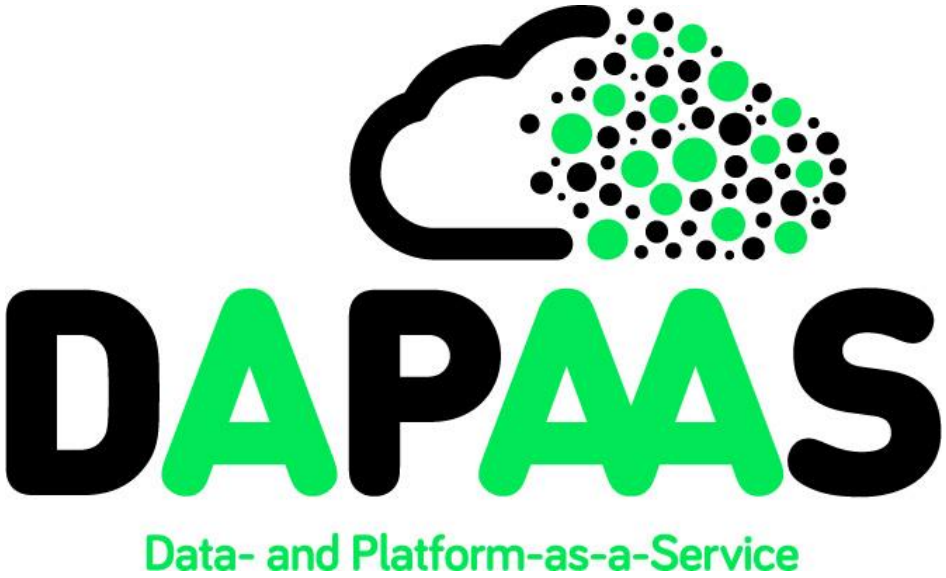
Small or medium-scale focused research project (STREP)

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 D2.3

Open Data PaaS prototype, v.2

Date:	31 July 2015
Author(s):	Alex Simov (Ontotext) , Marin Dimitrov (Ontotext), Nikolay Nikolov (SINTEF), Antoine Pultier (SINTEF), Dina Suhobok (SINTEF), Xianglin Ye (SINTEF), Dumitru Roman (SINTEF)
Dissemination level:	PU
WP:	WP2
Version:	1.0

Document metadata

Quality assurors and contributors

Quality assuror(s)	Rick Moynihan (Swirrl), Amanda Smith (ODI)
Contributor(s)	DaPaaS Consortium

Version history

Version	Date	Description
0.1	21.07.2015	Initial outline and Table of Contents (TOC)
0.2	23.07.2015	Preliminary content and input to Section 2
0.3	24.07.2015	Version for internal review
0.4	28.07.2015	Review comments implemented
1.0	30.07.2015	Adjustments and finalization.

Executive Summary

This report represents the updates on the Platform-as-a-Service (PaaS) layer for the final prototype of the DaPaaS platform, more recently known as DataGraft and deployed at DataGraft.net. For brevity the document focuses on the changes made since the first version of the platform described in D2.2[1].

Significant progress has been achieved in the development of Grafterizer (formerly Grafter GUI frontend framework), an interactive tool for data cleaning and transformation tasks based on Grafter.

New transformations management and execution components have been developed to support the access, persistence and execution of data transformations.

The catalogue services have been extended to support searching and metadata management for data transformations.

[1] Available via http://project.dapaas.eu/dapaas-reports

Table of Contents

EXECUTIVE SUMMARY **3**
TABLE OF CONTENTS **4**
LIST OF ACRONYMS **5**
LIST OF FIGURES **6**
1 INTRODUCTION **7**
2 UPDATED PLATFORM IMPLEMENTATION – FINAL PROTOTYPE **8**
2.1 DATAGRAFT PLATFORM INTERFACE 8
2.1.1 Features 8
2.1.2 Implementation of the platform interface 9
2.2 DATA CLEANING AND TRANSFORMATIONS DEVELOPMENT 9
2.2.1 Grafterizer features 10
2.2.2 Grafterizer implementation 11
2.3 FLOW, USER DOCUMENTATION, TUTORIALS 11
2.3.1 Wireframes 12
2.3.2 Scenario-based videos 12
2.3.3 User documentation 12
2.3.4 API documentation 12
2.4 TRANSFORMATIONS MANAGEMENT SERVICE 13
2.5 TRANSFORMATIONS CATALOGUE SERVICE 13
2.6 USER MANAGEMENT & ACCESS CONTROL 14
3 CONCLUSION **16**
4 APPENDIX A: DATAGRAFT WIREFRAMES DESIGN **17**
4.1 FLOW 1: CREATE FIRST DATA PAGE 17
4.2 FLOW 2: DASHBOARD ACTIONS 18
4.3 FLOW 3: SEARCH AND EXPLORE DATA 18
5 APPENDIX B: API DOCUMENTATION **19**
5.1 TRANSFORMATIONS MANAGEMENT API 19
5.2 TRANSFORMATIONS CATALOGUE API 20
5.3 API KEYS MANAGEMENT 21

List of Acronyms

API	Application Programming Interface
CSV	Comma Separated Values (format)
DaaS	Data-as-a-Service
DSL	Domain Specific Language
GUI	Graphical User Interface
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation (format)
PaaS	Platform-as-a-Service
REST	Representational state transfer
RDF	Resource Description Framework
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
DBaaS	Database-as-a-Service

List of Figures

Figure 1: Architecture of the PaaS layer 8
Figure 2. Data cleaning (pipeline) 9
Figure 3. Transformation of data to RDF 10
Figure 4. Screenshot of the Grafterizer UI 10
Figure 5. Transformations Management service 13
Figure 6. DCAT extension for transformations 14
Figure 7. API keys usage 15

1 Introduction

This report represents supporting documentation for the prototype developed for Deliverable D2.3 “Open Data PaaS prototype, v.2”.

