# **S**ervice-**O**riented **A**rchitecture (SOA) Overview

Main Modules and Their Responsibilities

Each microservice will have a specific task:

**Frontend/User Interface (UI) Module**:

A web interface allowing users to interact with the system, submit queries, and view the results. The UI will communicate with the backend microservices via a REST API.

* **Input**: User queries, metadata to classify/compare, user preferences for display.
* **Output**: Visualizations, comparison reports, JSON-LD/HTML data.
* **Role**: Provides an interface to submit queries, configure comparisons, view classification results.

**Backend Services**:

* **Query Service**:

Manages SPARQL queries to retrieve data from the RDF datasets.

* **Input**: SPARQL queries from the UI or backend services.
* **Output**: Queried RDF data (in XML, JSON, or JSON-LD format).
* **Role**: Interacts with the SPARQL endpoint, retrieves relevant RDF data, and passes it on to other services.
* **Visualization Service**:

Processes and renders data in a human-readable format

* **Input**: RDF data from the Query Service.
* **Output**: Visual representations in HTML, JSON-LD, charts, or tables.
* **Role**: Transforms metadata into visual representations for better understanding, including data graphs, timelines, or attribute relationships.
* **Classification Service**:

Applies Machine Learning (ML)/ Natural Language Processing (NLP) or rule-based techniques to classify data based on Schema.org or other ontologies.

* **Input:** RDF data or JSON-LD from Query Service.
* **Output:** Classified data, often annotated with Schema.org or another ontology.
* **Role:** Categorizes data based on content, product types, or other schemas. May use a mix of machine learning and rules-based classification.
* **Comparison and Alignment Service**:

Compares datasets or schemas, identifying similarities and differences, and performs ontology alignment.

* **Input**: Two or more RDF datasets for comparison.
* **Output**: Aligned or mapped datasets; similarity scores.
* **Role**: Matches and aligns similar data points or entities, identifies differences and similarities, and integrates ontology alignment tools.
* **Statistics Service**:

Computes and exposes statistics in RDF Data Cube vocabulary, allowing for visualization.

* **Input**: Processed RDF data, possibly from the Query Service.
* **Output**: Statistical summaries in RDF Data Cube vocabulary.
* **Role**: Calculates statistics (counts, distributions) on metadata and exposes these summaries, allowing further querying.

**Database:** A triple store (e.g., Apache Jena, Virtuoso) supporting SPARQL queries and storing RDF data.

**API Gateway:** Manages and routes requests, handling client authentication, load balancing, and caching.

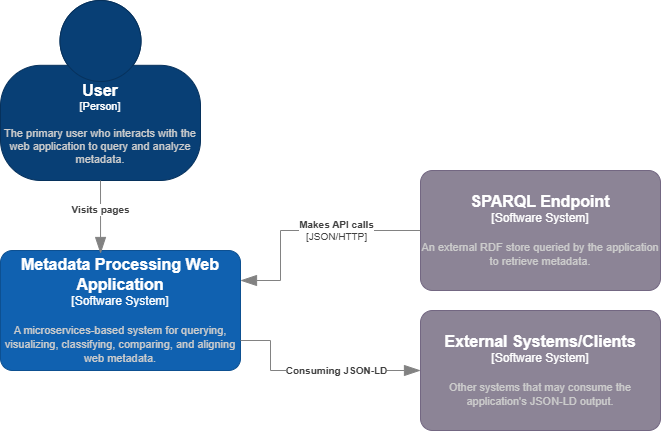
**Context Diagram (C4 Model Level 1)**

At the highest level, the Context Diagram shows how the web application fits into the broader environment and the main user groups and external systems it interacts with.

**Primary System**: **Metadata Processing Web Application** — A microservices-based system for querying, visualizing, classifying, comparing, and aligning web metadata.

**Primary Actors**:

* **User**: The primary user who interacts with the web application to query and analyze metadata.
* **SPARQL Endpoint**: An external RDF store queried by the application to retrieve metadata.
* **External Systems/Clients**: Other systems that may consume the application's JSON-LD output.

