# DELIVERABLE D4.2

**Tailored search application on top of data published using ODN – for test**

<table>
<thead>
<tr>
<th>Project</th>
<th>Components Supporting the Open Data Exploitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acronym</td>
<td>COMSODE</td>
</tr>
<tr>
<td>Contract Number</td>
<td>FP7-ICT-611358</td>
</tr>
<tr>
<td>Start date of the project</td>
<td>1\textsuperscript{st} October 2013</td>
</tr>
<tr>
<td>Duration</td>
<td>24 months, until 31\textsuperscript{st} September 2015</td>
</tr>
</tbody>
</table>

| Date of preparation                  | 22.9.2014                                    |
| Author(s)                            | Michiel Hildebrand, Roberto Cornacchia       |
| Responsible of the deliverable       | Roberto Cornacchia                           |
| Email                                | roberto@spinque.com                          |
| Reviewed by                          | Jan Gondol                                   |
| Status of the Document               | final version                                |
| Version                              | 1.1                                           |
| Dissemination level                  | PU (Public)                                  |
Table of Contents

Table of Contents
Executive summary
Deliverable context
  Purpose of the deliverable
  Related documents
Methodology used
  Step 1 - Implement Search by Strategy for Linked Data
  Step 2 - Develop a tailored prototype for Linked Data
  Step 3 - Develop a tailored prototype for Linked Data using ODN
  Step 4 - Develop a second tailored prototype for Linked Data using ODN
Spinque LD
Using Spinque LD on top of ODN
  Container
    Import RDF files
    SPARQL Importer
    LOD crawler
  Strategy editor
  Manager
  REST API
  Applications
Search application for the Rijksmuseum
  Basic search
  Improved ranking
  Multilingual search through DBPedia
  Searching multiple result types
  Recommending related artists and artworks
Search application for the Czech inspection authority
  Finding a specific business
  Exploring Linked Data
Search application for Czech restaurant inspections
  Finding restaurants on the map
  Finding restaurants by name and addresses
Outlook
1. Executive summary

Deliverable D4.2 is about one end of the open data supply-demand chain addressed in COMSODE: applications for end-users. The objective of this deliverable is described in the DOW as: *Tailored search application on-top of data published using ODN – for test: Freely accessible search application based on SPINQUE’s “Search by Strategy” concept, tailored to open data published as part of the COMSODE project.* Its output includes prototype applications that can show the feasibility of the Open Data Node approach, from the point of view of application development. This document describes the purpose of the deliverable, the methodology used, and the outputs.

The Open Data Node developed in COMSODE is intended to publish high-quality data that can be reused effectively by application developers (and, to some extent, by end-users directly). The prototype applications developed for D4.2 represent an early-stage validation of the ODN platform from a developer perspective. We show how we approached this validation in 4 steps, developing prototypes for a new Spinque product and three applications with it.

The main results achieved were:

- New technology developed, which will be part of the exploitation strategy (D7.3)
- Early validation of this technology
- Early validation of ODN output (D4.1)
- Early validation of additional benefits of the overall approach, such as ease of development of diverse applications on the same datasets and usage of diverse user interfaces for the same search technology

The applications developed (described below) are available online:

1. Search for Rijksmuseum
   [http://www.spinque.com/rijksmuseum](http://www.spinque.com/rijksmuseum)
2. Search for Czech inspection authority
   [http://devel.spinque.com/COMSODEApp-0.1/](http://devel.spinque.com/COMSODEApp-0.1/)
3. Search for Czech restaurant inspections

2. Deliverable context

**Purpose of the deliverable**

The purpose of this deliverable is to build early-stage demonstrators for the feasibility of building a tailored search application on top of the Open Data Node (ODN) architecture. Demonstrating more mature integration of ODN and tailored search applications is expected in deliverable D4.4. The expression “tailored search application” is used to indicate an application that can address effectively specific combinations of data domains and search needs. An example of tailored search application is one focused on recruitment (domain), from the point of view of IT job suppliers (user need).
Implementing the ODN platform and its related methodologies allows data publishers to make their data available in standard data formats for Linked Data¹, with a number of quality guarantees and with consistent workflows. Although such published data are typically human-readable, their size and structured representation do not allow end-users to make much sense of them in practice. They only become useful when tailored applications can re-interpret them and offer focused services to end-users. Search represents a very important class of applications, as searching through the published data is a very common user need.