This report describes the updates to the platform as a service (PaaS) layer for the final prototype of the DaPaaS platform, more recently known as DataGraft and deployed at DataGraft.net. The document focuses on the changes made since the first version of the platform described in D2.2[2].

Significant progress has been made through the development of Grafterizer (formerly Grafter GUI), an interactive tool for data cleaning and transformation tasks based on Grafter. An extensive description has been provided in Section 2.2.

A new integration component has been developed, responsible for the transformations management and execution tasks, providing support for transformations definitions access, persistence and execution. It provides a RESTful API to be used by other platform components (Grafterizer, DataGraft UI) as well as by 3rd party components (applications).

The catalogue services have been extended to support searching and metadata management for transformations.

The structure of the document is as follows:

- Overview of the updated platform architecture
- DaPaaS platform user interface description
- Grafterizer description
- Transformations management and catalogue services description
- Reference information included as Appendix: wireframes, APIs documentation

[2] Available via http://project.dapaas.eu/dapaas-reports

2 Updated Platform Implementation – Final Prototype

Figure 1 depicts the main software components of the final version of the platform layer, their relationships and associated APIs. The software components of the Platform Layer extend the capabilities offered by the Data Layer in four main service categories:

- **User Management & Access Control**, which manages user profiles and secure access control to transformations and datasets.
- **Interactive Data Cleaning & Transformation Development**, which provides functionalities for developing data cleaning & transformation processes.
- **Transformations Management**, which gives developers control over the deployed transformations and their configuration settings.
- **Transformations Catalogue** for searching and exploring transformations and their metadata.

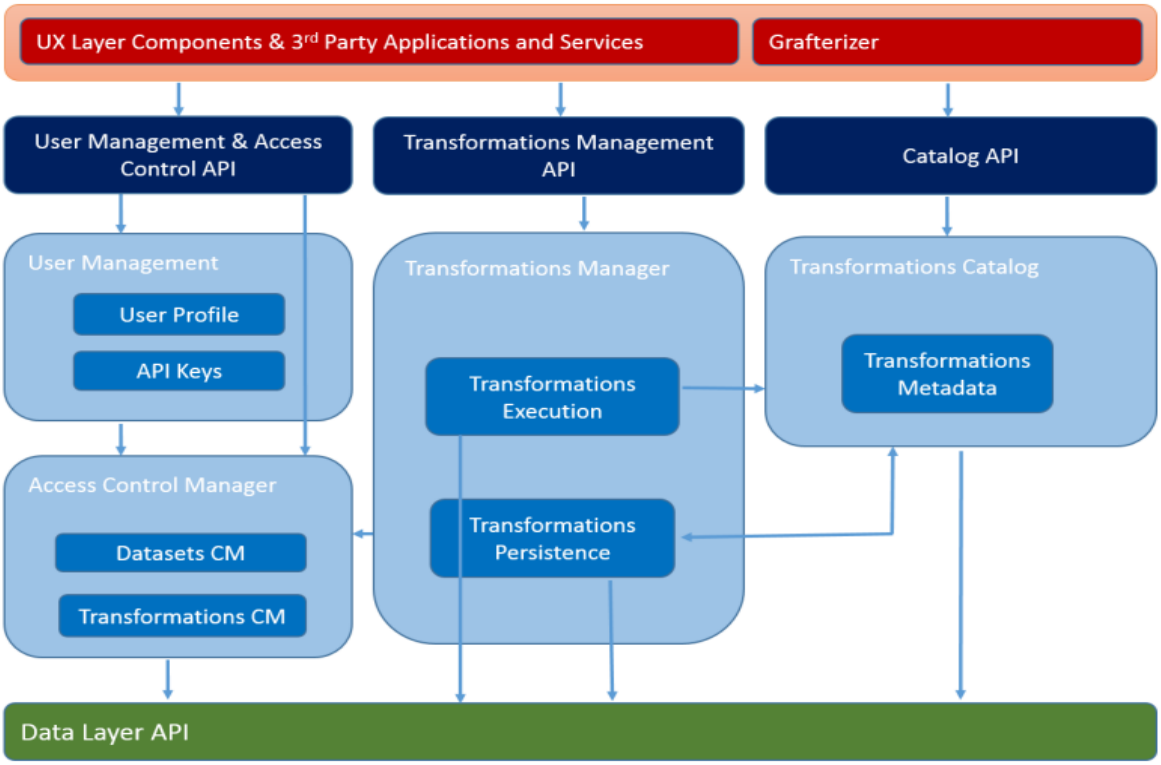


Figure 1: Architecture of the PaaS layer

2.1 DataGraft platform interface

The current version of the DataGraft platform has been deployed and is publicly available online on the domain datagraft.net through the DataGraft platform interface. The platform interface implements the intended platform functionality related to the publishing and data workflows through a web-based portal. The platform GUI is focused around two main types of user assets – data pages and transformations. This section discusses the basic UI features and technologies that have been used in the implementation of the platform interface.

2.1.1 Features

The user interface utilizes the backend APIs to implement the intended platform functionalities in interactive and graphical ways. The main features are as follows:

- **User management** – provides user-friendly log-in and log-out functions
- **Data publishing and access** – user interface that supports easy publishing of data

- **Data exploration** – a basic search and browse interface intended for users that are external to the platform
- **Dashboard** – user interface that provides an overview and management of the main types of user assets – data pages and transformations
- **Interactive Data cleaning and transformation GUI** – datagraft.net provides full integration with the Grafterizer user interface (described in detail below).
- **API access management** – user interface for management of API keys that can be used to access the platform APIs available for each particular user

Details on the individual features and how they are accessed can be found in section 2.2.

2.1.2 Implementation of the platform interface

The datagraft.net platform is built on modern web technologies and frameworks, like *bootstrap*[3], *jquery*[4], *d3*js[5] and *highcharts*[6].

