

***A Web Application for the Creation and Management of Ontology-based Entity Descriptions***

***Pantouvakis Christoforos  
CSD 3313***

**Επίβλεψη**: Παύλος Φαφαλιός

**Επόπτης** **καθηγητής**: Γιάννης Τζίτζικας

**Μέλος** **της** **Επιτροπής** **Παρακολούθησης/Αξιολόγησης**: Δημήτρης Πλεξουσάκης

Contents

[1. Introduction 5](#_Toc180059004)

This section introduces the purpose and main objectives of the application developed in this thesis. It explains how the application simplifies data management through the use of ontologies and the RDF data model, enabling users to create and organize knowledge graphs. The section also provides an overview of the key features, including entity documentation, configuration options, and knowledge graph extraction, while highlighting the user-friendly interface that caters to both technical and non-technical users……………………….…………………………………………………………………………………………………………..5

[1.1 Purpose of the Application 5](#_Toc180059005)

[1.2 Overview of Functions 5](#_Toc180059006)

[2. Work Background 6](#_Toc180059007)

This section provides an overview of key concepts and technologies that underpin this research. Understanding these foundational elements is essential for grasping the significance and potential impact of the work presented in this thesis. The focus will be on Data Representation Ontologies, the RDF Data Model, Knowledge Graph and SPARQL ......6

[2.1 Data Representation Ontologies 6](#_Toc180059008)

[2.2 The RDF Data Model 6](#_Toc180059009)

[2.3 Knowledge Graphs 7](#_Toc180059010)

[2.4 SPARQL 7](#_Toc180059011)

[3. Application Description 9](#_Toc180059012)

[3.1 Architecture and Technologies 9](#_Toc180059013)

[The project is built upon a modern web architecture that facilitates the dynamic generation and manipulation of data tables representing various entities and their relationships within a knowledge graph. The frontend is developed using React, allowing for a responsive and interactive user interface. The backend, implemented in Node.js, handles data processing and storage, utilizing Express as the web server framework to manage API requests efficiently. 9](#_Toc180059014)

[The knowledge graph serves as the core component of the application, enabling the organization of entities and their interconnections through an ontology-based structure. This architecture allows for seamless querying and visualization of data relationships, enhancing the overall user experience. 9](#_Toc180059015)

[3.2 Functionality 9](#_Toc180059016)

[The application supports various functionalities aimed at managing and visualizing entity data within the knowledge graph framework. Key features include: 9](#_Toc180059017)

[3.2.1 Configuration 9](#_Toc180059018)

[Users can customize their data inputs and entity definitions according to their specific needs, ensuring flexibility and scalability. The system allows users to create and manage their projects through a General Properties configuration interface, where they can define key aspects of their project. Within this setup, users can create necessary entities, tailoring each entity with specific "elements" that suit their requirements. These elements can include attributes, properties, and relationships between entities. Additionally, the platform supports the creation of custom vocabularies, allowing users to define elements with predefined insertions, ensuring consistency and standardization across their project. This flexibility empowers users to build robust data models that can adapt to different use cases, promoting seamless scalability as their project evolves. 9](#_Toc180059019)

[3.2.2 Documentation (Data Input) 9](#_Toc180059020)

[The system provides comprehensive documentation that guides users in uploading and managing their data effectively. This includes clear instructions on how to insert new entities (i.e., new data) into the system, as well as how to format and validate that data to ensure accuracy. Additionally, users can easily access and review existing entries, with the option to edit or update them if necessary. This ensures that the data remains up-to-date and aligned with the user’s evolving needs, all while maintaining data integrity and consistency throughout the system. 9](#_Toc180059021)

[3.2.3 Browsing 10](#_Toc180059022)

[3.2.4 Knowledge Graph Extraction 10](#_Toc180059023)

[3.3 Database 11](#_Toc180059024)

[The underlying database is written in MySQL and managed using MySQL Workbench, providing a robust and scalable solution for storing and managing data. It is designed to efficiently handle entity data and their relationships, ensuring fast access and retrieval of information. The database is divided into two distinct schemas to separate core application data from user-defined content. 11](#_Toc180059025)

[The first schema contains tables essential for the functioning of the program, including configuration settings, and data types that are used throughout the system. These tables also manage key aspects of the connection between entities, ensuring that the relationships and interactions within the knowledge graph are maintained. While these tables are critical for the system’s operation, they are not directly accessible to users, protecting the integrity and security of the application. This schema also includes all the entities defined in the system, but their structure and data are controlled at a programmatic level. By keeping these tables isolated, the application can maintain optimal performance, avoid conflicts, and prevent accidental modifications that could affect system stability and the correctness of entity relationships. 11](#_Toc180059026)

[The second schema is dedicated to vocabulary management, where users can define and organize their custom vocabularies. This schema supports tables that store elements such as entity attributes, relationships, and predefined insertions. These vocabulary tables allow users to flexibly create and extend their data models based on their specific project requirements, ensuring consistency across the system. 11](#_Toc180059027)

[The database is optimized to handle large datasets without compromising on speed or efficiency. Indexing strategies and query optimization techniques have been implemented to enhance performance, ensuring that even complex queries return results in a timely manner. Additionally, the use of foreign keys and constraints ensures data integrity, particularly when dealing with relationships between entities. 11](#_Toc180059028)

[This dual-schema structure provides both flexibility for users and robustness for the application, ensuring that the system remains scalable and performant as the amount of data grows. 11](#_Toc180059029)

[3.4 User Interface 11](#_Toc180059030)

[When a user selects an entity, three input fields will appear: Ontology Class, Property Name, and Property Value. Alongside these fields is a green button labeled 'Save Ontology Properties.' Users can utilize this button to insert optionally information for the selected entity into the database. 15](#_Toc180059031)

[In the navigation bar, when users click on Browsing, they will be presented with a dropdown menu displaying all the entities currently existing in the database. Upon selecting an entity, the system will retrieve all the insertions associated with the selected entity from the database, similar to the functionality in the List of Documentation Entities. 22](#_Toc180059032)

[In the navigation bar, when users click on Knowledge Graph, they will be presented with a table containing three columns: Subject, Predicate, and Object. This table will display all the triples that can be created with the insertions provided to the program. Additionally, at the top of the page, a button labeled “Download RDF” will be available, allowing users to download all the displayed triples in a file named “Knowledge\_graph.nt.” 23](#_Toc180059033)

[4. Application Installation and System Requirements 24](#_Toc180059034)

[4.1 Installation method/steps 24](#_Toc180059035)

[4.2 Hardware/software requirements 27](#_Toc180059036)

[5. Epilogue 28](#_Toc180059037)

# Introduction

This section introduces the purpose and main objectives of the application developed in this thesis. It explains how the application simplifies data management through the use of ontologies and the RDF data model, enabling users to create and organize knowledge graphs. The section also provides an overview of the key features, including entity documentation, configuration options, and knowledge graph extraction, while highlighting the user-friendly interface that caters to both technical and non-technical users.