****

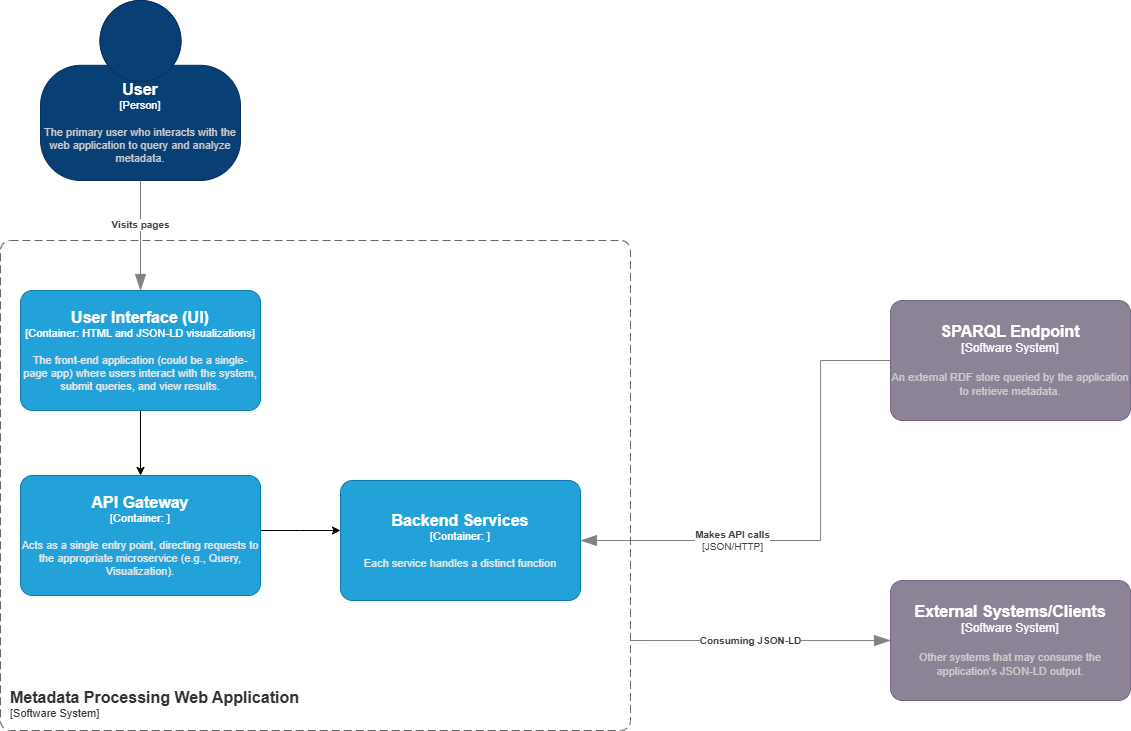
**Container Diagram (C4 Model Level 2)**

The **Container Diagram** breaks down the web application into individual containers, showing how each component interacts. Each container has a specific role and responsibility within the application.

**User Interface (UI)**: The front-end application is a single-page app where users interact with the system, submit queries, and view results.

**API Gateway**: Acts as a single entry point, directing requests to the appropriate microservice (e.g., Query, Visualization).

**Backend Services**: Each service handles a distinct function.



**Query Service**: Handles communication with the SPARQL endpoint, sending SPARQL queries to retrieve data.

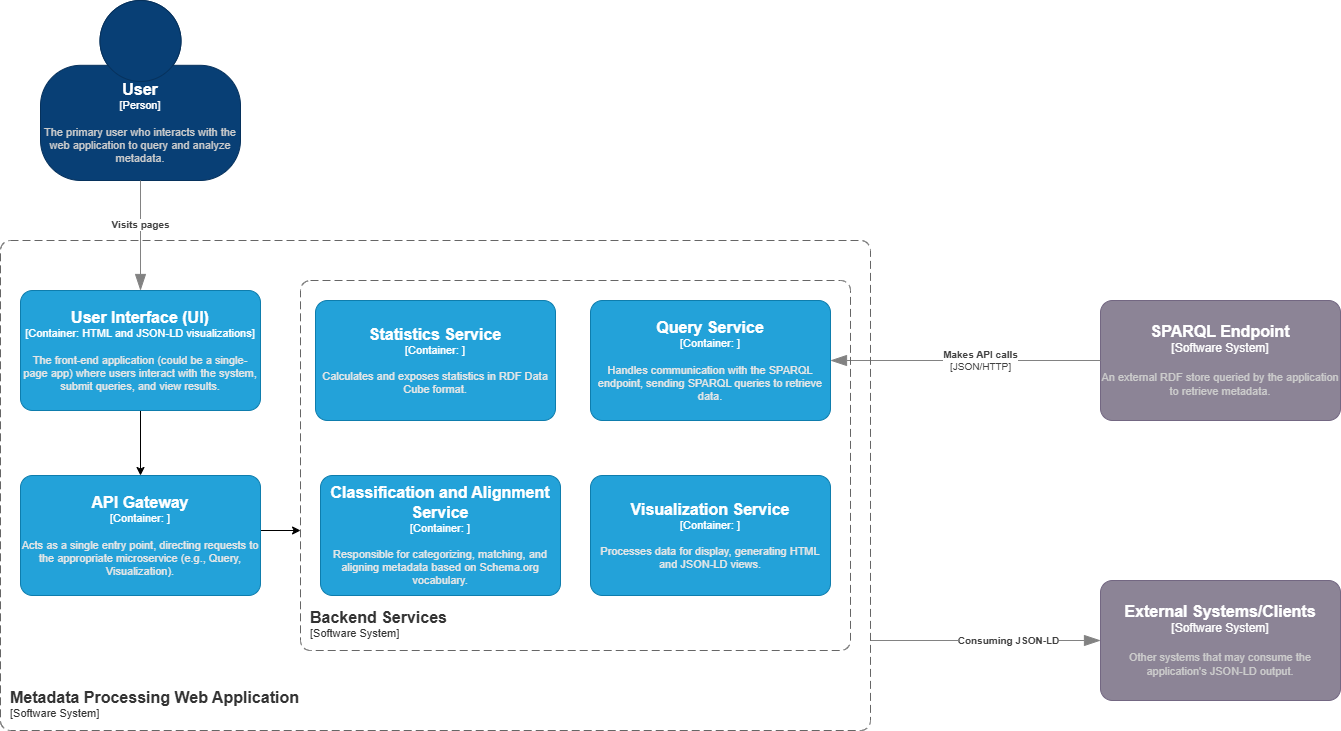
**Classification and Alignment Service**: Responsible for categorizing, matching, and aligning metadata based on Schema.org vocabulary.

