



## OpenGovIntelligence

Fostering Innovation and Creativity in Europe through Public  
Administration Modernization towards Supplying and Exploiting  
Linked Open Statistical Data

---

### Deliverable 3.2

#### Report on OpenGovIntelligence ICT tools - first release

---

<b>Editor(s):</b>	Arkadiusz Stasiewicz (NUIG) Mohammad Waqar (NUIG)
<b>Responsible Organisation:</b>	NUIG
<b>Version-Status:</b>	V1.0
<b>Submission date:</b>	31/10/2016
<b>Dissemination level:</b>	PU

## Deliverable factsheet

<b>Project Number:</b>	693849
<b>Project Acronym:</b>	OpenGovintelligence
<b>Project Title:</b>	Fostering Innovation and Creativity in Europe through Public Administration Modernization towards Supplying and Exploiting Linked Open Statistical Data

<b>Title of Deliverable:</b>	D3.2 – Report on OpenGovIntelligence ICT tools - first release
<b>Work package:</b>	WP3 – ICT tools development
<b>Due date according to contract:</b>	31/10/2016

<b>Editor(s):</b>	Arkadiusz Stasiewicz (NUIG) Mohammad Waqar (NUIG)
<b>Contributor(s):</b>	Dimitrios Zeginis (CERTH) Evangelos Kalampokis (CERTH) Efthimios Tambouris (CERTH) Konstantinos Tarabanis (CERTH) Arkadiusz Stasiewicz (NUIG) Mohammad Waqar (NUIG) Mohamed Adel (NUIG) Bill Roberts (SWIRRL) Rick Moynihan (SWIRRL)
<b>Reviewer(s):</b>	Bill Roberts (SWIRRL)
<b>Approved by:</b>	All Partners

<b>Abstract:</b>	This deliverable provides the description of the prototypes of software components delivered as a result of the first OpenGovIntelligence ICT tools development stage.
------------------	--

**Keyword List:**

software, components, linked data, statistical data, ICT tools

## Consortium

	<i>Role</i>	<i>Name</i>	<i>Short Name</i>	<i>Country</i>
1.	Coordinator	Centre for Research & Technology - Hellas	CERTH	Greece
2.	R&D partner	Delft University of Technology	TU Delft	Netherlands
3.	R&D partner	National University of Ireland, Galway	NUIG	Ireland
4.	R&D partner	Tallinn University of Technology	TUT	Estonia
5.	R&D partner	ProXML bvba	ProXML	Belgium
6.	R&D partner	Swirrl IT Limited	SWIRRL	United Kingdom
7.	Pilot Partner	Trafford Council	TRAF	United Kingdom
8.	Pilot Partner	Flemish Government	VLO	Belgium
9.	Pilot Partner	Ministry of Interior and Administrative Reconstruction	MAREG	Greece
10.	Pilot Partner	Ministry of Economic Affairs and Communication	MKM	Estonia
11.	Pilot Partner	Marine Institute	MI	Ireland
12.	Pilot Partner	Public Institution Enterprise Lithuania	EL	Lithuania

## Revision History

<i>Version</i>	<i>Date</i>	<i>Revised by</i>	<i>Reason</i>
0.1	30/08/2016	NUIG	First draft
0.2	15/09/2016	NUIG	General updates
0.3	14/10/2016	NUIG	Tools Description (draft)
0.4	26/10/2016	NUIG	Architecture details
0.5	28/10/2016	NUIG	General updates
0.6	29/10/2016	NUIG	General updates
0.7	30/10/2016	NUIG	Addressed review comments
1.0	31/10/2016	CERTH	Submission to EC

**Statement of originality:**

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

## Table of Contents

<b>DELIVERABLE FACTSHEET.....</b>	<b>2</b>
<b>CONSORTIUM.....</b>	<b>4</b>
<b>REVISION HISTORY .....</b>	<b>5</b>
<b>TABLE OF CONTENTS .....</b>	<b>6</b>
<b>LIST OF FIGURES .....</b>	<b>7</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>8</b>
<b>EXECUTIVE SUMMARY.....</b>	<b>9</b>
<b>1 INTRODUCTION .....</b>	<b>10</b>
1.1 SCOPE .....	10
1.2 AUDIENCE .....	10
1.3 STRUCTURE .....	10
<b>2 ARCHITECTURE OVERVIEW .....</b>	<b>11</b>
2.1 DATA PROVISION .....	11
2.2 DATA PLATFORM .....	14
2.3 PROCESS LAYER .....	15
2.4 SERVICE DESIGN .....	16
2.5 SERVICE PROVISION.....	18
2.6 MANAGEMENT.....	18
<b>3 OPENGOVINTELLIGENCE ICT TOOLS - FIRST RELEASE .....</b>	<b>20</b>
3.1 JSON API FOR DATA CUBE SPECIFICATION .....	20
3.2 JSON API FOR DATA CUBE IMPLEMENTATION .....	21
3.3 TABLE2QB AND GRAFTER .....	28
3.4 DATA CUBE BUILDER .....	30
3.5 DATA CUBE EXPLORER .....	33
3.6 OLAP BROWSER .....	36
<b>4 CONCLUSION .....</b>	<b>38</b>
<b>REFERENCES .....</b>	<b>39</b>

## List of Figures

FIGURE 1 OGI ARCHITECTURE .....	13
FIGURE 2 CUBE BUILDER DESKTOP UI.....	30
FIGURE 3 DATA CUBE EXPLORER - DATASET SELECTION .....	34
FIGURE 4 DATA CUBE EXPLORER - SAMPLE VISUALISATION .....	34

## List of Abbreviations

The following table presents the acronyms used in the deliverable in alphabetical order.

<i>Abbreviation</i>	<i>Description</i>
API	Application Programming Interface
CSV	Comma Separated Values
ICT	Information and Communication Technologies
LOSD	Linked Open Statistical Data
OLAP	OnLine Analytical Processing
RDF	Resource Description Framework
UI	User Interface
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
WP	Work Package



## Executive Summary

The present document is the deliverable “D3.2 – Report on OpenGovIntelligence ICT tools – first release” (referred to as D3.2). It provides detailed information about the result of the first stage of the OpenGovIntelligence development. The goal of WP3 (“ICT tools development”) is to develop the OpenGovIntelligence ICT tools as a suite of open source and commercial tools. The developed ICT tools will support the OpenGovIntelligence framework created at WP2 (“Framework creation”) and will enable:

- (a) the creation of Linked Open Statistical Data (LOSD) from various sources,
- (b) the expansion of LOSD with datasets from existing sources,
- (c) the exploitation of LOSD for the co-production of public services.

The first release of the OpenGovIntelligence ICT tools cover the tools regarding the creation and exploitation of LOSD. With regards to the creation of LOSD, OpenGovIntelligence designed and developed ICT tools that enable the transformation of public sector data to standard machine readable forms (specifically to RDF data cubes) and the validation of the generated RDF. With regards to the exploitation of LOSD, OpenGovIntelligence designed and developed ICT tools that enable visualisation and analysis of statistical data.