## 1.1 Purpose of the Application

In a world overflowing with data, organizing and understanding this information can be challenging. The application developed in this thesis is designed to make data management simpler by using tools like ontologies and the RDF (Resource Description Framework) data model. These tools help represent knowledge in a structured way, making it easier to create and display knowledge graphs. These graphs visually show relationships in data, making it easier to see patterns and gain insights.

The application allows users to document information about entities—such as people, objects, locations, and organizations—and connect them to form a structured knowledge graph. Through a user-friendly interface, users can configure the types of information they want to capture for various entity categories, link this information to relevant ontologies and vocabularies, and navigate the recorded data. While it helps organize and present data, it doesn’t focus solely on visualizing the graphs but rather on managing the relationships and structures within the data.

## Overview of Functions

The application includes several key features that simplify data management:

⦁ Configuration: Users can customize the system to meet their specific needs.

⦁ Documentation: Users can easily input and update data.

⦁ Browsing: Data is shown in various formats that enhance understanding.

⦁ Knowledge Graph Extraction: The application creates knowledge graphs to reveal connections and insights in the data.

These features work together to provide a comprehensive solution for managing complex data.

# Work Background

This section provides an overview of key concepts and technologies that underpin this research. Understanding these foundational elements is essential for grasping the significance and potential impact of the work presented in this thesis. The focus will be on Data Representation Ontologies, the RDF Data Model, Knowledge Graph and SPARQL.

## 2.1 Data Representation Ontologies

Data representation ontologies are structured frameworks that define a set of concepts and categories within a domain and establish relationships between them. These ontologies are used to model knowledge in a way that is both machine-readable and semantically rich, allowing for the effective organization and retrieval of information. Ontologies play a crucial role in the field of artificial intelligence, semantic web, and information systems, providing a common vocabulary for different systems to understand and communicate about a domain of knowledge.

The usefulness of data representation ontologies lies in their ability to facilitate interoperability between different systems, support data integration, and enhance the precision of information retrieval. By standardizing the representation of data, ontologies enable disparate systems to share, process, and interpret data consistently. They are particularly valuable in complex domains where data comes from various sources and formats, ensuring that the information is coherent and accessible. Furthermore, ontologies support reasoning and inference, allowing systems to derive new knowledge from existing data, which can be critical in decision-making processes.

Example: CIDOC CRM:

One prominent example of a data representation ontology is the CIDOC Conceptual Reference Model (CIDOC CRM). CIDOC CRM is an ontology developed for cultural heritage information, aimed at enabling the integration, mediation, and interchange of heterogeneous information from museums, libraries, and archives. It provides a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation, making it possible to integrate and share data across different institutions and platforms. By using CIDOC CRM, institutions can ensure that their data is semantically aligned, which enhances the quality and accessibility of cultural heritage information.

## 2.2 The RDF Data Model

The Resource Description Framework (RDF) is a data model that forms the backbone of the semantic web. RDF is designed to enable the representation of information about resources in a graph form, which consists of triples. Each triple comprises a subject, predicate, and object, which together represent a single fact or piece of information about a resource. The subject denotes the resource, the predicate denotes traits or aspects of the resource, and the object denotes the value of those traits or aspects. This simple yet powerful model allows for the flexible representation of data in a way that is both human-readable and machine-processable.

RDF's graph-based structure is particularly advantageous for representing complex relationships between different pieces of data. It allows data from various sources to be combined and queried in a seamless manner, making it a cornerstone of linked data initiatives and the broader semantic web ecosystem. RDF enables the integration of diverse data sets, supports the application of ontologies, and facilitates advanced querying and data analysis techniques.

## 2.3 Knowledge Graphs

Knowledge Graphs are structured representations that capture the relationships between entities in a graph format, making them an effective tool for organizing and retrieving complex information. Each entity (node) in a knowledge graph can represent a real-world object, concept, or idea, while the edges (links) represent the relationships between these entities. In essence, a knowledge graph is a knowledge base that utilizes a graph-structured data model, often derived from a set of RDF triples.

RDF Triples: A knowledge graph is constructed using RDF (Resource Description Framework) triples, which consist of a subject, predicate, and object. For example, in the triple "Alice (subject) knows (predicate) Bob (object)," Alice and Bob are entities, and the relationship is represented by the predicate "knows."

Applications: Knowledge graphs are widely used in various applications, including search engines (e.g., Google Knowledge Graph), recommendation systems, and artificial intelligence. They enhance the ability to draw insights from interconnected data and support advanced queries.

## 2.4 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is a powerful query language specifically designed for querying and manipulating RDF data. It allows users to write complex queries to retrieve specific information from knowledge graphs, similar to how SQL operates on relational databases.

Key Features:

Graph Pattern Matching: SPARQL allows users to specify patterns that describe the relationships they want to explore within the RDF triples.

Data Retrieval: Users can retrieve specific data points or relationships based on the conditions defined in their queries.

INSERT, DELETE, and UPDATE Operations: Beyond querying, SPARQL can also modify RDF data, allowing for dynamic updates to knowledge graphs.

Example Query:

*SELECT ?person ?friend*

*WHERE {*

*?person foaf:knows ?friend .*

*}*

This SPARQL query retrieves pairs of people who know each other, leveraging the RDF structure of the data.

Conclusion

Knowledge graphs and SPARQL together enable the representation and querying of complex relationships in a flexible manner. By utilizing RDF triples, knowledge graphs provide a rich framework for organizing information, while SPARQL offers the means to query that information effectively.