**Visualization Service**: Processes data for display, generating HTML and JSON-LD views.

**Statistics Service**: Calculates and exposes statistics in RDF Data Cube format.

**SPARQL Endpoint (External)**: External RDF data source that the Query Service interacts with.

**External Systems (Consumers)**: Systems that may consume JSON-LD output or statistical data.



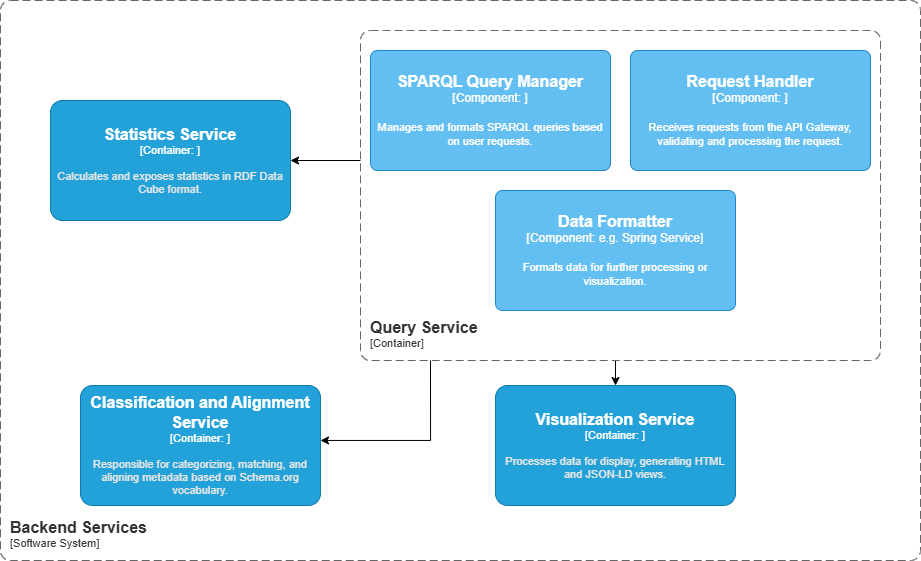
**Component Diagram (C4 Model Level 3)**

The **Component Diagram** further breaks down the **Query Service** as an example, showing how it processes a query request and interacts with the SPARQL endpoint and other services.

**SPARQL Query Manager**: Manages and formats SPARQL queries based on user requests.

**Request Handler:** Receives requests from the API Gateway, validating and processing the request.

**Data Formatter:** Formats data for further processing or visualization.



**User Perspectives**

From an end-user perspective, this application will provide the following functionalities:

* **Querying and Data Retrieval**: Users can query specific metadata via a user-friendly form, with the backend handling complex SPARQL queries.
* **Visualization**: Clear visual outputs for easy analysis, including graphs and charts.
* **Classification**: Metadata items are categorized based on product-specific attributes.
* **Comparison and Alignment**: Users can compare and align data, receiving summaries or visual indicators of similarities and differences.
* **Statistics**: Access to statistical insights about metadata usage and distribution across the dataset.