2.2 Data cleaning and transformations development

DataGraft supports activities related to data cleaning and transformation of data that are used to produce processed tabular data or RDF triples and host it on the DataGraft platform. These two functionalities have been implemented in the user interface tool Grafterizer[7]. Grafterizer (referred to as Grafter GUI in deliverable D2.2) is a graphical tool for specifying data cleaning and transformation workflows which consist of two steps (shown on figures below):

- *Pipeline specification* (Figure 2) – specification of individual steps for tabular data cleaning and preparation for mapping to RDF

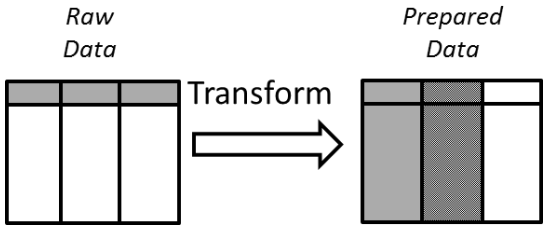


Figure 2. Data cleaning (pipeline)

- *Specification and execution of RDF mapping* (Figure 3) – mapping of prepared data to a linked data ontology or vocabulary and generating corresponding RDF triples

[3] http://getbootstrap.com/
[4] https://jquery.com/
[5] http://d3js.org/
[6] http://www.highcharts.com/
[7] https://github.com/dapaas/grafterizer

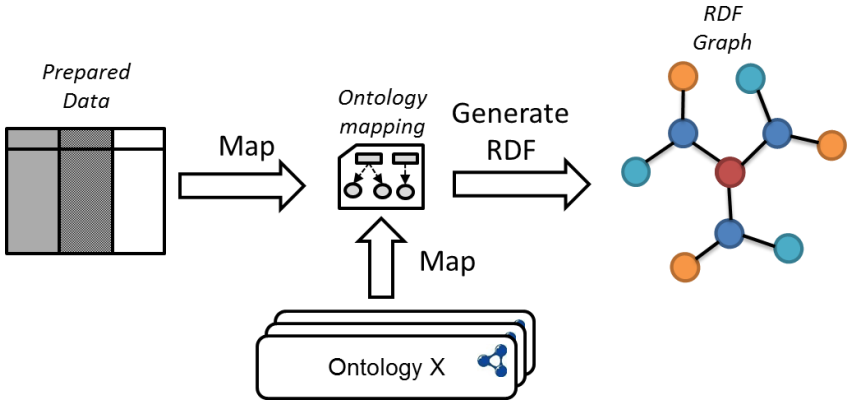


Figure 3. Transformation of data to RDF

The current version of Grafterizer supports features related to the specification, editing, management and storage of such data transformations. The tool has been fully integrated with the datagraft.net platform (see Figure 4) and is available for public use online.

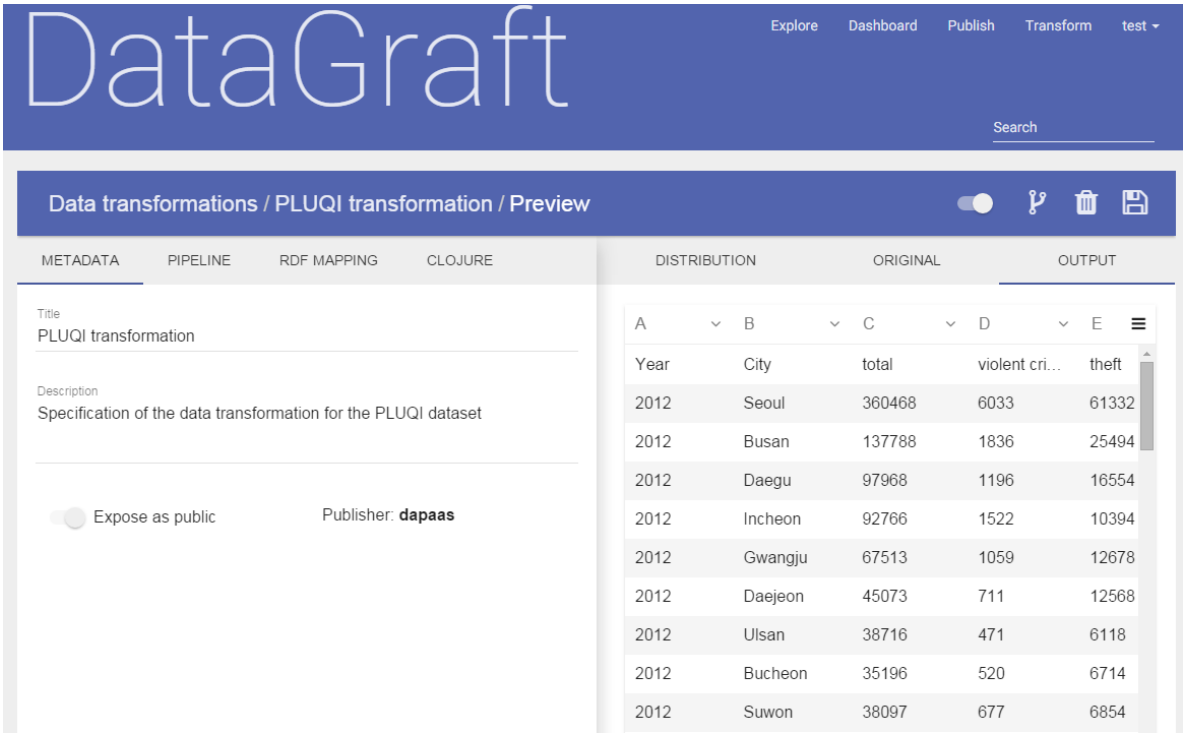


Figure 4. Screenshot of the Grafterizer UI

2.2.1 Grafterizer features

The Grafterizer tool provides an interactive user interface with a wide array of supported functionality useful in the process of cleaning and transforming data:

- **Live preview** - Grafterizer interactively displays the result of the tabular clean-up or transformation steps in a side-panel. It also retains a view of the original version of the uploaded tabular dataset. Additionally, in case errors in the transformation or RDF mapping are present, it provides an integrated error reporting capability.
- **Forking of existing transformations** – the user interface allows users to create copies of transformations by simply clicking a button.