The tools presented in this deliverable will be evaluated during the pilot implementations stage. Based on the outcomes, the existing tools will be improved. Moreover new tools will be developed in order to cover remaining features of the OpenGovIntelligence Framework.

## 1 Introduction

This section introduces the background of the work carried out in WP3 “ICT tools development”. Sub-section 1.1 presents the scope and the objectives of the current document, sub-section 1.2 describes the intended audience for this document, while sub-section 1.3 outlines the structure of the document.

### 1.1 Scope

This report documents the ICT tools developed during the first phase of WP3. To guide readers in understanding the context of these tools, Section 2 presents the high level architecture of the OpenGovIntelligence Framework, showing the overall organization of layers, components and their interactions. This first release focuses on the development of tools that will enable creation and exploitation of LOSD. In subsequent stages further tools are planned covering expansion of existing datasets and exploitation of LOSD for co-production of public services.

### 1.2 Audience

The intended audience for this document is the OpenGovIntelligence consortium, in particular partner organisations responsible for the development of pilot trials, the European Commission (EC) and those who are interested in challenges and needs for opening-up and exploiting LOSD for the co-production of innovative data-driven services on governments.

### 1.3 Structure

The structure of the document is as follows:

- Section 2 presents the OpenGovIntelligence Architecture and provides descriptions of each of the individual components.
- Section 3 provides a detailed description of the individual OpenGovIntelligence ICT tools developed during the first development stage of OpenGovIntelligence project;
- Finally, Section 4 concludes the report and outlines the future development plan.

## 2 Architecture Overview

The OGI Architecture for LOSD and data-driven public services enables stakeholders to collaborate towards the production of innovative data-driven public services by exploiting Linked Open Data technologies and statistical datasets. This is the initial version of the architecture that will be tested and refined during the first round of pilot evaluations. The architecture is presented in **Error! Reference source not found.** and it is organised as follows:

- The architecture is divided into five layers: (i) Data Provision, (ii) Data Platform, (iii) Process Layer, (iv) Service Design, and (v) Service Provision.
- Key management responsibilities are shared across all layers.
- Each layer has a set of components that performs tasks specific to that layer.

The architecture will guide the pilot implementations, as well as the other future implementations of the OGI software in other projects.

Each part of the architecture is described in detail in the following sections.

### 2.1 Data Provision

The Data Provision Layer implements fundamental functionalities needed to create high quality LOD and thus support the execution and scalability of the Service Design. The Data Provision services RDF creation, schema, code lists, vocabularies and metadata management services in order to enable the extensibility and scalability and assure the data quality of the proposed OpenGovIntelligence approach.

#### 2.1.1 LOD Creation

The Linked Open Data Creation group includes the following modules:

##### 2.1.1.1 Metadata Management

Metadata is essential to learn what data is available and understand and interpret the statistical data - without it the data is meaningless.

The metadata can be categorised as:

- *Structural metadata* - code or concept lists, dimension definitions, units of measure, etc.
- *Reference metadata* - information on methodology and quality
- *Process metadata* - description of production processes, or process metrics
- *Provenance* - description of the source of information, origin of the data, digital rights, etc.

OpenGovIntelligence metadata is represented as Linked Data, as it will be used as a medium to discover aggregated cubes that are stored in dispersed data repositories. Metadata Management

will be carried out by mapping the available descriptive, structural and administrative characteristics of the digested dataset using the Data Catalog Vocabulary (DCAT<sup>1</sup>) vocabulary, which is a widespread W3C standard for describing datasets.

#### **2.1.1.2 Vocabularies Management**

Re-use of accepted controlled vocabularies (for example those from W3C, Eurostat or national statistical organisations) is one of the most important principles of Linked Data. It enables the linking of different cubes at the schema level (i.e. measures, dimensions, attributes). Consequently, there is a need for managing controlled vocabularies that could be reused across different datasets. Vocabularies Management service will be responsible for creating, storing, searching, discovering and reusing existing controlled vocabularies.

#### **2.1.1.3 Code Lists Management**

Code Lists are commonly used among RDF data cubes in order to assure unambiguous definition of the dimensions within a dataset. Moreover, code lists may have a hierarchical structure. Code Lists Management service will be responsible for creating, storing, searching, discovering and reusing existing code lists.

#### **2.1.1.4 Schema Management**

Data gathered during the cube building and modelling process using RDF Data Cube vocabulary allows to store information at the schema level (i.e. dimensions, attributes, measures). The reuse of the vocabularies and the code lists enables linking of disparate cubes. Schema Management service will be responsible for creating, storing, searching, discovering and reusing existing schemas.

#### **2.1.1.5 RDF Creation**

The RDF Creation process is responsible for transformation of the data (i.e. public sector statistical data) to Linked Data from various existing source formats. This process enables search, discovery and querying of relevant data. In particular the RDF creation service is responsible for definition of the model to represent the data, then transforming the data to fit that model. The core of the model is the RDF Data Cube Vocabulary<sup>2</sup>.

#### **2.1.1.6 Cube Builder**

The Cube Builder service provides interfaces that allows a user to map raw data to RDF Data Cube structure by identifying dimensions, measures and attributes in the raw data. With the support of other components (e.g. Schema Management, Metadata Management) it maps code lists and vocabularies.

