THE UNIVERSITY OF DANANG

**UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**FACULTY OF ADVANCED SCIENCE AND TECHNOLOGY**

|  |  |
| --- | --- |
|  |  |

**CAPSTONE PROJECT REPORT**

**M1 REPORT**

**DESIGN AND DEVELOPMENT OF A DATA VISUALIZATION LIBRARY FOR SPARQL QUERIES**

|  |  |  |
| --- | --- | --- |
| **Student:** |  | **Minh Huy Do Class: 20PFIEV3** |

|  |  |  |
| --- | --- | --- |
| **Academic advisor:**  **Supervisor:**  **Co-Supervisor:** |  | **Dr. Thi My Hanh Le**  **Assoc. Prof., Dr. Aline Menin**  **Prof., Dr. Catherine Faron** |

**Danang, 03 / 2025**

**Contents**

[1. Overview 3](#_Toc194096706)

[1.1. Introduction 3](#_Toc194096707)

[1.2. Existing Solutions 3](#_Toc194096708)

[2. Preliminary Solution 5](#_Toc194096723)

[2.1 Preliminary techniques and solution: 5](#_Toc194096724)

[2.2 System architecture: 7](#_Toc194096725)

[2.3 Required tools and materials 7](#_Toc194096730)

[2.3.1 Programming language: 7](#_Toc194096731)

[2.3.2 Libraries and frameworks: 8](#_Toc194096732)

[2.3.3 Data resources: (SPARQL Endpoints): 8](#_Toc194096733)

[2.3.4 Development Environment: 8](#_Toc194096734)

[3. Expected Results 9](#_Toc194096735)

[4. Evaluation Method (User-based evaluations) 9](#_Toc194096738)

[4.1 Performance Evaluation 9](#_Toc194096739)

[4.2 Usability Testing 9](#_Toc194096744)

[4.3 Interactivity & Visualization Evaluation 9](#_Toc194096746)

[4.4 Comparison with Existing Tools 9](#_Toc194096749)

[5. Plan for Implementation 10](#_Toc194096752)

[References 12](#_Toc194096791)

# 1. Overview

## 1.1. Introduction

In today's data-driven world, vast and complex datasets are increasingly published using RDF (Resource Description Framework) [1] and made available as LOD (Linked Open Data) [2] These datasets hold immense potential for decision-making across various fields, but their value lies in the ability to effectively analyze and visualize the information they contain. Custom visualizations, while powerful, are often hard-coded, requiring specialized programming skills and making them challenging to reuse across different applications. Additionally, these solu ons lack transparency, making it difficult to understand how visualization and interactivity are configured. Simplifying the creation of visualizations for non-programmers, while taking into account both data structures and user intent, is therefore essential. Despite advancements, most exis ng visualiza on approaches are tailored for datasets in formats like CSV or JSON, overlooking the unique complexies of RDF datasets.

The objective of the project is to design and develop a library to simplify and enhance the creation of visualizations for SPARQL [3] queries, making them accessible to semantic web experts and decision makers without the need for advanced programming skills. Develop a visualization library inspired by tools such as ECharts [4], MGExplorer [5], and D3.js [6]. This library will transform SPARQL queries and visual mapping configurations into interactive, reusable visualizations that can be directly applied to a variety of projects.

## 1.2. Existing Solutions

Currently, there are only a few papers proposing tools to support RDF data visualization and SPARQL queries. However, they all have their own limitations, which prevent them from fully meeting users' needs for data visualization in a flexible and easy-to-use way. Some popular tools and papers include:

