Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

European Master in Software Engineering

Master Thesis

Updating a Spring Data Extraction and Analysis Pipeline

Author: Virginia González Díez

June, 2025

This Master Thesis has been deposited in ETSI Informáticos de la Universidad Politécnica de Madrid.

Master Thesis

European Master in Software Engineering

Title: Updating a Spring Data Extraction and Analysis Pipeline
June / 2025

Author: Virginia González Díez

Supervisor:

Alejandro Rodríguez González

Full Professor

Doctor in Computer Science

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software

Universidad Politécnica de Madrid

Co-supervisor:

Lucía Prieto Santamaría

Assistant Professor

Doctor in Software, Systems and Computing

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería del Software

Universidad Politécnica de Madrid

Abstract

Understanding human diseases and discovering new therapeutic solutions remain central challenges in biomedical research. Diseases are not caused by isolated factors, but by complex networks of genes, proteins, and phenotypes interacting in dynamic systems. Traditional research methods, which often analyse isolated elements, are insufficient for capturing this complexity. In response, the field of network medicine has emerged, using systems-based models to explore disease relationships and support drug repurposing (DR), a promising strategy that identifies new uses for existing drugs.

Within this context, the DISNET (DISease understanding and drug repurposing through complex NETworks) platform was developed to construct a multilayer biomedical knowledge base. It integrates data from structured databases and unstructured online sources, which includes Wikipedia and Mayo Clinic, using Natural Language Processing (NLP) techniques (specifically, the MetaMap tool) to extract and validate phenotypic information. This phenotypic layer is then connected with biological and pharmacological layers, enabling researchers to explore disease relationships and DR hypotheses through a public API and web interface.

However, despite its scientific relevance, the DISNET system faced technical issues by early 2023. Its core medical text extraction pipelines had stopped functioning, and the platform relied on outdated technologies including Java 8 and early Spring Boot versions. The microservices architecture, with 16 Docker containers requiring independent configuration, had become unmanageable for the current small development team composed by one or two active developers. Additionally, the web interface was partially non-functional, and the system's documentation was obsolete. These issues compromised the platform's reliability and functionality.

This Master's Thesis addresses these problems through the restoration and improvement of the DISNET platform. The primary goals were to restore the system's core functionality, reduce technical complexity, and improve maintainability. Specific objectives included reactivating the Medical Term Extraction Process (MTEP) pipelines for Wikipedia and Mayo Clinic, upgrading the software stack, consolidating the system's databases, and redesigning the web interface. An in-depth analysis of the DISNET architecture and design was conducted to understand the system's complexity and to develop an improvement plan.

The results of this work include a fully operational and simplified DISNET system. The extraction workflows now run reliably on the unified infrastructure, the number of containers has been reduced for ease of deployment, and the web interface has been improved with updated content. These improvements not only ensure that the system can be maintained by a small development team but also restore its role as a valuable, open-access research tool. With accurate and up-to-date phenotypic data, DISNET once again enables researchers to explore disease similarities, track knowledge evolution over time, and generate novel DR hypotheses.

Table of Contents

List of Figures

List of Tables

Abbreviations

CTB	Centre for Biomedical Technology
CUI	Concept Unique Identifier
DAP	DISNET Authorization Process
DISNET	Drug repositioning and DISease understanding through complex NETworks creation and analysis
DLEP	Disease List Extraction Process
DR	Drug Repurposing
DRIVE	Data Driven Disease Visualization and Drug Repurposing
DSVP	Data Storage and Validation Procedure
DWA	DISNET Web Application
ER	Entity-Relationship
GDA	Gene-disease association
GNN	Graph Neural Network
HDN	Human Disease Network
JDBC	Java Database Connectivity
JVM	Java Virtual Machine
LOC	Lines Of Code
LOD	Linked Open Data
MAPI	Main API
MCTE	Mayo Clinic Text Extraction
MEDAL	Medical Data Analytics Laboratory
MM	MetaMap
MMC	MetaMap Client
MMI	MetaMap Inserts
MMS	MetaMap Server
MTEP	Medical Term Extraction Process
NAS	Network Attached Storage
NeDRex	Network-based Drug Repurposing and exploration

NER	Named Entity Recognition
NLP	Natural Language Processing
PMTE	PubMed Text Extraction
POS	Part-Of-Speech
RDBMS	Relational Database Management System
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
STV	Semantic Type Validation
TE	Text Extraction
TVP	Term Validation Process
UMLS	Unified Medical Language System
UPM	Universidad Politécnica de Madrid
VAL	Validation
VDA	Variant-disease association
WSD	Word Sense Disambiguation
WTE	Wikipedia Text Extraction

1 Introduction

Understanding human diseases remains a central challenge in biomedical research today. Diseases, whether common like cancer and diabetes or rare genetic disorders, do not stem from a single cause. Instead, they are driven by complex networks of genes, proteins, environmental factors, and lifestyle choices interacting in dynamic ways [1]. These tangled networks are difficult to understand with traditional scientific methods, which often look at individual elements in isolation rather than the system as a whole. This has led to a growing need for new approaches that can look at the bigger picture and help us better understand how diseases are developed.

Advances in biomedical technology have made it possible to collect huge amounts of biological data from laboratory experiments, clinical investigations, healthcare records, and the internet of things [2]. One field that has provided an opportunity for future significant advances is the research of multi-omics data, which include the study of DNA (genomics), RNA (transcriptomics), proteins (proteomics), metabolites (metabolomics), epigenetic modifications (epigenomics) and drugs (pharmacogenomics) [3]. However, this wealth of data introduces another challenge: its volume, heterogeneity, and varying quality.

To address the limitations of traditional reductionist methods and to take advantage of the growing size of biomedical data, the field of **network medicine** emerged as a powerful approach [1]. Network medicine employs a systems approach, representing biological components, such as genes, proteins, metabolites, and diseases, as nodes in a network, with links representing relationships between them. This model has led to uncover hidden patterns and connections that may not be apparent when studying biological components in isolation.

In this context, the **Human Disease Network** (HDN) was constructed [4]. It is a network based model where nodes represent diseases, and links connect diseases that share at least one associated gene. Its main purpose is to map the genetic landscape of human diseases, reveal hidden connections between diseases and provide a holistic view of how diseases are related.

The construction of disease networks is used not only to improve our understanding of human diseases but it also plays a critical role in advancing **Drug Repurposing** (DR). DR is an area of research focused on identifying new therapeutic uses for existing drugs. Traditional drug development is a time-consuming process, often spanning 10 to 15 years, and is expensive, with costs for bringing a new compound to market reaching up to $2.5 billion [5]. Moreover, it carries a high risk of failure, as only about 10% of new drugs gain market approval [6]. DR offers a compelling solution, providing a faster and a more cost-effective approach thanks to existing drugs have already undergone extensive research and testing.

In this context, **DISNET**[1] **(*Drug repositioning and DISease understanding through complex NETworks creation and analysis*)** was created in 2017 by the Medical Data Analytics Laboratory (MEDAL)[2] located at the Centre for Biomedical Technology (CTB)[3] of the Universidad Politécnica de Madrid (UPM)[4], under the leadership of Dr. Alejandro Rodríguez González. DISNET project was

[1] *https://disnet.ctb.upm.es/*
[2] *https://medal.ctb.upm.es/*
[3] *http://www.ctb.upm.es/*
[4] *https://www.upm.es/*

under grant “RTI2018-094576-A-I00” from the Spanish Ministry of Science, Innovation and Universities. The aim of DISNET was to improve disease understanding by analysing how diseases work and interrelate between each other, with the primary goal of acting as a tool for DR.

To achieve this objective, DISNET developed an accessible knowledge base that integrates biomedical data from both structured and unstructured sources. This information is organized into three data layers: the phenotypic layer, the biologic layer and the pharmacologic layer. The phenotypic layer includes information about diseases and their associated symptoms. The biologic layer contains molecular level data related to diseases, including genes, proteins, metabolic pathways, genetic variants, RNAs, and more. The pharmacologic layer (also referred to as drug layer) contains information about drugs, their interactions and their connections to diseases.

The data that make up the phenotype layer are obtained through DISNET’s core functionality: a biomedical text extraction process from online sources including Wikipedia, Mayo Clinic, and PubMed [7]. The extracted texts are processed using MetaMap, which identifies relevant medical terms and maps them to concepts in the Unified Medical Language System (UMLS). Therefore, this extraction process allows DISNET to collect and integrate information from public heterogeneous sources, enriching the overall biomedical knowledge base. In addition, this extraction process was originally intended to be conducted twice per month, which allows to track the evolution of available phenotypic information over time.

To allow users to retrieve both the extracted phenotype data and the information compiled in the biologic and pharmacologic layers, DISNET provides a publicly accessible API. Detailed documentation on how to use this API is available on the official DISNET website, which also features information about the project and its team, data visualization tools for exploring the network, a list of related publications, and user registration and authentication features that allow registered users to save their API queries.

To support this knowledge base, execute the data extraction and processing pipeline, and provide both an API and a web interface, DISNET is built on a microservices architecture. This strategy breaks down the system into independent components, each responsible of a specific functionality. The architecture includes services for text extraction, MetaMap processing, term validation, semantic type validation, authorization, a web interface, the main API, database storage, and the central orchestrator.

The DISNET project has made it possible to achieve meaningful results. One of the core results is the development of the DISNET platform itself, a system capable of automatically extract disease-symptom associations from different data sources using Natural Language Processing (NLP) techniques [7]. The project has also leveraged the use of network analysis to evaluate biomedical Named Entity Recognition (NER) tools [8], as well as the use of disease networks to better understand diseases [9]. In the field of disease understanding, several configurations for nosologically grouping diseases have been proposed [10], [11], and the role of Wikipedia as a biomedical knowledge source and its evolution over time has been investigated [12], [13], [14]. In the field of DR, research has been carried out along several lines. One major focus has been the development of a data driven methodology to evaluate the potential of DR hypotheses [15]. Efforts have also targeted specific diseases, including COVID-19 [16], schizophrenia [17], and rare diseases [18], [19], [20]. Another key area of study

has involved research on the application of deep learning models on graphs to identify new DR opportunities [21], [22], with enhancements such as incorporating molecular drug structures [23], or using drug and protein embeddings as features [24]. Additional contributions include work in the areas of personalized DR and pathway-based DR [25], [26], and the investigation of gender differences in the DR field [27]. Furthermore, in the context of biomedical knowledge representation, the problem of mapping disease vocabularies has been addressed [28], [29], and an ontological model describing associations between biomedical concepts and their evidences has been developed [30].

Despite its contributions, the DISNET system faces certain limitations.

In relation to its functionality, none of the text extraction processes are currently operational. The Wikipedia and Mayo Clinic pipelines were being executed twice a month until January 2023, accumulating 120 and 107 extractions respectively, before they stopped functioning as intended. The PubMed extraction process was executed only once due to the huge volume of data on its platform. Although additional runs were initially planned, they could not be carried out.

A new version of the web interface, that included graphs showing analysis of the DISNET database, was developed in 2022. However, it was never deployed, and the system still relies on the outdated 2018 version. Moreover, in both versions, the login and signup functionalities are not currently operational.

In relation to its management, the system presents high complexity in both configuration and maintenance. It follows a microservices architecture, composed of up to 16 containers: 12 dedicated to services, 3 to databases, and 1 to the MetaMap tool. While Docker Compose simplifies container orchestration, each service has its own code repository, deployment process, and monitoring setup. Additionally, the need to configure properties across a large number of services increases the risk of inconsistency, while maintenance and upgrades make dependency updates both tedious and error-prone. While a microservices architecture is typically designed to support parallel development by multiple teams, in this case, only one or two developers work on the codebase simultaneously. As a result, the overhead introduced by this architecture outweighs its benefits.

Due to the just mentioned maintenance complexity, particularly for a system managed by a single developer, outdated dependencies have accumulated over time. The system currently relies on Java 8 (while the latest is Java 23) and Spring Boot versions 1.5 to 2.5 (latest is version 3.4), which can lead to security vulnerabilities, limited compatibility with modern libraries, and greater effort required to add new features.

Additionally, the DISNET system suffers from outdated documentation. The existing documentation, limited to a single PDF, outlines the system's purpose, architecture, services, endpoints, database diagrams, and configuration details. However, it fails to reflect updates and new functionalities introduced after 2021. This lack of up to date documentation complicates the onboarding process for new developers, increasing the time and effort required to understand the system.

To address these challenges, this work aims to restore DISNET's core functionalities and to reduce system complexity by simplifying its architecture and updating its technology stack. Significantly, the results of this work will

ensure the system remains a reliable resource for ongoing research in DR and disease understanding.

1.1 Objectives

The **main objective** of this Master's Thesis is to restore the functionality of the DISNET platform and improve its maintainability and configurability. To achieve this, the following **specific objectives** are proposed:

1. Conduct a state of the art review of the actual systems dedicated to the extraction of disease information for the construction of biomedical knowledge networks.
2. Analyse the current state of the DISNET system to identify existing issues and areas for improvement.
3. Define and plan the tasks needed to repair and improve the DISNET system in collaboration with the CTB team responsible for DISNET.
4. Develop and implement the defined tasks, which include:
    a. Restoration of the Medical Text Extraction Process (MTEP), which includes the Wikipedia and Mayo Clinic text extraction pipelines.
    b. Restoration of the DISNET web page and implementation of new defined changes.
    c. Consolidation of the three databases of the system into a single unified database.
    d. Simplification of the system architecture to reduce complexity and improve maintainability.

1.2 Timeline of the project

Figure 1 presents a Gantt chart detailing the main project activities carried out for this Master's thesis, scheduled from February to June.

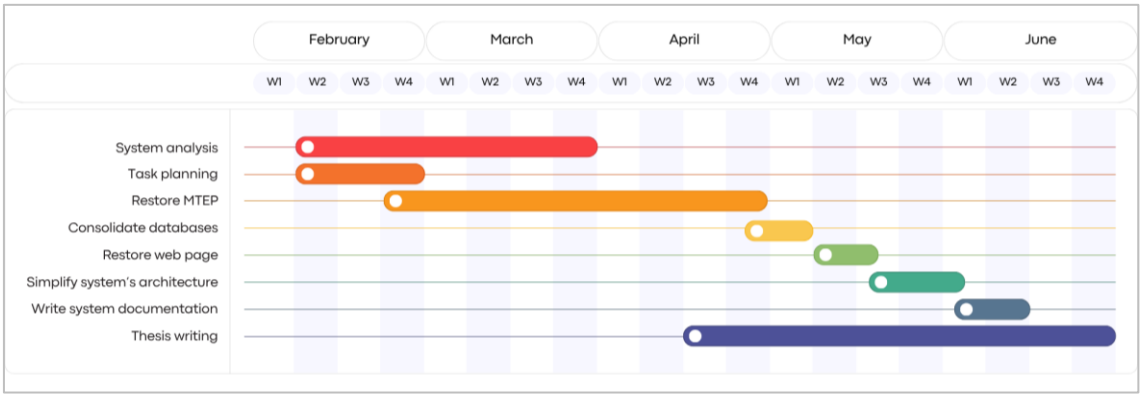


Figure 1: Gantt chart of the timeline of the project.

1.3 Outline

This master's thesis is organized into five chapters. ***Chapter 1*** has introduced the context of the work in which it is carried out, the problem statement, the thesis objectives and the project timeline. The content of the remaining chapters is summarized below:

- ***Chapter 2*** examines the state of the art, focusing on platforms similar to DISNET which are designed for integrating data to support DR.
- ***Chapter 3*** analyses the DISNET system, particularly its purpose, architecture and workflow of the main pipeline. It also elaborates on the main problems that this project seeks to resolve.
- ***Chapter 4*** outlines the required tasks needed to resolve the identified problems, their planning, and the steps made for their development.
- ***Chapter 5*** presents and discusses the results of this work.
- ***Chapter 6*** presents the overall conclusions drawn from the previous chapter and deeps on the future lines for extending the current work.

2 State of the art

The primary objective of this chapter is to present the current solutions available for the development of resources that help in the construction of disease networks aimed at facilitating DR. The discussion is framed within the context in which these platforms have been developed, focusing on their functionalities, advantages, and limitations. This aims to provide an overview of the current landscape in this field.

2.1 Platforms for data integration in drug repurposing

The following subsections describe platforms that are closely related to the context of DISNET. These platforms focus on integrating high volumes of data to support the generation of new DR hypotheses.

2.1.1 NeDRex

NeDRex (*Network-based Drug Repurposing and exploration*) [5] is a software platform designed to help researchers identify disease modules and repurposable drug candidates using biomedical networks [31]. It comprises three main components: NeDRexDB, NeDRexAPI and NeDRexApp. NeDRexDB is a knowledgebase that integrates data from more than 10 biomedical databases (OMIM, DisGeNET, DrugBank, UniProt, etc.). NeDRexAPI is a RESTful API that allows users to access, query, and build custom networks from the database. NeDRexApp is a Cytoscape[6] plugin for interactive analysis and visualization.

NeDRex builds heterogeneous biomedical networks and allows users to analyse networks using different network algorithms: MuST, DIAMOnD, BICoN, Closeness, and TrustRank to identify disease modules (clusters of genes and proteins related to a disease) and rank drugs that could affect those modules and be repurposed. All algorithms require a list of genes selected by the user as the starting point, except for BiCoN, which uses gene expression data.

NeDRex includes statistical validation tools that calculate empirical P-values to evaluate the predicted drugs. There are three validation types: drug list validation, disease module validation and joint validation of both.

One of the main limitations of NeDRex lies in its dependence on domain expertise. The selection of appropriate input genes and the interpretation of the results require substantial biological knowledge.

2.1.2 DisGeNET

DisGeNET[7] is a data integration platform focused on gene-disease and variant-disease associations. It compiles information from both curated databases and literature mining. The current version includes nearly 2 million gene-disease associations (GDAs) and more than 4 million variant-disease associations (VDAs), offering coverage across over 40,000 diseases and phenotypes.

DisGeNET offers several ways to access its data. It provides a web interface for exploring the data and a programmatic access via downloadable R, Python, Perl and Bash scripts for automated workflows. It also provides the DisGeNET Cytoscape App[8] that supports network-based visualization [32]. Additionally,

[5] https://nedrex.net/
[6] https://cytoscape.org/
[7] https://disgenet.com/
[8] https://apps.cytoscape.org/apps/disgenetapp

the DisGeNET-RDF allows users to perform complex queries along the databases [33].

However, DisGeNET is primarily a knowledge base, not a network analysis tool. It does not support the construction or visualization of disease networks within its platform. Researchers seeking to apply network-based methods must export DisGeNET data and process it using external tools.

2.1.3 CoVex

CoVex[9] is a disease specific platform developed in response to the COVID-19 pandemic. It is designed to explore the SARS-CoV-2 host interactome and to identify potential therapeutic targets through the integration of virus-host protein interactions, human protein–protein interactions, and drug–target data [34].

The platform provides an interactive web interface [10] and applies systems medicine algorithms for drug candidate prediction. CoVex allows users to visually explore the viral interactome and investigate how drugs may affect host pathways involved in SARS-CoV-2 infection.

However, CoVex is limited in scope, as it is only focused on COVID-19 research. It does not support different DR applications or the analysis of other diseases.

2.1.4 Hetionet

Hetionet[11] integrates information from 29 biomedical databases to form a large, heterogeneous network of biomedical entities [35]. The network includes over 47,000 nodes representing 11 types of biological and clinical entities (such as genes, drugs, diseases, and pathways), connected by more than 2.2 million relationships.

Hetionet is deployed using a Neo4j graph database, which supports real-time exploration using the Cypher query language. This makes it possible to trace relationships between entities and to investigate possible drug-disease interactions based on known biological pathways.

However, the platform is largely static, providing access only to precomputed results for a fixed set of diseases. Analytical tools are not integrated into the user interface and must be accessed separately. As a result, Hetionet serves more as a reference database than an interactive tool for generating drug repurposing hypotheses. Additionally, the Neo4j browser [12] is currently unavailable (as of June 2025).

2.1.5 Open Targets Platform

Open Targets Platform[13] is an open-source system designed to facilitate the identification and prioritization of drug targets by integrating data from multiple sources [36]. It combines genetic, genomic, transcriptomic, clinical, and chemical evidence to establish associations between diseases and potential therapeutic targets.

The Open Targets Platform includes information on targets (genes/proteins), diseases, and drugs, with evidence drawn from sources such as GWAS Catalog, UniProt, ChEMBL, ClinVar, Reactome, and Expression Atlas. Users can explore the data through a web interface, a GraphQL API, or bulk downloads.

[9] https://exbio.wzw.tum.de/covex/
[10] https://exbio.wzw.tum.de/covex/explorer
[11] https://het.io/
[12] https://neo4j.het.io/browser/
[13] https://platform.opentargets.org/

A core feature of Open Targets is its target-disease evidence scoring system, which helps prioritize targets based on the strength and reliability of supporting data. The platform also provides tools to analyse disease pathways, gene expression, and drug mechanisms of action, making it especially useful for initial phases of drug discovery and repurposing studies.

Although Open Targets is primarily designed to identify and validate drug targets rather than to construct full disease networks, it provides reliable associations and integrates molecular and clinical evidence in a structured and transparent way.

2.2 Discussion of the state of the art

The reviewed platforms demonstrate a wide range of strategies for integrating biomedical data to support DR. Each platform has specific strengths and limitations shaped by its focus and data sources. In this context, DISNET was created to offer a distinct perspective. DISNET focuses on phenotypic information as the basis for constructing disease networks, something that sets it apart from the other platforms.

DISNET achieves this through an architecture that organizes biomedical knowledge into three layers: phenotypic, biologic and pharmacologic. While the biological and pharmacological layers rely on structured data from expert curated databases (DisGeNET, WikiPathways, IntAct, PharmGKB), the phenotypical layer is built through automated text extraction from unstructured sources such as Wikipedia, PubMed, and Mayo Clinic. This layered integration allows DISNET to connect diseases based on phenotypic similarity.

As a result, DISNET enables the construction of disease networks based on shared symptoms by continuously extracting and updating phenotypic information. Additionally, the system stores historical phenotypic data, allowing researchers to observe the evolution of medical concepts over time.

DISNET provides access to its knowledge base through a RESTful API, promoting open access and enabling its integration into external research workflows. This ensures that all phenotypic, biological, and pharmacological information processed by the system is available for use by the scientific community.

In summary, DISNET offers the unique approach to combine unstructured phenotypic data with curated biological and pharmacological information to build disease networks that support DR. Its openly accessible framework, dynamic data integration, and focus on symptoms relationships make it a valuable tool for biomedical research. Therefore, the restoration and maintenance of DISNET are of significant importance to the advancement in the research of the field.

3 Analysis of the DISNET System

This chapter presents the analysis of the DISNET platform. The purpose of this analysis is to understand the functionality, architecture and workflow of the system. Additionally, it identifies and evaluates existing problems within the system that will be addressed in the following chapter.

3.1 System Overview

3.1.1 Purpose

The DISNET system final purpose is to serve as a disease understanding and DR tool by building a complex multilayer network based on the concepts of HDNs. Its core objectives are:

- Integrate biomedical heterogeneous knowledge from public sources.
- Contribute to disease understanding, with special emphasis on rare diseases.
- Provide a public tool for discovering DR hypotheses, offering access to free available data.

In addition to these core objectives, the deployment of DISNET has enabled several extended purposes:

- Track the evolution of extracted disease terms through time, offering an opportunity to analyse the progression of human knowledge about diseases.
- Support the *Data Driven Disease Visualization and Drug Repurposing* (DRIVE)[14] platform, which uses DISNET's data to generate and visualize drug repurposing hypotheses. Additionally, the web interface originally developed for DISNET served as a precursor to the DRIVE platform.
- Collaborate in research projects by the MEDAL laboratory, especially those focused on network medicine and DR.

3.1.2 Main Functionalities

To achieve its purpose, the DISNET system provides the following core functionalities:

- Extract unstructured disease information from diverse public web sources, specifically Wikipedia, Mayo Clinic and PubMed.
- Process the extracted information using text mining and NLP techniques. This step uses MetaMap to identify and extract medical concepts.
- Store the validated information in a structured database, enabling the construction of a complex multilayer network.
- Offer a publicly accessible API that enables the free and structured sharing of the knowledge generated by the system.
- Provide a publicly available website that includes information about the DISNET project and its team, the API's documentation, data visualization tools for exploring the knowledge database, a list of related publications, and user registration and authentication features that allow registered users to save their API queries.

[14] *https://live.drive-project.com/*

3.1.3 Users

DISNET users can be classified into two main categories: end users and developers.

End users consist of the scientific community, including clinicians, pharmaceutical professionals, bioinformaticians, data scientists, and especially biomedical researchers. These users interact with DISNET via its public API and through the available visualization tools. They use the platform to investigate diseases and the drugs, genes, pathways, proteins and symptoms they have in common, analyse the interconnections between diseases and their symptoms, and explore potential DR hypotheses. Through these activities, users are able to support their broader objective of generating new insights and conducting research in the field of disease understanding, ultimately contributing with meaningful findings to the scientific community. The MEDAL research team is also part of this user group, continuously using the data from the DISNET knowledge base for other research projects in the fields of network medicine and drug repurposing.

Developers, a group that consists of members of the MEDAL team, are responsible for the technical development and maintenance of the DISNET system. Their responsibilities include implementing new features and modules to support emerging research projects. In addition, they handle system maintenance activities, which involve using the text extraction platform to keep the biomedical information accurate and up to date, and ensuring the DISNET web interface and DISNET's API remains accessible and operational for the end users.

3.1.4 Development History

This system began its life as the result of the work from the doctoral thesis titled "*Characterization of Diseases Based on Phenotypic Information Retrieved through Biomedical Knowledge Extraction from Public Sources*" (translated from Spanish), authored by Gerardo Lagunes García [37]. His work led to the development of the core of DISNET, which included the functionalities for extracting and generating biomedical knowledge from various public sources (Wikipedia, Mayo Clinic, and PubMed), as well as the validation process using NLP techniques and MetaMap. He also implemented the web API to provide access (restricted to registered users) to the extracted data. As a result, he was the main developer of the DISNET system.

Following this, DISNET's services were extended by integrating both the graphical interface and the DisMaNET API as part of the DISNET platform. This work was carried out as part of the doctoral thesis of Eduardo García del Valle [38].

The biological layer was initiated with the bachelor's thesis of Lucía Prieto Santamaría, who developed the automated extraction process for biological data and the creation of the corresponding database structure [39]. Additional contributions to this layer were made by other researchers:

- Pablo Soto García expanded the biological layer with features such as ncRNA, miRNA, lncRNA, and circRNA [40].
- Natalia García Sánchez introduced relevant biological sequences into the layer to support drug repositioning efforts [41].
- Addition of new data on proteins, including biological sequences and alignments [42].

The pharmacological layer was initially developed by Sara Jaramillo Cárdenas as part of her master's thesis, focusing on the extraction and structuring of pharmacological data [43]. Later, Esther Ugarte Carro updated and refined the drug schema to align it more efficiently with the biological layer [44].

Several other individuals made important contributions:

- Adrián Ayuso Muñoz generated Graph Neural Network (GNN) models for drug repurposing predictions within the DISNET graph and development of the DR visualization for the web interface [45], [46].
- Alejandro Alonso Noguerales implemented a Java API wrapper to allow easier interaction with the DISNET REST API through Java code [47].
- Daniel Navarro Riaño implemented a Python API wrapper to allow easier interaction with the DISNET REST API through Python code [48].
- Marina Díaz Uzquiano collaborated in the improvement of the mappings across layers, the development of data visualizations and web interfaces, and the monitorization of extraction processes.
- David Fernández Lobón implemented the histogram visualizations available in the web interface under the "About > Database" section.
- Belén Otero Carrasco worked on rare disease data, further enriching the system [49], [50].
- Andrea Álvarez Pérez developed an ontology that semantically models associations between the biomedical concepts of DISNET [51].

3.2 Technology Stack

DISNET is an application based on microservices and composed of the following technologies that support its architecture and deployment:

- The system is deployed in production on a remote server running **Ubuntu** as the operating system.
- The back-end is implemented in **Java**, using the **Spring Boot** framework for building the application.
- **Docker** is used to containerize the microservices and **Docker Compose** handles their orchestration.
- **MySQL** serves as the relational database management system.
- The web interface is built using **Materialize CSS** for styling and **Thymeleaf** as the templating engine.

Table 1 compares the versions of the technologies currently used in the system and the latest available versions at the time of writing. The comparison reveals that most of the technologies are not up to date and could benefit from version upgrades.

*Table 1: Technologies used in the DISNET system and their current available versions (*as of June 2025).*

Technology	Version in App	Latest Available Version*
Ubuntu LTS	20.04 LTS	24.04 LTS

Java	8	24
Spring Boot	1.5.4.RELEASE – 2.5.2	3.5
Docker	28.0.0	28.2.2
Docker Compose	2.32.4	2.36.1
MySQL	8.4 LTS	8.4 LTS
MySQL Connector Java	5.1.44	9.3.0
Materialize CSS	0.100.2	1.0.0.1
Thymeleaf	2.1.5.RELEASE	3.1.3.RELEASE

The following subsections describe each of these technologies in more detail:

3.2.1 Java

Java[15] is a programming language and computing platform originally released by Sun Microsystems in 1995, and now owned by Oracle. It is known for its "write once, run anywhere" philosophy, as Java code is compiled into bytecode that can run on any system equipped with a Java Virtual Machine (JVM), making it highly portable across platforms and devices. Java offers several features, including object-oriented programming, which allows for modular and reusable code; robustness, thanks to its extensive compile-time checking; and multithreading capabilities which offer high performance [52]. Its relative simplicity and extensive ecosystem have made Java a popular choice for developing applications across a wide range of domains.

