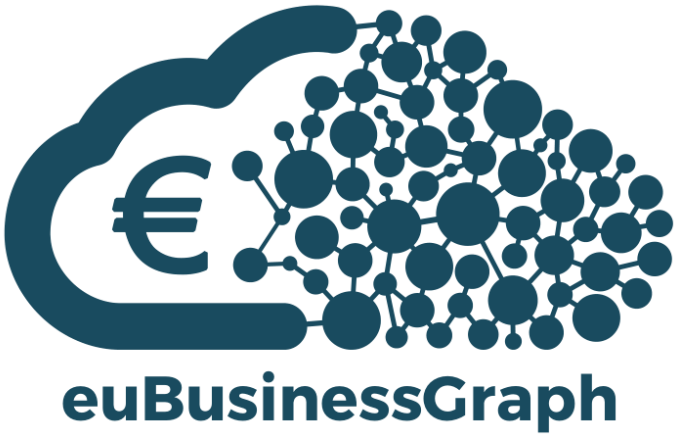
Innovation Action (IA)

ICT-14-2016-2017
H2020-ICT-2016-1

Enabling the European Business Graph for Innovative Data Products and Services



Deliverable D3.3:

Requirements Analysis, Architecture and API Specification for the euBusinessGraph Marketplace – v2

Date	28.03.2018
Author(s)	Brian Elvesæter (SINTEF), Bjørn Marius von Zernichow (SINTEF), Dumitru Roman (SINTEF), Matteo Palmonari (UNIMIB), Vincenzo Cutrona (UNIMIB), Flavio De Paoli (UNIMIB), Milena Yankova (ONTO), Matjaz Rihtar (JSI), Inna Koval (JSI), Miha Jenko (JSI)
Dissemination level	Public (PU)
Work package	WP3
Version	1.0

Document metadata

Quality assurors and contributors

Quality assuror(s)	Javier Paniagua (SDATI), Lars Tveit (EVRY)
Contributor(s)	euBusinessGraph Consortium

Version history

Version	Date	Description
0.1	19.02.2018	Initial Table of Contents (ToC).
0.2	23.02.2018	Updated structure after WP3 concall.
0.3	12.03.2018	Updated the introduction and overview to align with the business model for the marketplace.
0.4	16.03.2018	Updated requirements analysis and architecture specification.
0.5	26.03.2018	Integrated input from WP3 partners.
0.6	27.03.2018	Draft ready for internal peer review.
1.0	28.03.2018	Final formatting and layout.

Executive summary

The main goal of the euBusinessGraph project is to create the foundations of a European cross-border and cross-lingual business graph through aggregating, linking, and provisioning (open and non-open) high-quality company-related data, thereby demonstrating innovation across sectors where company-related data value chains are relevant. This is achieved by leveraging the power of technologies such as Data-as-a-Service and Linked Data.

The project will deliver a technology platform, the **euBusinessGraph Marketplace**, which aims to:

- Integrate, host, and sustain a scalable business graph data marketplace, with capabilities for data cleaning, enrichment, integration, interlinking, publication and hosting;
- Serve as an entry point for company-related data discovery, exploration and analytics; and
- Grow an ecosystem of 3rd party applications and company-related data services.

This report focuses on the **euBusinessGraph Marketplace** and presents:

- An updated **requirements analysis** for the **euBusinessGraph Marketplace**, the relevant stakeholders of the marketplace and their requirements in terms of capabilities (tools and services) of the marketplace aligned with requirements from the business cases in WP4 and the initial business model for the euBusinessGraph marketplace created in WP5;
- An updated **architecture and API specification** for the **euBusinessGraph Marketplace** in terms of the core components of the marketplace platform, and a set of APIs that will guide the development of the marketplace platform in the final phase of the project.

Table of contents

1 Introduction

This report presents Deliverable D3.3 "Requirements Analysis, Architecture and API Specification for the euBusinessGraph Marketplace – v2" of the euBusinessGraph project. This deliverable is developed as part of Work Package 3 (WP3) "euBusinessGraph Provisioning" and provides an updated version of Deliverable D3.1 based on further analysis of requirements from the business cases in WP4 and the initial business model for the euBusinessGraph marketplace created in WP5.

1.1 Objective

The objective of WP3 is to develop, integrate, deploy and maintain the technical infrastructure, methods, tools and services for reliable provisioning of the business graph by applying the Data-as-a-Service (DaaS) paradigm.

WP3 aims to adapt tools and approaches for tasks such as data transformation, cleaning and integration, enrichment and interlinking, storage and scalable querying, access control, methodologies for data publishing etc. The tools and approaches will simplify the business graph data publication and consumption process, and analytics tasks on top of the business graph.

The aim of this deliverable is two-fold:

1. To provide an updated **requirements analysis** for the **euBusinessGraph Marketplace**, the relevant stakeholders of the marketplace and their requirements in terms of capabilities (tools and services) of the marketplace aligned with requirements from the business cases in WP4 and the initial business model for the euBusinessGraph marketplace created in WP5;
2. To provide an updated **architecture and API specification** for the **euBusinessGraph Marketplace** in terms of the core components of the marketplace platform, and a set of APIs that will guide the development of the marketplace platform in the final phase of the project.

1.2 Relationships to other Work Packages and Deliverables

The development in WP3 follows an iterative approach resulting in two versions of the **euBusinessGraph Marketplace and Services** in month 12 and month 24 of the project as illustrated in Figure 1 below.

- Deliverable D3.1 (month 6) served as the specification for Deliverable D3.2 (month 12). The 1st version of the euBusinessGraph Marketplace and Services was based on initial requirements analysis from the business cases in WP4 (Deliverable D4.1), as well as some initial input from WP1 and WP2.
- Deliverable D3.3 (month 15) provides a revised specification for the 2nd version of the euBusinessGraph Marketplace and Services to be finalized in Deliverable D3.4 (month 24). This revised specification is based on the business case requirements (from Deliverable D4.1), dataset descriptions (from Deliverable D1.1), the design of the company model and identifier system (from Deliverable D2.1), and the initial business model for the marketplace (from Deliverable D5.2).

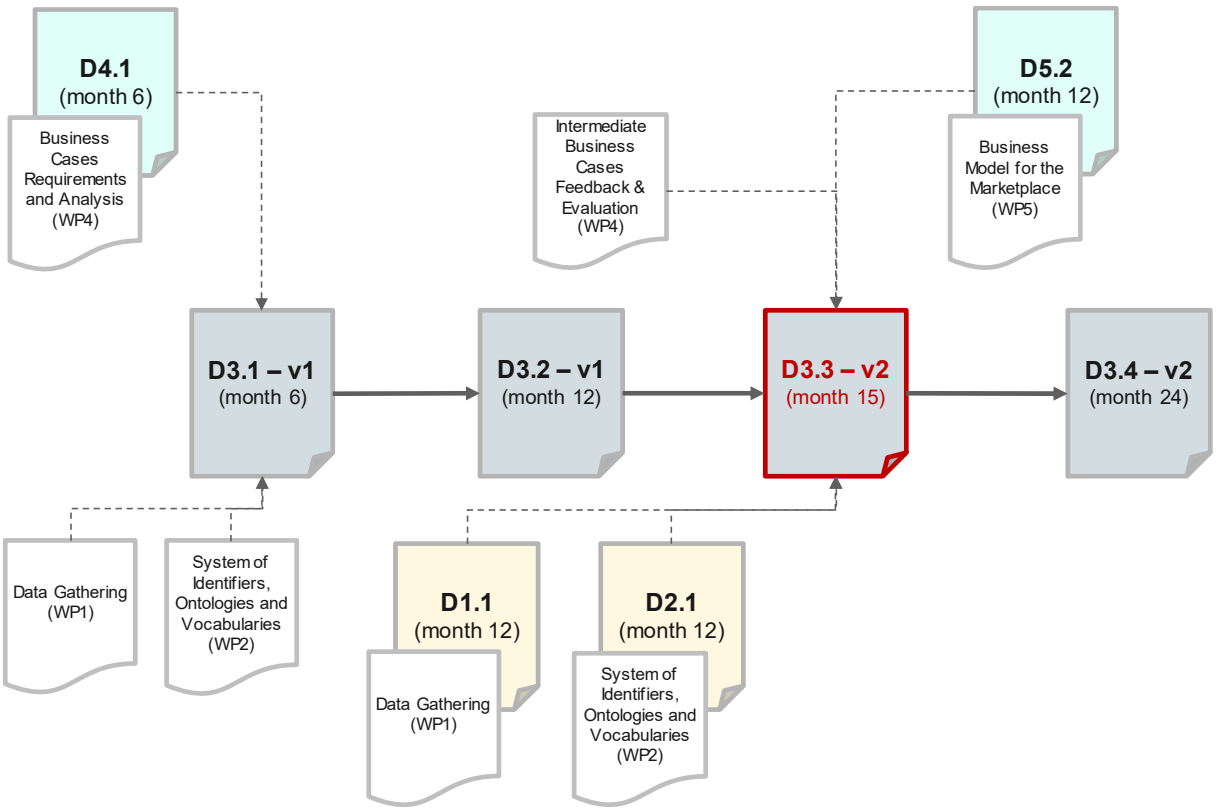


Figure 1: Relationship to other work packages and deliverables

1.3 Document structure

The remainder of this report is structured as follows.

- Section 2 provides an overview of the euBusinessGraph Marketplace and the relevant customer segments identified in the business model for the marketplace.
- Section 3 presents an updated requirements analysis for the euBusinessGraph Marketplace based on the customer segments identified in the business model for the marketplace.
- Section 4 outlines the revised architecture and API specification of the euBusinessGraph Marketplace platform services.
- Section 5 concludes this report and provides a summary of the development plans for the next period of the project.

2 euBusinessGraph Overview

This section gives an overview of the euBusinessGraph Marketplace and the relevant roles played in the euBusinessGraph context

2.1 Approach

An overview of the euBusinessGraph approach is provided in Figure 2 below, depicting the connection between data consumers and providers of the business graph data, through the enhanced environment developed in euBusinessGraph.

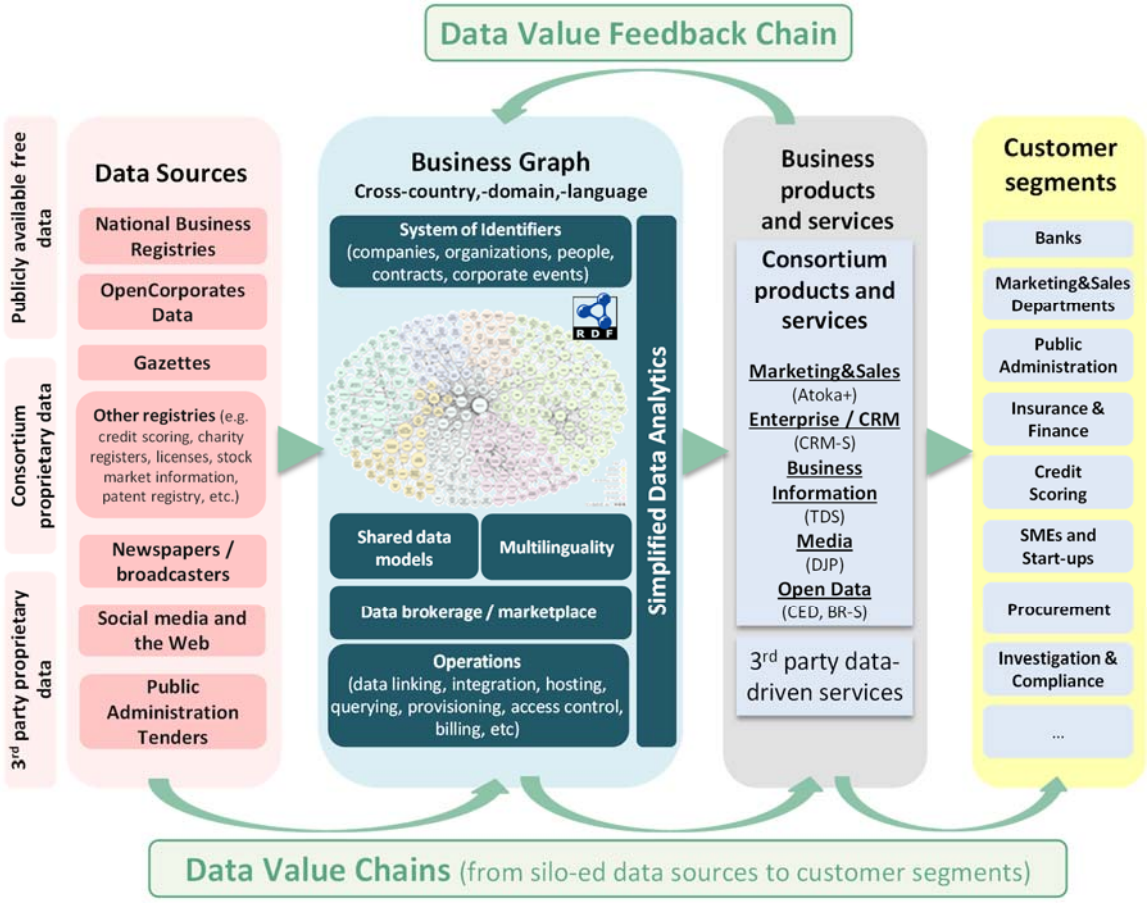


Figure 2: euBusinessGraph approach

The core element of the approach is the **business graph** component (second from the left in Figure 2). WP2 addresses the design of the business graph, which includes a system of identifiers for company-related data, shared data models and multilingual aspects. WP3 address the provisioning of the business graph, which includes support for data transformations, onboarding, hosting, access, analytics, data marketplace, and operations. In the process of transforming, integrating, publishing, and reusing data, two types of data value chains are enabled by euBusinessGraph:

- *Data value chain #1 – from silo-ed data sources to customer segments* (at the bottom of Figure 2): the value chain spans from the silo-ed data sources to various customer segments. The value is created through the set of capabilities provided by the business graph provisioning infrastructure, which are in turn leveraged to establish a set of innovative business products and services.
- *Data value chain #2 – data value feedback chain* (at the top of Figure 2): consists of the data insights generated by the proposed products and services being fed back into the business graph, therefore enhancing the value and scope of the data in the business graph. This is euBusinessGraph's mechanism to ensure that the business graph can host highly valuable and

high-quality information, while at the same time increasing the chances of a self-sustainable business graph.

2.2 Stakeholders

A stakeholder in this context represents a group or organization that has interests or concerns in the euBusinessGraph Marketplace. For the scope of the requirements analysis we consider three user segments identified in the Business Model for the euBusinessGraph Marketplace (see Appendix A). The three user segments are:

- **Data Providers:** This group of users has data which they would like to monetize, a data marketplace such as euBusinessGraph could provide a channel for them to achieve this. For example, an organisation which has information on outstanding invoices in their Enterprise Resources Planning (ERP) solution and provide this information to credit scoring companies at a fee.
- **Data Consumers:** This group of users require certain types of business-related data in the daily operation of their organisations to help them make better business decision. For example, an organisation gets credit scoring report at a fee to evaluate the strength of the financial footing of a potential customer or vendor. Such report can help them to analyse the risk they are being exposed to in every new business venture.
- **Service Providers:** This group of users can provide better visualisation of the data available on the data marketplace. They can potentially provide various data services on the vast amount of data available on a data marketplace such as data cleaning and data quality assurance.

2.2.1 Data providers

The **data providers** in euBusinessGraph are OCORP, CERVED, SDATI, DW, BRC and JSI. They provide the *data sources* (the left part of Figure 2) that will be offered via the business graph:

- *National Business Registries:* Data from authoritative business registries providing official information about companies at the national level (e.g., identifiers of companies, addresses, industry codes, etc.)
- *Gazettes:* Public records of company-related legal notices.
- *Other registries:* Data from registries created in various sectors such as credit bureau registries, charities, licenses, stock markets, patent registries, etc.
- *OpenCorporates:* Open data about over 92 million companies, primarily collected from public sources.
- *Newspapers/broadcasters:* Unstructured company-related data from media sources in various languages.
- *Social media and the Web:* Unstructured or semi-structured company-related data from companies' websites and social media.
- *Public Administration Tenders:* data about companies participating in public tenders.

2.2.2 Data consumers

The **data consumers** in euBusinessGraph aim to develop *business products and services* (the second from the right in Figure 2) that will access and use the data made available via the business graph for creation of data-driven products and services. Below we only list and provide a brief description of the business products and services to be developed in the business cases. Further details can be found in Deliverable D4.1.

- **OCORP Corporate Events Data Access Service (CED)** is a new product to provide cross-jurisdictional data and alerts about changes in companies, deriving these from official primary sources (primarily company registers and government gazettes), and making them available in a standardised form.

- **CERVED Tender Discovery Service (TDS)** is a new set of algorithms and services easing and facilitating discovery and participation of business companies in public administration tenders.
- **SDATI Atoka+** will extend the Atoka service, which currently only provides company-related data in Italy, to cover new jurisdictions, specifically company-related data in the United Kingdom and Norway.
- **EVRY CRM Service (CRM-S)** is a novel service utilizing machine-learning algorithms to deliver insights using data from the business graph to their customers' databases through an API.
- **DW Data Journalism Product (DJP)** is envisaged to be a web-based application that supports journalists in dealing with complex and large volumes of company related data across the three journalistic workflows: search, monitoring and content production.
- **BRC Norwegian Registries API Service (BR-S)** is a complete set of services to authoritative business data from Norway.

The **customer segments** (the first from the right in Figure 2) represent customers of the *business products and services*. The focus here is on the value created to the end users of the business products and services. This stakeholder is considered out of scope for the requirements analysis in WP3, as any requirements from such a stakeholder should be covered indirectly in the requirements from the business cases.

2.2.3 Service providers

The service providers in euBusinessGraph are SINTEF, OCORP, SDATI, ONTO, JSI and UNIMIB. They are the providers of the marketplace services such as system of identifiers, shared data models, multilingual support, data transformations, data onboarding, data hosting, data access, data analytics, data marketplace, and operations.

Additionally, analytics services that use data from the business graph can be offered as value-added services as part of the euBusinessGraph Marketplace. Some of the business products being developed in the business cases are candidate services to be included. One example is the credit score service being developed in the EVRY CRM-S business case.

3 Requirements Analysis

This section provides details on the requirements analysis and the resulting requirements for the euBusinessGraph Marketplace platform.

3.1 Data providers requirements

For the data providers (OCORP, CERVED, SDATI, DW, BRC and JSI), the requirements analysis were based on:

- Discussions through meetings and concalls.
- Requirements input collected for each business case in the project's Wiki collaboration platform.
- Analysis of the business cases descriptions in Deliverable D4.1.
- A set of initial dataset templates as part of Work Package 1 (WP1). The dataset templates covered aspects such as:
    - Data collection – how the data is collected, e.g., mandatory by law;
    - Data structure – description of the data structure;
    - Standards and metadata – standards followed and metadata available;
    - Dataset license and availability – Under which license the dataset is made available;
    - Data access – how the data can be accessed – e.g. queries endpoints, RESTful APIs, original data import, database dump, etc.;
    - Size, frequency, coverage and language – dataset size, update frequency, time and spatial coverage and language;
    - Data identification – the identifier used for the data;
    - Data privacy and sharing – is there sensitive datasets or parts of the datasets that should remain private, what kinds of access restrictions should be implemented – both for input datasets and the output (integrated) dataset for the use case;

These inputs were analysed with respect to technical requirements for the euBusinessGraph Marketplace platform. The technical focus for this analysis was on the data provider role that aims to share its data to the business graph. The results of this analysis has been summarised into the requirements table below.

Table 1: Data provider requirements (DPRs)

ID	Requirement	Requirement description
Data Preparation Services		
DPR-01	Dataset import	Dataset import for relevant data formats (e.g. RDF, CSV, JSON, XML, REST service).
DPR-02	Dataset cleaning and transformation	Data cleaning and transformation activities (RDF-ization).
Data Interlinking Services		
DPR-03	Entity linking	Entity linking get as input a text and produces a set of annotations.
DPR-04	Semantic labelling	Semantic labelling get as input a (weakly) structured source and produces a set of annotations that are used for generating mappings.
DPR-05	Link discovery	Link discovery get as input a knowledge graph and produces a set of mappings.

Data Hosting Services		
DPR-06	Data access	Specifying access through different channels (e.g. SPARQL endpoint, original data download, REST APIs, reporting service).
DPR-07	Data updates	APIs for accessing and modifying (updating) datasets.
DPR-07-a	API for incremental update	API for incremental update of data.
DPR-07-b	API for bulk update	API for bulk update of data.
DPR-08	Dataset metadata	Management of metadata, such as standardized name, description, language, including company-specific extensions, such as jurisdictions.
DPR-08-a	Common vocabulary	Ability to describe data through a common/standard vocabulary.
DPR-09	Big data storage	Storage capability for large data volumes (RDF).
Cross-Cutting Business Cases Analytics Services		
DPR-10	Multi-lingual annotation	Support for multi-lingual annotation of text with links to relevant/common concepts.
Marketplace and Operational Services		
DPR-11	License models	Support for different license models for accessing and APIs for accessing and modifying datasets, including dual license models that allows for both payment plus share-alike for public-benefit use.
DPR-11-a	Dataset-level license access control	Access control policy at dataset-level.
DPR-11-b	Data item license access restrictions	Access control policy at data-item level.
DPR-11-c	Advertise company data	Ability to advertise private graphs and direct consumers to them.
DPR-11-d	Shared revenue	Revenue originating from data flows are shared.
DPR-12	Security and access control	Security and access control for users and groups.
DPR-13	User registration and access management	Manage access of users to APIs and groups.
DPR-14	Secure access policy specification	Specify policy through different API keys for different users or groups.

3.2 Data consumers requirements

For the data consumers (CERVED, SDATI, EVRY and DW), the requirements analysis were based on:

- Discussions through meetings and concalls.
- Requirements input collected for each business case in the project's Wiki collaboration platform.
- Analysis of the business cases descriptions in Deliverable D4.1.

It should be noted that amongst the business cases there are partners that represent both the data provider and data consumer roles (i.e., CERVED, SDATI and DW) and thus see things from both perspectives. The result of this analysis has been summarised into the requirements table below.

Table 2: Data consumers requirements (DCRs)

ID	Requirement	Requirement description
Data Hosting Services		
DCR-01	Multiple dataset programmatic access channels	Access through different channels (e.g. SPARQL endpoint, original data download, REST APIs, reporting service).
DCR-01-a	Data dump	Ability to download large volumes of data as data dumps.
DCR-02	Searching and exploring existing datasets	Dataset search and browse/explore functionality.
DCR-02-a	Single access point	Single access point to information about company data.
DCR-02-b	High availability and efficient querying	Efficient querying based on indexation of the data based on pre-defined sets of indexes. High availability should be ensured for the indexed data by the hosting platform.
DCR-02-c	Extended company profile	Ability to access extended company profiles that integrates data from multiple data sources.
DCR-03	Access to dataset metadata information	Access to metadata information.
DCR-03-a	Detailed information about data	Detailed information about data that can be found in other data repositories.
Cross-Cutting Business Cases Analytics Services		
DCR-04	Multi-lingual annotation	Support for multi-lingual annotation of text with links to relevant/common concepts.
DCR-05	Event	Support for event detection.
DCR-06	Graph based analytics	Support for clustering and similarities amongst entities, e.g. resolving ambiguity.
DCR-07	Relation extraction	Support for extracting relations between entities.
Marketplace and Operational Services		
DCR-08	License models	Support for different license models for accessing and APIs for accessing and modifying datasets.
DCR-08-a	Shared agreement models	Integrated navigation to data with shared business/revenue agreements
DCR-09	Secure access to platform APIs	API keys for specifying access policies for different users.
DCR-10	User registration and access management	User sign-up, log-in and profile management.

3.3 Service providers requirements

The service providers requirements are based on needs for the data analytics services and results related to the development of the system of identifiers and the common data model in WP2.

3.3.1 Data analytics

The data analytics of the euBusinessGraph Marketplace platform are supported by the *Cross-Cutting Business Cases Analytics Services* that are specified in sections 4.4. Below we present user scenarios describing typical usage of these services:

- **Wikifier (with extra vocabularies):** The user would like to annotate a business related document. The standard Wikifier set up is providing back semantic annotations with Wikipedia concepts. However, for specific business related use-cases additional business entities present in document are of special interest. Extra vocabularies allow for cross-lingual business related semantic annotation that include companies from the long tail of national registers and products that are not described in Wikipedia. In such way, including extra vocabularies, we are able to extract relations between different business entities.
- **Event Registry (extended version for business environments):** The current Event Registry set up is based on event categorization with DMOZ taxonomy. However, DMOZ was recently closed (March 17th, 2017) and the project is no longer supported. Assuming we have the following scenario for Event Registry – the user would like to get a set of merger and acquisition events. The Event Registry (for business scenarios) would include a specific event type for each business related event (such as “Mergers and acquisitions”) along with a list of articles in different languages for each event.
- **Graph-based analytics:** The user would like to get listed mergers and acquisitions for specified company. By processing a set of news articles graph-based analytics is able to provide a number of relevant companies, along with information about the source where the merger and acquisition has been listed.