# ****Application Description****

## 3.1 Architecture and Technologies

### The project is built upon a modern web architecture that facilitates the dynamic generation and manipulation of data tables representing various entities and their relationships within a knowledge graph. The frontend is developed using React, allowing for a responsive and interactive user interface. The backend, implemented in Node.js, handles data processing and storage, utilizing Express as the web server framework to manage API requests efficiently.

### The knowledge graph serves as the core component of the application, enabling the organization of entities and their interconnections through an ontology-based structure. This architecture allows for seamless querying and visualization of data relationships, enhancing the overall user experience.

## 3.2 Functionality

### The application supports various functionalities aimed at managing and visualizing entity data within the knowledge graph framework. Key features include:

### 3.2.1 Configuration

### Users can customize their data inputs and entity definitions according to their specific needs, ensuring flexibility and scalability. The system allows users to create and manage their projects through a General Properties configuration interface, where they can define key aspects of their project. Within this setup, users can create necessary entities, tailoring each entity with specific "elements" that suit their requirements. These elements can include attributes, properties, and relationships between entities. Additionally, the platform supports the creation of custom vocabularies, allowing users to define elements with predefined insertions, ensuring consistency and standardization across their project. This flexibility empowers users to build robust data models that can adapt to different use cases, promoting seamless scalability as their project evolves.

### 3.2.2 Documentation (Data Input)

### The system provides comprehensive documentation that guides users in uploading and managing their data effectively. This includes clear instructions on how to insert new entities (i.e., new data) into the system, as well as how to format and validate that data to ensure accuracy. Additionally, users can easily access and review existing entries, with the option to edit or update them if necessary. This ensures that the data remains up-to-date and aligned with the user’s evolving needs, all while maintaining data integrity and consistency throughout the system.

### 3.2.3 Browsing

This page enables users to view their entities through a dropdown menu. Upon selecting an entity, all its corresponding insertions are displayed. If the selected entity contains another entity as part of its structure, the embedded entity is clickable and will redirect users to its respective page, where they can view its details. This intuitive navigation allows for seamless exploration of relationships between entities, making it easier for users to traverse and interact with the data.

### Knowledge Graph Extraction

Users have the ability to extract relevant information from the knowledge graph, facilitating further analysis and integration with other systems or applications. Once users interact with the knowledge graph feature, the system dynamically displays all valid RDF triples on the page. These triples are presented in an organized, readable format, allowing users to review the relationships and data points generated by the system. Additionally, at the top of the page, a dedicated button is available for users to easily download these triples in a .nt (N-Triples) file format. This download option ensures that users can efficiently export and utilize the data in other applications, or for deeper offline analysis and integration into broader datasets or ontologies.

## 3.3 Database

## The underlying database is written in MySQL and managed using MySQL Workbench, providing a robust and scalable solution for storing and managing data. It is designed to efficiently handle entity data and their relationships, ensuring fast access and retrieval of information. The database is divided into two distinct schemas to separate core application data from user-defined content.

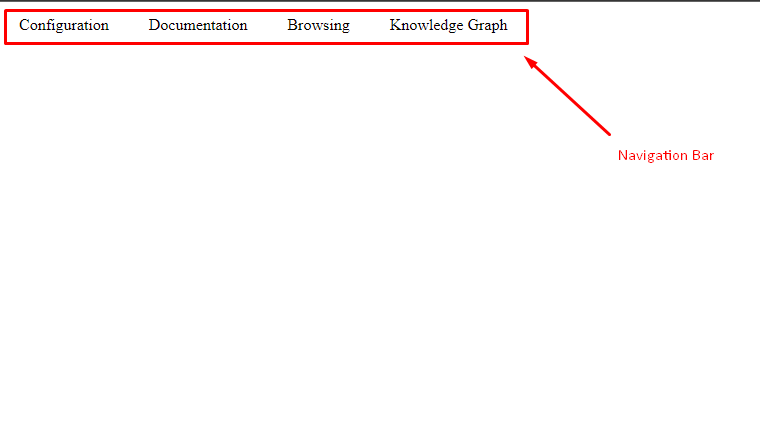
## The first schema contains tables essential for the functioning of the program, including configuration settings, and data types that are used throughout the system. These tables also manage key aspects of the connection between entities, ensuring that the relationships and interactions within the knowledge graph are maintained. While these tables are critical for the system’s operation, they are not directly accessible to users, protecting the integrity and security of the application. This schema also includes all the entities defined in the system, but their structure and data are controlled at a programmatic level. By keeping these tables isolated, the application can maintain optimal performance, avoid conflicts, and prevent accidental modifications that could affect system stability and the correctness of entity relationships.

## The second schema is dedicated to vocabulary management, where users can define and organize their custom vocabularies. This schema supports tables that store elements such as entity attributes, relationships, and predefined insertions. These vocabulary tables allow users to flexibly create and extend their data models based on their specific project requirements, ensuring consistency across the system.

## The database is optimized to handle large datasets without compromising on speed or efficiency. Indexing strategies and query optimization techniques have been implemented to enhance performance, ensuring that even complex queries return results in a timely manner. Additionally, the use of foreign keys and constraints ensures data integrity, particularly when dealing with relationships between entities.

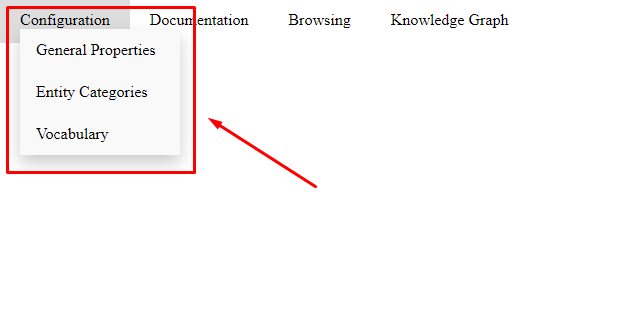
## This dual-schema structure provides both flexibility for users and robustness for the application, ensuring that the system remains scalable and performant as the amount of data grows.