3.2.2 Spring Boot

Spring Boot[16] is a framework built on top of the Spring Framework[17], developed to simplify the process of configuring and deploying Spring-based[18] applications. It provides a set of tools that reduce the need for extensive manual configuration and allows developers to create applications faster and easier. An important aspect of its simplicity is the use of *application.properties* files, which allow developers to externalize configuration settings.

3.2.3 Docker and Docker Compose

Docker[19] is a containerization technology used to create and manage containers. Containers are standalone units that package an application together with everything needed to run it. This approach ensures that applications run consistently across different environments.

Docker Compose[20] is a complementary tool that allows to manage multi-container applications where all containers run on the same Docker host. It allows developers to define and configure all application services in a single YAML file (*docker-compose.yml).* Once defined, the entire application can be deployed and started with a single command, simplifying the workflows.

[15] *https://www.java.com/*
[16] *https://spring.io/projects/spring-boot*
[17] *https://spring.io/projects/spring-framework*
[18] *https://spring.io/*
[19] *https://www.docker.com/*
[20] *https://docs.docker.com/compose/*

3.2.4 MySQL

MySQL[21] is an open source Relational Database Management System (RDBMS) that uses SQL to create and manage databases. MySQL stores data in tables of rows and columns organized into schemas.

3.2.5 Materialize CSS and Thymeleaf

Materialize[22] is a modern front-end framework based on Google's Material Design principles. It provides a collection of ready-to-use UI components (such as forms, buttons, navigation bars, and grids) that help ensure usability and uniformity across the application.

Thymeleaf[23] is a server-side Java template engine used for rendering dynamic web pages in web application. It allows developers to insert dynamic content into static templates.

3.3 System Architecture

The application is composed of a total of 16 Docker containers: 3 containers host database instances, 12 containers implement the different application microservices and the last container runs the MetaMap Server, necessary to identify medical concepts from the extracted texts. A diagram of the system's architecture can be found in **Figure 2**.

3.3.1 Database Containers

The system uses three distinct databases, each serving a specific role in the application's data architecture. These are the *Disease List Available Database*, the *Diseases Database* and the *Users Database*.

The ***Disease List Available Database*** serves as the backup to store the query results for later extracting the text without having to repeatedly query external sources. It contains a single schema called *addb*, whose table diagram is shown in **Figure A1**, and which comprises 12 tables. This schema stores the results of the SPARQL queries against DBpedia which includes the disease names, associated URLs, and classification codes. It also includes *safe disease* tables that act like a reference list of all diseases and their associated URLs that have ever been extracted. This way, if a future Wikipedia extraction fails to include a disease that was present before, the system does not lose that information.

The ***Diseases Database*** is a multi-schema database designed to represent various layers of biomedical knowledge. The schemas are the following:

- **edsssdb:** represents the phenotypic layer. It stores all the extracted information about diseases from the public sources of Wikipedia, Mayo Clinic and PubMed, including their symptoms, their codes, texts, semantic types, documents, and URLs. The table diagram for this schema is presented in **Figure 3**.
- **disnet_biolayer:** represents the biologic layer and includes 42 tables that represent biological entities and their relationships. It includes entities such as diseases, genes, proteins, genetic variants, biological pathways, non-coding RNAs, biomarkers, protein organisms and prevalence data of diseases. Each entity is linked to standardized identifiers like UMLs, Concept Unique Identifiers (CUIs), NCBI, GeneIDs, and UniProt accessions. Relationships between these entities, such as Disease–Gene, Gene–Protein, Protein–Protein, Disease–Variant, are also stored. The data

[21] https://www.mysql.com/
[22] https://materializecss.com/
[23] https://www.thymeleaf.org/

was obtained mainly from DisGeNET, but also from the UMLS Metathesaurus, Orphanet, NCBI Nucleotide, Panther, NCBI Protein, WikiPathways. The data from this schema was extracted once and is not being updated. The table diagram for this schema is shown in **Figure A2**.

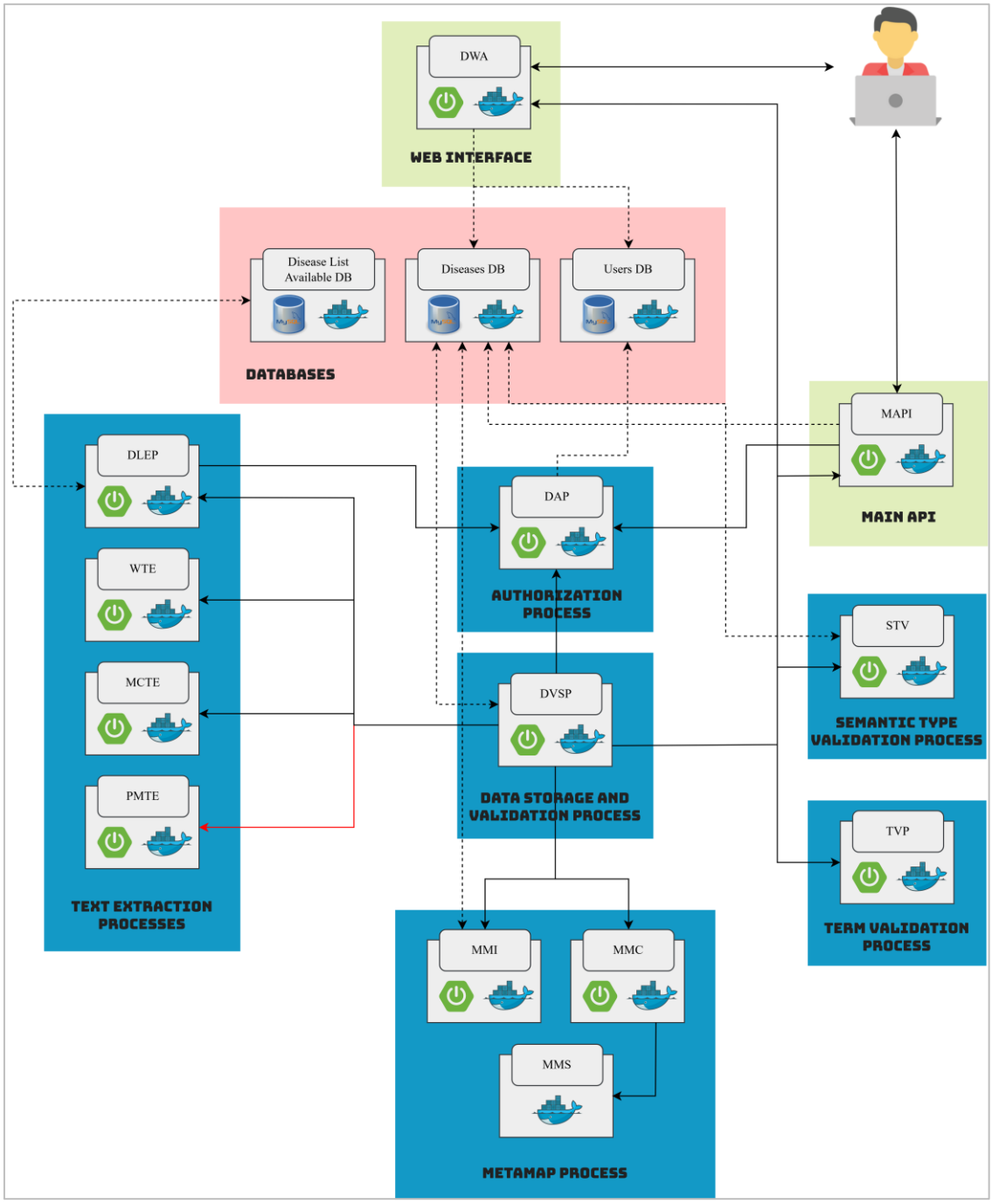


Figure 2: DISNET system architecture.

- **disnet_drugslayer:** it represents the pharmacologic layer and stores drug-related biomedical knowledge across 30 tables. It includes the entities of disease, drug, target, phenotypical effect and organism. Relationships modelled in the schema are disease-drug associations, drug-drug interactions, drug-target bindings, and drug-phenotype links. Data is obtained from the following sources: CTD, ChEMBL, RxNorm,

RxClass, SIDER, DrugBank and TWOSIDES. The table diagram for this schema is shown in **Figure A3**.

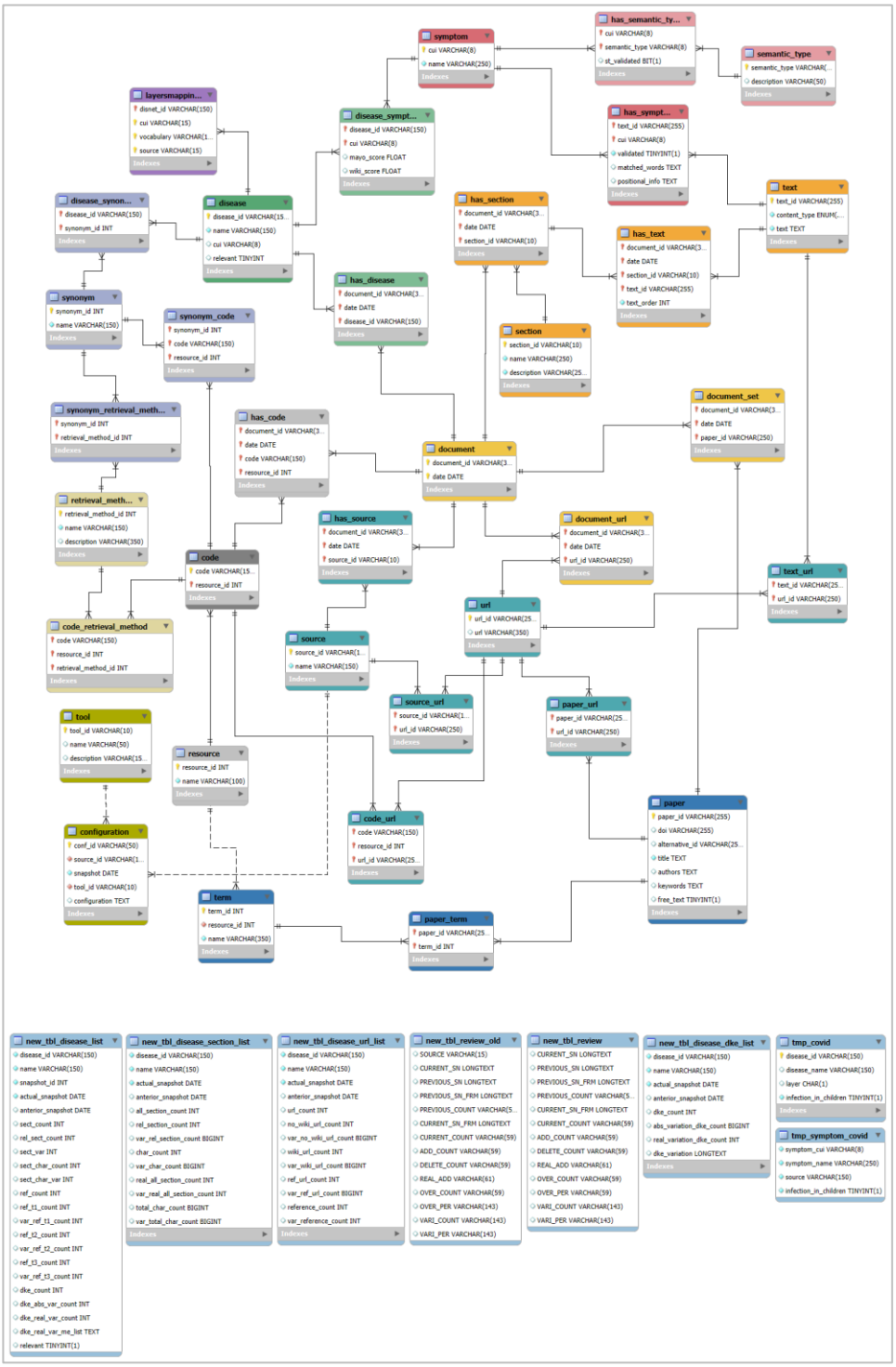


Figure 3: Table diagram of the edsssdb schema used in the Diseases Database.

- **umls:** the data from this schema is obtained from the UMLS Metathesaurus data files[24]. There are three tables MRCONSO, MRREL and MRSTY. MRCONSO contains the biomedical terms with their names, CUIs, source codes, code and synonyms, and more. MRREL stores

[24] https://www.nlm.nih.gov/research/umls/new_users/online_learning/Meta_006.html

relationships between concepts. MRSTY assigns semantic types to each concept. The table diagram for this schema is shown in **Figure A4**.

- **dr:** it means drug repurposing. The information stored in this schema was obtained from the three data layers of DISNET. This schema is used by the DRIVE application. It enables the exploration and visualization of repurposing hypotheses by using different data-driven methodologies. The table diagram for this schema is shown in **Figure A5**.
- **repu_temp:** the table diagram for this schema is shown in **Figure A6.**

The **Users Database** contains a single schema called *disnetdb***.** It stores all the information related to the registered users: the user data (name, password, academic info, country, etc.), the access token and the queries the user has executed. The table diagram for the *disnetdb* schema is presented in **Figure A7**.

3.3.2 Application Service Containers

The 12 different services of the application are explained below. **Figure 4** shows how these services are connected with the different database schemas.

- **Wikipedia Text Extraction (WTE):** it uses the URLs extracted using the Disease List Extraction Process (DLEP) service to extract the texts from the Wikipedia articles. It generates a JSON that stores all the diseases with their corresponding texts.
- **Mayo Clinic Text Extraction (MCTE):** it extracts texts about diseases from the Mayo Clinic webpage. It generates a JSON that stores the diseases with their corresponding diseases, texts and synonyms.
- **PubMed Text Extraction (PMTE):** it extracts texts about diseases from PubMed. It generates a JSON that stores the diseases with their corresponding texts and synonyms. Currently, this service is not integrated into the system's workflow and is not operational.
- **DISNET Authorization Process (DAP):** it handles the authorization of tokens submitted by users to the Main API (MAPI), granting or denying access based on token validity. Additionally, it authorizes specific tokens used by microservices within the Medical Extraction Process (MTEP) to securely communicate with other microservices in the application.
- **Disease List Extraction Process (DLEP):** the main purpose of this service is to perform SPARQL queries against DBpedia to obtain the diseases available in Wikipedia with their URLs and codes. It stores this information into the schema *addb* from the Disease List Available Database, to later be used by the WTE service.
- **MetaMap Client (MMC):** this service receives the list of texts extracted with the Wikipedia and Mayo Clinic services and consumes the MetaMap API (in the container disnet-metamap2016v2) to obtain the medical concepts that can be found in these texts. It generates a JSON that includes the used MetaMap configuration and the list of analysed texts along their concepts which describe their UMLS CUI, name of the detected concepts, their semantic types, the words found and their position inside the text.
- **MetaMap Inserts (MMI):** this service consumes the JSON generated from the MMC to store this information inside the *edsssdb* schema from the Diseases Database.

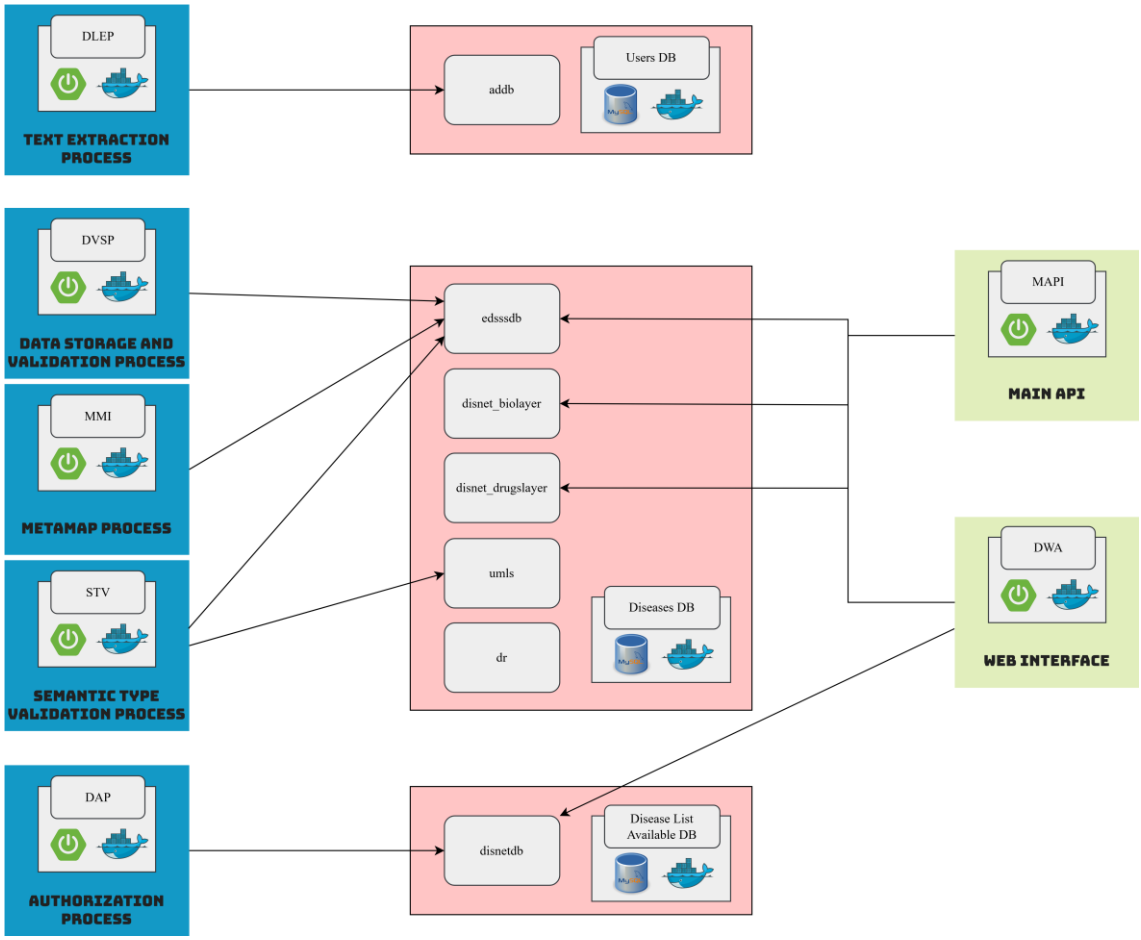


Figure 4: Diagram showing the microservices of DISNET and the specific database schemas they interact with.

- **Term Validation Process (TVP):** this service receives a list of terms and filters out only those that are valid medical terms or phenotypic traits. TVP applies three distinct methods to validate CUIs: CUI identification, equality validation, and similarity validation.
- **Semantic Type Validation (STV):** also filters terms based on their semantic types.
- **Main API (MAPI):** the main API is a RESTful interface that users interact with to access DISNET data. It provides access to all medical information available in both the phenotypic, biologic, and pharmacologic layers. The documentation for this API is available on DISNET's website[25].
- **DISNET Web Application (DWA):** this service deploys the official DISNET website, where users can register and obtain a token to access the MAPI. While the token is no longer required to use the API, logging in with it enables users to maintain a personal history of their API requests within the web interface.
- **Data Storage and Validation Procedure (DSVP):** located at the centre of the diagram, the DSVP service acts as the orchestrator of the MTEP workflow. It runs as a scheduled task twice a month, during which it performs the following steps: it extracts relevant texts communicating with the DSVP, WTE, MCTE, and PMTE services, sends them to the MMC

[25] https://disnet.ctb.upm.es/api/doc

for processing, and then validates the results using the TVP and STV services.

3.3.3 MetaMap Server (MMS)

The final container is called **disnet-MetaMap2016v2-container** which contains the local installation of MetaMap version MetaMap2016v2. This service allows to access the MetaMap program to map the biomedical extracted text to concepts in the UMLS Metathesaurus.

To run the MetaMap Server, the SKR-MedPost Part-of-Speech (POS) Tagger Server and the Word Sense Disambiguation (WSD) Server must be started beforehand.

3.4 Medical Term Extraction Process

The Medical Term Extraction Process (MTEP) refers to the complete execution of the DISNET system. It encompasses the extraction of disease information from public sources (Wikipedia, Mayo Clinic and PubMed) and includes all steps involved in extracting, processing, and structuring this information into the corresponding databases. This process is illustrated in **Figure 5**. The MTEP process begins with **Phase 1: text extraction**, which involves retrieving information from two primary sources: Wikipedia and Mayo Clinic. These extraction workflows run in parallel. However, data insertion into the database is not entirely simultaneous due to the automatic generation of unique IDs. This means that the Wikipedia process may wait if Mayo Clinic is inserting data. For Wikipedia, the system first queries DBpedia to obtain URLs linked to disease-related articles. Relevant sections of these articles are then extracted and stored in the *edsssdb* database. Once stored, the texts are processed using MetaMap NLP, which identifies medical concepts such as symptoms. These extracted symptoms are also inserted into the database, and a validation step using the TVP process filters and confirms the relevance of the identified terms.

In the case of Mayo Clinic, the process is more direct. There is no need to rely on an external service like DBpedia because Mayo Clinic already provides an indexed list of diseases with direct links to their respective pages. The relevant disease content is extracted from these pages and stored in the *edsssdb*, followed by semantic processing using MetaMap, just like with the Wikipedia texts. The resulting symptoms are saved in the database and validated through the TVP process to ensure consistency and accuracy.

Following successful extraction and symptom processing, the system enters **Phase 2: cross-layer mapping and symptom validation**. This phase begins with layer mappings, which link diseases across the layers of the DISNET system (the phenotypic, biologic, and pharmacologic layers) since each layer uses different identifiers. An additional semantic type validation is performed beyond TVP to double-check the correctness of the stored symptoms. Then, a table of disease symptom associations is generated. This step is crucial because initially, symptoms are indirectly connected to diseases through intermediate tables. A database function runs multiple inner joins to consolidate this information into a direct disease symptom relationship. Finally, a JSON statistics file is generated, summarizing the extracted and processed data. These statistics can later be accessed through the DISNET web interface.

If all previous steps are completed successfully, the system proceeds to **Phase 3: backup and reporting.** In this final phase, a full backup of the databases is created and stored on a Network Attached Storage (NAS) device. A report email

is then sent out, summarizing the process results, and the application workflow is finished.

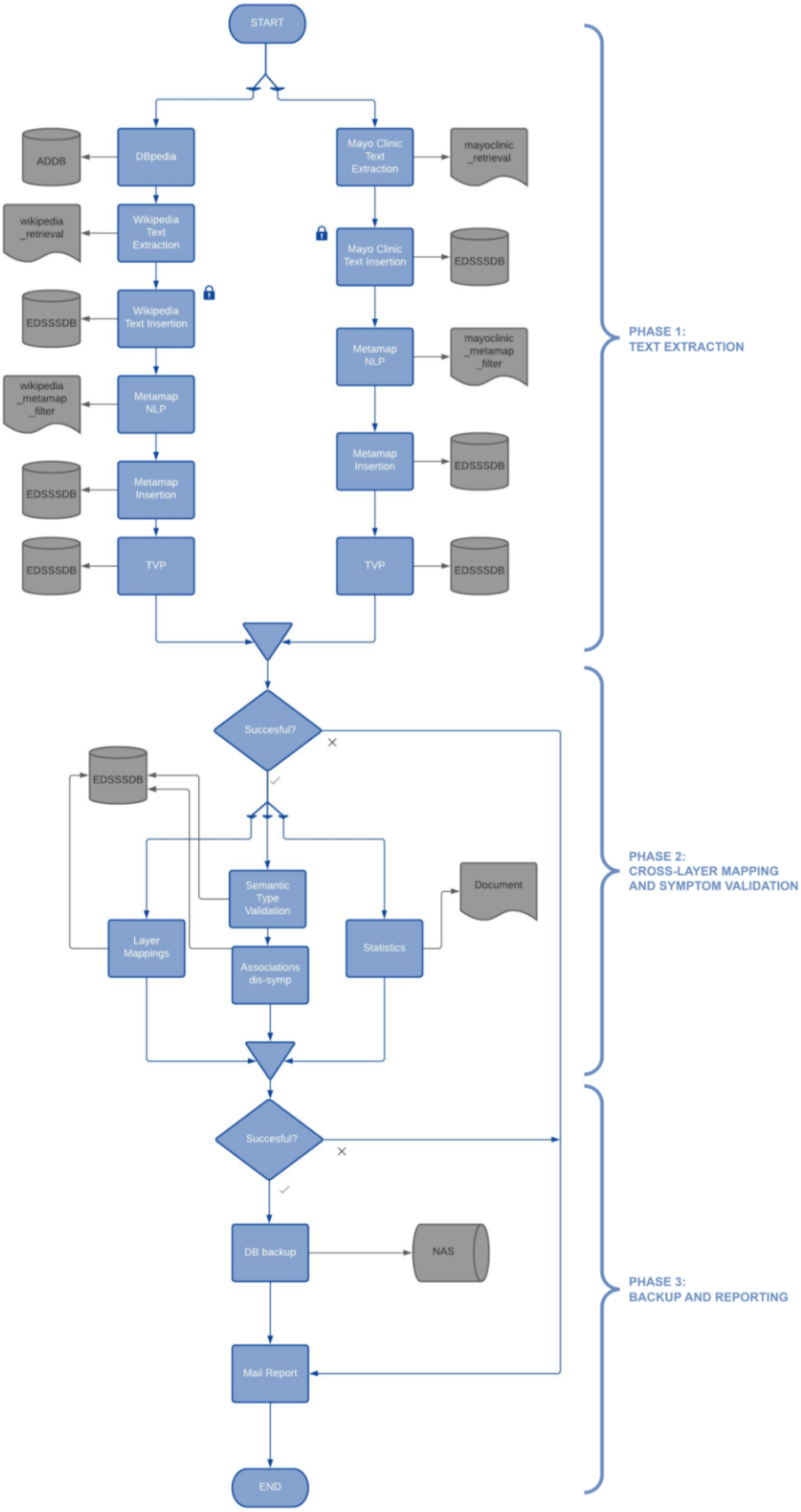


Figure 5: DISNET pipeline for executing the MTEP process.

3.4.1 Text Extraction

This section describes the text extraction processes used to retrieve disease related information from the public sources of Wikipedia and Mayo Clinic. Although PubMed is also a public source from which DISNET has previously extracted information (on a single occasion), it is not included in this analysis or in the scope of this thesis, as the current system workflow does not incorporate PubMed.

3.4.1.1 Wikipedia Text Extraction

3.4.1.1.1 Wikipedia

Wikipedia [26] is a free online encyclopaedia, written and maintained by a community of volunteers, through open collaboration and the wiki software *MediaWiki*[27]. It was founded on 2001 by Jimmy Wales and Larry Sanger, and it has been hosted by the non-profit *Wikimedia Foundation*[28].

According to its website[29], Wikipedia is the largest and most-read reference work in history. As of May 2025, there are more than 7 million articles in the English Wikipedia. The platform continues to grow steadily, with an average monthly growth of approximately 15.000 new entries.

In the medical domain, Wikipedia plays a significant role as a publicly accessible source of health-related knowledge. As of February 2019, there were approximately 223.000 medical articles available across 281 languages, with 34.516 of those articles found in the English Wikipedia[30].

3.4.1.1.2 DBpedia

DBpedia [31] is a project aimed at extracting structured content from the information created in the Wikimedia projects. It transforms Wikipedia content into a rich and interconnected dataset, following the principles of the Semantic Web and Linked Open Data (LOD). DBpedia represents this data in Resource Description Framework (RDF) format, which expresses information as triples with subject, predicate and object.

To make this data accessible, DBpedia offers a SPARQL (SPARQL Protocol and RDF Query Language) endpoint to allow users to query the DBpedia dataset using SPARQL, a specialized query language for RDF data. Through this public endpoint, users can construct and execute queries to extract precise information or discover relationships between entities.

One of the major challenges for DBpedia is keeping its dataset synchronized with Wikipedia, which is constantly updated by contributors around the world. Traditional DBpedia releases are based on monthly Wikipedia dumps, which means the extracted data may be outdated by the time it is published. To address this limitation, DBpedia Live was developed. DBpedia Live [32] continuously updates the dataset in near real-time as changes occur in Wikipedia. This feature ensures that the DBpedia data remains current and reflects the latest information added or modified in Wikipedia.

[26] https://en.wikipedia.org/
[27] https://www.mediawiki.org/
[28] https://wikimediafoundation.org/
[29] https://en.wikipedia.org/wiki/Wikipedia
[30] https://en.wikipedia.org/wiki/Wikipedia:WikiProject_Medicine/Stats
[31] https://www.dbpedia.org/
[32] https://www.dbpedia.org/resources/live/

3.4.1.1.3 Extraction Process

The extraction of disease information from Wikipedia is carried out using two services: DLEP and WTE. DLEP is used to obtain the diseases available in Wikipedia via DBpedia, with their URLs and codes. Later, the WTE service uses the URLs extracted from DLEP to extract the texts from the Wikipedia disease articles. A representation of this process is shown in **Figure 6**.

