**RelFinderNG**

**Technical Design Document**

**Version 5.0**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Description | Author |
| March 12,2020 | 0.01 | Initial Draft | Kumara Swamy Y |
| March 12,2020 | 0.02 | Visualization Graph Decision | Kumara Swamy Y |
| May 15,2020 | 0.03 | Technical details updated | Kumara Swamy Y |
| Nov 6, 2020 | 0.04 | Document update | Arun Prasath K |
| Dec 4,2020 | 0.05 | Docker & UI update | Arun Prasath K |
| Dec 8,2020 | 0.06 | Project structure and Code explanation | Arun Prasath K |
| Dec 19,2020 | 0.07 | Proxy server test | Arun Prasath K |

Table of Contents

[Introduction 7](#_Toc59298119)

[Purpose of this document 7](#_Toc59298120)

[Document organization 7](#_Toc59298121)

[Intended Audience 7](#_Toc59298122)

[Scope 8](#_Toc59298123)

[Definitions & acronyms 8](#_Toc59298124)

[Definitions 8](#_Toc59298125)

[Acronyms and abbreviations 8](#_Toc59298126)

[References 9](#_Toc59298127)

[Architecture and Design 9](#_Toc59298128)

[Basic priorities and high-level system architecture 9](#_Toc59298129)

[Deployment Diagram 10](#_Toc59298130)

[Angular + Visualization (D3/Vis) 11](#_Toc59298131)

[Technology Stack 11](#_Toc59298132)

[Visualization library 11](#_Toc59298133)

[Components 12](#_Toc59298134)

[Services 13](#_Toc59298135)

[Models 14](#_Toc59298136)

[Acceptance & Testing scope 14](#_Toc59298137)

[Acceptance 14](#_Toc59298138)

[Testing scope 15](#_Toc59298139)

[Limitations of old RelFinderNG 15](#_Toc59298140)

[User Interface mockups 15](#_Toc59298141)

[Initial mockups 15](#_Toc59298142)

[Search Entities: 16](#_Toc59298143)

[Endpoint Configuration 17](#_Toc59298144)

[Edit Endpoint Configuration 17](#_Toc59298145)

[Edit Endpoint Configuration popup: 18](#_Toc59298146)

[18](#_Toc59298147)

[Add New Endpoint Configuration 18](#_Toc59298148)

[Load Local Configuration file: 19](#_Toc59298149)

[Query Tool 22](#_Toc59298150)

[Proxy Server URL Configurations 23](#_Toc59298151)

[Download Configurations 23](#_Toc59298152)

[Information Menu 24](#_Toc59298153)

[Filter Menu 25](#_Toc59298154)

[Share the Graph result 25](#_Toc59298155)

[26](#_Toc59298156)

[Release Information 26](#_Toc59298157)

[UX Design 26](#_Toc59298158)

[Development Environment Setup 27](#_Toc59298159)

[Software Requirements 27](#_Toc59298160)

[Frontend Requirement 27](#_Toc59298161)

[Backend 27](#_Toc59298162)

[Frontend Code Setup 27](#_Toc59298163)

[Backend Code Setup 28](#_Toc59298164)

[Development Build and Deployment 28](#_Toc59298165)

[Build Frontend application: 28](#_Toc59298166)

[Build Backend application: 30](#_Toc59298167)

[Build Docker Images using Docker for Windows 31](#_Toc59298168)

[System Requirements 31](#_Toc59298169)

[Install Docker Desktop on Windows 31](#_Toc59298170)

[Start Docker Desktop 31](#_Toc59298171)

[Build Dockerimage for Angular application 33](#_Toc59298172)

[Build Dockerimage for Java proxy application 33](#_Toc59298173)

[Test proxy server using PostMan client 34](#_Toc59298174)

[Prerequisite 34](#_Toc59298175)

[Test Proxy Server 34](#_Toc59298176)

[Project Structure of RelfinderNG 36](#_Toc59298177)

[Workspace files 36](#_Toc59298178)

[e2e testing folder 37](#_Toc59298179)

[App source folder and configuration files 37](#_Toc59298180)

[Code walkthrough 39](#_Toc59298181)

[List of Service Classes 39](#_Toc59298182)

[Configuration Service Class 40](#_Toc59298183)

[Load Configuration Service Class 40](#_Toc59298184)

[Filter Process Service 40](#_Toc59298185)

[Query Tool Service 40](#_Toc59298186)

[SparkQLQueryBuilderService 41](#_Toc59298187)

[SparQL Connection Service 41](#_Toc59298188)

[SparQL Result Parser Service 41](#_Toc59298189)

[SparQL Property Service 41](#_Toc59298190)

[Constant Service 41](#_Toc59298191)

[Auto Complete Service 42](#_Toc59298192)

[List of component and Purpose 42](#_Toc59298193)

[Left Panel Component 42](#_Toc59298194)

[Right Panel Component 45](#_Toc59298195)

[Top Panel Component 45](#_Toc59298196)

# Introduction

RelFinderNG is imagined as a web platform which should ease up process of comparing how things are related with each other. It will enable users to discover and compare two or more objects on interactive way through web application. This “one click” app will save time and provide very useful information to interested parties.

## Purpose of this document

The purpose of this document is to expose the Technical design decisions that have been made by team RelFinderNG during the project which is being developed. The content of this document are not immutable: changes can happen during the implementation of the application and so new versions of this document will be produced.

The RelFinderNG helps to get an overview: It extracts and visualizes relationships between given objects in RDF data and makes these relationships interactively explorable. Highlighting and filtering features support visual analysis both on a global and detailed level. The RelFinderNG is based on the open source framework VisJS/D3, easy-to-use and works with any RDF dataset that provides standardized SPARQL access.

## Document organization

The document is organized as follows:

1. **Introduction** describes purpose of document, contents of this guide, intended audience and definitions and acronyms.
2. **Architecture and Design** describes architecture used in developing application and, in addition to that, shows diagrams so reader can understand why are architectural components and design choices taken.
3. **User interface mockups** shows graphical user interface of the web application separated in two chapters: initial mockups and new one.

## Intended Audience

The intended audience is:

* Customer
* Team members
* Supervisors

Document represents the guide that team members of RelFinderNG must use to develop the application. Also, besides team members, this document is for customer and supervisors. Their role is checking progress and read about the decisions the team has taken.

## Scope

Scope of this document is to provide an insight into detailed design of the RelFinderNG

Project. Front end Angular components design and architecture is also explained

## Definitions & acronyms

### Definitions

|  |  |
| --- | --- |
| Keyword | Definitions |
| Angular 8 | Framework which allows to create complex, customizable, modern, responsive and user friendly web applications. |
| D3 | A dynamic, browser based visualization library. The library is designed to be easy to use, to handle large amounts of dynamic data, and to enable manipulation of and interaction with the data. |
| Vis | Same as above |
| SPARQL | SPARQL Protocol and RDF Query Language |

### Acronyms and abbreviations

|  |  |
| --- | --- |
| Acronyms and abbreviations | Definitions |
| RDF | Resource Definition Framework |
| Triples | Triple is the atomic data entity in the RDF data model |
| Node | Node is the fundamental unit of which graphs are formed |
| Edge | An edge (or link) of a network (or graph) is one of the connections between the nodes (or vertices) of the network |
| Frontend | Refers to Angular 8 framework |
| REST API | Application endpoint which exposes all features from backend to frontend. |

### References

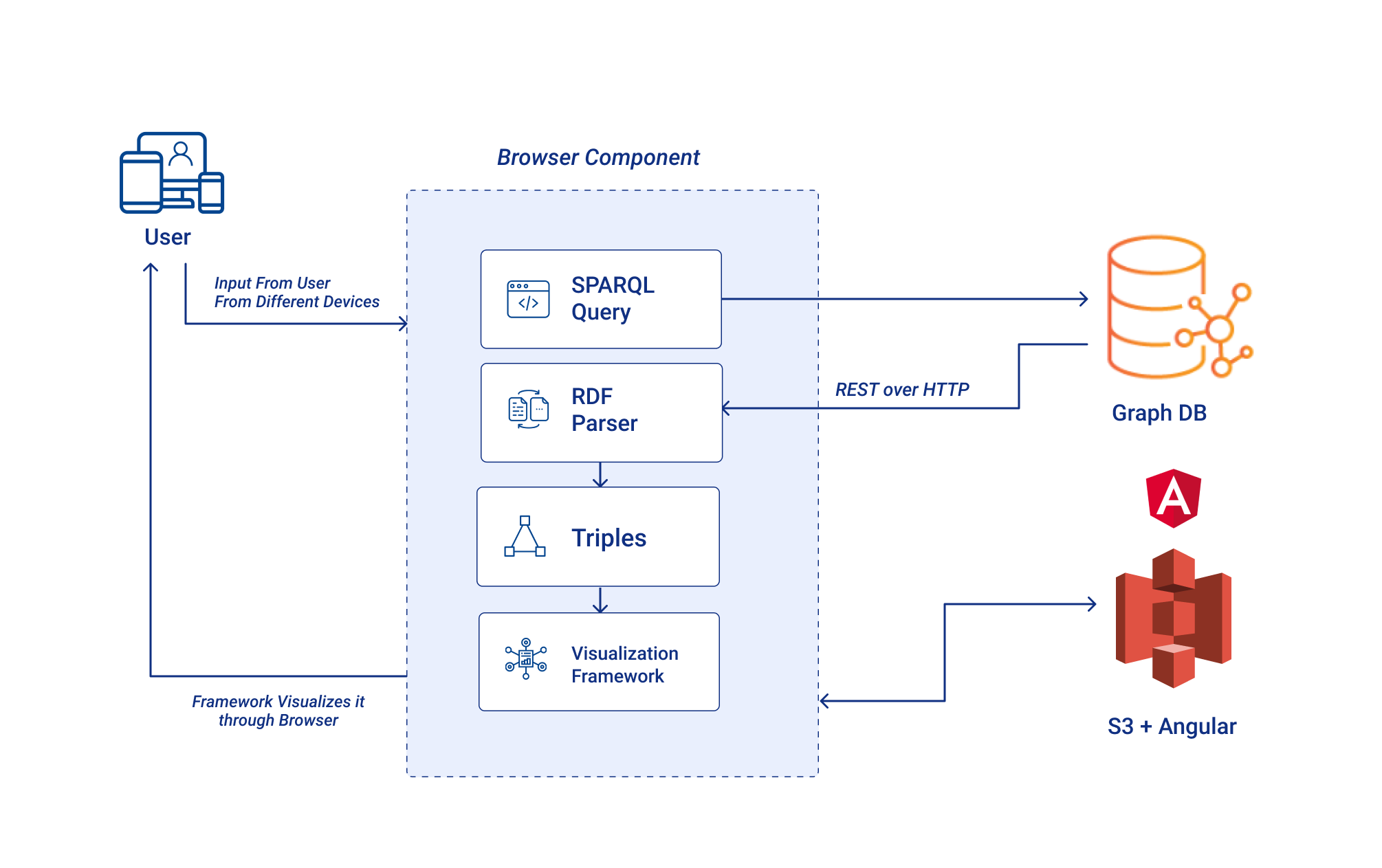
More information about RelFinderNG like more documentation and team members can be found here: [*http://www.visualdataweb.org/RelFinderNG.php*](http://www.visualdataweb.org/relfinder.php)

Project source: [*http://www.visualdataweb.org/RelFinderNG/RelFinderNG.php*](http://www.visualdataweb.org/relfinder/relfinder.php)

# Architecture and Design

## Basic priorities and high-level system architecture

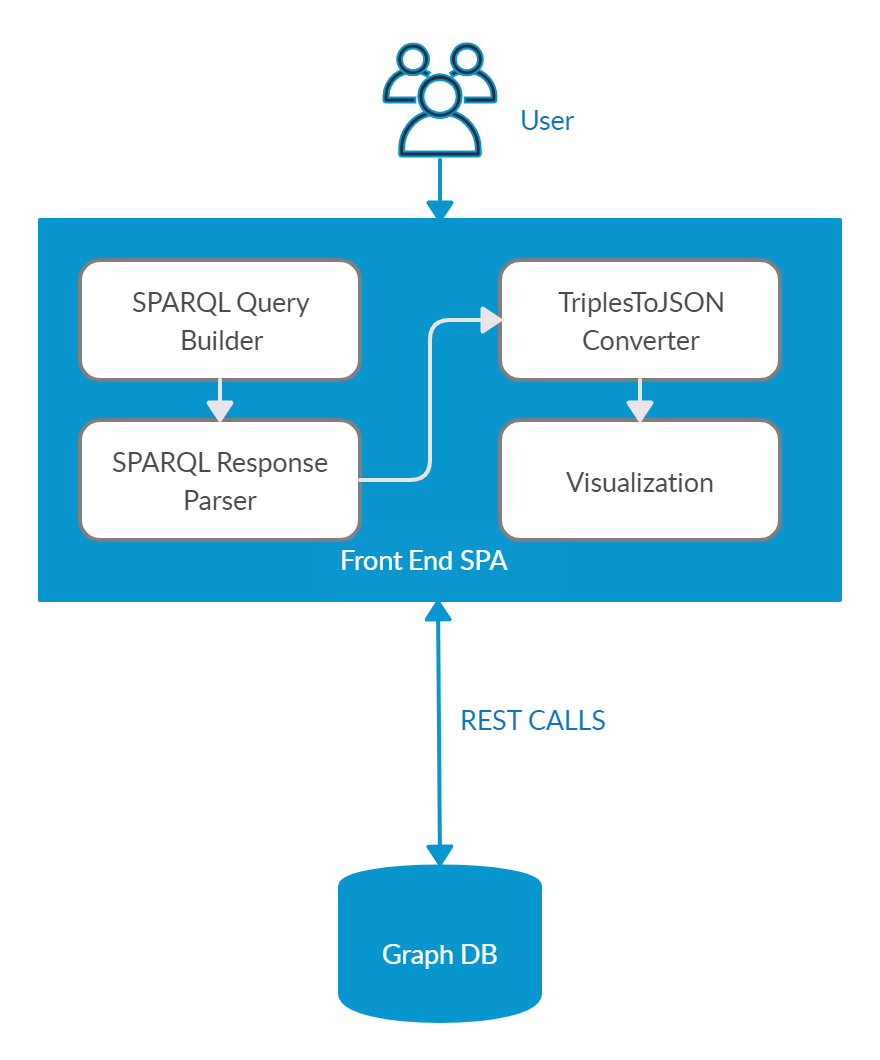
This besides being a good practice in the view of the overall system scalability is also good for the distributed incremental development process.

****

*Illustration 1: System architecture diagram*

## Deployment Diagram

The Deployment Diagram is a static and structured view of computer resources essential to the functionality of the system. It consists of actual devices (other systems of human users), application components running (frontend public and admin single page applications, REST API, comparisons modules and the database and filesystem) and connections between these components (REST calls).



*Illustration 2: Deployment Diagram*

## Angular + Visualization (D3/Vis)

## Technology Stack

|  |  |
| --- | --- |
| **Technology** | **Version** |
| Angular | 8.2.14 |
| Boostrap | 3.4.1 |
| Vis | 4.21 |
| Rxjs | 6.4.0 |

## Visualization library

As per analysis, for network graphing vis JS(vis-network) library is best and team decided to go with vis library. It has below functionality.