3.3.2 Common company data model

For sharing of data in the business graph the project and the data providers in the project has agreed to adopt a model in which a **common data subset is shared to the business graph**. Data providers joining the business graph share a common subset of their data described according to the **common company data model** defined in Deliverable D2.1. The model covers the following requirements:

- Capture the concept of a company and represent different types of companies.
- Represent company jurisdictions and registration information.
- Capture company contact information, such as the address and other locations.
- Capture social data of companies, such as their websites (together with Web languages), RSS/Atom feeds and Wikipedia URLs.
- Answer if a company is publicly traded or not, if it is state owned or not, and if it is registered in a startup register.
- Support languages: EN, IT, NO

3.3.3 System of identifiers

Companies are registered using various kinds of identifiers. Some of these identifiers are kept in registers and some others are self-issued and not centralised. The core company model can represent several types of identifiers:

- Official registration in a trade register: this registration should correspond to the company's jurisdiction and it is used to establish the legal existence of the company.
- Other official government registers for specialised purposes (e.g., bank license, insurance company license, register of startup companies)
- Official international registries (e.g., GLEI)

- Registration to non-official company databases (e.g., OCORP, SDATI, Wikidata, Dun & Bradstreet)
- Social networks (e.g., Facebook, Twitter)

To support multiple identifiers, for diverse purposes, the model has the following characteristics:

- the *Company* class can link to several *Identifier* instances
- the *Identifier* class is associated to an *Identifier System*.
- the *Identifier System* class models the different cases in which identifiers are handed to a company.

The *Identifier System* class describes characteristics of an identifier system that reflect whether the system has an identifier database, can be used to uniquely identify a company, has official character and in which jurisdictions and which are the rules used to determine if identifiers are valid within that system. Moreover, the Identifier System class also encodes information about the agents in charge of creation and maintenance, issuing and publishing of identifiers and information about web resources that can be used to search, browse and retrieve identifier information.

The *Identifier* class holds information about the identifier value and also about its lifecycle (issue date and expiration date, in case the identifiers in the system can expire).

Finally, each *Company* instance can have a redundant *official identifier* link so the model is compatible with the Registered Organisation vocabulary.

The fact that the current version of the model allows multiple identifiers of a company enables its use as a tool for matching and linking data from different data sources about the same company. To provide efficient search, matching and filtering, we envision the business graph to consist of a backbone that will function as an index, and link company data from various data providers to define a (virtual) business graph of company data. This is illustrated in Figure 3 below:

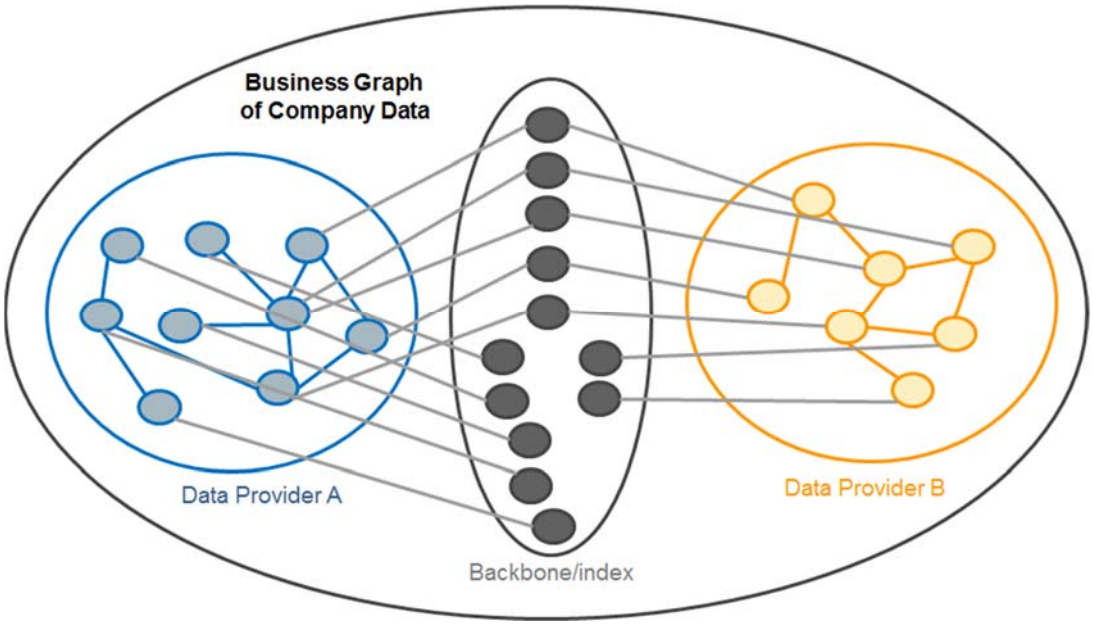


Figure 3: Connecting identifiers with the euBusinessGraph system

The backbone/index should contain entries about all companies that are available from the different data providers through the business graph. Each company entry in the backbone/index should contain all identifiers that are supported by the business graph. This enables the aggregation of data about the same company from different data providers, as shown in Figure 3.

4 Architecture and API Specification

This section describes platform architecture covering the data preparation, data interlinking, data hosting, cross-cutting business cases analytics, and marketplace and operational services. For each of these service categories we are planning on developing a set of features as shown in Figure 4 below. The services have been categorised according to the tasks in WP3, i.e., data preparation, data interlinking, data hosting, cross-cutting business cases analytics, and marketplace and operational services.

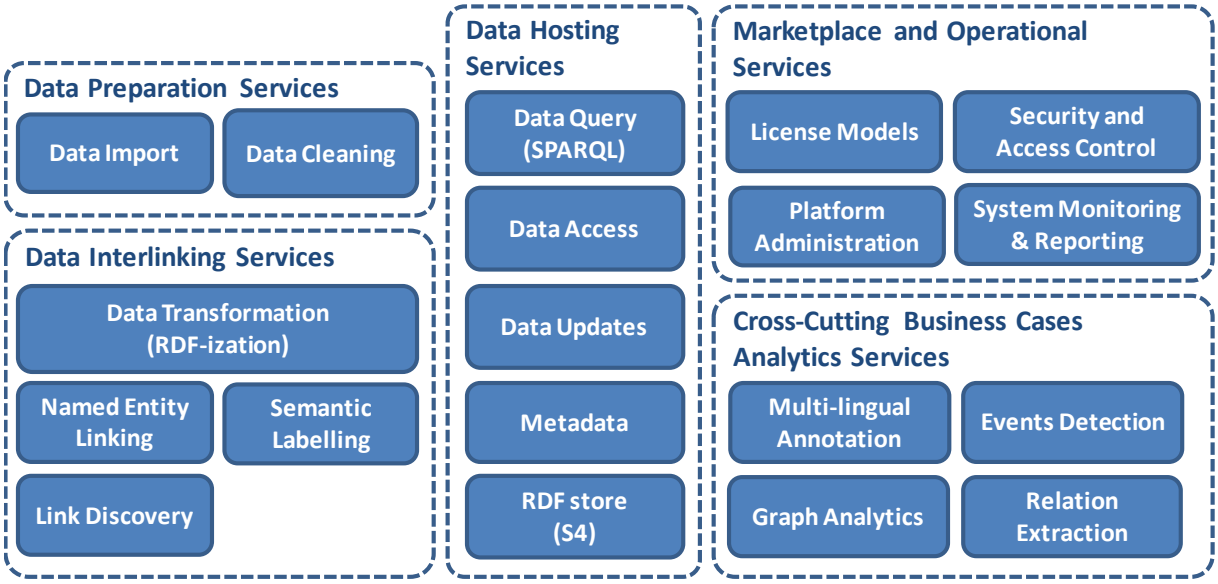


Figure 4: Platform services (revised)

In Deliverable D3.2 we presented the first release of the software components implementing the platform services. For each main service category section below we list the software components that we are developing and how they map to the services in the architecture shown in Figure 4. In Figure 5 below we give a high-level mapping of the software components maps to the different service categories.

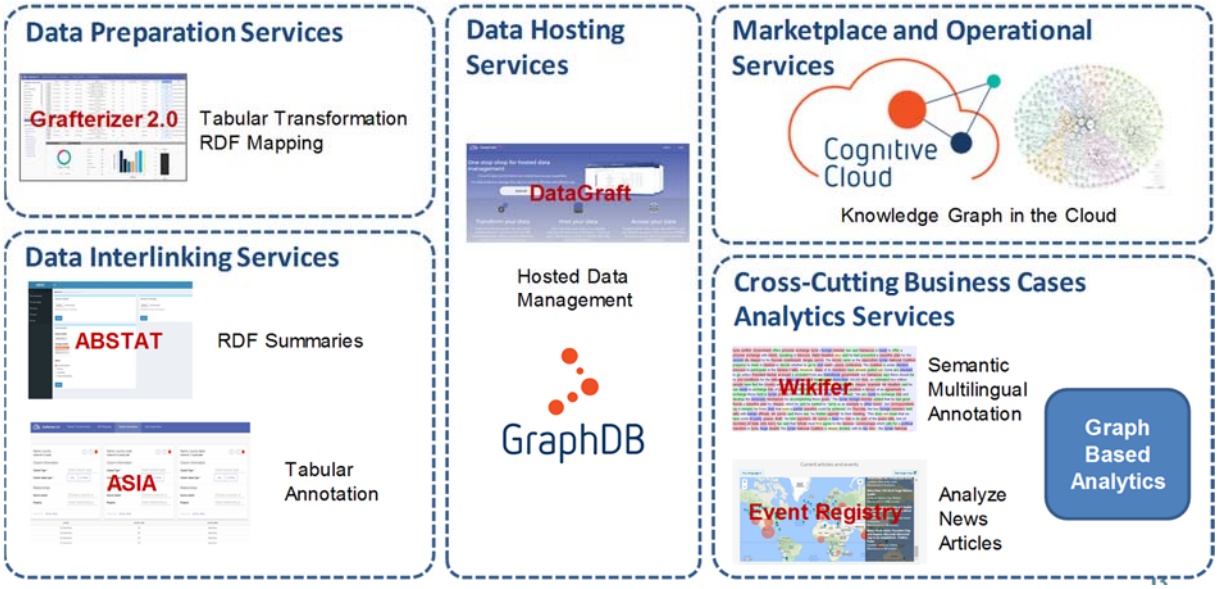


Figure 5: Platform components (revised)

For each software component we provide a more detailed description containing the sections from Table 3 below.

Table 3: Software component description template

Subsection	Description
Requirements	Short description of the relevant requirements from the business use cases being addressed by this software service/component.
Architecture	Architecture description of the implemented software service/component. Architecture component diagram.
Functionality	Description of the functionality provided by the software service/component. Screenshots of GUI if possible.
APIs	Short API description and links to online API documentation.
Software license and source code repository	Software license. For open source software (OSS) links to source code repository.
Installation and user guide	Short installation description and user guide. Links to online user guide.
Next steps (2nd release)	Short description of the development plan for the 2nd release of the software service/component in month 24.

4.1 Data Preparation Services

Table 4 lists the software components and how they map to the Data Preparation Services of Figure 4.

Table 4: Software components – Data Preparation Services

Software component	Implemented service	Implementation status
Grafterizer 2.0	Data Import	Grafterizer 2.0 currently supports import of CSV files. Files and transformations can be stored, shared, and reused in DataGraft.
Grafterizer 2.0	Data Cleaning	The current implementation of Grafterizer 2.0 includes functionality for suggestion-based tabular data cleaning, visual data profiling for data quality assessment, and semantic annotation of CSV data to RDF knowledge graphs.

4.1.1 Grafterizer 2.0

Grafterizer[1] is a web-based framework for tabular data cleaning and transformation, and RDF mapping. For deliverable 3.2, Grafterizer 2.0 implements an effective approach for visual data profiling that simplifies the process of preparing tabular data, and contributes to improving data quality. Moreover, a prototype service has been implemented (see section 3 – Data Interlinking Services) in Grafterizer 2.0 that supports semantic labelling of (weakly) structured CSV data sources.

Grafterizer 2.0 is an integrated framework that provides functionality for the following services:

1. **Data import and data cleaning:** Service that provides functionality for cleaning and transformation of tabular CSV data.
2. **RDF mapping:** Service that supports mapping of tabular CSV data to RDF knowledge graphs in a graphical tree-based interface.
3. **Tabular annotation:** Service that provides semantic annotation of tabular CSV data to RDF knowledge graphs (ASIA/ ABSTAT), section 3.

Section 2 of this report covers the implementation of visual data profiling capabilities and suggestion-based data cleaning and transformation, while Section 4.2 includes the integration of semantic tabular

[1] https://datagraft.io/

annotation (ASIA/ABSTAT). Tabular annotation provides an alternative approach to the existing RDF mapping functionalities in Grafterizer.

4.1.1.1 Requirements

User feedback shows that Grafterizer has a steep learning curve, and is complex to use. To meet the requirements of euBusinessGraph to provide data preparation services that simplify cleaning and transformation processes, one of the goals of D3.2 has been to extend the current capabilities of Grafterizer by implementing a visual data profiling approach that will contribute to improved data quality, and ease the generation of business graph data. As a baseline for this implementation, Figure 6 shows the Grafterizer user interface at the time of D3.1. The user experience was characterized by a lack of overall assessment of data quality, difficulties in finding relevant data cleaning functionality, and limited interaction with the data table view.

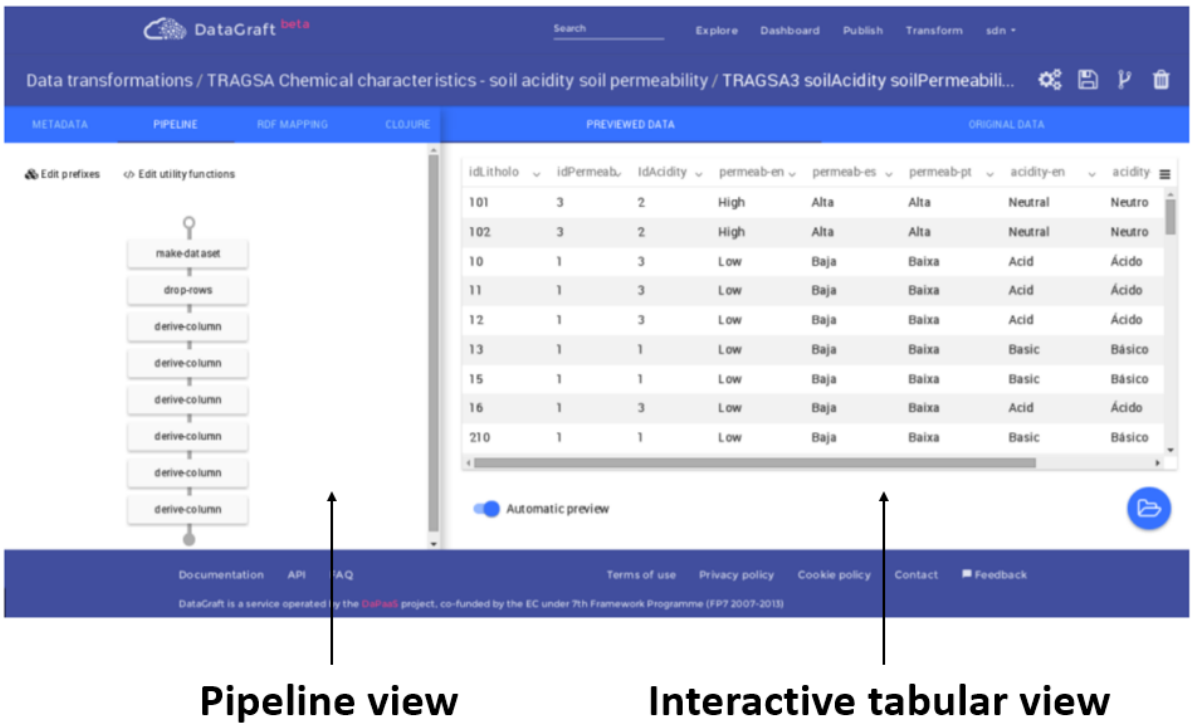


Figure 6: Grafterizer user interface at the time of D3.1

Visual data profiling is the statistical assessment of datasets to identify and visualize potential quality issues such as data outliers or missing data values. The motivation behind the implementation of a visual data profiling approach, is that the large datasets in euBusinessGraph require statistical assessment and analysis to achieve proper data quality. Visual data profiling has the potential to help data scientists make an informed decision on how to deal with data quality issues. Grafterizer 2.0 implements this visual data profiling approach that provides capabilities such as a table view interface that lets the user manipulate columns and rows directly. Furthermore, relevant data cleaning and transformation functionality appropriately addresses the goals that the user tries to achieve.

To facilitate the requirements process, a wireframe was created to describe the user interface and functionality, and the needs of users, that lead to a set of requirements.

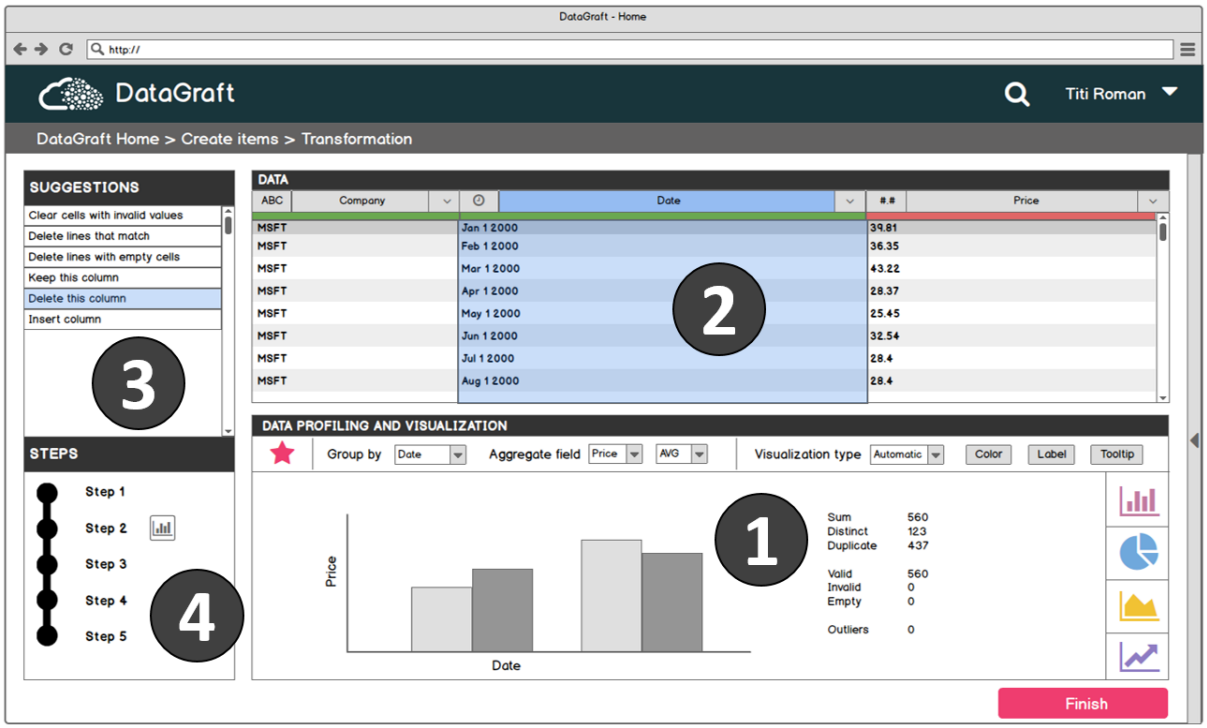


Figure 7: Grafterizer visual data profiling approach wireframe

The user interface of the visual data profiling approach illustrated in the wireframe in Figure 7, consists of the following main components and capabilities:

- A **visual data profiling** component (Figure 7, component 1).
- A tabular **table view** that provides data cleaning and transformation functionality (Figure 7, component 2).
- A sidebar that **suggests relevant data cleaning and transformation** actions to the user (Figure 7, component 3).
- A **steps pipeline** that reflects applied data cleaning and transformation steps (Figure 7, component 4).

The functionality of the visual data profiling approach in Figure 7 can be described in a sequence of steps that illustrates the visual data profiling cycle. As an example, consider a dataset with stock prices at various dates for five different companies (i.e. as shown in Figure 7):

1. The column 'Date' (Component 2, middle column, highlighted) is selected in the table view.
2. Suggested transformations display in the 'Suggestions' sidebar (Component 3).
3. The 'Data profiling and visualization' view provides a statistical assessment and profile of the data for the selected column 'Date' (Component 1). A statistical profile of the data gives useful information to the user that can make an informed decision on how to deal with data quality issues in the dataset.
4. Optional: User may select any sections of the visualizations that will further suggest transformations for that specific section only, e.g., if the user selects a specific range of dates, the suggested transformations will be valid for this range only. This approach reduces the time taken to assess all the transformations that are applicable in that range of data.
5. Selecting a suggested transformation will add a transformation step to the pipeline and the table view will update to reflect the changes (Component 4).
6. Repeat steps 1–5 to continue profiling, cleaning and transforming the dataset. The transformation sequence can be saved, and shared, for later reuse. The reuse of transformation sequences is particularly useful for periodical batch operations on data.

The functionality described in Figure 7 can be summarized as a set of requirements in Table 5 below.

Table 5: Grafterizer requirements

Requirements of the visual data profiling approach		Type
R1	Provide visual data profiling capabilities	**F**
R2	Provide data cleaning and transformation functionality	**F**
R3	Provide data cleaning and transformation suggestions	**F**
R4	Provide a pipeline that reflects applied data cleaning and transformation steps	**F**
R5	Provide a solution that is useful to the user	**NF**
R6	Provide a solution that is easy to use	**NF**
	Functional requirement	**F**
	Non-functional requirement	**NF**

4.1.1.2 Architecture

The high-level system architecture is based on a microservice architecture, and uses the design principles of Separation of Concerns (SoC). SoC is traditionally achieved in layered architectures, e.g. in a 3-Tier architecture, by defining interfaces and encapsulating information. A 3-tier architecture (i.e. Figure 8) would separate concerns into a presentation layer, an application tier, and a data layer.

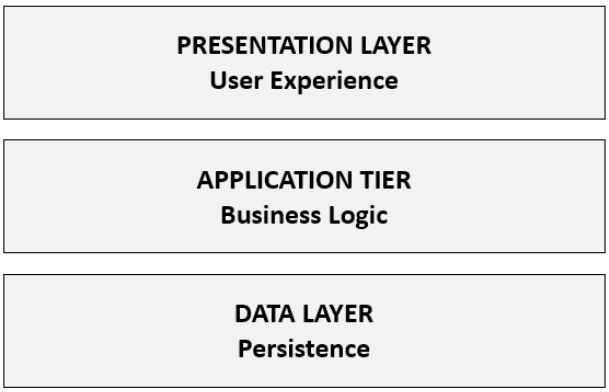


Figure 8: 3-tier architecture

A microservice architecture would take the SoC one step further by dividing the application tier and data layer into separate, domain-driven services that would operate autonomously from other services. A network-protocol would provide secure end point access to the services. While the SoC in a layered architecture is horizontal, the SoC in a microservice architecture would be both horizontal and vertical.

Based on the considerations above, the following architecture in Figure 9 represents a microservice approach that implements the design principles of SoC both horizontally and vertically. Angular 2 is selected as development framework for building the visual data profiling prototype, and the architecture is based on Angular 2 best practices for architectural patterns.

The green boxes in Figure 9 illustrate the backend services (application tier), while blue boxes show the frontend presentation layer. The data layer is currently omitted from the figure, but will be implemented in D3.4 as an integration with existing relational and graph databases in DataGraft.

