

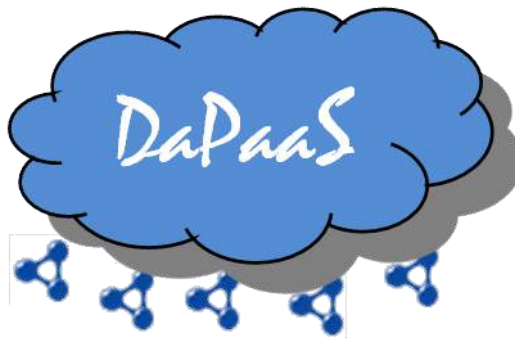
Small or medium-scale focused research project (STREP)

ICT SME-DCA Call 2013

FP7-ICT-2013-SME-DCA

**Data Publishing through the Cloud:
A Data- and Platform-as-a-Service Approach to Efficient
Open Data Publication and Consumption**

DaPaaS



Deliverable 1.1

**Open DaaS requirements, design &
architecture specification**

Date:	31 January 2014
Author(s):	Marin Dimitrov, Alex Simov, Petar Kostov, Dumitru Roman, Brian Elvesæter, Arne Berre, Rick Moynihan
Dissemination level:	PU
WP:	1
Version:	1.0

Document metadata

Quality assurors and contributors

Quality assessor(s)	Bill Roberts, Amanda Smith
Contributor(s)	DaPaaS Consortium

Version history

Version	Date	Description
0.1	10 January 2014	Outline.
0.2	16 January 2014	Initial architecture for Data Layer and description of components.
0.3	21 January 2014	Completed state of the art analysis for DaaS.
0.4	24 January 2014	First integrated version.
0.5	28 January 2014	Draft version ready for internal review.
0.6	29 January 2014	Addressed review comments.
0.7	30 January 2014	Final round of review.
1.0	30 January 2014	Final formatting and layout.

Executive Summary

The main goal of the DaPaaS project is to provide an integrated Data-as-a-Service (DaaS) and Platform-as-a-Service (PaaS) environment, together with associated services, for open data, where 3rd parties can publish and host both datasets and data-driven applications that are accessed by end-user data consumers in a cross-platform manner.

This deliverable focuses on the DaaS aspect of the DaPaaS Platform and provides four important outcomes related to the data management aspect of the DaPaaS platform:

- 1) A summary of the business requirements which the data management layer should support
- 2) A state-of-the-art overview of relevant solutions, technologies and standards;
- 3) An initial architecture design for a scalable data management and query layer; and
- 4) An overview of 3rd party tools which will be integrated for the data layer prototype implementation.

This deliverable is aligned and should be read in conjunction with the corresponding deliverables in WP2 (D2.1) and WP3 (D3.1). The outlined data layer design will be implemented at M12 (D1.2) and further refined at M21 (D1.3) based on user feedback.

Table of Contents

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
LIST OF ACRONYMS	6
LIST OF FIGURES	7
LIST OF TABLES	8
1 INTRODUCTION	9
2 REQUIREMENTS ANALYSIS	10
2.1 KEY ROLES IN DAPAAS.....	10
2.2 REQUIREMENTS FOR THE DATA LAYER	11
2.2.1 Instance Operator	11
2.2.2 Data Publisher	11
2.2.3 Application Developer	12
2.2.4 End-user Data Consumer	12
3 STATE OF THE ART OVERVIEW	14
3.1 DATA ACCESS	14
3.2 EXISTING DATA-AS-A-SERVICE SOLUTIONS	15
3.2.1 Azure	16
3.2.2 Factual.....	17
3.2.3 Socrata.....	17
3.2.4 DataMarket.....	18
3.2.5 Junar	20
3.2.6 PublishMyData.....	20
3.2.7 LOD2	20
3.2.8 European Open Data Portal.....	21
3.2.9 Project Open Data	22
3.2.10 COMSODE	22
3.3 RELEVANT TECHNOLOGIES & STANDARDS	23
3.3.1 5 Star Open Data	23
3.3.2 CKAN	23
3.3.3 DCAT.....	23
3.3.4 VoID.....	24
3.3.5 CSV on the Web (W3C).....	24
3.3.6 JSON-LD	24
3.3.7 OData	24
3.4 COMPARISON OF DAPAAS TO OTHER SOLUTIONS	25
4 DATA LAYER ARCHITECTURE	27
4.1 DATA MODEL	27
4.1.1 Key/Value Data.....	27
4.1.2 Tabular Data.....	27
4.1.3 RDF & Linked Data.....	27
4.2 TECHNICAL CAPABILITIES	27
4.2.1 RDF Storage & Access	28
4.2.2 Non-RDF Storage & Access	28
4.2.3 Full-text Search	28
4.2.4 Data Catalogue	28
4.2.5 Querying.....	28
4.2.6 Interlinking.....	28
4.2.7 Caching	29
4.2.8 Import / Export.....	29
4.2.9 Notifications & Statistics	29
4.3 NON-FUNCTIONAL ASPECTS	29
4.3.1 Scalability.....	29

4.3.2	User Management & Access Control	29
4.3.3	Availability.....	30
4.3.4	Monitoring.....	30
4.4	DATA LAYER ARCHITECTURE & COMPONENTS.....	30
4.5	SUMMARY OF ADDRESSED REQUIREMENTS	32
5	RELEVANT TOOLS	34
5.1	RDF DATABASES	34
5.1.1	OWLIM.....	34
5.1.2	Sesame	35
5.1.3	Apache Jena Framework	36
5.2	NOSQL DATABASES	36
5.2.1	Cassandra	37
5.2.2	HBase.....	37
5.2.3	CouchBase.....	38
5.2.4	Solr	38
5.3	RDB2RDF TOOLS.....	39
5.3.1	db2triples.....	39
5.4	PDF IMPORT TOOLS	39
5.4.1	Tabula.....	39
5.5	CSV IMPORT TOOLS.....	39
5.5.1	CSV-to-API.....	39
5.5.2	CSV.js	40
5.5.3	Mr. Data Converter	40
5.6	SILK FRAMEWORK.....	40
6	RECOMMENDATIONS	40
6.1	COMPONENTS	41
6.2	DEPLOYMENT & PROVISIONING	42
6.2.1	Hardware Requirements	42
6.2.2	Software Requirements	42
	BIBLIOGRAPHY	43

List of Acronyms

API	Application Programming Interface
CSV	Comma Separated Values (format)
CQL	Cassandra Query Language
DaaS	Data-as-a-Service
DCAT	Data Catalog Vocabulary
FOAF	Friend of a Friend (vocabulary)
JSON	JavaScript Object Notation (format)
PaaS	Platform-as-a-Service
R2RML	Relational to RDF Mapping Language
RDF	Resource Description Framework
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
SSH	Secure Shell
VOID	Vocabulary of Interlinked Datasets
XML	eXtensible Markup Language

List of Figures

Figure 1: High-level architecture of the DaPaaS Platform	9
Figure 2: Key roles/actors in DaPaaS	10
Figure 3: Data exchange in JSON.....	14
Figure 4: RDF data exchange in JSON.....	15
Figure 5: Azure DataMarket architecture, (c) Microsoft.....	16
Figure 6: Socrata Open Data Server architecture, (c) Socrata	18
Figure 7: DataMarket options for data consumers	19
Figure 8: DataMarket options for data publishers	19
Figure 9: PublishMyData SLA guarantees	20
Figure 10: LOD2 lifecycle for Linked Data Management, (c) LOD2 project.....	21
Figure 11: EU Open Data Portal - applications catalogue. (c) EU Open Data Portal	22
Figure 12: OData services, (c) David Chappell.....	25
Figure 13: Data Layer Architecture.....	31
Figure 14: Sesame Architecture	35

List of Tables

Table 1: Description of requirements from the Instance Operator (IO)	11
Table 2: Description of requirements from the Data Publisher (DP)	11
Table 3: Description of requirements from the Application Developer (AD)	12
Table 4: Description of requirements from the End-Users Data Consumer (EU).....	12
Table 5: DaPaaS differentiation (DaaS aspect).....	25
Table 6: Addressed requirements by components of the Data Layer.....	32
Table 7: Initial hardware requirements	42
Table 8: Initial software requirements	42

1 Introduction

This report represents Deliverable D1.1 "Open DaaS requirements, design & architecture specification" of the DaPaaS project. This deliverable is a result of Task T1.1 "Requirements, analysis & design of the Open Data-as-a-Service infrastructure".

The goals of this deliverable are to provide:

- An overview of requirements related to data management & querying of the DaPaaS platform;
- A state-of-the-art analysis of related Data-as-a-Service solutions (not necessarily focussing on Open Data);
- An initial architecture for the Data Layer prototype to be implemented by M12 of the project;
- Concrete recommendations on how the Data Layer should be implemented by integrating and adapting existing 3rd party tools, as well as with custom new development where necessary.

In line with the overall DaPaaS Platform architecture introduced in deliverable D2.1, the DaPaaS platform (Figure 1) is roughly divided into three layers, covering aspects related to data management, application management and UX (including data-driven portals and mobile access). Additionally, cross-aspects related to scalability, performance, access control and quota enforcement will have an impact on all layers of the platform.

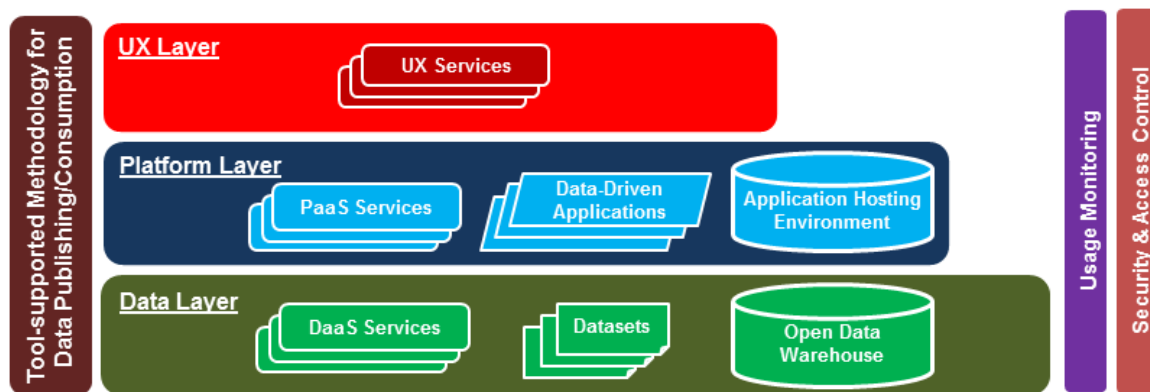


Figure 1: High-level architecture of the DaPaaS Platform

This deliverable focuses on the Data Layer part of the platform, and is organised as follows:

- Section 2 "Requirements Analysis" provides a summary of the key roles identified in Deliverable 2.1 (DaPaaS Project, 2014) and their respective business requirements for the DaPaaS platform. Only the requirements which are relevant to the Data Layer are taken into consideration.
- Section 3 "State of the Art Overview" provides an analysis of existing solutions relevant to the DaPaaS Data Layer, relevant technologies and standards, and highlights the difference of the DaPaaS Data Layer when compared to existing solutions.
- Section 4 "Data Layer Architecture" provides a technical analysis of the data model to be adopted for data hosting, the concrete functional and non-functional technical capabilities of the data layer, as well as an initial architecture for data management in DaPaaS.
- Section 5 "Relevant Tools" provides an analysis of various 3rd party tools which are deemed relevant to support the outlined technical architecture.
- Finally, Section 6 "Recommendations" provides concrete recommendations for the implementation of the 1st Data Layer prototype due in M12.