- **Specifying and editing data cleaning (pipeline) steps** – the supported clean-up and transformation functions that can be performed on tabular data can be added, edited, reordered or deleted. All functions are parametrised and editing allows users to change each of these parameters within the function with immediate feedback.
- **Data page generation** – based on the specified RDF mappings, users are able to directly produce and publish data pages where their data will be available for access through an endpoint.
- **Direct download of resulting data** – the cleaned-up/transformed data from Grafterizer (both CSV and mapped RDF) can be directly accessed and downloaded locally.
- **Customisation** – data clean-up and transformation can be heavily customised through embedding custom code, both as individual clean-up steps, or as parameters to certain steps. In addition, developers can directly edit the resulting Clojure code and see the result in Interactive mode.

Details on how the aforementioned features are implemented and available to users are published in the user documentation and video tutorials, discussed in sections 2.3.2 and 2.3.3.

2.2.2 Grafterizer implementation

Grafterizer is an evolution of the Grafter GUI as described in deliverable D2.1. As such, it is implemented as a graphical wrapper over the Grafter[8] library and Graftwerk service developed as part of D4.2. Grafter is a Clojure library that implements reusable, autonomous data transformations and RDF publishing. Grafterizer allows users to specify Grafter transformations in a much easier and more intuitive manner, compared to the traditional approach of coding in Clojure. It also provides instant feedback alongside the other features described in the previous section.

The Grafterizer interface works by submitting the transformation under development to the Graftwerk backend service along with the data to be transformed. Depending on the request Graftwerk will then either generate a preview of the transformed data that the UI can display, or return the transformed data.

The current version of Grafterizer is available as open source software under the Eclipse Public License v1.0 at GitHub. The tool has been implemented in AngularJS[9] as a web application using state-of-the-art web frameworks, libraries and other technologies including:

- Grafter - Transformation API & DSL described in D4.2
- Graftwerk – A backend service for executing grafter transformations and returning transformation previews. Described in D4.2.
- AngularUI – user interface suite for the AngularJS library.
- Angular Material – implementation of Material Design in AngularJS.
- Angular Loading Bar – automatic loading bar control for AngularJS.
- Angular Breadcrumb – AngularUI module for web page breadcrumbs.
- Font Awesome – symbols library used in buttons and graphics.
- jsedn – library for parsing and generation of Grafter/Clojure code.

2.3 Flow, user documentation, tutorials

Throughout the development of the datagraft.net platform, a set of documents have been produced, which are oriented towards the development team and/or end users of the platform. This section describes the output of documentation activities.

[8] http://grafter.org/
[9] https://angularjs.org/

2.3.1 Wireframes

In order to ensure a smooth user experience, during the implementation of the datagraft.net user interface, we have developed a set of wireframes. The wireframes depict a set of flows depending on the user type. There are three main flows:

- **New users** – depicting how new users can sign in and create their first data page
- **Returning users** – depicting how existing users can browse through their resources and perform actions on them
- **Non-signed users** – shows platform functionality available to users outside the platform such as searching and viewing datasets

The wireframes are available in Appendix A: DataGraft Wireframes Design. Note that the wireframes represent a simplified view of the content of the individual pages. The final implementation does not try to use the depicted UI elements literally, but a variation of them, which ensures a better look and feel of the user interface.

The user documentation and tutorials for datagraft.net have been designed to cover the full functionality of the platform. These consist of a set of scenario-based videos, a user documentation document, and a set of developer API specifications. The user documentation and developer API specifications are accessible through links, whereas the scenario-based videos have been embedded into the datagraft.net homepage.

2.3.2 Scenario-based videos

In order to showcase the supported functionality of the datagraft.net platform, we have developed two scenarios for usage. These have then been enacted and recorded, so that users can have a full visual demonstration of different platform capabilities. The two scenarios focus on the two main aspects of the platform – i.e., data transformation and datasets hosting, exploration and querying.

The **data transformation scenario** shows how users can create a simple data transformation over a small example tabular dataset, specify an RDF mapping, and generate RDF triples based on the transformation. Further, it displays how users are able to share and reuse transformations from the cataloguing services of the platform. The video is available on the home page of datagraft.net.

The **dataset hosting, exploration and querying scenario** displays how users are able to do the data management basics. It shows how the user interface can be used to publish data directly from an RDF format, or through a data transformation. It also shows the data workflows and data pages management functionality available to each user to control their resources on the platform. Lastly, it demonstrates how unsigned users may gain access to the published datasets and use the platform provided endpoint to access the data. The video is available on the home page of datagraft.net.

2.3.3 User documentation

In addition to the scenario-based videos that have been developed, we provide a user documentation, which details the individual user controls available in the datagraft.net GUI. It is screen-focused, so that the available user options on each navigation screen are explained. The documentation is available at datagraft.net/documentation.

2.3.4 API documentation

Apart from graphical interfaces, the datagraft.net provides a set of RESTful APIs that can be used to access the various services of the platform. Details on how to use each individual API have been detailed in the datagraft.net API documentation. It is divided into different sections that describe different aspects of the programmatic interfaces:

- General aspects – endpoint specification; security and authentication; internal dataset representation model; data exchange standards for requests
- RESTful APIs – accessing datasets catalog; accessing transformations catalog; working with RDF repositories; executing data transformations

The API specification has been done using API blueprint [10] and published in the platform at datagraft.net/documentation using aglio[11].

2.4 Transformations Management service

This service is an integration component between the transformations execution engine (Graftwerk), the catalogue services and the data layer (DaaS), exposing a simple API for clients to transform their data. It is also responsible for transformations definitions management (CRUD) to support reusable and repeatable transformations functionality. The transformations management is accessible via the *Transformations Catalogue* service APIs.