The output of COMSODE aims to address many different datasets during and beyond the time-span of the project. Such variety calls for the possibility to develop tailored applications on top of the ODN platform. Focusing on search, because of its broad applicability and interest, this deliverable develops prototypes in order to address the following questions:

- What are the requirements for developing on top of ODN?
- Is the output of ODN suitable for tailored search application development?

Related documents
List of related documents from project:
- DOW, version date 2013-08-06, pp. 13-14
- D4.1 - COMSODE publication platform - Open Data Node - for test
- D7.3 - Half-time exploitation strategy

3. Methodology used
The final goal of this deliverable is a tailored search application that uses data published by the ODN platform. We slightly extended the output of this deliverable (by developing more than one application) and split it into four development steps.

Step 1 - Implement Search by Strategy for Linked Data
Search by Strategy² is the approach used to achieve the ease of engineering of tailored search applications.

Search by Strategy is a novel concept, developed by Spinque, that simplifies the engineering of search systems by information specialists (e.g. librarians, patent experts, journalists, strategic analysts - anyone deeply involved in managing information). Strategies are high-level specifications of complete search processes, from source information to expected results. The design of search strategies is supported by a web-based, visual environment where operational "blocks" are connected into a graph. This allows to abstract data-access details away (the "what") and focus instead on picturing a visual strategy that corresponds to the mental process (the "how") of the information specialist, for any combination of information need and intended end-user. Making all such diverse strategies available as web services is then a matter of a click, and changing them is likewise simple.

¹ http://www.w3.org/wiki/LinkedData
² http://www.spinque.com/concept
During the first year of the COMSODE project we extended the existing Spinque platform to enable the Search by Strategy approach on Linked Data. This resulted in Spinque Linked Data (or Spinque LD) that enables a data publisher to create search strategies on top of RDF data managed and published in an Open Data Node. Section 4 gives some more details about this effort.

**Step 2 - Develop a tailored prototype for Linked Data**

During the first development step, the ODN platform was in development, which did not allow to use its output for testing Spinque LD. However, for the purpose of this testing phase, any available linked data of sufficient quality could have been used. We validated Spinque LD with the open dataset from the Rijksmuseum in Amsterdam, Netherlands, that was already available in RDF. We integrated this with additional Linked Open Data from DBPedia to create search strategies that improve the ranking and enable multilingual search on the Rijksmuseum collection. This led to an online application that is described in Section 5.

**Step 3 - Develop a tailored prototype for Linked Data using ODN**

When the ODN could produce real output for the some of the selected datasets (D3.1), we built a search application using such data. The dataset considered is provided by the Czech inspection authority\(^3\) (COI). The application is intended for internal usage by COI, to assist the navigation of their own database by their personnel. This led to an online application that is described in Section 6.

**Step 4 - Develop a second tailored prototype for Linked Data using ODN**

We built a second application, using the same datasets as in the previous step, but changing the targeted audience and user needs. The datasets from the Czech inspection authority have been used to inform end-users (general public) about the result of inspections at restaurants and other food-related stores. Also, this application uses a completely different user interface, showing how this is decoupled from the search functionalities. This led to an online application that is described in Section 7.

In the rest of this document, we first describe briefly the core changes we made to the Spinque platform to support RDF, corresponding to step 1 above. Then, we illustrate the workflow to enable search on top of the Open Data Node (ODN). Finally, we describe the three prototype applications corresponding to steps 2, 3, 4 above. We wrap up with conclusions and an outlook for the second year of the project.

### 4. Spinque LD

Spinque search applications are created as vertical solutions on top of the Spinque Core platform\(^4\). Spinque Core includes the back-end query engine, the management tools, and the general-purpose functionalities that are needed to apply the Search by Strategy approach: 1) define several search strategies, each addressing a specific user need on top of the data at

---

\(^3\) [http://www.coi.cz](http://www.coi.cz)

\(^4\) [http://www.spinque.com/spinque-core](http://www.spinque.com/spinque-core)
hand, 2) publish each strategy as an API element and 3) build end-user search applications using the generated custom API.

In order to apply the Search by Strategy concept in the context of COMSODE, Spinque branched the Spinque Core components into a new variant that specifically targets Linked Data that the Open Data Node makes available. The result of these branching and development activities is a brand new product, Spinque Linked Data or in short Spinque LD. This product has now reached the level of maturity that is required to successfully create prototype applications. By the end of the project, Spinque intends to enrich their commercial offering with a fully functional Spinque LD platform, in order to open up opportunities in the public sector and wherever Linked (Open) Data is in use.