2 Requirements Analysis

The DaPaaS Deliverable D2.1 (DaPaaS Project, 2014) provides the complete list of business requirements for all technical Work Packages (WP1-4) and the descriptions of the key roles (actors) that will interact with the DaPaaS platform.

For the sake of completeness, this section summarises the key roles and their respective requirements which are relevant to WP1 (DaaS) and the focus of this deliverable respectively. For further details on the roles, the overall requirements and architecture, the reader is referred to Deliverable D2.1.

2.1 Key Roles in DaPaaS

The key roles involved in a typical DaPaaS context and their relationships within the platform, are illustrated in Figure 2. The roles are:

- The **DaPaaS Developer** is responsible for implementing the DaPaaS software components and services for the integrated DaaS and PaaS environment. During the course of the project, this role is expected to be exclusively played by the DaPaaS consortium (although certain 3rd party software components will be adopted and integrated into the DaPaaS platform).
- A deployed instance of DaPaaS software, i.e. the DaPaaS Platform, is operated and maintained by an **Instance Operator**. During the course of the project, this role is played again by the DaPaaS consortium, however the DaPaaS software will also be deployable by 3rd party **Instance Operators**.
- The **Data Publisher** has the goal of publishing data on the DaPaaS Platform so that it is available to 3rd party application developers and end user data consumers.
- The **Application Developer** develops data-driven applications that utilise the data hosted on the DaPaaS Platform. The applications are deployed and hosted in the DaPaaS Platform.
- Finally, **End-Users Data Consumers** indirectly utilise platform hosted data through bespoke applications deployed on either the web or native clients such as desktop and mobile apps.

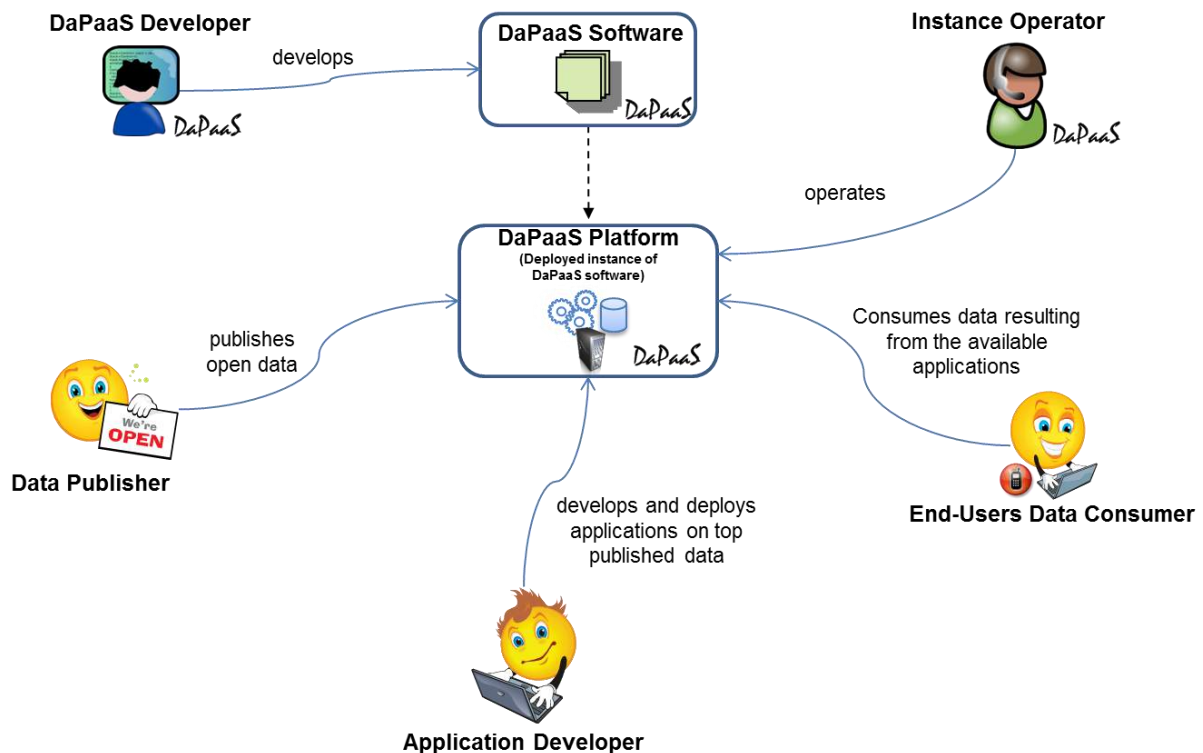


Figure 2: Key roles/actors in DaPaaS

2.2 Requirements for the Data Layer

This section provides the **subset** of the business requirements from the five business roles that are relevant to the Data Layer (WP1). The complete list of all business requirements is available in Deliverable D2.1 (DaPaaS Project, 2014).

2.2.1 Instance Operator

Table 1: Description of requirements from the Instance Operator (IO)

ID	Name	Brief description
IO-02	Platform performance monitoring	The Instance Operator shall be able to monitor the performance (e.g. storage and memory usage, bandwidth, CPU usage, etc.).
IO-03	Statistics monitoring (users, data, apps, usage)	The Instance Operator shall be able to retrieve statistics about users (e.g. number, profiles), data (e.g. number, size), apps and usage (e.g. dataset access, data consumption, number of service calls) as a basis for e.g. billing/invoicing for the usage of the platform.
IO-05	Policy/quota configuration and enforcement	The Instance Operator shall be able to configure usage policies, e.g. data/apps quotas per user. The platform shall ensure enforcement of these policies, e.g. support deployment of applications subject to quotas and additional restrictions.

Note that requirements IO-01, IO-04 and IO-06 fall outside the scope of the Data Layer (WP1) and they will be implemented within WP2.

2.2.2 Data Publisher

Table 2: Description of requirements from the Data Publisher (DP)

ID	Name	Brief description
DP-01	Dataset import	The Data Publisher should have the ability to import open data into the DaPaaS platform. The data is <i>not</i> restricted to RDF / Linked Data and it may include other formats such as CSV, JSON, etc. As part of the import process, automated data transformations may be applied (e.g. RDB2RDF direct mapping, CSV2RDF mappings, etc.).
DP-02	Data storage & querying	The Data Publisher should have access to APIs and query endpoints for accessing, querying and updating data stored on the platform.
DP-03	Dataset search & exploration	The Data Publisher should have the possibility to explore the dataset catalogue & select relevant datasets.

DP-04	Data interlinking	The Data Publisher should have the possibility to semi-automatically interlink data from different datasets. This applies only to data which is already in RDF form.
DP-06	Dataset bookmarking & notifications	The Data Publisher should have possibility to subscribe to datasets and receive notifications on datasets changes.
DP-07	Dataset metadata management, statistics & access policies	The Data Publisher should have possibility to specify metadata, descriptions and access control policies for the datasets.
DP-08	Data scalability	The platform should include mechanisms to scale to large data volumes.
DP-09	Data availability	The platform should include mechanisms to provide high availability of data and limited downtime.
DP-11	Secure access to platform	The Data Publisher shall have secure access (e.g. HTTPS/SSH) to the platform.

Note that some requirements fall outside the scope of WP1 and will be implemented in other Work Packages: DP-05, DP-10, DP-12 (WP2) and DP-13 (WP4).

2.2.3 Application Developer

Table 3: Description of requirements from the Application Developer (AD)

ID	Name	Brief description
AD-01	Access to Data Publisher services (DP-01 – DP-13)	The Application Developer shall have access to APIs and libraries to access, import, transform, store, query, etc., datasets to be used in the development of applications. Basically the Application Developer has similar requirements as outlined in DP-01 – DP-13. This includes also requirements for secure access to the platform, profile management.
AD-02	Data export	The Application Developer shall have the possibility to export data in various formats.

Note that most of the business requirements from the Application Developer point of view fall outside the scope of the Data Layer (WP1) and only AD-01 and AD-02 are the concern of this layer.

2.2.4 End-user Data Consumer

Table 4: Description of requirements from the End-Users Data Consumer (EU)

ID	Name	Brief description
EU-02	Search & explore datasets and applications	End-Users shall be able to search and explore datasets and applications available in the platform.

EU-03	Datasets and applications bookmarking and notifications	End-Users shall be able to bookmark and receive notifications (e.g. updates) of datasets and applications to which they subscribe.
EU-05	Data export and download	End-Users shall have the possibility to export data in various formats and download data from the platform.
EU-06	High availability of data and applications	High availability of data and apps

Note that requirements EU-01 and EU-04 fall outside the scope of the Data Layer (WP1).

3 State of the Art Overview

This section provides a summary of existing DaaS solutions as well as standards and technologies relevant to the data management aspect of the DaPaaS platform.

3.1 Data Access

While there are numerous legacy approaches for consuming data from remote data services, datastores or databases, such as ODBC¹, JDBC², RMI³, CORBA⁴, etc., modern Web architectures and data services rely extensively on lightweight, RESTful Web service based approaches exchanging data via standard protocols (HTTP) and formats (JSON or RDF).

A typical RESTful data service call may look like (example from the Socrata Open Data API⁵):

[HTTP://SODA.DEMO.SOCRATA.COM/RESOURCE/EARTHQUAKES.JSON?\\$LIMIT=3&\\$ORDER=DATETIME DESC](http://soda.demo.socrata.com/resource/earthquakes.json?$limit=3&$order=datetime_desc)

The results from the service call are returned to the caller application in JSON:

```
[ {
  "earthquake_id" : "12345"
}
, {
  "region" : "Socrata",
  "source" : "so",
  "location" : {
    "needs_recoding" : false,
    "longitude" : "-117.6135",
    "latitude" : "41.1085"
  },
  "magnitude" : "1.4",
  "number_of_stations" : "1",
  "datetime" : "2013-05-02T22:38:01",
  "earthquake_id" : "1234",
  "depth" : "7.60",
  "version" : "9"
}
, {
  "region" : "Utah",
  "source" : "nn",
  "location" : {
    "needs_recoding" : false,
    "longitude" : "-117.6135",
    "latitude" : "41.1085"
  },
  "magnitude" : "2.9",
  "number_of_stations" : "17",
  "datetime" : "2012-09-14T22:38:01",
  "earthquake_id" : "00388610",
  "depth" : "7.90",
  "version" : "9"
}
]
```

Figure 3: Data exchange in JSON

A more expressive way to query data services is provided by RDF databases, via Web based SPARQL endpoints which can answer arbitrary complex SPARQL queries and return data in one of the standard RDF serialisation forms.