* Vis JS have vis-network module separately, it has all the predefined features of network graph
* Node customization
* Edge customization
* Labels customization
* Events
* Localization
* Less code, with predefined functionality
* Allows two way data binding
* Size is large and interactivity is required, you have to go canvas instead of SVG

Repo **:** https://github.com/visjs/vis-network

Docs **:** <https://visjs.github.io/vis-network/docs/network/>

Metrics **:**

Avg Dom load 1 sec + Avg Graph render 250ms = below 2 sec

Finish: 1.07 s | DOMContentLoaded: 802 ms | Load: 1.06 s

Finish: 993 ms | DOMContentLoaded: 740 ms | Load: 985 ms

Finish: 1.06 s | DOMContentLoaded: 730 ms | Load: 1.05 s

**Below is D3 Pros & Cons**

Pros:

* D3 works with web standards like HTML, CSS and SVG, there is no new learning or debugging tool required to work on D3.
* D3 does not provide any specific feature, so it gives you complete control over your visualization to customize it the way you want.
* Size is less then we can go with SVG.
* Scalability it won't pixelate because its vector

Cons:

* Allows two way data binding
* D3 does not provide any specific feature, so it gives you complete control over your visualization to customize it the way you want. So more code
* https://www.quora.com/Why-isnt-D3-js-more-widely-used
* D3 doesn't always try to support older browsers.
* Size is large and interactivity is required, you have to go canvas instead of SVG
* D3 cannot easily conceal original data. If you're using data that you don't want shared, it can be challenging to use D3.

Metrics:

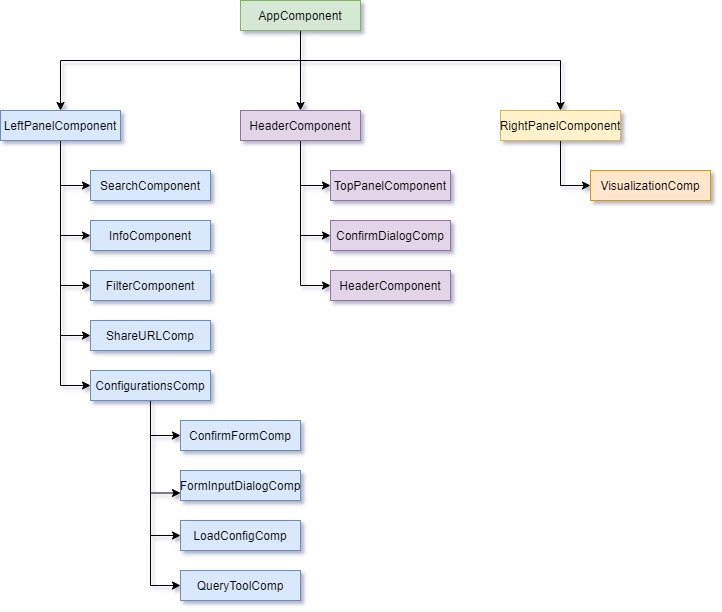
Avg Dom load 1.15 sec + Avg Graph render 500ms = below 2 sec

Finish: 1.83 s | DOMContentLoaded: 1.32 s | Load: 1.74 s

Finish: 1.54 s | DOMContentLoaded: 1.05 s | Load: 1.51 s

### Components

The following figure shows the realization of our frontend SPA. Components are the most basic building block of an UI in an Angular application. An Angular application is a tree of Angular components. A component controls a patch of screen called a view. RelFinderNG modules will contain multiple components. If component is reusable, for example list of faculties, it will go to SharedModules, and will be reused throughout the whole application. Components we plan to have (as shown on illustration 3) are AppComponent, LeftComponent which has 5 children components SearchComponent, ShareURLComponent, ConfigurationComponent, FilterComponent, InfoComponent and RightComponent which has 1 children component ie VisualizationComponent, and ConfigurationComponent has 4 children components ConfirmFormComponent, FormInputComponent, LoadConfigComponent, QueryToolComponent and RightComponent which has 3 children components TopPanelComponent, ConfirmDialogComponent, HeaderComponent.

**

*Illustration 3: Component Diagram*

### Services

Components should focus on presenting data and not fetch or save data directly. For those tasks we use services. Services are a great way to share information among classes. RelFinderNG web application will have services which get data from API connected to database. Services used in application are:

* ConfigurationService : To handle DB source configurations
* LoadConfigService : To load DB source configuration
* FilterProcessService : To handle filters functionality
* QueryToolService: To handle Query tool functionality
* SparQLConnectionService: To prepare the data for graph visualization
* SparkQLQueryBuilderService : To prepare the queries based on depth & inputs
* SparQLResultParserService : To pars the RDF/XML content
* SparQLPropertyService : To fetch node/resource data
* ConstantService : To maintain constant data across application
* QueryExecutionService: To execute given queries
* AutoCompletService : To autofill input data

### Models

Models and deserialization all our models need to be deserialized when queried from our API. This simply specifies the way the data from our API is mapped into our model classes. This interface will be implemented by our models; deserialize will take care of mapping our data to our object.

* VisModel
  + Node
  + Edge
  + NodeEdgeSet..etc
* ConfigModel
* EndPoint

# Acceptance & Testing scope

### Acceptance

Building the old RelFinderNG in latest technologies. As part of acceptance criteria, below mentioned features are implementing in new RelFinderNG tool.

* User can search with n number of inputs and defined depth configuration. Currently there is no limit for Number of inputs, we restrict the limit.
* User can add & import multiple Data Sources in JSON format, and customize the configuration of existing sources too. And also user can download the source configuration.
* User can execute SPARQL queries in Query tool based on source selection
* User can share the current search through URL.
* User can do filtering and see the results of the filtering in the graph. Below are the built in filters:
* length
* link
* class
* connectivity
* User can see Node specific description & localization
* User can see each relation paths in-between source and destination with coloring

### Testing scope

As per above acceptance criteria, deliver the project with below testing scope.

* Conduct unit, functional/integration testing ( Jasmine test framework with Karma tool)
* End to End testing (protractor end to end framework).
* Performance test with more number of nodes.

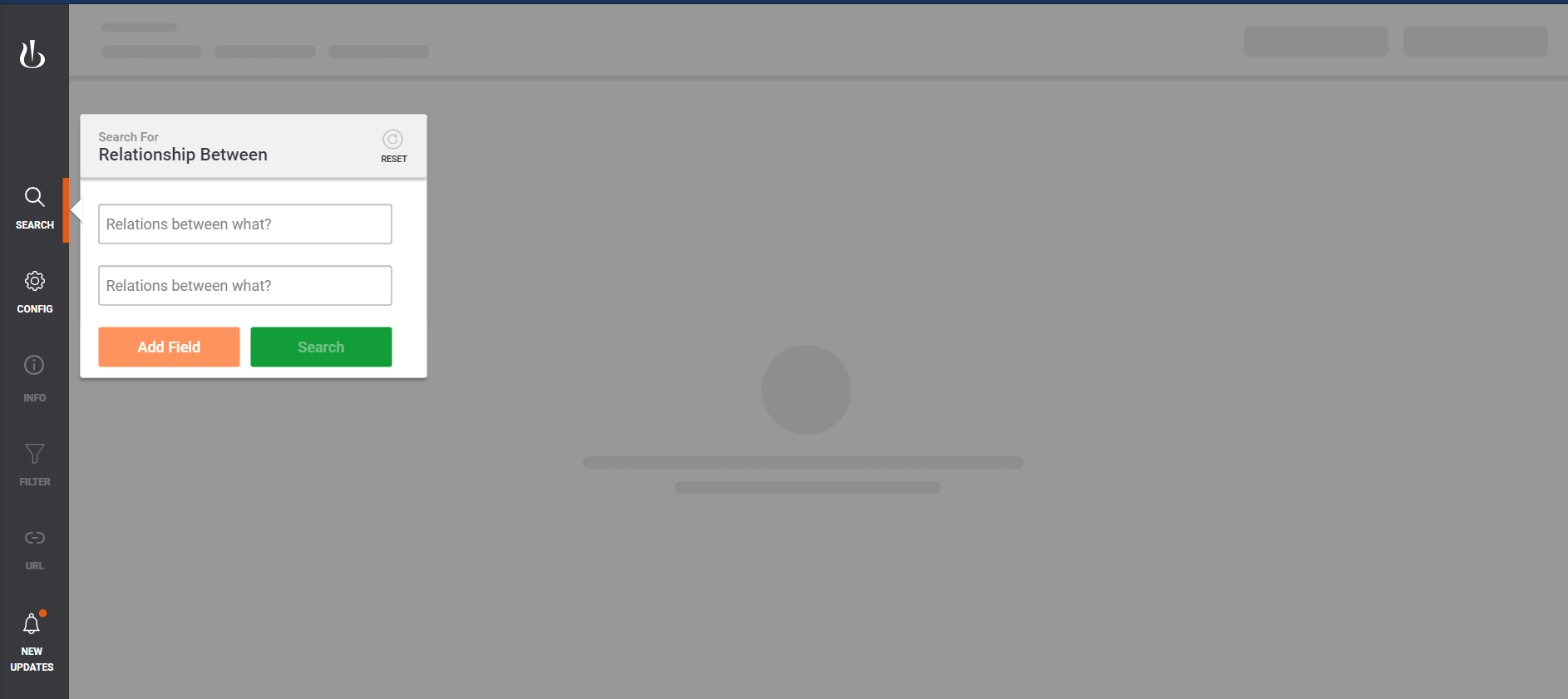
### Limitations of old RelFinderNG

Additionally identified below bugs in old RelFinderNG and those got resolved in new RelFinderNG tool

* Triples ordering shows wrong in graph ( with middle queries result set)
* Connectivity logic wrong
* length filter data not highlighting
* Data showing different from search to search with same inputs.

# User Interface mockups

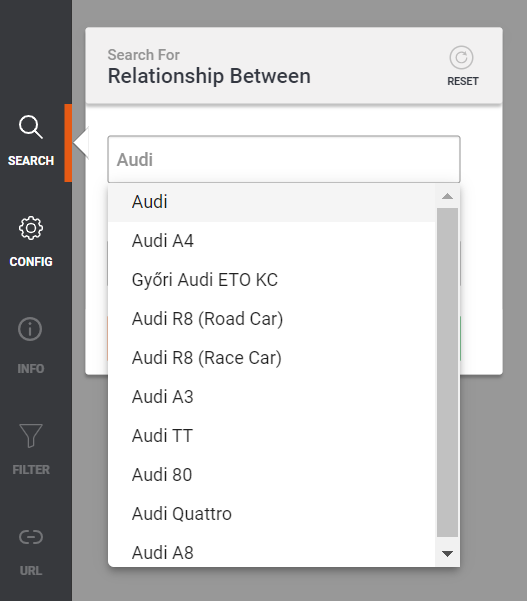
## Initial mockups

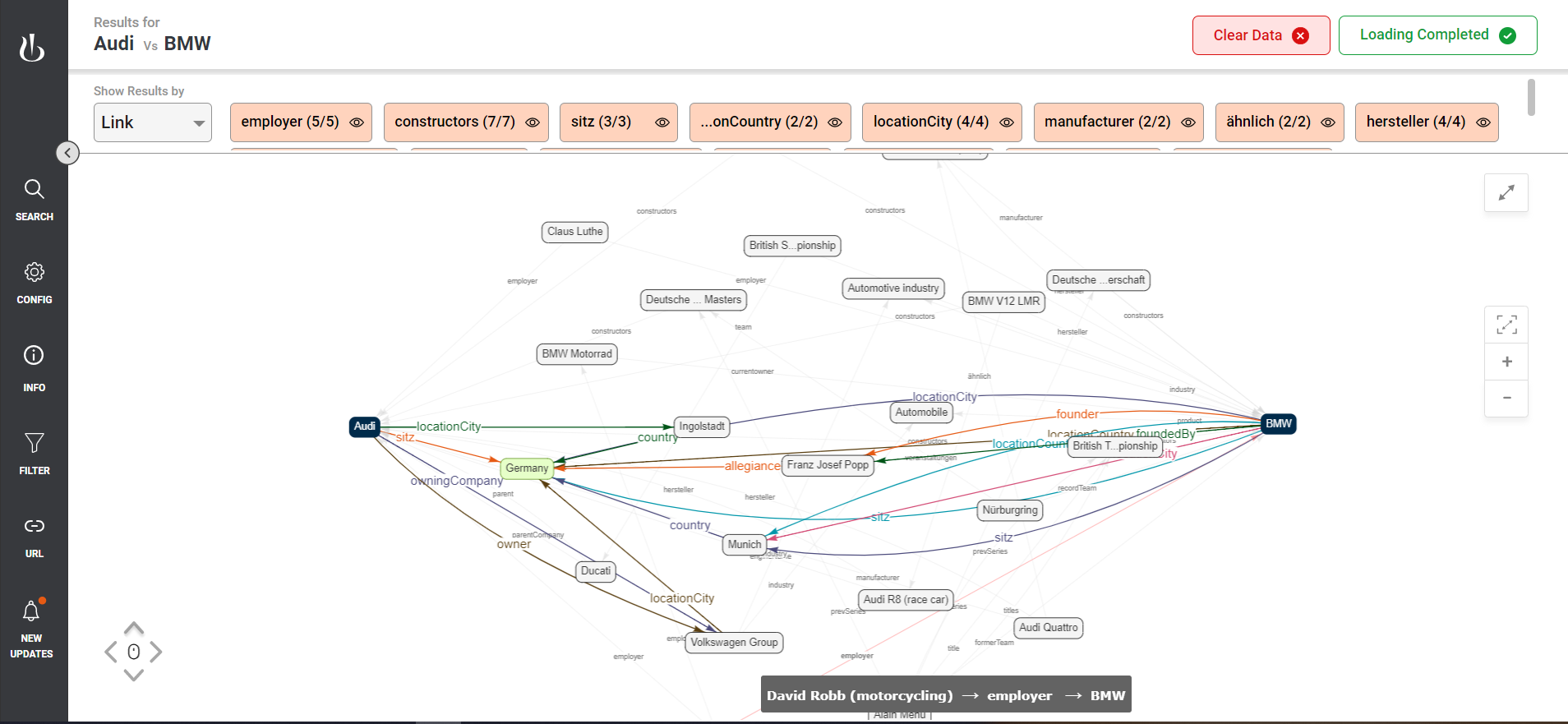


## Search Entities:

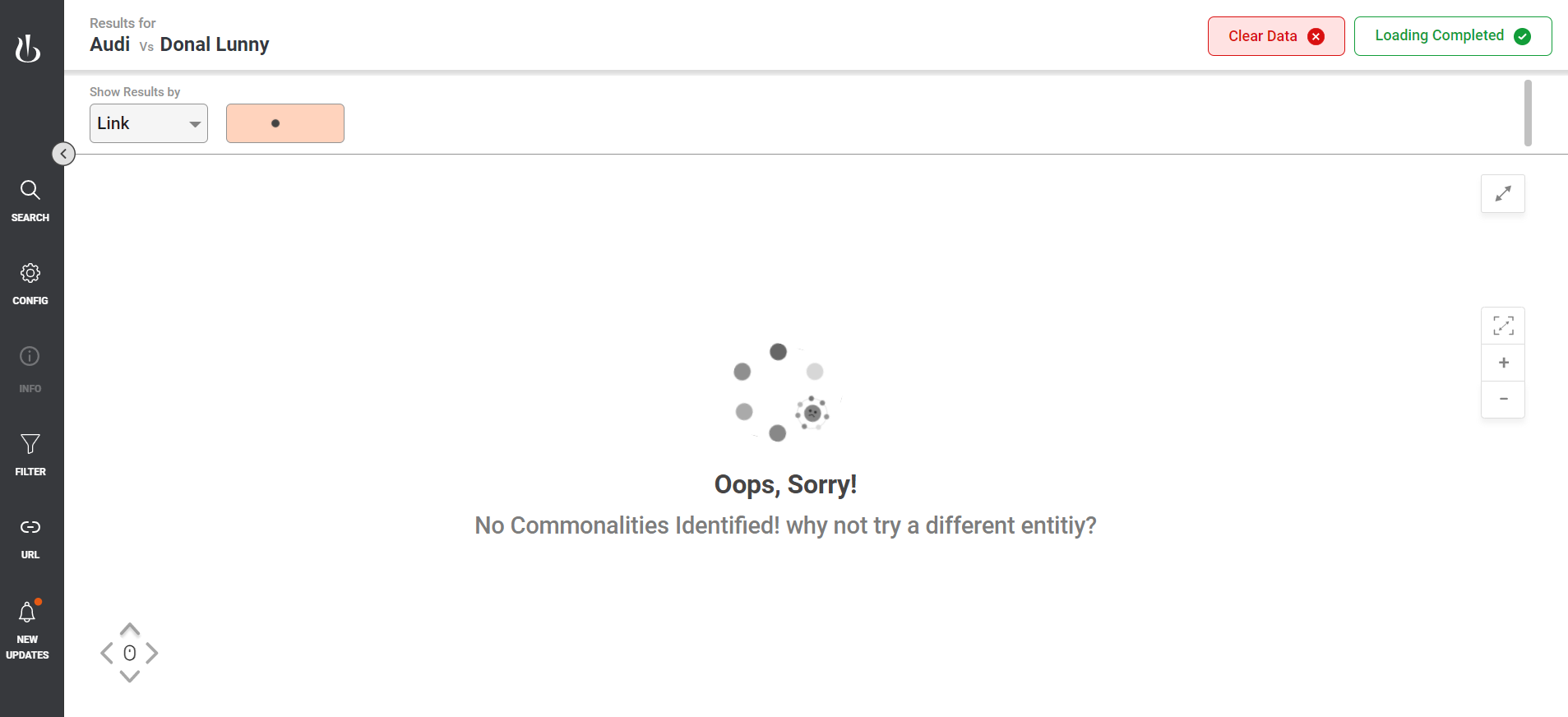
Click “SEARCH” menu in left side menu bar. In input field type the entity name. The entity list is listed in dropdown menu from selected endpoint. Select one option from the list.