The effort spent into creating Spinque LD included the following:

- The development of new data connectors for Linked Data:
  - Data dumps in RDF formats (RDF/XML, Turtle, NTriples),
  - Access to SPARQL end-points
  - Crawling of Linked Data

  These connectors enable data import data from an Open Data Node to Spinque LD.

- An extension of the internal data-model of Spinque to support the triple-oriented RDF model. Spinque Core already used a triple model, but this was not as generic as the one allowed in RDF. Spinque LD now supports the W3C standardized triple model as defined by RDF.

- An addition of a fully automatic indexing process for RDF data. In Spinque Core the indexing process involves a manual a step in which the maintainer describes the mapping of the data model to the internal Spinque representation. In Spinque LD data represented in RDF is indexed without the need of human intervention.

- A mapping of RDF specific index structures, as defined by the rdfs:subPropertyOf and rdfs:subClassOf properties, to efficient Spinque tree indexes. This enables powerful reasoning with hierarchical structures in Spinque LD search strategies.

- Addition of building blocks for specific RDF constructs. With these building blocks search strategies in Spinque LD can use typical relations found in Linked Data, such as equivalence relations in owl:sameAs.

5. Using Spinque LD on top of ODN

Spinque LD exists as a separate system alongside an Open Data Node. Figure 1 shows the workflow to create a search application in Spinque with data from ODN. We first define in a container which data is imported into Spinque. These data are then processed into an index in the Spinque database. Search algorithms are defined in the Strategy Editor, which are then published through the Manager in executable algorithms on the database. The published strategies are made available through the REST API on top of which search applications can be built. We describe the different steps in turn.

---

5 [http://www.w3.org/RDF/](http://www.w3.org/RDF/)

6 As described in the DOW and deliverable D7.3, the search component developed by Spinque is the only output of COMSODE that will be available under a proprietary license.
To make data from an ODN available for search it needs to be imported into Spinque. Within the COMSODE project we extended the data import facilities of Spinque to support RDF data. We support three methods to import RDF data into Spinque.

- **RDF files**
  ODN creates a RDF dump. Spinque imports the files from the RDF dump. This approach is best suited for large data sets that are not updated frequently.

- **SPARQL importer**
  ODN makes RDF data available through a SPARQL endpoint. Spinque imports data from the SPARQL endpoint as specified by a query. This approach is best suited to get all data from medium sized datasets and for incremental updates of data that frequently changes.

- **LOD crawler**
  ODN makes RDF data available as Linked Open Data. Spinque crawls data about a specific resource by resolving its URI. This approach is best suited to add new data (for a medium sized number of resources) to an existing Spinque index.

Data is imported into Spinque through a container definition. Containers are defined in an XML configuration file. We illustrate the containers for the three methods to import RDF.

**Import RDF files**

The example below shows how to import an RDF file. The attribute type defines the container that is used. The source contains the location of the file. Spinque uses Jena to parse RDF files. Jena uses the file extension to recognize the serialization format, as described at [https://jena.apache.org/documentation/io/](https://jena.apache.org/documentation/io/).

```xml
<container type="rdf">
  <source>C01_CZ_kontroly/coi-kontroly.ttl</source>
</container>
```
Each RDF file requires its own container. Through the group container multiple definitions are grouped together into valid XML.

```xml
<container type="group">
  <container type="rdf">
    <source>COI_CZ_kontroly/coi-kontroly.ttl</source>
  </container>
  <container type="rdf">
    <source>COI_CZ_sankce/coi-sankce.ttl</source>
  </container>
</container>
```

**SPARQL Importer**

The container for a SPARQL import requires the URL of the SPARQL endpoint and a construct query. Listing 3 shows an example that gets all data from an endpoint. By defining a more specific query the publisher can selectively import data. The number of results that are indexed can be limited by setting the `pageSize` option.

```xml
<container type="lod-sparql">
  <endPoint>http://odn.example.com/sparql/</endPoint>
  <options pageSize="200"/>
  <query>CONSTRUCT * WHERE {?s ?p ?o}</query>
</container>
```

**LOD crawler**

The container for an LOD crawler requires the URI of the resource that is the starting point for the crawl. The crawl will resolve this URI. Crawling is recursive in the sense that each URI returned in the RDF document is also crawled. By defining the `crawlDepth` option the crawling process is limited to a specific number of steps from the target URI. The crawling process can also be constrained by the number of URIs. Other options configure how the crawling is executed. The example below crawls URIs recursively up to two steps with a maximum of 10 URIs.

```xml
<container type="lod-crawl">
  <source>http://dbpedia.org/resource/Cheese</source>
  <options maxURIs="10" numPLDs="1" crawlDepth="2" numThreads="1" maxBatchSize="100"/>
</container>
```

**Strategy editor**

Spinque makes data searchable through strategies. A strategy is constructed in the Strategy Editor out of simple building blocks. A screencast that demonstrates the Spinque editor is available at https://www.youtube.com/watch?v=YFKJmWlk0yw.