¹ Open Data Base Connectivity

² Java Data Base Connectivity

³ Remote Method Invocation

⁴ Common Object Request Broker Architecture

⁵ <http://dev.socrata.com/consumers/getting-started>

Examples of such publicly available data service SPARQL endpoints include: FactForge ⁶, LinkedLifeData⁷, European Open Data Portal⁸, EuroStat⁹, World Bank¹⁰, etc.

An example of a remote SPARQL invocation may look like:

[HTTP://FACTFORGE.NET/SPARQL?QUERY=<SPARQL QUERY BODY>](http://factforge.net/sparql?query=<SPARQL QUERY BODY>)

The results from the service call are returned to the caller application in JSON (or other standard RDF serialisation format):

```
{
  "head": {
    "vars": [
      "airport",
      "label",
      "RR"
    ]
  },
  "results": {
    "bindings": [
      {
        "airport": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/London_Heathrow_Airport"
        },
        "label": {
          "type": "literal",
          "xml:lang": "en",
          "value": "London Heathrow Airport"
        },
        "RR": {
          "type": "typed-literal",
          "datatype": "http://www.w3.org/2001/XMLSchema#float",
          "value": "0.12"
        }
      },
      {
        "airport": {
          "type": "uri",
          "value": "http://dbpedia.org/resource/Gatwick_Airport"
        },
        "label": {
          "type": "literal",
          "xml:lang": "en",
          "value": "Gatwick Airport"
        },
        "RR": {
          "type": "typed-literal",
          "datatype": "http://www.w3.org/2001/XMLSchema#float",
          "value": "0.07"
        }
      }
    ]
  }
}
```

Figure 4: RDF data exchange in JSON

3.2 Existing Data-as-a-Service Solutions

This section provides a brief overview of various solutions providing general Data-as-a-Service functionality, or Open Data hosting / catalogues in particular.

First, we will introduce the definition of DaaS which we will adhere to throughout the document:

“Like all members of the “as a Service” (XaaS) family, DaaS is based on the concept that the product, data in this case, can be provided on demand to the user regardless of geographic or organizational separation of provider and consumer. Additionally, the emergence of service-oriented architecture (SOA) has rendered the actual platform on which the data resides also irrelevant”¹¹

We will analyse several DaaS / marketplace solutions (Azure, Factual, InfoChimps, DataMarket, Socrata) based on features such as:

- Operating party
- Data domain
- Means for populating new content
- Query languages

⁶ <http://factforge.net/>

⁷ <http://linkedlifedata.com/>

⁸ <https://open-data.europa.eu/en/linked-data>

⁹ <http://eurostat.linked-statistics.org/sparql>

¹⁰ <http://worldbank.270a.info/sparql>

¹¹ <https://en.wikipedia.org/wiki/DaaS>

- Data Management tools
- Data model
- Data size
- Data export means
- SLA availability

Additionally, we will provide an overview of solutions targeting Linked or Open Data in particular, such as Socrata, the LOD2 platform, the European Open Data Portal, PublishMyData and Project Open Data.

3.2.1 Azure

The Windows Azure platform provides a marketplace for applications and data. The data marketplace¹² (Microsoft Corp., 2011) currently provides access to 170 datasets¹³ in various categories: geo-spatial data, US Census data, maps, weather data, D&B corporate data, etc.

The monetization model is based on charging the data consumers (per API calls). The data marketplace itself relies on the various storage options of the Windows Azure platform and provides tools and APIs for data publishing, analytics, metadata management, account management and pricing, monitoring and billing, as well as a data portal for dataset exploration (Figure 5).

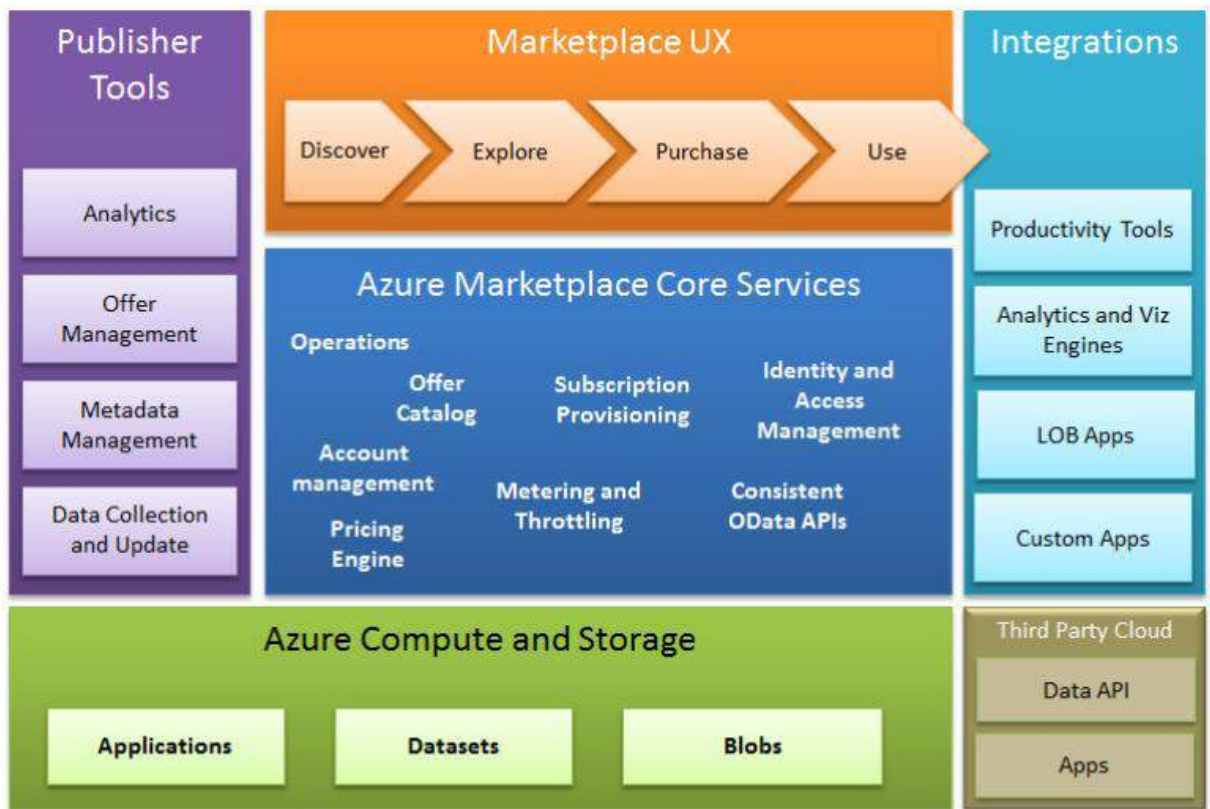


Figure 5: Azure DataMarket architecture, (c) Microsoft

The Azure Marketplace provides an option for data publishers to host and monetize 3rd party data via the data marketplace platform¹⁴.

¹² <https://datamarket.azure.com/>

¹³ <https://datamarket.azure.com/browse/data>

¹⁴ <http://msdn.microsoft.com/en-us/library/hh563871.aspx>

3.2.2 Factual

Factual¹⁵ provides high-quality data for more than 70 million local business and points of interest in 50 countries. Additionally, it provides a product database of over 650,000 products.

Factual used to provide the option for hosting thousands of 3rd party data sets (“Community Data”) but as of the time of writing this analysis, this activity has been discontinued.

Data is populated by means of Web crawls, data extraction and 3rd party data services. The data model is tabular, based on taxonomy of around 400 categories.

The pricing is based on a pay-per-use model, where data access (API calls) is monitored and billed. The free quota is set to 10,000 API calls per day for the Global Places database, and to 500 calls for the other datasets.

Data access is provided through a RESTful API. Simple key/value restrictions on attributes from the data model are possible through the API. Additionally, Factual provides a set of tools for data management, such as:

- Resolve API¹⁶ for entity mapping (reconciliation) and data de-duplication;
- GeoCoder API¹⁷ for converting geo-spatial coordinates into local street addresses and points of interest;
- World Geographies API¹⁸ for providing extended structured information about points of interest, including a translation in 19 languages.

3.2.3 Socrata

Socrata¹⁹ is a platform focusing on Open Data and services. The Socrata product portfolio includes:

- Open Data Portal²⁰ solution for organisations that need to publish Open Data. The ODP provides functionality for data publishing & clean-up, metadata generation, data-driven portals for data exploration and portal management;
- A hosted portal²¹ for Open Data;
- API Foundry²² for creating and deploying RESTful APIs on top of the data

The data hosted on the Socrata platform is accessible through the Socrata Open Data API (SODA), which provides RESTful interface for searching and reading data in XML, JSON or RDF.

Additionally, Socrata provides an open source Socrata Open Data Server²³, which includes the core components of the Open Data Portal product (Figure 6).

¹⁵ <http://www.factual.com/>

¹⁶ <http://www.factual.com/products/resolve>

¹⁷ <http://www.factual.com/products/geocode>

¹⁸ <http://www.factual.com/products/world-geographies>

¹⁹ <http://www.socrata.com/>

²⁰ <http://www.socrata.com/open-data-portal/>

²¹ <https://opendata.socrata.com/>

²² <http://www.socrata.com/api-foundry/>

²³ <http://open-source.socrata.com/>

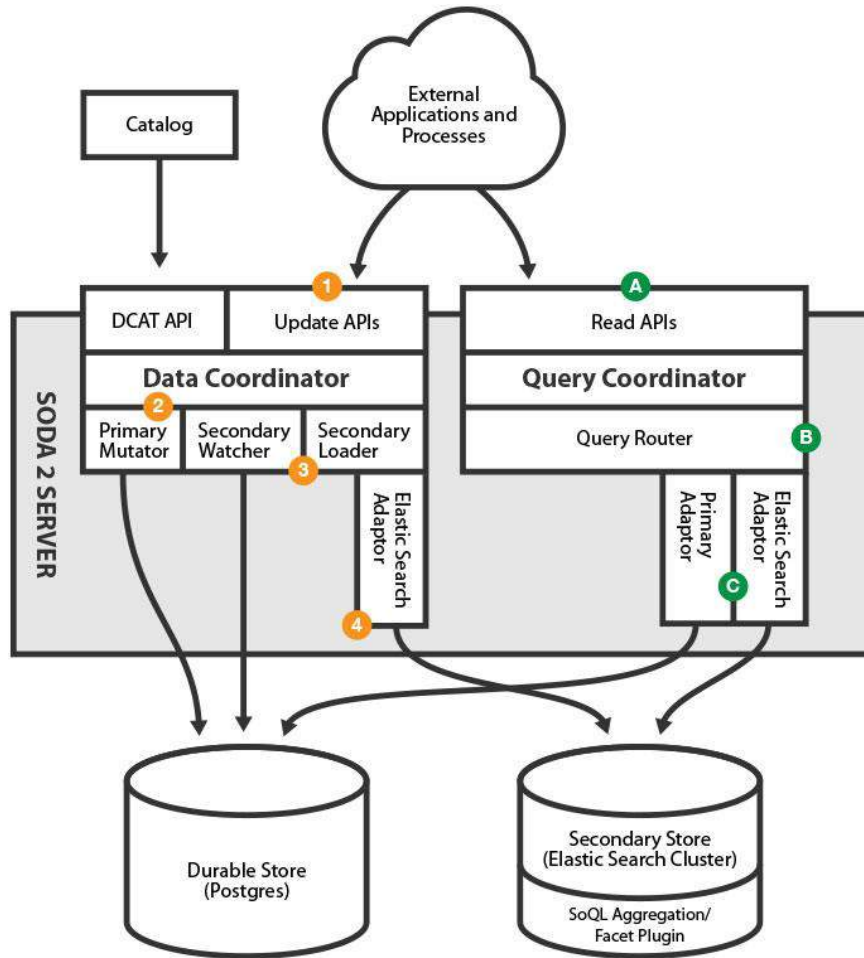


Figure 6: Socrata Open Data Server architecture, (c) Socrata

3.2.4 DataMarket

DataMarket²⁴ provides statistical data from almost 100 data providers²⁵, including UN, Eurostat, World Bank and IMF. Currently there are more than 70,000 datasets providing 350 million time series and 3 billion facts on the marketplace.