---

<sup>1</sup> <https://www.w3.org/TR/vocab-dcat/>

<sup>2</sup> <https://www.w3.org/TR/vocab-data-cube/>

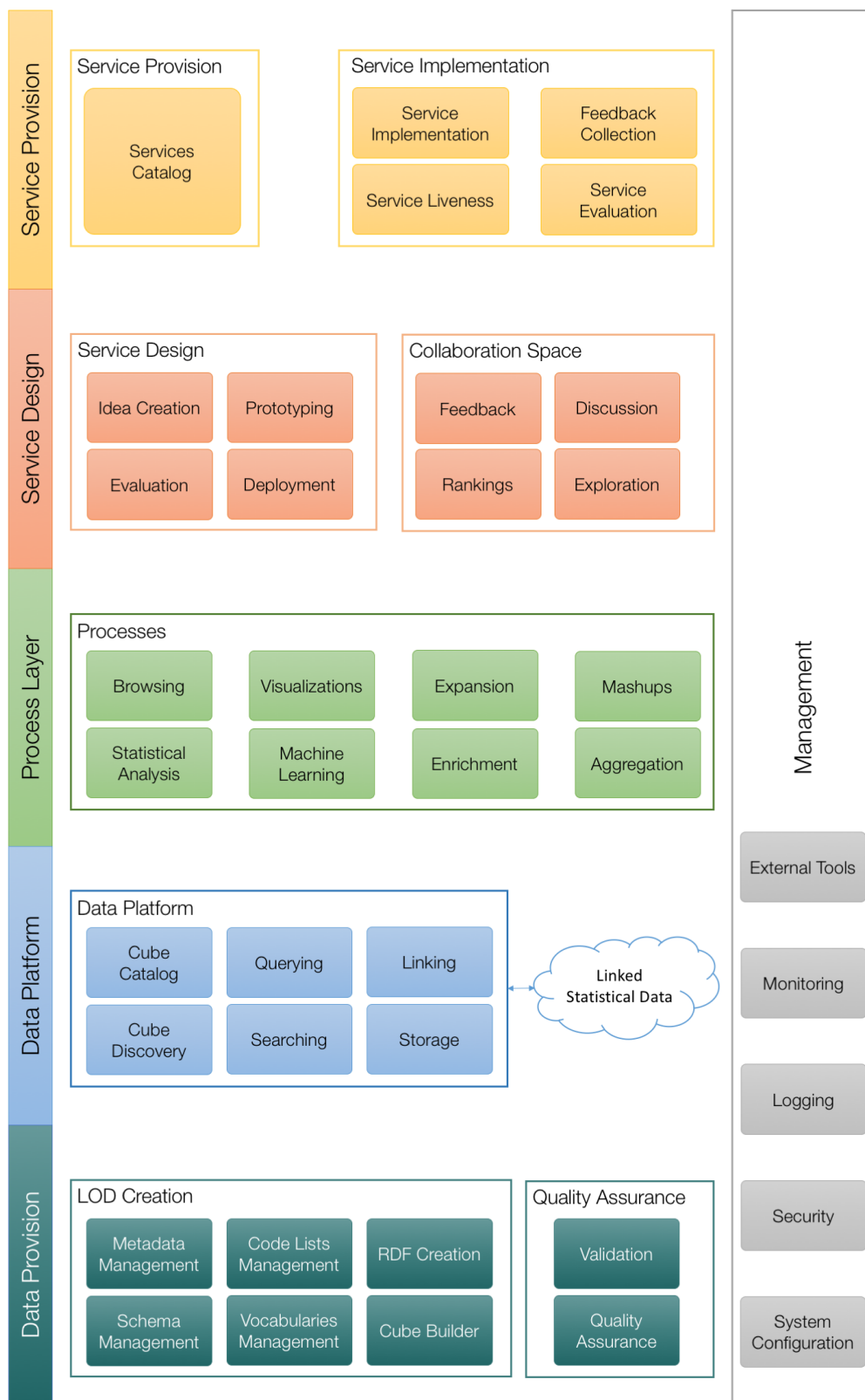


Figure 1 OGI Architecture

### 2.1.2 Quality Assurance

The Quality Assurance group is a set of activities, standards and processes that will ensure the validation as well as the high quality of the generated LOSD and their metadata and includes the following modules:

#### 2.1.2.1 Validation

Validation component is responsible for the validation of data cubes. It achieves this by verifying the schema requirements and detection of incorrect or missing values. Validation is performed on both data cube schema models and created data cube observations.

#### 2.1.2.2 Quality Assurance

Quality Assurance component is responsible for ensuring the quality of data and associated metadata. The component will use series of tests that can be applied in order to check the quality of the dataset and flag the information. This process tracks quality of data cubes and ensures that only high quality data is maintained inside the system.

## 2.2 Data Platform

The Data Platform Layer is responsible for the data storage and provides a catalogue service for available datasets that lets users search for aggregated cubes over many distributed repositories, based on their metadata. It consists of the following data driven processes:

### 2.2.1 Cube Catalogue

The Data Cube Catalogue provides an interface that enables browsing and exploration of the data cubes based on metadata associated with the data cubes. It lists available datasets and the user can search and filter results based on keywords, tags and categories etc.

### 2.2.2 Cube Discovery

Cube Discovery is the starting point for the services like Visualisation, Analysis, Mashups, etc. It allows users to navigate through the datasets and explore the contents of the datasets.

### 2.2.3 Querying

The Querying Service provides a user interface for writing and running SPARQL queries and previewing the results. Additional features might include syntax highlighting, a catalogue of stored queries, autocompleting, etc.

### 2.2.4 Searching

The Searching component provides a set of simple and advanced search and indexing capabilities, looking at the data itself, not just its metadata. Search service is essential for finding relevant data in large repository of datasets.

### 2.2.5 Linking

The Linking component helps in linking data to internal and external linked data resources.

### 2.2.6 Storage

The Storage component is responsible for persistence of the data and information. In particular it will store Data cubes outputted from the Data Provision Layer, mashups, outputs of the enrichment processes, analytics results, indexed parts of the RDF, metadata, user profiles, etc.

### 2.2.7 Linked Statistical Data Cloud

Linked Statistical Data represents the Linked Open Data available on the web (external to the Storage component).

## 2.3 Process Layer

The Process Layer allows an end user to interact with the system. It consists of the following:

### 2.3.1 Browsing

The Browsing component enables navigation through available data cubes, data cubes metadata and examination of the data cube observations' values.

### 2.3.2 Visualizations

The Visualization component involves meaningful interactive visualizations from the available datasets.

### 2.3.3 Expansion

The Expansion component enables the process of expanding a data cube with compatible data from elsewhere. If the compatibility rules are satisfied, this can be done in the following ways:

- Cube Expansion by adding additional measures,
- Cube Expansion by adding additional value to selected dimension.

The linking of measures of aggregated cubes results in larger cubes describing the same measure while the linking of the dimensions of different aggregated cubes facilitates data analytics and visualization services.

### 2.3.4 Mashups

The Mashups component let users combine data from multiple sources and create new services displayed in a single graphical user interface.

### 2.3.5 Statistical Analysis

Statistical Analysis module is responsible for range of statistical analysis functions and algorithms. It ranges from basic (e.g. standard deviation of a measure values against a dimension) to advanced (e.g. anomaly detection using multiple methodologies) methods which user can choose from.

### 2.3.6 Machine Learning

This component will use machine learning algorithms in order to mine and extract meaningful information about the data and hidden patterns as well as introduce intuitive visualisations.

### 2.3.7 Enrichment

The Enrichment component will enable linking of data cubes with external resources (e.g. DBpedia) and combination of data across statistical and non-statistical datasets. This process will allow to perform a richer data analysis and enhanced visualizations.

### 2.3.8 Aggregation

This component will compute aggregations across dimensions and hierarchies in selected data cubes. Aggregated values will be used for visualization, statistical analysis and machine learning purposes, thus enabling enhanced operations (e.g. roll-up/drill-down, dimension reduction etc.).

## 2.4 Service Design

Service Design implements the main functionalities offered by OpenGovIntelligence and enable the co-design of data-driven public services. It consists of the following:

### 2.4.1 Service Design

The Service Design Group of components helps stakeholders with proper ICT skills to be connected to data sources in order to design new services that attempts to address public needs. It includes the following modules:

#### 2.4.1.1 Idea Creation

The Idea creation component will support workflow for idea creation process aligned with OpenGovIntelligence framework. Platform will allow involvement of more than one stakeholder group in the creation process and support it by combining components available in the Process layer and the Data Platform layer.

#### 2.4.1.2 Prototyping

Prototyping enables rapid creation of the services prototypes and its dissemination to showcase proposed solutions.



#### 2.4.1.3 Evaluation

The Evaluation module allow to validate designed prototypes, collect feedback, improvement suggestions and additional features requests.

#### 2.4.1.4 Deployment

The Deployment component is responsible for the publication of the developed and validated solution.