Figure 2 shows a screenshot of the strategy editor. The interface consists of three main parts. On the left side the library with building blocks, categorized by function. In the middle the strategy canvas with a graphical representation of the search strategy. The strategy consists of building blocks, represented by the rectangles in the diagram, connected from
output to input, represented by the lines. The user creates a strategy by dragging building blocks from the library on to the strategy canvas. Building blocks are configured and connection are made by dragging lines between the input and output connectors. The right side of the interface contains tools to compile the strategy and test it. When building block in the strategy are configured to have inputs these appear as input fields in the tools area. In this case there is a single input field to enter keywords. By submitting an input the user can test the strategy by inspecting the result preview.

The screenshot is Figure 2 shows a basic search strategy. The strategy defines a search engine where the input is a textual query and the output is a ranked list of items matching the query. The building block labeled `genericIndex` provides the index with all the data. The block `keywords` creates the input for the strategy, and the block connected to it, `stem_TERM`, stems the keywords. This block is configured to use a czech stemming algorithm. Finally the block `rank_text_BM25` applies the BM25 search algorithm on the index with the stemmed keywords as input.

![Strategy editor screenshot](image)

Figure 2. Screenshot of the Strategy editor. On the left it shows the library with building blocks, in the middle the strategy canvas and on the right the tools to compile and the test the strategy.

**Manager**

To make a search strategy available it needs to be compiled and published as an API. This is done through the management interface of Spinque. Figure 3 shows a screenshot of the strategy management. It shows the list of strategies on the left. On the right side of the screen information is shown for the selected strategy. Here the display label can be changed. With the buttons a strategy is compiled and published. When a strategy is changed it needs to be compiled and published again.
Figure 3. Screenshot of the management console to maintain, compile and publish strategies into APIs.

REST API
When a strategy is published it is available through the Spinque REST API. The full specification of the API is available in the Spinque documentation. The general pattern for the API is:

/rest-rdf-1.5/{PROJECT}/q/{STRATEGY}/p/{INPUT_PARAMS}/{ACTION}

where {PROJECT} is the name of the project where the strategies exist in, {STRATEGY} is the name of the published strategy and {INPUT_PARAMS} contains the key/value pair for each input parameter. The last part of the request {ACTION} defines what is returned. The results request returns the items that match the query. Statistics about the results are available through the statistics request.

To request the results for the strategy business_search with the query grill becomes:

/rest-rdf-1.5/comsode/q/dashboard%252Fbusiness_search/p/q/grill/results

Applications
Search applications are built on top of the Spinque REST API. Spinque provides a default search interface that works out of the box with Spinque strategies. We used this interface for data from the Dutch Rijkmuseum and the Czech trade inspections. For a second tailored search application on top of the Czech trade inspections data, focusing on restaurants, we developed a new interface. We describe the applications in the next sections.
6. Search application for the Rijksmuseum\textsuperscript{7}

As described above in this document, the purpose of this application is to validate the development of Spinque LD, which applies the Search by Strategy concept to linked data. Using Spinque LD we created a search application for the publicly available collection of the Rijksmuseum in Amsterdam. The data contains artworks with descriptions (or annotations) provided by the cataloguers of the museum. Like many museums and other cultural heritage institutions these annotations cover the basic object characteristics such as the creator, date and material. In addition, cataloguers have described what is depicted on the artworks, the subject matter.

We explored different strategies to search in the Rijksmuseum collection. We started off with an RDF representation of the artwork collection and the thesauri that are used to described the artworks. We first demonstrate how to make a basic search engine on top of this data. Next we integrated the Rijksmuseum with another Open Dataset, DBPedia. As the Rijksmuseum application was developed in parallel with the development of the Open Data Node we manually integrated a small part of the Rijksmuseum with DBPedia. The Open Data Node will enable us to do at larger scale and more systematically. This is planned for the second year of the project.