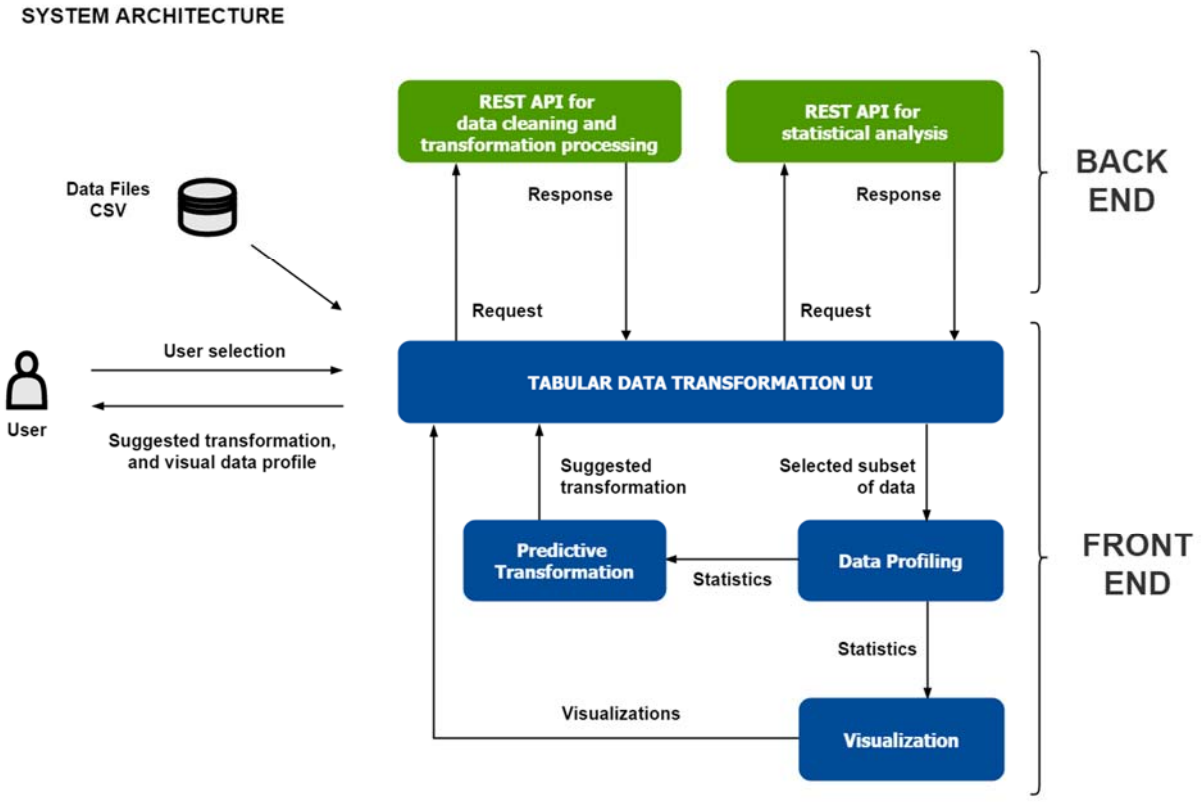


Figure 9: Grafterizer 2.0 visual data profiling microservice architecture

4.1.1.3 Functionality

In euBusinessGraph we have extended the Grafterizer framework with interactive data cleaning and transformation functionality, and visual data profiling. Figure 10 shows the working implementation of the wireframe from Figure 7. The implementation provides functionality for:

1. Suggestion-based data cleaning and transformation
2. Visual data profiling

Figure 10: Grafterizer working implementation of the visual data profiling approach

Suggestion-based data cleaning and transformation recommends relevant next steps in the data preparation process. The data cleaning and transformation steps are incrementally applied in a pipeline approach. The transformations sidebar implements a rule-based system that suggests relevant data cleaning and transformation procedures by considering data type, and whether a column or row has been selected.

Table 6 shows some examples of the implementation of logic for suggesting transformations, based on a rules matrix.

Table 6: Rules matrix for examples of suggested transformations

String	Number	Date	Column	Row	Functionality
True	true	true	true	**false**	Insert column right
True	true	true	true	**false**	Insert column left
True	true	true	**false**	true	Insert row above
True	true	true	**false**	true	Insert row below
True	true	true	true	**false**	Delete column
True	true	true	**false**	true	Delete row
false	true	**false**	true	**false**	Set empty cells to zero
True	**false**	**false**	true	**false**	Set text to uppercase
True	true	true	**false**	true	Set first row as header
True	**false**	**false**	true	**false**	Pad digits
false	**false**	true	true	**false**	Reformat dates

The application checks the statistical data profile that is returned by the visual data profiling service against the rules matrix illustrated in Table 6. Consider the following example:

The data profiling service returns a profile of the current data selected in a column of string values. The String value of the profile array is true, while the Number, Date and Row values are false:

```
[true, false, false, true, false]
```

The application checks this profile against the enumerated list of transformations in Figure xx, and matches the profile array with the rules array function number 9, 'Set to uppercase', as a possible transformation that will be suggested:

```
[true, false, false, true, false],        // (9) Set to uppercase
```

Visual data profiling analyses and determines data quality based on statistical properties, semantics and structure of data. The data quality assessment is presented to the user by means of statistical and scientific charts and visualizations.

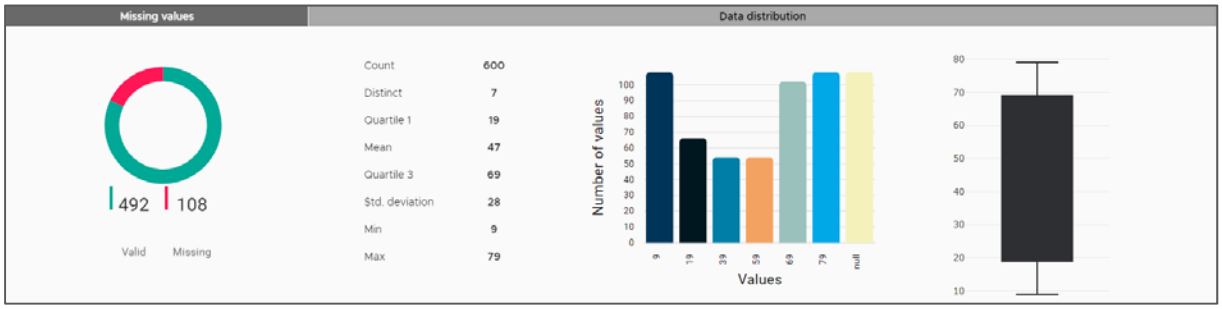


Figure 11: Grafterizer visual data profiling dashboard

The leftmost data profiling chart represents missing values and valid (non-null) values for the currently selected column. The three remaining charts (from left to right: table view, histogram, and box plot) represent the distribution of the currently selected column.

The visual data profiling service analyzes and assesses the quality of the dataset, and returns a statistical profile. This profile is an essential part of the underlying core application logic that suggests transformations and renders profiling charts:

1. **Count** – the total number of values in the selected column.
2. **Distinct** – the number of unique values. As an example, a column attribute 'week' might count in total 1000 rows and 7 unique values, one for each day.
3. **Histogram** – an array containing one value for each histogram bin.
4. **Quality** – an array that contains three different values, one value representing valid entries, one for invalid entries and another one for outliers.
5. **Boxplot** – array that contains all values necessary to render a boxplot chart, i.e. the first, second and third quartiles, and the median.
6. **Histogram_labels** – labels for the histogram chart visualization.
7. **Quality_labels** – labels for the quality chart visualization.

4.1.1.4 APIs

The euBusinessGraph Marketplace platform will be implemented with the capability of generating, publishing and invoking executable transformation services. The API specification for the data workflows and publishing defines methods for transformation resources and are maintained at

- https://datagraft.github.io/datagraft-API/dist/index.html?url=https://datagraft.github.io/datagraft-API/swagger.yaml

4.1.1.5 Software license and code repository

Grafterizer 2.0 is licensed under the Eclipse Public License – v 1.0:

- https://www.eclipse.org/legal/epl-v10.html

The source code repository for Grafterizer 2.0 is available on GitHub:

- https://github.com/datagraft/grafterizer-2.0

4.1.1.6 Installation and user guide

Installation and user guide for Grafterizer 2.0 are available at GitHub:

- https://github.com/datagraft/grafterizer-2.0

4.1.1.7 Next steps (2nd release)

4.1.1.7.1 Predictive, Intelligent Data Cleaning and Transformation

Data cleaning and transformation processes are most often a domain specific problem that focus on the statistical properties, semantics and structure of data. Grafterizer would benefit from an intelligent approach to the domain-specific data cleaning and transformation problem. Hence, based on the current column/ row selection in a dataset, the data scientist would be presented only relevant data profiling charts and suggestions for data cleaning and transformation.

Grafterizer will implement an intelligent system that learns from previous tasks to predict useful data cleaning and transformation actions. Such an approach will involve predictive interaction, a process in which the user is still involved to judge whether the next step is relevant or needs to be modified.

Examples of more intelligent data cleaning and transformation steps relevant to euBusinessGraph is to focus on suggestions that is needed to prepare the dataset for RDF-ization for the euBusinessGraph company model.

4.1.1.7.2 Multivariate Data Profiling

Multivariate data profiling provides an assessment of data quality in multiple combined columns. The visual data profiling approach that has been implemented in D3.2 applies univariate data profiling, i.e., one column at a time. While univariate analysis can uncover many data quality issues such as missing values and univariate outliers, more useful information is available from multivariate analysis, such as correlations between variables, that could assist the user in cleaning and transforming data.

An example is to add support for reconciliation in order to clean/enrich data being imported with regards to addresses and NUTS regions, i.e. harmonising the addresses of company data being imported to NUTS level 3 as described in Deliverable D2.1.

4.1.1.7.3 Direct Manipulation Interfaces

Direct manipulation interfaces (e.g. Microsoft Excel spreadsheet) is well known to most data scientists. The visual data profiling approach implemented in Grafterizer 2.0 provides a direct manipulation interface in a spreadsheet style table view that dynamically integrates with the approach. Furthermore, our evaluations of Grafterizer indicate that users prefer a direct manipulation spreadsheet for basic data cleaning and transformation operations.

The advantage of using a direct manipulation interface is that it reduces the distance between a user's intention and the capabilities provided by the system. Hence, the efforts required to reach a goal is reduced. As an example, a user that wants to delete a column in a dataset, would probably want to click the column directly to find an applicable action. An interface that does not rely on direct manipulation, could require the user to specify the intended action in a domain specific language.

The direct manipulation approach could provide a solution that is easy to use, and reduce the learning curve for inexperienced users. Still, not all types of data operations will benefit from direct manipulation interfaces. As an example, complex, or repeated, data cleaning and transformation sequences that require parameters may not be suitable. Future work in D3.4 will continue to explore the use of direct manipulation interfaces in visual data profiling approaches, and determine which data cleaning and transformation processes that can be executed directly in a spreadsheet table view.

While support for direct manipulation interfaces will generally improve the usability of the Grafterizer tool, this will be given low priority as focus on enhancing the tool according to specific needs of euBusinessGraph will be prioritized instead.

4.2 Data Interlinking Services

Based on further elaboration on the descriptions of data interlinking services in ASIA, the tool that is introduced to support assisted interpretation of tables and their RDF-ization in euBusinessGraph, we slightly modified the terminology used to refer to data interlinking services:

- In D3.2, *semantic labelling* is referred to as the task of assigning labels to element of the table. Since these labels are in practice implemented as annotations for the table, we use here the more intuitive term *semantic annotation* instead of semantic labelling. In particular, we remark that:
    - *Schema-level annotations* support what was referred to as the *schema alignment step* of semantic labelling in D3.1.
    - *Instance-level annotations* support what was referred to as *instances reconciliation step* of semantic labelling in D3.1.

Finally, if link discovery (i.e., discovery of links between two KGs - see D1.3) is needed to support business cases, link discovery services will be provided by wrapping one of the two mature link discovery tools that are available as open source tools: LIMES 2.0[2], released under AGPL-3.0 licence, and SILK[3], released under Apache Licence 2.0.

Table 7 lists the software components and how they map to the Data Interlinking Services of Figure 4.

Table 7: Software components – Data Interlinking Services

Software component	Implemented service	Implementation status
ABSTAT	Real-time vocabulary autocomplete using KG summaries	Implemented as a dedicated service accessible via API
ABSTAT	Search of vocabulary terms using KG summaries	Implemented as the ABSTAT search component
ABSTAT	Statistics about KG structure and vocabularies	Implemented as ABSTAT APIs.
ASIA	Schema -level semantic annotation (schema-level linking)	Implemented in manual annotation mode: ASIA provides suggestions via ABSTAT autocomplete service.
ASIA	Instance-level semantic annotation (with focus on company identifiers)	Not implemented in this version
ASIA	Data Transformation (RDF-ization)	Under development: mapping from schema-level annotations to Grafterizer data transformation and its implementation will be released at M13.

4.2.1 ABSTAT

ABSTAT (Linked Data Summaries with ABstraction and STATistics) is a framework developed by UNIMIB to support users and machines in better understanding big and complex RDF datasets. Given an RDF dataset and, optionally, an ontology (used in the data set), ABSTAT computes a summary that provides an abstract but complete description of the data set content. Summaries are published and made accessible via web interfaces, in such a way that the information that they contain can be consumed by human users and machines (via APIs). ABSTAT makes also use of a minimalization mechanism to keep summaries complete but as small as possible.

[2] http://aksw.org/Projects/LIMES.html
[3] http://silkframework.org/

ABSTAT summaries provide answers to questions like: what types of resources are described in a data set? What properties are used to link the resources? What types of resources are linked and by means of what properties? How many resources have a certain type and how frequent is the use of a given property? In practice, ABSTAT summaries describe the use of vocabularies in data sets. ABSTAT is a back-end infrastructure that is aimed at support different tasks:

- Data understanding. ABSTAT summaries provide a complete overview of the content of a data set; this feature was proved to be useful to support, for example, SPARQL query formulation.
- Vocabulary matching for table annotation. Summaries record rich statistics about the usage of vocabularies/ontologies in data sets. Thus we can use summaries to provide types and properties that match a string, for example, the header of a column in a table that we want to publish in RDF reusing existing vocabularies. In addition, statistics provide valuable information to algorithms aimed at suggesting the best properties and types to use when transforming a tabular data into RDF.

The first versions of ABSTAT is the result of research activities carried out at UNIMIB and was developed as a research prototype. Such version is not suitable to be integrated in a production environment. For this reason, a new and more robust, version of ABSTAT is created in euBusinessGraph. The new version of ABSTAT will incrementally include algorithms developed for the research prototype. These algorithms compute for example more sophisticated statistics, such as cardinality of relations, number of instances (which is defined considered the semantics of the ontology where types and properties are defined) and more.

4.2.1.1 Requirements

The requirements for ABSTAT are described in Table 8. Each requirement is identified by a unique code, so it can be referenced when needed.

Table 8: ABSTAT requirements

# Req	Description	Type
01	The system shall allow users to upload datasets and ontologies	F
02	The system shall allow users to trigger a summarization with datasets and ontologies that were uploaded as specified in 01. Computed summaries should be archived in a persistent back-end storage.	F
03	The summarization process can be configured by a user, who can specify parameters of the summarization (measures to compute, etc.).	F
04	A user shall be able to load a summary on a persistent storage, which support structured queries over the summaries via API (09) or via a GUI (06)	F
05	The user shall be able to index a summary on a search engine in order to support full-text search via a GUI (07).	F
06	A GUI should support browsing of summaries. Browsing consists in constrained queries over one or more summaries specified by the user (e.g., by specifying a desired subject and/or predicate and/or object).	F
07	A GUI should support full-text search over one or more summaries specified by the users. Different ranking functions can be supported.	F
08	The system shall allow users to download summaries from ABSTAT back-end storage even without prior 04 and 05.	F
09	The system should support constrained queries over one or more summaries via APIs.	F
10	The system should provide an API that implements an autocomplete mechanism using vocabularies that appear in the set of indexed datasets. Autocomplete should	F

	take a string as input and other optional pattern-level[4] constraints and return vocabulary terms (predicates and types) guaranteeing interactive time.	
11	The back-end storage (FR02) should contain summaries in a flexible and agnostic data format in order to make easy post manipulations.	NF
12	The autocomplete API (FR10) should guarantee an interactive time as it should support real-time autocomplete.	NF
		F: Functional Requirement **NF: Non-Functional Requirement**

4.2.1.2 Architecture

ABSTAT is designed to be modular and decoupled in order to support different scenarios of use. For example, in one scenario, summaries are computed and published – and made accessible to users - using a same node in a network. In another scenario, one node may compute the summaries, which, once computed, are then transferred to a different node that publish the summaries and make them accessible to users. In Figure 12 we can distinguish five main components.

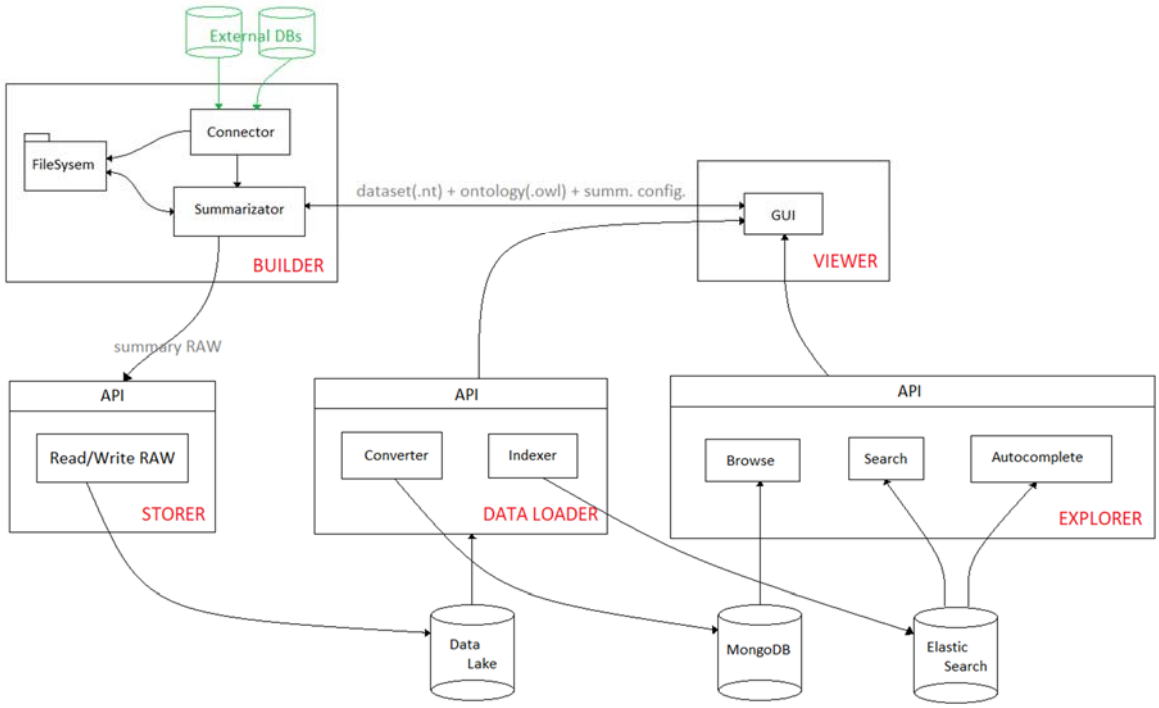


Figure 12: ABSTAT architecture

The ABSTAT Viewer provides a graphic user interface to serve different types of jobs such as summary exploration, execution of the summarization process using a wizard and indexing of summaries after execution. Summary exploration can be performed using constrained queries (by a desired subject and/or predicate and/or object) and full-text search. Both exploration modes can be run over one or more summaries. The summarization wizard provides a GUI to let users select datasets/ontologies from a list or using an upload module, configure the summarization process, and launch it. After the summary is computed, the user can load / index it on a persistent storage/search engine in order to support access to the summary through ABSTAT APIs or GUI. Indexing is required for full-text search.

The ABSTAT Builder is the component that executes the summarization algorithms and produces the summaries. The Summarizator sub-component requires as input a dataset (in N3[5] format) and an

[4] Patterns, also named *abstract knowledge patterns* (akp), are the basic information primitive in ABSTAT summaries.
[5] https://www.w3.org/TeamSubmission/n3/

ontology (in owl format) along with the configuration chosen by the user. If the user does not provide an input file because data are in a DB, the Connector sub-component allow for extracting a dump and storing it in the correct file to serve as input to the Summarizator.

The ABSTAT Storer component feeds a data lake storage with the raw data produced by the Builder. It also receives download requests from users who want to get raw summaries.

The ABSTAT Loader contains the Converter sub-components, which convert data available in formats different from N3 into the N3 format, so as to allow for summarization of RDF data represented using different formats. The Indexer sub-component indexes summaries in a search engine. Note that the Loader component receives the control input from the Viewer.

The ABSTAT Explorer is organized as a set of APIs to satisfy summary exploration requests from Viewer or users who want to use them directly.

Earlier versions of ABSTAT were developed for research purposes and made publicly available. However, the version of ABSTAT released as part of this deliverable can be considered the first official release made available on Bitbucket at https://bitbucket.org/disco_unimib/abstat-core with a licence (see Section 4.2.1.5). For this reason the version of ABSTAT released as part of this deliverable is numbered ABSTAT 0.1[6].

4.2.1.3 Functionality

ABSTAT (Linked Data Summaries with ABstraction and STATistics) is a framework that computes and provides access to synthetic descriptions of RDF data sets. These descriptions can be viewed as profiles of RDF data sets and are called summaries in ABSTAT.

The summary of a data set describes its content by listing every schema-level pattern that occurs in the data and a set of statistics. A schema-level pattern, named also abstract knowledge pattern (akp) or, more simply, *pattern*, is a triple <*Type_1*,*P*,*Type_2*> that tells there are instances of *Type_1* linked to instances of *Type_2* with the property *P*. With the term *type* we refer to either an ontology class (e.g., foaf:Person) or a datatype (e.g., xsd:DateTime). In addition to patterns, summaries provide several *statistics* for the patterns and their constituents, i.e., types and properties. The version of ABSTAT that is deployed in the EuBusinessGraph computes and includes statistics about the occurrence of patterns, types and properties. Occurrence assigned to a pattern <*Type_1*,*P*,*Type_2*> tells how many instances of *Type_1* are linked to instances of *Type_2* with the property *P*.

From large data sets that use rich type hierarchies the number of patterns that can be extracted can be very large. To provide more compact summaries ABSTAT can use the ontology used in the data set to extract a minimal set of specific patterns and discard patterns that are redundant. The minimal set of specific patterns (also referred to as *Minimal-Type Pattern Base* - *MTPB*) is sufficient to reconstruct the complete set of patterns that can be extracted from a data set by inferring redundant patterns. The mechanism used in ABSTAT to compute the MTPB using an ontology specified by the user is known as *minimalization*. The ontology is used to define which are the most specific types of instances that occur in the summarized triples. In this way, it is possible to extract from each RDF triple only the patterns that are more specific, i.e., the patterns that contain the most specific types of the instances occurring in the triple. When minimalization is used, the intuitive meaning of a pattern <*Type_1*,*P*,*Type_2*> with occurrence *n* is that there are *n* RDF triples <*s*,*P*,*o*> such that *s* and *o* have *Type_1* and *Type_2* respectively as minimal type.

If the user specifies the main pay-level domain of interest in the dataset (e.g., dbpedia.org for DBpedia_2014 dataset), ABSTAT can distinguish between resources (patterns, types and properties) that are *internal* (resources having the specified pay-level domain) and *external* (resources having a pay-level domain different from the one specified by the user). This distinction has the only purpose of letting users filter out patterns that include some external resource, which can be useful in some cases (e.g., the user may want to hide all patterns that contain the type *foaf:person* when looking at patterns extracted from DBpedia).

[6] The architecture depicted in Figure 12 and its components have been developed for this release. Only core summarization algorithms have been reused from the earlier research prototype.

More details about the minimalization process, its effect on summaries' size, and an evaluation of the informativeness of summaries can be found in a scientific paper referenced in Section 4.2.1.8.

Functionalities of ABSTAT currently deployed in the euBusinessGraph therefore include:

- Summarization of RDF data in N3 format with extraction of patterns and occurrence statistics (core functionality of the tool)
- Configuration and launch of the summarization algorithm via GUI with storage in MongoDB (addressing requirements [01,02,03,04,11])
- Indexing of computed summaries via GUI (addressing requirements [05])
- Browsing and full-text search using a browser (addressing requirements [06,07])
- Access to summaries via APIs (addressing requirements [09])
- Autocomplete service over arbitrary strings (addressing requirements [10,12])

Functionalities that will be implemented in the second release are described in Section 4.2.1.7.

4.2.1.4 APIs

The ABSTAT Storer exposes a set of APIs which allows the GUI and the users to navigate the available summaries. Here we describe the API used in the annotation process.

