluate usability and accessibility more thoroughly. The inclusion of elements such as the Board, the notification component, and the details page clearly exemplify how user experience has been optimized, allowing for a more intuitive and efficient management of the generated mappings.

The development was not without its challenges, particularly in transforming data to RDF format using the RMLio tools and integrating them as services in the final application, as well as in representing and managing the user-generated mappings as persistent entities in the back-end. However, these challenges were successfully overcome.

Despite the objectives achieved, areas for improvement and opportunities for future developments have been identified. Among these, the possibility of integrating new APIs to expand the functionality of the system stands out, as well as the continuous improvement of the user interface to make it even more intuitive and accessible.

Furthermore, there is significant potential to expand the system's mapping capabilities by including a greater variety of ontologies and data formats, thereby further increasing its applicability in various contexts. In addition, there is the opportunity to automatically detect the ontologies that best fit the context of the uploaded data, thereby improving the tool's ability to assist users in defining mappings.

13.1. Future Work

As future work or potential improvements, beyond the ones mentioned above, the system should be extended to accept more data formats in addition to CSV files, which was initially proposed as an objective but was postponed due to the complexity of implementing this functionality.

On the other hand, it is important to implement an intelligent ontology recommendation system that suggests the most suitable ontologies according to the context of the

uploaded data. This would enhance the program's intelligence, facilitating the creation of more precise and appropriate mappings for the users. Additionally, it remains to include a tool for loading the RDF code into linked data repositories and presenting it to the user in a visual and interactive manner to facilitate the evaluation of the results.

Aside from improvements in the user interface, components should be implemented to allow registered users to modify their personal information. Finally, although the RDF transformation service has been deployed, the application itself is pending deployment on one of the cloud servers.

Bibliography

- [1] media_1576e188070800895e8cf8e74a4a2582b9b20caaf.png (750×422). https://business.adobe.com/blog/basics/media_1576e188070800895e8cf8e74a4a2582b9b20caaf.png?width=750&format=png&optimize=medium.
- [2] our-process2.jpg (995×577). <https://www.bespokesoftwaredevelopment.com/blog/wp-content/uploads/2022/08/our-process2.jpg>.
- [3] waterfall-v-agile-iron-triangle-v03.png (1038×461). <https://www.process.st/wp-content/uploads/2024/02/waterfall-v-agile-iron-triangle-v03.png>.
- [4] W3C. Built-in datatypes. (n.d.), 2004. <https://www.w3.org/TR/xmlschema-2/>.
- [5] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, et al. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, March 2016. <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4792175/>.
- [6] Diferencias entre metodologías Agile y Waterfall | Blog UE, 2022. <https://universidadeuropea.com/blog/agile-vs-waterfall/>.
- [7] Kamran Ahmed. Developer Roadmaps. <https://roadmap.sh/>.
- [8] RDF 1.1 Concepts and Abstract Syntax. <https://www.w3.org/TR/rdf11-concepts/#section-blank-nodes>.
- [9] Web semántica - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Web_sem%C3%A1ntica.
- [10] Datos FAIR, May 2024. https://es.wikipedia.org/w/index.php?title=Datos_FAIR&oldid=160002606.
- [11] Linked Open Vocabularies. <https://lov.linkeddata.es/dataset/lov/vocabs/schema>.
- [12] XML Schema Date/Time Datatypes. https://www.w3schools.com/xml/schema_datatypes_date.asp.
- [13] XML Schema Part 2: Datatypes Second Edition. <https://www.w3.org/TR/xmlschema-2/>.
- [14] How to Process YAML with Jackson | Baeldung, April 2019. <https://www.baeldung.com/jackson-yaml>.
- [15] Eugen Paraschiv. Jackson - Custom Serializer | Baeldung, January 2014. <https://www.baeldung.com/jackson-custom-serialization>.
- [16] Spring Security: Upgrading the Deprecated WebSecurityConfigurerAdapter | Baeldung, July 2022. <https://www.baeldung.com/spring-deprecated-websecurityconfigureradapter>.