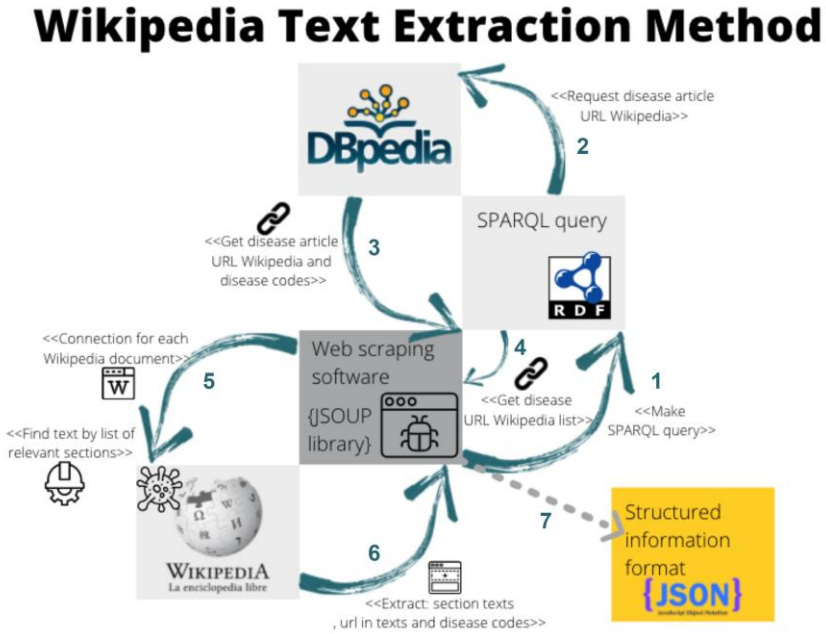


Figure 6: Workflow of the knowledge extraction process from Wikipedia.

3.4.1.1.3.1 DLEP Process

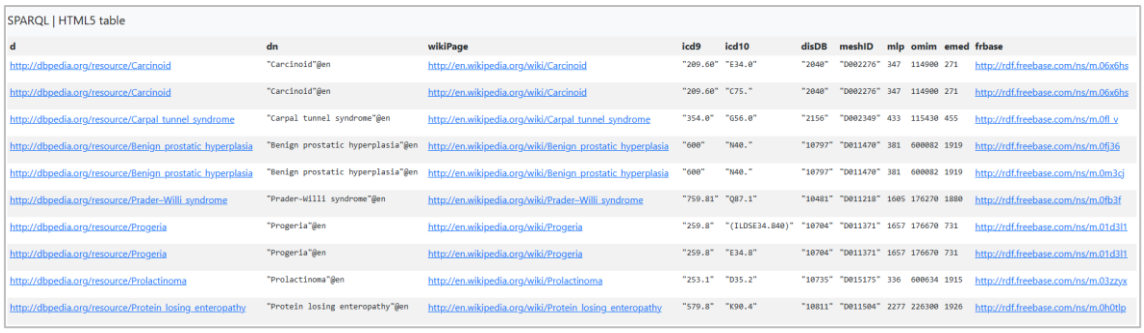
The DLEP process begins with the initialization of a disease album, which serves as a container to group all diseases extracted on a specific date.

Two SPARQL queries are then loaded from local files (one for DBpedia and the other one for DBpedia Live). This queries are executed against the official DBpedia and DBpedia Live SPARQL endpoints[33, 34] to retrieve disease-related data. An example of the query result for DBpedia is shown in **Figure 7**. Each query returns a structured set of results that include: the name of the disease, the URLs of the disease, which include the DBpedia URI, the Wikipedia page URL, and the Freebase URI, and the related medical classification codes: ICD-9, ICD-10, Diseases Database Identifier, MeSH, MedlinePlus, OMIM, and eMedicine.

Each result from the SPARQL queries is transformed into a *Disease* object containing all the relevant information. These objects are stored in a map, where the keys are *Code* objects containing a LinkedList of the classification codes, and the values are the corresponding *Disease* objects.

The data is then persisted in the *addb* schema of the *Diseases List Available Database*. A new album entry is inserted, with the number of diseases retrieved and the date of the extraction. The system iterates over the map of diseases and codes, inserting each one into the *disease* table. Associated URLs are inserted into the *url* and *disease_url* tables, while classification codes are stored in the *code* and *disease_code* tables.

[33] http://dbpedia.org/sparql
[34] http://live.dbpedia.org/live/



SPARQL | HTML5 table

d	dn	wikiPage	icd9	icd10	disDB	meshID	mlp	omim	emed	frbase
http://dbpedia.org/resource/Carcinoid	"Carcinoid"@en	http://en.wikipedia.org/wiki/Carcinoid	"209.60"	"E34.0"	"2048"	"D002276"	347	114900	271	http://rdf.freebase.com/ns/m.06x6hs
http://dbpedia.org/resource/Carcinoid	"Carcinoid"@en	http://en.wikipedia.org/wiki/Carcinoid	"209.60"	"C75."	"2048"	"D002276"	347	114900	271	http://rdf.freebase.com/ns/m.06x6hs
http://dbpedia.org/resource/Carpal_tunnel_syndrome	"Carpal tunnel syndrome"@en	http://en.wikipedia.org/wiki/Carpal_tunnel_syndrome	"354.0"	"G56.0"	"2156"	"D002349"	433	115430	455	http://rdf.freebase.com/ns/m.0fl_v
http://dbpedia.org/resource/Benign_prostatic_hyperplasia	"Benign prostatic hyperplasia"@en	http://en.wikipedia.org/wiki/Benign_prostatic_hyperplasia	"600"	"N40."	"10797"	"D011470"	381	600082	1919	http://rdf.freebase.com/ns/m.0fj36
http://dbpedia.org/resource/Benign_prostatic_hyperplasia	"Benign prostatic hyperplasia"@en	http://en.wikipedia.org/wiki/Benign_prostatic_hyperplasia	"600"	"N40."	"10797"	"D011470"	381	600082	1919	http://rdf.freebase.com/ns/m.0m3cj
http://dbpedia.org/resource/Prader–Willi_syndrome	"Prader-willi syndrome"@en	http://en.wikipedia.org/wiki/Prader–Willi_syndrome	"759.81"	"Q87.1"	"10481"	"D011218"	1605	176270	1880	http://rdf.freebase.com/ns/m.0fb3f
http://dbpedia.org/resource/Progeria	"Progeria"@en	http://en.wikipedia.org/wiki/Progeria	"259.8"	"(ILDSE34.840)"	"10704"	"D011371"	1657	176670	731	http://rdf.freebase.com/ns/m.01d3l1
http://dbpedia.org/resource/Progeria	"Progeria"@en	http://en.wikipedia.org/wiki/Progeria	"259.8"	"E34.8"	"10704"	"D011371"	1657	176670	731	http://rdf.freebase.com/ns/m.01d3l1
http://dbpedia.org/resource/Prolactinoma	"Prolactinoma"@en	http://en.wikipedia.org/wiki/Prolactinoma	"253.1"	"D35.2"	"10735"	"D015175"	336	600634	1915	http://rdf.freebase.com/ns/m.03zzyx
http://dbpedia.org/resource/Protein_losing_enteropathy	"Protein losing enteropathy"@en	http://en.wikipedia.org/wiki/Protein_losing_enteropathy	"579.8"	"K90.4"	"10811"	"D011504"	2277	226300	1926	http://rdf.freebase.com/ns/m.0h0tlp

Figure 7: Part of the result of the SPARQL query result for DBpedia.

Once the insertion is complete, a JSON file is generated and saved. This file contains a full snapshot of the album, including the list of diseases, and serves as an extraction history log.

To improve the dataset, the system merges diseases from the existing *safe_disease* table, diseases from the current album and diseases from the previous Wikipedia extracted album. This is done to insert any missing diseases into the current album. Finally, the system updates the *safe_disease* list and inserts the extracted diseases into the three dedicated tables: *safe_disease* (storing the disease), *safe_url* (storing the corresponding URL), and *safe_disease_url* (storing the relationship between disease and URL). In this way, the *safe_disease* tables serve as a repository to ensure that the current extraction contains all the extracted diseases.

3.4.1.1.3.2 WTE Process

The process begins by receiving the list of diseases with their links extracted by the previous process. Then, the configuration from the *sources.xml* file is loaded, which defines the relevant sections to be extracted from the Wikipedia articles. These sections are: Signs and symptoms, Sign, Signs, Symptom, Symptoms, Cause, Causes, Diagnosis, Diagnostic approach, Presentation, Symptoms and causes, Diagnostic, Causes of injury, Signs & symptoms, Symptoms and diagnosis, Symptoms and causes, and infobox.

Next, a list of documents (*Doc*) is created, where each object represents an individual web page. For each document, a connection is established through its URL, and if successful, the content is processed. The system also extracts bibliographic references found within the page. Additionally, two automatically generated sections are added: “Symptoms of [disease]” and “Causes of [disease]”.

The system then iterates through the predefined sections to check if the document contains any of them. If a match is found, the text within that section is extracted and classified based on its HTML tag: paragraphs (<p>), lists (<ul>, <ol>, <dl>), or tables (<table>). Each extracted item is encapsulated as a *Text* object (either a *Paragraph*, *List*, or *Table*) allowing the information to be stored in a structured format. To illustrate the result of structuring the Wikipedia information, **Figure 8** shows how information from each webpage element corresponds to specific objects. **Figure 9** presents the object-oriented structure of a single Wikipedia article.

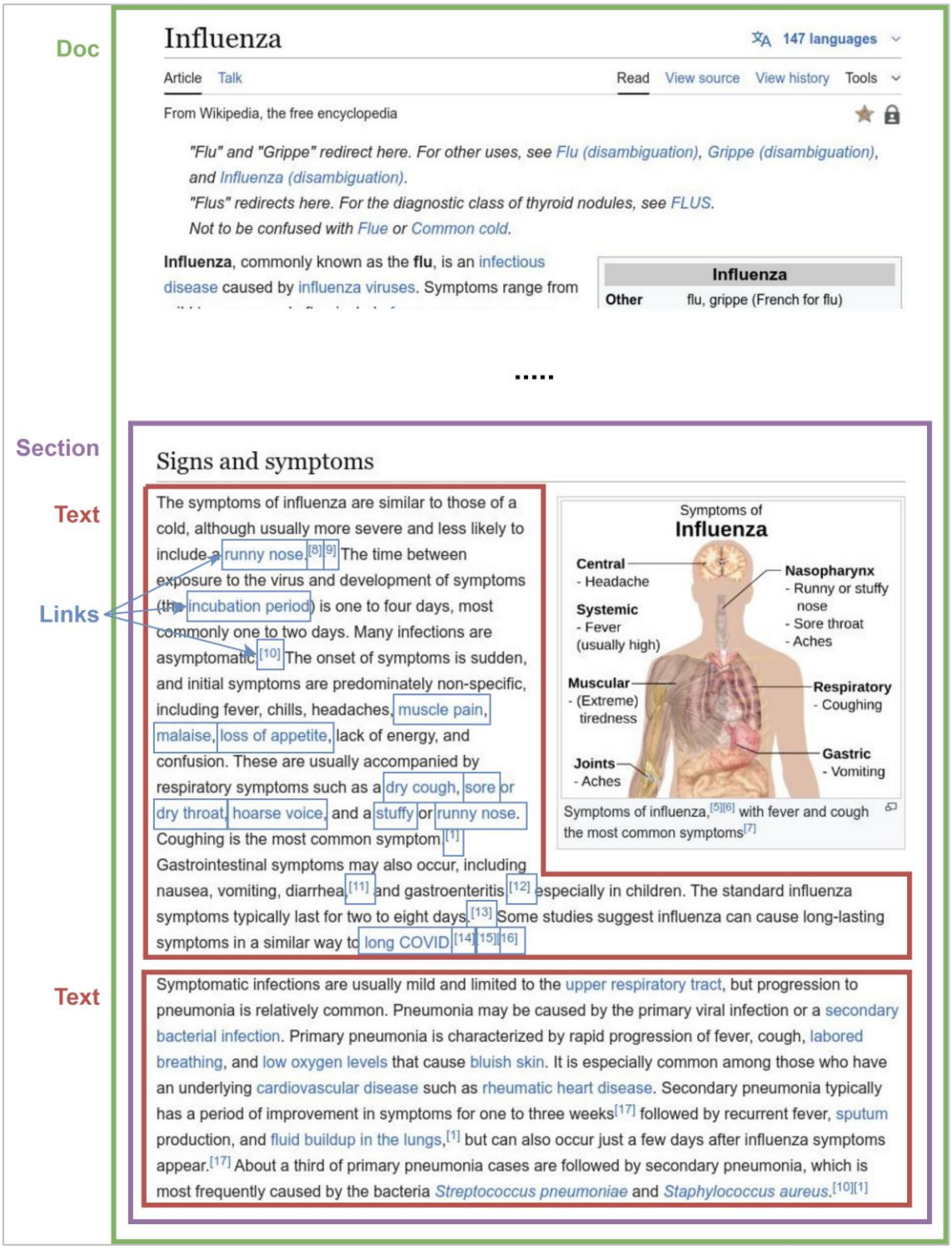


Figure 8: Screenshot of a Wikipedia article on Influenza showing the hierarchy of the components for the text extraction process.

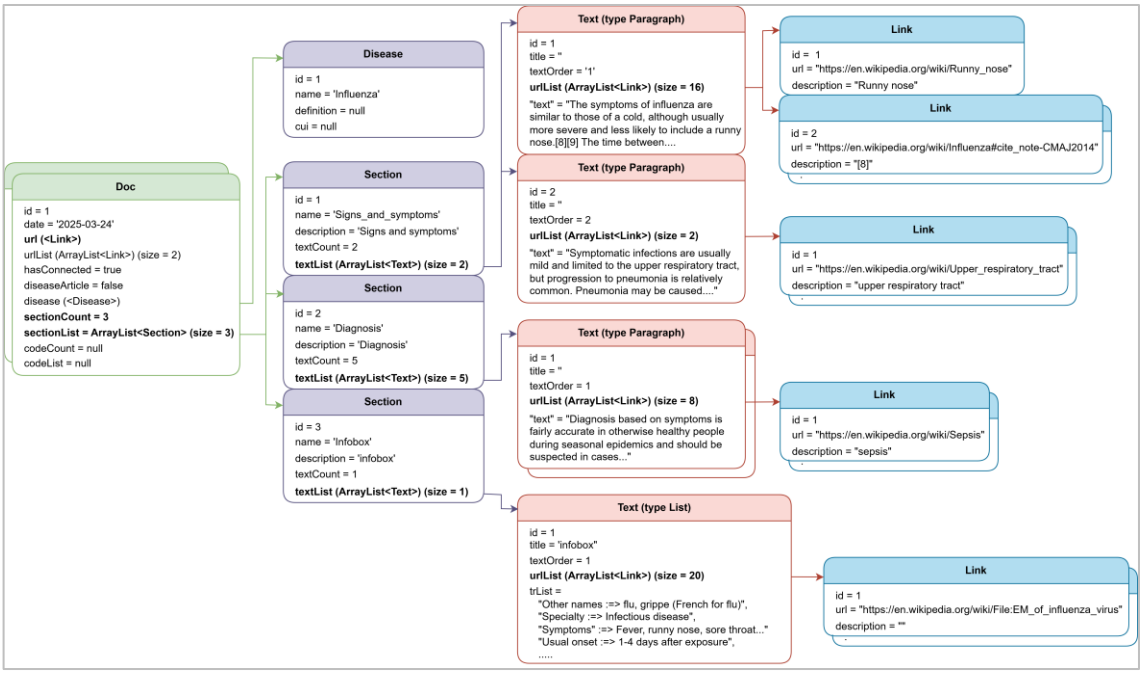


Figure 9: Diagram showing the structure information of a Wikipedia document.

3.4.1.2 Mayo Clinic Text Extraction

3.4.1.2.1 Mayo Clinic

According to its official website[35], Mayo Clinic is recognized as the largest nonprofit medical practice in the world. Its mission is to provide comprehensive, compassionate care to every patient, guided by the principle that "the needs of the patient come first". Mayo Clinic is consistently ranked among the top healthcare institutions globally. Its main campuses are located in Florida, Minnesota, Arizona, being this two last ones recognized as an Honor Roll member by *U.S. News & World Report's 2024-2025 "Best Hospitals" rankings*[36].

Mayo Clinic extends its services beyond its major campuses through two key systems: the Mayo Clinic Health System and the Mayo Clinic Platform. The Mayo Clinic Health System[37] is a network of hospitals and clinics that extends its reach to 39 communities across four regions in Minnesota and Wisconsin. The *Mayo Clinic Platform*[38] is a recent digital initiative focused on transforming healthcare through artificial intelligence, data science, and cloud technology.

In addition to its clinical activities, Mayo Clinic plays a significant role in the research and study of medical science. Its physicians and researchers collectively publish more than 10.000 articles peer-reviewed articles each year. Among its, *Mayo Clinic Proceedings*[39] is one of the most widely read and cited clinical journals in the field of medicine. Additionally, Mayo Clinic offers an extensive online library[40] of over 1.150 articles (as of May 2025) covering diseases, symptoms, and treatments, providing the public with reliable medical information.

[35] https://www.mayoclinic.org/about-mayo-clinic
[36] https://health.usnews.com/health-care/best-hospitals/articles/best-hospitals-honor-roll-and-overview
[37] https://www.mayoclinichealthsystem.org/about-us
[38] https://www.mayoclinicplatform.org/
[39] https://www.mayoclinicproceedings.org/
[40] https://www.mayoclinic.org/diseases-conditions

3.4.1.2.2 Extraction Process

Mayo Clinic provides a website with an alphabetically organized list of diseases. For each disease, Mayo Clinic presents information divided into different sections and subsections:

- **Symptoms & Causes:** Overview, Symptoms, Causes, Risk factors, Complications, Prevention.
- **Diagnosis & Treatment:** Diagnosis, Treatment, Clinical Trials, Lifestyle and home remedies, Preparing for your appointment.
- **Doctors & Departments:** Departments and specialties, Doctors who treat this condition, Research, Research Profiles.
- **Care at Mayo Clinic:** [disease name] care at Mayo Clinic, The Mayo Clinic experience and patient stories, Expertise and rankings, Locations, travel and lodging, Costs and insurance.

The most phenotypically relevant sections and those extracted by DISNET are *Symptoms*, *Causes*, and *Diagnosis*. These sections contain valuable information for building the phenotypic layer, such as the symptoms a disease presents, its causes, diagnostic processes, and treatments.

Although the list of diseases on the Mayo Clinic site is shorter than that of Wikipedia, it is an official website and thoroughly reviewed by health professionals, making the information highly reliable.

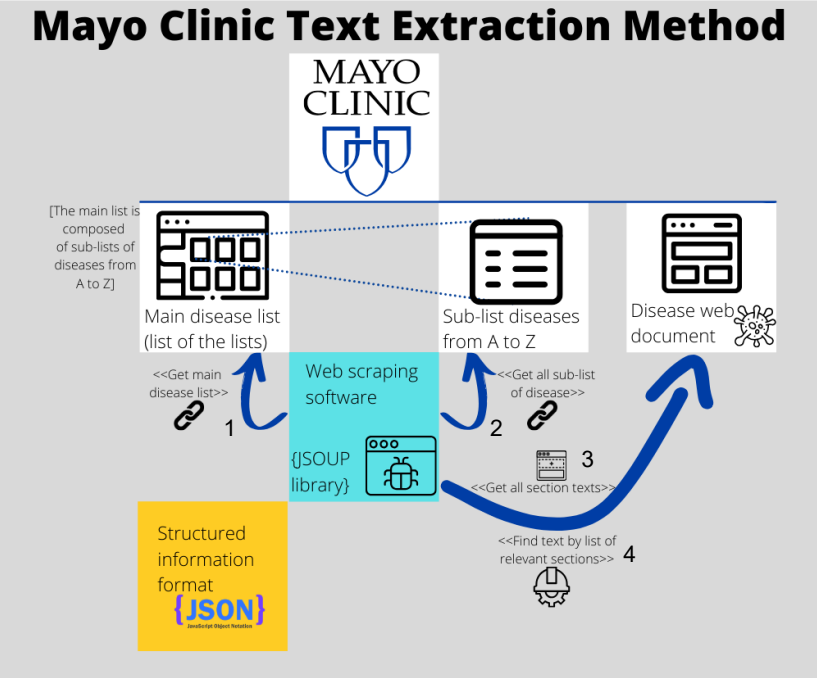


Figure 10: Workflow of the knowledge extraction process from Mayo Clinic.

The overall summary of the extraction process is visually presented in **Figure 10** while each individual step is represented in **Figure 11**, **Figure 12**, **Figure 13**, **Figure 14** and **Figure 15**, using diagrams to indicate the source and structure of the extracted information.

The first step is to **load the configuration** from a file called *sources.xml*, which contains the main URL[41] of the Mayo Clinic diseases site, the relevant sections

[41] https://www.mayoclinic.org/diseases-conditions

(*Symptoms & Causes*, *Diagnosis & Treatment*), and the specific subsections to extract (*Symptoms* and *Causes* from the first section, and *Diagnosis* from the second). The section information is saved into a *MenuItem* object which contains its name and a list of *Section* objects, which represent the subsections. This process is represented in **Figure 11**.

The second step is to **retrieve alphabetical links**, which is represented in **Figure 12**. In this step, the system navigates to the main diseases URL of the Mayo Clinic website, which offers an alphabetical index for browsing diseases. This index is visually represented as a panel with clickable buttons labelled A through Z, plus an additional button labelled with a hash symbol (#) for conditions that do not start with a standard letter (e.g., numbers or symbols). Each button corresponds to a unique URL containing a query parameter that filters diseases by their starting letter. All links are extracted and stored in a list called *listOfTheDiseaseList*, which holds exactly the 26 link strings (one for each letter and one for non-alphabetical entries).

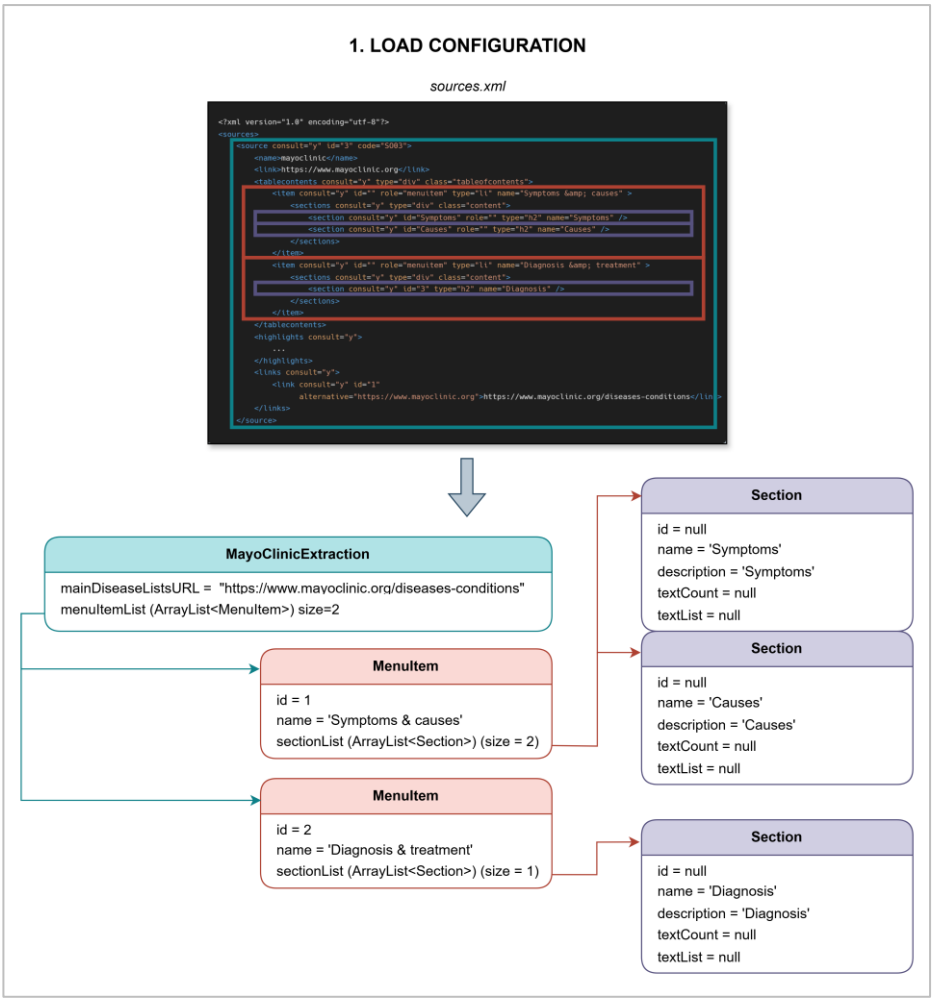


Figure 11: Diagram representing first step in the Mayo Clinic extraction process, showing how the XML configuration file defines the structure of menus and sections to be parsed.

The third step is to **create the disease document list** and is represented in **Figure 13**. Each alphabetical page from the previously retrieved list (A–Z and "#") is visited to collect all disease entries displayed on those pages. For example, the page corresponding to the letter "A"[42] lists diseases like *A fib* (short for *Atrial*

[42] https://www.mayoclinic.org/diseases-conditions/index?letter=A

fibrillation), *abdominal aortic aneurysm* and *abnormally excessive sweating*. In this page, two formats of displaying the diseases can be seen. The first format (without synonyms) shows only the official disease name as a clickable hyperlink leading to its detailed page. The second format (with synonyms) is used when a synonym is presented instead of the official term; in this case, the synonym appears as the main entry, followed by a line that says "See [disease name]", where this disease name is hyperlinked to the corresponding disease page.

Each disease found is processed into a structured object called a *Doc*. A *Doc* object stores essential metadata about the disease, including an identifier (*id*), the date the information was gathered (*date*), and the main URL linking to the disease's dedicated page (*url*), which is saved in a *Link* object. A *Link* object stores the url and a short description. Additionally, the *Doc* object contains a *Disease* object that holds the official name and the synonyms (if it has) of the disease. For instance, *A fib* would be stored as a synonym of *Atrial fibrillation*. All these *Doc* objects are added to a master list called *documentList*, which grows to include all of Mayo Clinic diseases (approximately 1,150 diseases as of May 2025).

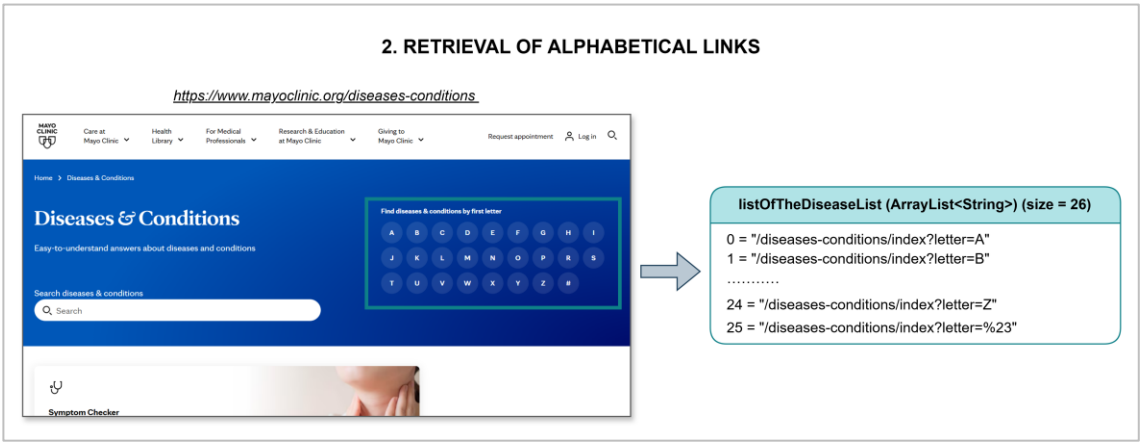


Figure 12: Diagram of step 2 in the Mayo Clinic extraction process, showing the source webpage and the resulting list of disease links.

Once each disease has been identified and stored as a *Doc* object and inside the list *documentList*, the fourth step is to **retrieve the relevant section links** of the diseases. For each disease, the system accesses its main page on the Mayo Clinic website and scans the navigation menu, which includes the sections previously explained. From these options, only the relevant sections are used. The system identifies the URLs associated with these relevant menu items and saves them in the *urlList* attribute of the corresponding *Doc* object.

The final step of the Mayo Clinic extraction process is to **extract the disease's subsections content**. In this step, the previously collected URLs for each disease's relevant sections are accessed. The HTML structure of each page is parsed to locate the main content area, and within that, the system identifies the text that belongs to the relevant subsections. The content is typically organized in the form of paragraphs or bullet-point lists. Each individual block of text, whether a paragraph or a list, is transformed into a *Text* object and categorized accordingly. Each of these *Text* objects is stored within a *Section* object that corresponds to its type (e.g., Symptoms, Causes, or Diagnosis).