### 2.4.2 Collaboration Space

This element will help to overcome space and time differentials and facilitate the co-design of data-driven public services using electronic communications and groupware by stakeholders coming from any country and/or time zone.

#### 2.4.2.1 Feedback

Feedback involves mechanisms that will be used to collect and manage the needs of the citizens, businesses and public authorities regarding the innovative data-driven public services.

It will include mechanisms and tools that will allow the OpenGovIntelligence stakeholders to provide comments (positive or negative), preferences, propositions and/or needs regarding:

- *data source* used by the service (i.e. enhancement of LOSD with new data, updated version of the data etc.).
- *exploitation methods* adopted in the co-produced public service (i.e. new types of data analyses or visualization that is not supported by a provided service)
- new services. For example a company could ask for the co-production of a new innovative service that will service its needs.

#### 2.4.2.2 Discussion

Discussion component enables stakeholders (co-designers of the services) to share thoughts, discuss new ideas, to brainstorm in sessions, chat and share assets in order to design new workflows that could generate new services.

#### 2.4.2.3 Rankings

Rankings component lets stakeholders to rate and rank the services based on i.e. usability and value added.

#### 2.4.2.4 Exploration

This module enables exploration of the functionalities offered by OpenGovIntelligence platform, available services, workflows and reusable components.

## 2.5 Service Provision

The Service Provision layer implements the co-provision of the public services that were previously co-designed in the Service Design layer. At this layer all modules are concerned with end user of the platform services.

### 2.5.1 Service Provision

#### 2.5.1.1 Services Catalogue

Services Catalogue enables discovery of the services deployed at the platform. It will provide a set of simple and advanced features for finding single service or group of services (e.g. by category or by type of the service).

### 2.5.2 Service Implementation

The Service Implementation Group includes the following modules:

#### 2.5.2.1 Service Implementation

This module allows end users of the platform to adopt and implement selected service using OpenGovIntelligence platform.

#### 2.5.2.2 Service Liveness

The main responsibility of this module is monitoring of the deployed service, providing performance and usage metrics and generating reports (i.e. errors and availability) for maintenance purposes.

#### 2.5.2.3 Feedback Collection

This component provides mechanism for collection of the feedback from end users of the services provided by OpenGovIntelligence platform. The outcomes may be used for user experience analysis and services evaluation for continuity, enhancements or elimination decisions.

#### 2.5.2.4 Service Evaluation

Service Evaluation component will evaluate deployed services based on multiple analysis factors (e.g. service liveness and feedback) in scheduled periods and provide factors, that may help to decide whether to continue, enhance or withdraw particular service.

## 2.6 Management

The Management Layer provides cross-platform functionalities such as logging, configuration, security, API etc. It is responsible for overall management and smooth functioning of the OpenGovIntelligence platform.

### 2.6.1 External Tools

OpenGovIntelligence platform provides APIs to external tools (i.e. data processing pipelines or external visualisation libraries) to let them user integrate OGI capabilities.

### 2.6.2 Monitoring

This component will provide information about liveness of the platform, usage performance, number of user of the platform and so on.

### 2.6.3 Logging

System logs will be used for tracking platform performance and resolve or debug issues if occurred.

### 2.6.4 Security

This component role is to ensure general security of the platform and provides authentication, authorization and user management capabilities.

### 2.6.5 System Configuration

System configuration is responsible for the management of configuration data.

**Note:** OpenGovIntelligence project will not develop software tools for every component of the architecture or an integrated system that matches this architecture as a whole. Moreover, some parts of the architecture might end up being human processes and some other instantiations of some other components-tools. Existing tools available for the purpose will re-used where ever possible and new tools will be developed when required. We have already identified a number of tools that can be re-used for example in collaboration space. This list of tools is included in deliverable D2.1 – OpenGovIntelligence framework – first release.

### 3 OpenGovIntelligence ICT tools - first release

This section provides a detailed description of the individual OpenGovIntelligence ICT tools developed during the first development stage of the OpenGovIntelligence project.

We describe the functionality and implementation of each of the tools.

In the first implementation stage, we have developed tools supporting data conversion and exploitation of the data. The next release of OpenGovIntelligence will include additional tools and will support a larger part of the OpenGovIntelligence framework.

#### 3.1 JSON API for Data Cube specification

The JSON-QB API is a standardised way for data users to programmatically select and retrieve data from a multidimensional statistical data cube. It is designed primarily for use by developers without specialist knowledge of statistics or specialist knowledge of the underlying RDF Data Cube representation. It supplements existing more complex ways of accessing data, such as via the SPARQL query language. The objective is to increase the ease of use of statistical data in visualisations and applications.

The OpenGovIntelligence is developing an openly licensed specification and implementation of the API. The longer term aim is that this could be offered by a wide range of data publishers as a method of exploiting their data and so creating greater impact. Data publishers could choose to use the OpenGovIntelligence implementation, or could create their own software that complies with the API specification, perhaps taking advantage of particular features of their own data platform.

##### 3.1.1 Functionality Description

The JSON-QB API provides programmatic access to data held in the open standard RDF Data Cube format, returning subsets of the data in a form that is easy for most programmers to work with.

##### 3.1.2 Implementation Description

The requirements for the tool are to provide access to multidimensional statistical data in a way that is familiar to a broad group of web developers, in particular to make it easy to create charts, maps and other visualisations. The design favours simplicity over completeness. As well as being easy to use, it should be relatively easy for data publishers to implement, and so have the maximum chance of becoming widely used standard.

##### 3.1.3 Availability

The specification can be accessed at <https://github.com/OpenGovIntelligence/json-qb>.

### 3.1.4 License

The licence of the specification is CC-BY 4.0 <https://creativecommons.org/licenses/by/4.0/>

### 3.1.5 Future development plan

The API is in its first experimental version. Feedback will be gathered from users of data and used to enhance the design and implementation.

### 3.1.6 Role in the architecture

JSON API for Data Cube specification can be represented as Querying component in Data Platform Layer.

## 3.2 JSON API for Data Cube implementation

This component is the implementation of the JSON API for Data Cube specification provided by the OpenGovIntelligence project. It aims to provide an easy to use API for web developers that use statistical data stored in the form of RDF Data cubes. The API implementation can be installed on top of any RDF repository and offer basic and advanced operations on RDF Data cubes assuming that: i) they are stored using the RDF Data Cube Vocabulary, ii) they follow a specific application profile (common practices) and iii) are accessible through a SPARQL endpoint. The results are always returned in JSON format.

### 3.2.1 Functionality Description

The API implementation offers functionality used by other components through simple REST calls. In the next tables we describe each function. Specifically, we provide the name of the function, a short description, the URI (including the REST method used e.g. GET), the parameters that the function requires as input, a sample JSON response and the SPARQL query that is executed to get the required results.