Provide at least 2 entities in text field and click search button.



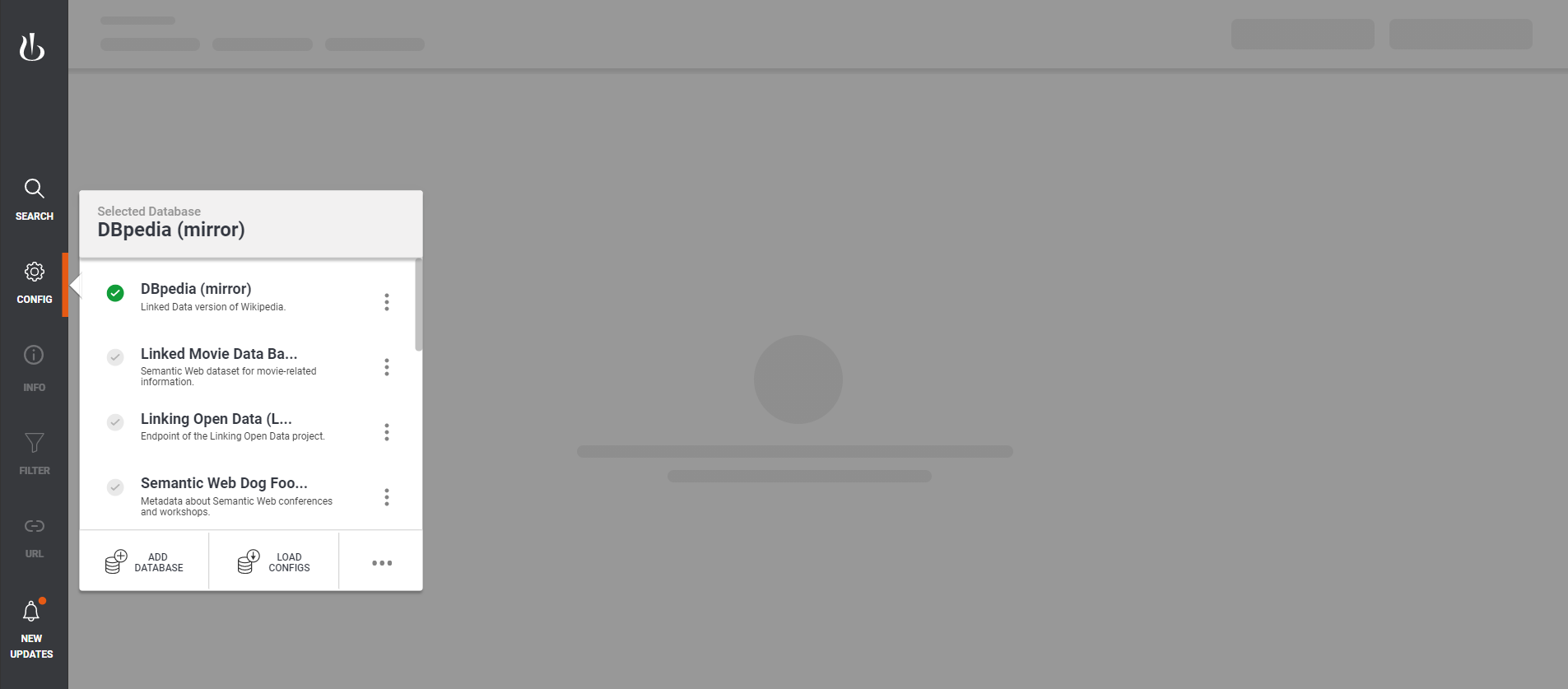


If no commonalities found between the searched entities below error message is shown in the graph panel



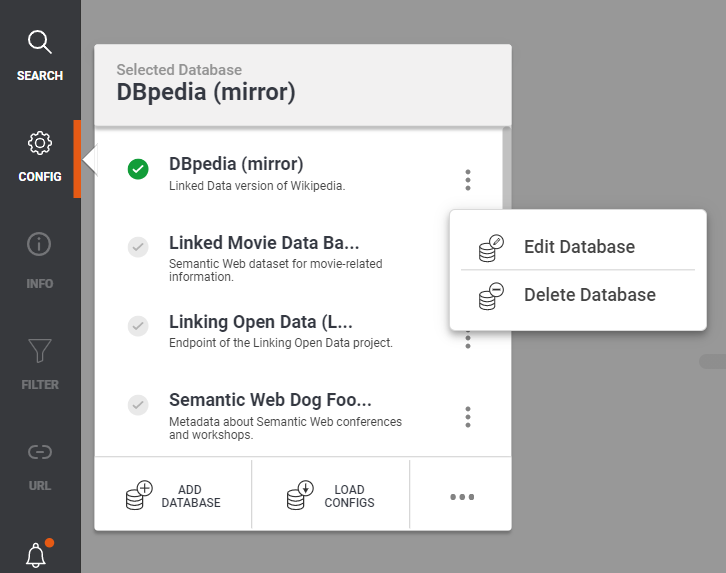
### Endpoint Configuration

Click “CONFIG” menu to view the endpoint configurations. In the popup the endpoint database is listed as shown in below picture.



## Edit Endpoint Configuration

Click on 3 dot icon next to the endpoint. Click “Edit Database” menu to edit the database. Click “Delete Database” menu remove endpoint from the list.



## Edit Endpoint Configuration popup:

Edit the details and Click “Save Data” button to save the Edit Database popup.

## 

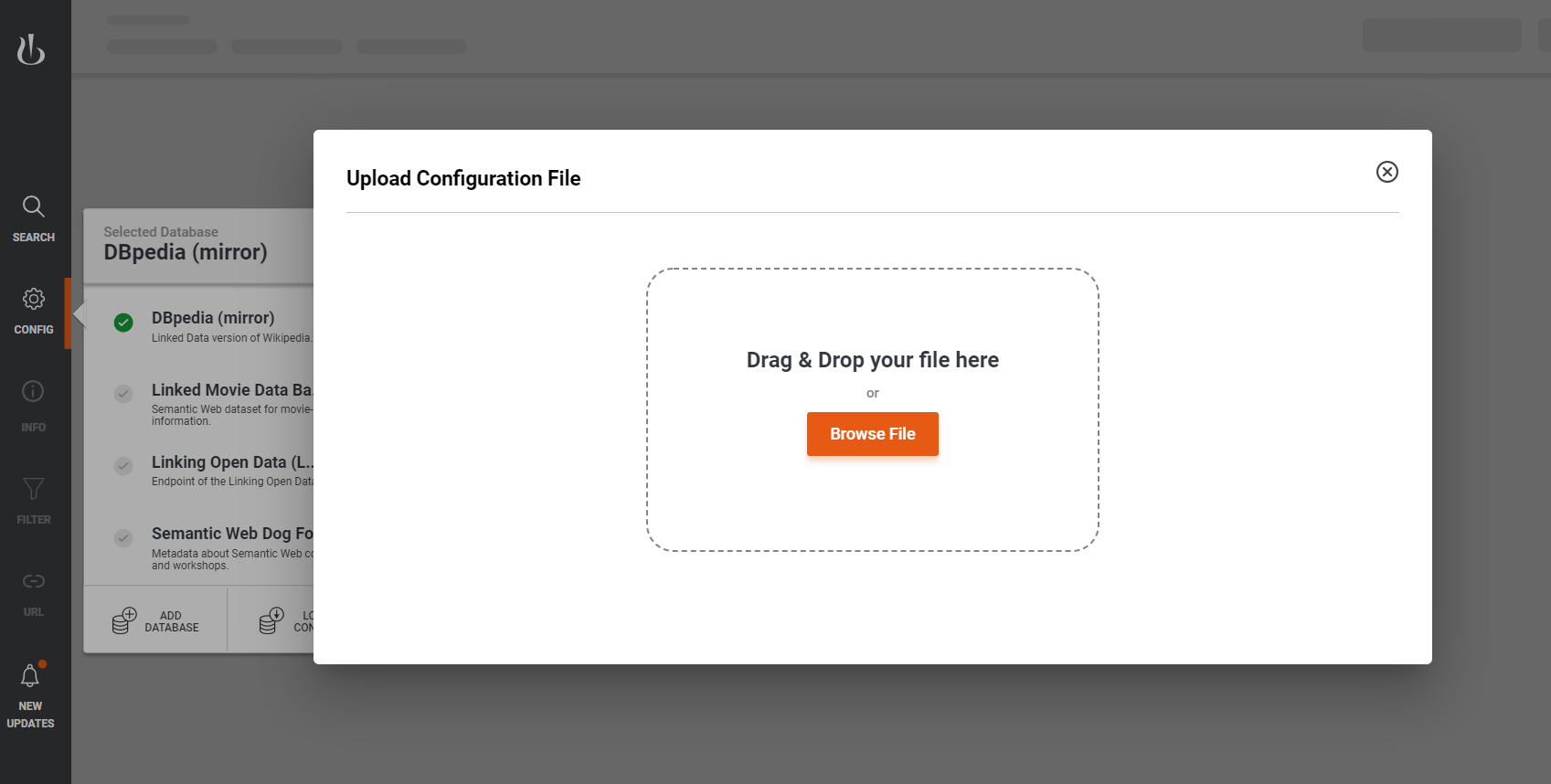
## Add New Endpoint Configuration

In configuration popup click on “ADD DATABASE” menu and empty configuration popup will be shown. In the popup provide the new endpoint details and save it. Newly saved endpoint is listed at last of existing endpoint list.