GET	**/api/v1/SolrSuggestions**		*concepts/properties syntactic autocomplete*	
Parameter	Values	Example	Required	Notes
subjectType (string)	(URI) concept	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl%23Agent	✗	
subjectType (string)	(URI) concept	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl%23Agent	✗	
subjectType (string)	(URI) concept	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl%23Agent	✗	
qString (string)	text	per	✓	<2 chars: will search for matchings which begin with the user input else: extension to central substrings.
qPosition (string)	subj | obj | pred	subj	✓	
dataset (string)	target dataset	dbpedia-2015-10	✗	If not set, all datasets are used
rows (int)	number of rows	100	✗	Results to be returned
start (int)	offset	0	✗	To be used for pagination
Usage example				

Request	/api/v1/SolrSuggestions?qString=a&qPosition=pred&rows=10&start=400
Response	[{"suggestion":"http://dbpedia.org/property/autocat", "dataset":"dbpedia-2015-10-extended", "frequency":7013 }, {"suggestion": “http://dbpedia.org/ontology/abbreviation", "dataset":"dbpedia-2015-10-extended-infobox", "frequency":6963, }, …]

4.2.1.5 Software license and code repository

ABSTAT 0.1 is licensed under the GNU Affero General Public License v3.0 licence:

- https://www.gnu.org/licenses/

The code is open source and can be found at:

- https://bitbucket.org/disco_unimib/abstat-core

4.2.1.6 Installation and user guide

A guide to install ABSTAT can be found in the Readme.me file in the ABSTAT repository on Bitbucket:

- https://bitbucket.org/disco_unimib/abstat-core

4.2.1.7 Next steps (2nd release)

In the next release, ABSTAT will include different features:

- Improved GUI, in particular for control of summarization and upload of datasets from other RDF storage systems (in particular, OntoText GraphDB);
- Inclusion of more advanced summarization options, previously included in the research prototype (pattern inference, cardinality profiles, minimalization over properties);
- Comparison of summaries and RDF shape specifications, e.g., in SHACL[7], for data quality evaluation;
- Distributed architecture with more local Builders that compute summaries stored and accessed from centralized storage.

4.2.1.8 References

[1] Azzurra Ragone, Paolo Tomeo, Corrado Magarelli, Tommaso Di Noia, Matteo Palmonari, Andrea Maurino, Eugenio Di Sciascio: **Schema-summarization in linked-data-based feature selection for recommender systems.** SAC 2017: 330-335

[2] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, Andrea Maurino. **ABSTAT: Ontology-driven Linked Data Summaries with Pattern Minimalization.** ESWC (Satellite Events) 2016: 381-395

4.2.2 Assisted Semantic Interpretation and Annotation of tables (ASIA)

ASIA is a semantic table enrichment tool integrated into Grafterizer 2.0, which provides Assisted Semantic Interpretation and Annotation of tables (CSV files). Semantic annotations create links from

[7] https://www.w3.org/TR/shacl/

elements of a table to existing Knowledge Graphs (KGs). ASIA supports the creation of annotations at the schema and at the instance level.

- *Schema-level annotations* link columns or column pairs of the table to types or properties used in a KG (in the following, we will use the term type to refer either to an RDFS class, e.g., foaf:Person, or to a datatype, e.g., xsd:double). For example, a user can annotate a column that contains company identifiers with the type dbo:Company, a class of the DBpedia ontology[8] used in a KG that describes companies. Schema-level annotations support what in D3.1 was referred to as the schema alignment step of semantic labelling.
- *Instance-level annotations* link individual values in the table to identifiers of entities described in a KG. For example, a user can annotate a column that contains the value "Università Milano Bicocca" to the identifier dbr:University_of_Milano-Bicocca that describe UNIMIB in the DBpedia[9] KG. Instance-level annotations support what in D3.1 was referred to as instances reconciliation step of semantic labelling.

The annotations created for a table with ASIA support two functionalities:

- **RDF-ization**: annotations support the automatic transformation of the table into a KG represented in RDF, without requiring the specification of machine-readable mappings from CSV to RDF.
- **Content Enrichment**: instance-level annotations create links to identifiers of a reference KG; these linked identifiers can be used to fetch additional data from the KG or from other sources that use the same identifiers. For example, by linking company names to company identifiers in the EuBusinessGraph KG, the user can use data enrichment widgets to fetch the CEO of the companies (where represented in the KG), and add this information in the table.

In ASIA, users can create annotations from a GUI but they are assisted in this complex process by table interpretation algorithms. These algorithms are implemented as different interlinking services, which suggest annotations and help users make informed decisions when they transform the data into KGs.

In EuBusinessGraph ASIA is used for RDF-ization. Thus, in the rest of this section we concentrate on this functionality.

Before describing requirements for ASIA in EuBusinessGraph, we discuss the main characteristics of the RDF-ization process and the challenges for a semantic enrichment tool that aims at supporting this process.

Semantic table annotation for RDF-ization in euBusinessGraph. KGs represented using semantic Web standards are represented using the RDF data model and are organized using shared vocabularies, possibly defined by ontologies modelled using RDFS and OWL languages. In euBusinessGraph, organizations that want to add their data to the Business Graph need to publish their data as KGs trying to interlink their data to the existing graph. In this process, two interlinking tasks are defined, each one with its own main objective.

- Schema-level interlinking has the objective of supporting providers to reuse, when possible, properties and classes and datatypes used in the Business Graph, so as to harmonize, at schema level, the new constituents of the Business Graph with the existing graph. If new data providers reuse vocabularies used in euBusinessGraph at a large extent, new data can be easily queried from the euBusinessGraph platform.
- Schema-level interlinking (or, *value reconciliation*) has the objective of supporting providers to reconcile, when possible, values that occur in the table to standard or shared systems of identifiers used within the Business Graph data, e.g., identifiers of companies, locations, or administrative areas. When values are reconciled, the KG published by the new data providers are linked to the rest of the graph at the instance level, thus supporting querying across different constituents of the Business Graph.

We add few important remarks to specify the objectives of table annotation in euBusinessGraph.

[8] http://wiki.dbpedia.org/services-resources/ontology
[9] http://wiki.dbpedia.org/

- **Need for assistance also for schema-level interlinking.** Performing even schema-level semantic annotation manually can be resource and time consuming, because it requires a domain expert that knows which are all the properties/types available in several KGs, and she has to choose one-by-one the correct property/type that better describe the semantics of each table column. The objective of ASIA is to ease the semantic annotation task by helping the user make informed decision about the terminology to be used in the annotations, and, therefore, in the data that will be contributed to the Business Graph.
- **Schema-level interlinking and instance-level interlinking**. Schema-level interlinking is also useful to support the more difficult task of value reconciliation by making information available that is useful in the reconciliation process. For example, by specifying that a column lists values that belong to a type defined in a company ontology, a reconciliation algorithm may compare the values in the column only against instances in the KG that belong to the specified type (a technique often referred to as *blocking*). This and similar information (e.g., the specification of the RDF property of which values in the column will be objects) can improve scalability and accuracy of reconciliation algorithms.
- **Machine intelligence and the role of users in table annotation.** Table annotation and its subtasks, e.g., annotation of columns with RDF types and properties, reconciliation of values, etc., are complex and error-prone tasks. Automatic algorithms are needed to support the users in annotating a table, but users want to control the process and refine suggestions given by algorithms, since it is well known that no automatic algorithm can be perfectly accurate. Table annotation in euBusinessGraph should therefore provide automatic algorithms that support the annotation process as well as control via a user interface.
- **Flexibility in table annotation algorithms**. Reuse of existing vocabularies is important and useful for the reasons highlighted above, and algorithms used in table annotation should help users reuse existing vocabularies. However, there may be pieces of the data that are not adequately covered by existing vocabularies or data providers may prefer to use their own terminology in some cases, possibly by indirectly mapping it to other vocabularies (e.g., adding a new class as a subclass of an existing class). When a data provider introduces new terminology in a new piece of the Business Graph, this terminology may become useful to support new annotations in the future (by the same provider or other). In other words, usage of terminology in the Business Graph is an important driver for deciding which terminology to be used when publishing new data in the Business Graph; table annotation should encourage vocabulary reuse but allow for new terminology to be used.

ASIA vs STAN and ASIA 0.1. ASIA is developed starting from a previous tool named STAN (Semantic Table ANnotation), a prototype that suggests table annotations using only information about values occurring in the table, in such a way that: the table annotation functionalities are integrated in Grafterizer 2.0, the data manipulation tool used in the project, and the logic described here above can be fully implemented. In this first release of ASIA, referred to as ASIA 0.1, we focus on the macro-level objectives listed here below.

- Develop ASIA as a table annotation component integrated in Grafterizer 2.0 and that can be modularly extended so as to provide schema-level and instance-level interlinking.
- Support schema-level interlinking incorporating information about terminology usage. To this end ASIA is made interoperable with ABSTAT, a data summarization tool developed for RDF, which provides profiles of RDF data sets that report the usage of terminology in existing data sets.

4.2.2.1 Requirements

The annotation of one column specifies a type for the values appearing in the column. In addition, if the column implicitly describes a relation, i.e., the values in the column are related to values of another column, the user needs to define this relation. In essence, to do so, she has to specify: the RDF property that models the intended relation, the *source column* of the relation, i.e., the column from which subjects for the property will be selected. Thanks to this specification, it will be possible to generate a set of n RDF triples $<s_i,p,o_i>$ with $1 \leq i \leq n$ from the table such that:

- p is the RDF property used to represent the relation,

- values s_i are taken from the *source column,*
- values o_i are taken from the column that is being annotated,
- s_i and o_i occur in the *i*-th row of the table and will be respectively subject and object of the *i*-th triple generated from the table.

In other words, columns that describe values of a relation (and will provide objects for the relation) will be linked to the columns that describe the subjects of the relation. The ones described above are the basic elements of an annotation model, which specifies when annotations are valid, i.e., contain enough information to generate valid RDF triples. The model, whose details are omitted here has been defined as activity of euBusinessGraph.

Even if ASIA is integrated with Grafterizer 2.0, which allows users to manipulate tables before the RDF-ization, we cannot assume that users always want to RDF-ize the whole table manipulated with Grafterizer 2.0; sometimes a user might be interested in RDF-ize only a subset of the table, without filtering it in advance. Since RDF-ization is a functionality provided by DataGraft, ASIA should produce annotations that can generate transformations compliant with the data transformation model used in Grafterizer 2.0.

Finally, ASIA should guide the user during the annotation task by providing procedures to *validate annotations* that ensures the validity of the RDF triples generated by the RDF-ization algorithm for a given set of annotations defined by a user.

All the above requirements are summarized in Table 9.

Table 9: ASIA requirements

# Req	Description	Type
01	The user can choose arbitrarily which columns to annotate, and which ones not	F
02	The annotation form must be validated in order to guide the user through the annotation task	F
03	Columns that are not annotated must be filtered out from the RDF-ization process	F
04	Each annotation must be complete enough to allow the RDF-ization process, that is each annotation form must be sufficient to produce a valid RDF	F
05	The annotation model designed within ASIA must be compliant with the transformation model provided by Grafterizer 2.0, or alternatively a mapping between the two models should be provided	F
06	ASIA should provide suggestions about types and properties, and some statistics that can help the user in making decision.	F
07	Suggestions should be provided within an autocomplete function, that is the user will receive suggestions when she will start typing a property or a type.	F
08	The autocomplete functionality should fulfil real-time constraints.	NF
	F: Functional Requirement **NF: Non-Functional Requirement**	

4.2.2.2 Architecture

Following the design principles of Separation of Concerns (SoC) (already described in Section 4.1.1.2), ASIA is designed with a layered architecture. The Business Logic is developed within the ASIA Backend component, while the Presentation Layer contains the ASIA Frontend component. We describe in the following the two main components of ASIA, depicted in Figure 13:

- ASIA Backend, which contains the logic related to the annotation suggestion service and that is responsible for the intercommunication with third-party services (like ABSTAT); in particular:
    - ASIA Backend is currently integrated with ABSTAT autocomplete service via API; as a result, the Backend provides to ASIA, the real-time autocomplete service offered by

ABSTAT as well as statistics computed by the same tool, thus fulfilling requirements 06,07,08.

- ASIA Frontend, which is the frontend customized and integrated within Grafterizer 2.0, and that contains two main modules:
    - Annotation Validation, which is the component that fulfils requirements from 01 to 04;
    - Annotation-To-Transformation, which is the component that fulfils the requirement 05.

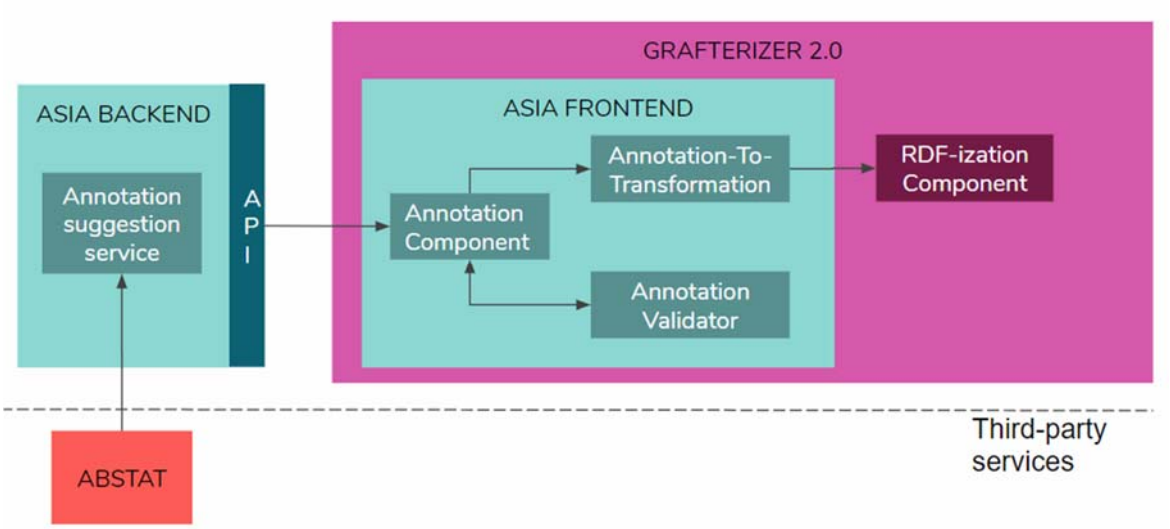


Figure 13: ASIA architecture

4.2.2.3 Functionality

Given a table, ASIA provides an interface that guides the user through the annotation task using a column-wise approach. This approach allows users to define annotations for each column in such a way that the annotations encode enough information to automatically generate the transformations required for RDF-ization.

The first release of ASIA provides a syntactic autocomplete function, that is, a set of recommendations based on what the user is typing. In addition, along with suggestions ASIA provides additional information to help the user in making decision (how to choose the correct property/type to be used?). ASIA provide suggestions and usage statistics by exploiting the summarization capability of ABSTAT. An ABSTAT summary also includes statistics about patterns, properties and types in the datasets. Patterns and statistics model different aspects of the dataset, such as usage, domain and quality. ABSTAT is useful for the semantic annotation as it support a suggestion mechanism of predicates and types based on a context (the table headers) by looking on the summaries that it has produced.

When the user start typing a type (but the same applies also for the properties case), ASIA suggests at most 15 types, which are the types that are most used in the KGs summarized by ABSTAT. Along with the type itself, ASIA provides also the name of the KG from which the type is kept, and the number of instances of that type that are found in the KG (occurrences). Types and properties statistics can help users to identify which suggestion is more probably to fit in the annotation context.

Each subsequent release of ASIA (and, eventually, the second release) will add layers of intelligence to its recommendation strategies, using richer evidence, for example, the relationships already stated between columns (that can be used to filter out some types/properties).

4.2.2.4 APIs

The ASIA Backend exposes a set of APIs, designed to reach high levels of modularity and maintainability. In particular, it is important to support the agile improvement of the interlinking algorithms used in the Backend of ASIA, so as to add layers of intelligence in the recommendation strategies. Here we describe only the APIs designed for the euBusinessGraph project purposes.

GET	/api/suggestions/abstatSyntactic		*syntactic autocomplete based on ABSTAT suggestions*	
Parameter	Values	Example	Required	Notes
query (string)	text	person	✓	At least 2 chars are required
position (string)	subj | obj | pred	subj	✓	
dataset (string)	target dataset	dbpedia-2015-10	✗	If not set, all datasets are used
rows (int)	number of rows	100	✗	If not set, no filters will apply
start (int)	offset	0	✗	To be used for pagination
Usage example				
Request	/api/suggestions/abstatSyntactic?query=person&position=subj			
Response	[{"suggestion":"http://dbpedia.org/ontology/Person", "dataset":"dbpedia-2015-10-extended", "frequency":1926091, "namespace": null }, {"suggestion": “http://dbpedia.org/ontology/Person", "dataset":"dbpedia-2015-10-extended-infobox", "frequency":1926091, "namespace":null }, …]			

GET	/api/suggestions/abstatDatasets		*List of summaries available in ABSTAT*	
Parameter	Values	Example	Required	Notes
N/A	N/A	N/A	N/A	
Usage example				
Request	/api/suggestions/abstatDatasets			
Response	[{"uri":"http://ld-summaries.org/linkedgeodata"}, {"uri":"http://ld-summaries.org/census-AU"}, {"uri":"http://ld-summaries.org/aemet"}, {"uri":"http://ld-summaries.org/dbpedia-2014"}, {"uri":"http://ld-summaries.org/swdf"}, {"uri":"http://ld-summaries.org/pharmgkb"}, {"uri":"http://ld-summaries.org/dbpedia-2015-10-extended-infobox"}, {"uri":"http://ld-summaries.org/system-test"}, …]			

4.2.2.5 Software license and code repository

ASIA 0.1 is licensed under the GNU Lesser General Public License v3.0 licence:

- https://www.gnu.org/licenses/

The code is open source and can be found at:

- https://bitbucket.org/disco_unimib/asia

4.2.2.6 Installation and user guide

ASIA is integrated into Grafterizer. Installation guide will be provided as part of Grafterizer installation guide. A guide on how to install ASIA backend can be found in ASIA repository on Bitbucket.

- https://bitbucket.org/disco_unimib/asia

4.2.2.7 Next steps (2nd release)

Starting from the next release, ASIA suggestions will be provided others recommendation strategies based on other evidences, for instance the relationships already stated between columns. Furthermore, previous annotations will be used by ASIA in order to customize the suggestions: the main idea is to recommend types and properties used by the user in similar datasets with two main advantages: speed up the process when doing repetitive tasks, and encourage reuse of terminology that is already used in the graph, which would reduce the risk of unnecessary diversification of terminology when new data are contributed to the existing graph. In addition, instance-level linking services using company identifiers in the graph will be integrated into ASIA.

4.3 Data Hosting Services

The business graph will be easier to manage and maintain if larger portions of it are centralized. The *Data Hosting Services* aim to provide just that, by provisioning a reliable hosting service for the business graph. They will be based on the Ontotext Cognitive Cloud[10]. Part of this cloud-based service is a semantic graph (triplestore) database-as-a-service (GraphDB Cloud[11]) that will serve as the main provider of data hosting for the Data Marketplace.

The Data Hosting Services follow a microservices pattern where each component is an independent part of the whole. Communication is carried out in a standardized manner, via API calls. This ensures that the system is more manageable and eases maintaince and future expansions. All Ontotext Cognitive Cloud components utilize this microservices architecture.

The requrements specified in this section are based on the preliminary user group analysis that we carried out and described in Deliverable 3.1. Those requirements are still subject to change depending on changes within the vision of the stakeholders or on new data.

Table 10 lists the software components and how they map to the Data Hosting Services of Figure 4.

Table 10: Software components – Data Hosting Services

Software component	Implemented service	Implementation status
GraphDB Cloud	DataQuery (SPARQL)	The service is ready.
Cognitive Cloud/DataGraft	Data Access	API keys
DataGraft	Data Updates	DataAccess component will be provided by integration of DataGraft and GraphDB Cloud. The scope of the first release is to facilitate the data providers to ETL data via DataGraft into

[10] https://ontotext.com/products/cognitive-cloud/
[11] https://cloud.ontotext.com

		GraphDB Cloud. GraphDB Cloud exposes a well defined database operation API based on RDF4J API.
DataGraft	Metadata	This component needs more requirements and specifications and will be provided in the second release. This component overlaps with the master database from the Marketplace and operational services as depicted in Figure 24.
GraphDB Cloud/Cognitive Cloud	RDF store	The service is ready. However changes in the requirements throughout the project can be addressed at a later stage.

4.3.1 DataGraft

DataGraft provides a user interface that enables user data and account management, user assets cataloguing and dataset and database management, whereas Grafterizer is an interactive tool for data cleaning and data transformation.

4.3.1.1 Requirements

Most of the development effort in euBusinessGraph is related to re-implementing and adding new features to Grafterizer (see Section 4.1.1). DataGraft is used as a platform to integrate Grafterizer and ASIA (see Section 4.2.2). We have also added support for GraphDB (see Section 4.3.2) in order to provide a streamline process for onboarding data to the business graph. As DataGraft is mainly used as an integration framework we refer to the sections for Grafterizer, ASIA and GraphDB for specific requirements related to euBusinessGraph.

4.3.1.2 Architecture

Both DataGraft and Grafterizer have been implemented through a microservice architecture consisting of a large number of sub-components, each contained in a Docker[12] container. The DataGraft and Grafterizer tool user interfaces have been integrated to provide a consistent user experience, whereby their connected microservices communicate with each other through REST. The individual components are illustrated in Figure 14.

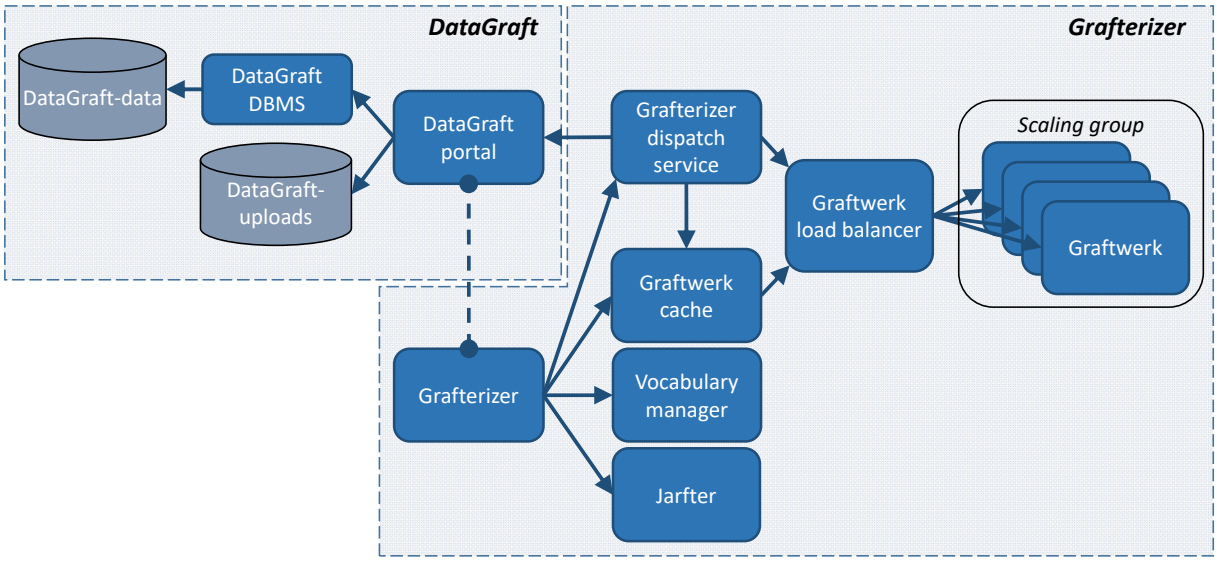


Figure 14. DataGraft and Grafterizer microservices architecture

DataGraft consists of the following components:

[12] https://www.docker.com/

- DataGraft portal: The portal serves several functions. It provides the web-based front-end that is used by the euBusinessGraph data publishers. It also implements the connection to the GraphDB *data hosting and access services*.
- DataGraft DBMS: This component represents the database management system (PostgreSQL[13]) for the user data and asset catalogue. Data are stored in a separate volume (Docker volume or Amazon S3[14] in production).

Grafterizer has the following sub-components:

- Grafterizer: Front-end component that implements the interactive GUI for data cleaning and data transformations.
- Grafterizer dispatch service: A server component for the Grafterizer front-end that handles request authentication on its behalf (in order to ensure security) and dispatches requests for input and output across the multiple services.
- Graftwerk: A sandboxed server component that executes the data cleaning and transformation scripts that are generated by the Grafterizer front-end over the set of input data sent by the dispatcher. Graftwerk uses a proprietary load-balancing component in order to distribute the traffic coming when a larger number of users use the transformation tool.
- Graftwerk cache: A FIFO cache service for the Grafterizer front-end requests to Graftwerk.
- Vocabulary manager: Simple RDF vocabulary management service for imported vocabularies used in the RDF mapping in the front-end. Enables searching through concepts and importing.
- Jarfter: A web service component for compiling executable JARs for transformations generated by the Grafterizer front end.