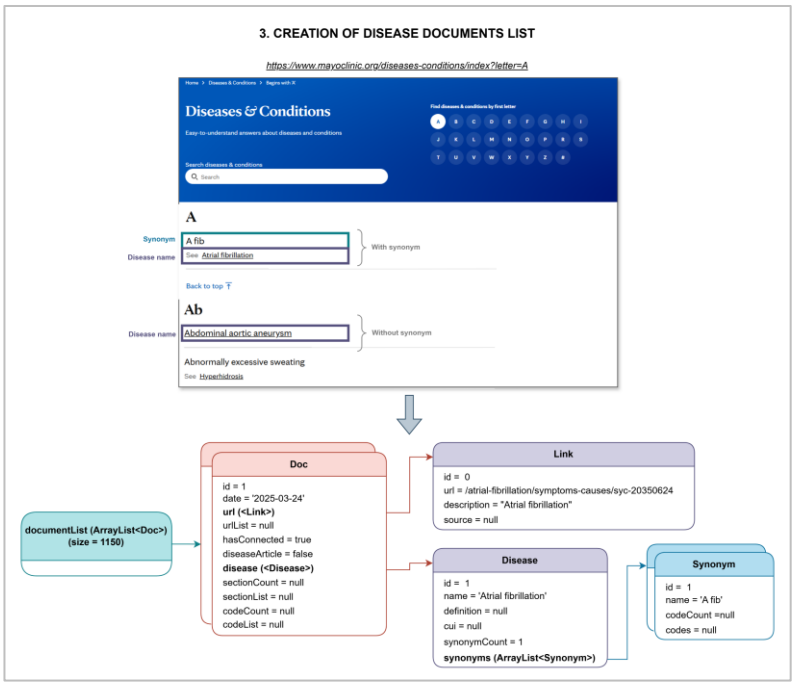


Figure 13: Diagram of step 3 in the Mayo Clinic extraction process, showing the creation of the disease documents list from each alphabetical disease list.

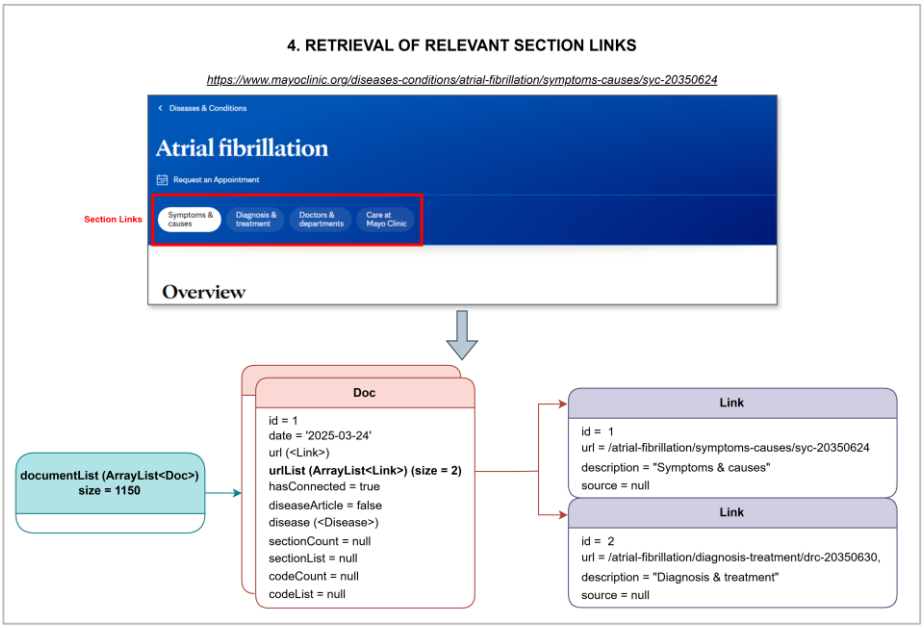


Figure 14: Diagram showing the fourth step of the Mayo Clinic extraction process, illustrating the extraction of the disease's subsection links.

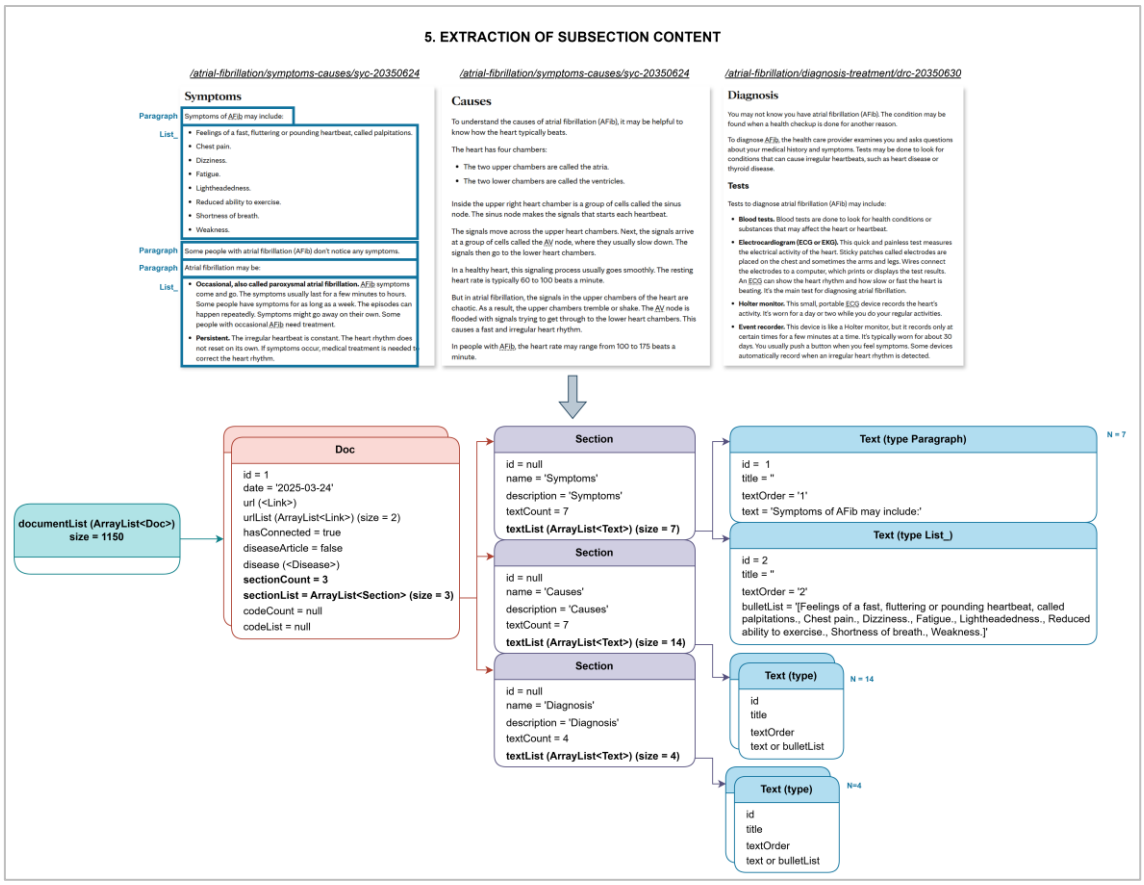


Figure 15: Diagram showing the last step of the Mayo Clinic extraction process, illustrating the extraction and structuring of subsection content from each disease document.

3.4.1.3 MetaMap Process

3.4.1.3.1 MetaMap

MetaMap is a program created by the U.S. National Library of Medicine that maps biomedical text to concepts in the UMLS Metathesaurus, the largest thesaurus in the biomedical domain [53]. It processes unstructured text and identifies medical terms, linking them to standardized UMLS concepts and identifiers. To do this, MetaMap applies a knowledge-based method that relies on symbolic, NLP and computational linguistic techniques.

3.4.1.3.2 MetaMap Process

The MetaMap processing workflow involves three components: the MMC, the MMI, and the MMS.

3.4.1.3.2.1 MetaMap NLP

The process begins when the DSVP module sends a request to the MMS container, including a list of clinical texts and the configuration parameters for MetaMap. Upon receiving the request, MMS first validates the configuration, ensuring that the specified terminological source and semantic types are supported and correctly defined.

Once validated, each text is preprocessed by cleaning special characters and preparing the input for analysis. The MMC then delegates the semantic analysis task to MMS, which performs the actual natural language processing. The MMS extracts medical concepts from each text, including standardized identifiers (CUIs), concept names, semantic types, matched terms, and their positions in the text. The output is structured and linked back to each original input.

The full set of processed texts and their identified concepts is returned as a comprehensive response object and stored as a JSON file within DISNET's persistent storage.

3.4.1.3.2.2 MetaMap Insertion

The MMI process allows the insertion of clinical symptoms detected by MetaMap into the database from medical texts. The system loads the previously generated JSON file from MMC, which contains the identified texts and concepts. Symptoms and semantic types are inserted (avoiding duplicates), and for each text containing concepts, *HasSymptom* objects are created to link the text with the unique symptoms, including matched words and their positions.

3.4.1.4 Validation Process

The TVP Validation process validates clinical symptoms extracted from medical texts by comparing them against a knowledge base stored in the *allFindings.fd* file, which contains standardized medical terms along with their CUIs, synonyms, and ontology codes.

First, the DSVP module retrieves all symptoms stored in the database for a given source and snapshot date. Duplicate entries are removed to generate a list of unique symptoms, which is then sent to the TVP system for validation. Within TVP, each symptom is compared to the knowledge base using three methods: exact match by CUI, match by name or synonyms, and textual similarity. If a match is found, the symptom is considered validated.

Validated symptoms are then marked in the database, and a summary of the process is saved, indicating the total number of terms found, how many were unique, and how many were successfully validated. This procedure ensures that only clinically relevant and trustworthy concepts are integrated into the system's knowledge base.

3.4.2 Website

The DISNET website offers the sections mentioned below. Screenshots of the interfaces of this sections can be found on *8.3 Annex C: Original DISNET website*.

- **Home page:** the landing page of the DISNET platform. **Figure 16** shows the interface of this section and includes the navigation bar used to access the other areas of the site.
- **API:** provides the API documentation.
- **Visualize data:** offers interactive tools to explore and visualize the DISNET knowledge base in a graph format. Users can perform searches using either "Search by disease" or "Search by symptom", allowing them to define a list of diseases or symptoms and view which drugs, genes, pathways, proteins or symptoms they have in common.
- **Drug repurposing:** a section intended to present information related to DISNET in a blog style format.
- **About:** contains three subsections. *Team* showcases current and past members of the DISNET team. *Database* provides information about the DISNET database, including statistics from the phenotypic layer such as the number of extractions per data source or the total number of extracted diseases. *Resources* lists publications, seminars, theses, and talks related to the DISNET project.

- **Log in / Sign up:** allows users to create an account or log in. Originally intended to let users save their API queries for future reference.

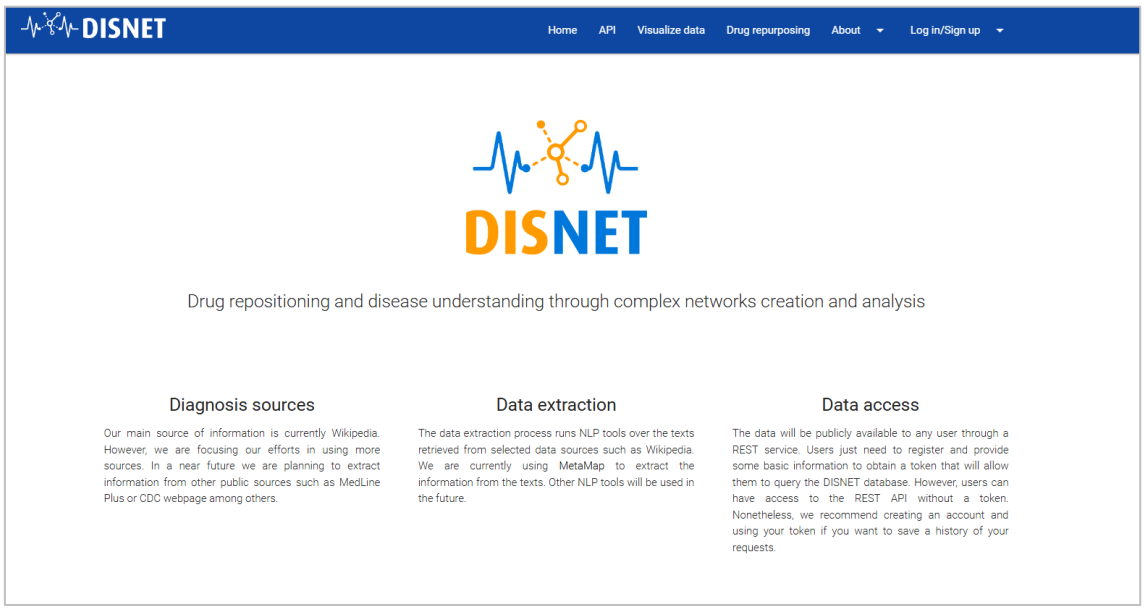


Figure 16: Home page of the DISNET platform.

3.5 Identification of Problems

This Master's Thesis originated from a growing accumulation of issues encountered during the use of DISNET. With the original creator no longer actively involved and the absence of a dedicated developer to maintain the system, several problems began to surface.

- **Failure when connecting to databases**: during the execution of the DISNET extraction, some services fail to connect to the databases. The causes are the following:
  - The MySQL databases were upgraded from version 5.7 to version 8.4. However, the JDBC (Java Database Connectivity) driver used in the system was not updated accordingly. As a result, the application continued to use an outdated driver version that lacked support for newer authentication mechanisms introduced in MySQL 8.4. Specifically, this led to *caching_sha2_password* authentication errors, as the old driver was not compatible with the default authentication plugin used by the updated MySQL version.
  - Database authentication errors due to mismatched credentials.
- **Failures in the text extraction processes:** the extraction processes of Wikipedia and Mayo Clinic are failing to function as intended. The causes are the following:
  - The extraction logic is tightly coupled to the specific structure and layout of the web pages (of Wikipedia, Mayo Clinic, and PubMed). When these external websites update their HTML structure or modify how content is displayed, the extraction processes may not be able to locate the required text correctly. Therefore, there's a high risk that unexpected changes could cause the extraction process to fail or misinterpret the content.

- The DBpedia Live service is currently down, which causes the entire extraction process to fail.

- **Failures in the Web Interface:** several functionalities of the DISNET web interface are currently not operational. The sections About > Database and Visualize cannot be accessed. The user registration and login features for new users do not function. Additionally, the deployed version of the website does not match the version stored in the GitLab repository. There are two versions of the interface (one from 2018 and an updated version from 2022) but the system is currently running the older 2018 version. Causes:
  - The About > Database and Visualize sections rely on data from the system's databases, which are unreachable due to the previously mentioned connectivity issue.
  - The user registration and login features are also affected by database connection issues. Moreover, the registration form fails to render the list of countries correctly due to an outdated version of Materialize CSS, Finally, the confirmation email sent during user registration is not delivered, as recent changes in Google's security policies require the use of application-specific passwords.
- **Accumulation of outdated dependencies:** causes:
  - There has been a lack of maintenance due to a prolonged absence of active maintenance which has allowed dependencies to become outdated.
  - The system has been developed and maintained by a single developer which limited the scope for continuous maintenance.
- **Absence of automated testing:** there are currently no automated tests within the system. Causes:
  - Absence of initial testing strategy for the development cycle, possibly due to the system originally being developed by a single developer.
  - Insufficient planning for long term error detection. This can produce the following risks:
    - **Security risks:** outdated dependencies can introduce potential security vulnerabilities.
    - **Performance concerns:** older versions of dependencies might not optimize resource usage as effectively as newer versions.
    - **Maintenance challenges:** outdated dependencies can result in technical debt, where later efforts to upgrade or integrate newer components become more difficult.
- **Maintainability and configuration complexity:** causes:
  - **Fragmentation of functionality:** the system is broken into 12 microservices spread over 16 containers (including 3 different databases and 1 container for MetaMap deployment). Managing service interactions becomes more complex. It complicates configuration, maintainability and monitoring.

3.5.1 Analysis of Metrics

3.5.1.1 SonarQube

SonarQube[43], also known as Sonar, is an open-source platform developed by SonarSource for automated code evaluation and static analysis. It supports more than 30 programming languages and is used to identify issues related to code quality, including maintainability, reliability and security. It informs about duplicated code, code standards, unit testing, code coverage, cyclomatic complexity, comments and software design. It focuses on three aspects of code which are: security, reliability, and maintainability.

Security is the protection of the software from unauthorized access, use, or destruction. Reliability is a measure of how the software is capable of maintaining its level of performance under stated conditions for a stated period of time. Maintainability refers to the ease with which the software code can be repaired, improved and understood.

Apart from these three software quality attributes, SonarQube offers information about the test coverage, duplications, size and complexity of a codebase. Test coverage informs how much of the source code has been covered by test cases. Duplications calculate the percentage of repeated code segments. Size metrics refer to metrics related to the size of the codebase. These metrics are Lines of Code (LOC), number of lines, statements, functions, classes, files, comment lines and the comment lines density.

Referring to complexity, there are two complexity metrics: cyclomatic complexity and cognitive complexity. Cyclomatic complexity calculates the number of paths through the code. For the Java language, it is calculated by incrementing the value by one each time it detects one of these keywords: “if”, “for”, “while”, “case”, “&&”, “||”, “?”, “->”. Cognitive complexity calculates how hard it is to understand the code’s control flow. It is calculated adding one point for each structure that breaks the straight-through reading of code (if, for, while, catch, etc.) and for each time a flow-breaking structure is nested inside another.

Table 2: Code quality metrics per service.

Service	# Security Issues	# Reliability Issues	# Maintainability Issues	% Coverage
DSVP	1	**292**	**7.2k**	14.3%
WTE	1	40	486	1.9%
MCTE	1	24	309	1.8%
PMTE	3	33	327	0.0%
DAP	0	40	379	0.0%
DLEP	2	83	596	0.0%
MMC	1	18	**4.8k**	66.4%
MMI	0	10	227	0.7%
TVP	1	11	139	2.2%
STV	0	26	115	25.1%
MAPI	1	236	**1.4k**	3.9%
DWA	1	318	878	3.9%

[43] https://www.sonarsource.com/products/sonarqube/

Table 3: Codebase size and complexity per service.

Service	# LOC	# Total Lines	# Classes	# Functions	% Comments	Cyclomatic Complexity	Cognitive Complexity
DSVP	**24.966**	**37.476**	**320**	**3.156**	**16.8%**	**3.869**	**2.928**
WTE	4.357	8.659	54	546	34.	862	**999**
MCTE	3.607	6.323	55	515	**22.7%**	719	390
PMTE	4.298	7.187	62	557	**19.0%**	853	398
DAP	5.449	8.001	105	957	9.0%	912	215
DLEP	8.019	11.790	129	1.126	14.2%	1.159	412
MMC	3.111	3.874	27	158	9.0%	215	107
MMI	8.521	1.200	79	738	9.0%	1.025	286
TVP	1.403	2.105	24	151	15.8%	214	109
STV	624	795	15	16	0.6%	20	20
MAPI	**18.347**	**26.593**	295	2.354	14.1%	**2.547**	**1.104**
DWA	**12.058**	**16.669**	165	1.486	9.5%	**1.486**	344

Table 4: Comparison between in-project duplications and cross-project duplications per service.

Service	% In-Project Duplications	% Cross-Project Duplications
DSVP	8.0%	38.6%
WTE	**16.8%**	42.6%
MCTE	13.3%	45.8%
PMTE	2.6%	31.4%
DAP	8.0%	**62.5%**
DLEP	5.9%	22.0%
MMC	0.0%	56.9%
MMI	**14.1%**	**71.4%**
TVP	2.0%	44.3%
STV	4.8%	7.0%
MAPI	5.0%	25.1%
DWA	13.0%	43.1%

Table 2 shows code quality metrics per service. According to this table, it can be seen that the service with the highest number of reliability and maintainability issues is DSVP, the central orchestrator. Additionally, services like PMTE, DAP and DLEP have 0% test coverage, indicating a lack of automated testing, which increases maintainability and the effort to understand these codebases.

Table 3 provides details about the codebase size and complexity for each service. According to this table, DSVP is the also the largest service in terms of LOC and the one with the highest complexity. Additionally, the simplest services in terms of code complexity are STVP, TVP and MMC. WTE and MCTE display high complexity relative to their size.

Table 4 compares in-project and cross-project code duplications. DAP and MMI standout with very high cross-project duplication rates which suggests the need for refactoring and extracting shared components.

4 Proposed Improvement

The purpose of this chapter is to explain the development plan followed to perform the implementation to achieve the restoration and improvement of DISNET.

4.1 Tasks

The stakeholders (supervisors and DISNET contributors) requested the following improvements to the DISNET platform:

1. Restore the Medical Text Extraction Process (MTEP), therefore the Wikipedia (Task 1.1) and Mayo Clinic (Task 1.2) text extraction processes.
2. Restore functionality of the web page and implement defined changes (details of these changes in **Table 5**).
3. Update the JDBC database connectors for database compatibility.
4. Consolidate the three databases into a single unified database.
5. Parameterize and centralize system configuration.
6. Simplify the system's architecture by reducing the number of containers.

Table 5 provides more information about each task, including why is it important to complete it (rationale) and when it is considered done (acceptance criteria).

Table 5: Detailed description of improvement tasks.

Task 1 Restore Medical Term Extraction Process (MTEP) functionality	
Description	The MTEP must be fixed and restored to its original functionality.
Rationale	The MTEP is the core component of the DISNET data pipeline, responsible for extracting, validating and storing data from the unstructured public sources of Wikipedia and Mayo Clinic. This process is currently not operational due to problems in the Wikipedia and Mayo Clinic text extraction services. Restoring the MTEP is critical to ensure that DISNET can achieve its purpose.
Task Summary	• Analyse and understand the MTEP process. • Restore Wikipedia Text Extraction (WTE) functionality (see Task 2.1) • Restore Mayo Clinic Text Extraction (MCTE) functionality (see Task 2.2) • Verify that the MTEP process works as intended.
Acceptance Criteria	The task is considered complete when: • The MTEP works successfully without errors and sends a report email indicating the duration of the different steps of the process.
Priority	Must have

Task 1.1 Restore Wikipedia Text Extraction (WTE) functionality	
Description	The WTE must be fixed and restored to its original functionality.
Rationale	The WTE is one the components of the MTEP. To restore the MTEP functionality is necessary to restore the WTE process.
Task Summary	• Analyse and identify the root causes of the WTE failures. • Debug the system step by step and address errors through restoration of the affected logic. • Verify that the WTE process runs correctly and produces the expected output.
Acceptance Criteria	The task is considered complete when: • WTE completes processing of Wikipedia sources without any errors. • JSON output matches expected historical results in terms of content and structure. That is, content areas (disease names, synonyms, text sections) are correctly extracted • Logs show normal processing behaviour, without critical errors.
Priority	Must have

Task 1.2 Restore Mayo Clinic Text Extraction (MCTE) functionality	
Description	The MCTE must be fixed and restored to its original working functionality.
Rationale	The MCTE is one the components of the MTEP. To restore the MTEP functionality is necessary to restore the MCTE process.
Task Summary or Details	• Analyse and identify the root causes of the MCTE failures. • Debug the system step by step and address errors through restoration of the affected logic. • Verify that the MCTE process runs correctly and produces the expected output.
Acceptance Criteria	The task is considered complete when: • MCTE completes processing of Mayo Clinic sources without any runtime errors. • JSON output matches expected historical results in terms of content and structure. That is, content areas (disease names, synonyms, text sections) are correctly extracted. • Logs show normal processing behaviour, without critical errors.
Priority	Must have

Task 2 Restore functionality of the web page	
Description	The web page must be fixed and updated with several changes.
Rationale	The deployed web page (2018 version) is different from the version on the repository's main branch (2022 version). The 2022 version is preferred because it includes new features, but it currently has some broken functionality: • The data visualization tool is not working due to a failure to connect to the databases. • The login and signup features do not work. • The team members and publications sections are outdated. The following changes are also planned: • There is an API used by the DRIVE website to fetch updated users and publications. This API must also be used to keep the team members and publications current. • The user functionality of login and signup must be removed. • The "Drug Repurposing" section must be removed.
Task Summary	• Identify and fix the source of the issues. • Understand and integrate the API for team members and publications. • Remove the user functionality. • Remove the "Drug Repurposing" and "Login/Signup" sections from the navigation bar.
Acceptance Criteria	The task is considered complete when: • The website works correctly and all the specified changes are implemented.
Priority	Must have

Task 3 Update the JDBC database connectors	
Description	The system services that connect to the database shall be updated to ensure compatibility with the latest JDBC drivers for MySQL version 8.4.
Rationale	The database MySQL version was updated from 5.7 to 8.4. However, the JDBC connectors used by the system have not been updated and have become deprecated. This has led to compatibility issues. A common recurring problem is that recent MySQL versions use the *caching_sha2_password* authentication plugin by default, which replaced the older *mysql_native_password*. However, older JDBC connectors do not support *caching_sha2_password,* which lead to authentication errors.

Task Summary	• Identify all services that establish connections with the databases. • Upgrade the JDBC connectors to versions compatible with MySQL 8.4. • Verify that all services can connect to the databases successfully.
Acceptance Criteria	The task is considered complete when: • All services that interact with the database can connect without errors. • No authentication or compatibility issues occur during connection.
Priority	Should have

Task 4 Consolidate the three databases into a single unified database	
Description	The three databases (Disease List Available Database, Diseases Database and Users Database) shall be migrated into a single database, maintaining the structure of each schema. This process shall be automated by a script.
Rationale	This consolidation aims to centralize data management and simplify maintenance.
Task Summary or Details	Create a script for the automated migration of the *addb* and *disnetdb* schemas into the Diseases Database that performs the following steps: • Exports the addb and disnetdb schemas (with all their data) and imports them into the Diseases Database. • Migrates users and permissions. • Deletes the old container databases and volumes.
Acceptance Criteria	The task is considered complete when: • The schemas *addb* and *disnetdb* are inside the Diseases Database. • The users and permissions from the Disease List Available Database and the Users Database have been migrated to the Diseases Database. • The containers and volumes of the Disease List Available Database and the Users Database are deleted.
Priority	Should have

Task 5 Parameterize and centralize configuration	
Description	The system configurations shall be parameterized, that means any hardcoded values in the code must be extracted and moved into configuration files. Additionally, these configuration files should be centralized and managed at the environment level.

Rationale	The system runs across a large number of containers, but only one or two developers are responsible for managing them. If the configuration files are scattered across different locations, it becomes hard to manage them. By parameterizing and centralizing configurations, it becomes much easier to update and maintain the system. This also reduces the effort required to apply changes.
Task Summary or Details	• Identify all hardcoded configuration parameters in the codebase. • Refactor the code to move these parameters into configuration files. • Standardize the format and structure of the configuration files. • Centralize all configuration files at the environment level. • Verify that the system functions correctly after refactoring and centralization.
Acceptance Criteria	The task is considered complete when: • Relevant hardcoded parameters have been moved to configuration files. • Configuration files follow a consistent structure and are centrally managed. • The system runs correctly after applying the changes.
Priority	Could have

Task 6 Simplify the system's architecture by reducing the number of containers	
Description	The number of containers (16) shall be reduced.
Rationale	The current fragmentation increases complexity in development and maintenance. Some services have related code that could be logically grouped to simplify the structure.
Task Summary or Details	• Identify services related in functionality. • Create the new architecture. • Merge defined services into a single container. • Update Docker compose and related configuration files to reflect the new architecture. • Ensure existing functionality remains unaffected.
Acceptance Criteria	The task is considered complete when: • The total number of containers has been significantly reduced (from 16 to less than 10) without functional loss. • The MTEP process has not been affected and works correctly.
Priority	Should have

4.1.1 New architecture

As mentioned in the defined tasks, a decision to unify multiple services and reduce the number of Docker containers was made. The current architecture with many separate containers introduces complexity and makes the system harder to maintain. In many cases, containers were isolated despite serving related functionalities, resulting in duplicated configurations and increased maintainability.

The main goals of this architectural reorganization are to improve maintainability, configurability and operability. By consolidating related services, the new design reduces the number of configuration files, the effort for logging and monitoring and simplifies the deployment process. Additionally, it tries to improve the onboarding for new developers, because a less fragmented architecture is easier to understand.

But it also can introduce limitations. If one feature crashes or misbehaves, it could bring down the whole unified service, rather than just one isolated container.

As part of Task 4, the following database unifications have been planned:

1. Disease List Available Database, Diseases Database and Users Database will be unified into **Diseases Database** to centralize the database management.

As part of Task 5, the following service unifications have been planned:

2. MetaMap Inserts (MMI), MetaMap Service (MMS) and MetaMap Client (MMC) will be unified into **MetaMap (MM)** to unify the MetaMap functionality.
3. Semantic Type Validation (STV) and Term Validation Process (TVP) will be unified into **Validation (VAL)** to unify the validation functionality.
4. Disease List Extraction Process (DLEP), Wikipedia Text Extraction (WTE), Mayo Clinic Text Extraction (MCTE) and PubMed Text Extraction (PMTE) will be unified into **Text Extraction (TE)** to unify the text extraction for the phenotypic layer functionality.

With this restructuring, the system will be reduced from 16 containers to 8. The new architecture is shown in **Figure 17**.