Name	Get all available cubes
Description	Returns all the available cubes of an RDF repository.
URI	GET /cubes
Parameters	-
Response	<pre>{"cubes":[   {"URI":"http://example.com/cube/unemployment",</pre>

	<pre> "label":"Unemployment rate" }, {"URI":"http://example.com/ cube/population", "label":"Population" }.... ] } </pre>
SPARQL query	<p>PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a></p> <p>PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a></p> <pre> select distinct ?cube ?label where {     ?cube rdf:type qb:DataSet     OPTIONAL{?cube skos:prefLabel rdfs:label ?label.}} </pre>

Name	Get dimensions
Description	Returns all the dimensions of a cube
URI	GET /dimensions
Parameters	dataset: the URI of the cube
Response	<pre> {"dimensions":[     {"URI":"http://example.com/dimension/refArea",     "label":"Reference area"     },     {"URI":"http://example.com/dimension/refPeriod",     "label":"Reference period"     }.... ] } </pre>
SPARQL query	<p>PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a></p> <p>PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a></p> <pre> select distinct ?dim ?label where {     &lt;http://example.com/cube/unemployment&gt; qb:structure ?dsd.     ?dsd qb:component ?cs.     ?cs qb:dimension ?dim.     OPTIONAL{?dim skos:prefLabel rdfs:label ?label.}} </pre>

Name	Get measures
Description	Returns all the measures of a cube

URI	GET /measures
Parameters	dataset: the URI of the cube
Response	<pre>{   "measures": [     {       "URI": "http://example.com/measure/employmentRate",       "label": "Employment Rate"     },     {       "URI": "http://example.com/measure/unemploymentRate",       "label": "Unemployment Rate"     }     ....   ] }</pre>
SPARQL query	<pre>PREFIX qb: &lt;http://purl.org/linked-data/cube#&gt; PREFIX skos: &lt;http://www.w3.org/2004/02/skos/core#&gt; select distinct ?measure ?label where {   &lt; http://example.com/cube/unemployment&gt;      qb:structure   ?dsd.                                      ?dsd qb:component ?cs.                                      ?cs qb:measure ?measure.                                      OPTIONAL{ ?measure      skos:prefLabel   rdfs:label ?label.}}</pre>

Name	Get attributes
Description	Returns all the attributes of a cube
URI	GET /attributes
Parameters	dataset: the URI of the cube
Response	<pre>{   "attributes": [     {       "URI": "http://example.com/attribute/unit",       "label": "Unit of measure"     },     {       "URI": "http://example.com/attribute/obsStatus",       "label": "Observation status"     }     ....   ] }</pre>
SPARQL query	<pre>PREFIX qb: &lt;http://purl.org/linked-data/cube#&gt; PREFIX skos: &lt;http://www.w3.org/2004/02/skos/core#&gt; select distinct ?attribute ?label where {   &lt; http://example.com/cube/unemployment&gt;      qb:structure   ?dsd.                                      ?dsd qb:component ?cs.                                      ?cs qb:attribute ?attribute.</pre>

	OPTIONAL{?attribute skos:prefLabel rdfs:label ?label.}}
--	---

Name	Get dimension values
Description	Returns all the values of a dimension that appear at a specific cube
URI	GET /dimension-values
Parameters	dataset: the URI of the cube dimension: the URI of the dimension
Response	<pre>{   "values": [     {       "URI": "http://example.com/value/female",       "label": "Female"     },     {       "URI": "http://example.com/value/male",       "label": "Male"     }     ....   ] }</pre>
SPARQL query	<pre>PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a> PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a> select distinct ?val ?label where {     ?observation qb:dataSet &lt;http://example.com/cube/unemployment &gt;.     ?observation &lt; <a href="http://example.com/dimension/sex">http://example.com/dimension/sex</a> ?val.     OPTIONAL{?val skos:prefLabel rdfs:label ?label}}</pre>

Name	Get attribute values
Description	Returns all the values of an attribute that appear at a specific cube
URI	GET /attribute-values
Parameters	dataset: the URI of the cube attribute: the URI of the dimension



Response	<pre>{   "values": [     {       "URI": "http://example.com/value/count",       "label": "Count"     },     {       "URI": "http://example.com/value/ratio",       "label": "Ratio"     }     ....   ] }</pre>
SPARQL query	<pre>PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a> PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a> select distinct ?val ?label where {     ?observation                                qb:dataSet &lt;http://example.com/cube/unemployment &gt;.     ?observation &lt;http://example.com/attribute/unit&gt; ?val.     OPTIONAL{ ?val      skos:prefLabel   rdfs:label ?label}}</pre>

Name	Get slice
Description	Returns a set of observations of cube that match to particular values of selected dimensions
URI	GET /slice
Parameters	dataset: the URI of the cube 0 or more dimension identifiers and the allowed values e.g. <a href="http://example.com/dimension/sex">http://example.com/dimension/sex</a> = <a href="http://example.com/value/female">http://example.com/value/female</a>
Response	<pre>{   "observations": [     {       "URI": "http://example.com/observation/1",       "http://example.com/dimension/refArea ": "<a href="http://example.com/Greece">http://example.com/Greece</a>",       "http://example.com/dimension/refPeriod ": "<a href="http://example.com/2015">http://example.com/2015</a>",       "http://example.com/measure/unemploymentRate": 0.24     },     {       "URI": "http://example.com/observation/2",       "http://example.com/dimension/refArea ": "<a href="http://example.com/Greece">http://example.com/Greece</a>",       "http://example.com/dimension/refPeriod ": "<a href="http://example.com/2016">http://example.com/2016</a>",       "http://example.com/measure/unemploymentRate": 0.26     }   ] }</pre>

	<pre> },.... ] } </pre>
SPARQL query	<pre> PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a> PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a> select  ?obs ?refArea ?refPeriod ?unemployment where {     ?obs                                     qb:dataset     &lt;<a href="http://example.com/cube/unemployment">http://example.com/cube/unemployment</a> &gt;.     ?obs                                     &lt;<a href="http://example.com/dimension/sex">http://example.com/dimension/sex</a>&gt;     &lt;<a href="http://example.com/value/female">http://example.com/value/female</a>&gt;.     ?obs                                     &lt;<a href="http://example.com/dimension/refArea">http://example.com/dimension/refArea</a>&gt;     ?refArea.     ?obs                                     &lt;<a href="http://example.com/dimension/refPeriod">http://example.com/dimension/refPeriod</a>&gt;     ?refPeriod.     ?obs     &lt;<a href="http://example.com/measure/unemploymentRate">http://example.com/measure/unemploymentRate</a>&gt; ?unemployment. } </pre>

Name	Get table
Description	Returns a 2D table representation of the cube's observations that match to particular values of selected dimensions
URI	GET /slice
Parameters	<p>dataset: the URI of the cube</p> <p>col: the URI of the dimension used for the columns of the 2D table</p> <p>row: the URI of the dimension used for rows of the 2D table</p> <p>0 or more dimension identifiers and the allowed values</p> <p>e.g. <a href="http://example.com/dimension/sex">http://example.com/dimension/sex</a> = <a href="http://example.com/value/female">http://example.com/value/female</a></p>
Response	<p>The response is in json-stat format e.g.</p> <pre> {"class" : "dataset",   "id" :   ["<a href="http://example.com/dimension/refArea">http://example.com/dimension/refArea</a>", "<a href="http://example.com/dimension/refPeriod">http://example.com/dimension/refPeriod</a>"],   "size" : [10,5],   "dimension" : {     "<a href="http://example.com/dimension/refArea">http://example.com/dimension/refArea</a>" : {       "category" : {         "index" :         ["<a href="http://example.com/Greece">http://example.com/Greece</a>", </pre>