## 3.4 User Interface

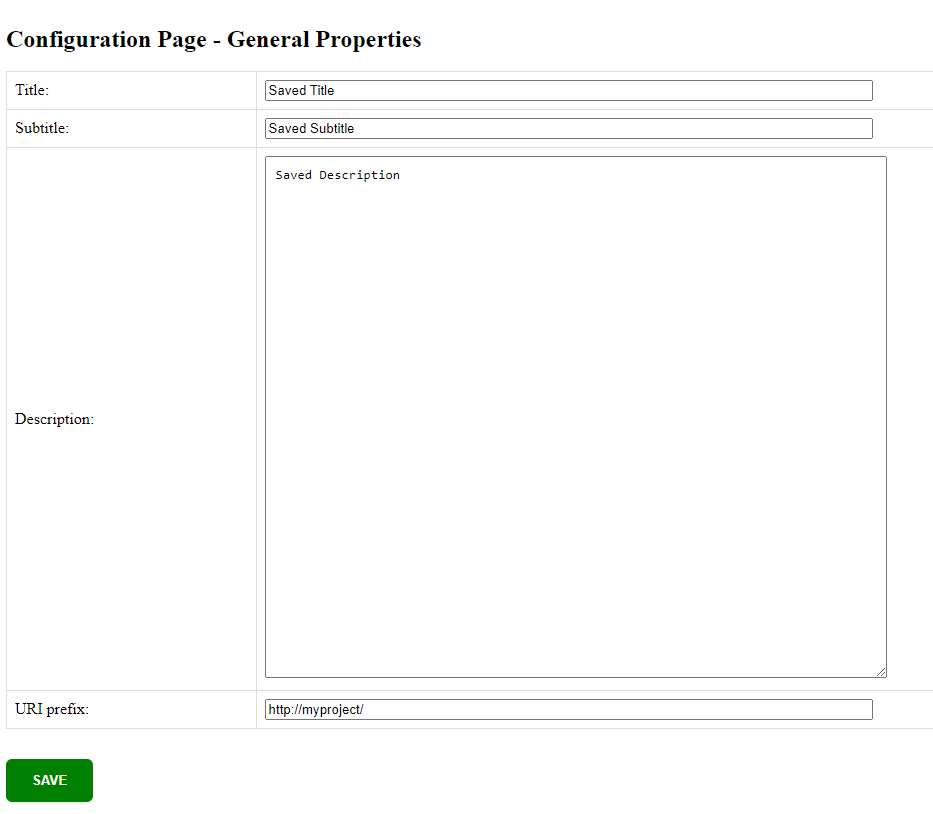


**Navigation bar:**

**When the application is launched, the navigation bar should be the only element displayed on the user's screen, presenting the available options for the user to select.**



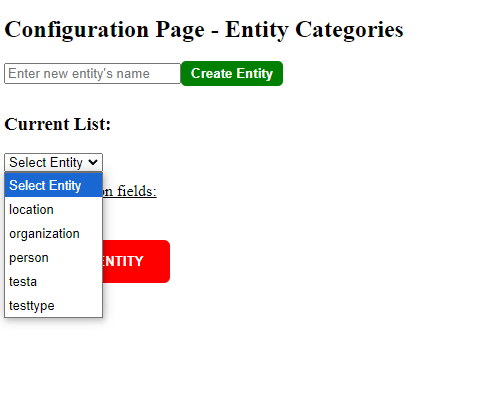
**Configuration:  
  
When the user selects the configuration option, three additional options will become accessible: General Properties, Entity Categories, and Vocabulary.**



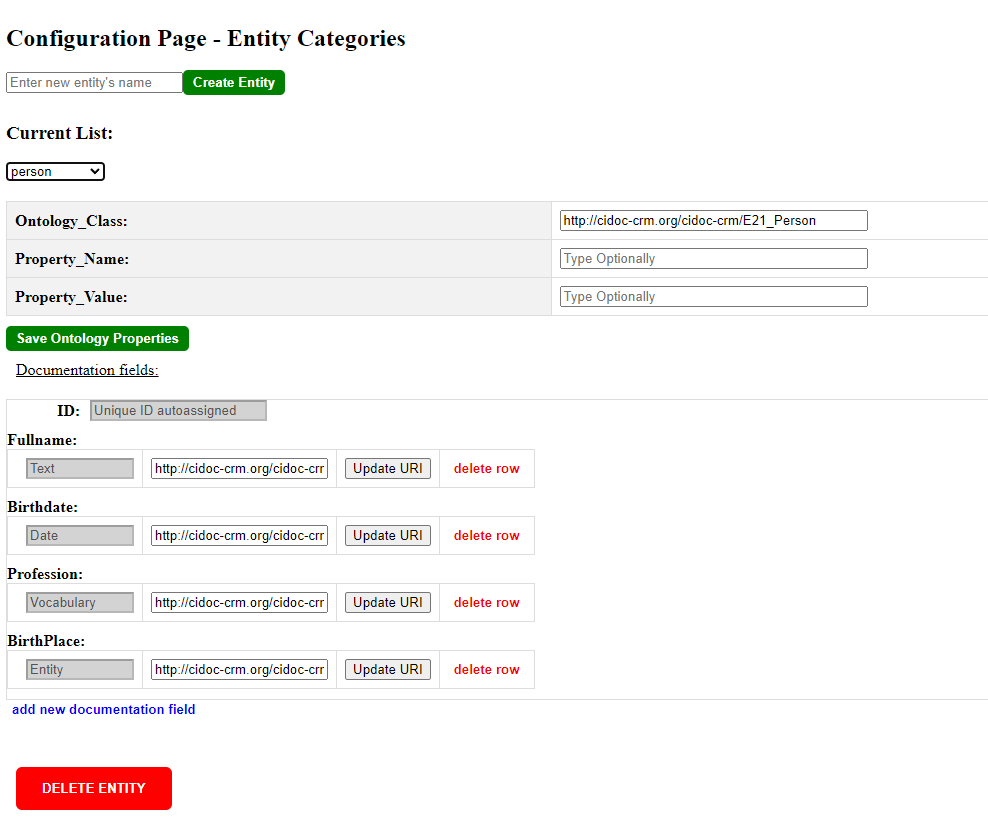
**In the General Properties section, there are four input fields, each labeled to indicate its purpose:**

1. **Title**
2. **Subtitle**
3. **Description**
4. **URI Prefix**

**Upon the initial launch of the project, these input fields are empty, allowing users to enter any desired information. The entries made in these fields do not affect the underlying code, with the exception of the URI Prefix, which initializes the URIs. The remaining inputs serve solely to provide context for editors regarding the project.**



In the Entity Category section, there is an input field accompanied by a green button labeled 'Create Entity.' Users can type the desired entity name into this field and press the green button. This action will create a new table in the database, named according to the input provided, which will represent the specified entity.