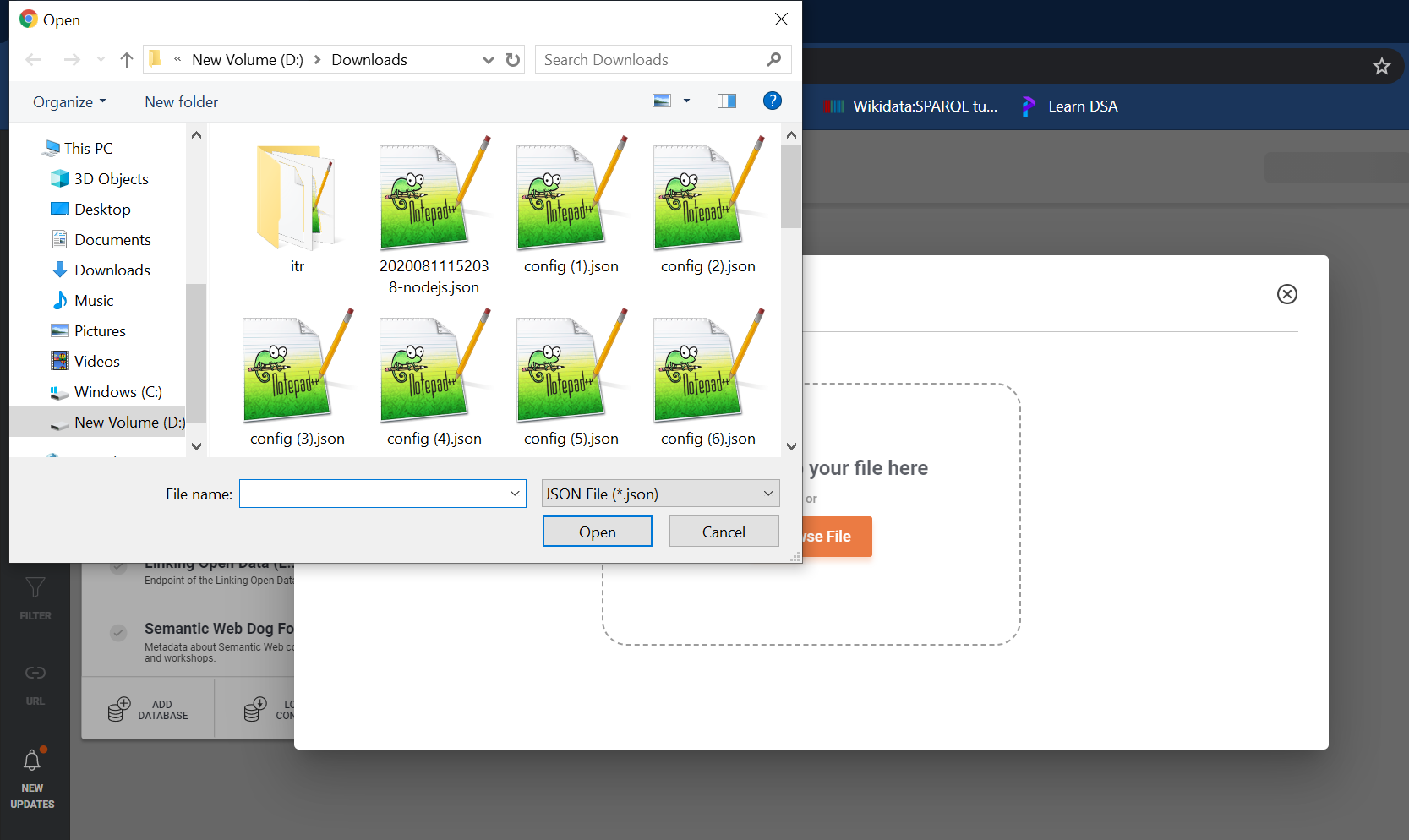


## Load Local Configuration file:

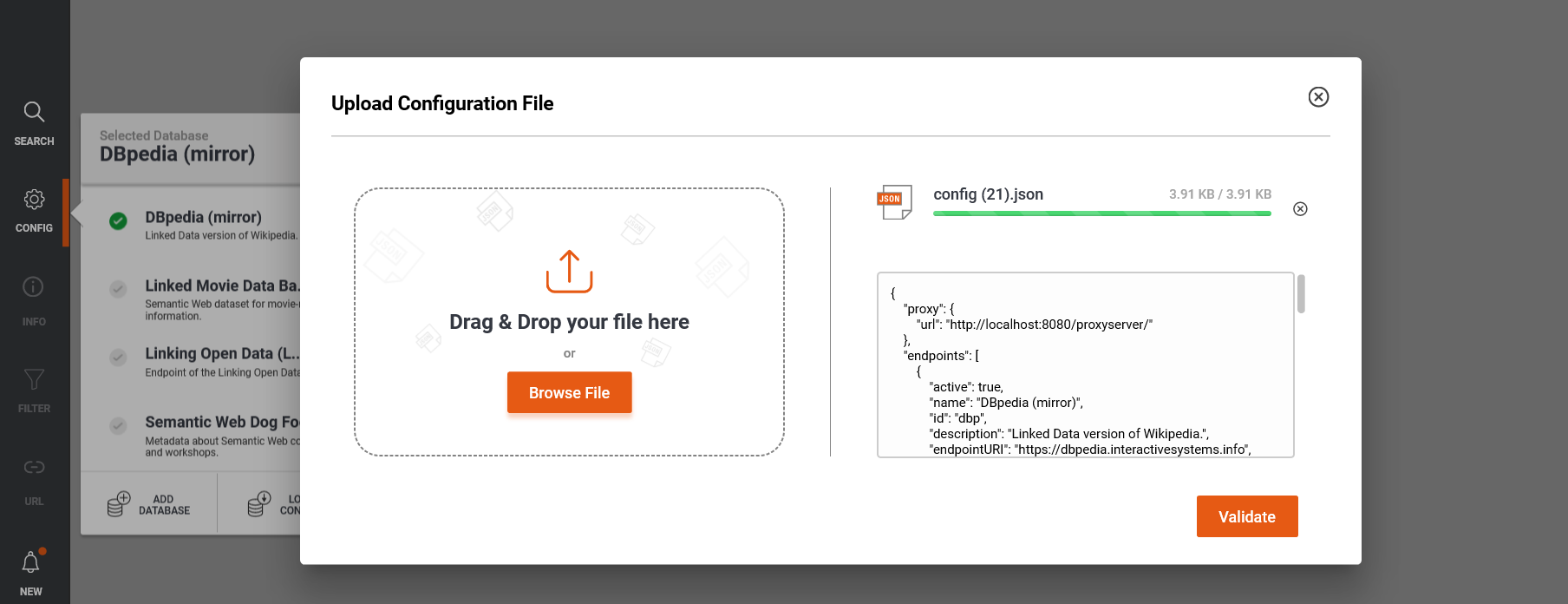
Click “LOAD CONFIGS” menu to upload existing endpoint configuration from local system. It will open below load configuration Popup. From there you can upload the local configuration.



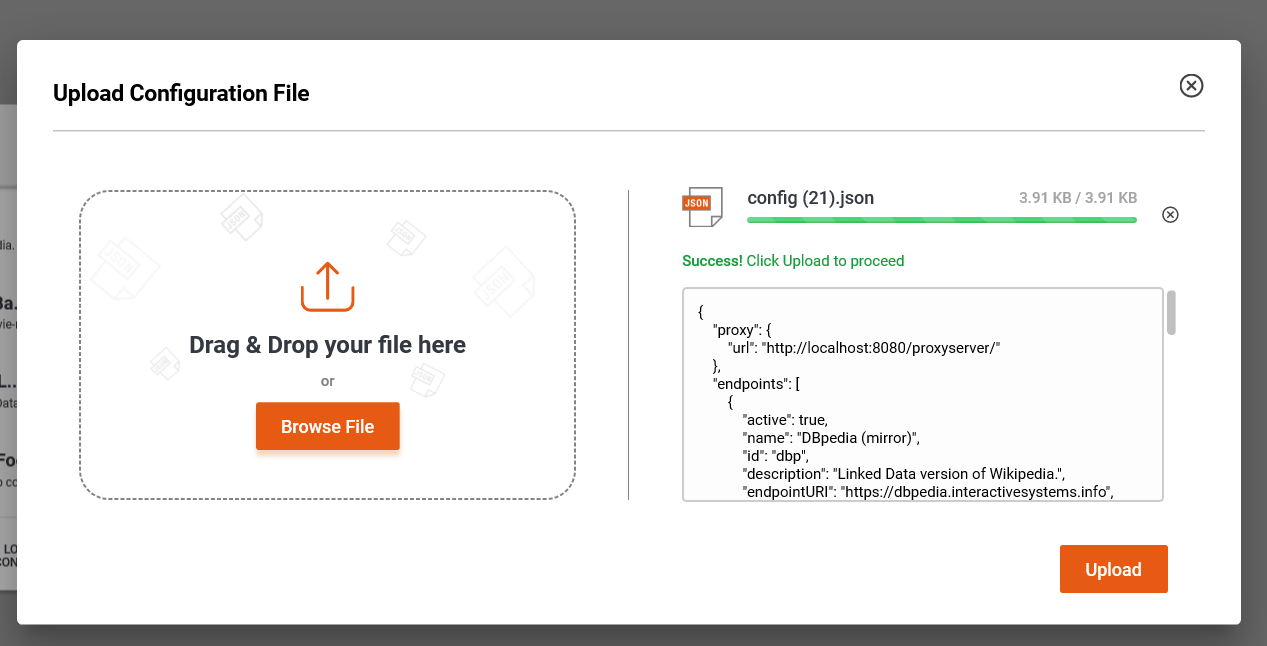
Click “Browse File” button, it file explorer menu select local configuration file with .json extenstion. It will upload the file to application.

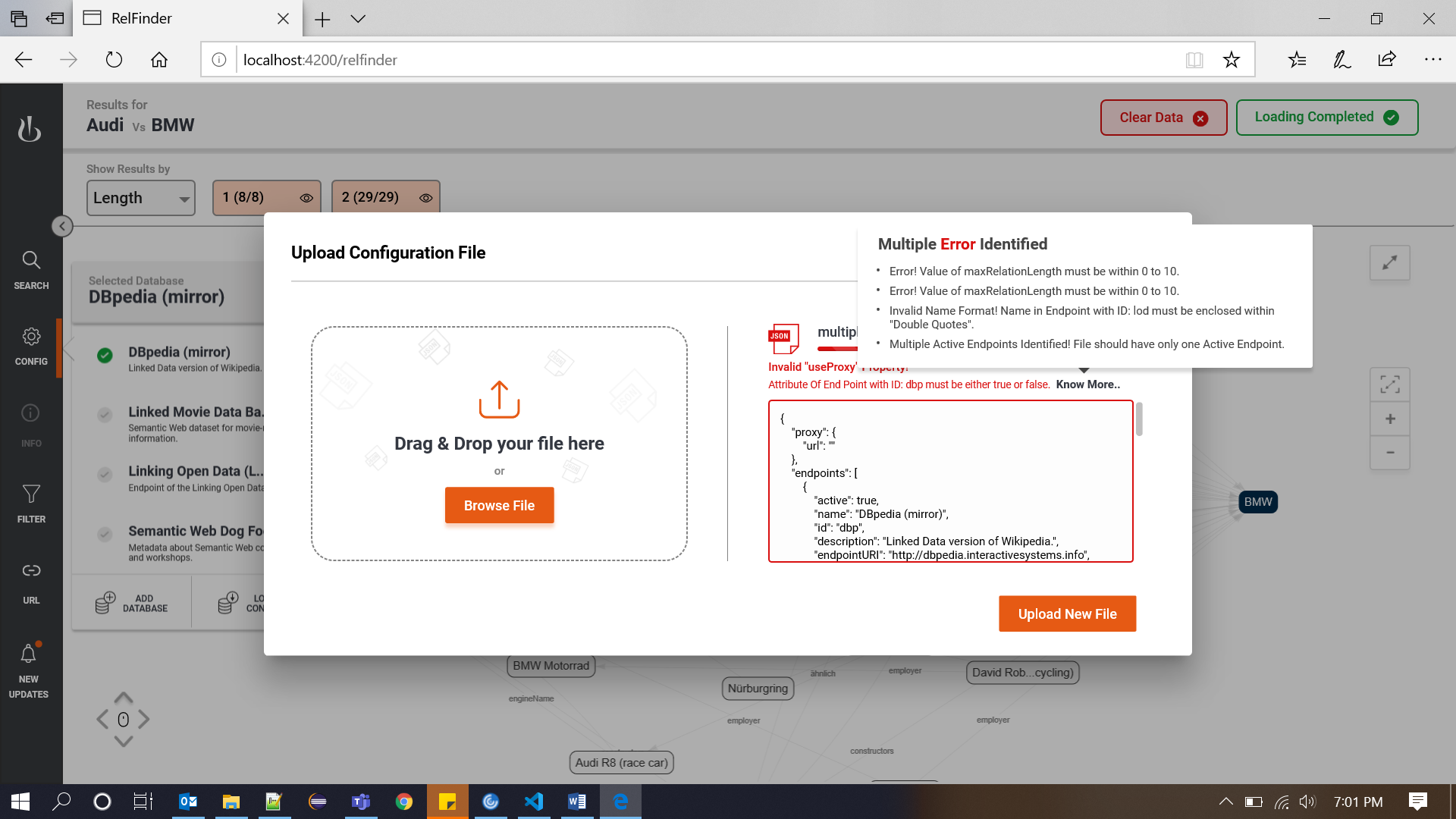


Click “Validate” button at bottom of popup to validate and check uploaded file content.

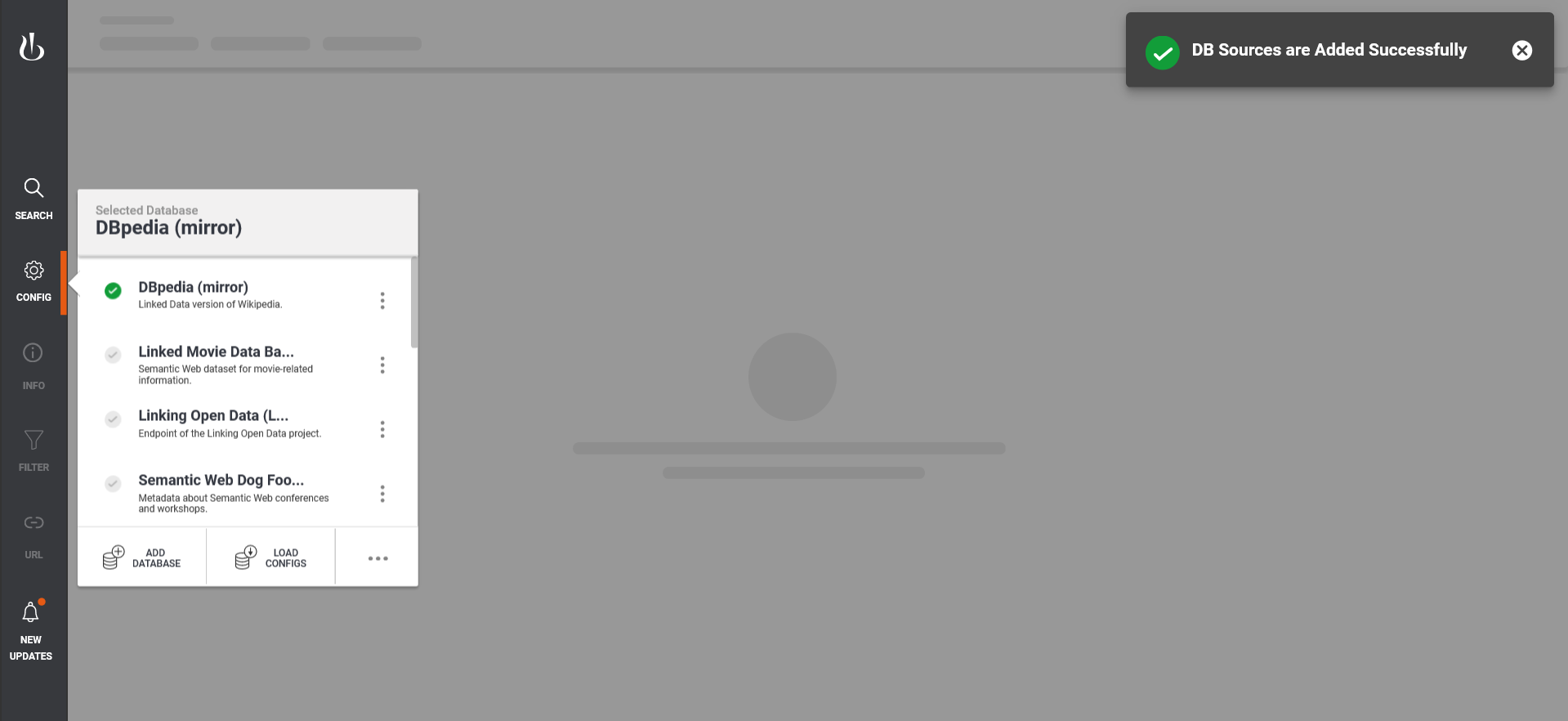


If the file content is valid, success message will show otherwise you will see error messages. You can upload valid file by clicking “Upload New File” button.





Once file is uploaded, upload success notification is shown. In this process it will override existing configuration.

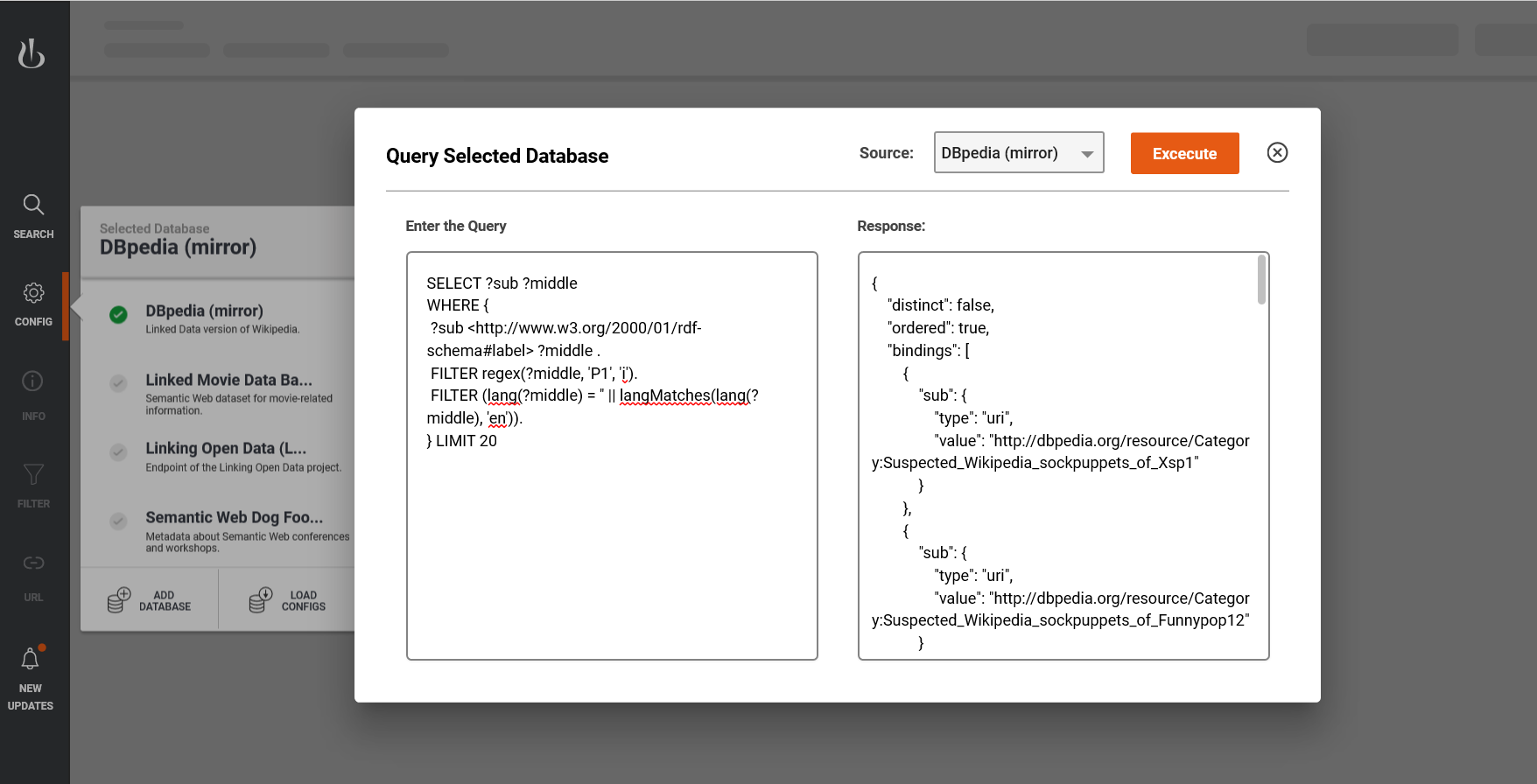


# Query Tool

In “CONFIG” popup, Click bottom 3 dot icon -> Click “Query Tool” Menu.