4.3.1.3 Functionality

DataGraft is used to integrated a streamlined data onboarding process using the software components Grafterizer (see Section 4.1.1), ASIA (see Section 4.2.2) and GraphDB (see Section 4.3.2) as illustrated in Figure below. This process ensures a self-service data provisioning through:

- Interactive/Smart tabular data transformation (Grafterizer 2.0)
- Interactive/Smart data annotations (ASIA)
- Data provisioning as a service (DataGraft and GraphDB)

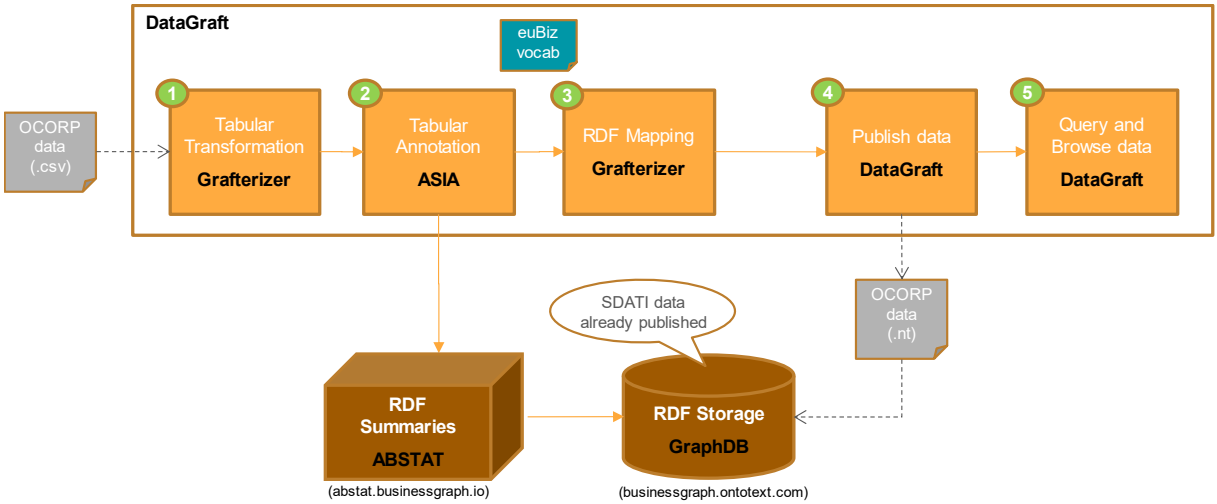


Figure 15: Business graph data provisioning

[13] https://www.postgresql.org/
[14] https://aws.amazon.com/s3/

4.3.1.4 APIs

The API specification for DataGraft is maintained at:

- https://datagraft.github.io/datagraft-API/dist/index.html?url=https://datagraft.github.io/datagraft-API/swagger.yaml

4.3.1.5 Software license and code repository

DataGraft is licensed under the Eclipse Public License – v 1.0:

- https://www.eclipse.org/legal/epl-v10.html

The source code repository for DataGraft is available on GitHub:

- https://github.com/datagraft/

4.3.1.6 Installation and user guide

DataGraft is provided as an online hosted service at:

- https://datagraft.io/

User guide documentation is available at GitHub:

- https://github.com/datagraft/datagraft-reference/blob/master/documentation.md

4.3.1.7 Next steps (2nd release)

The focus of the DataGraft development in euBusinessGraph is on the Grafterizer and ASIA software components that are integrated in the DataGraft platform. Thus some development effort is required on the DataGraft platform itself to ensure proper integration of Grafterizer and ASIA, as well as maintaining and adding support for GraphDB and the Cognitive Cloud in order to provide a self-service data provisioning process for onboarding data to the business graph.

4.3.2 GraphDB on the Cognitive Cloud

The Ontotext Cognitive Cloud provides:

- A scalable semantic graph database-as-a-service and analytics – GraphDB Cloud. The main data hosting provider for the Data Marketplace.
- Enterprise text analysis services for news, life sciences and social media.
- Access to knowledge graphs such as DBpedia, Wikidata and GeoNames.
- Secure and fully managed service.

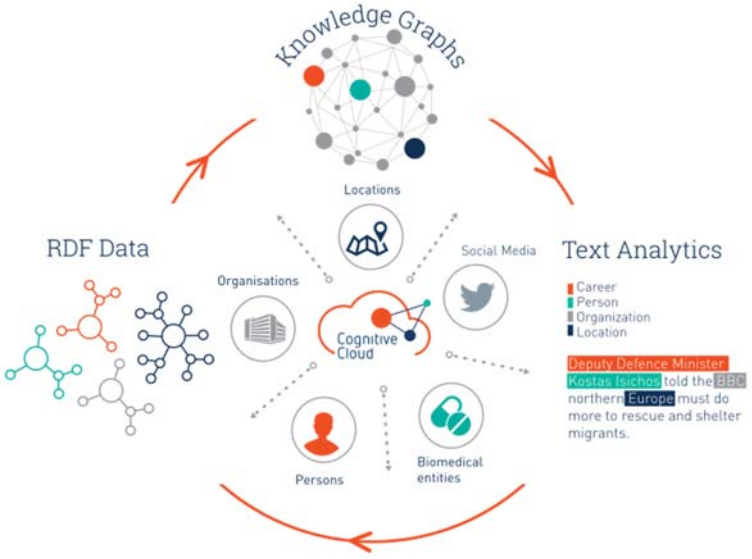


Figure 16: Ontotext Cognitive Cloud

GraphDB Cloud, is a product running on the Ontotext Cognitive Cloud Platform that is designed to serve scenarios with small, medium and enterprise database size and query load.

With GraphDB Cloud, users can have their private database instance up and running within seconds. They will no longer need to deal with DBA specific tasks such as installation and upgrades, provisioning and deployment, backups and restores, as well as ensuring the database availability and security. GraphDB Cloud is instantly available and easily accessible so that users can build smart data prototypes and productions faster, and at a lower cost, without spending valuable time on installing, configuring, or developing their own infrastructure components.

GraphDB Cloud is available in various tiers (levels), including free (basic), standard, enterprise, and custom (GraphDB Enterprise Cluster) tiers.

The Ontotext Cognitive Cloud platform is based on enterprise-grade technology from Ontotext including GraphDB and high-performance text mining solutions successfully applied at some of the largest enterprises in the world. We have taken proven technology used in publishing & media, government agencies and life sciences and made available with an accessible business model. The Ontotext Cognitive Cloud is designed to deliver fully managed, available, secure, and scalable solution that can be accessed via simple RESTful services at a low cost.

The Cognitive Cloud and GraphDB Cloud in particular will allow data providers from euBusinessGraph to reserve a database and then ETL their dataset by using the transformation services of the DataGraft platform. GraphDB Cloud will ensure that the euBusinessGraph platform will scale to many data providers and the Cognitive Cloud will provide valuable and scalable components such as payments, user management, monitoring, etc., to data

4.3.2.1 Requirements

The requirements for the data hosting layer are summarized in the following tables.

Table 11: Data consumer requirements

ID	Requirement	Description
Data Hosting Services		
DCR-01	Multiple dataset programmatic access channels	Access through different channels (e.g. SPARQL endpoint, original data download, REST APIs, reporting service).
DCR-01-a	Data dump	Ability to download large volumes of data as data dumps.
DCR-02	Searching and exploring existing datasets	Dataset search and browse/explore functionality.
DCR-02-a	Single access point	Single access point to information about company data.
DCR-02-b	High availability and efficient querying	Efficient querying based on indexation of the data based on pre-defined sets of indexes. High availability should be ensured for the indexed data by the hosting platform.
DCR-02-c	Extended company profile	Ability to access extended company profiles that integrates data from multiple data sources.
DCR-03	Access to dataset metadata information	Access to metadata information.
DCR-03-a	Detailed information about data	Detailed information about data that can be found in other data repositories.

Table 12: Data provider requirements

ID	Requirement	Description
Data Hosting Services		
DPR-06	Data access	Specifying access through different channels (e.g. SPARQL endpoint, original data download, REST APIs, reporting service).
DPR-07	Data updates	APIs for accessing and modifying (updating) datasets.
DPR-07-a	API for incremental update	API for incremental update of data.
DPR-07-b	API for bulk update	API for bulk update of data.
DPR-08	Dataset metadata	Management of metadata, such as standardized name, description, language, including company-specific extensions, such as jurisdictions.
DPR-08-a	Common vocabulary	Ability to describe data through a common/standard vocabulary.
DPR-09	Big data storage	Storage capability for large data volumes (RDF).

4.3.2.2 Architecture

GraphDB Cloud is designed to use the underlying cloud infrastructure of Amazon Web Services (AWS). GraphDB Cloud provides a flexible service, capable of dynamically expanding or shrinking the consumed cloud resources depending on the demand. The architecture is shown in the figure below.

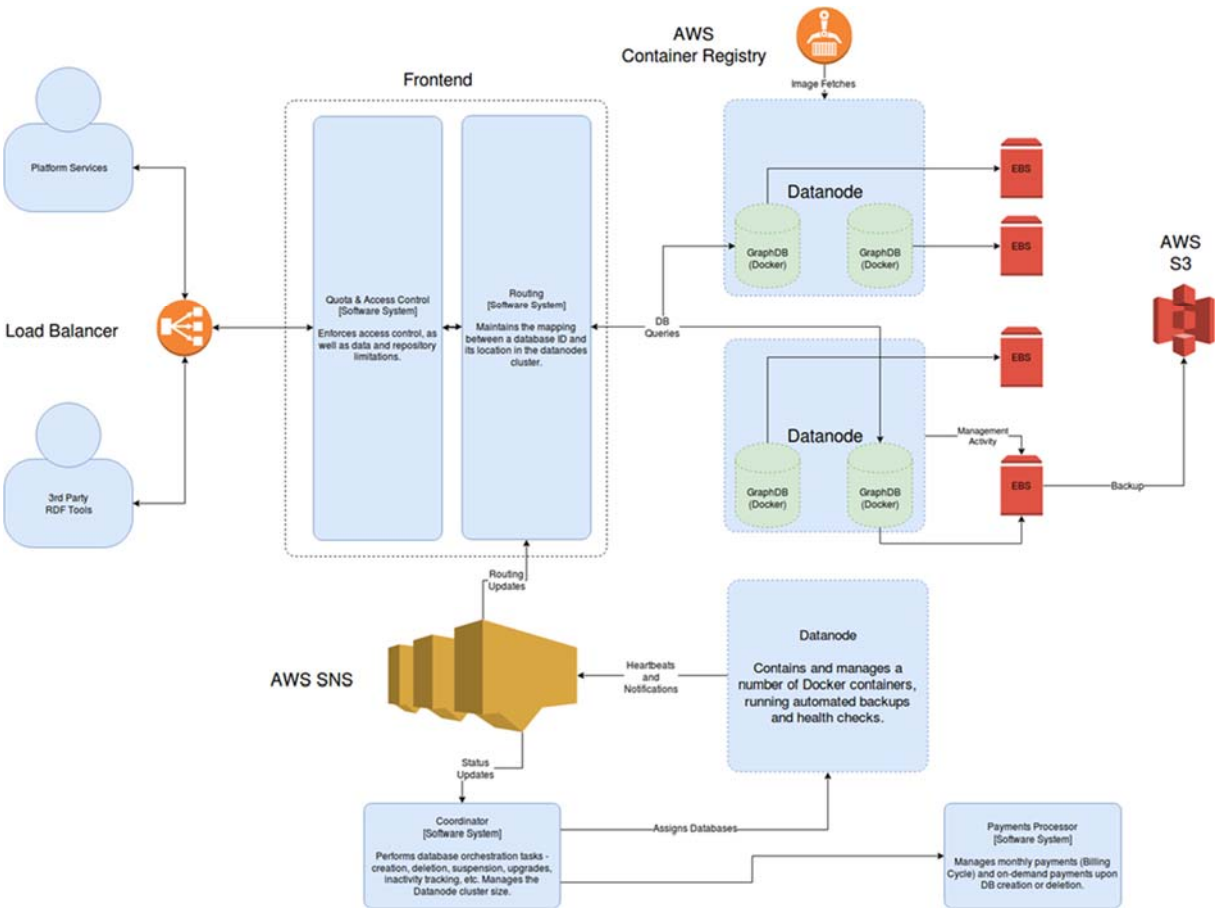


Figure 17: GraphDB architecture

GraphDB Cloud architecture comprises of several components, in brief:

- Frontend – stateless node responsible for the interaction with the clients of the DBaaS. The frontend serves as a router as well as a quota and access control service.
- Coordinator – orchestrates and keeps track of the health of the database cluster. Responsible for all database management actions, such as creation, deletion, upgrading, making backups, etc. Scales the number of datanodes as needed.
- Infrastructure services – services delivered by the Cloud infrastructure provider (in this case Amazon), such as: load balancers, message queues (SNS), Docker container registries, backup storage, etc.

The architecture follows the microservices design principles where a complex application is decomposed into small independent processes that communicate with each other via RESTful APIs.

The above architecture allows for:

- Flexible resource provisioning (scaling up and down) depending on the specific system demands;
- Minimal impact of system failures (component malfunction);
- Easier components replacement (failure recovery or upgrades).

All of the complexity of resource provisioning and software configuration of GraphDB Cloud is hidden behind management APIs. These APIs are used for integration with other systems. DataGraft uses them to provision and manage databases on demand. These APIs will be used when creating a new database for a euBusinessGraph user. This happens when a user creates their first SPARQL endpoint asset (i.e., their first RDF repository in their assigned database). After the database is created, again using the API, the platform obtains the URL of the newly provisioned repository, and can change the availability of the SPARQL endpoint from public to private (which requires a specially assigned API key to issue queries, also produced by the management API). The management API supports a large number of other database and repository management tasks.

Moreover, the DBaaS performs automated backups of the user data on regular intervals of time.

The hosted data are exposed by the DBaaS using two types of RESTful data access APIs:

- **Per repository** - SPARQL endpoint API – allows for issuing SPARQL SELECT queries; returns query results available in various formats
- **Per Database** - RDF4J (formerly known as Sesame) – allows for lower-level access; only accessible using a specially assigned API key. Each database instance is assigned a unique access URL that includes user account id and a database name.

The hosted data can be accessed via a user interface[15] that provides functionality to query, search, navigate and explore the data.

4.3.2.3 Functionality

The Data Hosting Service, based on the Ontotext Cognitive Cloud, as described within the introduction, will have to be improved and modified in order to conform to the needs of the Data Marketplace.

The hosting task improvements include scalability, performance and reliability of a large number of semantic graph databases running in the Cloud. This way large volumes of data can be managed and simultaneous queries and data access requests can be supported. The hosted datasets are accessible to third-party applications via various standard data access mechanisms: SPARQL query and Linked Data endpoints, as well as various RESTful APIs. Data may also be uploaded into the platform via standard read/write APIs for managing RDF data, such as the RDF4J API[16].

The incorporation of the cloud services with the overall Knowledge Marketplace means that the customers of euBusinessGraph can easily process the results which they obtain from the marketplace.

[15] http://graphdb.ontotext.com/documentation/standard/workbench.html
[16] http://docs.rdf4j.org/rest-api/

If the customers want to, they could utilize the components of the Cognitive Cloud to explore, transform and load the data that they obtain from the marketplace. Moreover, the Cognitive Cloud's scalability and availability as well as the already existent monitoring services mean that the master database will be easier to manage and maintain than creating a custom solution for it.

The service components are:

- GraphDB Cloud which can contain any RDF-ised datasets and provides SPARQL query access to the data. For the purpose of improving database availability and resilience, the RDF warehouse will not comprise a single database instance but a distributed set of database instances on a Cloud infrastructure;
- A standard SPARQL endpoint for querying data;
- RDF4J API for querying of RDF data. RDF4J provides light-weight industry standard way for querying and processing RDF data.
- A linked data endpoint, providing read-only access to RDF data;
- A data exporting service, to access the dataset in a variety of standard formats.

4.3.2.4 APIs

The Cognitive Cloud provides various REST APIs available online at:

- http://cloud-docs.ontotext.com/display/S4docs/GraphDB+Cloud+REST+APIs

The API calls allow users to manage their accounts and databases. They also expose the other parts of the cloud functionality such as text analytics and knowledge graphs. Specific to the euBusinessGraph project, the Cognitive Cloud will also be adapted to display dataset-related information.

To use any of these APIs requires an API key for Basic HTTP authentication. The API keys are private for each user account and can be generated in the Cognitive Cloud Management Console.

The RESTful services accept JSON as input and return JSON as output.

Cognitive Cloud services support gzip compression of the service's response. You can enable it by setting an Accept-Encoding: gzip HTTP header.

4.3.2.4.1 API Keys

Access to the Cognitive Cloud REST API requires a private API key, which consists of two randomly generated parts, the "key ID" and "password". Each customer account can generate an unlimited number of API keys (for different applications or users) and each API key can be enabled/disabled or deleted as necessary.

To create a new API key, go to the "API & Keys" section of the Cognitive Cloud Management Console at https://cloud.ontotext.com and click the "Create a new key" button.

Note: You must store the key ID and password at a secure location. For security purposes if you lose your API key password, you will not be able to recover it. You will need to delete your existing key and generate a new one.

All available API calls definitions, header information, security, encryption, etc., is available at the GraphDB Cloud API documentation page[17].

4.3.2.5 Software license and code repository

GraphDB Cloud is proprietary software with freemium business model. Users can create free of charge databases with full functionality. The only limitation is the amount of data that can be indexed by the database. This model will allow potential data providers to evaluate the service and provide small subsets of their data for testing purposes. Then those providers can be converted to paying customers.

[17] http://cloud-docs.ontotext.com/display/S4docs/GraphDB+Cloud+REST+APIs

The accumulation of many small data providers that are using the system free of charge will gradually lead to the creation of a critical mass of data, enabling meaningful searches for data consumers.

4.3.2.6 Installation and user guide

Getting started with Ontotext Cognitive Cloud involves the following steps:

- Register a personal account at http://cloud.ontotext.com.
- Log in with the newly created account.
- Start work with GraphDB Cloud by clicking on "Create Database" and selecting either paid or free tier of GraphDB.
- When your database has launched, start GraphDB Workbench to upload your first dataset and run SPARQL queries.
- If you prefer to use the existing REST APIs to load data - Generate a personal API key for accessing the REST APIs (via the Management Dashboard)
- As you get comfortable using GraphDB Cloud consider using Clone, Upgrade, Delete, and Backup features.
- Take a look at the API documentation and sample code in Java, C#, Python, JavaScript, Groovy, NodeJS, Ruby and PHP:
    - cURL example for uploading an RDF file to GraphDB Cloud[18]
    - examples for uploading RDF data to GraphDB Cloud via the GraphDB Workbench[19]
    - cURL example for updating data in GraphDB Cloud via SPARQL Update[20]
    - examples for updating data in GraphDB Cloud via SPARQL Update with the GraphDB Workbench[21]
    - cURL example for querying GraphDB Cloud via SPARQL[22]
    - examples for querying GraphDB Cloud with the GraphDB Workbench[23]
    - an example for visualising & exploring RDF data in GraphDB Cloud with Metreeca Graph Rover[24]

4.3.2.7 Next steps (2nd release)

As a second release we plan to add functionality to enable full text search and faceted search as part of standard SPARQL of the fully managed database-as-a-service.

4.4 Cross-Cutting Business Cases Analytics Services

Table 13 lists the software components and how they map to the Cross-Cutting Business Cases Analytics Services of Figure 4.

---

[18] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-cURL%28dataupload%29
[19] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-GraphDBWorkbench%28dataupload%29
[20] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-cURL%28SPARQLUpdate%29
[21] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-GraphDBWorkbench%28SPARQLUpdate%29
[22] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-cURL%28dataquery%29
[23] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-GraphDBWorkbench%28dataquery%29
[24] http://cloud-docs.ontotext.com/display/S4docs/Fully+Managed+Database#FullyManagedDatabase-MetreecaGraphRover

Table 13: Software components – Cross-Cutting Business Cases Analytics Services

Software component	Implemented service	Implementation status
Wikifier (extended product version for business environments)	Multi-lingual annotation	The service is ready. Subsystems are in development.
Event Registry	Multi-lingual event extraction and visualization	The service is ready. Subsystems are in development.
Graph-based analytics	Relation extraction	In development. Prototype deployed.
Graph-based analytics	Graph analytics	In development

4.4.1 Wikifier

JSI's semantic multilingual annotation service will be based on JSI Wikifier. JSI Wikifier is a web service which takes a text document as input and annotates it with links to relevant Wikipedia concepts.

Wikipedia as a source of possible semantic annotations can be treated as a large and fairly general-purpose ontology. Each page is thought of as representing a concept, while the relations between concepts are represented by internal hyperlinks between different Wikipedia pages, as well as by Wikipedia's category membership and cross-language links. Because Wikipedia is available in a number of languages, with cross-language links being available to identify pages that refer to the same concept in different languages, it is much easier to support multilingual and cross-lingual annotation.

4.4.1.1 Requirements

The requirements for Wikifier service are provided in the table below.

Table 14: Requirements – Wikifier

ID	Requirement	Description
Wikifier		
Req1	Semantic annotation	The service should provide semantic annotation of textual input
Req2	Cross-linguality and Multilinguality	The text document for sematic annotation should be in one of the supported languages. This list of supported languages consists of all languages for which, as of this writing, a localized Wikipedia with at least 10000 pages is available (from the List of Wikipedias), plus a small number of other languages.
Req3	Web API	The service is accessible via RESTful API (with an access key). The users should register at the Wikifier website.
Req4	Extra vocabularies	The service should embed a set of business related extra vocabularies (such as company vocabularies).

4.4.1.2 Architecture

Figure 18 describes the main components of Wikifier architecture:

- WikiData (a free and open knowledge base).
- In-memory stores (that incorporate WikiData graph, Wordnet and Extra vocabularies. WordNet (https://wordnet.princeton.edu/) is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations.

- Processing and disambiguation mechanisms based on PageRank. PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results.
- User access via Web interface and RESTful API.

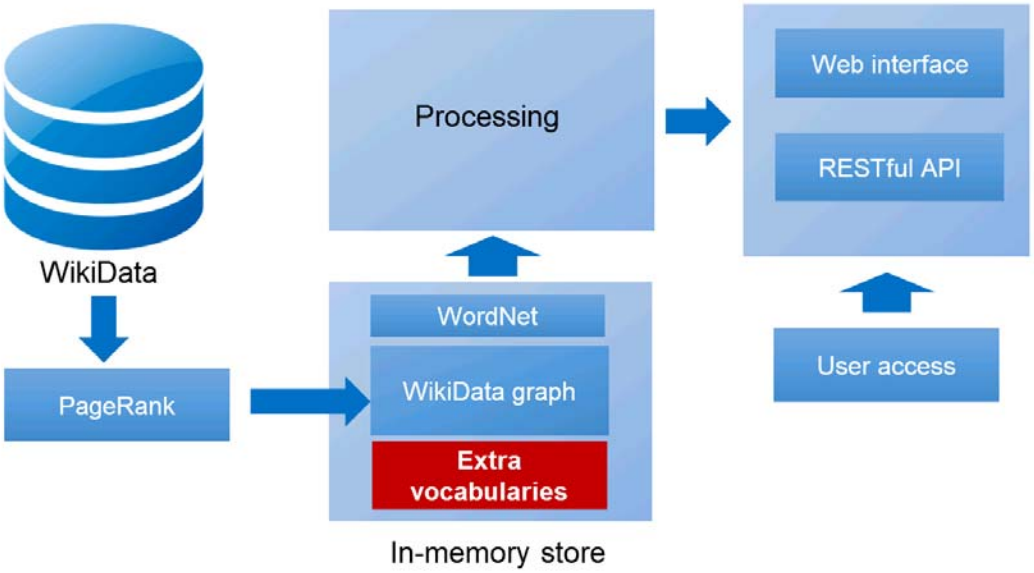


Figure 18: Wikifier architecture

4.4.1.3 Functionality

JSI's Wikifier runs as a Web service and is accessible via API on the following URL: http://www.wikifier.org/annotate-article. When querying the service, many worker threads can run in parallel, as the internal data structures built from Wikipedia are read-only and can be shared between threads. This allows for highly efficient processing of a large number of documents. Our implementation currently processes on average more than 500K requests per day.