* **Echarts:** ECharts is a powerful JavaScript library that provides a variety of interactive charts such as bar charts, line charts, network charts, etc.
* **Advantages:** Supports many types of beautiful and highly customizable charts. Well integrated with web applications and has high performance. Supports dynamic interactions, giving users a better experience when analyzing data.
* **Disadvantages:** Does not directly support RDF or SPARQL, users need to process data before being able to use ECharts. No module available to visualize data in RDF graph structure.
* **D3.js:** D3.jsa popular JavaScript library for dynamic data visualization. It provides the ability to create complex charts by directly manipulating the DOM and SVG.
* **Advantages:** Provides extreme customization for different types of charts. Good support for graphical data visualization. Has a large development community and rich documentation.
* **Disadvantages:** Requires advanced JavaScript programming knowledge to use effectively. No RDF or SPARQL support available, requires data processing before display. Creating complex visualizations can be time-consuming and laborious.
* **MGExplorer - LDViz:** MGExplorer is a tool developed by the WIMMICS [7] team and it allows exploring SPARQL results through another tool, LDViz [8], which is used to explore RDF data through a graphical interface, allowing users to search and browse data by link structure.
* **Advantages:** Supports visualization of RDF data in graph form. Integrated with direct SPARQL query capability.
* **Disadvantages:** LDViz allows to configurate different SPARQL queries to be visualized with MGExplorer. But, indeed, the visual mapping is limited to the existing visualization techniques.
* **VOWL (Visualization of Ontologies Web Language):** VOWL [9] is a specialized tool for visualizing ontologies based on RDF/OWL, focusing on displaying the relationships between classes and properties in an ontology.
* **Advantages:** Provides a visual approach to understanding the structure of the ontology. Graph-based data visualization, suitable for viewing relationships between entities. Supports interaction and direct navigation in the data model.
* **Disadvantages:** Only focuses on ontology visualization, not a general tool for all RDF data. Does not support many different types of graphs other than node-link graphs. Not much customization of the display interface.
* **Some related papers :** Encodable: Configurable Grammar for Visualization Components [10] and Knowledge Graph Based Visual Search Application [11].
* **Disadvantages:** Encodable lacks chart extensions like legends, coordinate axes, and many interactive features that visual charts need. And the Knowledge Graph only allows users to use a very limited number of charts.

Given these limitations, the proposed solution aims to create an accessible, reusable JavaScript-based library tailored for SPARQL query visualization, lowering the entry barrier for users who have non-experts on visualization. The library will only need some programming skills and specially regarding web semantics.

# 2. Preliminary Solution

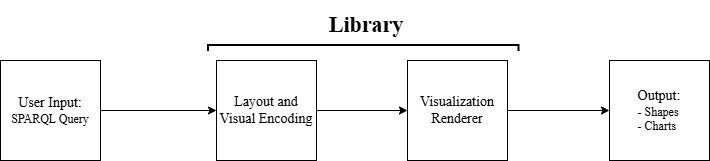
## 2.1 Preliminary techniques and solution:

The proposed solution is a comprehensive, JavaScript-based library designed to empower users with the ability to generate interactive, dynamic, and reusable visualizations from SPARQL query results, all without requiring in-depth programming knowledge. This solution aims to bridge the gap between technical and non-technical users, offering a powerful yet accessible tool for visualizing RDF data. Key features of the library include:

* **Support various RDF data, diverse chart type:** The library is built to support a wide range of RDF data formats (Turtle format, XML, …), ensuring compatibility with diverse datasets. Ensure that users can work with any RDF data, irrespective of how it’s structured or serialized. Allow users to choose the most appropriate chart type depending on the nature of the data and the insights they want to extract. Whether it is a network chart, bar chart, or pie chart, the library offers a variety of options that can meet a variety of visualization needs.
* **Vega Grammar - Flexible customization:** One of the core strengths of the library is its flexible customization, which allows users to easily tailor visualizations to their needs without writing complex code. To achieve this, we decided to use Vega grammar [12] - a powerful declarative language for creating data-driven visualizations. Vega allows users to define charts through a clear JSON structure, making it easy to describe visualization components such as axes, legends, colors, interactions, and animations. Users can customize and extend the available configurations or create new ones based on their specific needs. Since Vega uses JSON, users do not need to write pure JavaScript to create charts, making customization simpler.
* **Web Component – High performance and Scalability Solution:** We decided to use Web Components to build the library because it is a modern, standard and more scalable solution than traditional JavaScript frameworks. The main reasons include:
* **Support for native web standards:** Web Components are natively supported by browsers without the need for external libraries or frameworks, ensuring long-term compatibility.
* **Strong packaging capabilities and high reusability:** Shadow DOM technology isolates the CSS and JavaScript of each component, avoiding conflicts with other parts of the application. The built components can be reused in many different projects without major modifications.
* **High performance:** Since Web Components run directly in the browser without the need for additional runtime, performance is optimized, which is especially important when dealing with large data visualization.
* **Easy integration into any web application:** Web Components can be used directly in HTML without depending on any specific framework, allowing for more flexible integration.