The features of the DataMarket offering include:

- Embeddable visualisations of the data
- Data export
- Live feeds for data updates
- Ability for data publishers to monetize data via the marketplace
- Custom data driven portals for publishers
- Open Data portal²⁶

Various features are available to data consumers and data providers for free, or for price in a pay-per-use manner (Figure 7 & Figure 8).

²⁴ <http://datamarket.com/>

²⁵ <http://datamarket.com/data/>

²⁶ <https://datamarket.com/topic/list/countries/>

For data consumers

	Free	Data Hub
Full access to data from 100+ sources	✓	✓
Powerful search	✓	✓
Charting and data visualization	✓	✓
Comparison of data sets	✓	✓
Data download (Excel, CSV)	✓	✓
Export charts (PPT, PNG, PDF, SVG)	✓	✓
Live data feeds (R, Excel)	✓	✓
Embedding interactive charts	✓	✓
Topic pages	✓	✓
Document search (PDF, PPT, DOC)		✓
Custom embeds		✓
Customized home page		✓
Group and user management		✓
Group sharing		✓
Fine-grained access controls		✓
Integration of proprietary/custom data sources		✓
Support	Email	Email & 24h hotline
Price	FREE	Contact us

Figure 7: DataMarket options for data consumers

For data publishers

	Pro	Data Delivery Engine
Publish data	✓	✓
Sell data	✓	✓
Embed interactive charts	✓	✓
Your own publisher page	✓	✓
Automated data updates		✓
Custom embeds		✓
Dashboards		✓
Branded profile		✓
Advanced user management		✓
Running on 3rd party domain		✓
Integration with your web site		✓
Single sign-on integration		✓
Full site branding		✓
Support	Email	Email & 24h hotline
Monthly	\$59	Contact us
Revenue sharing	You get 75%	You get 100%

Figure 8: DataMarket options for data publishers

The data on the marketplace is accessible through a RESTful API in JSON format.

3.2.5 Junar

Junar²⁷ is a cloud-based Open Data platform which enables government organisations and SMEs to collect, enrich, publish and analyse open data. Data can be consumed either directly via the Junar API, or via various visual widgets.

The Junar Website provides no details on how the data collection, enrichment, publishing or analysis tasks are organised, so at this point it's not possible to assess neither the technical, nor the business potential of the platform, but DaPaaS will continue to monitor Junar's progress.

3.2.6 PublishMyData

PublishMyData²⁸ is Swirrl's LinkedData publishing platform. It provides a fully hosted, as-a-service solution for organisations that need to publish Open and Linked Data. PublishMyData aims to help publishers provide their data in ways that suit a wide variety of audiences, from lay web users to more specialist audiences such as spreadsheet users (via CSV), and software developers.

Hosted datasets are catalogued online with DCAT and are arranged into a human browsable web-based catalogue. Datasets are additionally accessible in programmatic and machine readable ways such as via RESTful APIs, a SPARQL endpoint and raw data-dumps.

Similar to Socrata, Swirrl also provides the core PublishMyData platform as an open source product²⁹.

The current list of customers for PublishMyData includes various government agencies in the UK. The pricing³⁰ for data publishers is set up as follows:

- One time setup fee
- Monthly subscription fee
- A traffic charge per one million API requests
- Monthly data storage charge per one million RDF triples

PublishMyData does **not** charge data consumers in any way (only publishers), and from the point of view of the consumers, the data hosted on the platform is completely free for access.

PublishMyData provides a SLA of at least 99.5% for its platform (Figure 9)

Availability	Refund
> 99.5%	N/A
99.0 - 99.5%	5%
95.0 - 99.0%	20%
< 95%	50%

Figure 9: PublishMyData SLA guarantees

3.2.7 LOD2

The LOD2³¹ project (LOD2 Project, 2012) aims at providing an open source, integrated software stack for managing the complete lifecycle of Linked Data, from data extraction, enrichment, interlinking, to maintenance (Figure 10).

²⁷ <http://www.junar.com/>

²⁸ <http://www.swirrl.com/publishmydata>

²⁹ https://github.com/swirrl/publish_my_data

³⁰ <http://www.swirrl.com/publishmydata#pricing>

³¹ <http://lod2.eu/>

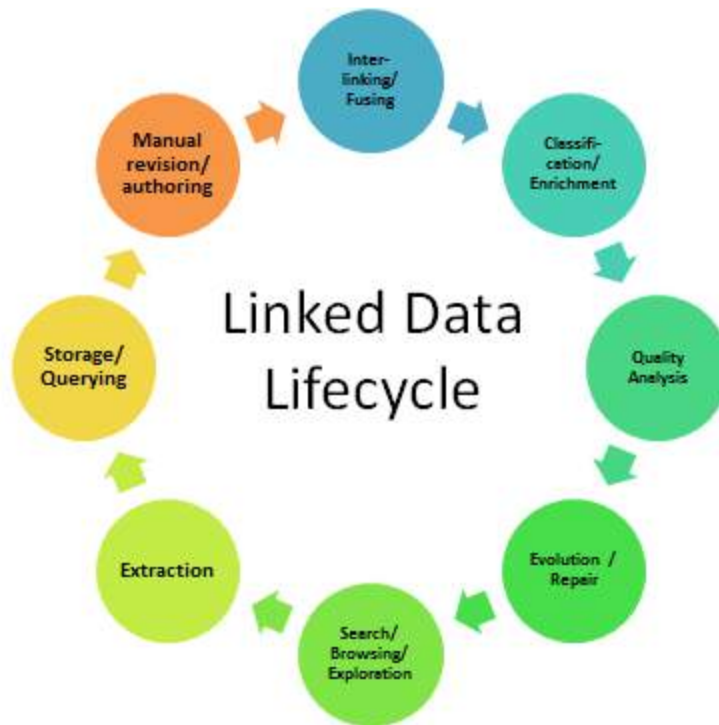


Figure 10: LOD2 lifecycle for Linked Data Management, (c) LOD2 project

The LOD2 stack comprises of:

- *Integrated set of software packages* which can be deployed via the Debian packaging system. At present, the LOD2 stack includes various open source components such as: Apache Stanbol (NLP Middleware Server), the CubeViz statistical data browser, DBpedia Spotlight (Entity Recognition and Linking), D2RQ server for RDB2RDF Mapping, DL-Learner (Machine Learning in OWL), OntoWiki (data wiki), ORE (Knowledge Base Debugging), PoolParty taxonomy editor, the SemMap spatial data browser, Sig.ma data browser, Sieve (Quality Assessment and Fusion), Silk & Limes data interlinking frameworks, Virtuoso RDF database, and Valiant for XML-to-RDF transformation.
- *A centralised knowledge base* (SPARQL endpoint) for integration between the various components in the stack.
- *A set of RESTful services and web applications* deployed on top of the software stack components and services.

3.2.8 European Open Data Portal

The EU Open Data Portal³² provides a metadata catalogue for Open Data hosted by various European government agencies. The catalogue metadata is available as Linked Data via a standard SPARQL endpoint³³. Additionally, the EU Open Data Portal provides a catalogue for various applications utilising Open Data (Figure 11)

³² <https://open-data.europa.eu/en/data/>

³³ <https://open-data.europa.eu/en/linked-data>

Visualisation applications

The datasets you can access via the EU Open Data Portal contain raw data. Since data in this format can be difficult to read, applications are used to visualise and to reuse them. Discover the potential of EU open data with the applications created by data providers: see the gallery below.


Share your app

Have you used our data to create an application?

[Please let us know!](#)

The Digital Agenda Scoreboard

Explore graphs




DG CONNECT

The Digital Agenda Scoreboard web application allows you to interactively create graphs to view the data you are interested in. About a hundred indicators have been selected and then divided into thematic groups, illustrating some key dimensions of the European information society. You can use these indicators to assess progress with respect to the targets set out in the Digital Agenda, to compare progress across countries over time, or to analyse the profile of a particular country.

[Go to application...](#)

The European Forest Data Centre (EFDAC) Map Viewer



JOINT RESEARCH CENTRE (JRC)

The EFDAC Map Viewer is a customised web map client that allows users to visualise, navigate and query maps and derived geo-datasets available in EFDAC. This includes the JRC forest cover maps, species distribution and suitability maps, information on forest patterns, forest conditions and forest resource information. (e.g. forest area and growing stock).

[Go to application...](#)

Figure 11: EU Open Data Portal - applications catalogue. (c) EU Open Data Portal

3.2.9 Project Open Data

While technically not a Data-as-a-Service solution, Project Open Data³⁴ is still highly relevant to the Open Data domain, since it provides a set of open source tools, methodologies and use cases for successfully publishing and utilising Open Data. The methodologies and recommendations provide concrete step-by-step guide to government agencies on how to publish and catalogue open data in machine readable formats³⁵, and what are the recommended licenses³⁶ for open data.

Some of the tools of interest, part of the software stack include:

- Database to API (dynamically generate RESTful APIs from a database);
- CSV to API (Dynamically generate RESTful APIs from static CSVs);
- Spatial Search;
- Catalogue Generator for automated data catalogue generation;
- JSON validators, PDF filters, and various data conversion tools ;
- DKAN – a Drupal based open data portal compliant with CKAN.

3.2.10 COMSODE

The COMSODE (Components Supporting the Open Data Exploitation) project³⁷ is an SME-driven research project funded under the same call as DaPaaS. It aims to develop a methodology for data publication for data owners (e.g. public bodies) and re-users (applications of SMEs, NGOs, public bodies, etc.), and provide an open source implementation of a data publication platform (software

³⁴ <http://project-open-data.github.io/>

³⁵ <http://project-open-data.github.io/catalog/>

³⁶ <http://project-open-data.github.io/open-licenses/>

³⁷ <http://www.comsode.eu/>

Copyright © DaPaaS Consortium 2013-2015

Page 22 / 43

components and tools). Whereas similar in goals to DaPaaS, the key differentiation is that DaPaaS targets a hosted platform for data publication and application development of data intensive applications. Both COMSODE and DaPaaS are in early stages at the time of this writing, and initial contact between the two projects has been established to investigate the possibility of cooperation in the near future.