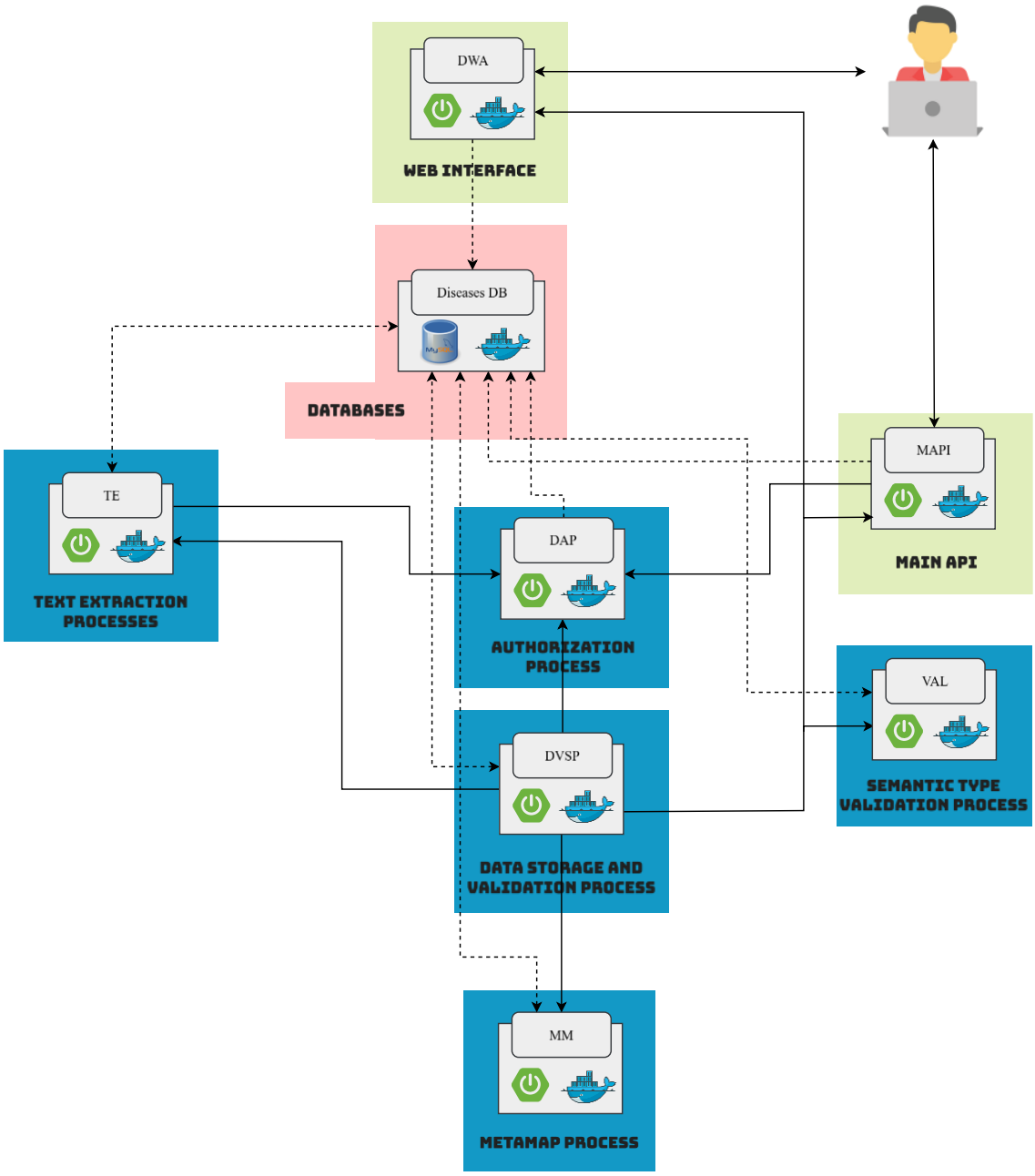


Figure 17: DISNET new system architecture.

4.2 Improvement Plan

Figure 18 presents a Gantt chart detailing the planning activities and development tasks involved in the implementation of the improvement plan, scheduled between February and June.

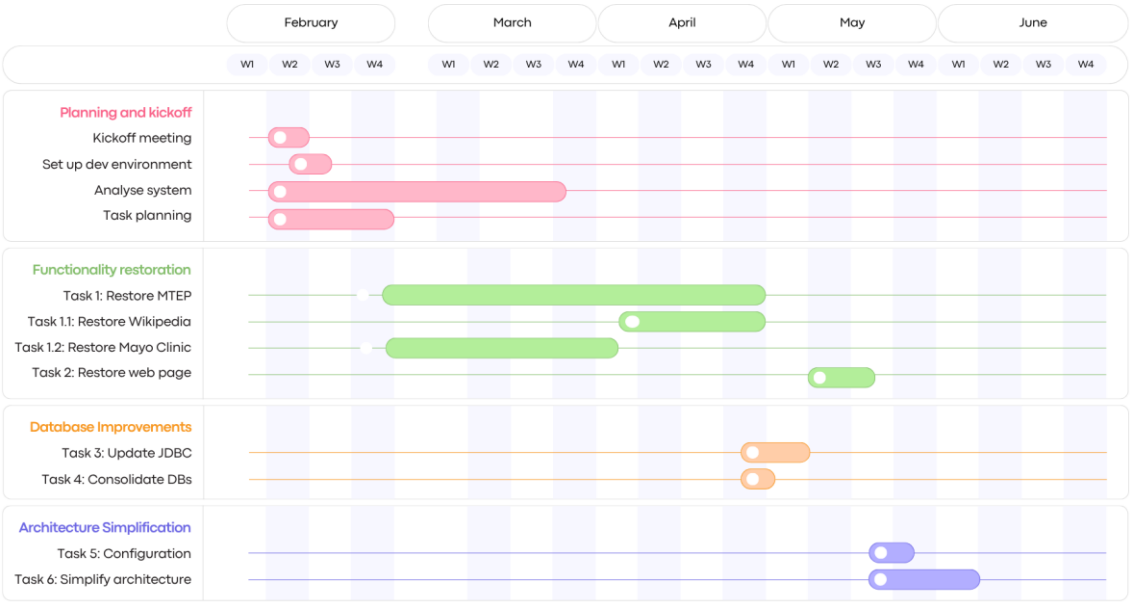


Figure 18: Gantt chart for the development of the defined tasks.

4.3 Development

In the development phase, a combination of tools was employed to debug the system effectively and identify execution errors: P6spy and JVM debugging configurations. **P6spy** is a framework that enables database activity to be seamlessly intercepted and logged with no code changes to existing applications. In the development process, it was used to log all executed SQL queries, which helped in identifying slow or incorrect queries and understanding how the application interacts with the database in real time.

In parallel, **JVM debugging** options were used to enable remote debugging. This allowed to connect a remote debugger (integrated in IDEs like IntelliJ or Eclipse) to the running application. This setup made it possible to debug the services running inside remote Docker containers allowing to investigate the behaviour of the real deployed system.

The following subsections explain how the defined tasks have been resolved and the problems that emerged.

4.3.1 Task 1

4.3.1.1 Task 1.1

Bug 1.1.1

A *NullPointerException* occurs when performing the DBpedia extraction.

Expected result of the bug
The system should continue with the processing of diseases from DBpedia even when DBpedia Live is down.

```
Total: 9520
End population.
2025-05-24T05:45:28.780711052Z Update list with the disease Safe List
```

Actual result
The system is crashing due to the NullPointerException and does not continue with the processing of the extracted diseases from DBpedia.

```
java.lang.NullPointerException: null
```

Root cause analysis
When DBpedia Live is down, the query result is null. Since there is no null check implemented, a *NullPointerException* is thrown during processing.

Fix Details
The Populate.java class was updated to check if the DBpedia Live query result is null before proceeding with processing.

Bug 1.1.2

The DBpedia service is not extracting all of the available diseases in DBpedia.

Expected result of the bug
The system should retrieve all disease entries available in DBpedia.

Actual result
The system retrieves only 10.000 entries.

Root cause analysis
DBpedia's SPARQL endpoint has a limit of 10,000 results per query. The existing implementation does not take into this limitation, which leads to incomplete data retrieval. The solution to retrieve all diseases is to implement pagination.

Fix Details
Pagination was added to the *GetDiseasesFromDBPedia.java* class to handle all the available diseases. The updated query appends pagination parameters:

```
String pagedQuery = baseQuery + "\nORDER BY ?d\nLIMIT " + pageSize + "\nOFFSET " +
offset;
```

Bug 1.1.3

The DBpedia service is extracting invalid codes such as "**-**", "**,**" or whitespace.

Root cause analysis
Many disease entries in DBpedia contain codes such as "**-**", "**,**" or whitespace, which likely serve as placeholders for missing or null values. These codes are currently being extracted, which messes data processing. It is preferable to treat these cases as null values rather than retaining the symbols.

Fix Details
The SPARQL query was edited to filter out non alphanumeric codes using regular expressions. This ensures only meaningful codes are returned.

```
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX category: <http://dbpedia.org/resource/Category:>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT ?d ?dn ?wikiPage ?icd9 ?icd10 ?disDB ?meshID ?mlp ?omim ?emed ?frbase
WHERE {
          ?d a <http://dbpedia.org/ontology/Disease>.
          ?d foaf:isPrimaryTopicOf ?wikiPage.

          OPTIONAL {
              ?d <http://dbpedia.org/ontology/icd9> ?icd9.
              FILTER regex(str(?icd9), "[A-Za-z0-9]", "i")
          }
          OPTIONAL {
              ?d <http://dbpedia.org/ontology/icd10> ?icd10.
```

```
            FILTER regex(str(?icd10), "[A-Za-z0-9]", "i")
        }
        OPTIONAL {
            ?d <http://dbpedia.org/ontology/diseasesDB> ?disDB.
            FILTER regex(str(?disDB), "[A-Za-z0-9]", "i")
        }
        OPTIONAL {
            ?d <http://dbpedia.org/ontology/meshId> ?meshID.
            FILTER regex(str(?meshID), "[A-Za-z0-9]", "i")
        }
        OPTIONAL {
            ?d <http://dbpedia.org/property/medlineplus> ?mlp.
            FILTER regex(str(?mlp), "[A-Za-z0-9]", "i")
        }
        OPTIONAL {
            ?d <http://dbpedia.org/ontology/omim> ?omim.
            FILTER regex(str(?omim), "[A-Za-z0-9]", "i")
        }
        OPTIONAL {
            ?d <http://dbpedia.org/property/emedicinetopic> ?emed.
            FILTER regex(str(?emed), "[A-Za-z0-9]", "i")
        }

        OPTIONAL {
                ?d owl:sameAs ?frbase.
                FILTER(regex(str(?frbase), "freebase", "i")).
        }
      OPTIONAL {
              ?d rdfs:label ?dn.
              FILTER (lang(?dn) = 'en')
      }
}
```

Bug 1.1.4

The DBpedia service is incorrectly deleting certain disease entries.

Root cause analysis

Disease entries in DBpedia are being stored in a Map where the key is a LinkedList of the disease codes and the value is the Disease object. However, this structure causes different problems.

- Entries with different codes were sometimes treated as identical which causes some diseases to overwrite others. This is caused because the Code class treats two objects as equal if they share even one matching key, but its *hashCode* is based on all keys. This mismatch breaks how the map work and causes different disease entries to overwrite each other.
- When two diseases share the same set of codes, only one is saved and the other is discarded. For example, this is happening with diseases Achondrogenesis and Achondrogenesis type 1B.

Fix Details

The data structure was changed from a Map<Code, Disease> to a List<Disease>. The Disease object contains attributes to save the respective codes. This way, all disease entries are saved even if they share or have similar codes.

Bug 1.1.5

The Wikipedia service is not extracting text correctly due to changes in the HTML structure of Wikipedia articles.

Root cause analysis
Wikipedia has updated the HTML structure of its pages. Previously, section titles (<h2>, <h3>, <h4>) were direct elements. Now, these headers are nested inside containers. As a result, the system fails to locate section titles and misses content sections. Also, when the system is extracting text from a section and encounters the title of the next one, it fails to recognize it as a new section. Instead, it continues extracting text as if it still belongs to the previous section which results in mismatched content.

Fix Details
The extraction logic was updated to handle the new HTML design. A method was added to detect header tags (<h2>, <h3>, <h4>) nested inside parent elements. This way, the sections titles are correctly identified and extracted. Also, the system detects the start of the next section and therefore stops to extract content if it is not a relevant section.

4.3.1.2 Task 1.2

Bug 1.2.1

The alphabetical disease links are not being extracted from the Mayo Clinic source.

Expected result of the bug
The system should extract a list of alphabetical links from the Mayo Clinic page. Output log:

```
26 (elements) lists of diseases recovered.
```

Actual result
Output log:

```
0 (elements) lists of diseases recovered.
```

Root cause analysis
The HTML structure of the Mayo Clinic page has changed. The system was not updated to reflect these modifications, and therefore it fails to locate the correct elements.

Specifically, the alphabetical links were previously located within a tag with the class "*acces-alpha*". They are now found within a tag with the class "*cmp-alphabet-facet—inner*".

Fix Details
The sources.xml configuration file was updated to reflect the new class name for the HTML tag.

Original line:

```
<tag consult="y" id="" class="acces-alpha" type="ol" desc="disease_list" />
```

Updated line:

```
<tag consult="y" id="" class=" cmp-alphabet-facet—inner" type="ol"
desc="disease_list" />
```

Bug 1.2.2

A *NullPointerException* is thrown during the process of creating the disease document list.

Expected result
The system should extract the list of diseases inside an alphabetical page. Output log:

```
1 to . mayoclinicExtract https://www.mayoclinic.org/diseases-conditions/index?letter=A ==> OK(200)
1. disease: Atrial fibrillation (https://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624)
```

Actual result
Output log:

```
1 to . mayoclinicExtract https://www.mayoclinic.org/diseases-conditions/index?letter=A ==> OK(200)
ERROR 1 --- [nio-8080-exec-1] e.c.u.m.component.MayoClinicExtraction : Error creating the document list in extract() method => java.lang.NullPointerException
```

Root cause analysis
The structure of the Mayo Clinic webpage has changed. The system was still referencing outdated HTML identifiers and therefore could not locate the necessary content. Before, the list of diseases was located in an element with id="index". Now, the content is found in an element with *id="cmp-skip-to-main__content"*. Because the system looks for a non-existent element, it retrieves null and attempts to access it, triggering a *NullPointerException*.

Fix Details
The *sources.xml* file was updated to reflect the new HTML structure.

Original line:

```
<tag consult="y" id="index" class="index" type="div" desc="disease_sub_list" />
```

Updated line:

```
<tag consult="y" id="cmp-skip-to-main__content" class="index" type="div" desc="disease_sub_list" />
```

Bug 1.2.3

A connection error occurs when attempting to access disease URLs from the Mayo Clinic site.

Expected result
Each disease link should be accessed successfully. Output log:

```
1. mayoclinicLinksDiseaseExtract https://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624 ==> OK(200)
2. mayoclinicLinksDiseaseExtract https://www.mayoclinic.org/diseases-conditions/abdominal-aortic-aneurysm/symptoms-causes/syc-20350688 ==> OK(200)
3. mayoclinicLinksDiseaseExtract https://www.mayoclinic.org/diseases-conditions/hyperhidrosis/symptoms-causes/syc-20367152 ==> OK(200)
...
```

Actual result
Output log:

```
ERROR 1 --- [nio-8080-exec-1] e.c.upm.midas.component.ConnectDocument : Exception to connect with the page: (https://www.mayoclinic.orghttps://www.mayoclinic.org/diseases-conditions/atrial-fibrillation/symptoms-causes/syc-20350624)
```

```
2025-06-04T11:50:15.628544124Z
2025-06-04T11:50:15.628545902Z org.jsoup.HttpStatusException: HTTP error fetching URL
```

Root cause analysis
The error occurs due to incorrect URL concatenation in the *createCompleteURL* method. The method appends the relative path to the base URL, but in this case, the *link.attr("href")* already contains the full absolute URL. This leads to a malformed result with the base URL repeated.

Fix Details
Update the assignment of *diseaseUrl* to avoid concatenating the base URL unnecessarily.

Original line:

```
diseaseUrl = createCompleteURL(link.attr(Constants.HTML_HREF).trim());
```

Updated line:

```
diseaseUrl = link.attr(Constants.HTML_HREF).trim();
```

Bug 1.2.4

Error 403 FORBIDDEN is returned when connecting to the Mayo Clinic articles.

Root cause analysis
The extraction process makes a high number of requests to the Mayo Clinic website (3 connections per disease and there are approximately 1200 diseases, therefore 3600 requests). This caused the website to block excessive requests and return HTTP 403 FORBIDDEN errors.

Fix Details
To address this, a randomized delay was implemented between requests to prevent being blocked. Additionally, in the event of a failed request, the system now retries up to three times with a 30 second delay between attempts.

Bug 1.2.5

There is no information from the “Diagnosis" section for any article.

Root cause analysis
The *sources.xml* file is using the id value “3” for the description “Diagnosis”.

Fix details
The id value in the *sources.xml* was changed from “3” to “Diagnosis”.

Bug 1.2.6

The final result contains duplicated synonyms and the page of diseases starting with symbols is not being processed.

Root cause analysis
This bug is caused because URLs for diseases starting with symbols are being encoded incorrectly, which resulted in URLs finished in */index?letter=%2523* instead of the correct */index?letter=%23*. The double encoding caused the application to process the page for diseases starting with letter “A” instead of the pages starting with symbols. As a result, the “A” page was processed twice, which led to duplicated synonyms.

Fix details
This bug was resolved by upgrading jsoup (used to parse HTML) to version 1.19.1 (previously version 1.10.2 was used).

4.3.2 Task 2

Task 2.1

Remove the user functionality.

Fix details
After the fix, the user functionality remains in the codebase but is has been disabled. Therefore, no user can access the login, signup or profile features from the interface. The endpoints of these features have been protected to prevent public access using the class *SecurityConfiguration*, which is used to configure web security using Spring Security. This approach allows for easier future reactivation of the functionality if needed.

Task 2.2

Integrate the API for team members and publications.

Fix details
The controllers responsible for displaying the team members and publications have been refactored to retrieve information dynamically from an external API instead of relying on static content.

A Feign client was implemented to communicate with the external API, with methods to retrieve all team members, individual team member data by id and the resources categorized in publications, talks, theses or others. The *TeamService* and *ResourceService* classes were updated to consume this client and incorporate the business logic for filtering the data. The processed data is injected into the model attributes for rendering the Thymeleaf templates of the web page views. The HTML templates were also updated to dynamically iterate over the lists of team members and publications.

The configuration of the endpoints, host, user and password to connect to the API were externalized in the corresponding *application.properties* and *.env* files.

Task 2.3

Remove the “Drug Repurposing” and “Login/Signup” sections from the navigation bar.
Fix details
The html file for the header of the webpage (*header.html*) was edited to not display the “Drug Repurposing” and “Login/Signup” sections.

4.3.3 Task 3

The objective of Task 3 is to update the JDBC database connectors. This task is necessary because the MySQL version of the databases was updated from MySQL 5.7 to MySQL 8.4 but the JDBC connectors were not. As a result, compatibility issues emerged. The most important of these issues is related to authentication. When some services attempt to establish a database connection, it resulted in the following error:

```
Caused by: java.sql.SQLException: Unable to load authentication plugin
'caching_sha2_password'.
```

This error stems from the fact that MySQL 8.4 uses *caching_sha2_password* as the default authentication plugin, which is not supported by older versions of the MySQL Connector.

Initially, the connection of the services with the databases was facilitated by the *mysql-connector-java* dependency, which was included in the *pom.xml* file without specifying an explicit version. As a result, the version of the MySQL connector was determined by the version of Spring Boot used in the project. For example, Spring Boot version 1.5.4.RELEASE corresponded to MySQL Connector version 5.1.42[44], while Spring Boot version 2.5.2 corresponded to version 8.0.25[45].

The original dependency was configured as follows:

```
<dependency>
<groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
          <scope>runtime</scope>
</dependency>
```

To address Task 3, resolve authentication issues and ensure consistent compatibility across all services, the MySQL connector dependency was updated to reference the latest available version: 9.3.0. The new configuration is:

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>9.3.0</version>
</dependency>
```

4.3.4 Task 4

The goal of Task 4 is to develop a Bash script that automatically consolidates the three system's databases into a single unified database. The script must also handle the migration of users along with their associated permissions and remove the old database volumes once the process is complete. The final coded script can be found on the Gitlab repository created to show the work of this Master's thesis[46] under the name of *unify_disnet_databases.sh*.

4.3.5 Task 5

The goal of Task 5 is to parameterize and centralize the system's configuration.

To achieve parameterization, all hardcoded values within the codebase were externalized into the corresponding *application.properties* files. This allows configuration to be modified without altering the source code.

For configuration centralization, these *application.properties* files were moved from within the services to the microservices environment. To ensure that each service can locate its configuration file, the following option was added to the Dockerfile used during container startup:

```
--spring.config.location=file:/properties_config/
```

[44] *https://docs.spring.io/spring-boot/docs/1.5.4.RELEASE/reference/html/appendix-dependency-versions.html*
[45] *https://docs.spring.io/spring-boot/docs/2.5.2/reference/html/dependency-versions.html#dependency-versions*
[46] *https://medal.ctb.upm.es/internal/gitlab/disnet/tfm_virginia_gonzalez_diez/blob/master/unify_disnet_databases.sh*

4.3.6 Task 6

The purpose of Task 6 is to simplify the system's architecture by reducing the number of containers. This way, the work is to consolidate multiple services into a single unified service, as outlined in section *4.1.1. New architecture*.

To implement the consolidation, the following steps were carried out:

1. All services targeted for consolidation were updated to use Spring Boot version 2.5.2.
2. The dependencies of the services being merged were added to the *pom.xml* file of the target service.
3. Package names were standardized to enable integration of classes from different services.
4. The source code files from the original services were copied into the unified service.
5. The main class for the unified service was defined.
6. The *application.properties* file was modified to include all necessary configurations from the merged services.
7. The *docker-compose.yml* file was updated to remove containers corresponding to the deprecated services, remove outdated *depends_on* references and add the necessary volume configurations. The final version of the docker-compose file is available in the GitLab repository[47].
8. The *.env* file was updated to reflect the new paths and variables related to the consolidated service.

The code for the unified services can be found on the services folder of the GitLab repository[48].

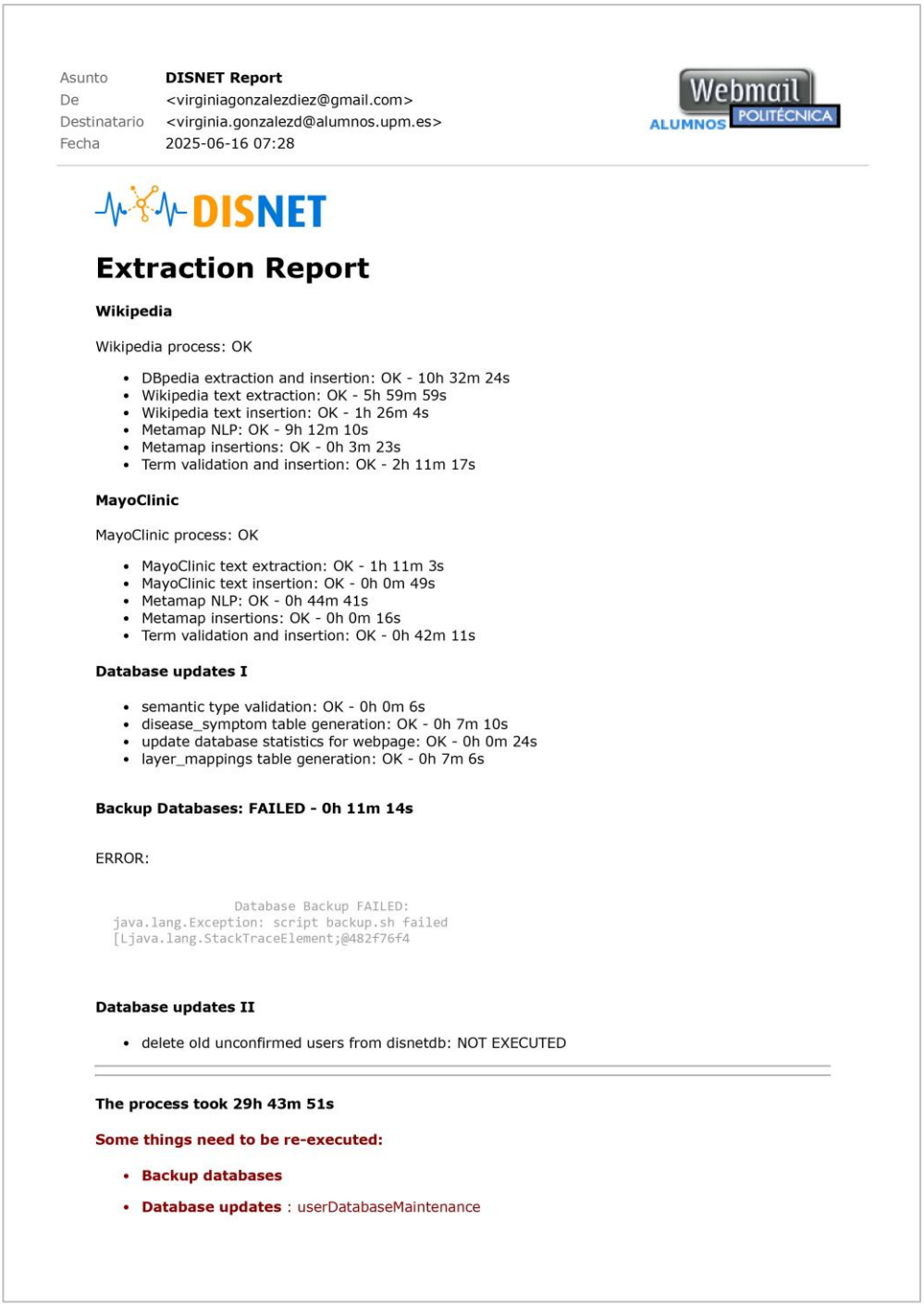
[47] *https://medal.ctb.upm.es/internal/gitlab/disnet/tfm_virginia_gonzalez_diez/blob/master/docker-compose.yml*
[48] *https://medal.ctb.upm.es/internal/gitlab/disnet/tfm_virginia_gonzalez_diez/tree/master/services*

5 Results

The purpose of this chapter is to show the results of the development phase which focused on fixes and improvements of the DISNET system.

5.1 Medical Text Extraction Process

The MTEP works successfully without errors and sends a report email indicating the duration of the different steps of the process. On **Figure 19**, the final report shows the MTEP process correctly extracts and processes content from Wikipedia and Mayo Clinic. Therefore, "Phase 1: text extraction" of the MTEP has been successfully restored.



Asunto **DISNET Report**
De <virginiagonzalezdiez@gmail.com>
Destinatario <virginia.gonzalezd@alumnos.upm.es>
Fecha 2025-06-16 07:28

Webmail ALUMNOS POLITÉCNICA

DISNET

Extraction Report

Wikipedia

Wikipedia process: OK

- DBpedia extraction and insertion: OK - 10h 32m 24s
- Wikipedia text extraction: OK - 5h 59m 59s
- Wikipedia text insertion: OK - 1h 26m 4s
- Metamap NLP: OK - 9h 12m 10s
- Metamap insertions: OK - 0h 3m 23s
- Term validation and insertion: OK - 2h 11m 17s

MayoClinic

MayoClinic process: OK

- MayoClinic text extraction: OK - 1h 11m 3s
- MayoClinic text insertion: OK - 0h 0m 49s
- Metamap NLP: OK - 0h 44m 41s
- Metamap insertions: OK - 0h 0m 16s
- Term validation and insertion: OK - 0h 42m 11s

Database updates I

- semantic type validation: OK - 0h 0m 6s
- disease_symptom table generation: OK - 0h 7m 10s
- update database statistics for webpage: OK - 0h 0m 24s
- layer_mappings table generation: OK - 0h 7m 6s

Backup Databases: FAILED - 0h 11m 14s

ERROR:

```
Database Backup FAILED:
java.lang.Exception: script backup.sh failed
[Ljava.lang.StackTraceElement;@482f76f4
```

Database updates II

- delete old unconfirmed users from disnetdb: NOT EXECUTED

The process took 29h 43m 51s

Some things need to be re-executed:

- **Backup databases**
- **Database updates** : userDatabaseMaintenance

Figure 19: Extraction report after restoring functionality.

The following *database updates I* tasks correspond to "Phase 2: cross-layer mapping and symptom validation", which also perform correctly. However, the *backup databases* task, that belongs part of the "Phase 3: backup and reporting", reports an error because the NAS is not currently available. This issue is known but not considered critical for this work.

The extraction process also generates JSON output files containing the results from the different extraction services. These files are available in a dedicated OneDrive folder[49]. The quantitative results are displayed in **Figure 20**, which compares the number of diseases extracted from the three different sources of DBpedia, Wikipedia and Mayo Clinic at two points in time: January 2023 (last known extraction before this work) and June 2025.

- DBpedia shows the most significant increase in the number of diseases due to the implementation of pagination. This allowed to process all DBpedia entries which had been omitted previously.
- Wikipedia also experienced an increase as a result of the increase in DBpedia entries. However, the number of processed diseases from Wikipedia is lower than the number of DBpedia entries because some of these articles that DBpedia retrieves do not include the relevant medical sections and are therefore excluded. Some other DBpedia entries contain malformed or outdated URLs (often due to articles being deleted or moved on Wikipedia).
- Mayo Clinic suffered a reduction in the number of extracted documents. This is due to the fact that previously the synonyms were being handled incorrectly: in January 2023, synonym entries were incorrectly saved as separate diseases rather than being linked to the primary disease entity. This issue has been addressed which has led to a reduced number of extracted diseases but improved data accuracy.

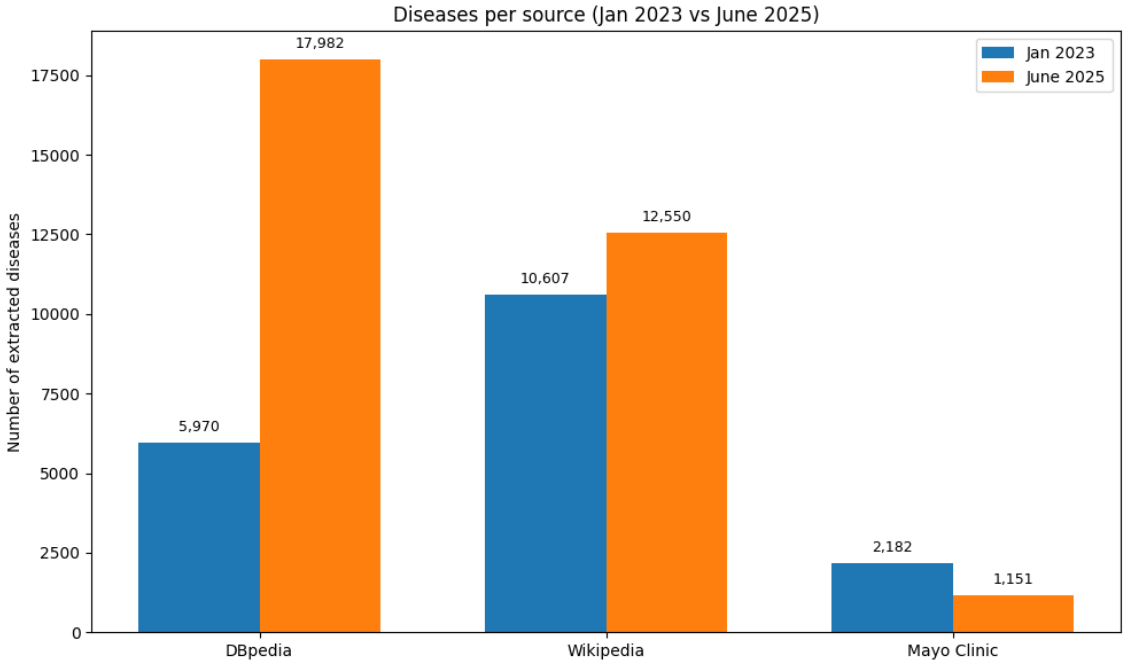


Figure 20: Extracted diseases per source (January 2023 vs June 2025).

[49] https://short.upm.es/ox4yj

5.2 Web interface

The web interface now incorporates all the implemented changes, as can be seen on **Figure 21, Figure 22** and **Figure 23.** The navigation bar has been updated to show the four intended sections of the website. Additionally, the Team Members and Publications pages now show up to date information dynamically retrieved from the API.

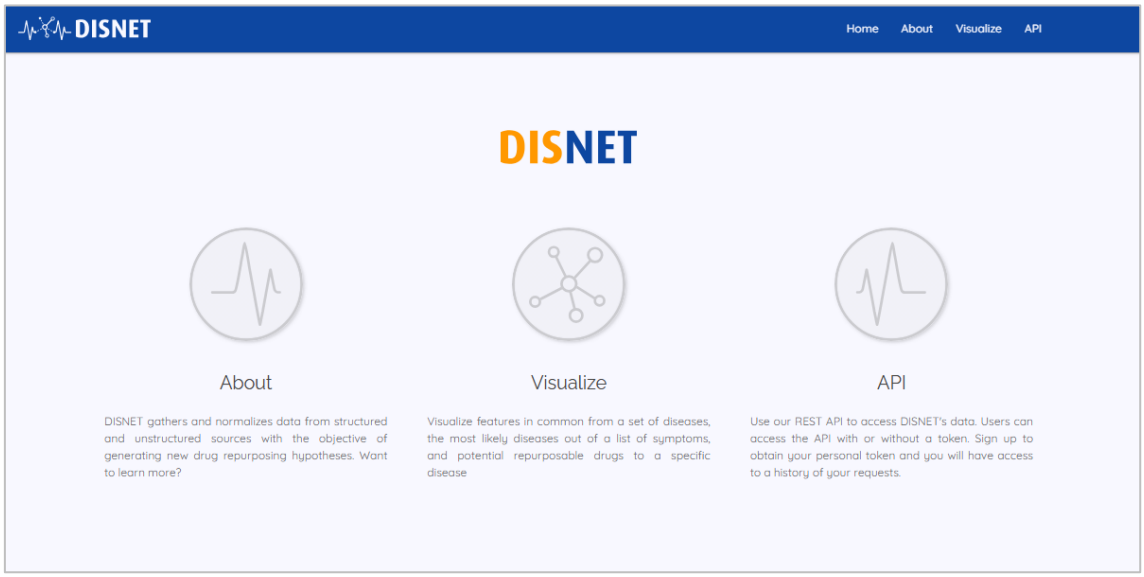


Figure 21: Updated DISNET main web page.

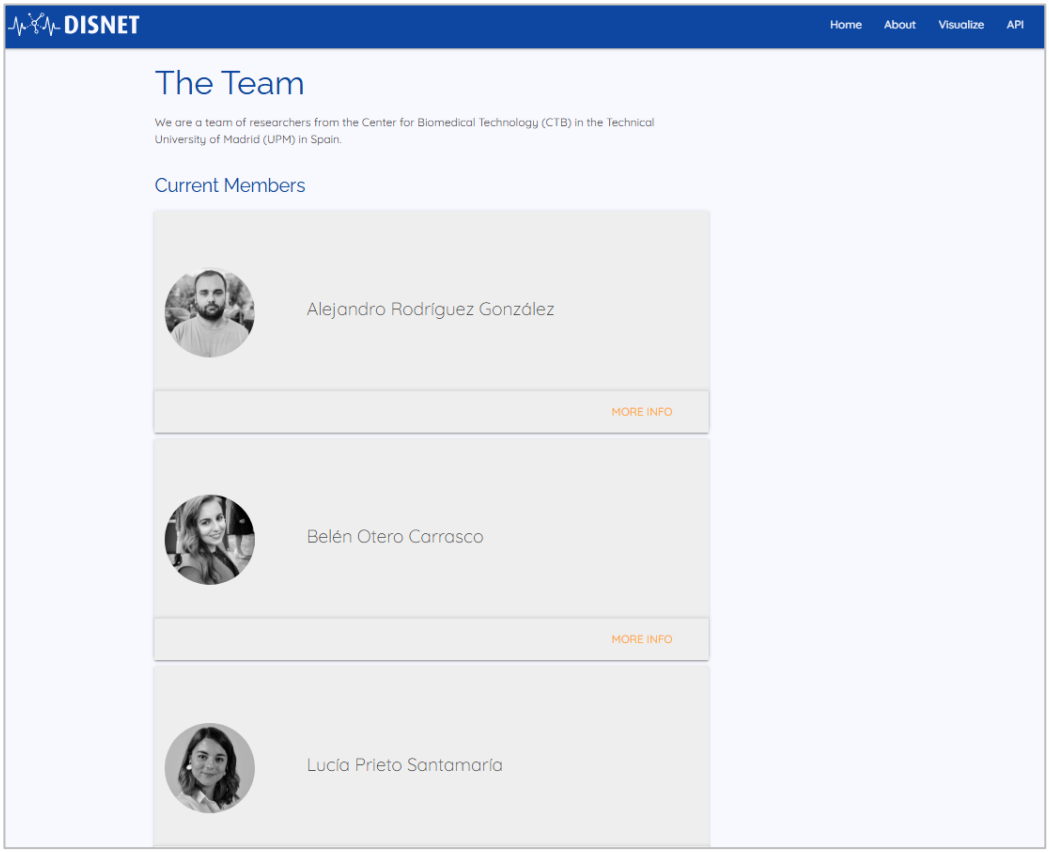


Figure 22: Updated DISNET team members web page.

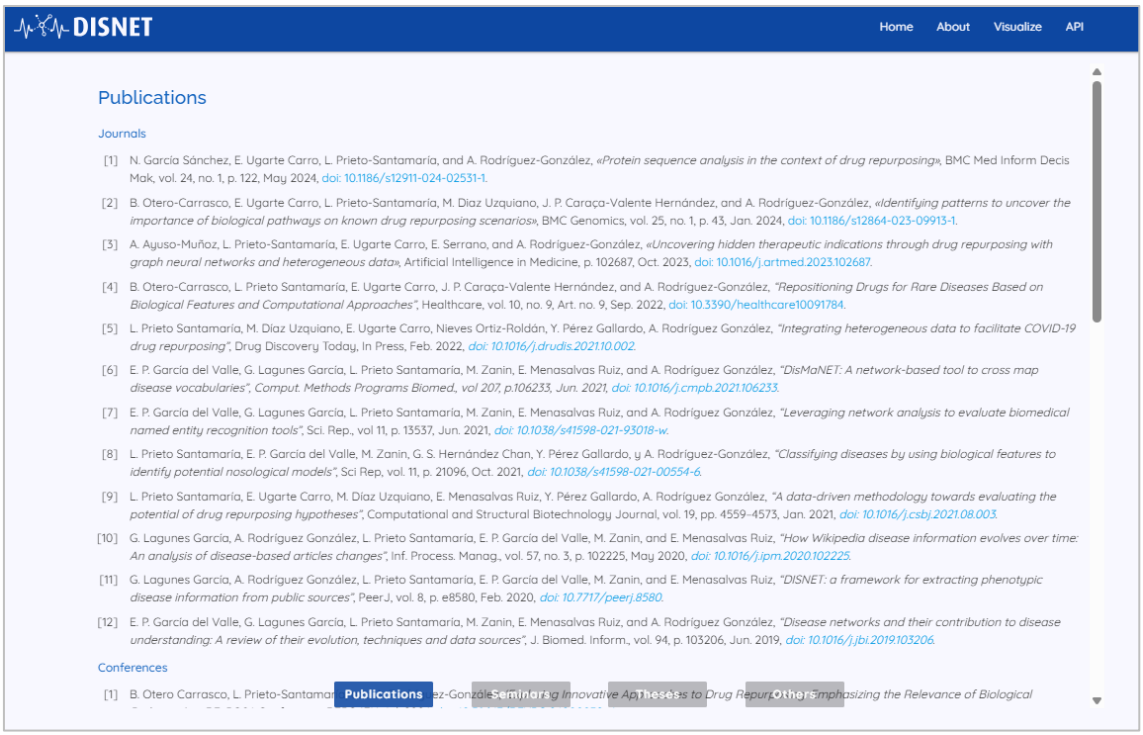


Figure 23: Updated DISNET publications web page.

5.3 Unification of containers

To validate the unification of DISNET services, a *docker compose up* command was executed to start all the DISNET services, as shown in **Figure 24**. This confirms that all defined containers were successfully created, started and deployed.