When choosing between MDN Web Component [13] and Stencil [14], we decided to use MDN Web Component for the following reasons: it is based on official web standards, MDN Web Component is highly stable and is not affected by changes in third-party frameworks. Unlike Stencil, MDN Web Components does not require the use of TypeScript or JSX, providing more flexibility in the development process.

## 2.2 System architecture:



*Figure 1: Block diagram of the system*

The system is designed with a 5-step processing process:

* **User Input (SPARQL Query):** Users enter SPARQL queries to retrieve RDF data from endpoints or load RDF data directly. Support queries from popular sources such as DBpedia, Wikidata, custom RDF data stores.
* **Layout and Visual Encoding:** Define the data (components and vega grammar that users use). Assign visual attributes (color, size, shape) to data elements. Provide the ability to customize visual encoding rules.
* **Visualization Renderer:** This is the important step of the library. Use the functions of the visualization library corresponding to the data to display the data. Support dynamic, interactive charts (zoom, filter, highlight).
* **Output:** Display the results (charts) on the web interface according to the input data and the selected chart.

## 2.3 Required tools and materials

To develop a SPARQL data visualization library, the necessary tools and resources include:

### 2.3.1 Programming language:

* JavaScript: The main language for library development, ensuring the ability to run on browsers.

### 2.3.2 Libraries and frameworks:

* D3.js:
* Powerful JavaScript library for DOM and SVG based data visualization.
* Supports drawing graphs, charts, and tree structures suitable for RDF data.
* Provides interactivity such as zooming, dragging, highlighting data.
* Echarts:
* Dynamic charting library supporting chart types.
* Easily integrate with data from SPARQL to visualize statistical information.
* Web Components:
* Provides the ability to encapsulate UI components for easy reuse.
* Helps build intuitive configuration interfaces.

### 2.3.3 Data resources: (SPARQL Endpoints):

* Dbpedia (<https://dbpedia.org/sparql>): Provides data from Wikipedia in RDF format.
* Wikidata (<https://query.wikidata.org/>): Contains semantic data from various domains.
* Custom sources: User can import SPARQL results from files or personal endpoints.

### 2.3.4 Development Environment:

* Visual Studio Code: Popular source code editor, supporting many utilities to help write JavaScript code effectively. Currently we are using VS Code but in the future will aim to expand to more frameworks.
* Node & NPM (Node Package Manager): Provides a runtime environment for JavaScript, allowing code to be executed outside the browser. And NPM [15] is a library management system, helping to install and manage dependencies for the project.

# 3. Expected Results

The project aims to create a JavaScript library that can visualize SPARQL query results in an intuitive, easy-to-use way. The expected outcomes of the project include:

* A reusable JavaScript library for visualizing SPARQL query results that does not require complex programming and supports a variety of visualization types.
* Reducing the technical barrier: Users do not need in-depth visualization expertise to create data visualizations. Allowing users to easily change the display style, color, size, layout without editing the source code and improving the user experience with an interactive interface, making it easy to explore and analyze data.

# 4. Evaluation Method (User-based evaluations)

## 4.1 Performance Evaluation

* Scalability: Test the library’s ability to handle large RDF datasets. Evaluate how well the library maintains performance when rendering complex datasets.
* Some support tools :
* Chrome DevTools : To measure rendering time.
* Benchmark.js : To measure execution time of JavaScript functions.

## 4.2 Usability Testing

* User Feedback: Collect feedback from key user groups, including semantic web researchers. Use UserTesting and NPM to allow users to actually experience the library and give feedback.

## 4.3 Interactivity & Visualization Evaluation

* Interactivity: Evaluate whether users can interact with visualizations by zooming, filtering, and manipulating the data dynamically.
* Aesthetics and Visualization Quality.

## 4.4 Comparison with Existing Tools

* Competitive Evaluation: Compare the library with other tools like VOWL, and MGExplorer to identify its strengths and weaknesses.
* Improvement Analysis: Identify areas for improvement and gather insights for future versions of the library.