Wikifier currently supports a total of 134 languages. These are all languages in which a Wikipedia with at least 1000 pages is available. Annotations are returned in JSON format and can optionally include detailed information about mentions' support, alternative candidate annotations and WikiData/DbPedia class membership of the proposed annotations.

The output of the service is used as a pre-processing step by Event Registry and Graph-based analytics services. Since Wikifier is an important service that tackles cross-linguality and multilinguality, the news and events represented in different languages can be processed with this service and later analysed in cross-cutting business cases analytics tasks.

4.4.1.4 APIs

For Wikifier, we are continuing to develop the Web API documentation, which is written according to the OpenAPI 2.0 standard, and is available online.[25] Each request to the Web API must contain a `userKey` string value that users can obtain for free by registering on the service's Web site.[26]

If users include the correct **`extraVocabularies`** parameter value, Wikifier enables non-Wikipedia vocabularies (e.g. company names from company registers) for the annotation procedure. The Web API call then returns an additional array of entities in JSON under the key `extraVocabularies`. The annotations produced with additional vocabularies for now do not use context provided in entity properties (e.g. company description) for entity disambiguation. The accepted `extraVocabularies` values are not public, as the vocabularies are still under development – the feature can be used upon request.

[25] https://github.com/JSI-EuBusinessGraph/jsi-wikifier-api
[26] http://www.wikifier.org/register.html

JSI Wikifier can be called with HTTP GET request to a URL of the following form:

http://www.wikifier.org/annotate-article?text=...&lang=...&...

or a POST request with a Content-Type of application/x-www-form-urlencoded and pass the parameters (everything after the ? character) in the request body.

Table 15: Wikifier API parameters

Parameter	Description
`userKey`	a 30-character string that uniquely identifies each user (register here to get one). This parameter is required.
`Text`	the text of the document that you want to annotate. Use UTF-8 and %-encoding for non-ASCII characters (e.g. `text=Beyonc%C3%A9`).
`Lang`	the ISO-639 code of the language of the document. Both 2- and 3-letter codes are supported (e.g. `en` or `eng` for English, `sl` or `slv` for Slovenian, etc.). You can also use `lang=auto`
`secondaryAnnotLanguage`	for each annotation, the Wikifier can report, in addition to its name and Wikipedia link in the Wikipedia for the language of the input document, also the name and link of the corresponding page in the Wikipedia for a "secondary" language; the `secondaryAnnotLanguage` parameter specifies the code of this secondary language (default: `en`, i.e. English).
`wikiDataClasses`	should be `true` or `false`, determines whether to include, for each annotation, a list of wikidata (concept ID, concept name) pairs for all classes to which this concept belongs (directly or indirectly).
`wikiDataClassIds`	like `wikiDataClasses`, but generates a list of concept IDs only (which makes the resulting JSON output shorter).
`Support`	should be `true` or `false`, determines whether to include, for each annotation, a list of subranges in the input document that support this particular annotation.
`Ranges`	should be `true` or `false`, determines whether to include, for each subrange in the document that looks like a possible mention of a concept, a list of all candidate annotations for that subrange. This will *significantly* increase the size of the resulting JSON output, so it should only be used if there's a strong need for this data.
`includeCosines`	should be `true` or `false`, determines whether to include, for each annotation, the cosine similarity between the input document and the Wikipedia page corresponding to that annotation. Currently the cosine similarities are provided for informational purposes only and are not used in choosing the annotations. Thus, you should set this to `false` to conserve some CPU time if you don't need the cosines for your application.
`maxMentionEntropy`	set this to a real number *x* to cause all highly ambiguous mentions to be ignored (i.e. they will contribute no candidate annotations into the process). The heuristic used is to ignore mentions where *H(link target*

	| *anchor text = this mention)* > *x*. (Default value: −1, which disables this heuristic.)
`maxTargetsPerMention`	set this to an integer *x* to use only the most frequent *x* candidate annotations for each mention (default value: 20). Note that some mentions appear as the anchor text of links to many different pages in the Wikipedia, so disabling this heuristic (by setting *x* = −1) can increase the number of candidate annotations significantly and make the annotation process slower.
`minLinkFrequency`	if a link with a particular combination of anchor text and target occurs in very few Wikipedia pages (less than the value of `minLinkFrequency`), this link is completely ignored and the target page is not considered as a candidate annotation for the phrase that matches the anchor text of this link. (Default value: 1, which effectively disables this heuristic.)
`pageRankSqThreshold`	set this to a real number *x* to calculate a threshold for pruning the annotations on the basis of their pagerank score. The Wikifier will compute the sum of squares of all the annotations (e.g. *S*), sort the annotations by decreasing order of pagerank, and calculate a threshold such that keeping the annotations whose pagerank exceeds this threshold would bring the sum of their pagerank squares to $S \cdot x$.
`partsOfSpeech`	should be `true` or `false`, determines whether to include information about parts of speech (nouns, verbs, adjectives, adverbs) and their corresponding WordNet synsets. This feature is only supported for English documents; the Brill tagger is used for POS tagging. (Default value: `false`).
`verbs`	like `partsOfSpeech`, but only reports verbs.
`nTopDfValuesToIgnore`	if a phrase consists entirely of very frequent words, it will be ignored and will not generate any candidate annotations. A word is considered frequent for this purpose if is one of the *nTopDfValuesToIgnore* most frequent words (in terms of document frequency) in the Wikipedia of the corresponding language. (Default value: `0`, which disables this heuristic. If you want to use this heuristic, we recommend a value of `200` as a good starting point.)

4.4.1.5 Software license and code repository

JSI Wikifier runs as a RESTful API service, for which a user key is required. The usage of JSI's Wikifier is governed by fair use.

4.4.1.6 Installation and user guide

For JSI's Wikifier no installation is needed as it runs as a Web service. A user guide i.e. documentation is available on its Website:

- http://wikifier.org/info.html

4.4.1.7 Next steps (2nd release)

The next steps for JSI Wikifier development would include the extension of available extra vocabularies.

Another direction for Wikifier development is a better support for languages which have small amount of Wikipedia articles. This could be improved by allowing a second stage of processing, in which Wikipedias in languages other than the language of the input document would also be used to identify mentions and provide candidate annotations.

An additional improvement would be to incorporate local disambiguation techniques for annotations stemming from non-Wikipedia vocabularies as a method of mimicking the current Wikipedia and PageRank-based disambiguation approach.

4.4.2 Event Registry

Event Registry[27] is a system that can analyze news articles and identify in them mentioned world events. The system is able to identify groups of articles that describe the same event. It can identify groups of articles in different languages that describe the same event and represent them as a single event. From articles in each event it can then extract an event's core information, such as event location, date, who is involved and what it is about. Extracted information is stored in a database. A user interface is available that allows users to search for events using extensive search options, to visualize and aggregate the search results, to inspect individual events and to identify related events.

4.4.2.1 Requirements

The requirements for Event Registry service are provided in the table below.

Table 16: Requirements – Event Registry

ID	Requirement	Description
Event Registry		
Req1	Data collection	The extensive coverage of different news sources for data collection (currently the system processes news from over 75.000 news sources).
Req2	Cross-linguality and Multilinguality	The articles in different languages are collected and processed by the service
Req3	Pre-processing	The articles in the mentioned languages are then processed with a set of linguistic tools. A very important component for Event Registry is the named entity recognizer which detects the named entities mentioned in the articles and disambiguates them. Since events are associated with a date we also try to identify in the text date mentions using a set of regular expressions for different languages.
Req4	Event construction	Groups of articles describing the same event are aggregated in the event, information is extracted based on event level.
Req4	Event categorization	Event categorization is an important requirement for business cases analytics services, since it allows obtaining and analysing the relevant subset of event for specific categories of interest (such as business and finance).
Req6	Web API	The service is accessible via API (with an access key). The users should register at the product website.
Req7	Web interface	Event Registry has a web interface that allows for displaying the search results, viewing trending concepts, clustering and tag clouds for events, displaying event and article information.

[27] http://eventregistry.org/about

4.4.2.2 Architecture

Figure 19 presents Event Registry's pipeline.

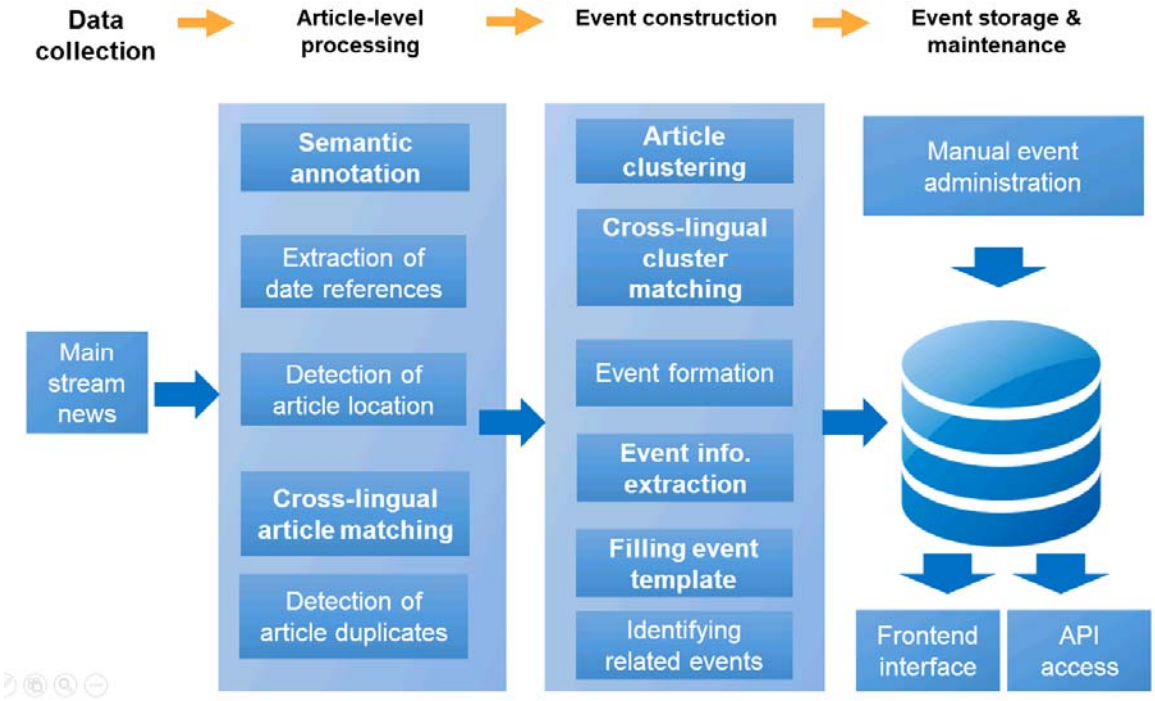


Figure 19: Event Registry architecture

Event Registry architecture includes the following components:

- Mainstream news (the system is provided with news coming from thousands of mainstream sources).
- Article-level processing (including semantic annotation, extraction of date references and locations, addressing cross-lingual articles). The important part of this step is detection of articles duplicates.
- The Event construction part includes articles clustering into events, addressing cross-lingual clusters, formation of events, categorization of event types, creation of event and identifying related events.
- Finally, the system is accessible via frontend interface and API access.

The current Event Registry set up is based on event categorization with DMOZ taxonomy. However, within euBusinessGraph project new mechanisms of event categorization are implemented, allowing more efficient analytics on top of news from specific category (such as business news).

4.4.2.3 Functionality

Data in Event Registry can be accessed through the web interface or directly through the available API. In order to access data in Event Registry an API key is required. Accessing data through the API can be done by issuing HTTP GET requests with specific parameters. There are also Python and Node.js SDK libraries (https://github.com/EventRegistry/).

4.4.2.4 APIs

For Event Registry, we are continuing to develop the Web API documentation, which is written according to the OpenAPI 2.0 standard, and is available online:

- https://github.com/JSI-EuBusinessGraph/jsi-eventregistry-api

Below we describe the basics of Python SDK functionality.

`EventRegistry` is the main class to use in order to send a request to the Event Registry. `ReturnInfo` is a class that can be used to specify which properties should be returned for each returned data type (events, articles, concepts). A complete documentation of the Python SDK is available online.[28]

Table 17: Event Registry API parameters

Parameter	Description
`apiKey`	The API key for your account that should be used when making the requests.
`host`	The URL where the Event Registry is available.
`logging`	Should the executed web requests be logged in the logs folder in the package directory?
`minDelayBetweenRequests`	What should be the minimum delay in seconds between two requests sent to Event Registry.
`repeatFailedRequestCount`	In case a request fails (for example, timeout), how many times should it be repeated before giving up.
`settingsFName`	If provided it should be a full path to `settings.json` file where `apiKey` and/or host can be loaded from. If `undefined`, we will look for the settings file in the `eventregistry` module folder.

Table 18: ReturnInfo Arguments

Parameter	Description
`ArticleInfoFlags`	details about the articles to return
`EventInfoFlags`	details about the events to return
`SourceInfoFlags`	details about the news sources to return
`CategoryInfoFlags`	details about the categories to return
`ConceptInfoFlags`	details about the concepts to return
`LocationInfoFlags`	details about the locations to return
`StoryInfoFlags`	details about the stories to return
`ConceptClassInfoFlags`	details about the concept classes to return
`ConceptFolderInfoFlags`	details about the concept folders to return

Table 19: Relation extraction API parameters

Parameters	Description
/extractRelations	Extract relations from a given text
text	Document text to detect relations from
types	An array of relation types to detect; empty defaults to detecting all available relation types

[28] https://github.com/EventRegistry/event-registry-python

documentInfo	Document source information, including this object commits results of this operation to the database
/getRelations	Get relations of a given type for a given entity label
entityLabel	Entity label text
relationTypes	An array of relation types to return for the given label; empty defaults to detecting all available relation types

4.4.2.5 Software license and code repository

Event Registry can be access in free use and commercial use.

Event Registry allows new users certain number of search operations for free in order to be able to validate suitability of the service for your purpose. The users of the free access should not use the service for any commercial or scientific research purposes. The users of the free access should not try to overcome the free limit by creating multiple accounts or accessing the service through different IP addresses.

Free access restrictions. The data provided by the free access is limited to the articles and events from the last 30 days. Unregistered users can in total use up to 50 tokens (see usage tracking for explanation of tokens). Afterwards, registration for at least a free account is required. A user registered with a free account can use up to 500 tokens per day and can use up to 2,000 tokens total. After using these tokens, a paid account will be required for continued use.

Research collaboration. Event Registry welcomes any potential collaboration with other research individuals and organizations.

For **commercial** purposes, the users can subscribe to our service.

4.4.2.6 Installation and user guide

The frontend of Event Registry can be accessed at http://eventregistry.org. The installation and user guides are provided at GitHub (https://github.com/EventRegistry).

4.4.2.7 Next steps (2nd release)

The next steps of Event Registry development include improvements in event categorization for specific events.

4.4.3 Graph Based Analytics

The aim of graph-based analytics services is to cover:

- relation extraction
- clustering
- similarities
- connections between entities

In particular, the relation extraction service is operating on local data and discovering the relations between entities (companies, people and products/brands).

4.4.3.1 Requirements

The requirements for Graph-based analytics service are provided in the table below.

Table 20: Requirements – Graph-based analytics

ID	Requirement	Description
Graph-based analytics		
Req1	Design Requirements/REST inspired	The service should provide a RESTful API.
Req2	Design Requirements/JSON	The service should provide results in JSON format.
Req3	Cross-linguality and multilinguality	The important aspects of the relation extraction service are related to the characteristics of the available datasets, such as multi-linguality and unstructured nature of the data.
Req4	Documentation Requirements	The API access should be clearly documented for external users. The documentation can have interactive character.
Req5	Process Requirements – API development	API should be developed and tested in phases.
Req6	Process Requirements – Usage of GitHub	GitHub repository can be used for managing the project.
Req7	Process Requirements – Documentation of changes	Changes and errors should be documented.
Req8	Relation extraction	The service should use machine learning and data mining methods for relation extraction.
Req9	Business cases analytics	The service should provide analytical solution on top of graph data.

4.4.3.2 Architecture

The majority of work on Graph Based Analytics service has been currently concentrated at relation extraction tasks that are the basis for any analytical graph functionality.

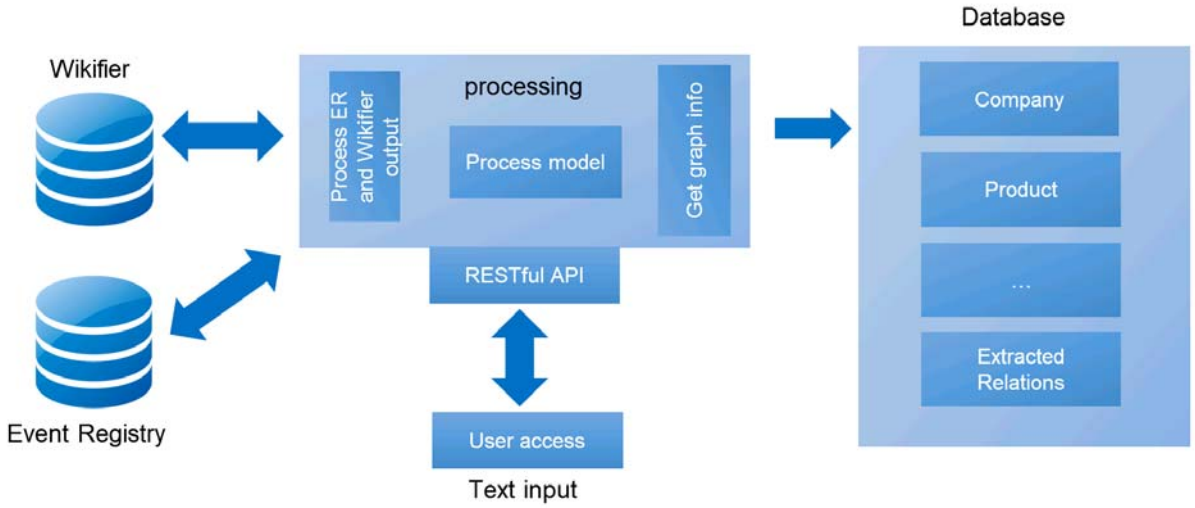


Figure 20: Graph-based analytics architecture

Figure 20 presents the architecture for the Graph Based Analytics service.

The architecture of the prototype version of relation extraction services included the following components:

- Annotation of news with Wikifier (including synsets detection from Wordnet).
- Using events and articles from Event Registry for relation extraction.
- Database that stores information for companies and extracted relations.
- User access via RESTful API.

For entity extraction we have introduced **derived entity label generation** as an additional entity label pre-processing task. By enriching entity instances in Wikifier-ready datasets with derived entity labels, we can increase search recall of entities in unstructured data

During relation extraction development we have pivoted to acquiring and working with labelled data. This data was incorporated into our current approach to relation extraction modelling, outlined in Next steps.

For modelling the first version of relation extraction we have used word embeddings in a shallow linear neural network layer architecture.

4.4.3.3 Functionality

Motivation for derived entity label generation is two-fold:

- The entities from the datasets that we have analysed in general come with only one complex string label. However, what is often found in unstructured data (e.g. news articles) is not necessarily the full title of a company or a product. For example, journalists often refer to company entities by their brand name, trademark, or common name, which are often substrings of the full entity label. News search should support annotation of substring variations of entity labels.
- Some entity labels have very common dataset-specific substrings, such as company titles having a suffix that describes their corporate status (e.g. INC.). These types of dataset-specific artefacts can lower search recall in unstructured data.

We conclude that entity extraction systems with a dictionary-based approach such as JSI's require simplified entity labels for higher recall.

Derived entity label generation starts by tokenizing entity label text, then clusters the tokens into separate groups. Token scoring is required for token clustering, for which we use the Term Frequency - Inverse Document Frequency (TF-IDF) approach. Figure 21 demonstrates how hierarchical clustering

separates an entity label's tokens scored with TF-IDF into groups, which we can then arbitrarily combine to derive new labels, whilst keeping their sequential order.

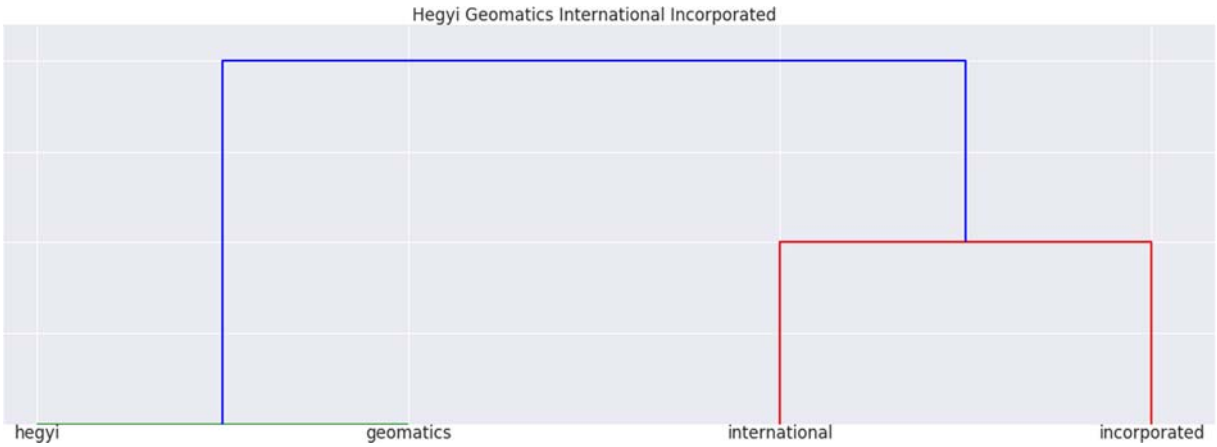


Figure 21: An entity label's tokens clustered into groups, displayed with a dendrogram

We designed the dataset pre-processing stage to let analysts set a relative token frequency threshold according to the analysis of the dataset, which discards derived entity labels whose tokens have a relative frequency higher than the given threshold. This helps to ensure that derived entity labels do not include dataset-specific stopwords.

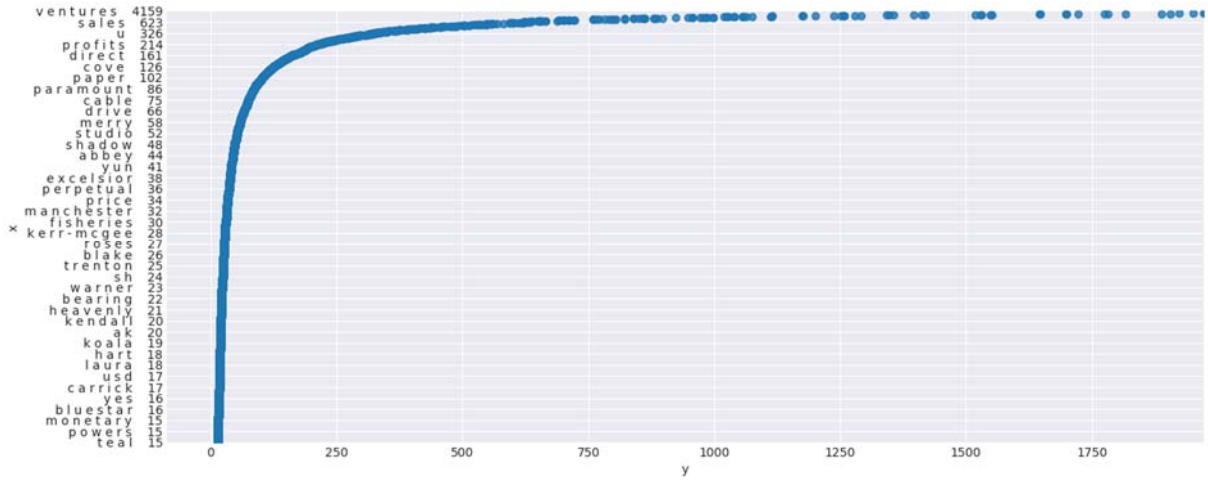


Figure 22: An interactive decision-making utility for setting the token frequency threshold

The current methodology for relation extraction includes learning of specified relations (such as mergers and acquisitions) based on a labelled dataset (Crunchbase 2013 Snapshot is used as a data source), with involving of active learning.

Figure 23 presents a snapshot of relation extraction example from Event Registry news.