```
[user@host] $ docker compose up -d
[+] Running 9/9
 ✓ Network disnet_disnet-network                                    Created                                    0.0s
 ✓ Container disnet-diseases_database-mysql84-container             Started                                    0.2s
 ✓ Container disnet-web-container                                   Started                                    0.3s
 ✓ Container disnet-authorization-api-rest-container                Started                                    0.3s
 ✓ Container disnet-semantic_type_validation-api_rest-container     Started                                    0.3s
 ✓ Container disnet-metamap2016v2-container                         Started                                    0.3s
 ✓ Container disnet-get_disease_list_api_rest-container             Started                                    0.4s
 ✓ Container disnet-api_rest-container                              Started                                    0.4s
 ✓ Container disnet-data_storage_and_validation_procedure_api_rest-container  Started                          0.7s
```

Figure 24: Successful deployment of unified DISNET containers.

To evaluate the unification of DISNET services, the DISNET system previously consisted of 16 Docker containers, which included 12 application services, 3 database instances and 1 container for the local instance of MetaMap. Each service required separate configuration and deployment steps. In the updated system, services with complementary functionalities were merged and the number of databases was reduced, which is detailed in **Table 6.**

To evaluate configuration complexity, it is important to note that there is a single *.env* file managing shared environment variables. Before, there were 12 *application.properties* files distributed across multiple repositories. Now, this has been reduced to 7 *application.properties* files centralized in the environment.

Table 6: Comparison of number of containers between the original and updated system.

Component	Original system	Updated system
Application services	12	7
Databases	3	1
MetaMap	1	0
Total	16	8

6 Conclusions and future lines

First, this Master's thesis has successfully brought back the functionality of the Medical Text Extraction Process (MTEP) in the DISNET platform, reintegrating extraction of data from Wikipedia and Mayo Clinic (**Objective 4a**). As part of this restoration, the pipelines were refined to ensure that the extracted information was accurate. Specifically, the improvements have allowed to retrieve data that accurately represents the information presented on the web sources. Moreover, in the Wikipedia pipeline a higher number of diseases could be extracted, and in the Mayo Clinic pipeline, the extracted data was more accurate.

Second, the web interface was restored to connect with the database and was updated to reflect the required changes (**Objective 4b**). As user authentication was no longer essential, it was removed. Additionally, the interface was adjusted to rely on an external API for keeping user and publications information current.

Third, this work has helped into the discover that the original DISNET architecture was overly complex for its use case. The initial microservices system included 16 containers with 12 separate application services, which introduced unnecessary deployment and configuration overhead for a platform primarily maintained by one or two developers. This work reduced the complexity of this system by unifying the three system databases into one (**Objective 4c**) and reducing the number of services to seven (**Objective 4d**). The MTEP was successfully executed after this unification.

Despite the achievement of all development objectives, the work carried out in this Master's thesis also encountered several limitations. First, the PubMed extraction pipeline remains inactive and excluded from the current system due to its large volume of data. This leaves an important source of biomedical literature from being incorporated into DISNET. Second, the system currently lacks a centralized interface for monitoring the extraction processes, which complicates the management of the MTEP. Third, while the phenotypic layer is dynamically updated, the biological and pharmacological layers remain static and disconnected from regular update routines. Finally, the text validation pipeline remains heavily dependent on MetaMap, a tool whose performance can be outpaced by modern NLP techniques [54].

To address these limitations, the following future lines of work are proposed to improve the DISNET platform's capabilities:

- Reintroduce the PubMed extraction pipeline by following a similar approach as for the restoration of the Wikipedia and Mayo Clinic pipelines. This would enable periodic extraction of a more extensive biomedical source, resulting in a higher number of diseases and information integrated into the knowledge base.
- Develop an administrative MTEP dashboard. This dashboard should be implemented into the DISNET web page and be accessible only to administrative users. This dashboard should allow to select the required source(s) for extraction, the execution or not of term validation processes, real time progress monitoring, visualization of the results and options to rerun specific steps.
- Integrate the biologic and pharmacologic layer pipelines. As mentioned in *3.1.4 Development History*, these layers were built through a onetime

automated process. The integration of this processes can ensure all DISNET layers remain updated.

- Incorporate modern NLP models for concept extraction like cTAKES[50] or QuickUMLS [51] [55], [56]. This models could replace or complement MetaMap and could improve concept recognition performance.

[50] *https://github.com/apache/ctakes*
[51] *https://github.com/Georgetown-IR-Lab/QuickUMLS*

7 Bibliography

[1] A.-L. Barabási, N. Gulbahce, and J. Loscalzo, “Network medicine: a network-based approach to human disease,” *Nat Rev Genet*, vol. 12, no. 1, pp. 56–68, Jan. 2011.

[2] C. J. Cremin, S. Dash, and X. Huang, “Big data: Historic advances and emerging trends in biomedical research,” *Curr Res Biotechnol*, vol. 4, pp. 138–151, 2022, doi: https://doi.org/10.1016/j.crbiot.2022.02.004.

[3] A. R. Sonawane, S. T. Weiss, K. Glass, and A. Sharma, “Network Medicine in the Age of Biomedical Big Data,” *Front Genet*, vol. 10, p. 294, Apr. 2019.

[4] K.-I. Goh, M. E. Cusick, D. Valle, B. Childs, M. Vidal, and A.-L. Barabási, “The human disease network,” *Proc Natl Acad Sci U S A*, vol. 104, no. 21, pp. 8685–8690, May 2007.

[5] H. I. Roessler, N. V. A. M. Knoers, M. M. van Haelst, and G. van Haaften, “Drug Repurposing for Rare Diseases,” *Trends Pharmacol Sci*, vol. 42, no. 4, pp. 255–267, Apr. 2021, doi: 10.1016/j.tips.2021.01.003.

[6] J. J. Hernandez *et al.*, “Giving Drugs a Second Chance: Overcoming Regulatory and Financial Hurdles in Repurposing Approved Drugs As Cancer Therapeutics,” *Front Oncol*, vol. 7, p. 273, Nov. 2017.

[7] G. Lagunes-Garc\’\ia, A. Rodr\’\iguez-González, L. Prieto-Santamar\’\ia, E. P. Garc\’\ia Del Valle, M. Zanin, and E. Menasalvas-Ruiz, “DISNET: a framework for extracting phenotypic disease information from public sources,” *PeerJ*, vol. 8, p. e8580, Feb. 2020.

[8] E. P. Garc\’\ia Del Valle, L. Lagunes Garc\’\ia Gerardo and Prieto Santamar\’\ia, E. Zanin Massimiliano and Menasalvas Ruiz, and A. Rodr\’\iguez-González, “Leveraging network analysis to evaluate biomedical named entity recognition tools,” *Sci Rep*, vol. 11, no. 1, p. 13537, Jun. 2021.

[9] E. P. García del Valle, G. Lagunes García, L. Prieto Santamaría, M. Zanin, E. Menasalvas Ruiz, and A. Rodríguez-González, “Disease networks and their contribution to disease understanding: A review of their evolution, techniques and data sources,” *J Biomed Inform*, vol. 94, p. 103206, 2019, doi: https://doi.org/10.1016/j.jbi.2019.103206.

[10] L. Prieto Santamaría *et al.*, “Analysis of New Nosological Models from Disease Similarities using Clustering,” in *2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS)*, 2020, pp. 183–188. doi: 10.1109/CBMS49503.2020.00042.

[11] L. Prieto Santamaría, E. P. García del Valle, M. Zanin, G. S. Hernández Chan, Y. Pérez Gallardo, and A. Rodríguez-González, “Classifying diseases by using biological features to identify potential nosological models,” *Sci Rep*, vol. 11, no. 1, p. 21096, 2021, doi: 10.1038/s41598-021-00554-6.

[12] G. Lagunes García, L. Prieto Santamaria, E. P. del Valle, M. Zanin, E. Menasalvas Ruiz, and A. Rodríguez González, “Wikipedia Disease Articles: An Analysis of their Content and Evolution,” in *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, 2019, pp. 664–671. doi: 10.1109/CBMS.2019.00136.

[13] G. Lagunes-García, A. Rodríguez-González, L. Prieto-Santamaría, E. P. García del Valle, M. Zanin, and E. Menasalvas-Ruiz, “How Wikipedia

disease information evolve over time? An analysis of disease-based articles changes," *Inf Process Manag*, vol. 57, no. 3, p. 102225, 2020, doi: https://doi.org/10.1016/j.ipm.2020.102225.

[14] E. P. del Valle, G. Lagunes García, L. Prieto Santamaría, M. Zanin, E. Menasalvas Ruiz, and A. Rodríguez González, "Evaluating Wikipedia as a Source of Information for Disease Understanding," in *2018 IEEE 31st International Symposium on Computer-Based Medical Systems (CBMS)*, 2018, pp. 399–404. doi: 10.1109/CBMS.2018.00076.

[15] L. Prieto Santamaría, E. Ugarte Carro, M. Díaz Uzquiano, E. Menasalvas Ruiz, Y. Pérez Gallardo, and A. Rodríguez-González, "A data-driven methodology towards evaluating the potential of drug repurposing hypotheses," *Comput Struct Biotechnol J*, vol. 19, pp. 4559–4573, 2021, doi: https://doi.org/10.1016/j.csbj.2021.08.003.

[16] L. Prieto Santamaría, M. Díaz Uzquiano, E. Ugarte Carro, N. Ortiz-Roldán, Y. Pérez Gallardo, and A. Rodríguez-González, "Integrating heterogeneous data to facilitate COVID-19 drug repurposing," *Drug Discov Today*, vol. 27, no. 2, pp. 558–566, 2022, doi: https://doi.org/10.1016/j.drudis.2021.10.002.

[17] M. M. Tercero, L. Prieto-Santamaría, and A. Rodríguez-González, "Exploring Drug Repurposing Opportunities for Schizophrenia: A Network Medicine Approach," in *2024 IEEE 37th International Symposium on Computer-Based Medical Systems (CBMS)*, 2024, pp. 106–111. doi: 10.1109/CBMS61543.2024.00026.

[18] B. Otero-Carrasco, L. Prieto Santamaría, E. Ugarte Carro, J. P. Caraça-Valente Hernández, and A. Rodríguez-González, "Repositioning Drugs for Rare Diseases Based on Biological Features and Computational Approaches," *Healthcare*, vol. 10, no. 9, 2022, doi: 10.3390/healthcare10091784.

[19] B. Otero-Carrasco, L. Prieto Santamaría, E. Ugarte Carro, J. P. Caraça-Valente Hernández, and A. Rodríguez-González, "A Computational Drug Repositioning Method for Rare Diseases," in *Bio-inspired Systems and Applications: from Robotics to Ambient Intelligence*, J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, and H. Adeli, Eds., Cham: Springer International Publishing, 2022, pp. 551–561.