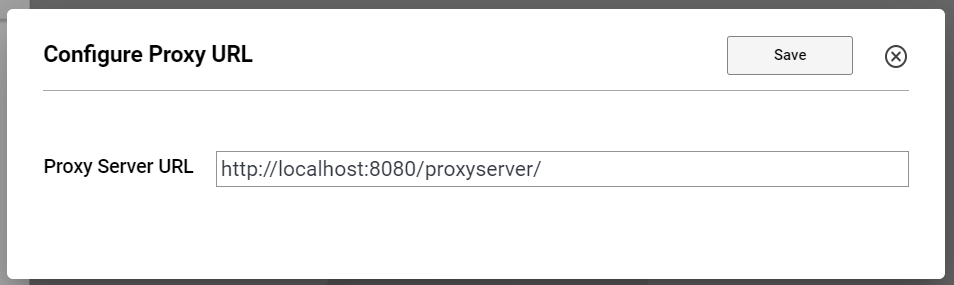


Type query in left side input box and select endpoint from “Source” dropdown list. Then click execute button, the response will be shown in right side box as below image.



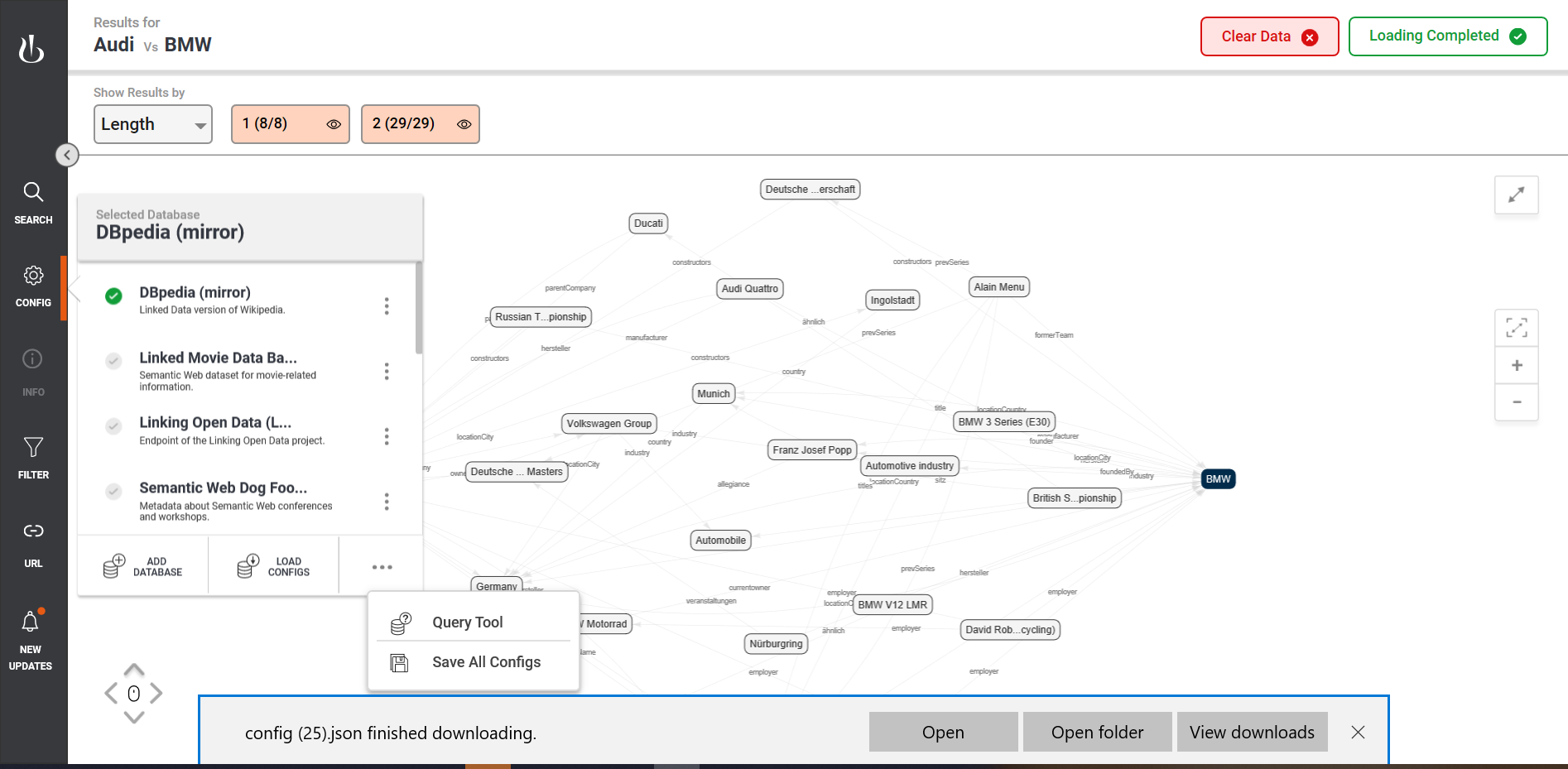
# Proxy Server URL Configurations

Go to CONFIG menu -> click 3 dot icon in menu bottom -> click “Proxy Configs” menu. It will open new modal like below and provide valid URL of proxy server and click Save.



# Download Configurations

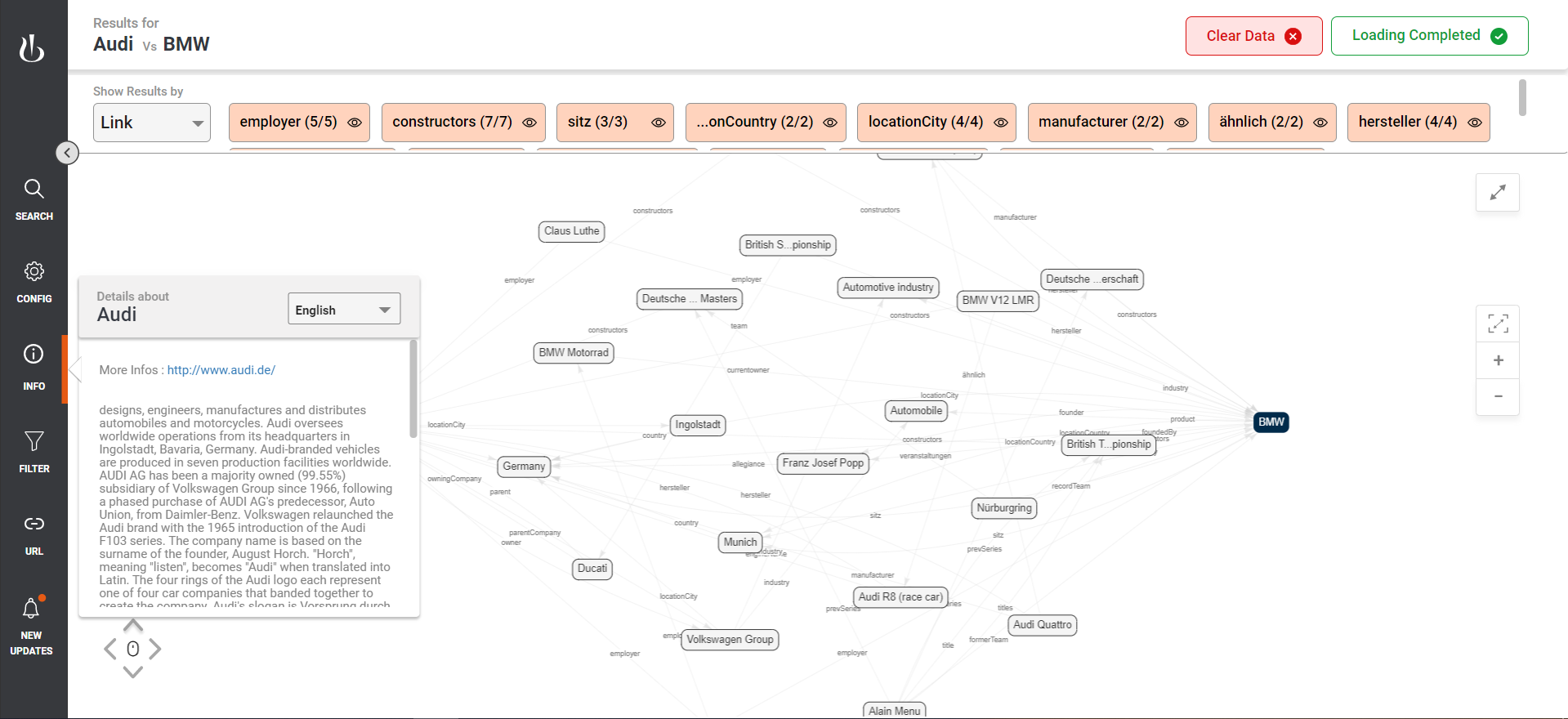
Go to CONFIG menu -> click 3 dot icon in menu bottom -> click “Save All Configs” menu. It will download all configurations.



# Information Menu

Click on any node in loaded graph. Click “INFO” menu to read the information about the selected note. You can select desired language in language dropdown box as shown in below picture.

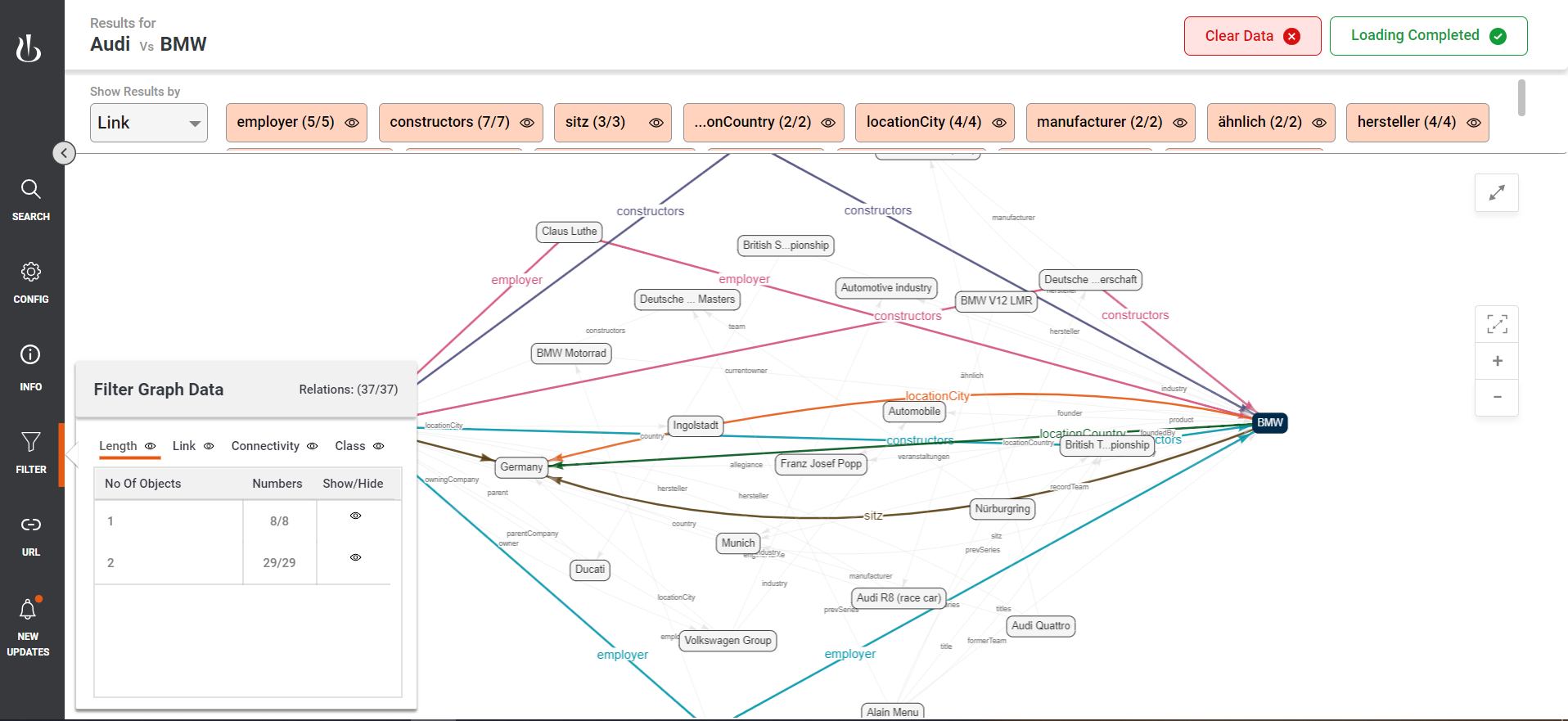
Initially “INFO” menu is disabled. It will only visible only if any one of node is selected.



# Filter Menu

After Graph load, “FILTER” menu is enabled. You can select any one of filter type and select one any cell in table. Selected cell will high light the nodes in graph as shown below

Click Eye icon to hide/visible node in the graph. These action also can be performed in top panel filter.