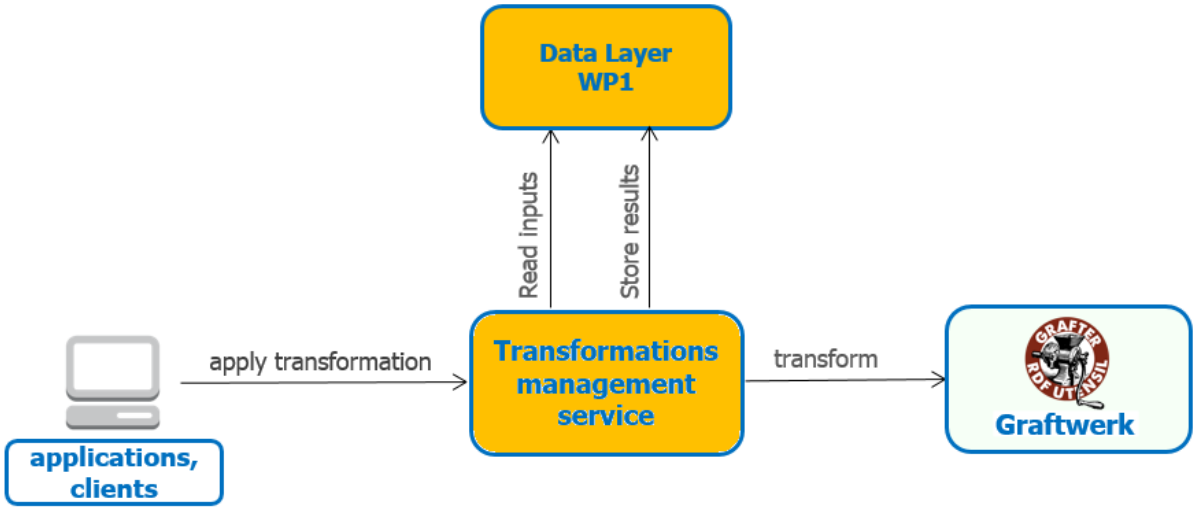


Figure 5. Transformations Management service

On Figure 5 is the general workflow for applying transformations on user's data and storing the result into the data layer. The service APIs offer rich variety of parameters for flexible customisation of the transformation process:

- Input data can be retrieved either from the data layer (pre-existing data) or it can be supplied with the request by the user
- The result from the transformation application can be stored in the data layer (tabular data or RDF) or it can be delivered directly to the user
- The transformation definition (actual transformation) can also be retrieved from the *Transformations Catalogue* service or it can be provided with the user request. Additional parameters specific to the transformation invocation can also be provided (result type: RDF or tabular; transformation command, etc.)

The complete API documentation is provided in the Appendix here and at URL: https://datagraft.net/documentation.

2.5 Transformations Catalogue service

The support for persistence and lifecycle management of transformations definitions is the main responsibility of the transformations catalogue service. It supports the major CRUD operation, as well as search for pre-existing transformations.

Each transformation definition is accompanied by its metadata: title, description, ownership, inputs and outputs specifications and others.

[10] https://apiblueprint.org/
[11] https://github.com/danielgtaylor/aglio

In order to align with the metadata representation for datasets we extended the DCAT model with support for transformations (Figure 6). The new type of objects (*dapaas:Transformation*) shares a lot of metadata properties of datasets. However, there are some specific properties for which we use specific namespace.

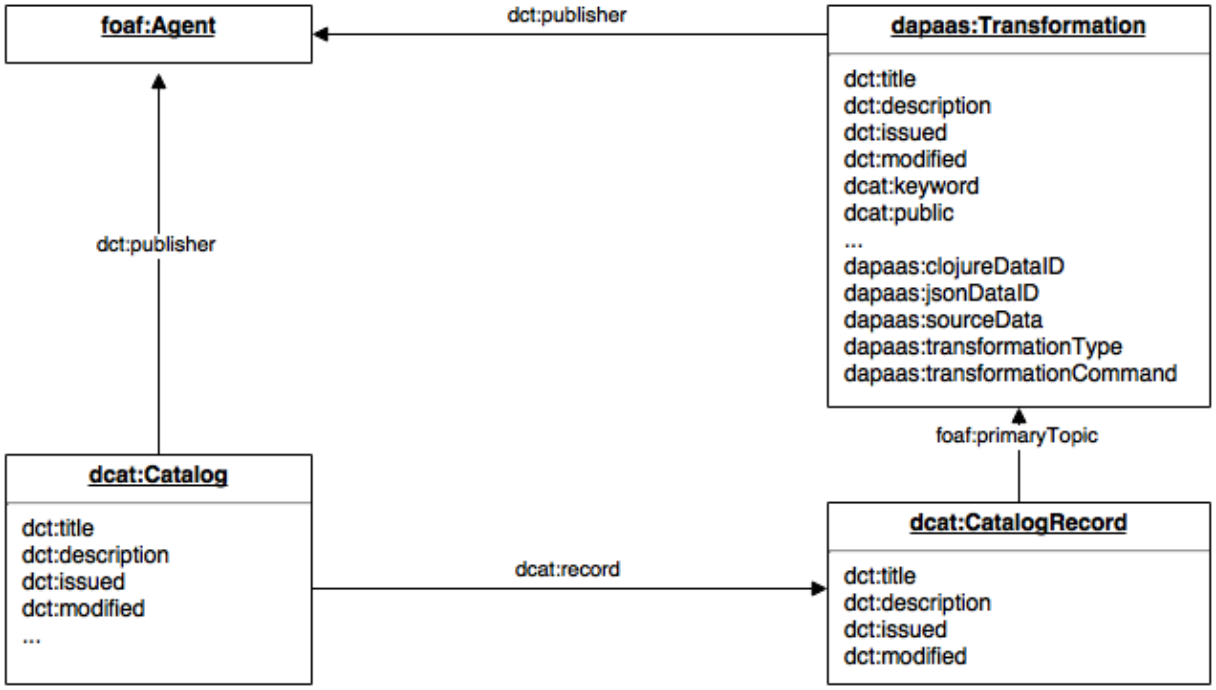


Figure 6. DCAT extension for transformations

This model represents data transformations as instances of *dapaas:Transformation* where the actual transformation definition is referred by *clojureDataID* and *jsonDataID*.

