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 3.2

Cross platform data delivery framework

Date:	31 July 2014
Author(s):	Momchill Zarev
Dissemination level:	PU
WP:	3
Version:	1.0

Document metadata

Quality assurors and contributors

Quality assuror(s)	Bill Roberts, Amanda Smith, Alex Simov
Contributor(s)	Dumitru Roman

Version history

Version	Date	Description
0.1	July 01, 2014	Initial structure
0.2	July 14, 2014	Comments and improvements
0.3	July 20, 2014	Revised version with screenshots and descriptions
0.4	July 24, 2014	Internal review comments
0.5	July 29, 2014	Addressed internal review comments
1.0	July 31, 2014	Final improvements

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 and linked 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 preliminary prototype for the graphical user interface (GUI) of the DaPaaS Platform. The deliverable introduces the current prototype through a set of GUIs for various capabilities of the platform, with a particular focus on:

- Web interface to access data using various visualization mechanisms researched in D3.1;
- Web interface to manage publisher's data and datasets (for data publishers and application developers);
- Cross platform UI (Web-based) that can be accessed from both desktop browser and mobile devices;
- Analytics information of customer behaviour (track data views, time spent on the site, bounce rate, unique visitors, etc);
- Contact and feedback forms.

This deliverable presents the basic framework/prototype for the DaPaaS Platform GUI and does not include actual data at this time. The UI will be further refined during the course of the project.

Table of Contents

List of Figures

1 Introduction

This document represents supporting documentation for Deliverable D3.2 (Nature: Prototype) and addresses GUI requirements outlined in Deliverable D3.1[1]. The goals of this deliverable are to provide:

- A prototype of GUIs for different roles considered in DaPaaS (in particular end user data consumers and data publishers). The prototype is based on DaPaaS platform requirements outlined in Deliverable D3.1 and includes GUIs for uploading, managing and accessing data from the platform.
- Visualization widgets. The previous deliverable outlines major frameworks for data visualization and mandatory widgets. This prototype implements visualization widgets based on D3 JS library and DaPaaS requirements.
- Cross platform data delivery.
- Analytics information of customer behaviour (track data views, time spent on the site, bounce rate, unique visitors, etc).
- Contact and feedback forms.

Deliverable D3.1 outlined a set of requirements for visualization components. The tables below summarize how the current version of the prototype as of M9 addresses the requirements.

Table 1: Visualization requirements for DaPaaS Key Roles (D3.1) and how they are addressed in the current version of the prototype

ID	Name	Brief description of how the current version of the prototype addresses the requirements.
IO-06	UI for Instance Operator	The Instance Operator user interface is not part of this prototype and shall be addressed later in the project when the back-end capabilities for the instance operator are made available.
DP-12	UI for Data Publisher	The Data Publisher graphical user interfaces are addressed in this prototype and different screens are described in this document, Sections 2.5, 2.6, 2.7, 2.8.
AD-07	UI for Application Developer	The Application Developer interfaces include all data publishing screens described in Sections 2.5, 2.6, 2.7, 2.8, and additional screens for application management (to be addressed in the next phase).
EU-04	Mobile and desktop GUI access	The End-Users Data Consumers screens are addressed in the prototype and described in this document, Sections 2.1, 2.3, 2.4, 2.5, 2.8, 2.9.

[1] http://project.dapaas.eu/dapaas-reports/deliverable-31-analysis-requirements-design-architecture-specification-for-the-data-access-framework

Table 2: Description of requirements for Visualization Types (D3.1) and how they are addressed in the current version of the prototype

ID	Name	Brief description of how the current version of the prototype addresses the requirements.
UI-01	Cross platforms and mobile support	As a result of research and analysis in Deliverable D3.1, UI is built based on D3 JS library for visualization, and jQuery JS library for communication and overall user interaction. Both libraries support cross platforms with extensive mobile support.
UI-02	Support for bar and column charts	Bar chart widget is implemented and described in Section 2.1.2.
UI-03	Support for line and area charts	Line chart widget is implemented and described in Section 2.1.3.
UI-04	Support for pie charts	The pie chart widget is implemented and described in Section 2.1.6.
UI-05	Support for scatter plots and bubble charts	Scatter plot widget is implemented.
UI-06	Support for tables	Table widget is implemented and described in Section 2.1.1.

Deliverable D3.1 outlined a set of libraries for visualization components (D3 and jqPlot) and overall UX development (jQuery and Dojo). Based on that evaluation, the prototype uses D3 JS library to implement visualization widgets and jQuery library for overall UI implementation and communication with backend.

This rest of the document provides a set of screenshots on how the implemented GUIs look like (Section 2), describes installation instructions for accessing and deploying the code for the prototype (Section 3), and outlines the focus for the next implementation phase (Section 4).

2 UI Screens and Workflows

2.1 Data-driven Portals

Data-driven portals are the main interaction point for the data uploaded to the DaPaaS Platform, where the publisher specifies how this data should be visualized. A data-driven portal is configured / generated by the data publisher and can contain different visualization widgets, as shown below.