3.3 Relevant Technologies & Standards

This section provides an overview of various standards and technologies relevant to the Data Layer of DaPaaS.

3.3.1 5 Star Open Data

While technically not an official standard, the 5 Star Open Data ranking (EPSI, 2010) is an approach promoted initially by Sir Tim Berners Lee, that aims to provide a “Maslow pyramid” for open data, comprised of the following levels:

- 1 star – data is available on the web under some Open License
- 2 stars – data is available as structured data in a machine readable format
- 3 stars – data is accessible under a non-proprietary data format (CSV, JSON, RDF, ...)
- 4 stars – URIs are used to identify data fragments, so that fine-grained access to the data is possible
- 5 stars – data is inter-linked with other data on the Web

3.3.2 CKAN

CKAN is an open source software data catalogues, managed by the Open Knowledge Foundation. It is currently the most widely used data catalogue software for open data and particularly popular with public sector organisations across the world. A list of deployed instances is available at <http://ckan.org/instances> and includes the government data catalogues of EU, US, UK and many other countries and cities around Europe and worldwide.

The most popular use of CKAN is as a straightforward catalogue, with links to downloadable files (whether managed directly by CKAN or elsewhere). Metadata is added by data owners through web forms. There is also an API for creating or editing records and for accessing metadata in machine readable forms.

It can be integrated with Web content management systems such as Drupal and Wordpress, to allow customisation of websites and incorporation of other typical CMS features. CKAN also incorporates community building features, such as commenting, social media links, ability to 'follow' a dataset etc.

For structured data (primarily spreadsheets and CSV files), CKAN can also store the data and offer an API to access it³⁸, as well as a user interface. This works by importing each tabular data source into its own table in a PostgreSQL relational database. API requests make use of the table identifier and column names based on the contents of the original file. The API allows arbitrary SQL queries to be executed against the store.

3.3.3 DCAT

DCAT³⁹ stands for the Data Catalog Vocabulary, an RDF vocabulary designed to promote interoperability between data catalogues on the Web. It has recently reached 'recommendation' status in the W3C standardisation process.

It incorporates standards for describing the licence, methods of access, topic, coverage, creation and update dates for datasets in a catalogue, making use of other existing vocabularies where appropriate, notably the Dublin Core metadata standard. DCAT is applicable to all kinds of dataset formats.

³⁸ <http://docs.ckan.org/en/latest/datastore.html#the-datastore-api>

³⁹ <http://www.w3.org/TR/2014/REC-vocab-dcat-20140116/>

The widely used CKAN data catalogue software makes use of DCAT to provide RDF metadata for catalogue entries. PublishMyData also makes use of DCAT for describing datasets.

3.3.4 VOID

VOID⁴⁰ stands for 'Vocabulary of Interlinked Datasets'. It is a W3C 'Interest Group Note' as part of the work of the Semantic Web Interest Group. It is not expected to become a W3C recommendation.

VOID is specifically intended for describing RDF datasets (unlike DCAT which applies to any kind of datasets). Like DCAT, VOID incorporates Dublin Core metadata for describing the general features of a dataset, as well as the Friend of a Friend vocabulary (FOAF).

It allows the dataset owner to describe how to access the data, via SPARQL endpoint or by data download. It also enables descriptions of the content of the dataset, in terms of structure and size, and in how the dataset is connected to other datasets. This information may be useful to humans or to software using the data.

VOID can be used in combination with DCAT to give a richer description of an RDF dataset than is possible with DCAT alone.

3.3.5 CSV on the Web (W3C)

The mission of the *CSV on the Web Working Group*⁴¹, part of the Data Activity, is to provide technologies whereby data dependent applications on the Web can provide higher interoperability when working with datasets using the CSV (Comma-Separated Values) or similar formats. As well as single CSV files, the group will define mechanisms for interpreting a set of CSVs as relational data. This will include the definition of a vocabulary for describing tables expressed as CSV and locatable on the Web, and the relationships between them. In this way, it will be possible to see CSVs as 5-star data (data that is interlinked to other data) using the Web as an intelligent data platform rather than as a simple distribution system for files containing inaccessible data.

3.3.6 JSON-LD

JSON-LD (W3C, 2014) is an official W3C Recommendation (as of January 2014) which defines a JSON-based format for serialising and exchanging Linked Data.

The added value of JSON-LD is that it provides a relatively simple way for web developers to benefit from RDF and Linked Data concepts (for example interlinking and global identifiers), while keeping backward compatibility with the JSON format and existing web frameworks.

3.3.7 OData

The OData⁴² specification (Microsoft Corp., 2013) provides an easy way for data services to expose operations for managing and querying data resources as RESTful APIs. OData (Figure 12) provides recommendations for resource identifiers (URIs), HTTP based interface definition over the data service or application, an abstract data model for describing the exposed data, as well as recommendations for the data exchange format (JSON).

⁴⁰ <http://www.w3.org/TR/void/>

⁴¹ http://www.w3.org/2013/csvw/wiki/Main_Page

⁴² <http://www.odata.org/>

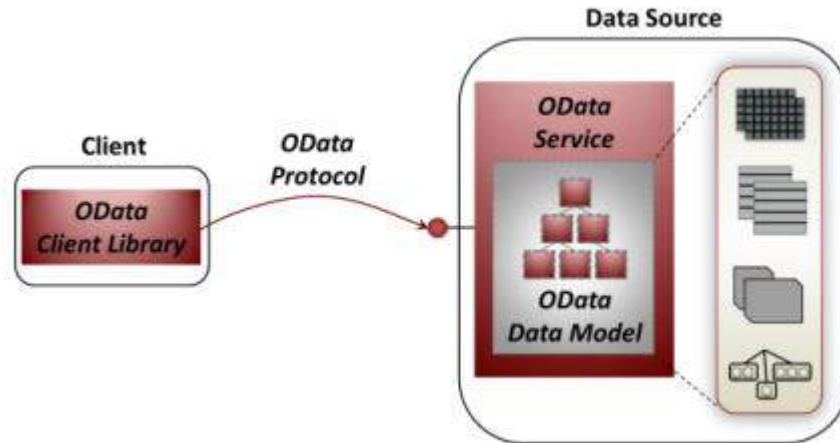


Figure 12: OData services, (c) David Chappell

3.4 Comparison of DaPaaS to Other Solutions

Based on the business requirements overview in section “Requirements Analysis” and the state-of-the-art analysis in section “Existing Data-as-a-Service Solutions”, Table 5 provides a summary of the similarities and key differentiations of DaPaaS and other solutions, as far as the DaaS aspect is concerned.

Table 5: DaPaaS differentiation (DaaS aspect)

Solution	Key similarities	Key DaPaaS differentiation
Azure Data Marketplace	Similar to DaPaaS, Azure aims at providing a fully hosted, as-a-service solution for data and applications	<ul style="list-style-type: none"> • Focus on Open Data • Focus on Linked Data and providing richer ways to interlink and query data
Factual	Hosted data service for tabular data	<ul style="list-style-type: none"> • Factual is focussed only on geo-spatial and product data • Focus on Open Data from different domains • Linked Data and providing richer ways to query data • Interlinking and mapping between datasets
Socrata	DaaS solution for open data	<ul style="list-style-type: none"> • Focus on Linked Data and SPARQL endpoints for complex data queries • Richer ways to interlink and align data from different datasets
DataMarket	As-a-service data provider, data driven portals	<ul style="list-style-type: none"> • Ability for 3rd parties to host data on the platform • Focus on Linked Data and SPARQL endpoints for complex data queries • Richer ways to interlink and align data from different datasets

Junar	Not really possible to assess the technical capabilities of the Junar data layer (no information available on the website)	<ul style="list-style-type: none"> • Most probably Linked Data management and semantic interlinking is out of scope for Junar
PublishMyData	PMD has a subset of DaPaaS functionality Including: Multi-format linked data publishing, API support, dataset catalogue etc	<ul style="list-style-type: none"> • PublishMyData is a DaPaaS component as Swirrl is a partner in the project. • Interlinking & other platform services • Application hosting
LOD2	Software stack for Linked Data management, no particular focus on Open Data, not a hosted solution	<ul style="list-style-type: none"> • As-a-service hosted solution • Ability for 3rd parties to host data on the platform • Handle Linked as well as non-RDF data
EU Open Data Portal	Provides a catalogue of externally hosted datasets (but not data hosting itself)	<ul style="list-style-type: none"> • As-a-service hosted solution • Ability for 3rd parties to host data on the platform • Richer ways to interlink and align data from different datasets
Project Open Data	A software stack for Open Data management, but not a hosted solution	<ul style="list-style-type: none"> • As-a-service hosted solution • Focus on Linked Data and SPARQL endpoints for complex data queries • Ability for 3rd parties to host data on the platform • Richer ways to interlink and align data from different datasets
COMSODE	Data publication platform and methodology, focus on open data	<ul style="list-style-type: none"> • As-a-service hosted solution • Ability for 3rd parties to host data on the platform

4 Data Layer Architecture

Section “Requirements for the Data Layer” provided the end-user requirements for the data layer of DaPaaS. Based on these high-level business requirements, this section outlines the “technical view” of the DaPaaS data layer with concrete capabilities and components supporting the identified business requirements.

4.1 Data Model

The DaaS solutions analysed in section “Existing Data-as-a-Service Solutions” provide either a tabular (entity/attribute) data model of the hosted data, or a graph based RDF model. In some cases the data is available in its original form, e.g. key/value access to documents, spreadsheets and other files without a pre-defined data structure.

Since DaPaaS is providing a data hosting environment for both RDF and non-RDF data, the data layer cannot commit to one unified view only and will provide the option for hosted data to be represented and thus available in various ways, based on the data provider's choices.

4.1.1 Key/Value Data

Key/value access is the simplest way to access data hosted on the data layer, by referencing the data object identifier (a URL) or by providing a simple restriction over the metadata associated with the data object (e.g. author, timestamp, keywords, etc.) This is useful for data which does not abide to a predefined logical data model and structure, such as office formats (PDF, spreadsheets), images, etc.

Some simple extraction from such files (PDF, spreadsheets) into more structured tabular data or RDF data – for example a table – will be provided by the import adapters in the data layer.

A query of this type may look like:

KEY=OBJECT_ID

4.1.2 Tabular Data

A lot of data that will be hosted on the platform abides to a tabular logical model, for example relational tables, spreadsheets / CSV files, etc. Such data may be queried by more complex attribute/value restrictions.

Depending on the preferences of the data provider, some of the legacy tabular data may be converted into RDF / Linked Data via standard (RDB2RDF) or custom approaches (CSV-to-RDF).

A query of this type may look like:

ATTRIBUTE1=VALUE1&ATTRIBUTE2=VALUE2&...