	<pre> "http://example.com/UK",...]     }},     "http://example.com/dimension/refPeriod" : {     "category" : {         "index" : ["http://example.com/2015", " http://example.com/2016",...]     } }...,     "value" : ["0.24", "0.26" ...] //values are stored in row- major order } </pre>
SPARQL query	<pre> PREFIX qb: <a href="http://purl.org/linked-data/cube#">http://purl.org/linked-data/cube#</a> PREFIX skos: <a href="http://www.w3.org/2004/02/skos/core#">http://www.w3.org/2004/02/skos/core#</a> select  ?unemployment where {     ?obs                                     qb:dataSet &lt;http://example.com/cube/unemployment &gt;.     ?obs                                     &lt;<a href="http://example.com/dimension/sex">http://example.com/dimension/sex</a>&gt; &lt;http://example.com/value/female&gt;.     ?obs                                     &lt;<a href="http://example.com/dimension/refArea">http://example.com/dimension/refArea</a>&gt; ?refArea.     ?obs                                     &lt;<a href="http://example.com/dimension/refArea">http://example.com/dimension/refArea</a>&gt; ?refPeriod.     ?obs &lt;<a href="http://example.com/measure/unemploymentRate">http://example.com/measure/unemploymentRate</a>&gt; ?unemployment. } ORDER BY ?refArea ?refPeriod </pre>

### 3.2.2 Implementation Description

The implementation of the API uses the following technologies:

1. Jersey<sup>3</sup> framework for the implementation of the RESTful services. Jersey is an implementation of the JAX-RS<sup>4</sup> Reference Implementation.
2. SPARQL<sup>5</sup> for querying the RDF repositories.
3. Rdf4j<sup>6</sup> Java framework for processing RDF data
4. Json-stat java library<sup>7</sup> to create and consume json-stat data
5. Gson<sup>8</sup> Java library to serialize Java Objects into their JSON representation

<sup>3</sup> <https://jersey.java.net/>

<sup>4</sup> <https://jax-rs-spec.java.net/>

<sup>5</sup> <https://www.w3.org/TR/rdf-sparql-query/>

<sup>6</sup> <http://rdf4j.org/>

<sup>7</sup> <https://github.com/statisticsnorway/json-stat.java>

<sup>8</sup> <https://github.com/google/gson>

### 3.2.3 Availability

Source code is available at GitHub: <https://github.com/OpenGovIntelligence/json-qb-api-implementation>

### 3.2.4 License

The software is available as open source under the Apache License 2.0.

### 3.2.5 Future development plan

The plans for next year development include:

1. Support pagination of results. In the case of big cubes this will enable the limiting of returned result.
2. Support advanced filtering of dimension values. For example, enable filtering by ranges, patterns, filtering by geography etc.
3. Support multilinguality. In some cases, cubes contain labels in multiple languages. The API should be extended to support them.
4. Support hierarchical data. In some cases, dimensions have hierarchical structure. The API should be extended to support hierarchical data.

### 3.2.6 Role in the architecture

JSON API for Data Cube implementation can be represented as Querying component in Data Platform Layer.

## 3.3 Table2qb and Grafter

Grafter is an open source software library to support Extract Transform Load processes, with a particular emphasis on transforming tabular data to RDF. Its development has been supported by the EU FP7 DaPaaS and OpenCube projects, and further improvements have been implemented as part of OpenGovIntelligence. These include bug fixes, performance improvements and better error messages.

Grafter has been used to implement a process to convert statistical data to RDF Data Cube from a tabular input file in CSV format, that follows a particular template. We have called this software 'Table2qb' (pronounced 'table to cube').

The input template has one statistical observation per row, with a column for each dimension, attribute and measure. Column headers and cell contents are mapped to appropriate URIs for representing the data in RDF Data Cube. The format is designed to be simple for statisticians to generate, without the need for programming skills and without the need for knowledge of RDF.

Work is in progress to make the mapping from table contents to URIs more flexible and configurable, to improve the validation of user inputs and to define a standard way of associating metadata with the input file.

### 3.3.1 Functionality Description

Grafter provides a basis for a wide range of data transformation tasks, such as cleaning tabular data and transforming it into RDF.

The Table2qb tool, implemented with Grafter, takes data in a specific tabular structure, either as a CSV or Excel file, and converts it into an RDF Data Cube: representing the data as a series of observations with dimensions, attributes and measures, and generating the associated Data Structure Definition.

### 3.3.2 Implementation Description

Grafter is implemented using the Clojure programming language. Clojure is a functional programming language that runs on the Java Virtual Machine. This makes it efficient, able to run on all common operating systems, and able to make use of the large range of existing Java libraries.

Table2qb is implemented using Clojure, and the Grafter library in particular. The prime design criterion for Table2qb is that it should be simple for non-programmers to generate the input data in the required format, for example by using Excel or other spreadsheet software.

### 3.3.3 Availability

Grafter can be obtained from <https://github.com/Swirrl/grafter>. Table2qb is also on GitHub at <https://github.com/OpenGovIntelligence/table2qb>.

### 3.3.4 License

The software is available as open source under the Eclipse Public License.

### 3.3.5 Future development plan

Further work on both the Table2qb tool and the supporting Grafter library is planned over the next year. This will involve testing and evaluating Table2qb with a range of users and a range of datasets. The lessons learned will be used to improve the ease of use and flexibility. It is already known that improved validation and error reporting is a high priority area to work on.

### 3.3.6 Role in the architecture

Table2qb and Grafter can be represented as both: RDF Creation and Cube Builder components in Data Provision Layer.

## 3.4 Data Cube Builder

Data Cube Building enables RDF creation by the transformation of data sources in non-RDF formats to RDF Data Cubes (in particular conversion of the CSV files).

Created Data Cubes may be stored in the linked data space in order to link and enrich the data cubes with external linked knowledge based on pre-selected linkable dimensions.

RDF Data Cube creation process is using the corresponding schema structure.

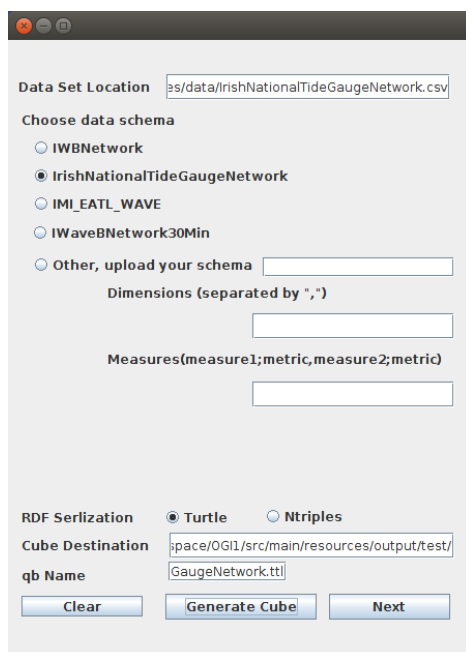
### 3.4.1 Functionality Description

Data Cube Builder can be used through the following interfaces: (A) Desktop UI, (B) Command Line, (C) Web Application and as (D) Web Service.