# 5. Plan for Implementation

|  |  |  |
| --- | --- | --- |
|  | **Tasks** | **Timeline** |
| 1 | * Do the web semantic’s course. * Study tutorial web components. * Check how to export components at npm package. | 1 week  (06/02 – 13/02) |
| 2 | * Create some chart’s examples: Pie chart, Bar chart, Node Link Diagram, Choropleth map. * Do the english version course of semantic web. | 4 days  (13/02 – 17/02) |
| 3 | * Separate each chart components. * Hide data from outside. * Create color picker for charts. * Import the package for new project. * Do the RDF Lab. | 1 week  (18/02 – 25/02) |
| 4 | * Do the RDF Schema Lab. * Do the SPARQL Lab. | 1 week  (25/02 - 03/03) |
| 5 | * Apply Pie Chart and Bar Chart with Vega Grammar. * Use SPARQL query’s results. | 1 week  (04/03 - 11/03) |
| 6 | * Extract the vega specifications for each different charts. * Create legend for charts, add the title for axes of barChart | 1 week  (11/03 – 18/03) |
| 7 | * Use Vega Lite instead of Vega. * Put the default color for charts if the parameter color is not present. * Apply more subtypes per chart type. * Find the data for Map and Node Link Diagram | 1 weeks  (18/04 – 25/03) |
| 8  (Predict) | * Extend the capabilities of the library by integrating deeper with Vega Lite. * Apply advanced visualization techniques to more chart types: subtypes of bar charts and pie charts * Improve the flexibility (eg. Change the colors, …) and interoperability of the library. * (Continuous) Test and improve the stability of bar chart and pie chart. * (Continuous) Refine the source code, optimize the library structure to improve performance. | 4 weeks  (25/03 – 22/04) |
| 9  (Predict) | * Focus on choropleth map and node link diagram. * Apply more charts: Treemap, Scatter, … * (Continuous) Refine the source code, optimize the library structure to improve performance. * (Continuous) Test and improve the stability of choropleth map and node link diagram with larger data sets. | 4 weeks  (22/04 – 20/05) |
| 10 (Predict) | * Exporting the library as npm package * Testing on other platforfms (reactjs, anguilarm etc) | 3 weeks  (20/05 – 10/06) |
| 10  (Predict) | * Comprehensively evaluate the library, fix bugs, and improve user experience. * Test on different environments to ensure compatibility. * Collect feedback from real users and adjust the product based on feedback. | 2 weeks  (10/06 – 24/06) |
| 11  (Predict) | * Analyze user experiments outcomes * Synthesize research and development results into detailed reports. * Prepare presentations, product demos, and highlight key improvements. * Share experiences and lessons learned from the library development process. | 1 week  (24/06 – 01/07) |

# References

|  |  |
| --- | --- |
| [1] | "Semanctic Web Standards - Resouce Description Framework," [Online]. Available: https://www.w3.org/RDF/. |
| [2] | "W3C - Linked Open Data," [Online]. Available: https://www.w3.org/egov/wiki/Linked\_Open\_Data. |
| [3] | "W3C - SPARQL 1.1 Query Language," [Online]. Available: https://www.w3.org/TR/sparql11-query/. |
| [4] | "Echart," [Online]. Available: https://echarts.apache.org/en/index.html. |
| [5] | Menin, A., Cava, R., Dal Sasso Freitas, C. M., Corby, O., & Winckler, M., *Towards a Visual Approach for Representing Analytical Provenance in Exploration Processes.,* 05-09 July 2021. |
| [6] | "D3js," [Online]. Available: https://d3js.org/. |
| [7] | "Project-team: WIMMICS," [Online]. Available: https://www.inria.fr/en/wimmics. |
| [8] | Menin, A., Maillot, P., Faron, C., Corby, O., Freitas, C. M. D. S., Gandon, F., & Winckler, M., "A tool to assist the multidimensional exploration of SPARQL endpoints.," 2023. |
| [9] | S. N. F. H. T. E. Steffen Lohmann, "Visualizing Ontologies with VOWL". |
| [10] | K. Wongsuphasawat, *Encodable: Configurable Grammar for Visualization Components,* 10 2020. |
| [11] | Pawandeep Kaur Betz, Tobias Hecking, Andreas Schreiber, Andreas Gerndt, "Knowledge Graph Based Visual Search Application". |
| [12] | "Vega-Lite - A grammar of Interactive Graphics," [Online]. Available: https://vega.github.io/vega-lite/. |
| [13] | "MDN Web Component," [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web\_components. |
| [14] | "Stencil JS," [Online]. Available: https://stenciljs.com/. |
| [15] | "NPM JS," [Online]. Available: https://www.npmjs.com/. |