Below, there is a current list featuring a dropdown menu that displays all the tables (entities) that have been created thus far. Users can select the desired entity from this list to add attributes related to that entity.****

# When a user selects an entity, three input fields will appear: Ontology Class, Property Name, and Property Value. Alongside these fields is a green button labeled 'Save Ontology Properties.' Users can utilize this button to insert optionally information for the selected entity into the database.

Below, all the attributes of the selected entity are displayed, along with a labeled input field for the URI of each attribute. Each attribute also features two buttons:

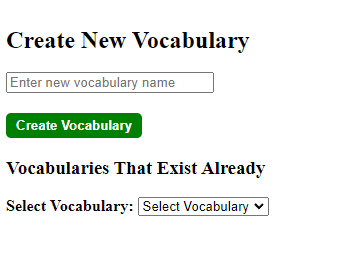
* A button labeled 'Update URI,' which inserts the URI into the database.
* A red button labeled 'delete row,' which removes the entire attribute from the entity.

At the end of the attributes list, there is a button labeled 'Add New Documentation Field.' Upon clicking this button, two input fields and one dropdown menu will appear:

1. The first input field is for the Attribute Name.
2. The dropdown menu allows the user to select the Data Type of the attribute, which can be another entity, a vocabulary, or a standard data type.
3. Lastly, there is an optional input field for the URI of the attribute.

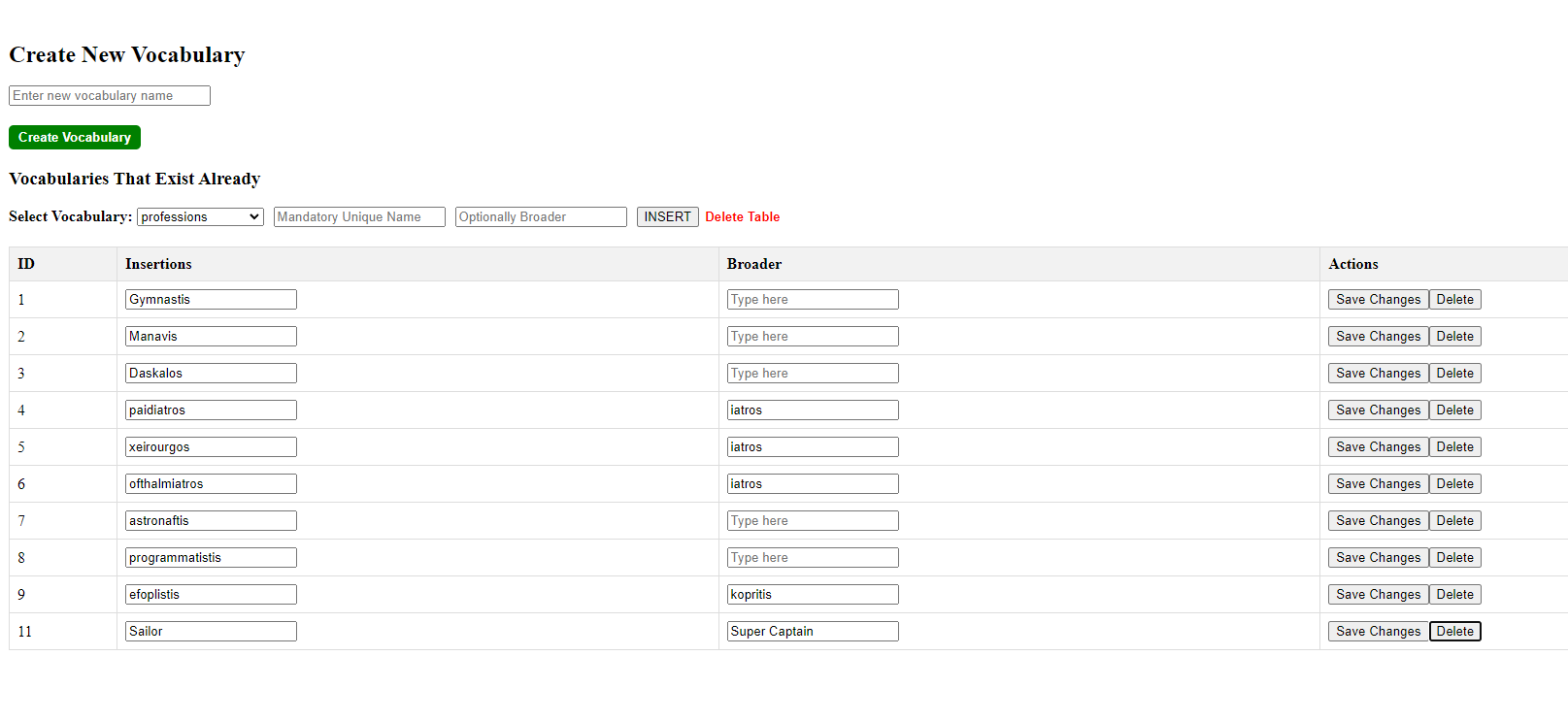
In the dropdown menu for selecting data types, the options will be organized in the following order: first, other entities; second, standard data types; and finally, vocabularies.

Lastly, there is a prominent red button labeled ‘DELETE ENTITY’. This button will delete the entire entity, including all associated attributes and any insertions within it.



Similar to the Entity Categories section, there is a Vocabulary section. When a user selects Vocabulary, a new page will be displayed, allowing the user to create a new vocabulary within a different schema in the database.

Also there is a dropdown below that contains all the existing vocabularies that have been already created.

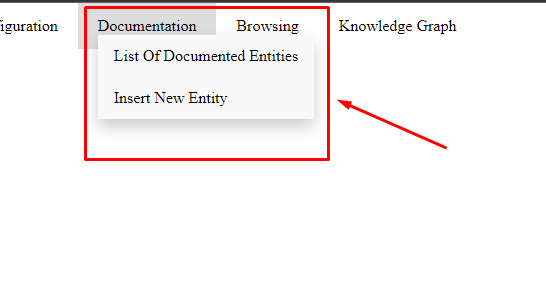


When a user creates a vocabulary, each vocabulary entry consists of two fields: one mandatory field for a unique name and another optional field for a broader term.