Application user is configuring the conversion process by passing required arguments through the desired interface. Data Cube Builder will output the RDF Data Cube regarding the arguments passed.

### 3.4.2 Implementation Description

Data Cube Building functionalities are written in Java and are extending Tarql (statistical observation capturing tool) by adding the corresponding cube schema. At this stage tool support data cube schemas for the Marine Institute pilot datasets. Jframe, Jcommander and Sparkjava are used to offer interfaces for the Cube Builder.



The screenshot shows a Java Swing window titled "Data Cube Builder". It contains the following fields and controls:

- Data Set Location:** A text field containing "ps/data/IrishNationalTideGaugeNetwork.csv".
- Choose data schema:** A group of radio buttons with the following options:
  - ☐ IWBNetwork
  - ☒ IrishNationalTideGaugeNetwork
  - ☐ IMI\_EATL\_WAVE
  - ☐ IWaveBNetwork30Min
  - ☐ Other, upload your schema (with an adjacent text field)
- Dimensions (separated by ","):** A text field.
- Measures(measure1;metric,measure2;metric):** A text field.
- RDF Serlization:** Two radio buttons: ☒ Turtle and ☐ Ntriples.
- Cube Destination:** A text field containing "space/OGI1/src/main/resources/output/test/".
- qb Name:** A text field containing "GaugeNetwork.ttl".
- Buttons:** "Clear", "Generate Cube", and "Next".

Figure 2 Cube Builder Desktop UI

## (B) Command Line,

Using the following set of arguments:

<b>Arguments</b>	<p>- Usage: OGI EU [options]</p> <p>Options:</p> <p>--csvFilePath, -csv CSV input file Location and name</p> <p>--dimOrMeasures, -l Customized Dim and Measures are Not Available at this stage</p> <p>--help, -help, -h Help Default: false</p> <p>--marineDatasetName, -schema Data Set Schema Currently Supporting (IWaveBNetwork30Min, IMI_EATL_WAVE, IrishNationalTideGaugeNetwork and IWBNetwork)</p> <p>--qbFileName, -qbN Cube output file Location and name</p> <p>--qbPath, -qb Cube output file Location and name</p> <p>--run, -r Which main class to Run! (cmd, webservice or desktop)</p> <p>--serialization, -format Output Cube RDF serlization format (turtle or ntriples) Default: turtle</p>
<b>Example</b>	<pre>\$mvn          exec:java          -Dexec.args="-run:cmd          - csv:src/main/resources/data/IWBNetwork.csv          -schema:IWBNetwork          - format:turtle -qb:src/main/resources/output/ -qbN:IWBNetwork.ttl"</pre>

## (D) Web Service.

<b>Arguments</b>	<pre>csv=inputFileNameAndLocation schema=marineInstituteDatasetId serialization=turtle qbPath=outputFileLocation qbName=outputFileName</pre>
------------------	--

<b>Example</b>	<code>/cubeBuilderAPI/cubeBuilderArgs?csv=inputFileNameAndLocation&amp;schema=marineInstituteDatasetId&amp;serialization=turtle&amp;qbPath=outputFileLocation&amp;qbName=outputFileName</code>
----------------	--

Sample data transformation output from row datasets in CSV format to RDF cubes:

<b>Dataset</b>	IrishNationalTideGaugeNetwork.csv
<b>CSV structure</b>	longitude,latitude,altitude,time,station_id,Water_Level,Water_Level_LAT,Water_Level_OD_Malin,QC_Flag  degrees_east,degrees_north,m,UTC,,m,m,m,  -6.0683,53.3915,0.0,2016-08-09T00:00:00Z,Howth Harbour,2.902,2.902,-0.06,0.0
<b>Cubiq structure of observations</b>	OGI:observations_51fa99a5-cb7b-4f76-afc2-a4016f9da8f0  rdf:type qb:Observation ; OGI:longitude "-6.0683" ; OGI:latitude "53.3915" ; OGI:altitude "0.0" ; OGI:time "2016-08-09T00:00:00Z" ; OGI:year "2016" ; OGI:month "2016-08" ; OGI:day "2016-08-09" ; OGI:station_id "Howth Harbour" ; OGI:water_Level "2.902" ; OGI:water_Level_LAT "2.902" ; OGI:water_Level_OD_Malin "-0.06" ; OGI:qC_Flag "0.0" ; qb:dataSet OGI:IrishNationalTideGaugeNetwork_ds .



### 3.4.3 Availability

The software can be obtained from <https://github.com/OpenGovIntelligence/data-cube-builder>

### 3.4.4 License

The software is available as open source under the Apache License 2.0.

### 3.4.5 Future development plan

The next development phase of Data Cube Builder will involve development of schema for other pilots partners (e.g. Lithuanian Pilot) and facility of custom schema usage.

Moreover, support for other raw data sources formats to be digested by the Data Cube Builder eg. JSON and TSV is planned.

### 3.4.6 Role in the architecture

Data Cube Builder can be represented as both: RDF Creation and Cube Builder components in Data Provision Layer.

## 3.5 Data Cube Explorer

Data Cube Explorer is responsible for listing, discovering, analysing and visualization data from the cubes. It serves as integrated user interface for the OGI platform. The tool is offering customizable statistical analysis and machine learning services using Data Cubes RDF as input data.

### 3.5.1 Functionality Description

Exploration, Data Cube Exploration and Visualization are served through web application build with Django<sup>9</sup> (High-Level Python Web Framework).

API, Receiving multiple kinds of requests eg. Data Cube Creation, and Data Cube Queries then routing those requests to the responsible module for processing.

### 3.5.2 Implementation Description

Data Cube exploration uses pivottable.js visualization and for computing summary of dataset. It is written in JavaScript and Django framework for server side operations .

User is able can choose one of the available dataset and select visualisation template (i.e. bar chart) and configure it by selecting dimensions and functions (i.e. aggregation or sum).