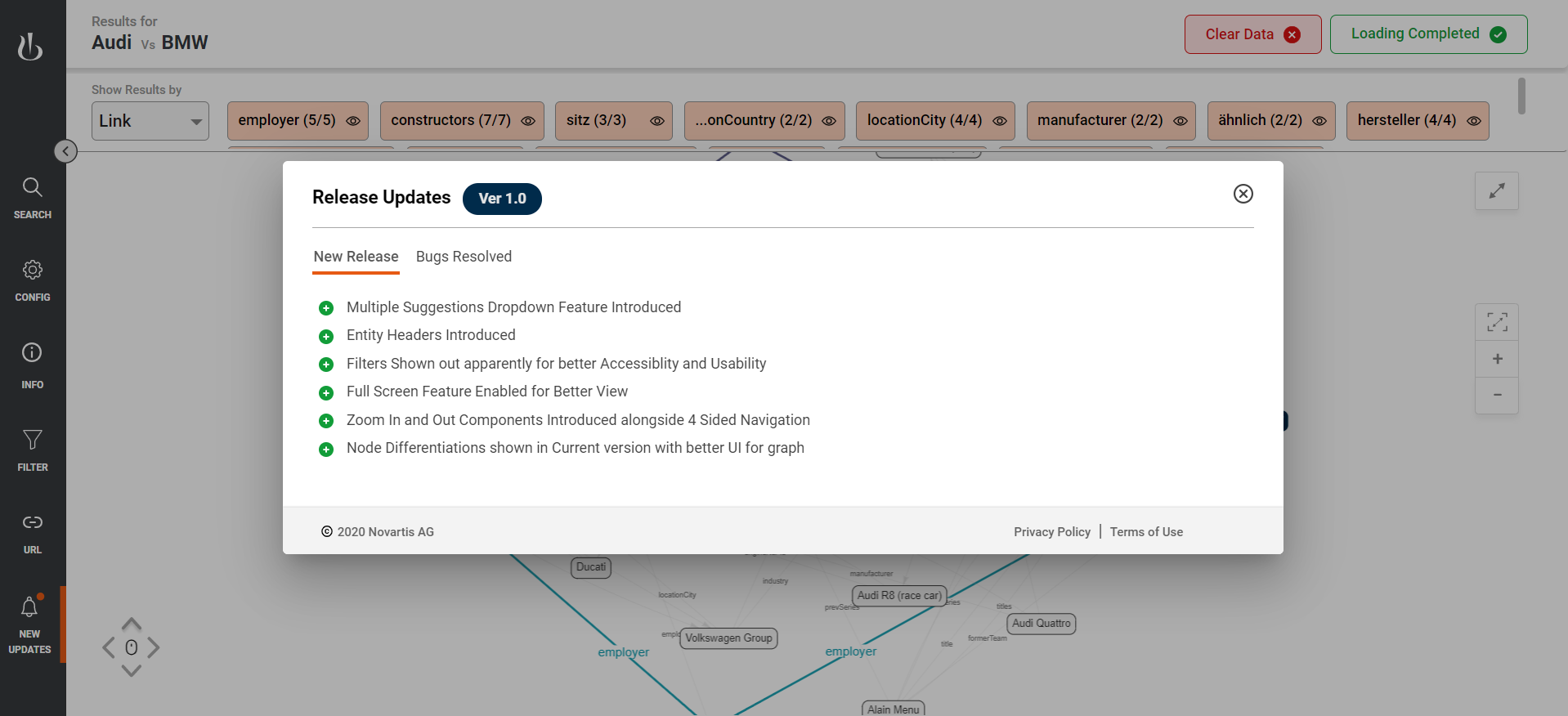
# Share the Graph result

After graph load, “URL” menu is set visible. Select this menu and copy the URL in clipboard and share the URL to another user.

## 

# Release Information

Click “NEW UPDATES” menu in left side menu bar, The release information and bugs resolved list shown in the popup.



## UX Design

<https://www.figma.com/file/sfjq1kJKEiQdw8rAS8Inbe/Altimetrik-Works?node-id=687%3A0>

# Development Environment Setup

## Software Requirements

### Frontend Requirement

1. Node 12
2. Angular 8.3.25
3. VSCode (latest)

### Backend

1. Java 8
2. Maven 3
3. Eclipse IDE (latest)

## Frontend Code Setup

1. Download and Install latest NodeJS and verify node installation by executing below command.

node –v

1. Install Angular 8.3 in local machine by executing below command

*npm install –g @angular/cli@8.3.25*

1. Clone RelfinderNG repository by executing below command

*git clone <RelfinderNG git URL> -b <branch\_name>*

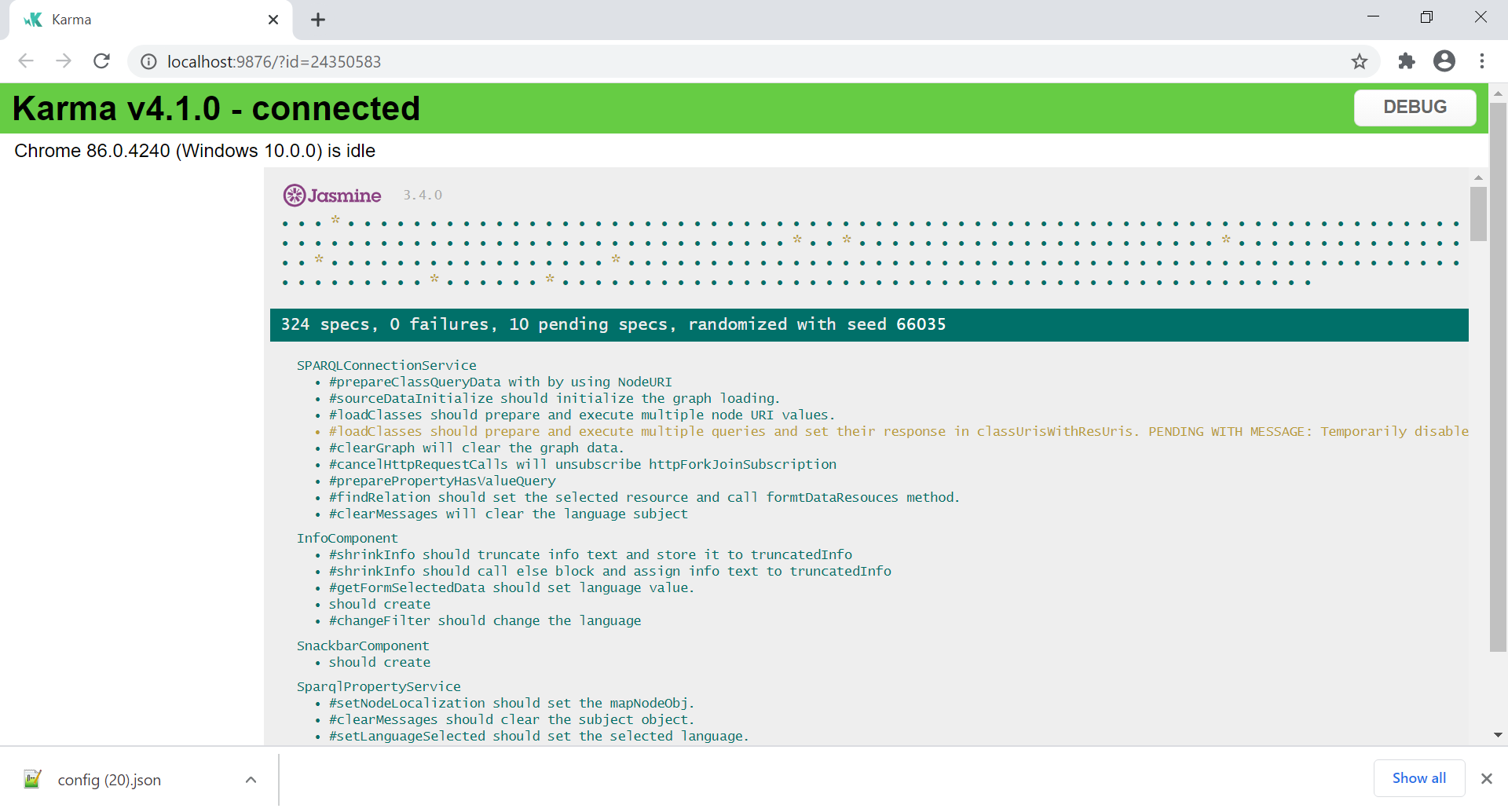
1. Open command prompt and navigate to project folder
2. Execute “**npm install**” command. This command will install node module in project folder
3. Execute “**ng serve -o**” command. This command will compile angular application and opens application in system default browser.

By default application will runs in 4200 port.

1. RelfinderNG application is running and do entity search
2. To test frontend application execute below commad

*ng test –codeCoverage=true*

after successful compilation, node will open browser and executes test cases like below picture

**

1. To Run e2e test cases, execute below command

ng e2e

*This command will open browser and executes e2e test cases*

## Backend Code Setup

1. Install JDK 8 and set “JAVA\_HOME” environment variable
2. Download and unzip maven 3 and set “MAVEN\_HOME” and “M2\_HOME” environment variable
3. Clone Relfinder proxy server application from the repository
4. Import the project in eclipse IDE and right click on project, navigate Run As -> maven build
5. Right click on project and select Run As -> Java application

# Development Build and Deployment

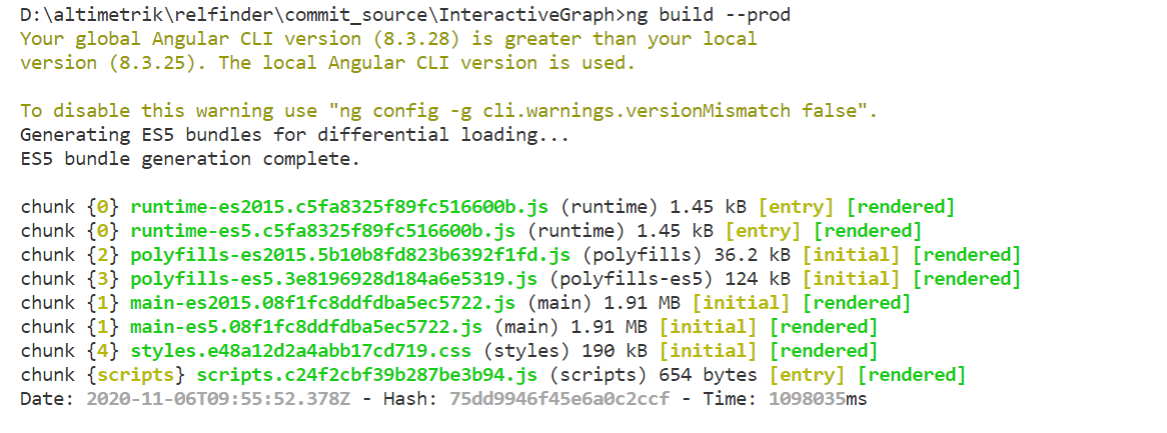
## Build Frontend application:

1. Configure proxy server URL in src/app/config/config.json file



Or you can change the proxy server URL in “proxy config” menu in configuration popup.

1. Open command prompt and navigate to project folder
2. Execute “*npm build --prod*” command
3. After successful build console looks like below picture



1. Copy build folder contend and upload/move the contend to webserver deployment folder
2. Start the server application and go to <http://localhost:4200> in browser

## Build Backend application:

1. Download Maven and add ‘MAVEN\_HOME’ and ‘M2\_HOME’ Environment Variables

Maven can be downloaded from this [**location**](https://maven.apache.org/download.cgi). I have extracted it in location – D:/Latest Setup/apache-maven-3.0.4. You can choose your own location.

1. Set the M2\_HOME and MAVEN\_HOME variable to maven installation folder.
2. Include ‘maven/bin’ directory in ‘PATH’ variable

**To run maven from command prompt, this is necessary. Update the PATH variable with 'Maven-installation/bin' directory.**

1. Verify maven in console

Maven installation is complete. Now lets test it from windows command prompt.

* Go to start menu and type cmd in application location search box.
* Press ENTER. A new command prompt will be opened.
* Type mvn -version in command prompt and hit ENTER.

1. Open command prompt and navigate to proxy server project folder
2. Execute “mvn install” command
3. Copy jar file from <project\_name>/target/<project\_name>.jar file
4. Move the jar file to server and execute “java –jar <project\_name>.jar” command
5. This command runs application in server under https://<domain.com>/proxyserver URL

# Build Docker Images using Docker for Windows

## System Requirements

* Windows 10 64-bit: Pro, Enterprise, or Education (Build 16299 or later).

For Windows 10 Home, see [Install Docker Desktop on Windows Home](https://docs.docker.com/docker-for-windows/install-windows-home/).

* Hyper-V and Containers Windows features must be enabled.
* The following hardware prerequisites are required to successfully run Client Hyper-V on Windows 10:
  + 64 bit processor with [Second Level Address Translation (SLAT)](https://en.wikipedia.org/wiki/Second_Level_Address_Translation)
  + 4GB system RAM
  + BIOS-level hardware virtualization support must be enabled in the BIOS settings. For more information, see [Virtualization](https://docs.docker.com/docker-for-windows/troubleshoot/#virtualization-must-be-enabled).

## Install Docker Desktop on Windows

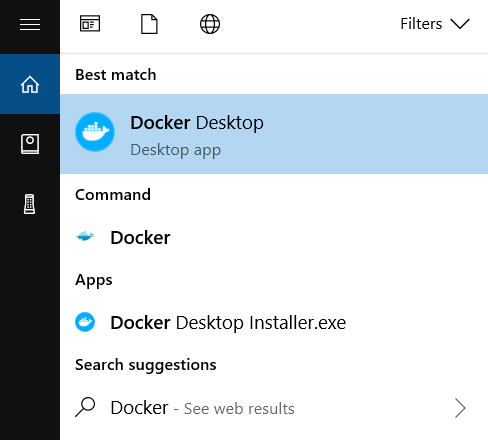
1. Double-click **Docker Desktop Installer.exe** to run the installer.

If you haven’t already downloaded the installer (Docker Desktop Installer.exe), you can get it from [**Docker Hub**](https://hub.docker.com/editions/community/docker-ce-desktop-windows/). It typically downloads to your Downloads folder, or you can run it from the recent downloads bar at the bottom of your web browser.

1. When prompted, ensure the **Enable Hyper-V Windows Features** option is selected on the Configuration page.
2. Follow the instructions on the installation wizard to authorize the installer and proceed with the install.
3. When the installation is successful, click **Close** to complete the installation process.
4. If your admin account is different to your user account, you must add the user to the **docker-users** group. Run **Computer Management** as an administrator and navigate to **Local Users and Groups** > **Groups** > **docker-users**. Right-click to add the user to the group. Log out and log back in for the changes to take effect.

## Start Docker Desktop

Docker Desktop does not start automatically after installation. To start Docker Desktop, search for Docker, and select **Docker Desktop** in the search results.

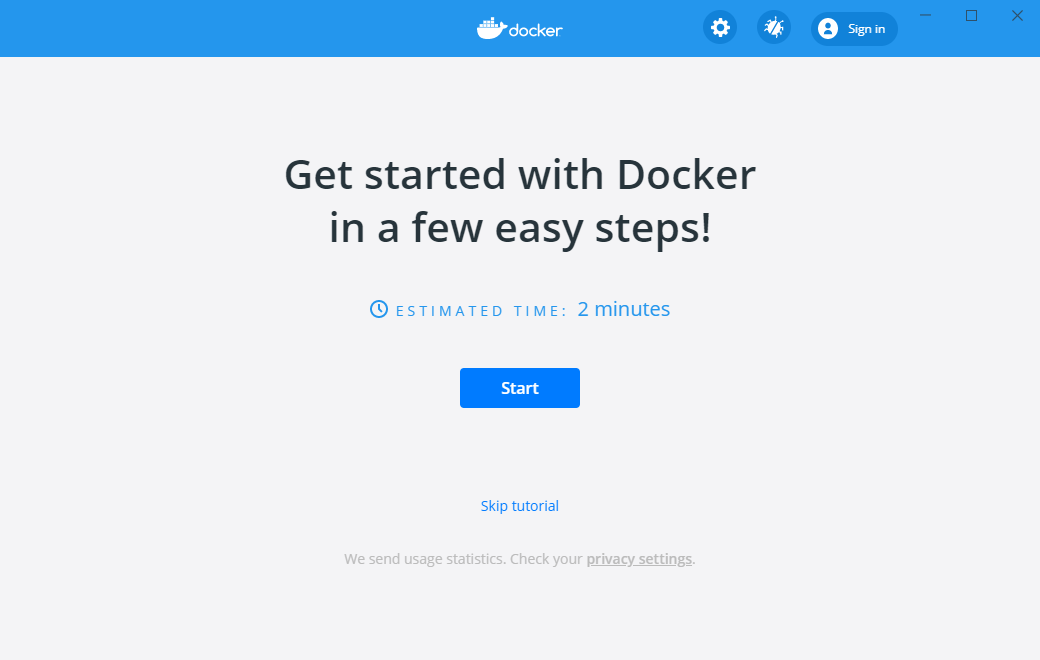


When the whale icon in the status bar stays steady, Docker Desktop is up-and-running, and is accessible from any terminal window.

whale on taskbar

If the whale icon is hidden in the Notifications area, click the up arrow on the taskbar to show it. To learn more, see [Docker Settings](https://docs.docker.com/docker-for-windows/#docker-settings-dialog).