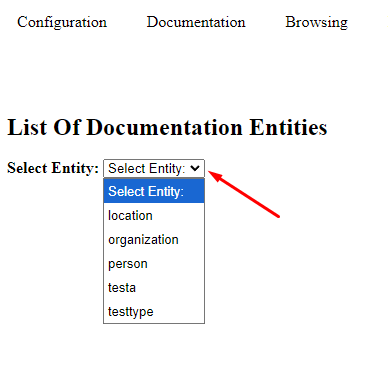
When a vocabulary is selected, two input fields will appear next to the dropdown menu, each with a placeholder indicating its purpose. Additionally, there are two buttons: one labeled 'Insert' and another labeled 'Delete Vocabulary.

Below this section, a table is presented with four columns: ID, Insertions, Broader, and Actions.

* The ID column uniquely identifies each entry in the table.
* The Insertions column displays the data or records added to the system.
* The Broader column represents the hierarchical relationships or categorizations of the inserted data.
* The Actions column contains buttons that allow the user to either save changes made to a specific row or delete the corresponding entry from the table.

**Documentation**:

**When the user selects the documentation option, two additional options will become accessible: List of Documentation Entities, and Insert New Entity.**



**In the ‘List of Documentation Entities’, users will find a dropdown menu labeled ‘Select Entity.’ This menu displays all the entities that have been created thus far. Users can select their desired entity from this list to view its associated attributes.**

 When a user selects an entity, a table will be displayed containing all the attributes of the selected entity, along with an additional column labeled Actions in the last row. The column includes the insertions created for the selected entity.

As Illustrated in the picture:

* If a cell within a column is light gray, it indicates that the cell is empty.
* Conversely, if the insertions are blue, it signifies that the insertion corresponds to another entity. The attributes of the selected entity are separated by dashes (-).
* Last column features action buttons for Edit and Delete.

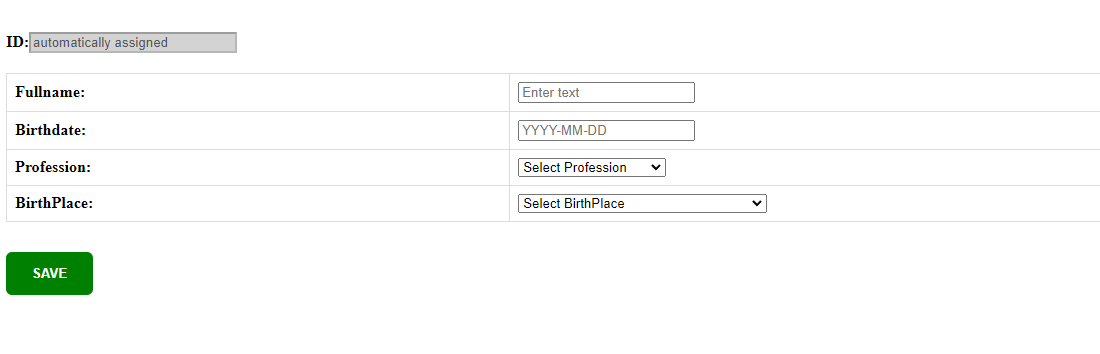


In the event that a user selects Edit, the corresponding row will become editable, as demonstrated in the previous image. Standard insertions will be displayed in input fields, while vocabularies and entities will be presented in their respective dropdown menus. This design allows the user to select an alternative without the risk of making an error.

Additionally, the Edit button will be renamed Save, while the Delete button will be renamed Cancel.



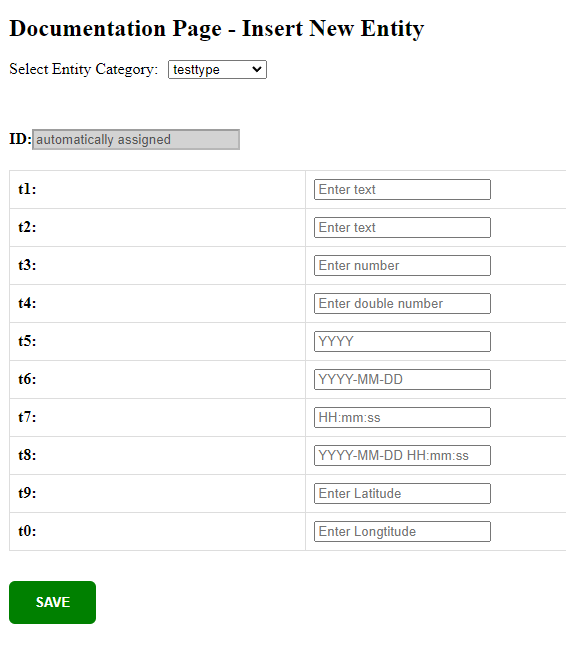
In the ‘Insert New Entity’, users will find a dropdown menu labeled ‘Select Entity Category’. This menu displays all the entities that have been created thus far. Users can select their resired entity from this list to view its associated attributes and insert their desired values.



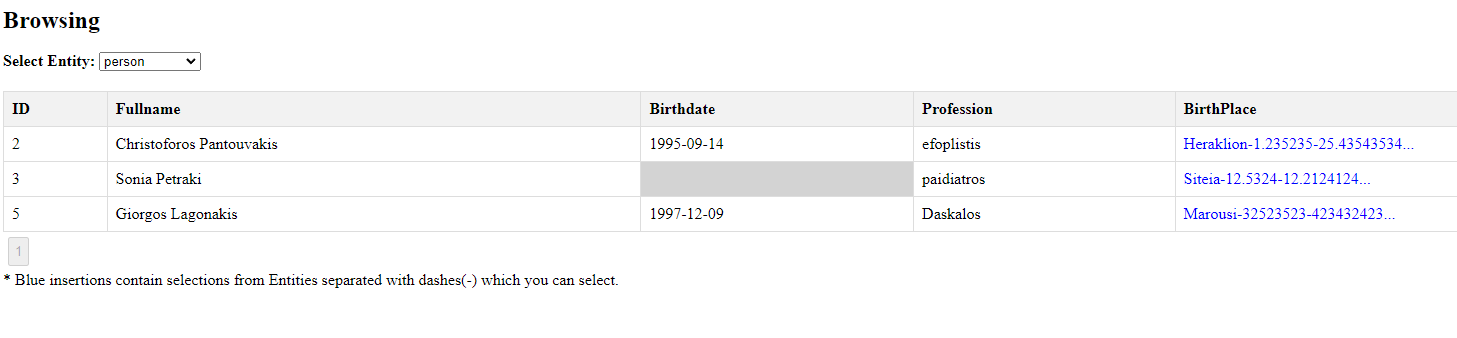
Each entity is assigned a unique ID automatically by the system, and the format is illustrated in the previous image.