4.1.3 RDF & Linked Data

The RDF and Linked Data model provide the most flexibility in terms of representing, interlinking and querying data. Data stored as RDF on the platform will be accessible via arbitrary complex and expressive SPARQL queries, may be queried in a federated manner, or may be interlinked with other semantic data on the platform (depending on data provider preferences).

4.2 Technical Capabilities

This section outlines the technical capabilities and components for the Data Layer, based on the business requirements of the different roles/actors identified in section “Requirements for the Data Layer”.

4.2.1 RDF Storage & Access

In order to provide functionality for managing and querying RDF data, the Data Layer will provide an RDF datastore component, which is accessible via standard Jena⁴³ & OpenRDF⁴⁴ APIs and exposes a SPARQL endpoint.

Section 5.1 provides a brief overview of such software components.

4.2.2 Non-RDF Storage & Access

Since some of the data stored on the platform will not conform to the RDF data model, a datastore component providing efficient and scalable key/value queries and queries over tabular/nested structures is also required.

The non-RDF data may be RDF-ised (based on data owner preferences) and available for SPARQL querying from the RDF data store as well. When data is RDF-ised, the original form (CSV, JSON, etc.) will be preserved, since some applications may be using this data form via simple key/value queries and API access, and not utilise more expressive ways to query and explore data such as SPARQL.

Section 5.2 provides a brief overview of such software components.

4.2.3 Full-text Search

Some of the Open Data hosted on the platform may include textual content and documents, or large amounts of textual fields within structured data. In order to perform efficient lookup & discovery on such data, some full-text search capability is also required for the Data Layer.

Full-text search capabilities are usually available in the data management components (Sections 4.2.1 and 4.2.2) and there exist very scalable standalone solutions as well (Section 5.2.4).

4.2.4 Data Catalogue

A data catalogue API is required for efficient dataset exploration and discovery. Void (3.3.4) and DCAT (3.3.3) provide such emerging standards (and APIs), so such technical capability should be available in the data layer prototype.

4.2.5 Querying

As clarified in section “Data Model”, the data layer will provide various types of query capabilities with different expressivity:

- Simple key/value lookups for binary or non-structured data;
- More complex filtering and attribute/value restrictions for structured, unstructured (text) and semi-structured data;
- Complex and expressive SPARQL queries over RDF data.

The query functionality will be available to end-user applications and other services on the platform in two ways:

- RESTful APIs
- Standard SPARQL endpoint

4.2.6 Interlinking

While more complex, semi-automated or manual data transformation capabilities will be exposed in the Platform Layer of DaPaaS, some simple and automated interlinking of RDF data can already be performed within the Data Layer, without the need for significant data publisher involvement (apart from

⁴³ <https://jena.apache.org/index.html>

⁴⁴ <http://www.openrdf.org/>

some configuration of the interlinking process parameters). Such capability will be available in the Data Layer so that automated alignment & interlinking between RDF datasets will be possible.

4.2.7 Caching

Caching of query results may be required in order to improve Data Layer performance and scalability and reduce the overload of the data storage and management components of the platform. It is important to note that efficiently caching SPARQL query results is very challenging, due to the expressivity and complexity of such queries. However key/value lookups and simple attribute/value queries as well as catalogue metadata which are accessed frequently, provide good candidates for performance speedup via caching.

4.2.8 Import / Export

Dataset import and export is an important feature of the Data Layer of the platform. As already clarified, DaPaaS will import open data in variety of formats and will not restrict data to RDF-only (as some other hosting solutions do). At data import time various adapters may provide means for automated RDF-isation of the data being imported, e.g. direct RDB2RDF mapping of relational/tabular data, direct CSV-2-RDF transformations, etc.

Datasets will be available for export and download as archived data dumps, based on data publisher preferences.

4.2.9 Notifications & Statistics

The Data Layer should provide various notifications about data changes and performance. For example:

- New datasets added to the platform;
- New data added to a dataset / existing data deleted or modified;
- Combined datastore load – number of queries per second, number of I/O operations, data input/output, etc., for the management aspects of the system and the Instance Operator;
- Account activity – number of queries per user, data in/out, etc. so that proper quotas and resource limits may be enforced in order to guarantee “fair use” of the system;
- Various additional notifications to other layers and components of the DaPaaS platform.

4.3 Non-functional Aspects

In addition to the technical capabilities outlined in Section 4.2, which correspond to the functional requirements, certain non-functional technical requirements or aspects will be considered during the implementation and operation of the Data Layer.

4.3.1 Scalability

The data management and querying components should be able to scale up to large volumes of data and concurrent queries. Such components should be able to be deployed in a distributed (replicated or shared-nothing) way so that the Data Layer scales up together with the size of data hosted and the number of concurrent queries by the application and User Interface layers.

4.3.2 User Management & Access Control

The data publisher should be able to specify who has access to the data that it uploads on the DaPaaS platform. The data management & querying components should support at least the following access control levels for the hosted data (based on publisher preferences):

- **Public dataset** - readable and writeable by all DaPaaS users/applications;
- **Read-only dataset** - readable by all DaPaaS users/applications, writable only by dataset owner;
- **Private dataset** - accessible only by dataset owner (private dataset).

The data management components for DaPaaS will be selected so that they support such data access control mechanisms.

4.3.3 Availability

Proper technical means should be provided, so that data availability is ensured. This includes:

- Backups of data hosted on the platform;
- Means for detecting data service failures and automatically provisioning replacement services.

When a live prototype is deployed on a Cloud provider, means for automated backups and load-balancing and failover will be ensured as well (for example Amazon Web Services provides such functionality).

4.3.4 Monitoring

Means for automated service monitoring will be part of the Platform Layer (see more details about this in Deliverable D2.1).

4.4 Data Layer Architecture & Components

Section 4.2 outlined the technical capabilities that will be implemented in the data layer of the platform. This section outlines how these capabilities are mapped to concrete software components and what is the interaction between the different components.

Figure 13 provides the architecture of the DaPaaS Data Layer. A typical example of a dataflow in the Data Layer follows steps such as:

1. Data Publisher (DP) imports a new dataset into the platform, either via the APIs or via the UI.
2. At import time the DP may decide to apply direct transformations over the imported data (RDB2RDF, CSV RDF-isation, other types of simple and direct RDF-isation).
3. Data enters the Open Data Warehouse:
 - a. RDF data goes directly into the RDF/metadata datastore.
 - b. Non-RDF data goes into the content store.
 - i. Optionally, the data may be RDF-ised and new data added to the RDF datastore as well.
 - c. Textual data is automatically indexed.
 - d. Optionally, analytics may be automatically performed over the data (e.g. RDF Rank calculations).
4. Optionally, automated interlinking of the new data may be applied and the resulting mappings are returned to the DP for revision and approval. The approved mappings can be imported by the DP into the platform as well.
5. Optionally DP may apply changes to its data (deletions and updates).
6. Application Developers (AD) and End-user Data Consumers (EU) browse the hosted catalogues either via the VoiD / DCAT APIs, or via a graphical UI (outside of the Data Layer)
7. AD and EU access data via the APIs and query endpoints
 - a. Optional caching is employed when possible, in order to speed-up the performance
8. Notifications & Statistics are sent to the centralised management infrastructure of the platform
9. Optionally, DP/AP/EU may export some dataset and get a local copy of its original format as well as any newly generated metadata.

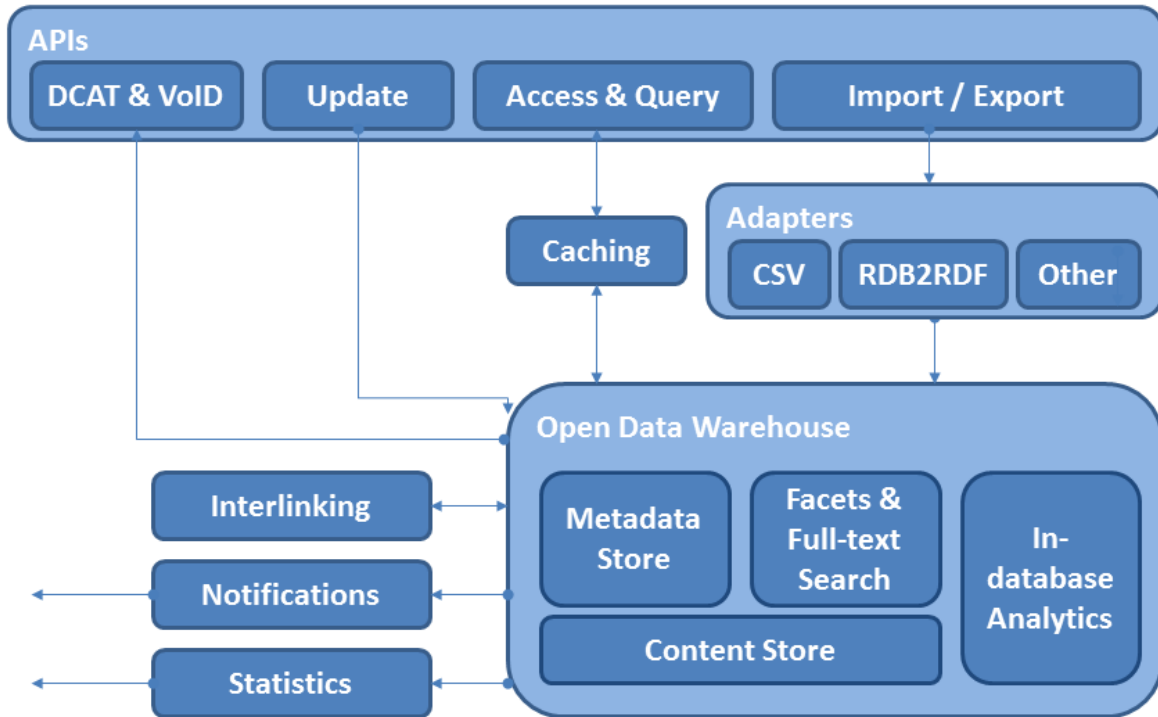


Figure 13: Data Layer Architecture

4.5 Summary of Addressed Requirements

Table 6 clarifies how the business requirements (introduced in section “Requirements for the Data Layer”) are addressed by the components of the Data Layer (a '+' is used to indicate this relation for each requirement).

Table 6: Addressed requirements by components of the Data Layer

DaPaaS Platform Requirement	Import & Export	Import Adapters	Data Access & Query	Data Updates	Catalogue Access	Caching	Interlinking	Metadata Store	Content Store	Full-text Search	In-database Analytics	Notifications & Statistics
IO-01												
IO-02												+
IO-03												+
IO-04												
IO-05			+					+	+			+
IO-06												
DP-01	+	+										
DP-02			+	+				+	+	+		
DP-03					+						+	
DP-04							+				+	
DP-05												
DP-06								+	+			+
DP-07					+			+	+			
DP-08						+		+	+	+		+
DP-09								+	+	+		+
DP-10												
DP-11			+	+	+							
DP-12												
DP-13												
AD-01			+	+	+	+		+	+	+	+	
AD-02	+											
AD-03												
AD-04												
AD-05												
AD-06												
AD-07												
AD-08												
EU-01												

EU-02			+		+	+		+	+	+	+	
EU-03												+
EU-04												
EU-05	+											
EU-06								+	+			

5 Relevant Tools

This section provides an overview of various tools which are relevant to the technical capabilities and components identified in sections “Technical Capabilities” and “Data Layer Architecture & Components”.