When the initialization is complete, Docker Desktop launches the onboarding tutorial. The tutorial includes a simple exercise to build an example Docker image, run it as a container, push and save the image to Docker Hub.



You are now successfully running Docker Desktop on Windows.

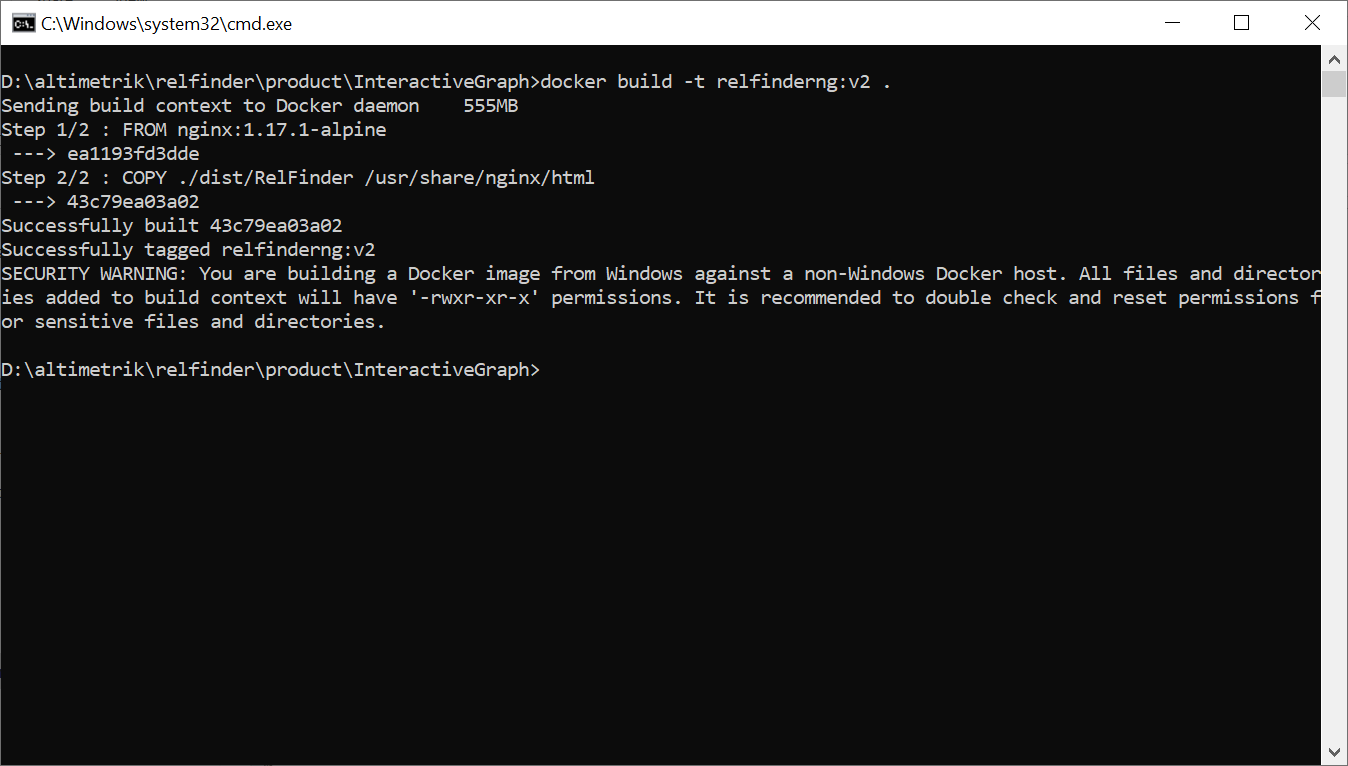
## Build Dockerimage for Angular application

1. Check Docker Desktop is up-and-running
2. Open command prompt and navigate to RelfinderNG project folder
3. Run below command to build Docker image

***docker build –t <image\_name>:<tag> .***

Example: docker build –t relfinderng:v1 .

It will download dependencies mentioned in Dockerfile. When the image is successfully created command look like below



1. Run Docker image by executing below command

**docker run --name relfinder -d -p 4200:80 <image\_name>:<tag>**

Example: docker run --name relfinder -d -p 4200:80 relfinderng:v1

1. Open browser and enter “http://<domain.com>:4200/”. The application will loaded from Dockerimage.

## Build Dockerimage for Java proxy application

1. Check Docker Desktop is up-and-running
2. Open command prompt and navigate to RelfinderNG project folder
3. Run below command to build Docker image

***docker build –t <image\_name>:<tag> .***

Example: docker build –t relfinderproxy:v1 .

It will download dependencies mentioned in Dockerfile. When the image is successfully created image name and id will be displayed in command prompt.

1. Run Docker image by executing below command

**docker run --name relfinderproxy -d -p 8080:8080 <image\_name>:<tag>**

Example: docker run --name relfinderproxy -d -p 8080:8080 relfinderproxy:v1

1. You can access the application by calling “http://<domain.com>:8080/proxyserver” URL in rest client application.

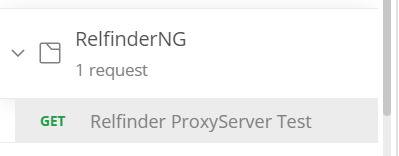
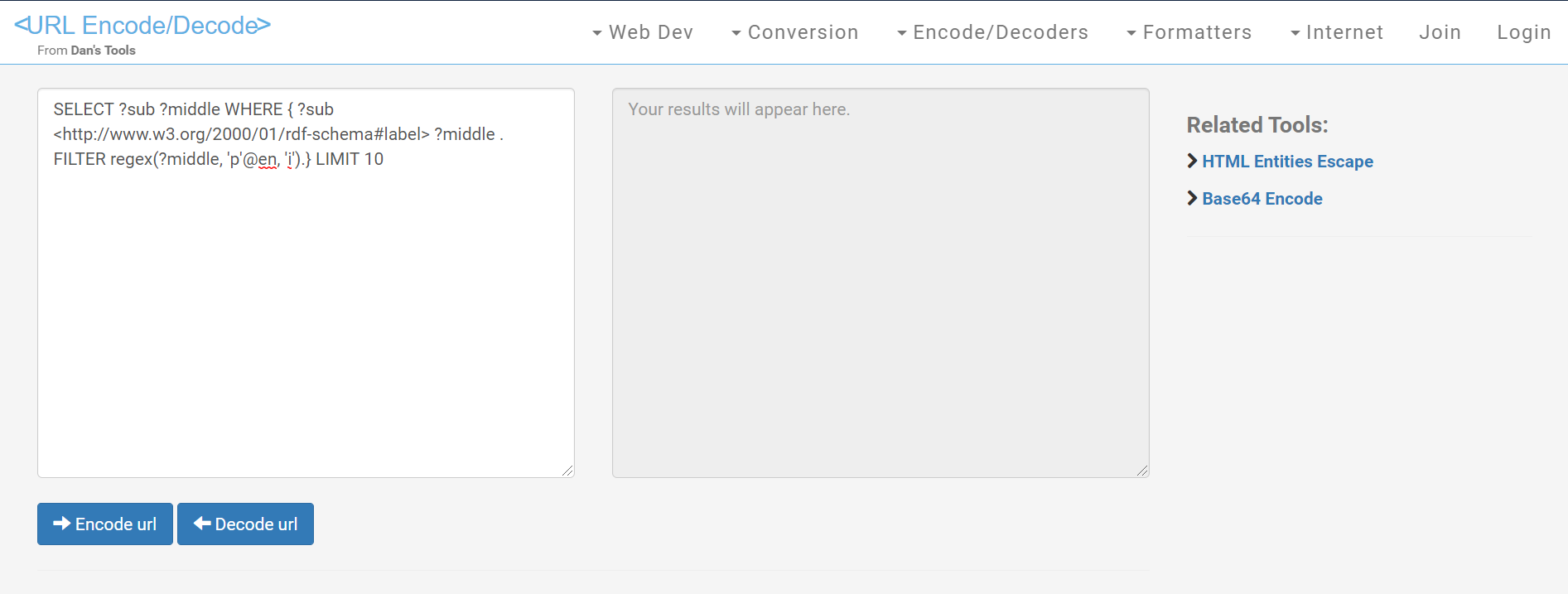
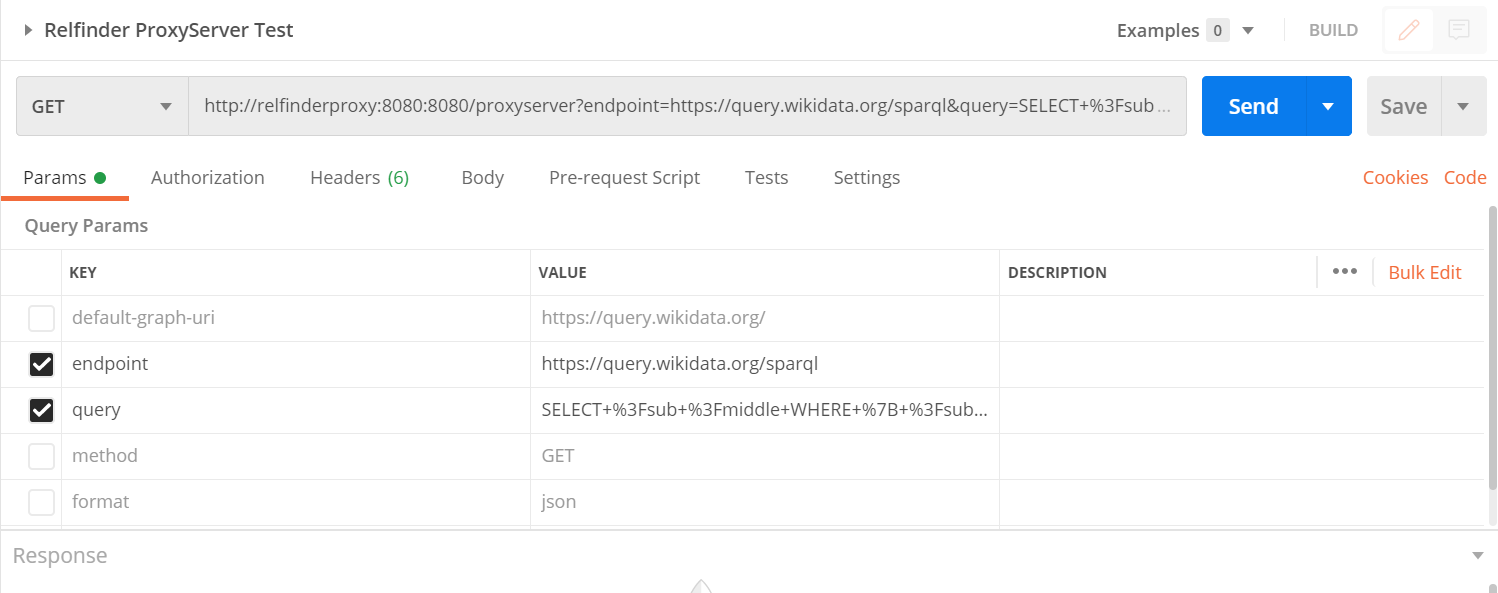
# Test proxy server using PostMan client

## Prerequisite

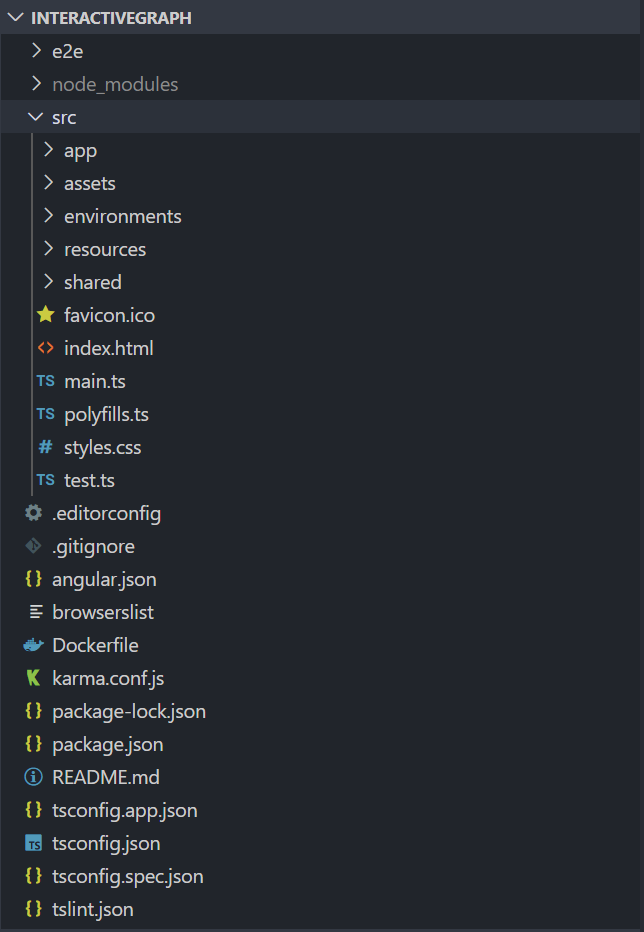
* Visit [Postman](https://www.postman.com/downloads/) website and download the latest version of Postman required for your platform
* Install Postman
* Download RelfinderNG.postman\_collection.json



## Test Proxy Server

* Open Postman application
* Click **Import**, click **Choose Files** and specify RelfinderNG.postman\_collection (file attached in prerequisite). An import success message appears for collection imported
* Expand the collection and select “Relfinder ProxyServer Test” 
* Check/Change proxy server URL in address bar
* Goto [URL encode](https://www.url-encode-decode.com/) site and encode sparql query as shown in below picture and copy encoded text
* Paste copied text in query attribute value in postman 
* Change other attribute based on endpoint requirements
* Click “Send” button and wait for proxy server response. You can see the response in same window

# Project Structure of RelfinderNG



Angular creates the following workspace and starter project files

* An initial skeleton app project (in the src/ subfolder).
* An end-to-end test project (in the e2e/ subfolder).
* Related configuration files.

## Workspace files

The top level of the workspace contains a series of workspace-wide configuration files.

|  |  |
| --- | --- |
| **Configuration Files** | **Purpose of the configuration file** |
| .editorconfig | Configuration for code editors. |
| .gitignore | Specifies intentionally untracked files that Git should ignore. |
| angular.json | CLI configuration defaults for all projects in the workspace, including configuration options for build, serve, and test tools |
| that the CLI uses, such as TSLint, Karma, and Protractor. |
| node\_modules | Provides npm packages to the entire workspace. |
| package.json | Configures and keeps track of npm package dependencies that are available to all projects in the workspace. |
| package-lock.json | Provides version information for all packages installed into node\_modules by the npm client. If you use the yarn client, |
| this file will be yarn.lock instead. |
| tsconfig.json | Default TypeScript configuration for applications in the workspace, including TypeScript and Angular template compiler options. |
| tslint.json | Default TSLint configuration for apps in the workspace. |
| README.md | Introductory documentation for the application |

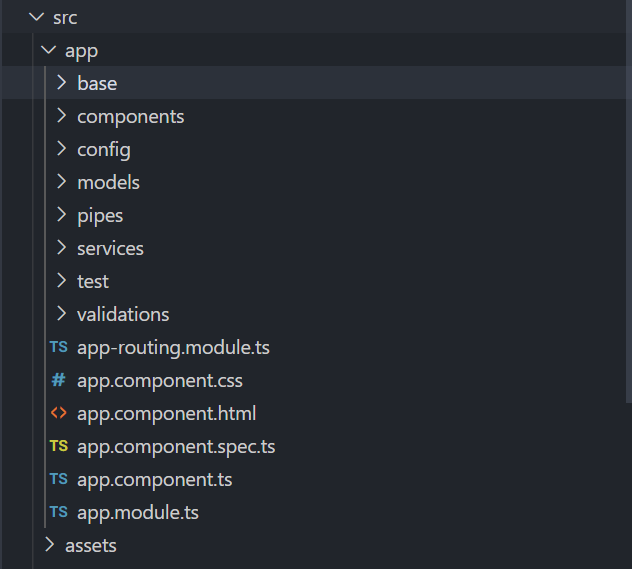
## e2e testing folder

An e2e/ subfolder contains configuration and source files for a set of end-to-end tests that correspond to the initial app. Workspace-wide node\_modules dependencies are visible to this project.