Each entity is assigned a unique ID automatically by the system. Additionally, there is a green button at the end of the page labeled 'SAVE.' When users enter their values into the input fields or make their selections, they can press the 'SAVE' button to submit their insertion.

The format is illustrated in the previous image.



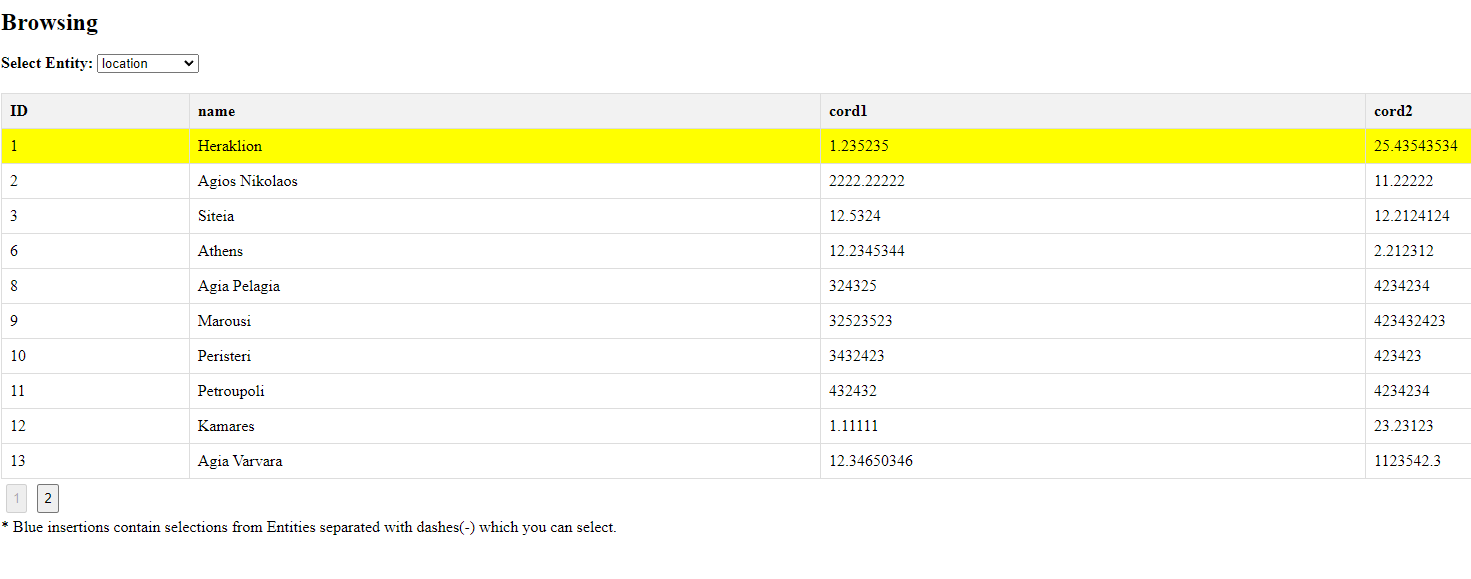
In the previous image, the placeholders for different types of insertions can be observed, designed to enhance the user experience.



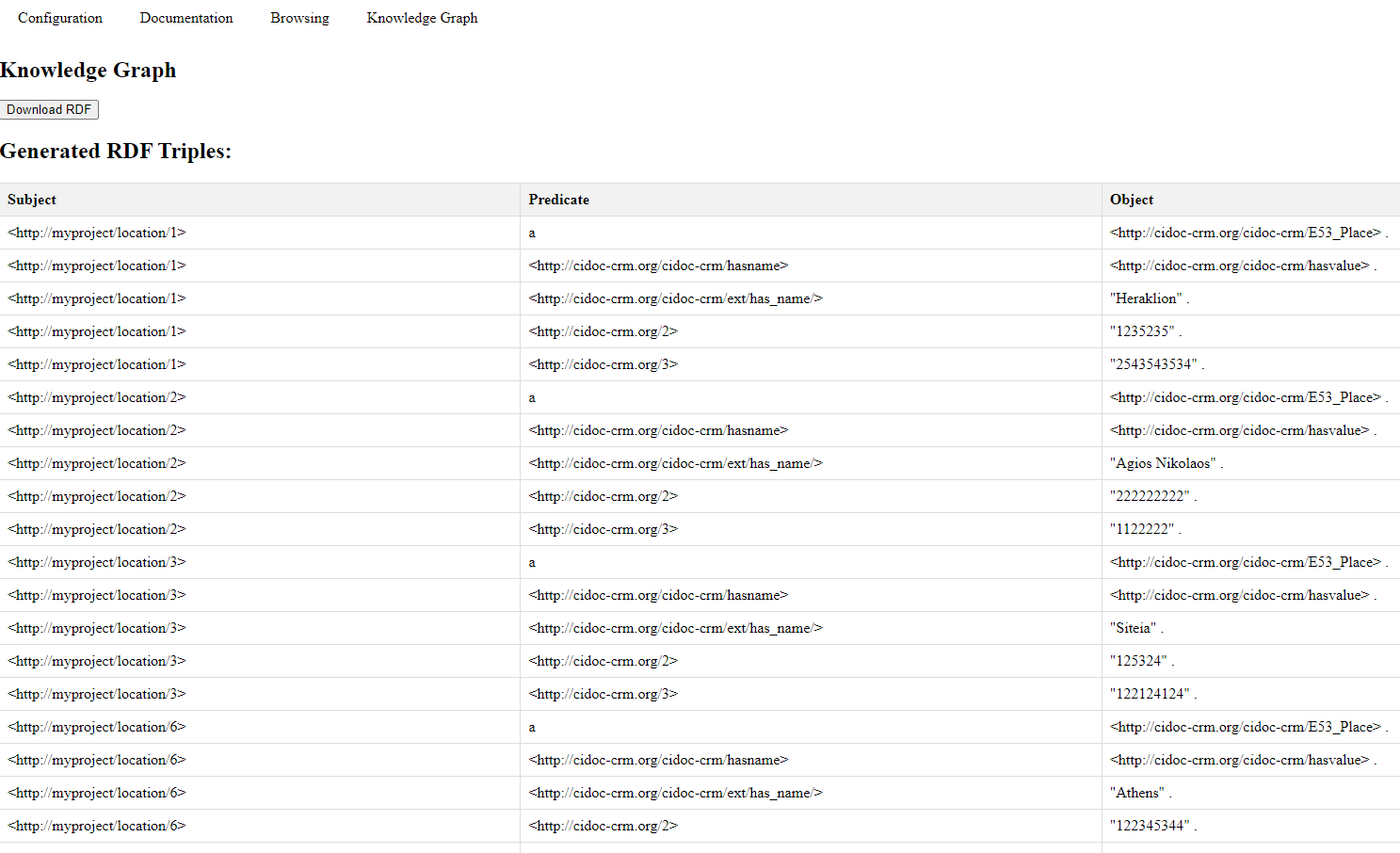
**Browsing**:

# In the navigation bar, when users click on Browsing, they will be presented with a dropdown menu displaying all the entities currently existing in the database. Upon selecting an entity, the system will retrieve all the insertions associated with the selected entity from the database, similar to the functionality in the List of Documentation Entities.

The key difference between the two is that in the Browsing section, there is no Actions column, which means users do not have the authority to edit or remove insertions.



One notable feature of the Browsing page is that all entities are accessible. If the total number of characters in an insertion exceeds 30, only the first 30 characters will be displayed. Furthermore, the entities are clickable, redirecting the user to the respective entity page, where the selected row will be highlighted in yellow to indicate the user’s selection.



**Knowledge Graph**:

# In the navigation bar, when users click on Knowledge Graph, they will be presented with a table containing three columns: Subject, Predicate, and Object. This table will display all the triples that can be created with the insertions provided to the program. Additionally, at the top of the page, a button labeled “Download RDF” will be available, allowing users to download all the displayed triples in a file named “Knowledge\_graph.nt.”

# 4. Application Installation and System Requirements

## 4.1 Installation method/steps

Database

To use the application, you need to install and configure MySQL Workbench as the database to store data.

1. Download MySQL

You can download MySQL from the official MySQL website:

*https://dev.mysql.com/downloads/*

2. Install MySQL

Follow the installation instructions for your operating system. During installation:

* Set a root password (you will need this later).
* Choose a default port (typically 3306).
* Enable MySQL to start automatically if desired.

3. Create a Database and User

After installing MySQL, follow these steps to set up the required database and user:

Open the windows command-line and go to the installation folder (by default):

*cd C:\Program Files\MySQL\MySQL Server 8.0\bin*

Log in as the root user:

*mysql -u root -p*

Create the Database:

*CREATE DATABASE your\_database;*

Create a new user:

*CREATE USER 'your\_username'@'localhost' IDENTIFIED BY 'your\_password';*

*GRANT ALL PRIVILEGES ON your\_database.\* TO 'your\_username'@'localhost';*

*FLUSH PRIVILEGES;*

**Important**: Save these credentials for later use in the backend file server.js.

*const connection = mysql.createConnection({*

*host: 'localhost',*

*user: 'your\_username', // Replace with your MySQL username*

*password: 'your\_password', // Replace with your MySQL password*

*database: 'your\_database' // Replace with your Database name*

*});*

Logging into MySQL Workbench

After setting up the database and user, you need to log into MySQL Workbench for the first time:

Open MySQL Workbench.

Click on the MySQL Connections to open a new connection.

Enter your root user credentials and click OK.

Once logged in, execute the commands in the FirstInitialize.sql **after replacing** the first line with your database name file to set up any additional tables or data necessary for your application. You can find the file here:

*https://github.com/Pantouvakis/Knowledge-Graph-System-Using-RDF-and-Ontologies/blob/main/FirstInitialize.sql*.

Ensure that the following software is installed on your system:

Node.js (v12 or higher): You can download Node.js from the official site:

*https://dev.mysql.com/downloads/*

npm: This comes bundled with Node.js.

Steps to Set Up the Project

1.Clone the project from the repository:

*git clone https://github.com/Pantouvakis/Knowledge-Graph-System-Using-RDF-and-Ontologies*

2.Navigate to the Project Directory

Open a terminal or command prompt and navigate to the folder where the project was cloned

3.Install Dependencies

After navigating to the project directory, install all the necessary dependencies by running:

1. Navigate to the frontend directory:

*cd frontend*

*npm install*

1. Navigate to the backend directory:

*cd ../backend*

*npm install mysql2*

*npm install*

1. Start the Backend Server

The backend server is built using Node.js and Express. To start the server, run:

*node server.js*

In terminal after you start the backend you will see that the server is running if everything is correct.

1. Start the Frontend (React)

To run the React frontend application, navigate to the frontend directory and start the React development server:

*npm start*

This will launch the frontend on http://localhost:3000 by default, but ensure that it corresponds with the backend port if configured differently.

Once the setup is complete, both the backend and frontend servers should be running

Frontend will be available at http://localhost:3000

## 4.2 Hardware/software requirements

* Operating System: Windows, macOS, or Linux
* RAM: At least 4 GB
* Processor: Any modern dual-core processor or better
* Node.js Version: v12 or higher
* Browser: Latest version of Chrome, Firefox, or any modern browser

# Epilogue

In conclusion, this project has successfully demonstrated the value of entities and knowledge graphs in enhancing data representation and understanding. Through the development process, I encountered various challenges that deepened my understanding of the complexities involved in managing data relationships. The findings suggest significant potential for the application of knowledge graphs in diverse fields, and I look forward to exploring further enhancements and implementations of this work. Ultimately, this project contributes to the ongoing conversation about the importance of effective data management and representation in our increasingly data-driven world.