The new API supporting the transformations catalogue and management is aligned with the datasets catalogue APIs from WP1. It supports:

- New transformation registration;
- Existing transformation update;
- Transformations search and catalogue on metadata;
- Accessing the actual transformation;
- Transformations removal.

The complete API is provided in the **Appendix B**: API Documentation sections.

2.6 User Management & Access Control

The user management services are responsible for maintaining user profiles and authentication mechanisms. All functionalities and APIs from v1 of the platform are supported in the final prototype. The new features are support for API keys authentication – a flexible approach for managing access control to protected resources (datasets or transformations). The essence of this approach is that the user authenticates with API keys (instead of using main profile credentials) to the various services of the platform. A single account may have different API keys, which can be enabled, disabled or deleted and thus the access to certain resources can be granted, temporary disabled or discontinued. In case of API key loss or being compromised, a replacement key can easily be issued.

The API keys can also be temporal, expiring after certain period of time or permanent. The service APIs (except the management services) accept authentication only with API keys, i.e. data access services, transformation services, catalogues.

The following Figure 7 demonstrates the workflow for issuing and using the API keys. Depending on the request type, the key generated is temporal or permanent.

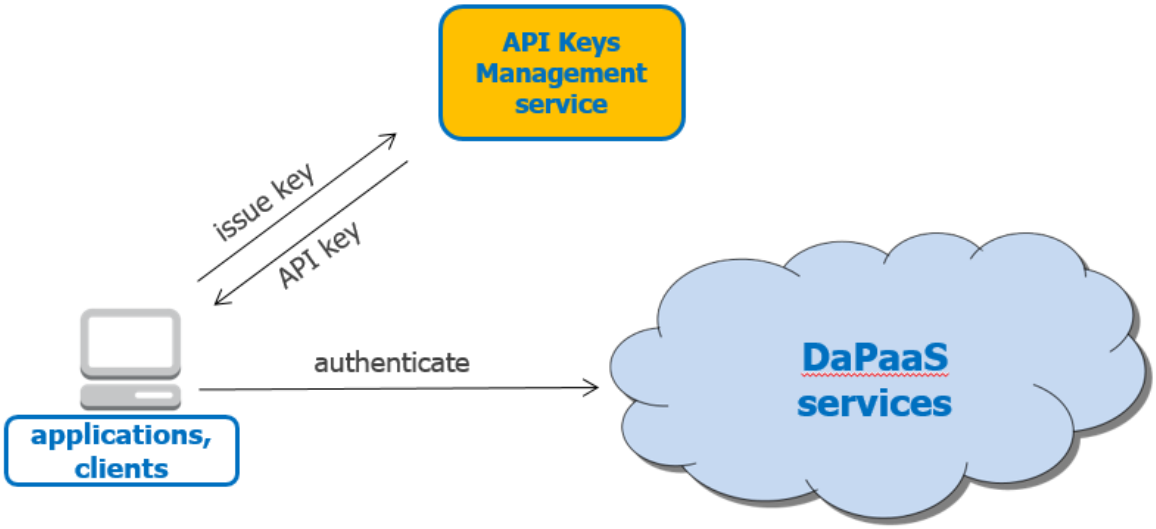


Figure 7. API keys usage

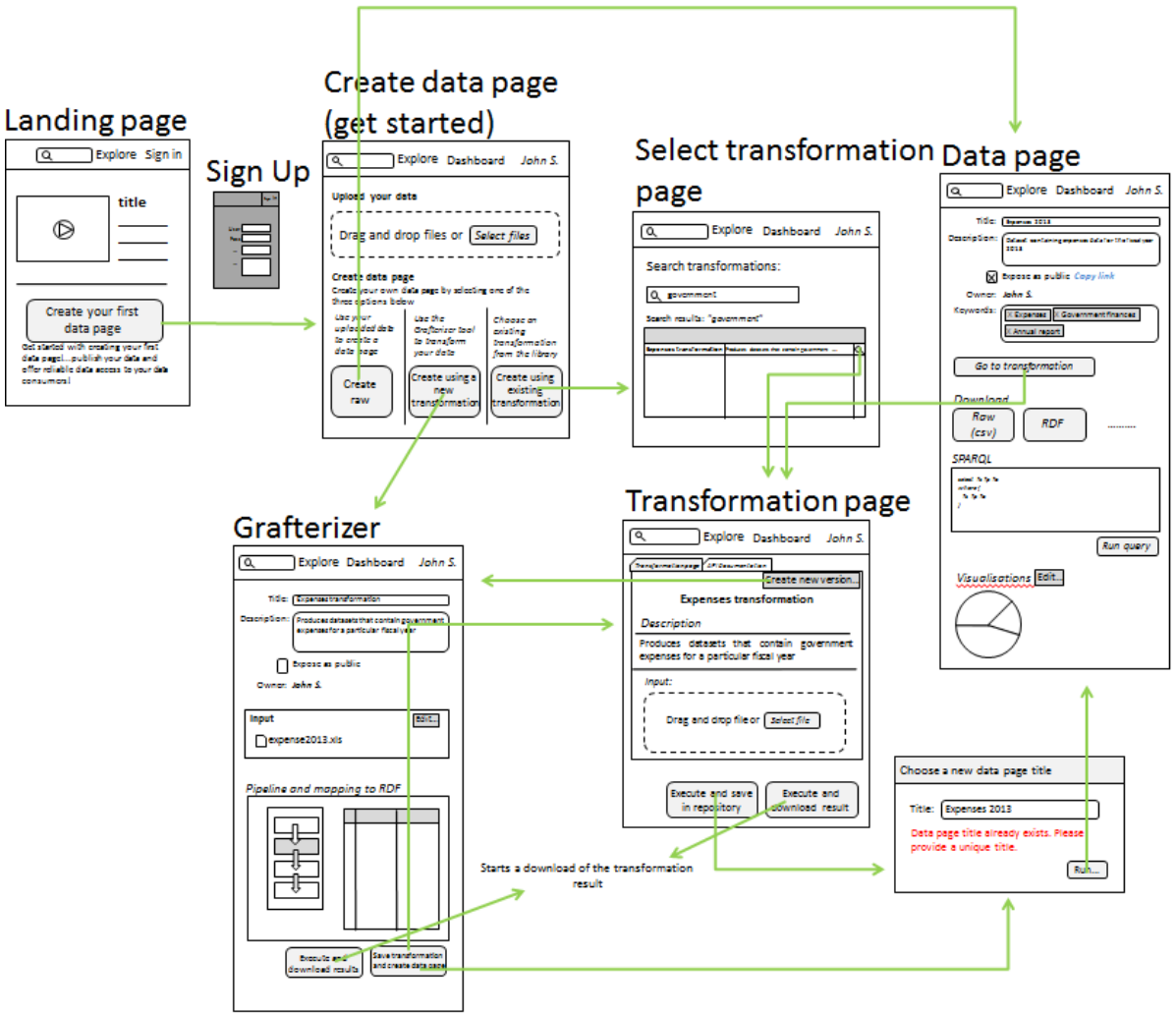
API documentation is included in the Appendixes here and at: https://datagraft.net/documentation.