Using DBPedia we improve the ranking, enable multilingual search, and provide recommendation of related artists and artworks. We encourage the reader to try it out in the Rijksmuseum search prototype at:

http://www.spinque.com/rijksmuseum

Basic search

Creating a basic search engine with Spinque for the Rijksmuseum collection is straightforward, and is explained in the previous section. The strategy is available in the search application. Select the strategy from the drop-down list at the top left, it is named \textit{artwork search}. Try to find the famous painting from Vermeer by searching with the Dutch title “melkmeisje”.

Improved ranking

The ranking is probably not what you expected. Why do we \textbf{not} get the famous masterpiece by Vermeer as the first result? The problem is that the textual features in the collection do not predict if an artwork is famous or not. Using the facets to filter on the creator Johannes Vermeer we can solve the problem, but the point is that the public will expect the masterpieces on top. In this case there is a simple solution. The Rijksmuseum has explicitly annotated the top 1000 most famous artworks. With Spinque we can use this information in a search strategy and give a prior to the ranking. Figure 4 shows the strategy where artworks that are in the top 1000 get a higher initial weight than other artworks. This is done by filtering out the artworks in top 1000, left branch in the strategy, from the artworks and then mixing these back together with different probabilities. The building block \texttt{mix\_2\_ANY} mixes the results setting the initial probability for the top 1000 to 0.9 while the other artworks get an

\textsuperscript{7}The description of the search application is also published on the COMSODE blog

initial probability of 0.1. Further down in the strategy this initial probability is combined with the probability from the BM25 ranking algorithm. In other words we have defined in the strategy that artworks from the top1000 are more relevant search results.

![Diagram of search strategy for artworks from the Rijksmuseum. A prior is given to artworks from the top 1000.](image)

Figure 5 Search strategy for artworks from the Rijksmuseum. A prior is given to artworks from the top 1000.

In the search application you can try the strategy with the improved ranking. The strategy is named `artwork_search_top1000`. Click the button `change` and select this strategy. Again try the query “melkmeisje”. The results are also shown in the screenshot of Figure 5.
Multilingual search through DBPedia
Wikipedia (or DBPedia) contains information about well known artists such as Vermeer, including relations to other artists. By integrating DBPedia in the Rijksmuseum search application we can prioritize the artworks that are found there (thus that are described on Wikipedia). This is an alternative (or maybe a complementary) approach to get the masterpieces on top. In the search application this is available under the strategy *artwork_search_dbpedia*. Searching for “melkmeisje” should give you the same result as before.

DBPedia also contains the title of the artwork in other languages. Now you can also search in English “milkmaid”, in Spanish “La Lechera”, in Polish “Mleczarka” and other languages. Quite handy for a museum with an international audience.

Searching multiple result types
In the previous examples we searched for artworks. Sometimes the object of interest is something else, for example the artist (e.g. Vermeer). The search strategy *aggregated_search* illustrates how different result types can be integrated in a single result list. Search for “vermeer” to see an example of a result list containing artworks as well as artists.

Recommending related artists and artworks
We can also use the rich information contained in DBPedia for recommendation. Select the result Vermeer, Johannes by clicking the + icon at the bottom right corner. You see more information about this artist on the right side of the application. The descriptive text about Vermeer comes from the Rijksmuseum thesaurus and the list of artworks created by Vermeer is also taken from the Rijksmuseum data. The list of related artists, however, is derived from DBPedia and is not available in the Rijksmuseum data itself. Using these links we can browse to other famous Dutch artists, such as Gabriel Metsu, and explore their
artworks. To find the artworks created by Gabriel Metsu we again use information from the Rijksmuseum itself. In other words we just used DBPedia to browse from one artist in the Rijksmuseum to the artworks of a related artist.

7. Search application for the Czech inspection authority

The purpose of this application is to demonstrate the feasibility of developing a tailored search application on top of ODN. The application is available online at: http://devel.spinue.com/COMSODEApp-0.1/.

The Czech Trade Inspection Authority checks businesses around the country. The reports of these inspections are made available as Linked Open Data within the COMSODE project. In addition, this data is linked to the Czech business registry ARES, providing additional information about the business that was checked. This includes the activities of a businesses as defined by the NACE code list from the EU. We explored how Spinque could enable search on top of these integrated datasets with multiple strategies that are made available through an explorative search application.

**Finding a specific business**

The screenshot in Figure 6 shows the search application with the data from the Czech trade inspection. On the top left the user selects one of the published strategies. In this case the strategy to find a business by it’s name is selected. In the text input field the query tesco was entered and the screenshot shows the list of results. By selecting a different strategy the user can search for businesses through the addresses or the activities. Instead of searching for businesses a strategy can be selected to find inspections. For example, to find inspections in a specific city or where specific products that were confiscated.