- [17] How To: Authentication Middleware in Next.js. <https://blog.openreplay.com/how-to--authentication-middleware-in-nextjs>.
- [18] Converting a Spring MultipartFile to a File | Baeldung, June 2020. <https://www.baeldung.com/spring-multipartfile-to-file>.
- [19] Arijit Sarkar. Bidirectional Relationship Using @OneToMany/@ManyToOne Annotation In Spring Boot, October 2023. <https://medium.com/@arijit83work/bidirectional-relationship-using-onetomany-manytoone-annotation-in-spring-boot-3b9>
- [20] GSSwain. SpringBoot + DigitalOcean Droplets, August 2021. <https://gsswain.medium.com/springboot-digitalocean-droplets-410b8bbc6fe6>.
- [21] Rajendra Prasad Padma. “What the CORS” ft Spring Boot & Spring Security, March 2021. <https://rajendraprasadpadma.medium.com/what-the-cors-ft-spring-boot-spring-security-562f24d705c9>.
- [22] Diferencias entre metodologías Agile y Waterfall | Blog UE, March 2022. <https://universidadeuropea.com/blog/agile-vs-waterfall/>.
- [23] <https://rml.io/yarrml/tutorial/getting-started/#complete-yarrml-document>.
- [24] Chaewonkong. TypeScript, What Good For?, April 2023. <https://medium.com/@chaewonkong/typescript-what-good-for-8240dc9173c7>.
- [25] Why You Should Use Java for Backend Development, February 2023. <https://www.freecodecamp.org/news/java-for-backend-web-development/>.
- [26] Next.js Benefits: 5 Key Reasons for Your Next Front-End Development. <https://www.intuz.com/blog/5-reasons-why-you-should-use-next.js-for-your-front-end-development>.
- [27] Vincent Okonkwo. Different Between Next.js, Vite And Solid.js, July 2023. <https://medium.com/@okonkwovincent63/different-between-next-js-vite-and-solid-js-3fb69d33c07f>.
- [28] neversaint. Python Pandas equivalent in JavaScript, May 2023. <https://stackoverflow.com/q/30610675/17248314>.
- [29] danfo.readCSV | Danfo.js, October 2022. https://danfo.jsdata.org/api-reference/input-output/danfo.read_csv.
- [30] Linked Open Vocabularies (LOV). <https://lov.linkeddata.es/dataset/lov>.
- [31] Animal. <https://saref.etsi.org/saref4agri/Animal>.
- [32] About: weight (g). <https://dbpedia.org/ontology/weight>.
- [33] How to write unit tests in JavaScript with Jest, July 2021. <https://dev.to/dstrekelj/how-to-write-unit-tests-in-javascript-with-jest-2e83>, abstract = Unit testing is an important and often overlooked part of the development process. It is considered...
- [34] Ionut Banu. Build Spring Boot Docker image using multi-stage Dockerfile (2), March 2021. <https://ionutbanu.medium.com/>

[build-spring-boot-docker-image-using-multi-stage-dockerfile-2-13b9f1e89393](#),
abstract = In this article I am showing you a very efficient method to create a
Docker image for a Spring Boot application. This is the second one in...

- [35] Docs Home :: DigitalOcean Documentation. <https://docs.digitalocean.com/>.
- [36] Autenticación vs autorización: Diferencias y ejemplos de cómo funcionan. <https://www.redeszone.net/tutoriales/seguridad/diferencias-autenticacion-autorizacion/>.

Part IV

Annexes

Glossary of terms

APIs An API (short for Application Programming Interface) is a set of definitions and protocols that allows different applications or systems to communicate with one another.. [10](#)

heterogeneous data Heterogeneous data are sets of data that come from different sources or systems and have varied formats, structures, or types.. [6](#)

knowledge graphs Also known as knowledge graphs in English, they are data structures that represent information through nodes and edges, where nodes represent entities (such as people, places, concepts, etc.) and edges represent the relationships between these entities. These graphs are used to organize and model knowledge in a manner that reflects the reality of the world or a specific domain, allowing both humans and machines to understand and process the information effectively.. [2](#)

ontologies An ontology is a formal definition of types, properties, and relationships among entities that truly or fundamentally exist for a particular domain of discourse. [3](#)

semantic modelling Semantic modeling is a technique used in computer science and information science to represent and organize data in a way that captures the meaning or semantics of the information within a specific domain.. [2](#)

Acronyms

FAIR Findability Accessibility Interoperability Reusability. [3](#)

HTTP Hypertext Transfer Protocol. [16](#)

IDE Integrated Development Environment. [16](#)

RML RDF Mapping Language. [3](#)

TDD Test Driven Development. [23](#)