3 Conclusion

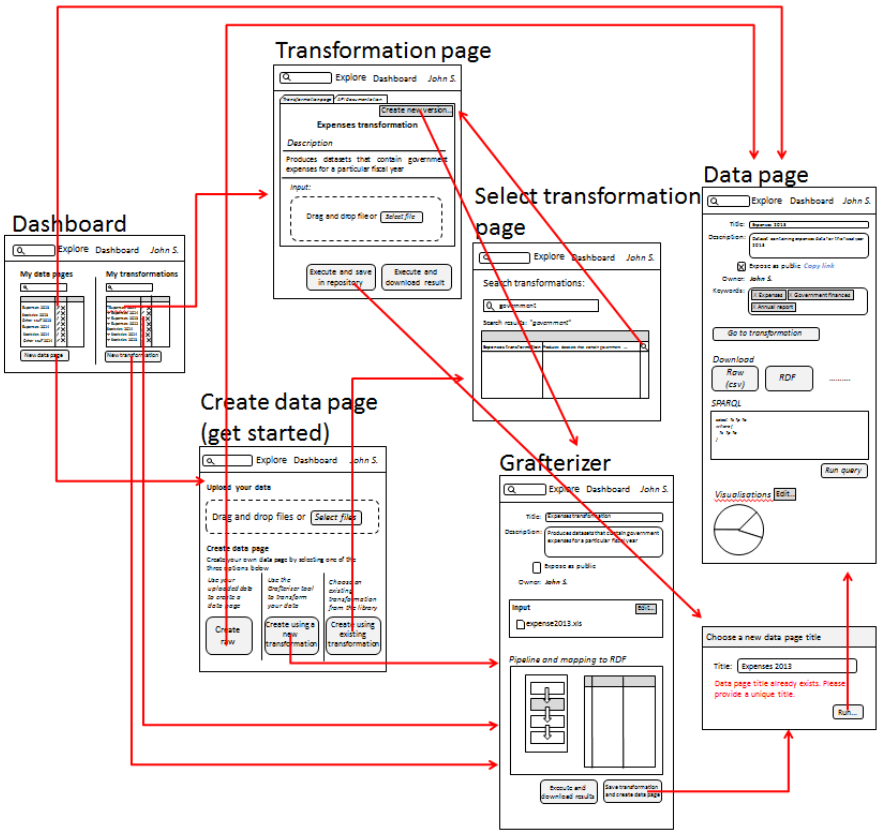
This document described the updated version of the platform-as-a-service (PaaS) layer of the DaPaaS platform, more recently known as DataGraft and deployed at DataGraft.net. The development of the platform was directed towards support of data cleaning and transformation applications based on Grafter. Significant improvement has been made in the user interface supporting Grafter transformations creation and editing. The underlying platform services have been extended to serve the new demands for transformations management and persistence. Towards the public offering of DataGraft, the access control and user authentication mechanisms have been extended to allow flexible access to the resources of the platform to external users or applications.