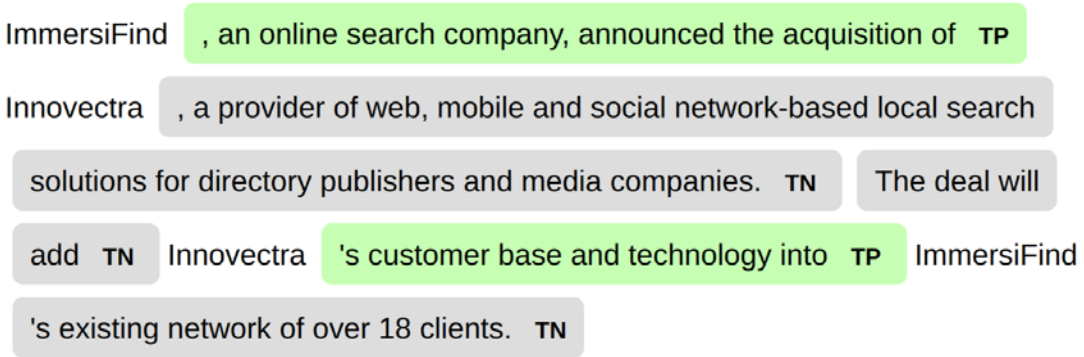


Figure 23: Relation extraction example

Machine learning and data mining techniques are used based on a set of features obtained from JSI Wikifier. A detailed description of implemented techniques will be provided in the deliverable **D2.2 Cross-lingual / Multi-lingual Data Management Approach for Structured and Unstructured Data**.

4.4.3.4 APIs

For Graph-based analytics, we are continuing to develop the API documentation, which is written according to the OpenAPI 2.0 standard, and is available online:

- https://github.com/JSI-EuBusinessGraph/jsi-graph-based-analytics-api

Derived entity label generation is implemented as a Python module and is primarily intended to be a dataset preprocessing step. An internal command-line utility script was developed for ease of use. The utility takes several positional and optional arguments, some of which are listed in the table below.

Table 21: Command-line options for the Derived Entity Generation utility

Argument name	Description
`--taskName`	The task that is to be executed. Values: • **countTokens** - Counts space-separated tokens in the dataset and outputs `-n` most common tokens and their counts; useful for dataset analysis. • **generateNewLabels** - Generates a Wikifier-compliant input JSON file with derived entity labels; optionally `--elbow` accepts a token frequency value and skips the interactive utility for analysis of the input dataset, which is then used to calculate the relative token frequency threshold for discarding certain derived entity labels.
`--dendrogram`	Visualizes each entity label tokenized and clustered as a dendrogram. Mainly used for analysis and debugging.
`--help`	Displays help, lists all command-line options.

Table 22: Relation extraction

Parameters	Description
/extractRelations	Extract relations from a given text
Text	Document text to detect relations from
Types	An array of relation types to detect; empty defaults to detecting all available relation types
documentInfo	Document source information, including this object commits results of this operation to the database
/getRelations	Get relations of a given type for a given entity label
entityLabel	Entity label text
relationTypes	An array of relation types to return for the given label; empty defaults to detecting all available relation types

4.4.3.5 Software license and code repository

The software license will be defined at a later time. Code is written in C, Python and Java, and stored in local SCM repositories.

4.4.3.6 Installation and user guide

The Graph-based analytics service is deployed on JSI servers. Users are able to access the relation services through the RESTful API.[Error! Bookmark not defined.] User registration to obtain an API access key will be required shortly.

4.4.3.7 Next steps (2nd release)

The next steps for the Graph Based Analytics relation extraction service would be to obtain or generate better relation-specific datasets, improve the deployed algorithms, and continue research into different machine learning techniques, statistical analysis and pattern detection paradigms.

The next step for derived entity label generation is to upgrade the annotation process for Wikifier's business related extra vocabularies extension. The upgrade should produce more context-aware annotations, similar to those produced by the JSI Wikifier's default "wikification" functionality. This could be achieved by implementing an algorithm on top of document comparison techniques, such as cosine similarity-based document search.

For graph-based analytics, we are in the process of developing a relation extraction focused methodology for dataset generation and a modelling framework. The current development vector assumes we will use the Crunchbase 2013 Snapshot[29] as a distant supervision seed, with which we will be able to generate a robust dataset for relation extraction modelling. This step will require us to use JSI's Wikifier for annotation of Event Registry data. By using Event Registry's multilingual news Event clusters, we expect to be able to apply the relation extraction service in a multi-lingual scenario.

4.5 Marketplace and Operational Services

The Marketplace and Operational Services as defined in Deliverable 3.1 have been our first attempt to collect requirements and address the design issues of the system. In the current deliverable more work has been put into shaping the specification of the Marketplace and correspondingly we present our current version of the marketplace services design in terms of a component architecture diagram. This section also represents the current and respectively intermediate (working version) specification including definitions of the marketplace services API that will be finalized for the upcoming Deliverable 3.3.

Table 23 lists the software components and how they map to the Cross-Cutting Business Cases Analytics Services of Figure 4.

Table 23: Software components – Marketplace and Operational Services

Software component	Implemented service	Implementation status
Congnitive Cloud: Licence Models Component	Licence Models	In development
Cognitive Cloud / GraphDB Cloud Administration interfaces and API	Platform Administration	Developed
Cognitive Cloud Security component	Security and Access Control	Developed
Cognitive Cloud: Monotoring and Reporting (based on AWS Cloud Watch)	System Monitoring & Reporting	In development

4.5.1 Marketplace on the Cognitive Cloud

4.5.1.1 Requirements

Detailed description of the different stakeholders and their requirements are provided in section 2 "euBusinessGraph stakeholders" and section 3 "Requirements specification" of Deliverable D3.1. Here we will provide only requirements that are important for the Marketplace and Operational Services.

The different stakeholders involve the data providers, the data consumers, and the technology providers.

[29] Distributed under Creative Commons Attribution License [CC-BY] (© 2013)

4.5.1.1.1 Data provider requirements

Here we present a refined set of requirements provided by the data providers.

Table 24: Data provider requirements

ID	Requirement	Requirement description
Marketplace and Operational Services		
DPR-11	License models	Support dual license models that allows for both payment or share-alike for public-benefit use of the datasets uploaded by data providers.
DPR-11-a	Dataset-level license access control	Access control policy at a database or repository level. .
DPR-11-b	Data item license access restrictions	Ability configure types of data data consumer will receive after payment or agreeing with a particular licence model.
DPR-11-c	Advertise company data	Revenue originating from data flows is shared.
DPR-11-d	Shared revenue	Security and access control for users .
DPR-12	Security and access control	Manage access of users to APIs .
DPR-13	User registration and access management	Support dual license models that allows for both payment or share-alike for public-benefit use of the datasets uploaded by data providers.

4.5.1.1.2 Data consumer requirements

The data consumers have the following requirements:

Table 25: Data consumers requirements

ID	Requirement	Requirement description
Marketplace and Operational Services		
DCR-08	License models	Support for different license models for accessing and APIs for accessing and modifying datasets.
DCR-08-a	Shared agreement models	Integrated navigation to data with shared business/revenue agreements
DCR-09	Secure access to platform APIs	API keys allowing to query, navigate and download datasets.
DCR-10	User registration and access management	User sign-up, log-in and profile management.

4.5.1.1.3 Technology provider requirements

The technology providers have requirements for data analytics services that need access to the following types of data:

- Company info
- People info
- Product/brand info

4.5.1.2 Architecture

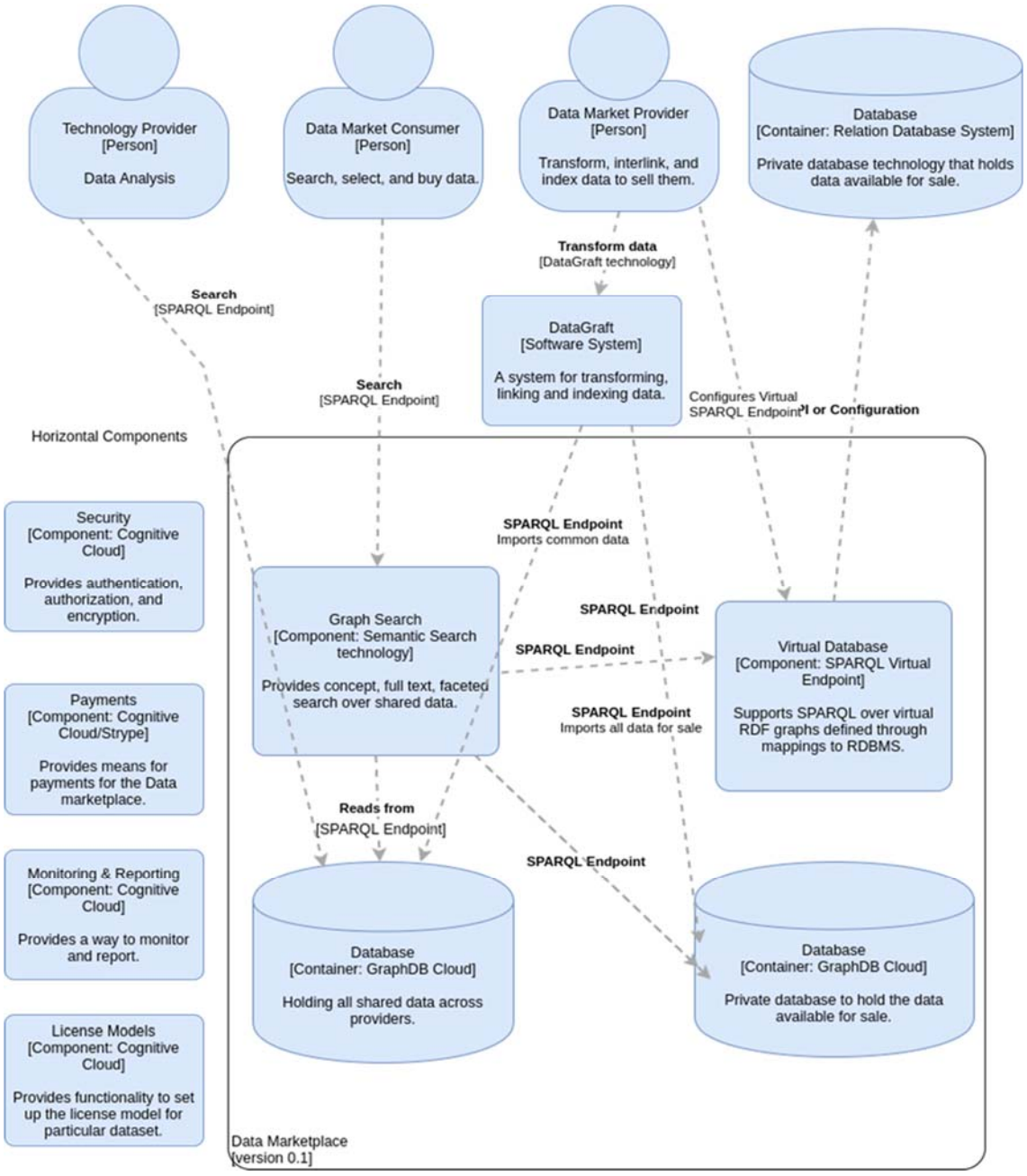


Figure 24: Component Diagram of the Marketplace and Operational services[30]

The euBusinessGraph Marketplace provides different graphs of data which can be analysed, searched and explored by invoking queries directly across a set of diverse datasets sharing common properties. The current understanding between the partners of euBusinessGraph is that the marketplace will adhere to a **"common shared dataset"** principle. That means that data providers joining the business graph will share a common subset of their data described according to a common data model.

The common shared data are kept in GraphDB database in the form of graph of linked data as defined in "euBusinessGraph Semantic Data Model". This instance of GraphDB will be called a master, will serve as master data provider and will defer to proprietary and fully manged GraphDB Cloud instances on the Ontotext Cognitive Cloud or to a virtual database to request additional data.

[30] C4 model for software architecture - https://c4model.com/

The individual Datamarket data providers will decide whether to transform and upload all of their data or to create a virtual database to proxy their relational database instance. The proxy will hold the data available for sale on the Marketplace. In both cases the data provider must transform their data into a canonical semantic form and link it according to the common shared data model, before making it available to the master database service.

All databases, including the virtual endpoints, will expose a SPARQL endpoint. The virtual database endpoint will be implemented as a set of APIs that the Data provider must implement or query translation engine such as the open source ONTOP[31] framework.

The Market consumer will search for data via search user interfaces and/or APIs provided by the Graph search component. The available functionality is:

1. full text search in the indexed shared data fields such as name, description, address, etc.;
2. concept search for name of organisation, location in address, etc.;
3. faceted search to filter data based on various conditions such as organisation name, country, postal code, various classifications, etc.

The search component will allow users to query the master GraphDB. The Data consumers will use this component to search for data. When users choose to buy data, this component will be responsible for collecting all data from the data provider and provide it for downloading in multiple formats. These format include but are not limited to various flavours of RDF, JSON, JSON-LD, and XML, etc.

The search component will combine intelligent scoring and ranking of results by applying different masking weights on the concepts and full text individual rankings of results for the benefit of better retrieval of data.

The *security component* is responsible for ensuring the proper authorization and authentication for accessing all public service on the platform. As an additional security measure, all access to the platform services will utilize transport encryption (SSL/TLS). The components of the system include:

- Account management – all users will have personal accounts for accessing the euBusinessGraph platform that will be managed by the security component;
- Authentication and authorisation – all access to the platform marketplace portal will be authenticated via a user name and password, while all access to the REST services exposed on the platform will be authenticated via private API key/secret pairs.
- Datasets will offer different access levels: public for open source datasets, and private available to the data consumers after payment. Appropriate authorisation measures will be taken to ensure that users have access only to appropriate data.

The *monitoring and reporting* component is responsible for monitoring all operations on the platform. Its components include:

- Service monitoring – for ensuring that services are operational and operate within expected performance levels (platform operators should be notified if services are down or operating with deteriorated performance);
- Usage monitoring and logging – all access to platform resources should be logged, so that various reports, billing information and audits may be prepared;
- Quota management – usage should be restricted within the predefined quota limits for the various services;
- Reporting – various system reports will be available to platform operators with daily/weekly/monthly statistics about platform usage. These reports will be generated based on input parameters and a set of APIs supported by the platform;

The *license models* component is responsible for managing and enforcing data access based on license models and agreements. The platform should support different license models such as:

[31] https://github.com/ontop/ontop

- Free access to public open data;
- Payment models restricting access to data based on payments; and
- Dual license models, allowing payment plus share-alike for public-benefit use.

According to the requirements from the data providers, license models for full datasets are not sufficient, as there is a need to have different licenses models at the property level in the datasets.

The *Payments component* will be responsible for managing the payment process by providing the capability for credit card payments. The system will not keep any credit card information and will use Stripe[32] as a payment provider. The Marketplace will provide two business models: subscription and pay per data usage. Pay per usage will be measured by the number of data records (number of triples) downloaded by a Data consumer. In addition the payment component will be responsible for the billing process. It will aggregate all the usage of a particular account on a monthly basis in the pay per use case. In that way, based on predefined billing metrics such as volume of data accessed, number of queries, or number of datasets accessed, a usage cost can be calculated for the account and packaged as a monthly bill. This component will issue invoices.

4.5.1.3 Functionality

The *Marketplace and Operational Services* will ensure the availability of a data brokerage system in the form of a data marketplace where data and datasets that are part of the business graph can be provisioned and accessed. The focus here will be on the implementation of a mechanism for controlled access to business graph data, together with services for user management and data access mechanisms. In addition, the operational services needed for the marketplace will be addressed. Components for platform monitoring, availability, administration quota enforcement, branding and billing will be created. The underlying infrastructure of the marketplace will be based on the Cognitive Cloud[33] and DataGraft.

For euBusinessGraph, our minimal viable product (MVP) is an attempt to deliver the core features outlined in this deliverable. The aim of this MVP is to explore the various business models based on how much data will be shared to attract potential data consumers on the Marketplace. This MVP is meant to evolve into the (meta) Search component of the Marketplace. The MVP is described in Section 6.2.

The Marketplace will implement the following scenarios.

- **Scenario 1**. A data producer has data for sale. She will export the data from her relational database system that keeps the data and will transform and interlink to the Marketplace schema with DataGraft. Then she will upload the data tshe wants to share with data consumers to the master GraphDB database. The rest of the data will be uploaded to a private GraphDB Cloud instance that she will provision via the Cognitive Cloud. The Data provider will pay for the GraphDB Cloud instance and the possibility to provide her data for sale via the master database.
- **Scenario 2**. A data producer has data for sale, but there are business or regulatory limitations to share all of the data in a provisioned instance of GraphDB Cloud. She then will export the data she wants to share with the data consumer and will link and transform it to the master schema. She will then configure a virtual database on the Marketplace that will link her internal relational database with the search component. The Data provider will pay for the virtual database and the possibility to provide her data for sale via the master database.
- **Scenario 3**. A data consumer needs organisational data about companies from particular sector in a London district. She will use the faceted search user interface to filter organisations by location and sector. There will be data about such companies from two providers. She will explore the available data by each provider and will choose to buy the data from the one of the providers. Since the data consumer is using a subscription she will not pay for the data immediately. The billing component will charge her credit card beginning of the month and will issue and invoice for the payment.

[32] https://stripe.com/
[33] https://ontotext.com/products/cognitive-cloud/

4.5.1.4 APIs

The market place will implement the following APIs:

4.5.1.4.1 Security and access control

The API specification for security and access control defines methods for account and API key resources. If an error occurs, a 'message' containing an explanation of what went wrong (e.g., "incorrect username or password", "expired session", etc.) will be included in the resulting output.

Table 26: Security and access control APIs

Methods	Resource	Description
PUT	/accounts/login	Performs log-in with username and password or uses a pre-existing social network ID (Google+ ID, Twitter ID or Facebook ID). • Input: o 'username' – username. o 'password' – password. o 'google_id' – Google+ ID (alternative). o 'twitter_id' – Twitter ID (alternative). o 'facebook_id' – Facebook ID (alternative). • Output: HTTP result code (with 'message').
POST	/accounts	Creates a new account with username and password or associates a pre-existing one (Google+ ID, Twitter ID or Facebook ID). 1. Input: a. 'username' – user name. b. 'password' – password. c. 'google_id' – Google+ ID (alternative). d. 'twitter_id' – Twitter ID (alternative). e. 'facebook_id' – Facebook ID (alternative). f. 'role' – user role. g. 'name' – full name of user. h. 'email' – email address of user. 2. Output: HTTP result code (with 'message').
PUT	/accounts/logout	Performs log-out. 3. Output: HTTP result code (with 'message').
GET	/accounts/login_status	Gets login status. 4. Output: 'status' – "authenticated" or "not authenticated"
GET, PUT	/account/details	GET retrieves account details. 5. Output: Account details with username, email, name, role, phone, address, etc. PUT updates account details: 6. Input: a. 'username' – user name. b. 'password' – password. c. 'role' – user role. d. 'name' – full name of user. e. 'email' – email address of user. f. 'phone' – phone number g. 'address' – address 7. Output: HTTP result code (with 'message').
PUT	/accounts/password	Changes the password. • Input: 'new_password' – new password. • Output: HTTP result code (with 'message').

POST	/accounts/password/reset	Requests a password reset. • Input: 'email' – email address. • Output: HTTP result code (with 'message').
PUT	/accounts/password/confirm	Confirms a password reset trequest. • Input: o 'email' – email address. o 'new_password' – new password. o 'token' – verification token. • Output: HTTP result code (with 'message').
GET, POST	/api_keys/	GET retrieves API Keys. • Output: List of API keys POST creates new API key • Output: New API key record o 'api_key' – key id o 'secret' – secret phrase
POST	/api_keys/temporary	Creates a temporary API Key (expires after 24h). • Output: New API key record o 'api_key' – key id o 'secret' – secret phrase
PUT	/api_keys/<api_key>/enable	Enables an API Key. • Output: HTTP result code (with 'message').
PUT	/api_keys/<api_key>/disable	Disables an API Key. • Output: HTTP result code (with 'message').
DELETE	/api_keys/<api_key>	Deletes an API Key. • Output: HTTP result code (with 'message').

4.5.1.4.2 Usage reporting

The API for usage reporting provides a method that provides usage reports of the platform resources (e.g., number of requests, incoming/outgoing traffic) that are automatically logged by the euBusinessGraph platform.

Table 27: Usage reporting APIs

Methods	Resource	Description
GET	/usage	Gets the usage report in the given period. • Input: o 'from' – from date. o 'to' – to date. o 'resources' – list of resource types (or all) that should be included in the report. • Output: Usage report with user information (user ID, name, email) and usage records (start time, end time, number of requests, incoming traffic, outgoing traffic, SPARQL requests, SPARQL traffic, transformation requests, transformation traffic, etc.)

4.5.1.5 Software license

The Marketplace will be delivered with commercial licence and freemium business model. In this model some of the data providers will sell their data, and some will provide open data and with dual licences paid and free of charge for particular types of usage, for instance for academic use.

4.5.1.6 Installation and user guide

Detailed user guide will be provided to both Data providers and Data consumers.

4.5.1.7 Next steps (2nd release)

Currently, only the data hosting layer, master data, security, payments and billing, and monitoring components are implemented. The rest of the components, as depicted in Figure 1, will be implemented and configured in fully working system for the second release.

5 Conclusions

This document provides an overview of the euBusinessGraph Marketplace platform, introducing the relevant stakeholders of the platform, and outlining a set of requirements for the platform from the perspectives of the stakeholders.

Furthermore, the document provides an updated architecture design for the platform, together with a updated set of APIs for the components of the architecture. The architecture, components, and APIs will evolve during the course of the project, as the business cases will become more mature.

The requirements, architecture, and APIs outlined in the document will serve as input for the next implementation phase of the platform resulting in Deliverable D3.4 in month 24.

It should be noted that the technical results in the project should not be to solve the individual business cases, but focus on making the business graph and its services as smart and useful as possible to the business cases. The different business cases are different in nature, but should represent a diverse set of needs and requirements that can be used as input for the technical solutions to be developed in the project.

A summary of the implemented services and their implementation status is given in Table 28. The table lists all the software components described in this report.

Table 28: Summary of implemented services and implementation status

Software component	Implemented service	Implementation status
Grafterizer 2.0	Data Import	Grafterizer 2.0 currently supports import of CSV files. Files and transformations can be stored, shared, and reused in DataGraft.
Grafterizer 2.0	Data Cleaning	The current implementation of Grafterizer 2.0 includes functionality for suggestion-based tabular data cleaning, visual data profiling for data quality assessment, and semantic annotation of CSV data to RDF knowledge graphs.
ABSTAT	Real-time vocabulary autocomplete using KG summaries	Implemented as a dedicated service accessible via API
ABSTAT	Search of vocabulary terms using KG summaries	Implemented as the ABSTAT search component
ABSTAT	Statistics about KG structure and vocabularies	Implemented as ABSTAT APIs.
ASIA	Schema -level semantic annotation (schema-level linking)	Implemented in manual annotation mode: ASIA provides suggestions via ABSTAT autocomplete service.
ASIA	Instance-level semantic annotation (with focus on company identifiers)	Not implemented in this version
ASIA	Data Transformation (RDF-ization)	Under development: mapping from schema-level annotations to Grafterizer data transformation and its implementation will be released at M13.
GraphDB Cloud	DataQuery (SPARQL)	The service is ready.
Cognitive Cloud/DataGraft	Data Access	API keys

DataGraft	Data Updates	DataAccess component will be provided by integration of DataGraft and GraphDB Cloud. The scope of the first release is to facilitate the data providers to ETL data via DataGraft into GraphDB Cloud. GraphDB Cloud exposes a well defined database operation API based on RDF4J API.
DataGraft	Metadata	This component needs more requirements and specifications and will be provided in the second release. This component overlaps with the master database from the Marketplace and operational services as depicted in Figure 24.
GraphDB Cloud/Cognitive Cloud	RDF store	The service is ready. However changes in the requirements throughout the project can be addressed at a later stage.
Wikifier	Multi-lingual annotation	Technical readiness (TRL7 – subsystem development)
Graph Based Analytics	Relation extraction	TRL 4 – technology development
Graph Based Analytics	Graph analytics	TRL 3 – research to prove feasibility
Congnitive Cloud: Licence Models Component	Licence Models	In development
Cognitive Cloud / GraphDB Cloud Administration interfaces and API	Platform Administration	Developed
Cognitive Cloud Security component	Security and Access Control	Developed
Cognitive Cloud: Monotoring and Reporting (based on AWS Cloud Watch)	System Monitoring & Reporting	In development

Appendix A Business Model for euBusinessGraph Marketplace

A.1 Product Vision Board

Product envisioning describes what the future product will look like and do.

euBusinessGraph is positioned to be the key initiative to simplify cross-border and cross-lingual collection, reconciliation, aggregation, and provisioning of company-related data from (authoritative and non-authoritative) public and private sector sources. The aim of euBusinessGraph is to enable cross-sectorial innovation based on company-related data.