[20] B. Otero-Carrasco *et al.*, "Orphan Drugs and Rare Diseases: Unveiling Biological Patterns through Drug Repurposing," in *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*, 2023, pp. 185–191. doi: 10.1109/CBMS58004.2023.00214.

[21] A. A. Muñoz *et al.*, "REDIRECTION: Generating drug repurposing hypotheses using link prediction with DISNET data," in *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, 2022, pp. 7–12. doi: 10.1109/CBMS55023.2022.00009.

[22] A. Ayuso-Muñoz, L. Prieto-Santamaría, E. Ugarte-Carro, E. Serrano, and A. Rodríguez-González, "Uncovering hidden therapeutic indications through drug repurposing with graph neural networks and heterogeneous data," *Artif Intell Med*, vol. 145, p. 102687, 2023, doi: https://doi.org/10.1016/j.artmed.2023.102687.

[23] A. Ayuso-Muñoz, L. Prieto-Santamaría, A. Álverez-Pérez, B. Otero-Carrasco, E. Serrano, and A. Rodríguez-González, "Enhancing Drug

Repurposing on Graphs by Integrating Drug Molecular Structure as Feature," in *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*, 2023, pp. 192–197. doi: 10.1109/CBMS58004.2023.00215.

[24] R. Artiñano-Muñoz, L. Prieto-Santamaría, A. Pérez-Pérez, and A. Rodríguez-González, "DRAGON: Drug Repurposing via Graph Neural Networks with Drug and Protein Embeddings as Features," in *2024 IEEE 37th International Symposium on Computer-Based Medical Systems (CBMS)*, 2024, pp. 170–175. doi: 10.1109/CBMS61543.2024.00036.

[25] B. Otero-Carrasco, E. Ugarte Carro, L. Prieto-Santamaría, M. Diaz Uzquiano, J. P. Caraça-Valente Hernández, and A. Rodríguez-González, "Identifying patterns to uncover the importance of biological pathways on known drug repurposing scenarios," *BMC Genomics*, vol. 25, no. 1, p. 43, 2024, doi: 10.1186/s12864-023-09913-1.

[26] A. Álvarez-Pérez, L. Prieto-Santamaría, E. Ugarte-Carro, B. Otero-Carrasco, A. Ayuso-Muñoz, and A. Rodríguez-González, "Exploring disease-drug pairs in Clinical Trials information for personalized drug repurposing," in *2023 IEEE 36th International Symposium on Computer-Based Medical Systems (CBMS)*, 2023, pp. 179–184. doi: 10.1109/CBMS58004.2023.00213.

[27] B. O. Carrasco, A. P. Pérez, E. M. Ruiz, J. P. C.-V. Hernández, L. P. Santamaría, and A. Rodríguez-González, "Drug repositioning with gender perspective focused on Adverse Drug Reactions," in *2022 IEEE 35th International Symposium on Computer-Based Medical Systems (CBMS)*, 2022, pp. 435–440. doi: 10.1109/CBMS55023.2022.00084.

[28] E. P. García del Valle, G. Lagunes García, L. Prieto Santamaría, M. Zanin, E. Menasalvas Ruiz, and A. Rodríguez-González, "DisMaNET: A network-based tool to cross map disease vocabularies," *Comput Methods Programs Biomed*, vol. 207, p. 106233, 2021, doi: https://doi.org/10.1016/j.cmpb.2021.106233.

[29] E. P. del Valle, G. Lagunes García, E. Menasalvas Ruiz, L. Prieto Santamaría, M. Zanin, and A. Rodríguez-González, "Completing Missing MeSH Code Mappings in UMLS Through Alternative Expert-Curated Sources," in *2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS)*, 2019, pp. 174–179. doi: 10.1109/CBMS.2019.00044.

[30] A. and S. L. P. and P.-V. M. and B.-O. C. and R.-G. A. Pérez Andrea Álvarez and Iglesias-Molina, "EBOCA: Evidences for BiOmedical Concepts Association Ontology," in *Knowledge Engineering and Knowledge Management*, L. and K. O. and T. N. and E. F. J. Corcho Oscar and Hollink, Ed., Cham: Springer International Publishing, 2022, pp. 152–166.

[31] S. Sadegh *et al.*, "Network medicine for disease module identification and drug repurposing with the NeDRex platform," *Nat Commun*, vol. 12, no. 1, p. 6848, 2021, doi: 10.1038/s41467-021-27138-2.

[32] J. Piñero, J. Saüch, F. Sanz, and L. I. Furlong, "The DisGeNET cytoscape app: Exploring and visualizing disease genomics data," *Comput Struct Biotechnol J*, vol. 19, pp. 2960–2967, 2021, doi: https://doi.org/10.1016/j.csbj.2021.05.015.

[33] N. Queralt-Rosinach, J. Piñero, À. Bravo, F. Sanz, and L. I. Furlong, "DisGeNET-RDF: harnessing the innovative power of the Semantic Web to explore the genetic basis of diseases," *Bioinformatics*, vol. 32, no. 14, pp. 2236–2238, Apr. 2016.

[34] S. Sadegh *et al.*, "Exploring the SARS-CoV-2 virus-host-drug interactome for drug repurposing," *Nat Commun*, vol. 11, no. 1, p. 3518, 2020, doi: 10.1038/s41467-020-17189-2.

[35] D. S. Himmelstein *et al.*, "Systematic integration of biomedical knowledge prioritizes drugs for repurposing," *Elife*, vol. 6, p. e26726, Sep. 2017, doi: 10.7554/eLife.26726.

[36] A. Buniello *et al.*, "Open Targets Platform: facilitating therapeutic hypotheses building in drug discovery," *Nucleic Acids Res*, vol. 53, no. D1, pp. D1467–D1475, Jan. 2025.

[37] G. Lagunes García, "Caracterización de enfermedades basada en su información fenotípica recuperada mediante la extracción de conocimiento biomédico de fuentes de información públicas," 2020. [Online]. Available: http://dx.doi.org/10.20868/UPM.thesis.64801

[38] E. García del Valle, "Creation of Networks for the Analysis of Disease Similarities," Sep. 2022. doi: 10.20868/UPM.thesis.71764.

[39] L. Prieto Santamaría, "Extracción, cálculo de similitudes y análisis de características biológicas para la creación de redes humanas complejas," Madrid, Jun. 2018. [Online]. Available: https://oa.upm.es/52568/

[40] P. Soto García, "Study and integration of new biological features data from public sources in the context of human complex disease networks," Máster Universitario en Biología Computacional, UPM, 2020.

[41] N. García Sánchez, "Biological sequence analysis in the context of drug repurposing and human disease complex networks," Grado en Biotecnología, UPM, 2022.

[42] Á. Gutiérrez Castanedo, "Entendimiento de casos exitosos de reposicionamiento de fármacos mediante análisis de secuencias de proteínas," Madrid, Jun. 2023. [Online]. Available: https://oa.upm.es/76408/

[43] S. Jaramillo Cárdenas, "Analysis of public data sources and drugs information extraction towards the elaboration of human complex disease networks," Máster Universitario en Biología Computacional, UPM, 2020.

[44] E. Ugarte Carro, "Drug repurposing approaches and validation through DISNET data," Madrid, Jun. 2021. [Online]. Available: https://oa.upm.es/69193/

[45] A. Ayuso Muñoz, "Análisis y desarrollo de modelos de predicción de links en redes mediante técnicas de Deep Learning," Madrid, España, May 2022. [Online]. Available: https://oa.upm.es/71002/

[46] A. Ayuso Muñoz, "Graph deep learning for drug repurposing," Jun. 2023. [Online]. Available: https://oa.upm.es/75261/

[47] A. Alonso Noguerales, "Desarrollo de la API (wrapper) para el acceso mediante código Java a servicios web de DISNET," Madrid, España, May 2022. [Online]. Available: https://oa.upm.es/69822/

[48] D. Navarro Riaño, “Desarrollo de API (wrapper) para acceso mediante código Python a servicios web de DISNET,” Grado en Ingeniería Informática, UPM, 2022.

[49] I. Montero Callejas, “Desarrollo de procesos para el análisis y visualización de datos fenotípicos y biológicos en el contexto de las redes de enfermedades humanas,” Grado en Ingeniería Biomédica, UPM, 2021.

[50] B. Otero Carrasco, “Discovery and Analysis of Biological Patterns to Enhance Drug Repurposing Strategies,” Nov. 2024. doi: 10.20868/UPM.thesis.83874.

[51] A. Á. Pérez, A. Iglesias-Molina, L. P. Santamaría, M. Poveda-Villalón, C. Badenes-Olmedo, and A. Rodríguez-González, “EBOCA: Evidences for BiOmedical Concepts Association Ontology,” in *Knowledge Engineering and Knowledge Management*, O. Corcho, L. Hollink, O. Kutz, N. Troquard, and F. J. Ekaputra, Eds., Cham: Springer International Publishing, 2022, pp. 152–166.

[52] J. Gosling and H. McGilton, *The Java Language Environment: A White Paper*. Sun Microsystems, 1996. Accessed: May 28, 2025. [Online]. Available: https://www.oracle.com/java/technologies/language-environment.html

[53] A. R. Aronson, “Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program,” *Proc AMIA Symp*, pp. 17–21, 2001.

[54] L. Bai, M. D. Mulvenna, Z. Wang, and R. Bond, “Clinical Entity Extraction: Comparison between MetaMap, cTAKES, CLAMP and Amazon Comprehend Medical,” in *2021 32nd Irish Signals and Systems Conference (ISSC)*, 2021, pp. 1–6. doi: 10.1109/ISSC52156.2021.9467856.

[55] G. K. Savova *et al.*, “Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications,” *J Am Med Inform Assoc*, vol. 17, no. 5, pp. 507–513, Sep. 2010.

[56] L. Soldaini, “QuickUMLS: a Fast,” *Unsupervised Approach for Medical Concept Extraction*, 2016.

8 Annexes

8.1 Annex A: Table Diagrams of the DISNET Databases

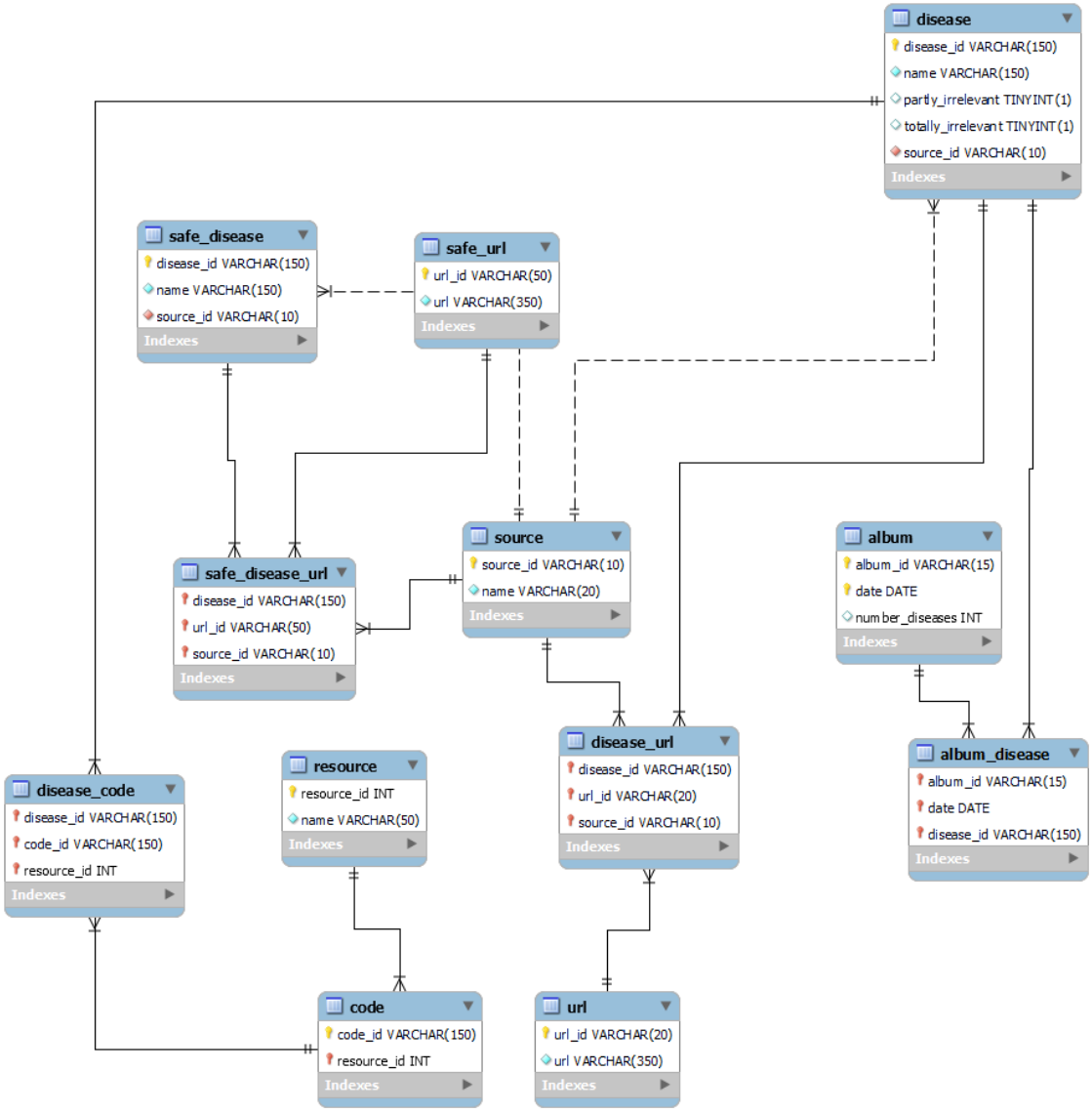


Figure A1: Table Diagram of the addb schema used in the Disease List Available Database.

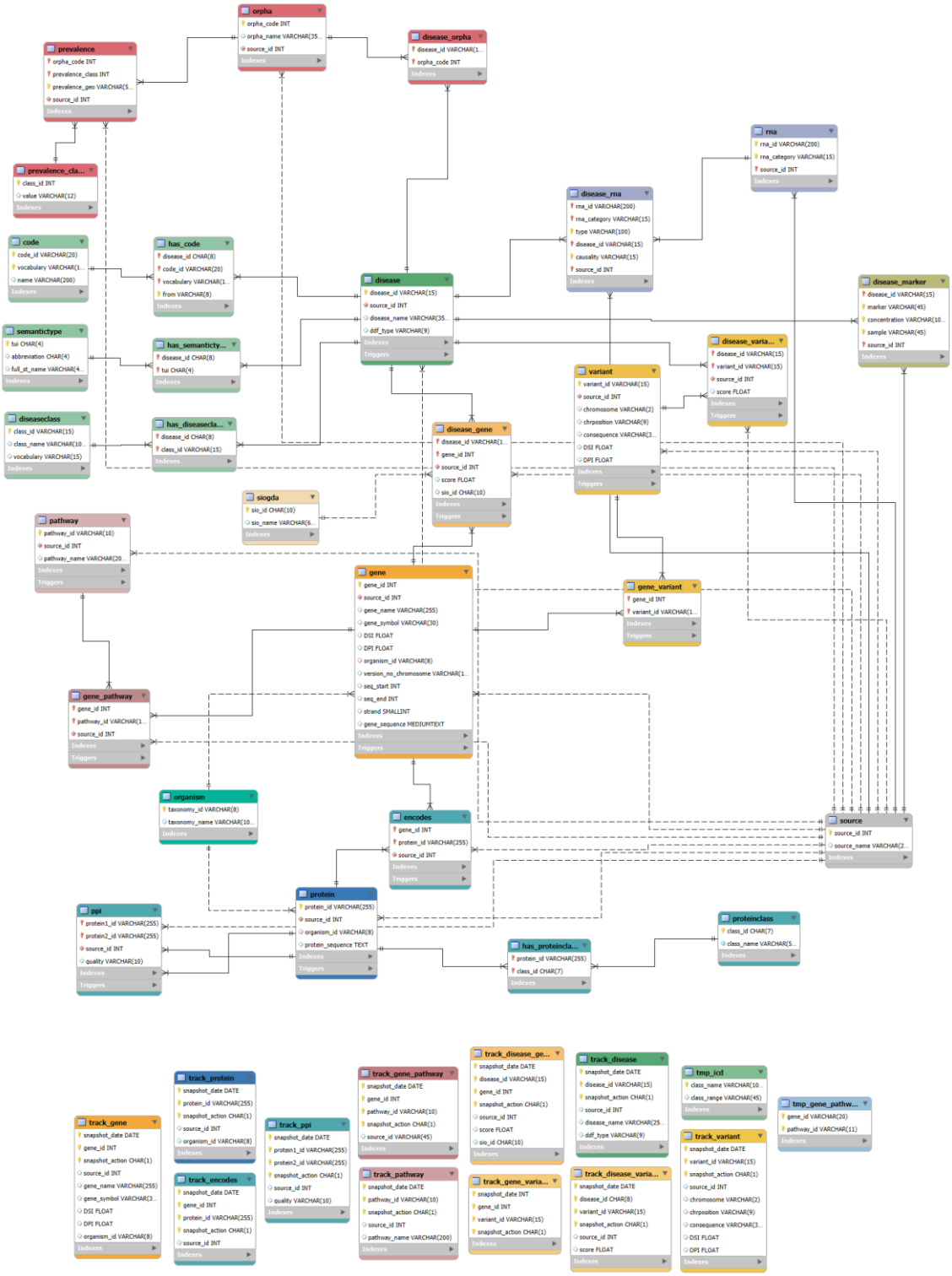


Figure A2: Table Diagram of the Disnet Biolayer schema used in the Diseases Database.

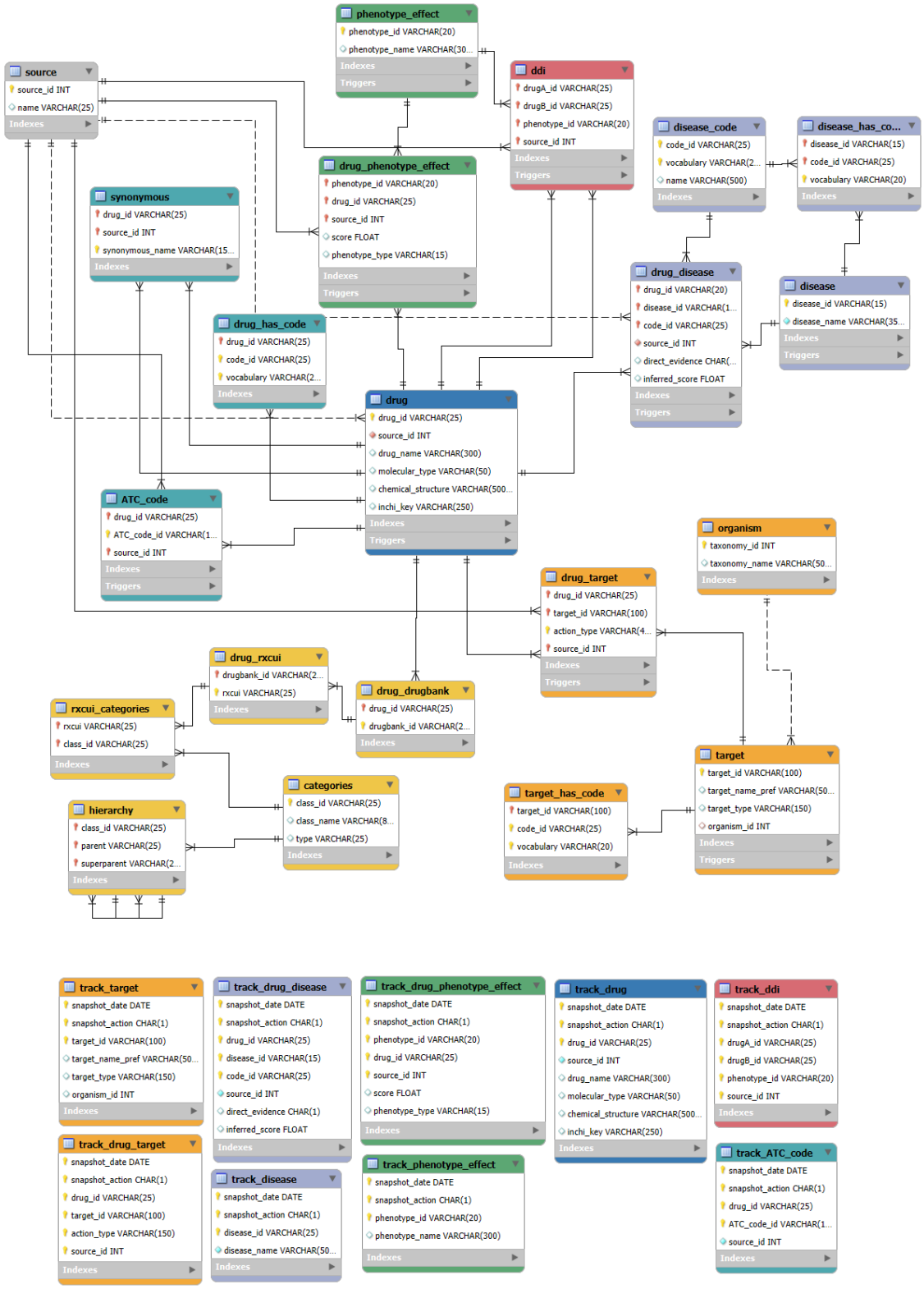


Figure A3: Table Diagram of the Disnet Drugslayer schema used in the Diseases Database.

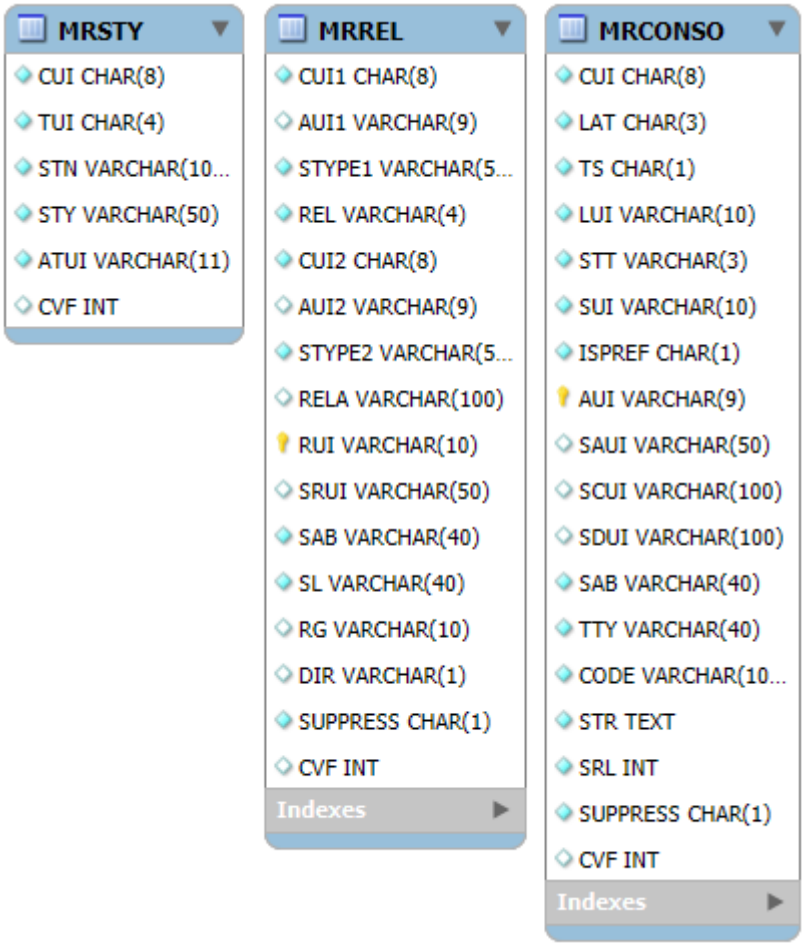


Figure A4: Table Diagram of the umls schema used in the Diseases Database.

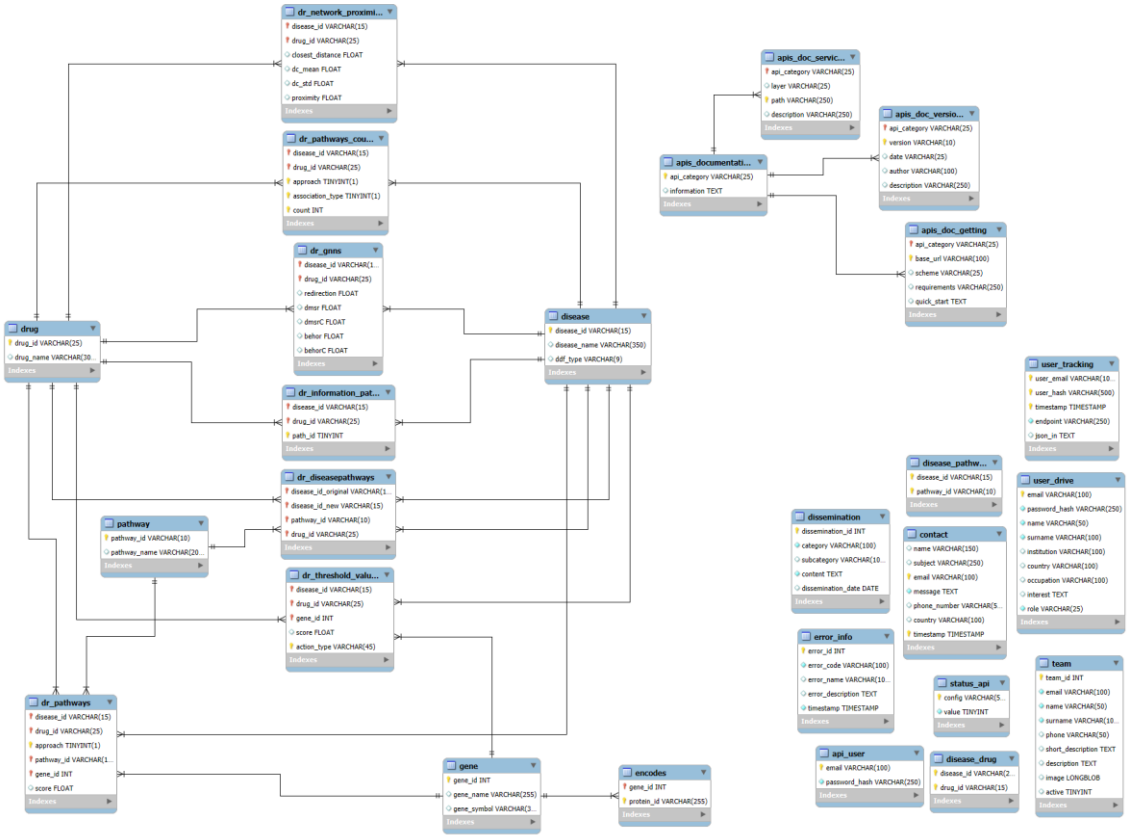


Figure A5: Table Diagram of the dr schema used in the Diseases Database.

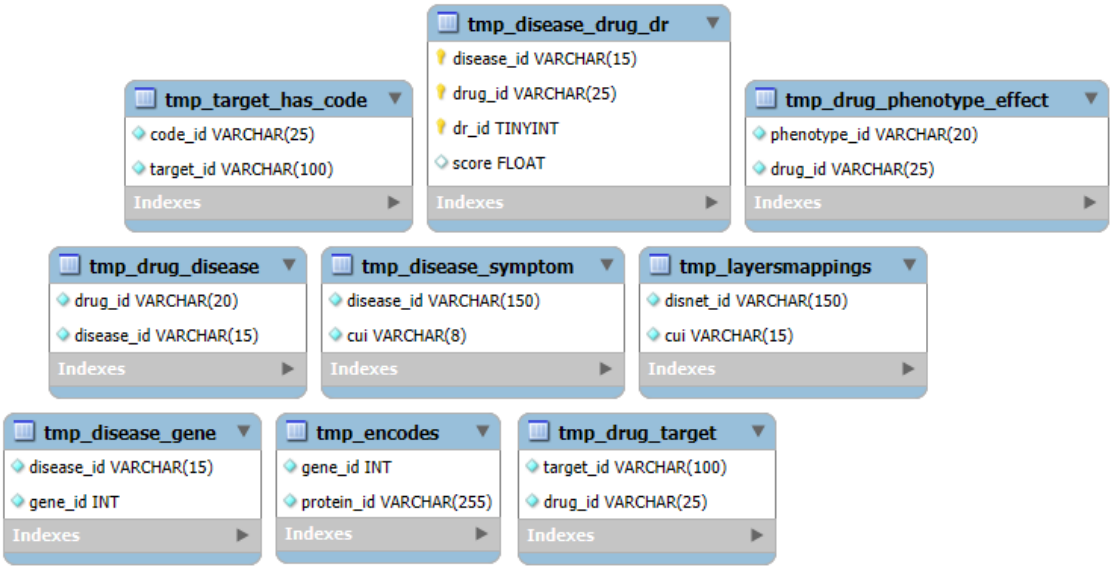


Figure A6: Table Diagram of the repu_temp schema used in the Diseases Database.

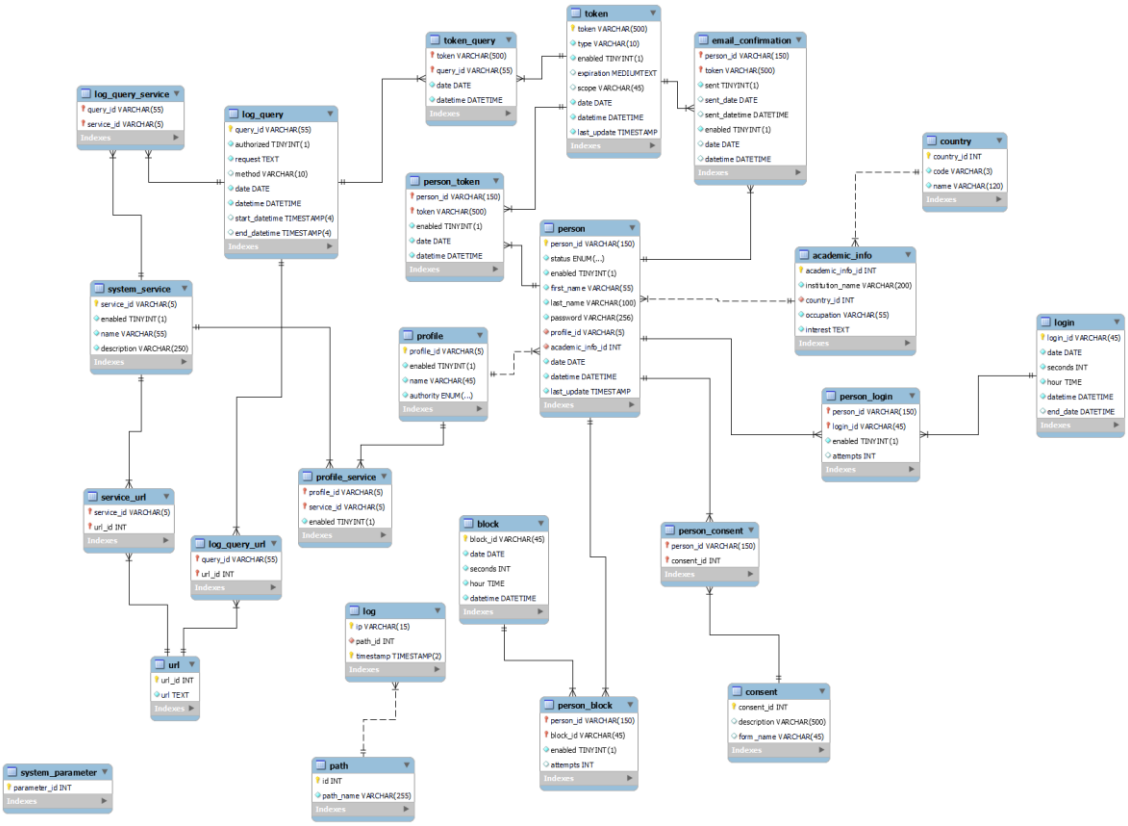


Figure A7: Table Diagram of the disnetdb schema used in the Users Database.

8.2 Annex B: Class diagrams for text extraction

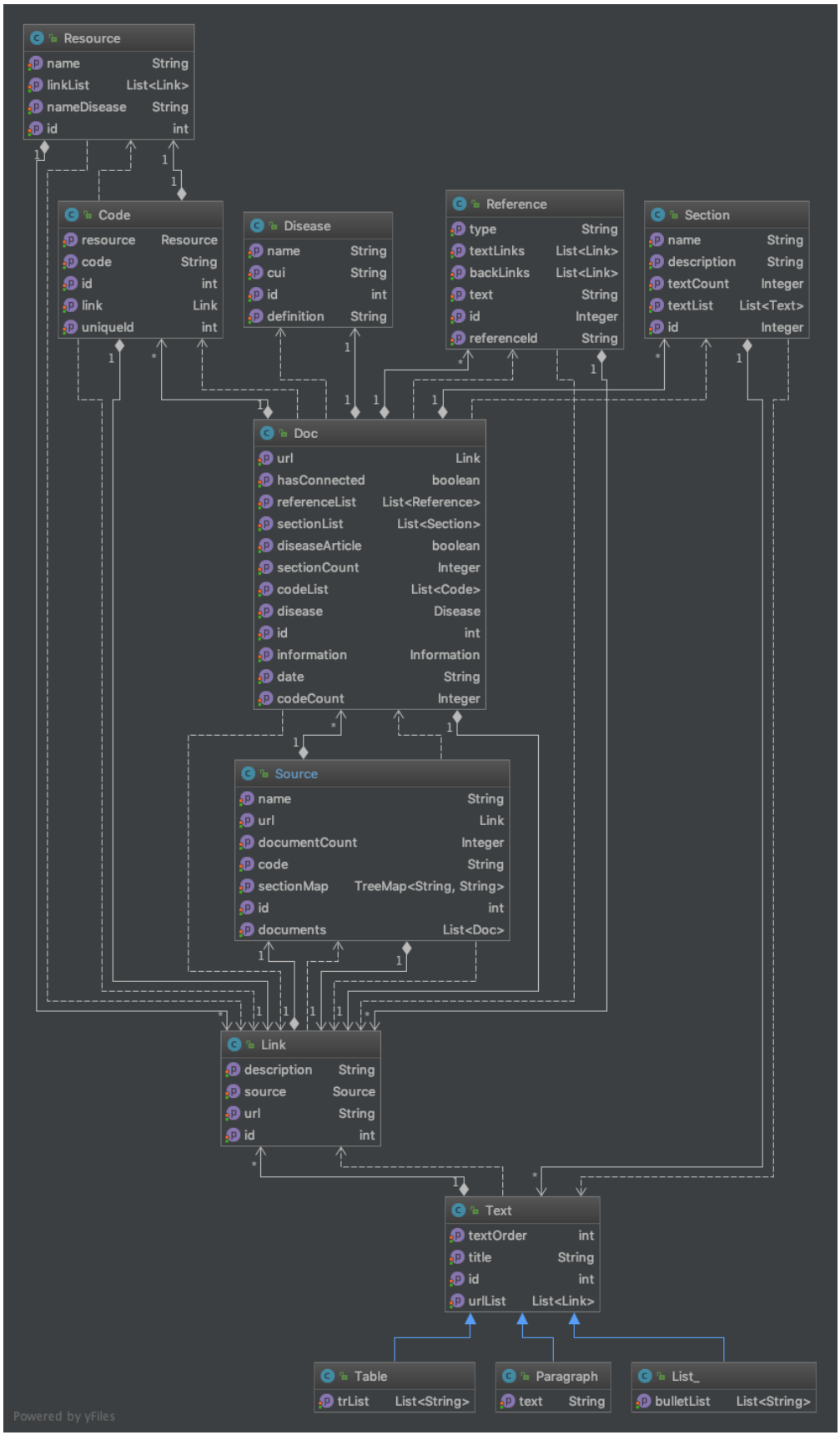


Figure B1: Class diagram for Wikipedia Text Extraction.

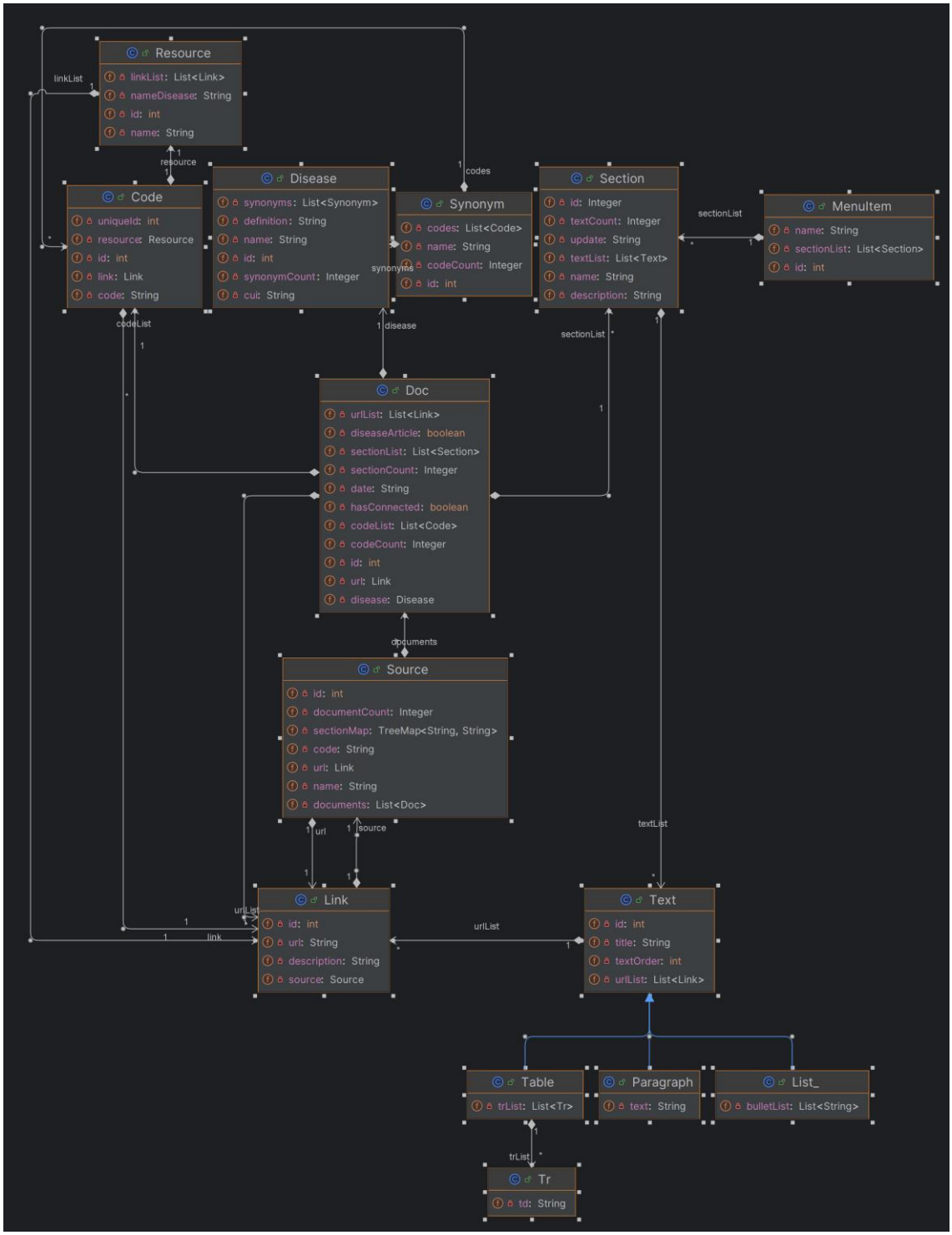


Figure B2: Class diagram for Mayo Clinic Text Extraction.

8.3 Annex C: Original DISNET website

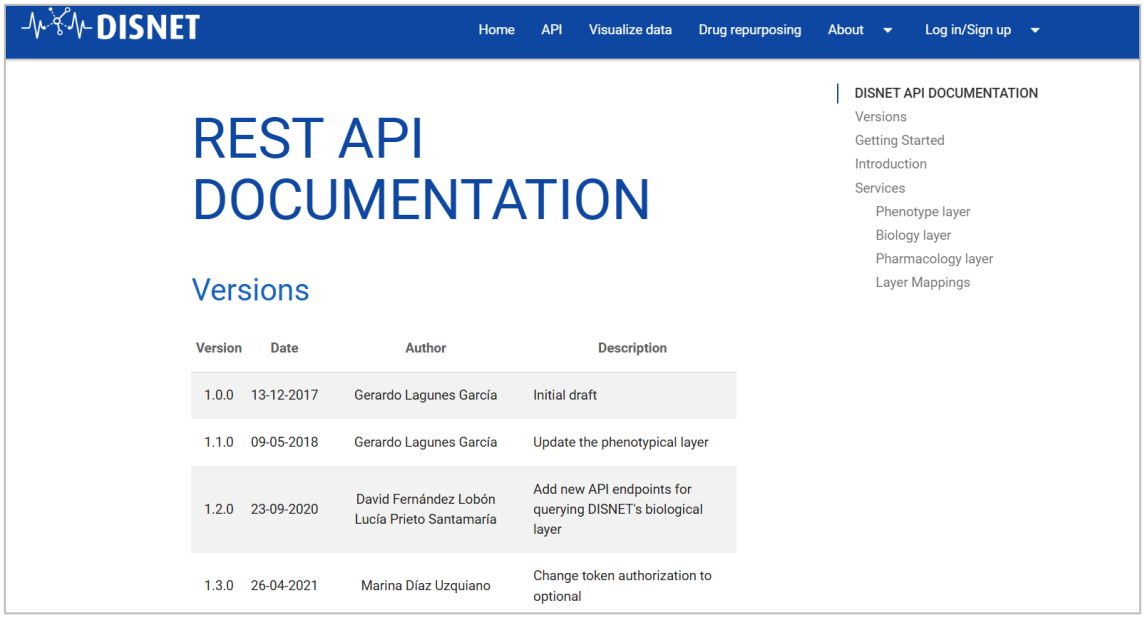


Figure C1: Webpage for displaying DISNET API documentation.

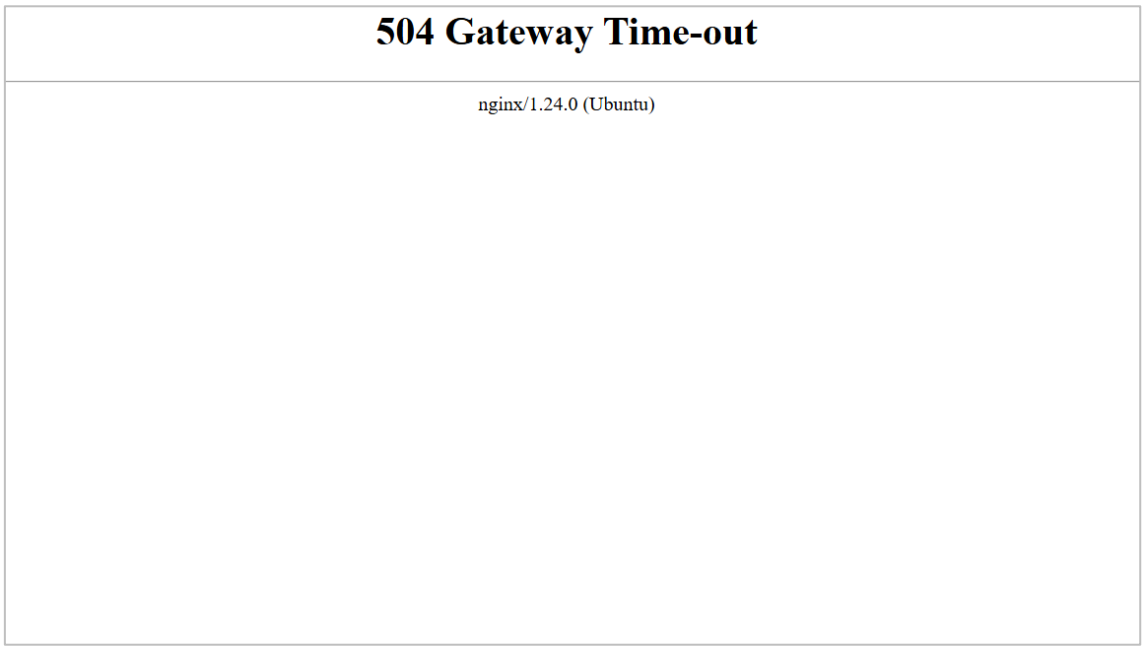


Figure C2: Webpage for displaying visualization data, as can be seen it is unavailable. This is due to failures when connecting to the databases.

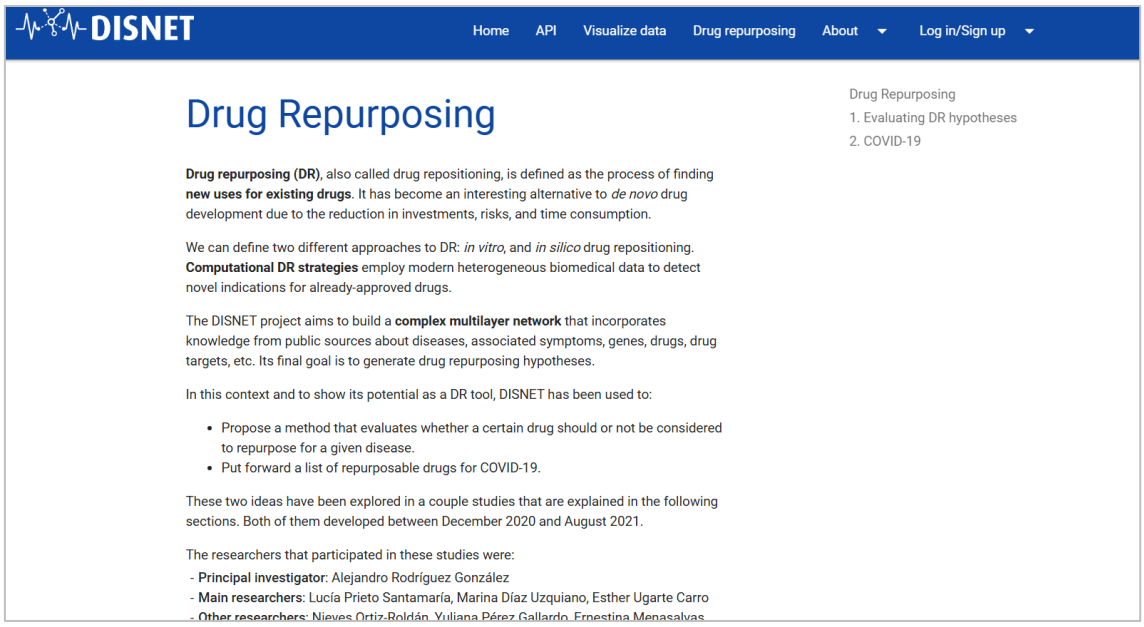


Figure C3: Screenshot of the Drug repurposing section of the DISNET's original website.

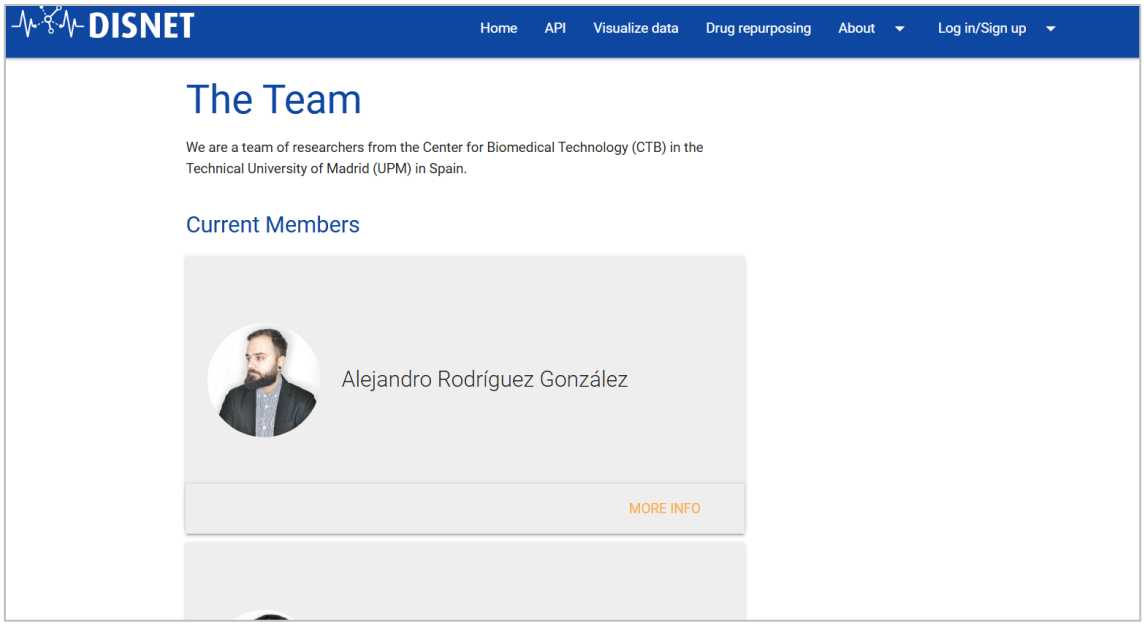


Figure C4: Screenshot of the About > Team section of the DISNET's original website.

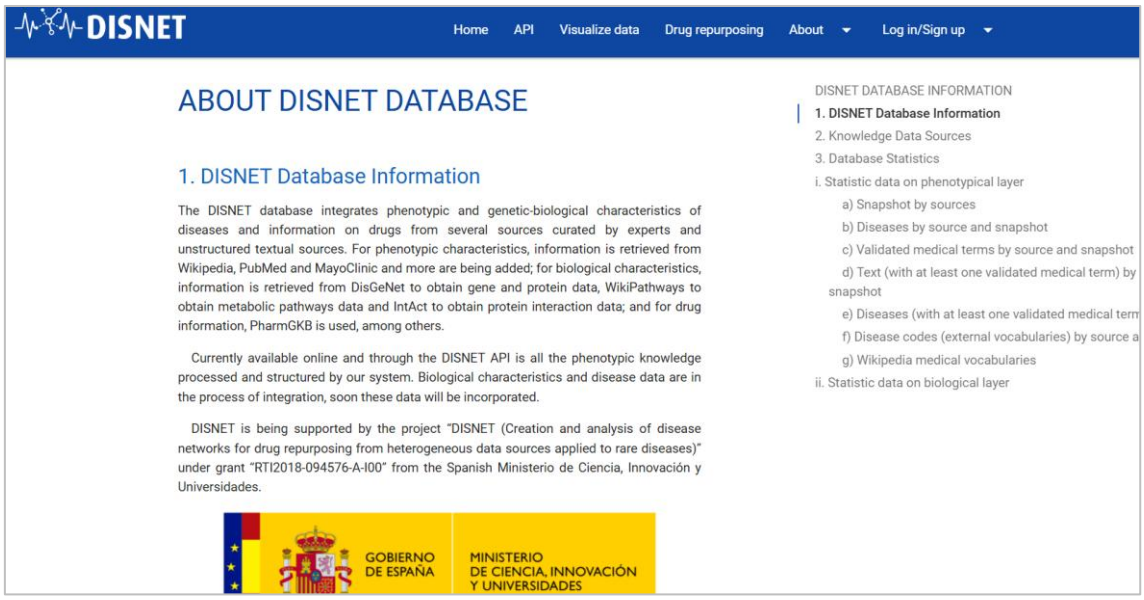


Figure C5: Screenshot of the section About > Database of the DISNET's original website.

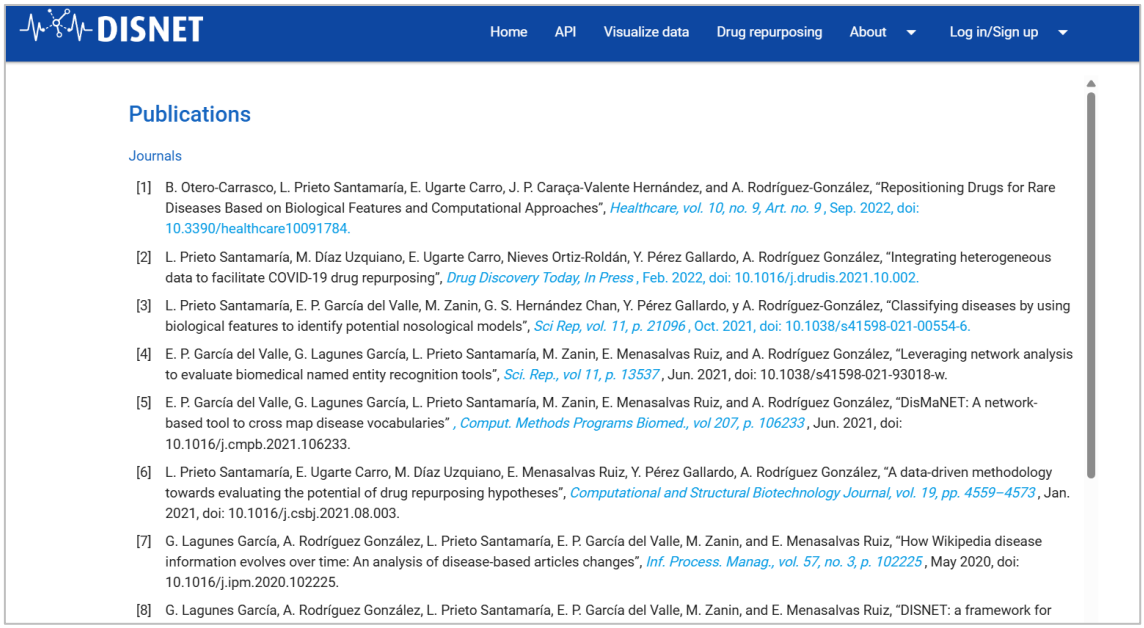


Figure C6: Screenshot of the section About > Resources of the DISNET's original website.

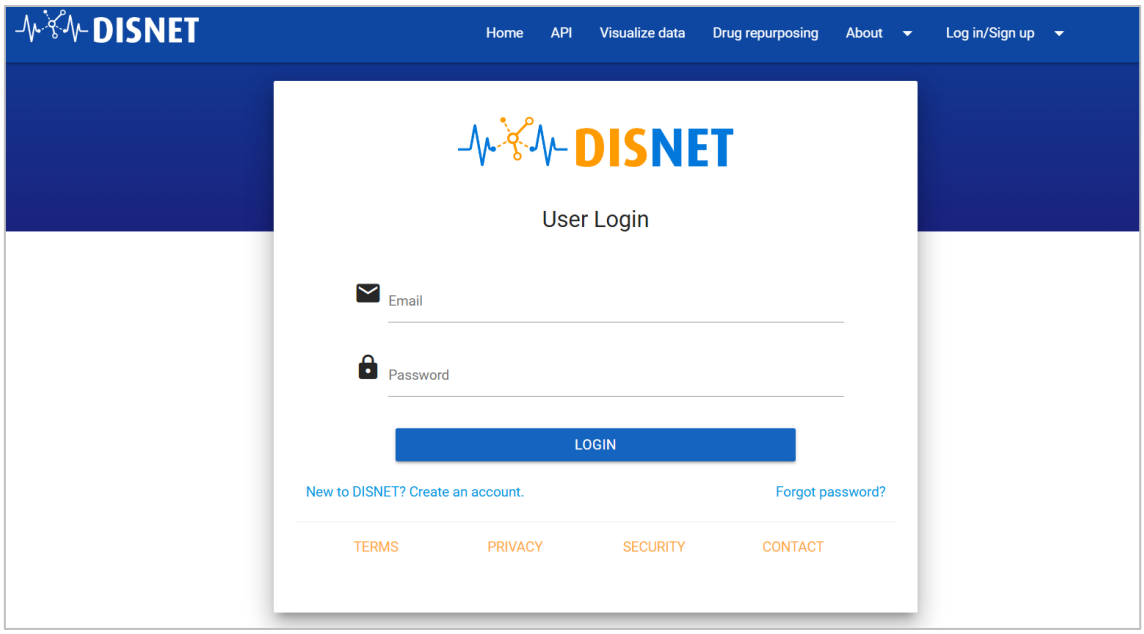


Figure C7: Screenshot of the Log in section of the DISNET's original website.

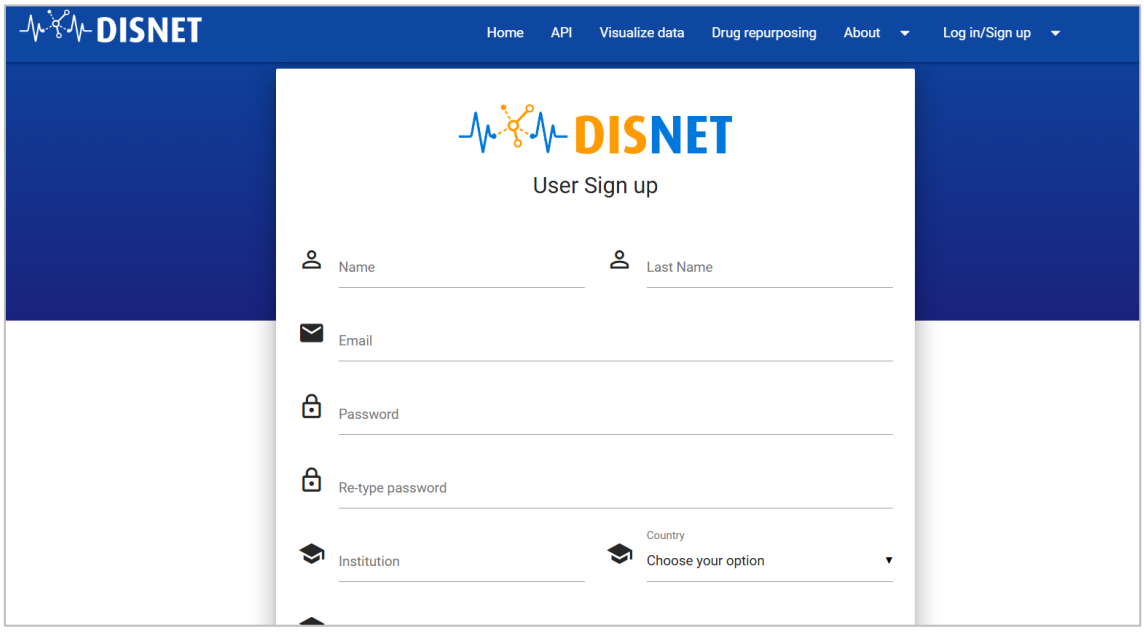


Figure C8: Screenshot of the Sign up section of the DISNET's original website.

8.4 Annex D: SonarQube Metrics

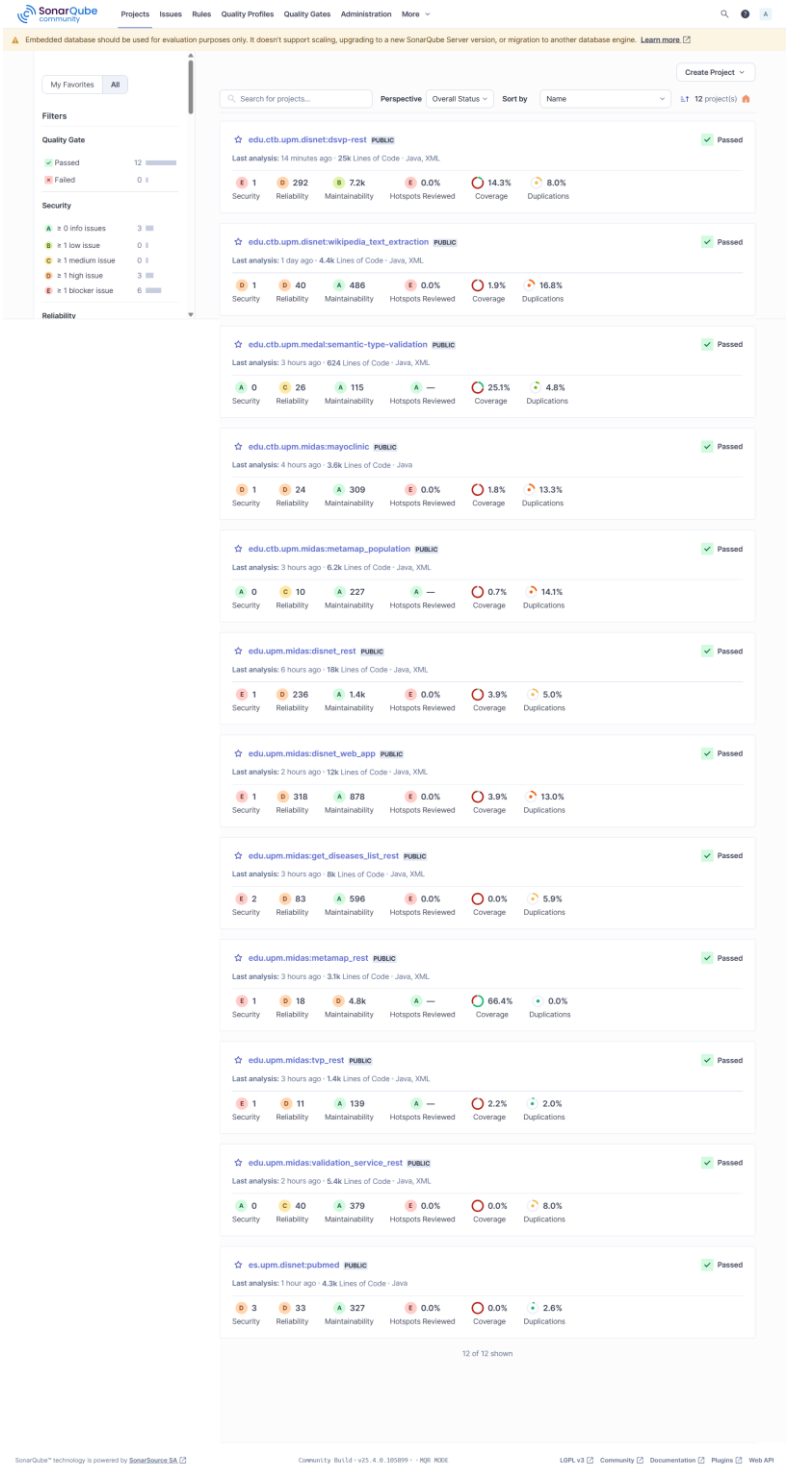


Figure D1: Snapshot of SonarQube metrics result for in-project analysis.

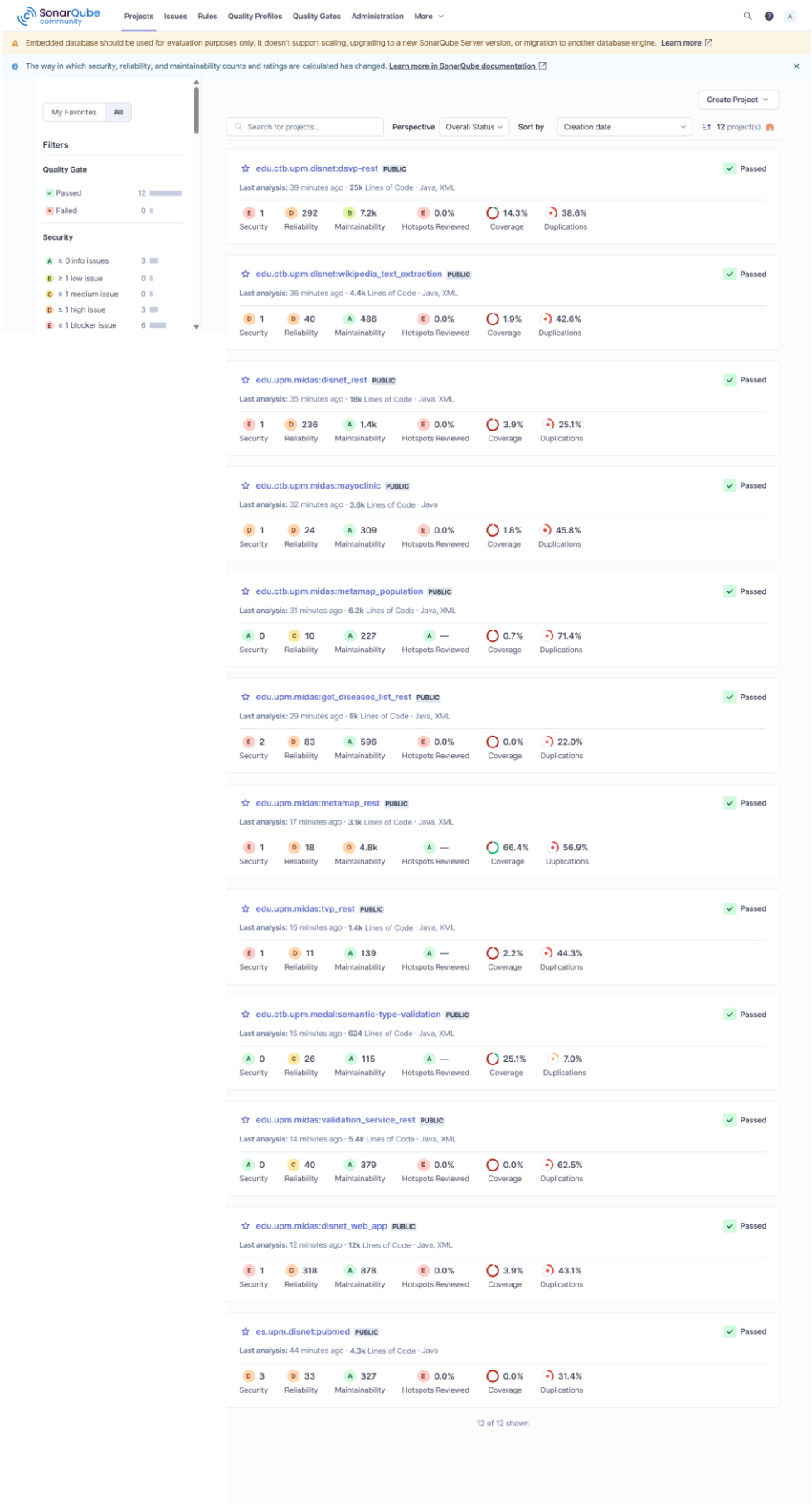


Figure D2: Snapshot of SonarQube metrics result for cross-project analysis.