4 Appendix A: DataGraft Wireframes Design

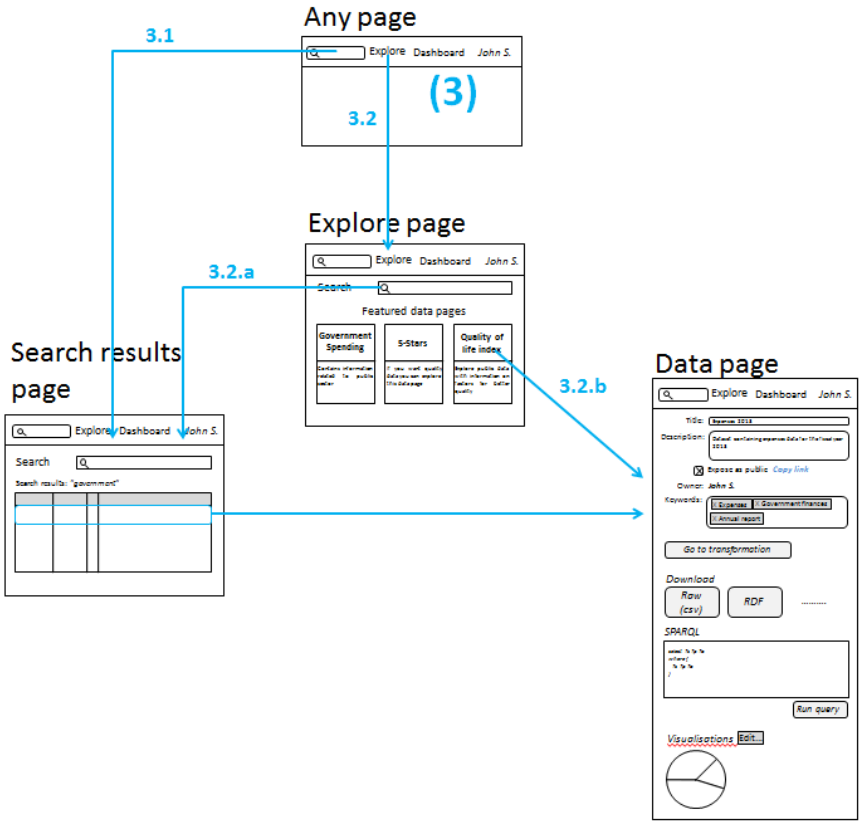
4.1 Flow 1: create first data page



4.2 Flow 2: dashboard actions



4.3 Flow 3: search and explore data



5 Appendix B: API Documentation

5.1 Transformations Management API

DESCRIPTION	METHOD	URL	INPUT	OUTPUT
Performs PIPE transformation with Graftwerk and loads the result in certain repository	POST	/grafter/transformation/pipe	Result placeholder specification: '**result-distribution**' - URI of existing distribution (catalogue) '**result-file**' - the name for the result file '**result-type**' - mime type of the result (*application/edn* or *text/csv*) Transformation spec: '**transformation-id**' - the id of the specific Grafter transformation to be used or ... '**transformation-code**' - (file attachment) inline transformation code '**command**' - the command in the transformation to be executed input specification: '**input-file**' - (file attachment) input file or '**input-distribution**' - URI of the distribution to be reused	Operation completion status (HTTP response code)
Performs GRAFT transformation with Graftwerk and loads the result in certain repository	POST	/grafter/transformation/graft	Result placeholder specification: '**result-distribution**' - target distribution to store the result '**repository-graph**' - optional graph in the repository Transformation spec: '**transformation-id**' - the id of the specific Grafter transformation to be used ... or ... '**transformation-code**' - (file attachment) inline transformation code '**command**' - the command in the transformation to be executed input specification: '**input-file**' - (file attachment) input file or '**input-distribution**' - URI of the distribution to be reused	Operation completion status (HTTP response code)
Performs data transformation with	POST	/grafter/transformatio	Transformation spec: '**transformation-id**' - the id of the	response from Graftwerk

Graftwerk and directly returns the result		n/preview	specific Grafter transformation to be used ... or ... '**transformation-type**' - 'graft' or 'pipe' '**command**' - the command in the transformation to be executed '**input-distribution**' - the input file from a distribution request body (multipart) - '**input-file**' - input file - '**transformation-code**' - (file) inline transformation code (alternative to transformation-id)	

5.2 Transformations Catalogue API

DESCRIPTION	METHOD	URL	INPUT	OUTPUT
List transformations	GET	/transformations/catalog	HTTP Headers: **Accept** - serialization format: • application/ld+json • application/rdf+xml • ... any RDF type **showShared** - shows shared transformations as well. Valid values (y|n), defaults to 'n'	List of transformations catalogue records using the DCAT vocabulary in RDF or JSON-LD
Search for transformations (on metadata)	GET	/transformations/search?q=...	HTTP Headers: **Accept** - serialization format **showShared** - shows shared transformations as well. Valid values (y|n), defaults to 'n' **q** - plain text query	List of transformations catalogue records in DCAT
Get transformation description	GET	/transformations	HTTP headers: **transformation-id** - URI of the transformation, taken from the catalogue **Accept** - serialization format (same as above)	Complete application description using the DCAT vocabulary
Create a new transformation	POST	/transformations	**Content-Type** - "multipart/mixed", "multipart/form-data" Attachments: **meta** - transformation metadata as RDF or JSON-LD. Use 'Content-type' to determine the metadata format **tr-clojure** - the code of the transformation as Clojure **tr-json** - the JSON serialization of the	URI of the new transformation in the format: {"@id":"http://dapaas.eu/..."}

			transformation. The transformation code parts are optional and might be managed separately	
Update a transformation	PUT	/transformations	**Content-Type** - "multipart/mixed", "multipart/form-data" **transformation-id** - URI of the transformation (unless metadata is provided!) Attachments: **meta** - transformation metadata as RDF or JSON-LD. Use 'Content-type' to determine the metadata format **tr-clojure** - the code of the transformation as Clojure **tr-json** - the JSON serialization of the transformation All parts here are optional so they can be managed separately	HTTP result code
Delete a transformation	DELETE	/transformations	**transformation-id** - URI of the transformation to be removed	HTTP result code
Retrieve transformation code as Clojure code	GET	/transformations/code/clojure	**transformation-id** - uri of the transformation description	Clojure code (Content-type:application/x-clojure)
Retrieve transformation code as JSON code	GET	/transformations/code/json	**transformation-id** - uri of the transformation description	Json code (Content-type:application/json)
Delete the transformation clojure code	DELETE	/transformations/code/clojure	**transformation-id** - uri of the transformation description containing the code	HTTP result code
Delete the transformation JSON code	DELETE	/transformations/code/json	**transformation-id** - uri of the transformation description containing the code	HTTP result code

5.3 API Keys Management

DESCRIPTION	METHOD	URL	INPUT	OUTPUT

List the API keys for the authenticated user	GET	/api_keys	-	List of API keys descriptions in JSON:
				`[{"api_key":<api_key>,`
				`"enabled" : true },`
				`... ]`

Generate a new API key	POST	/api_keys	-	API key and secret in JSON format.

				{"api_key": <api_key>, "secret" : <secret>} Note that this is the only time the service provides the key secret
Generate a new temporary API key	POST	/api_keys/temporary	-	API key and secret in JSON format. {"api_key": <api_key>, "secret" : <secret>} The key expires after 24 hours and is automatically removed
Enable API key	PUT	/api_keys/<api_key>/enable	The API key as part of the request URL	HTTP result code
Disable API key	PUT	/api_keys/<api_key>/disable	The API key as part of the request URL	HTTP result code
Delete API key	DELETE	/api_keys/<api_key>	The API key as part of the request URL	HTTP result code