The target customers of the data marketplace cover a wide range of business entities, which have processes, products or services that incorporate company-related data in their offerings and operations. This could potentially be customers that consume and provide data, or service providers that perform various types of data service on the data available on the data marketplace.

In one sentence, the vision statement for euBusinessGraph is to deliver world-class company datasets and analytics across borders and languages (as shown in Figure 16).

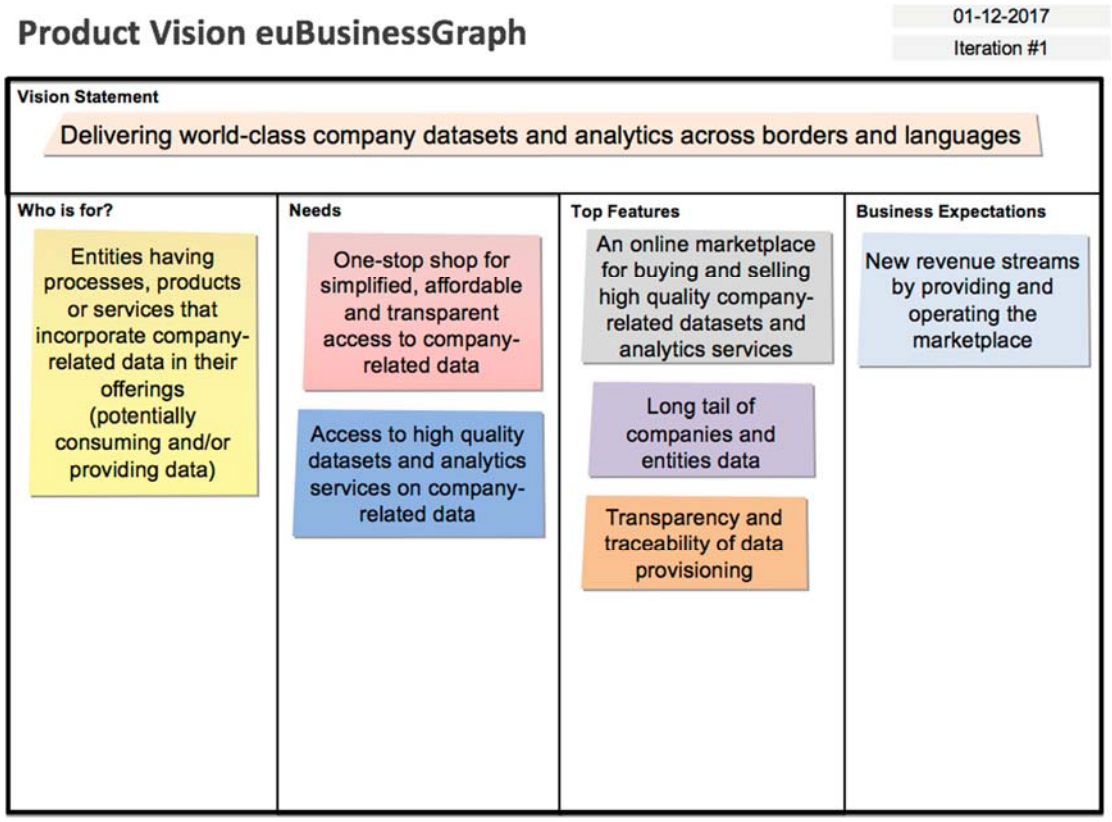


Figure 16: euBusinessGraph Product Vision Canvas

A.2 Lean Business Model Canvas

The lean business model canvas is a chart with elements describing euBusinessGraph Data Marketplace's value proposition, customers and finances, the problem the data marketplace is trying to solve, the solutions it presents, key metrics, and competitive advantage (Figure 17). This canvas is a fast and effective way to communicate the business model of euBusinessGraph Data Marketplace in graphical form to internal and external stakeholders.

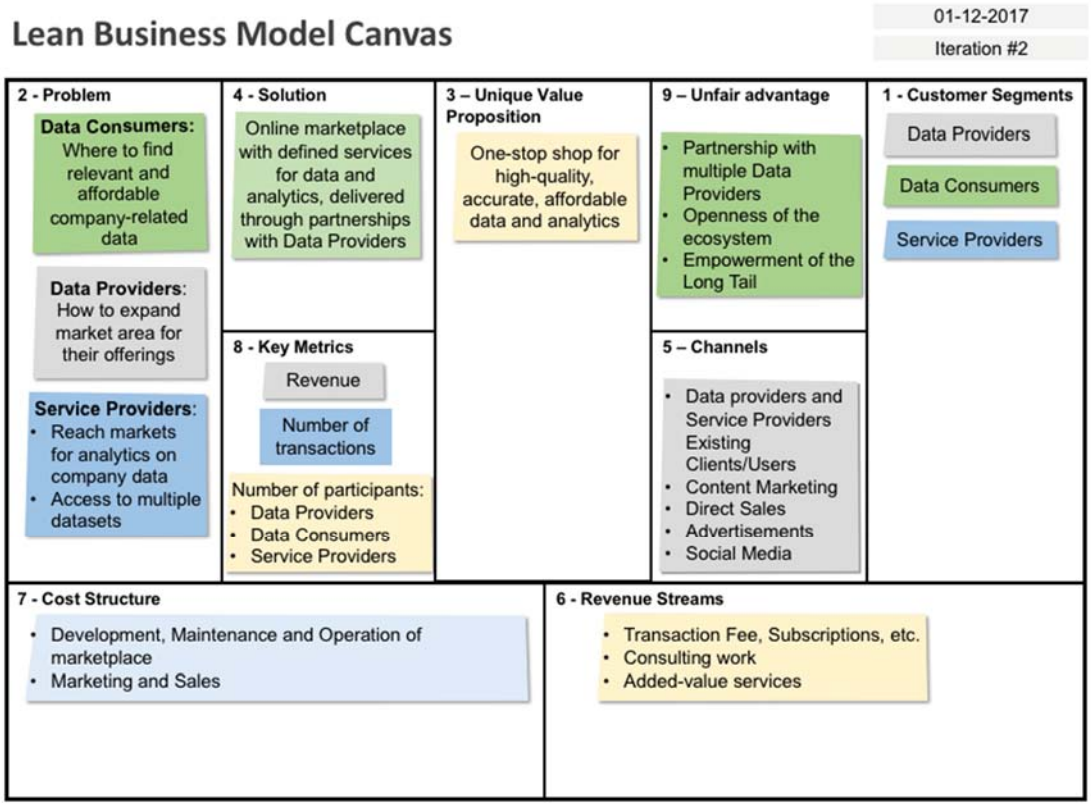


Figure 17: euBusinessGraph Lean Business Model Canvas

A.3 Value Proposition Canvas

The Value Proposition canvas describes how we are going to create values for the three user segments identified for the euBusinessGraph Data Marketplace. The three user segments are:

- Data Provider – this group of users has data which they would like to monetize, a data marketplace such as euBusinessGraph could provide a channel for them to achieve this. For example, an organisation which has information on outstanding invoices in their Enterprise Resources Planning (ERP) solution and provide this information to credit scoring companies at a fee.
- Data Consumer – this group of users require certain types of business-related data in the daily operation of their organisations to help them make better business decision. For example, an organisation gets credit scoring report at a fee to evaluate the strength of the financial footing of a potential customer or vendor. Such report can help them to analyse the risk they are being exposed to in every new business venture.
- Service Provider – this group of users can provide better visualisation of the data available on the data marketplace. They can potentially provide various data services on the vast amount of data available on a data marketplace such as data cleaning and data quality assurance.

This type of canvas consists of a Customer Profile describing the user segments in the business model and a Value Proposition Map describing features of a specific value proposition for a product. We created three canvases (Figure 18, 19 and 20), one for each of the three user segments we identified above.

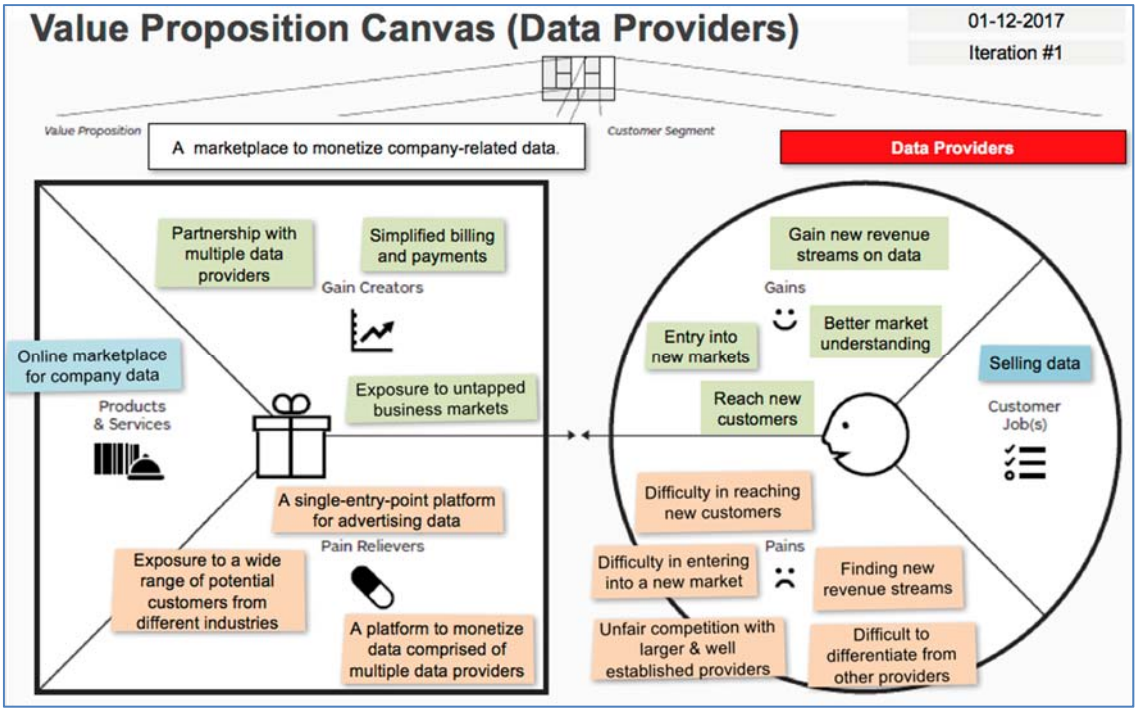


Figure 18: Value Proposition Canvas (Data Providers)

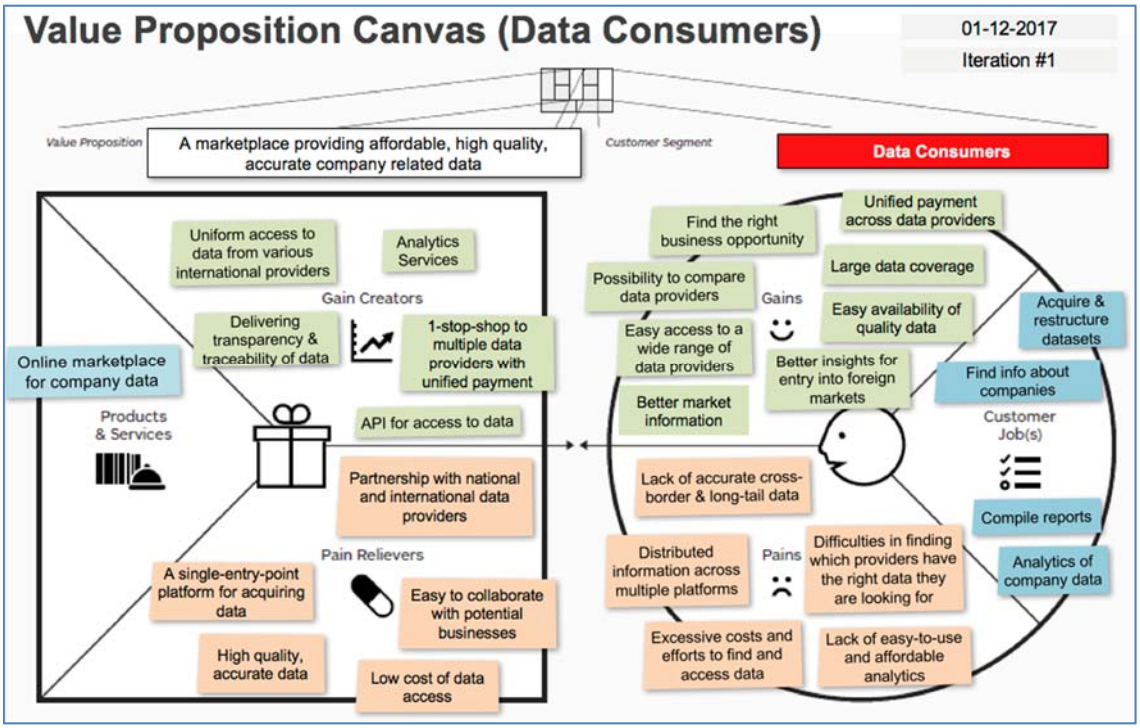


Figure 19: Value Proposition Canvas (Data Consumers)

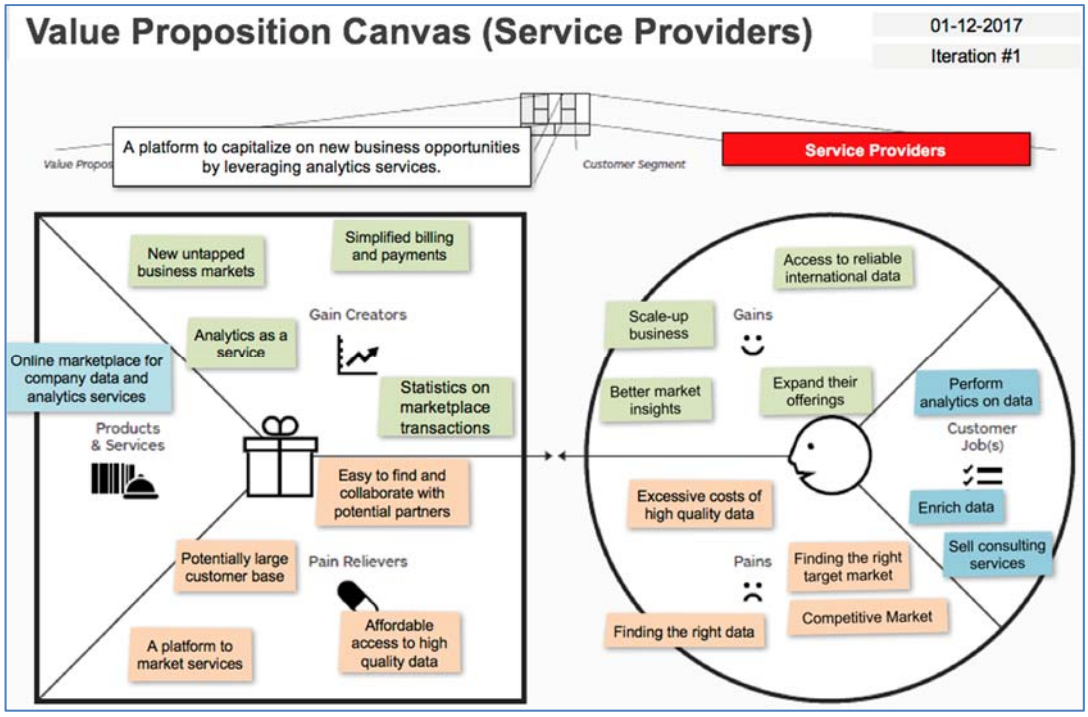


Figure 20: Value Proposition Canvas (Service Providers)

A.4 Issues to be decided

A.4.1 Organisation

It might be beneficial if the marketplace is established as a separate business entity, outside the consortium. The marketplace could either be established as a stock holder company, or as a cooperative society. For transparency and ease of management, the former would be most appropriate.

Another possible model is to let a given partner set up and administrate the marketplace on behalf of the other consortium partners. This partner would administrate the governing of the marketplace and the sale of data on behalf of the other partners and would need to cover their costs out from the generated revenue. EVRY's InfoTorg subsidiary is a good candidate for this model, as an established vendor of third-party business data.

A.4.2 Charter

The charter needs to have a clear statement on what the purpose of the marketplace is, the scope and the limitations as well as the statutes governing the organisation of the marketplace. A governing board should be selected, drawing its members from the various organisations behind the marketplace.

A.4.3 Operating Model

Three different operating models have been suggested:
1. One partner in the consortium as the operator of the marketplace.
2. Two or more partners in the consortium as the operators of the marketplace.
3. An organisation outside the consortium becomes the operator of the marketplace.

All three models have their limitations. The two main dimensions in this is the cost and profit sharing structure, and the licensing and ownership of data. In order to ensure a fair and transparent structure, and to avoid transfer of ownership of data and responsibility from one partner to another, a separate governing body and separate organisation is needed. This body must be owned by all partners with equal access and equal rights.

This governing body could then purchase needed services from either one or more of the partners in the consortium, or from an external service provider. This would ensure a needed degree of flexibility to

develop the marketplace autonomously outside the business interests of given partners and would also provide transparency with regards to the profits and costs.

A.4.4 Business Model

We suggest a subscription model for the marketplace, where customers choose a license fitting their needs, and is billed monthly, semi-annually or yearly for this license. The subscription cost should be competitive with vendors such as Dun & Bradstreet, etc.

The profit should be distributed to the shareholders either per share, or per actual use of their data.

A.4.4.1 Licensing Models

The licensing models should be simple to understand and use. Some data are already public domain (such as the Norwegian Business Register) and should probably be offered for free, while other data is considered proprietary. Depending on the type of data and the ownership, several license tiers could be considered. Another issue is the number of monthly requests a user performs towards the service, which should be billed accordingly.

A.4.4.2 Suggested License tiers

- Free: Public domain data only
- Business: Public domain data, as well as some value-added data
- Enterprise: Full access to everything

Other tiers should be considered

A.4.4.3 Suggested Request tiers

- Free: Less than 5000 requests per month
- Small: 5000-10000 requests/month
- Medium:10000-50000 requests/month
- Large: 50000-250000 requests/month
- Enterprise: Per negotiation

Other tiers should be considered

A.4.4.4 Considerations for income distribution

This licensing model is equivalent to what the competitors offer, with the exception that it is possible to use low volumes of the public data for no charge. This will make the service attractive for startups and non-commercial organisations that need the data but do not require large volumes.

Another issue is that the data should not be licensed per country. All data should be available through the same interface. This is to increase usability and to avoid a confusing and costly scheme where users are billed differently per country of use. The marketplace should instead keep track of the data usage for each subscription and divide the profit according to actual use.

For instance, Company Alpha has an enterprise data license, with a maximum number of requests set to 250k requests per month. 75 % of the made requests are for SpazioDati's data, while the remaining 25 % is for OpenCorporates. The system should automatically calculate the revenue for the partner companies based on the usage share.

This is not necessarily complicated: several other services, such as for instance Spotify, perform similar calculations.

A.4.5 Ownership

The marketplace could be set up in many possible ways. It could either be a joint stock company owned by the consortium partners, allowing them full control of the revenue, or by a single entity either within or outside the consortium that will own and govern the marketplace, paying license fees to the consortium partners for the use of their data.

A.4.6 Service level agreements

Business customers would most likely require some sort of service level agreements in connection to their subscription. Such an agreement would detail both the required degree of uptime of the service, procedures regarding downtime, warning procedures, as well as requirements regarding the data quality. Each data provider should make a statement regarding their commitment and procedures on updates, error handling, data quality, etc.

A.4.7 Administration and governance

This section discusses some of the issues that need to be addressed in the establishment of a data marketplace governed by the consortium.

A.4.7.1 Membership

Membership concerns should be part of the statutes.

- **Rules for admitting new members:** Questions such as which organisations should be eligible for future partnership need to be addressed.
- **Rules when a member desires to withdraw:** The procedure for leaving the consortium should be established. It should contain points such as criteria for withdrawing, transition period and cost-sharing.
- **Exclusion rules:** When and how a member can be excluded from the consortium or organisation, and the rights that the organisation retains. This should only be applicable in certain cases where a partner fails to meet its obligations.
- **Minimum level of participation:** A minimum level of participation should be agreed upon. This includes items like the level of data sharing if applicable, the responsibility for costs and deficits and participations in the organisations.
- **Decision making process:** In the following steps we will analyse questions related to:
    - How decisions will be reached.
    - Which issues will need consensus and which issues will need a simple or absolute majority.

A.4.7.2 Budgets

The rules for budgeting and reporting must be well defined and established.

- **Operations:** An operating budget should be decided upon.
- **Marketing:** A budget for marketing, if necessary, should be set aside.
- **Infrastructure and investments:** Depending on the operating model, a budget for infrastructure and other investments might be necessary.

Appendix B Requirements matrix

Description		OCORP	CERVED	SDATI	EVRY	DW	BRC	JSI
Stakeholder role (DP=Data Provider, DC=Data Consumer)		DP	DP, DC	DP, DC	DC	DP, DC	DP	DP
Data preparation services								
Dataset Import	Support for file formats (CSV, JSON, XML, RDF) and REST service	JSON, REST APIs	CSV, JSON, REST APIs	CSV, REST API	-	JSON, REST API	CSV, JSON, XML, RDF	?
Data Cleaning & Transformation	Support for data cleaning & transformation	✓	✓	✓	-	✓	✓	?
RDF-ization	Mapping to RDF	✓	✓	✓	-	✓	✓	?
Data Interlinking Services								
Named entity linking	Support for named entity linking	?	?	?	-	✓	?	-
Semantic labelling	Support for semantic labelling	✓	✓	✓	-	✓	?	-
Link discovery	Support for link discovery	✓	✓	✓	-	✓	?	-
Data Hosting Services								
Data queries	Support for data queries	✓	✓	✓	✓	✓	✓	✓
Company data search/discovery	Search for (type) of company data available	✓	✓	✓	✓	extended company profile	✓	✓
Faceted search	Faceted and filters	-	✓	✓	✓	brands	-	-
Data access	Support for data access	✓	✓	✓	✓	✓	✓	✓
SPARQL endpoint	Data can be queried through a SPARQL endpoint	-	-	-	-	-	✓	-
Data dump	Data dump available for download/transfer	✓	✓	✓	✓	?	✓	✓
REST service	Data is available through RESTful web services	✓	✓	✓	✓	✓	✓	✓
Data updates	Support for data updates	✓	✓	✓	-	✓	✓	✓
Frequency	Frequency of updates (Y=Yearly, M=Monthly, W=Weekly, D=Daily, V=Variable, C=Continuously)	D, V	D	W	-	D	C	M, Y
Incremental update	API for incremental update	✓	✓	✓	-	✓	✓	✓
Bulk update	API for bulk update	✓	✓	✓	-	-	-	-
Metadata	Support for metadata	✓	✓	✓	-	✓	✓	✓

Spatial coverage	Jurisdictions (e.g. countries) covered by the dataset (UK=United Kingdom, IT=Italy, N=Norway, R=Russia)	UK	IT	UK, IT, R	-	European, Global	N	✓
Data store (RDF)	Support for big data store	✓	✓	✓	-	✓	✓	✓
Dataset size	Size of dataset	1.3 billion records (1TB)	GBs per month	GBs per month	-	35.000 articles per year per language	1 million entities	?
Cross-Cutting Business Case Analytics Services								
Multi-lingual annotation	Support for multi-lingual annotation	✓	✓	✓	✓	✓	✓	✓
Language	Language of the dataset[34] (en=English, it=Italian, no=Norwegian, ru=Russian)	en	it	en	✓	articles in 30 different languages	no	✓
Events detection	Support for events detection	?	?	?	-	✓	?	-
Graph analytics	Support for graph based analysis	?	?	?	-	✓	?	-
Relation Extraction	Support for extracting relations between entities	?	?	?	-	✓	?	-
Marketplace and Operational Services								
License Models	Dataset license[35] (F=Free, P=Payment, D=Dual (payment plus share-alike for public-benefit use))	D	P	P	P	P	F	F
Open data	Data available as open data	✓	-	-	-	-	NLOD	-
Dataset-level access models	Access models to datasets (P=Public, M=Matching only, R=Restricted to license model)	P, R	M, R	M, R	-	R	P	-
Data property-level access model	Access models to data properties (P=Public, M=Matching only, R=Restricted)	P, R	M, R	M, R	-	R	P	-

[34] http://data.okfn.org/data/core/language-codes
[35] https://en.wikipedia.org/wiki/Creative_Commons_license

Security & Access Control	Support for authentication and user management	✓	✓	✓	✓	✓	✓	✓
System Monitoring & Reporting	Support for monitoring and reporting	✓	✓	✓	✓	✓	✓	✓
Platform Administration	Support for platform administration	✓	✓	✓	✓	✓	✓	✓