---

<sup>9</sup> <https://www.djangoproject.com/>

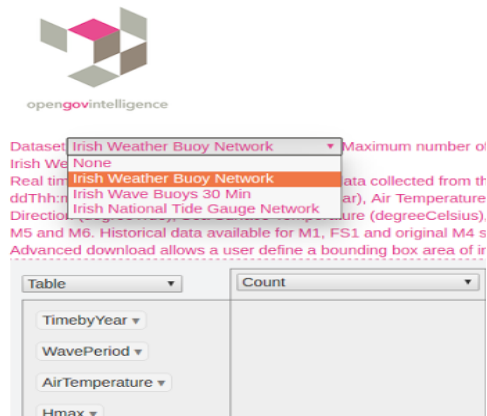


Figure 3 Data Cube Explorer - Dataset selection

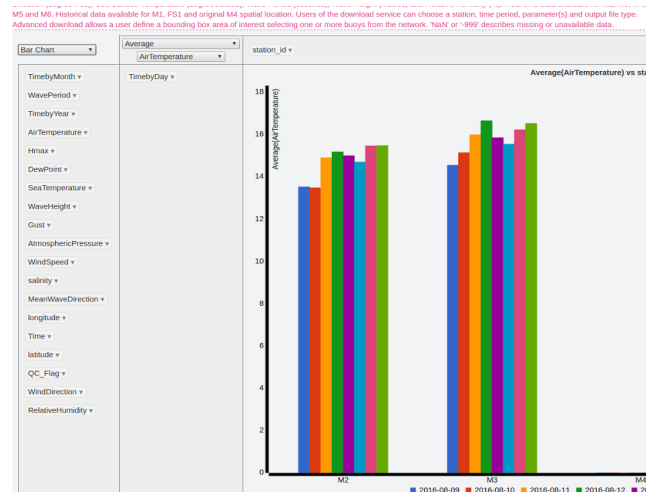


Figure 4 Data Cube Explorer - Sample Visualisation

In the current version of the tool, additional API was developed in order to query the database.

Type	Description	Function Name
Preset Queries	List available Linked Cubes	<code>JSONArray LqbQueryingForLqbSpaces (String limit)</code>
	List Linked Cube metadata	<code>JSONArray LqbQueryingForDimAndMeasures (String marineDatasetURI)</code>

	Retrieve Data of certain Linked Cube	JSONArray LqbQueryingForLqbData (String marineDatasetURI, String limit)
<b>Customized/ on-demand Queries</b>	Send Sparql Query over Linked Cubes	JSONArray LqbDirectQuerying (String sparqlQuery)

Data Cube API functionalities, currently available for Data Cube Exploration APIs:

API link	Description
listing available cubes	/cubeQueryingAPI/listLqbs?limit=numberOfRetrievedRecords
retrieve cube metadata	/cubeQueryingAPI/LqbMeta?dsuri=marineInstituteDatasetURI
retrieve data of certain cube	/cubeQueryingAPI/listdataofLqb?dsuri=marineInstituteDatasetURI &limit=numberOfRetrievedRecords
sparql endpoint	/cubeQueryingAPI/cubeQueryingArgs ?query=sparqlQueryToExecuteAgainstLinkedCubeSpace

### 3.5.3 Availability

The software is available at <https://github.com/OpenGovIntelligence/data-cube-explorer>

### 3.5.4 License

The software is available as open source under the Apache License 2.0.

### 3.5.5 Future development plan

Next step in the tool development will enable the usages of statistical and machine learning functions and algorithms over the data cube dimensions and attributes values (starting from basic

statistical exploration functions to a more complex machine learning predictive algorithms). WEKA<sup>10</sup> library is a candidate for providing the data mining capabilities.

Use of JSON API for Data Cube as communication channel with data store is one of the priorities in the development cycle.

### 3.5.6 Role in the architecture

Data Cube Explorer can be represented as multiple components in Process Layer: Browsing, Visualisation and Statistical Analysis components.

## 3.6 OLAP Browser

The OLAP Browser enables the exploration of RDF data cubes by presenting each time a two-dimensional slice of the cube as a table. The OLAP Browser is based on the “OpenCube OLAP Browser”<sup>11</sup>. “OpenCube OLAP Browser” was developed as a widget of Information Workbench platform<sup>12</sup>, in this new implementation we aim to disengage from the limitations that Information Workbench imposed and implement a more powerful and user friendly standalone tool. The limitations that Information Workbench include the “poor” user interface, bad performance, use of specific RDF repositories, bundled with Information Workbench etc.

The OLAP Browser builds upon the functionality offered by the JSON QB REST API implementation. Specifically, it calls the API functions and visualizes the returned JSON results. OLAP Browser support visualization of data represented in json-stat<sup>13</sup> format

### 3.6.1 Functionality Description

Currently the OLAP Browser supports the following functionalities:

1. List all the available cubes of an RDF repository. Make use of the `GET /cubes` function provided by the JSON QB REST API.
2. Present the structure (measures, dimension, attributes) of a selected cube. Make use of the `GET /dimensions`, `GET /measures` and `GET /attributes` functions of the JSON QB REST API.
3. Enable the selection of the dimensions (rows and columns) of the 2D table to be visualized.

---

<sup>10</sup> <http://www.cs.waikato.ac.nz/ml/weka/>

<sup>11</sup> <http://opencube-toolkit.eu/opencube-olap-browser/>

<sup>12</sup> [https://www.fluidops.com/en/products/information\\_workbench/](https://www.fluidops.com/en/products/information_workbench/)

<sup>13</sup> <https://json-stat.org/>

4. Enable filtering of fixed dimensions/attributes (i.e. the dimensions/attributes of the cube that are not shown in the 2D table). Make use of the `GET /dimension-values` and `GET /attribute-values` functions of the JSON QB REST API.
5. Presents in table the values of a two-dimensional slice of an RDF data cube. Make use of the `GET /table` function of the JSON QB REST API. The OLAP Browser presents the table based on the json-stat results returned by the API.

### 3.6.2 Implementation Description

The implementation of the API uses the following technologies:

1. D3.js<sup>14</sup> JavaScript library for easily manipulating and visualizing the JSON data returned by the JSON QB REST API.
2. JQuery<sup>15</sup> a cross-platform JavaScript library to simplify the client-side scripting of HTML.
3. Ajax to make REST calls to the JSON QB REST API, receive responses and update the web page without reloading.
4. Json-stat JavaScript library<sup>16</sup> to parse and consume json-stat data

### 3.6.3 Availability

Source code is available at GitHub: <https://github.com/OpenGovIntelligence/qb-olap-browser>

### 3.6.4 License

The software is available as open source under the Apache License 2.0.

### 3.6.5 Future development plan

The plans for next year development include:

1. Enable the integrated view of multiple compatible cubes on the fly without the need to create a new merged cube
2. Perform roll-up and drill-down OLAP operations
3. Perform dimension addition/reduction OLAP operation

### 3.6.6 Role in the architecture

OLAP Browser can be represented as multiple components in Process Layer: Browsing, Visualisation and Statistical Analysis components.

---

<sup>14</sup> <https://d3js.org/>

<sup>15</sup> <https://jquery.com/>

<sup>16</sup> <https://github.com/badosa/JSON-stat>

## 4 Conclusion

In this deliverable, we have described the components delivered as the result of the first OpenGovIntelligence development phase. The next release of OpenGovIntelligence will consist of a larger number of the tools supporting a more complete part of the OpenGovIntelligence framework. Existing tools available for the purpose will re-used where ever possible and new tools will be developed when required.

The next stage of the project will involve evaluation of the delivered components in use cases organized by the pilot partners.

## References

- [1] D1.1 OpenGovIntelligence challenges and needs*
- [2] D2.1 OpenGovIntelligence framework - first release*
- [3] D3.1 OpenGovIntelligence ICT tools - first release*
- [4] D4.1 Pilots and Evaluation plan*