**Figure 6. Screenshot of the Spinque search application for the data from the Czech trade inspection.**

**Exploring Linked Data**

Besides search through predefined strategies the application provides functionality for data-driven exploration. With the facets shown on the right side of the application the user
can drill down the results. For example, by filtering the businesses matching the keyword *tesco* to those that occur in a specific city.

Another approach to start exploration is by clicking the plus icon on the bottom right of a search result. This brings up a result browser as shown in Figure 7. This contains the metadata of the selected item and below it a list of related resources. In this case it shows the inspections that were conducted at the business and the physical locations of the different establishments of the business. The related resources that are shown for a result are itself defined in a strategy.

![Screenshot of the object explorer showing the inspections that were conducted at Tesco, left column, and all the establishments of Tesco, right column.](image)

Yet another way of exploring the data is with the two special facets shown at the bottom of the facet list in Figure 6, *relations [in]* and *relations [out]*. These two facets list the relations coming in and out of the current result sets. By selecting a relation from these facets the user traverses the result set to a new result set. This provides a powerful mechanism to explore the graph. For example, by selecting the relation *object* the user traverses the graph from the businesses to the inspections that were done at them. Figure 8 shows a visualization of the
inspections with Spinque’s generic timeline widget. From the inspections we can even traverse the graph one step further and and find all the sanctions that were given.

8. Search application for Czech restaurant inspections

The purpose of this application is twofold: 1) to demonstrate that the same data can be easily be used to build applications targeted at different audiences; 2) to demonstrate that search functionalities and user interfaces are completely decoupled. The application is available online at: http://www.spinque.com/czech-restaurant-inspections. We created a second tailored search application on top of the data from Czech trade inspections. This applications targets the general public and focuses on the inspections of restaurants. It was build on the same index as the previous search application. The required backend functionality is completely defined by Spinque strategies. There was no need for other data management or server-side application logic. The frontend of the application is made using standard client side Web technologies HTML, CSS and JavaScript. The Google Map JavaScript library and API is used for the Map visualization.

Finding restaurants on the map

The application provides two main functions. First, the user can find restaurant inspections through a geographical map. Figure 9 shows a screenshot of the application. The map shows markers at the location of a restaurant. As indicated in the legend the color of the marker indicates if issues are found at a restaurant during an inspection. By panning and zooming on the map the user can navigate through a particular area of interest. The application will update the markers that are shown on the map based on the users viewport. In this case the map shows the restaurants in an area in Prague.
When a user selects a marker a panel appears in the interface showing the basic information about the restaurant, such as the addresses. It also lists all the inspections that were conducted at the restaurant. This functionality is also provided by Spinque strategies.

![Figure 9. Screenshot of the tailored search application for the general public with trade inspections of Czech restaurants. It shows restaurants in the Czech republic on a map. The colors indicated if inspections were conducted and if issues were found.](image)

The search functionality for this function of the application is defined in the Spinque strategy shown in Figure 10. It returns the businesses that are within a geographical area. The constraints of the area determined on the client side using the Google Maps library. We take the latitude and longitude from the center of the map that is displayed and the maximum of width and the height as the allowed distance. This is provided as input the strategy. The strategy uses the center and distance to filter all the objects from the database by their latitude and longitude coordinates using the building block `filterOnPointDistance`. Next we traverse the graph from addresses to the actual restaurants. Finally, we rank the restaurants using the number of inspections that are conducted at them.
Finding restaurants by name and addresses

The second function of the application allows a user to find a specific restaurant by name. In a similar way as on Google Maps the application provides suggestion when the user starts typing in the text input field. When the user hits enter a full list of search results are presented. The auto-completion and the full search functionality are defined as Spinque search strategies.
9. Outlook

In the first year of the project we extended Spinque to support search on Linked Data. We used Spinque Linked Data to implement three prototype search applications, including two applications with data from the Czech inspection authorities that is created and published with the COMSODE Open Data Node. In this document we described the workflow to create search applications on top of ODN. For the prototype applications the steps in this workflow were all done by employes of Spinque. In the second year of the project we aim to bring the Spinque solution closer to the data publishers. To realize this we plan to improve the support to work with RDF data in Spinque, train partners in the COMSODE project to work with Spinque and create search strategies, and together create tailored search applications.