5.1 RDF Databases

A triplestore is a purpose-built database for the storage and retrieval of triples, a triple being a data entity composed of subject-predicate-object.

Much like a relational database, one stores information in a triplestore and retrieves it via a query language. Unlike a relational database, a triplestore is optimized for the storage and retrieval of triples. In addition to queries, triples can usually be imported/exported using RDF and other formats.

Some triplestores have been built as database engines from scratch, while others have been built on top of existing commercial relational database engines (i.e. SQL-based).

5.1.1 OWLIM

OWLIM is a high-performance semantic repository created by Ontotext. It is implemented in Java and packaged as a Storage and Inference Layer (SAIL) for the Sesame RDF framework. OWLIM is a native RDF rule-entailment and storage engine. The supported semantics can be configured through rule-set definition and selection. Included are rule-sets for OWL-Horst, unconstrained RDFS with OWL Lite and the OWL2 profiles RL and QL. Custom rule-sets allow tuning for optimal performance and expressivity.

Reasoning and query evaluation are performed over a persistent storage layer. Loading, reasoning and query evaluation proceed extremely quickly even against huge ontologies and knowledge bases.

OWLIM can manage billions of explicit statements on desktop hardware and can handle tens of billions of statements on commodity server hardware.

Features⁴⁵ of OWLIM include:

- Pure Java implementation, ensuring ease of deployment and portability;
- Compatible with Sesame 2, which brings interoperability benefits and support for all major RDF syntaxes and query languages;
- Compatible with Jena with a built in adapter layer;
- Customisable reasoning, in addition to RDFS, OWL-Horst, and OWL 2 RL support;
- Optimized owl:sameAs handling, which delivers dramatic improvements in performance and usability when huge volumes of data from multiple sources are integrated.
- Clustering support brings resilience, fail-over and scalable parallel query processing;
- Geo-spatial extensions for special handling of 2-dimensional spherical data allowing data using the WGS84 RDF vocabulary to be indexed and processed quickly using a variety of special geometrical query constructions and SPARQL extensions functions;
- Full-text search support, based on either Lucene or proprietary search techniques;
- High performance retraction of statements and their inferences – so inference materialisation speeds up retrieval, but without delete performance degradation;
- Powerful and expressive consistency/integrity constraint checking mechanisms;
- RDF rank, similar to Google's PageRank, can be calculated for the nodes in an RDF graph and used for ordering query results by relevance, visualisation and any other purposes;
- RDF Priming, based upon activation spreading, allows efficient data selection and context-aware query answering for handling huge datasets;
- Notification mechanism, to allow clients to react to statements in the update stream.

⁴⁵ Some features are not available in *OWLIM-Lite* edition (refer <http://owlim.ontotext.com/> for details)

OWLIM family versions:

- **OWLIM-Lite** is fast semantic repository, supporting non-trivial inference with tens of millions of statements on contemporary desktop hardware.
- **OWLIM-SE** is an extremely scalable semantic repository: it can load tens of billions of RDF statements, using non-trivial inference and delivering outstanding multi-user query performance. OWLIM-SE is a robust engine packed with advanced features that bring unmatched efficiency to a huge variety of application scenarios:
 - optimized owl:sameAs handling that delivers dramatic improvements in performance and usability when huge volumes of data from multiple sources are integrated
 - hybrid querying capabilities that combine SPARQL with efficient full-text search, geo-spatial constraints and ranking of query results
- **OWLIM-Enterprise** is a replication cluster infrastructure based on OWLIM-SE. It offers industrial strength resilience and linearly scalable parallel query performance, with support for load-balancing and automatic fail-over

5.1.2 Sesame

Sesame⁴⁶ is an open source Java framework for storage and querying of RDF data. The framework is fully extensible and configurable with respect to storage mechanisms, inferencers, RDF file formats, query result formats and query languages. Sesame offers a JDBC-like user API, streamlined system APIs and a RESTful HTTP interface supporting the SPARQL Protocol for RDF.

Out of the box, Sesame supports SPARQL and SeRQL querying, a memory-based and a disk-based RDF store and RDF Schema inferencers. It also supports most popular RDF file formats and query result formats. Various extensions are available or are being worked at elsewhere.

Here follows a high-level overview of Sesame's components:

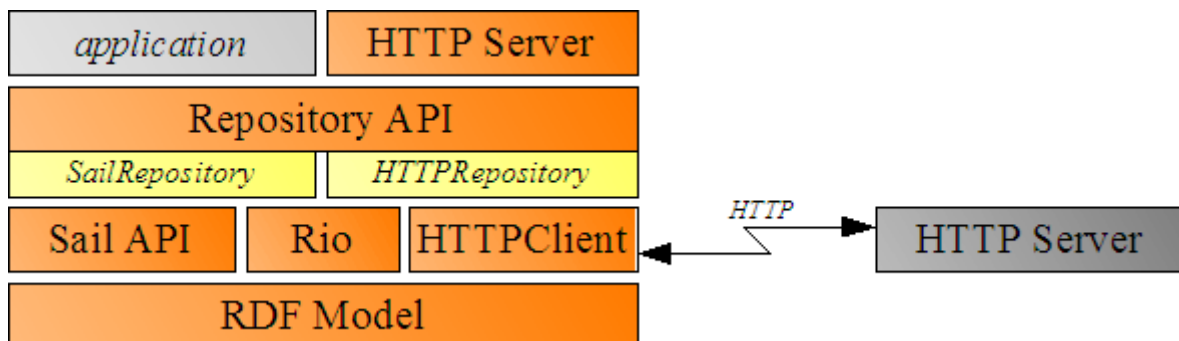


Figure 14: Sesame Architecture

All the way at the bottom of the diagram is the **RDF Model**, the foundation of the Sesame framework. Being an RDF-oriented framework, all parts of Sesame are to some extent dependent on this RDF model, which defines interfaces and implementation for all basic RDF entities: URI, blank node, literal and statement.

Rio, which stands for "RDF I/O", consists of a set of parsers and writers for various RDF file formats.

The Storage And Inference Layer (SAIL) API is a low level System API for RDF stores and inferencers. Its purpose is to abstract from the storage and inference details, allowing various types of storage and inference to be used (mainly of interest for triplestore developers).

The **Repository API** is a higher level API that offers a large number of developer-oriented methods for handling RDF data. It offers various methods for uploading data files, querying, and extracting and manipulating data. There are several implementations of this API, the ones shown in this figure are the

⁴⁶ <http://www.openrdf.org/>

SailRepository and the *HTTPRepository*. The former translates calls to a SAIL implementation of choice, the latter offers transparent client-server communication with a Sesame server over HTTP.

The top-most component in the diagram is the **HTTP Server**. The HTTP Server consists of a number of Java Servlets that implement a protocol for accessing Sesame repositories over HTTP. The details of this protocol can be found in Sesame's system documentation, but most people can simply use a client library to handle the communication. The HTTPClient that is used by the HTTPRepository is one such library.

5.1.3 Apache Jena Framework

Jena⁴⁷ is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL and updated through SPARUL.

Jena is similar to Sesame; though, unlike Sesame, Jena provides support for OWL (Web Ontology Language). The framework has various internal reasoners and the Pellet reasoner⁴⁸ (an open source Java OWL-DL reasoner) can be set up to work in Jena.

Jena architecture overview:

- The **Graph layer** is the base layer in Jena. It is very granular and is a very minimal implementation of the RDF specification. It permits a wide range of implementations, such as in-memory or persistent triple stores.
- The **Model layer** extends the core functionality in the Graph layer in such a way, that by allowing developers to work with objects of type "Resource" or "Property" or "Statement", instead of "Node" or "Triple".
- The **OntModel Layer** supports inference capabilities, that is, the ability to work with triples that are implied, in addition to the triples that have been explicitly defined.

5.2 NoSQL Databases

A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability. NoSQL databases are often highly optimized key-value stores intended primarily for simple retrieval and appending operations.

In the context of the CAP theorem⁴⁹, NoSQL stores often compromise consistency in favor of availability and partition tolerance. Barriers to the greater adoption of NoSQL data stores in practice include: the lack of full ACID transaction support, the use of low-level query languages, the lack of standardized interfaces, and the huge investments already made in SQL by enterprises.

The promise of the NoSQL database has generated a lot of enthusiasm, but there are many obstacles to overcome before they can appeal to mainstream enterprises. Here are a few of the top challenges.

- **Maturity** - RDBMS systems, been around for a long time, are stable and richly functional. In comparison, most NoSQL alternatives are in pre-production versions with many key features yet to be implemented. Living on the technological leading edge is an exciting prospect for many developers, but enterprises should approach it with extreme caution.
- **Support** - enterprises want the reassurance that if a key system fails, they will be able to get timely and competent support. All RDBMS vendors go to great lengths to provide a high level of enterprise support. In contrast, most NoSQL systems are open source projects, and although there are usually one or more firms offering support for each NoSQL database, these companies often are small start-ups without the global reach, support resources, or credibility of an Oracle, Microsoft, or IBM.

⁴⁷ <http://jena.apache.org/>

⁴⁸ <http://clarkparsia.com/pellet>

⁴⁹ <http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>

- Analytics and business intelligence - NoSQL databases have evolved to meet the scaling demands of modern Web 2.0 applications. Consequently, most of their feature set is oriented towards the demands of these applications. However, data in an application has value to the business that goes beyond the insert-read-update-delete cycle of a typical Web application. NoSQL databases offer few facilities for ad-hoc query and analysis. Sometimes a simple query requires significant programming expertise, and commonly used BI tools do not provide connectivity to NoSQL. There are some solutions such as HIVE or PIG which approach the problem, providing easier access to data held in Hadoop clusters and perhaps eventually, other NoSQL databases.
- Administration - the design goals for NoSQL may be to provide a zero-admin solution, but the current reality falls well short of that goal. NoSQL today requires a lot of skill to install and a lot of effort to maintain.
- Expertise - There are literally millions of developers throughout the world, and in every business segment, who are familiar with RDBMS concepts and programming. In contrast, almost every NoSQL developer is in a learning mode. This situation will be addressed naturally over time, but for now, it's far easier to find experienced RDBMS programmers or administrators than a NoSQL expert.

5.2.1 Cassandra

Apache Cassandra⁵⁰ is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters, with asynchronous masterless replication allowing low latency operations for all clients.

Cassandra's data model is a partitioned row store with tuneable consistency. Rows are organized into tables; the first component of a table's primary key is the partition key; within a partition, rows are clustered by the remaining columns of the key. Other columns may be indexed separately from the primary key. Tables may be created, dropped, and altered at runtime without blocking updates and queries.