## App source folder and configuration files

Inside the src/ folder, the app/ folder contains your app's logic and data. Angular components, templates, and styles go here. An assets/ subfolder contains images and anything else your app needs. Files at the top level of src/ support testing and running your app.

|  |  |
| --- | --- |
| **File Name** | **Purpose of the configuration files** |
| .app/ | Contains the component files in which your app logic and data are defined. |
| .assets/ | Contains image files and other asset files to be copied as-is when you build your application. |
| environments/ | Contains build configuration options for particular target environments. By default, there is an unnamed standard development environment and a production ("prod") environment. You can define additional target environment configurations. |
| browserslist | Configures sharing of target browsers and Node.js versions among various front-end tools. |
| favicon.ico | An icon to use for this app in the bookmark bar. |
| index.html | The main HTML page that is served when someone visits your site. The CLI automatically adds all JavaScript and CSS files when building your app, |
| so you typically don't need to add any <script> or<link> tags here manually. |
| main.ts | The main entry point for your app. Compiles the application with the JIT compiler and bootstraps the application's root module (AppModule) to run in the browser. You can also use the AOT compiler without changing any code by appending the -–aot flag to the CLI build and serve commands. |
| polyfills.ts | Provides polyfill scripts for browser support. |
| styles.sass | Lists CSS files that supply styles for a project. The extension reflects the style preprocessor you have configured for the project. |
| test.ts | The main entry point for your unit tests, with some Angular-specific configuration. You don't typically need to edit this file. |
| tsconfig.app.json | Inherits from the workspace-wide tsconfig.json file. |
| tsconfig.spec.json | Inherits from the workspace-wide tsconfig.json file. |
| tslint.json | Inherits from the workspace-wide tslint.json file. |



|  |  |
| --- | --- |
| **App Source Files** | **Functions of the source files** |
| .app/base | Folder contains Viz network liberary code. Here we modified library codes as per our requirement |
| .app/components | Folder contains UI block of RelfinderNG application |
| .app/models | Folder contains typescript POJO files |
| .app/pipes | Folder contians angular pipe type script files |
| .app/services | Folder contains singleton service class |
| .app/test | Folder contains unit test configuration files which are used when jasmin unit tests |
| .app/validations | Folder contains form validation file |
| .app/app.component.ts | Defines the logic for the app's root component, named AppComponent. The view associated with this root component becomes the root of the view hierarchy as you add components and services to your app. |
| .app/app.component.html | Defines the HTML template associated with the root AppComponent. |
| app/app.component.css | Defines the base CSS stylesheet for the root AppComponent. |
| app/app.component.spec.ts | Defines a unit test for the root AppComponent. |
| app/app.module.ts | Defines the root module, named AppModule, that tells Angular how to assemble the application. Initially declares only the AppComponent. As you add more components to the app, they must be declared here. |
| assets/\* | Contains image files and other asset files to be copied as-is when you build your application. |

# Code walkthrough

## List of Service Classes

Angular services are singleton objects that get instantiated only once during the lifetime of an application. They contain methods that maintain data throughout the life of an application, i.e. data does not get refreshed and is available all the time. The main objective of a service is to organize and share business logic, models, or data and functions with different components of an Angular application. Available services listed below

* ConfigurationService : To handle endpoint configurations
* LoadConfigService : To load endpoint configuration form external json file
* FilterProcessService : To handle filters functionality
* QueryToolService: Handles HTTP calls and executes query in popup
* SparQLConnectionService: To prepare the node relationship data for graph visualization
* SparkQLQueryBuilderService : To prepare the queries based on depth & inputs
* SparQLResultParserService : To pars the RDF/XML content
* SparQLPropertyService : To fetch node/resource data
* ConstantService : To maintain constant data across application
* AutoCompletService : To autofill input data

### Configuration Service Class

This service class load list of endpoint configuration on application load from *config.json* file and keeps the details as singleton object. Using this service endpoint details can be used across all component. Also using this service endpoint details can be updated and deleted.

|  |  |
| --- | --- |
| Class Name | ConfigurationsService |
| Class Path | src\app\services\configurations\configurations.service.ts |

### Load Configuration Service Class

This service class loads external endpoint configuration file and validates the endpoint attributes and its values. This service call return list of errors to the load configuration component if the file attributes fail at validation process. It the file is valid it returns success message to calling component. Based on returned list the uploaded configurations are overrides existing endpoint configuration with new configuration details. This is temporarily store in configuration service until the page is reloaded. After reload the existing configuration form config.json file will be loaded in configuration popup list.

|  |  |
| --- | --- |
| Class Name | LoadConfigService |
| Class Path | src\app\services\configurations\load-config.service.ts |
| Used in | LoadConfigComponent |

### Filter Process Service

A service which handles node filter functionalities in Filter Component and Top panel component. Using this service user can hide, unhide the nodes in graph. This service takes relationship query responses and prepares the filter category details in JSON format. The filter service prepare below categories based on relationship query response.

* + 1. Class
    2. Link
    3. Length
    4. Connectivity

|  |  |
| --- | --- |
| Class Name | FilterProcessService |
| Class Path | src\app\services\filters\filter-process.service.ts |
| Used in | FilterComponent, TopPanelComponent |

### Query Tool Service

A service which handles interacts with selected endpoint server with query and displays the response in the popup. Based on the endpoint configuration this service triggers HTTP call to the endpoint server.

|  |  |
| --- | --- |
| Class Name | QueryToolService |
| Class Path | src\app\services\query-tool\query-tool.service.ts |
| Used in | QueryToolComponent |

### SparkQLQueryBuilderService

A service which is responsible for preparing entity relationship queries. Based on endpoint configuration attributes (i.e node depth) query is prepared and keep list of query for entities. This service called from SPARQLConnectionService class. If you want to change query patter generation, you can modify buildQueries() method in the service.

|  |  |
| --- | --- |
| Class Name | SPARQLQueryBuilderService |
| Class Path | src\app\services\sparql\sparqlquery-builder.service.ts |
| Referred in | SPARQLConnectionService |

### SparQL Connection Service

A service which is responsible for endpoint communication. It takes the list of relationship queries from SPARQLQueryBuilderService e and triggers N number of HTTP calls to endpoint server based on number of queries. Based on success response additional consecutive queries are called.

|  |  |
| --- | --- |
| Class Name | SPARQLConnectionService |
| Class Path | src\app\services\sparql\sparqlconnection.service.ts |
| Referred in | GraphService |

### SparQL Result Parser Service

A service which is responsible for parsing triples from HTTP response. The SPARQLConnectionService return data in JSON format. This parser service parse the JSON response and prepare the triples data and orders the data.

|  |  |
| --- | --- |
| Class Name | SPARQLResultParserService |
| Class Path | src\app\services\sparql\sparqlresult-parser.service.ts |
| Referred in | SPARQLConnectionService |

### SparQL Property Service

A service which is responsible for getting data of selected node in graph from the parsed triples data. Also it helps to get data based on langurage selected in filter popup.

|  |  |
| --- | --- |
| Class Name | SparqlPropertyService |
| Class Path | src\app\services\sparql\sparql-property.service.ts |
| Referred in | SPARQLResultParserService |

### Constant Service

A service which holds common constant data across the application.

|  |  |
| --- | --- |
| Class Name | ConstantsService |
| Class Path | src\app\services\util\constants.service.ts |

### Auto Complete Service

A service which is responsible for preparing entity finding query, executing and displaying result in search component auto dropdown list. This service takes input from active endpoint configuration and prepare queries based on query type pattern. There are 3 types of query type available. Each query type pattern will generate different type of query. To add new query pattern update the query type in configuration and add implementation in **executeQuery** method in this service class.

|  |  |
| --- | --- |
| Class Name | AutoCompleteService |
| Class Path | src\app\services\autocomplete.service.ts |
| Referred in | SearchComponent |

## List of component and Purpose

Application major components are divided into 4 group as below

1. Left panel component
2. Right panel component
3. Top panel component
4. Default component

### Left Panel Component

This group of component contains code for left side menu bar UI components. This components contains below sub components

1. Search Component
2. Filter Component
3. Info Component
4. Release Info Component
5. Share URL Componen
6. Configuration Component

Above sub components contains specific UI functionality.

|  |  |
| --- | --- |
| Component Name | LeftPanelComponent.ts |
| Component Path | src\app\components\left-panel\left-panel.component.ts |

#### Search Component

This component UI block of left side menu entity search popup. Using this UI block user can find entity names. User also reset the form at any time and search multiple entities at same time. This popup is shown at application load. This component save recent 5 distinct search entities in browser local storage. Stored data is available for quick entity selection. Stored data lives until browser session is deleted

|  |  |
| --- | --- |
| Component Name | SearchComponent.ts |
| Component Path | src\app\components\left-panel\search\search.component.ts |
| Dependent Services | AutoCompleteService, SPARQLConnectionService and others |

#### Filter Component

This component UI block of left side menu filter popup. Here node details are categorized in to 4 types. Using these we can filter nodes and toggle the node visibility in graph panel. By default filter menu is disabled and it is enabled only when the graph is being loaded.

|  |  |
| --- | --- |
| Component Name | FilterComponent.ts |
| Component Path | src\app\components\left-panel\filter\filter.component.ts |
| Dependent Services | FilterProcessService.ts |

#### Info Component

This component UI block of left side menu information popup. By default info menu is disable and only enabled when user click on any node. This popup shows information about the node and change the language of information. By changing the language it also changes the graph node language.

|  |  |
| --- | --- |
| Component Name | InfoComponent.ts |
| Component Path | src\app\components\left-panel\info\info.component.ts |
| Dependent Services | SPARQLConnectionService, ConstantsService and ConfigurationsService |

#### Release Info Component

This component UI block of left side menu application release information popup. This popup data is loaded from relase-info.json file. It used to show what feature is added and list of bug fixes.

|  |  |
| --- | --- |
| Component Name | ReleaseInfoComponent.ts |
| Component Path | src\app\components\left-panel\release-info\release-info.component.ts |
| Dependent Services | Nil |

#### Share URL Component

This component UI block of left side menu application share URL popup. This popup loads base64 encoded search entities and active endpoint in URL format. User can share the link and can populate the graph without configuring custom endpoint details.

|  |  |
| --- | --- |
| Component Name | ShareUrlComponent.ts |
| Component Path | src\app\components\left-panel\share-url\share-url.component.ts |
| Dependent Services | AutoCompleteService, SPARQLConnectionService |

#### Configuration Component

This component contains Configuration popup UI. Here list of endpoints are displayed here. This component depends on below sub components based on their functionality

1. Config Form Component
2. Load Config Component
3. Proxy Dialog Component
4. Query Tool Component

|  |  |
| --- | --- |
| Component Name | ConfigurationsComponent.ts |
| Component Path | src\app\components\left-panel\configurations\configurations.component.ts |
| Dependent Services | ConfigurationsService, ClearConfirmationDialogService |

##### Config Form Component:

This material popup component UI building block of Configuration popup. Here user provide endpoint details in input fields and save the endpoint details. If it is new endpoint configuration, the detail will be stored in ConfigurationsService class.

|  |  |
| --- | --- |
| Component Name | ConfigFormComponent.ts |
| Component Path | src\app\components\left-panel\configurations\config-form\config-form.component.ts |
| Dependent Services | ConfigurationsService, ClearConfirmationDialogService |

|  |  |  |
| --- | --- | --- |
| **#** | **Label** | **Description** |
| 1 | Name | Name of the endpoint |
| 2 | ID | Mandatory attribute. Unique ID for Endpoint Configuration |
| 3 | Descripton | Decription of endpoint |
| 4 | Endpoint URI | Mandatory attribute. The URL of SparkQL Endpoint. |
| 5 | Don't append '/sparql?' | Prevents RelFinder from appending '/sparql?' to the defined endpointURI. |
| 6 | Default Graph URI | The default graph which should be used for the search |
| 7 | Is Virtuoso Server | If the SPARQL server is a 'Virtuoso' server, you can set this value = true. Incase, If you are Unsure, Its suggested to leave it unchecked. This attribute is not required since query type attribute does the same job |
| 8 | User Proxy | Indicates to utilize proxy server for endpoint HTTP calls |
| 9 | Method | HTTP method for SPARQL requests. |
| 10 | Max Relation Length | The maximum length of relations RelFinder will search for. |
| 11 | AutoComplete URIs | A list of properties which will be used for autocomplete search and Input Disambiguation Process. You can always Identify Resource name referring to Link (http://www.w3.org/2000/01/rdf-schema#label) |
| 12 | Autocomplete Language | View Graph Data in Different Languages based on Dropdown Selections |
| 13 | Ignored Properties | Properties which should not be Shown in the result graph. |
| 14 | Abstract URIs | A list of property URI of abstracts or discriptions of the resource, which should be displayed in the info boxes, based on availability. |
| 15 | Image URIs | A list of property URI of images which should be displayed in the info boxes, based on availability. |
| 16 | Link URIs | A list of property URI of web links which should be displayed in the info boxes |
| 17 | Query Type | Type of query pattern to be generated |

List of attributes and usage are listed below table

##### Load Config Component:

This material popup component UI building block of uploading endpoint configuration JSON file. This component verifies file contend using LoadConfig Service and overrides existing endpoint configuration temporally until page is reloaded. If the file has error, it will not allow to override the configurations and displays error in same popup window.

|  |  |
| --- | --- |
| Component Name | ConfigFormComponent.ts |
| Component Path | src\app\components\left-panel\configurations\config-form\config-form.component.ts |
| Dependent Services | ConfigurationsService, ClearConfirmationDialogService |

##### Proxy Dialog Component:

This material popup component UI building block of Proxy server URL configuration. After adding URL it will update existing proxy server in configuration for temporally until page reloaded.

|  |  |
| --- | --- |
| Component Name | ProxyDialogComponent.ts |
| Component Path | src\app\components\left-panel\configurations\proxy-dialog\proxy-dialog.component.ts |
| Dependent Services | Nil |

##### Query Tool Component:

This material popup component UI building block of Query tool modal window. Here user can add custom query and check the response.

|  |  |
| --- | --- |
| Component Name | QueryToolComponent.ts |
| Component Path | src\app\components\left-panel\configurations\query-tool\query-tool.component.ts |
| Dependent Services | QueryToolService.ts |

### Right Panel Component

This Component UI block contains graph visualization area and page loading skeleton design. This component just binds visualization sub component.

|  |  |
| --- | --- |
| Component Name | LeftPanelComponent.ts |
| Component Path | src\app\components\left-panel\left-panel.component.ts |
| Sub Component | VisualizationComponent |
| Sub Component Path | src\app\components\right-panel\visualization\visualization.component.ts |

### Top Panel Component

This Component UI block contains code for filter section above graph window. Filtering

|  |  |
| --- | --- |
| Component Name | TopPanelComponent.ts |
| Component Path | src\app\components\left-panel\left-panel.component.ts |