Main features of Cassandra include:

- Decentralized - every node in the cluster has the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.
- Replication - Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra's distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.
- Scalability - read and write throughput both increase linearly as new machines are added, with no downtime or interruption of applications.
- Fault-tolerant - data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centres is supported. Failed nodes can be replaced with no downtime.
- Query language CQL (Cassandra Query Language) - a SQL-like alternative to the traditional RPC interface. Language drivers are available for Java (JDBC), Python (DBAPI2) and Node.JS (Helenus).

5.2.2 HBase

Apache HBase⁵¹ is an open-source, distributed, versioned, column-oriented store build on top of Hadoop and HDFS.

⁵⁰ <http://cassandra.apache.org/>

⁵¹ <http://hbase.apache.org/>

HBase is applicable for cases where random, real-time read/write access to Big Data is needed. HBase's goal is the hosting of very large tables - billions of rows X millions of columns -- atop clusters of commodity hardware.

HBase features include:

- Linear and modular scalability.
- Strictly consistent reads and writes.
- Automatic and configurable sharding of tables
- Automatic failover support in distributed cluster deployment.
- Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables.
- Easy to use Java API for client access.
- Block cache and Bloom Filters for real-time queries.
- Query predicate push down via server side Filters
- Thrift gateway and a RESTful Web service that supports XML, Protobuf, and binary data encoding options
- Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX

5.2.3 CouchBase

CouchBase is a distributed, document oriented database using JSON as a native data model. Each document is uniquely named in the database, and CouchBase provides a RESTful HTTP API for reading and updating (add, edit, delete). Documents consist of any number of fields, metadata and attachments. Document updates are all or nothing, either succeeding entirely or failing completely. The database never contains partially saved or edited documents.

CouchBase features include:

- Flexible data model based on JSON
- Primary and secondary indices build as views on documents
- Data querying based on: explicit key, list of keys, ranges of keys
- MapReduce support for incremental data indexing and querying
- Horizontal scalability, both cross-cluster data sharding and replication supported
- Built-in object-level cache, based on memcached
- Fully supported SDKs for Java, C#, PHP, C, Python and Ruby

5.2.4 Solr

Solr⁵² is the popular, fast open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, near real-time indexing, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest Internet sites.

Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Jetty. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has an extensive plug-in architecture when more advanced customization is required.

⁵² <http://lucene.apache.org/solr/>

5.3 RDB2RDF Tools

5.3.1 db2triples

The db2triples library is a software tool for extracting data from relational databases and loading data into an RDF triple store. It implements R2RML and Direct Mapping standards defined by W3C's RDB2RDF. The implementation has been validated by the RDB2RDF working group and it has served as a bench test during the different phases of definition and validation of the RDB2RDF recommendations.

In a nutshell, db2triples accepts as input RDBMS connection parameters and a R2RML transformation document (and no mapping in case of *Direct Mapping* mode). The output is the RDF dump of the database data based on the transformation mode used. The tool manages the memory efficiently, which in turn enables it to process large amounts of data.

Supported RDBMS include MySQL and PostgreSQL.

Db2triples is an open source software implemented in Java, published under the LGPL license. The source code and building instructions are available at Github⁵³.

5.4 PDF Import Tools

5.4.1 Tabula

Tabula⁵⁴ is a free tool for extracting tabular data out of (text-based) PDF files. It provides simple web interface which lets the user to upload input PDF documents and extract the relevant data. The backend functionality is shaped as a web service accessible over HTTP from any client browser. The output serialisation format is either CSV or TSV. Tabula is free and available under the MIT open-source license.

Tabula applies different heuristics to cover a broader range of tables and styles in the PDF document: tables with, without or a mixture of ruler lines, cell shading and colouring. There are certain limitations caused by the complexity of the tasks, such as processing scanned PDFs or complex (non rectangular) tables having rows or columns spanning through several cells. Other limitations are due to immaturity of the early stage of development.

5.5 CSV import tools

5.5.1 CSV-to-API

CSV-to-API is a lightweight tool serving as data wrapper on top of static CSV dumps. It exposes its functionality as RESTful API, accepting as input URL of the source CSV data and outputs the transformed data into JSON, XML or HTML format. This way the tool itself is completely decoupled from the data it processes and can be deployed on any server and reused for different data sources.

Additionally CSV-to-API serves as a filter sitting between the CSV data and the client providing simple data filtering and sorting functionalities.

The generated data uses the original CSV column names as key names. This implies certain restrictions on the CSV sources.

Example:

```
http://example.com/csv-to-api/?source=http://www.gsa.gov/dg/data_gov_bldg_star.csv
&Region+Code=11
&format=html
&sort=Bldg+Zip
```

(export data filtered by Region Code, sorted by Bldg Zip in HTML format)

⁵³ <https://github.com/antidot/db2triples/>

⁵⁴ <https://github.com/jazzido/tabula>

The tool is open source, written in PHP and available on GitHub⁵⁵.

5.5.2 CSV.js

CSV.JS is a simple pure JavaScript CSV library focused on the browser side processing. It offers different processing operations on CSV data including import from inline data, remote data (URL) or from HTML5 file object. The internal data representation is a JavaScript object which can be serialized naturally as JSON. The implementation is based on jQuery framework.

5.5.3 Mr. Data Converter

Mr. Data Converter is a completely browser side application based on JavaScript. It processes CSV/TSV data and can generate XML, JSON, ASP/VBScript or basic HTML table formatting as well as arrays in PHP, Python (as a dictionary) and Ruby. It is a free open source tool available at GitHub⁵⁶.

5.6 Silk Framework

The Silk Link Discovery Framework⁵⁷ is a tool supporting data publishers in accomplishing the task of discovering relationships between data items within different Linked Data sources. Using the declarative Silk - Link Specification Language (Silk-LSL), developers can specify which types of RDF links should be discovered between data sources as well as which conditions the data items must fulfil in order to be interlinked. These link conditions may combine various similarity metrics and can take the graph around a data item into account, which is addressed using an RDF path language. Silk accesses the data sources that should be interlinked via the SPARQL protocol and can thus be used against local as well as remote SPARQL endpoints.

The main features of the Silk Framework are:

- Flexible, declarative language for specifying linkage rules
- Support of RDF link generation in RDF
- Employment in distributed environments (by accessing local and remote SPARQL endpoints)
- Usable in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist
- Scalability and high performance through efficient data handling:
 - Reduction of network load by caching and reusing of SPARQL result sets
 - Multi-threaded computation of the data item comparisons
 - Optional blocking directive, which allows users to reduce the number of comparisons on cost of recall, if necessary.
- Active Learning of expressive linkage rules using genetic programming⁵⁸
- Silk Workbench - a web application which guides the user through the process of creating link specifications, results evaluation, and linkage rules learning.

6 Recommendations

This section provides recommendations for 3rd party software components and frameworks to be adopted, based on the architecture outline in section “Data Layer Architecture & Components” and the analysis of various tools in section “Relevant Tools”. Additionally, an initial provisioning plan (in terms of hardware & software requirements) for the data layer components is provided.

⁵⁵ <https://github.com/project-open-data/csv-to-api>

⁵⁶ <https://github.com/shancarter/Mr-Data-Converter>

⁵⁷ <http://www4.wiwiw.fu-berlin.de/bizer/silk/>

⁵⁸ <http://dws.informatik.uni-mannheim.de/fileadmin/lehrstuehle/ki/pub/IseleBizer-ActiveLearningOfExpressive-LinkageRules-JWS2013.pdf>

6.1 Components

Referring back to the Data Layer Architecture and the relevant tools reviewed so far, here we provide recommendations on which tools should be considered when developing the corresponding architectural components.

Metadata Store

Any of the free RDF frameworks can be used for the DaaS Platform, however taking into account the large amount of data the platform aims to host and the complexity of queries capable to answer, OWLIM is the preferred choice.

Content Store

Depending on the type of the data we are hosting a different content store implementation is more appropriate. For storing tabular data and being able to answer more complex queries, a column store implementation is necessary. Both HBase and Cassandra are good candidates, each having its advantages. Cassandra performs better on data updates however it comes with the price of eventual consistency. HBase has better integration with PIG⁵⁹ and HIVE⁶⁰ which is beneficial for mass data processing.

For more complex / nested data structures represented as JSON, the recommended content store is CouchBase (but alternatives like MongoDB will also satisfy the data layer requirements).

In-database analytics

For doing in-database analytics we can apply two specific techniques supported by OWLIM: RDF ranking and spreading activation. The former provides data ranking on the base of local connectivity of the nodes, while the latter provides context-aware data selection and query answering.

Faceted & Full-text Search

For various faceted general purpose full-text indexing and search, Solr provides sufficient capabilities.

Import/Export Adapters

There are plenty of tools supporting data transformations from various input formats. For RDFization of RDBMS data we'll use db2triples (one of the first implementations of W3C's *R2RML* and *DirectMapping*)

Interlinking

The Silk framework is a quite promising solution for doing automated interlinking between entities from different datasets. Though its major strength is in the active supervised linkage rules learning (applicable for WP2), the capabilities to efficiently generate linkage data on large datasets makes it a good candidate for the Data Layer as well.

⁵⁹ <https://pig.apache.org/>

⁶⁰ <http://hive.apache.org/>

6.2 Deployment & Provisioning

This section provides requirement estimations regarding the deployment of the DaaS infrastructure in terms of hardware and software components. Note that this is the initial roadmap and some changes can be expected.

6.2.1 Hardware Requirements

Table 7: Initial hardware requirements

Resource	Amount (base)	Comments
RAM	128 GB	The amount of RAM has direct impact on the open data warehouse performance (various indexes and internal caches)
HDD	SATA 2TB	This amount will be shared between the content store, metadata store and FTS indexes. The estimated storage amount is capable to host the popular linked open datasets and any overhead related to indexing and additional metadata persistence. SSD disks will improve the performance under extremely heavy query loads but SATA disks will be sufficient.
CPU	2x8 Xeon	Computational power is also important aspect related to efficient data transformations (import/export) as well as complex query answering.

6.2.2 Software Requirements

Table 8: Initial software requirements

Software	Type/version	Comments
OS	Any	None of the components demand certain OS requirements (mostly Java based implementations)
JVM	1.6+	The RDF database requires Java Virtual Machine v1.6 (or newer)
Application server	Tomcat 7.x	Tomcat is required for deploying the RDF database and its administrative tools
Hadoop/HDFS	2.x	Needed for running the content store (HBase)
HBase	0.96+	We recommend the latest stable release
OWLIM	5.4+	This is not mandatory requirement, unless scalable RDF storage and querying is necessary

Bibliography

- DaPaaS Project. (2014). *D2.2 - Open PaaS requirements, design & architecture specification*.
- EPSI. (2010). *The Five Stars of Open Data*.
- LOD2 Project. (2012). *Managing the Life-Cycle of Linked Data with the LOD2 Stack*. 11th International Semantic Web Conference. Boston.
- Microsoft Corp. (2011). *Windows Azure Marketplace*.
- Microsoft Corp. (2013). *Open Data Protocol (OData)*.
- W3C. (2014). *JSON-LD 1.0 - A JSON-based Serialization for Linked Data*. W3C.