2.1.1 Tabular View Widget

Tabular view displays data as a table with pagination, option to search/filter table content and browse content. Example for tabular view is displaying safety cameras in London:

Easting	Northing	Speed limit	Location	Fixed Camera Type
547752	186619	30mph	- Valence Avenue - Becontree Avenue	Red Light
545310	185088	30mph	A124 Longbridge Road - Upney Lane	Red Light
549067	184517	20mph	A1240 Heathway - Church Elm Lane	Red Light
548860	183545	30mph	A1306 Ripple Road - Merrielands Crescent	Red Light
548987	185794	30mph	A1240 Heathway - Oxlow Lane	Red Light
547002	183623	40mph	A13 Ripple Road - Renwick Road	Red Light
550046	184204	30mph	B178 Ballards Road south of Boleyn Gardens	Speed Camera
550677	184245	30mph	Rainham Road South By Canberra Crescent E B (Omni) E+W	Speed Camera
550677	184245	30mph	A1112 Rainham Road South jw Canberra Crescent	Speed Camera
549292	183517	30mph	A1306 New Road By South Close	Speed Camera
543476	184240	50mph	A406 North Circular Road near A124 Slip Road	Speed Camera
548614	186874	30mph	Green Lane By Tenterden Road eb	Speed Camera
545813	185445	30mph	Longbridge Road By Malvern Drive North-East Bound (Dual System	Speed Camera
545056	183264	50mph	A13 Alfreds Way By Movers Lane Easbound Off Slip To Movers □E14	Average Speed Camera
545526	183414	50mph	A13 Alfreds Way By Movers Lane Easbound On Slip From Movers (E	Average Speed Camera
545660	183457	50mph	A13 Alfreds Way By By Saxham Road E16a	Average Speed Camera
547735	183584	50mph	A13 Ripple Road By Gale Street	Average Speed Camera

Figure 1: Table view

2.1.2 Bar Chart Widget

Bar charts can be used to compare data. In the example below, a bar chart has been used to show number of bicycle hires throughout London in the past 6 months:

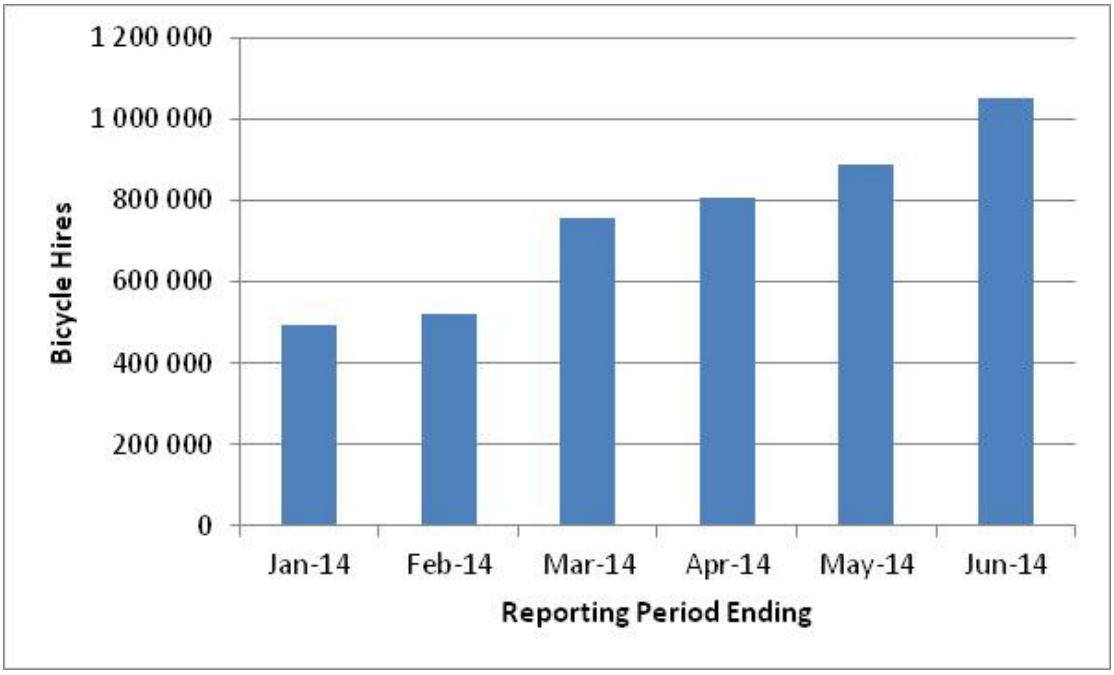


Figure 2: Bar chart widget

2.1.3 Line Chart Widget

Line charts can be used to monitor trends or compare data. In the example below, the line chart shows public transport journeys in London:

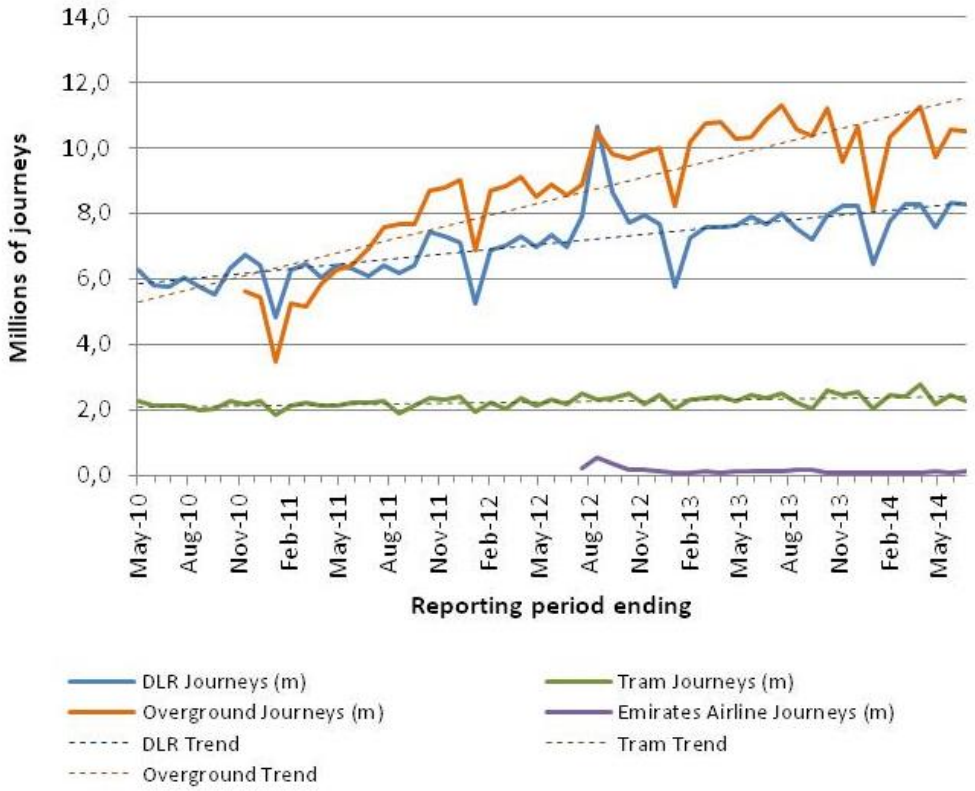


Figure 3: Line chart widget

2.1.4 Geodata Widget

This widget is used to display geo data on map, and supports different kind of map views, including integration with google maps or open street maps. It allows user interaction with map, zooming, panning, and click and explores locations.

Figure 4: Map view #1

Figure 5: Map view #2

2.1.5 Timeline Widget

Timeline is useful to display events happing during time and relationship between these events. This kind of widget can show historical information, milestones during long term events and similar data.

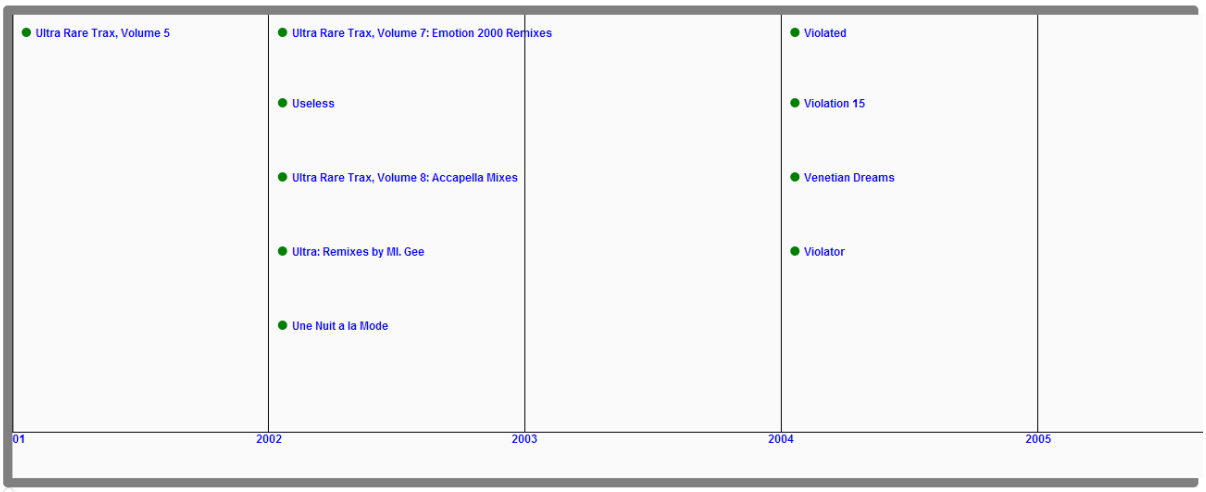


Figure 6: Timeline widget

2.1.6 Pie Chart Widget

A pie chart is used to compare relatively different components, usually of the same type. This pie chart example demonstrates a visualization of buses by type in London.

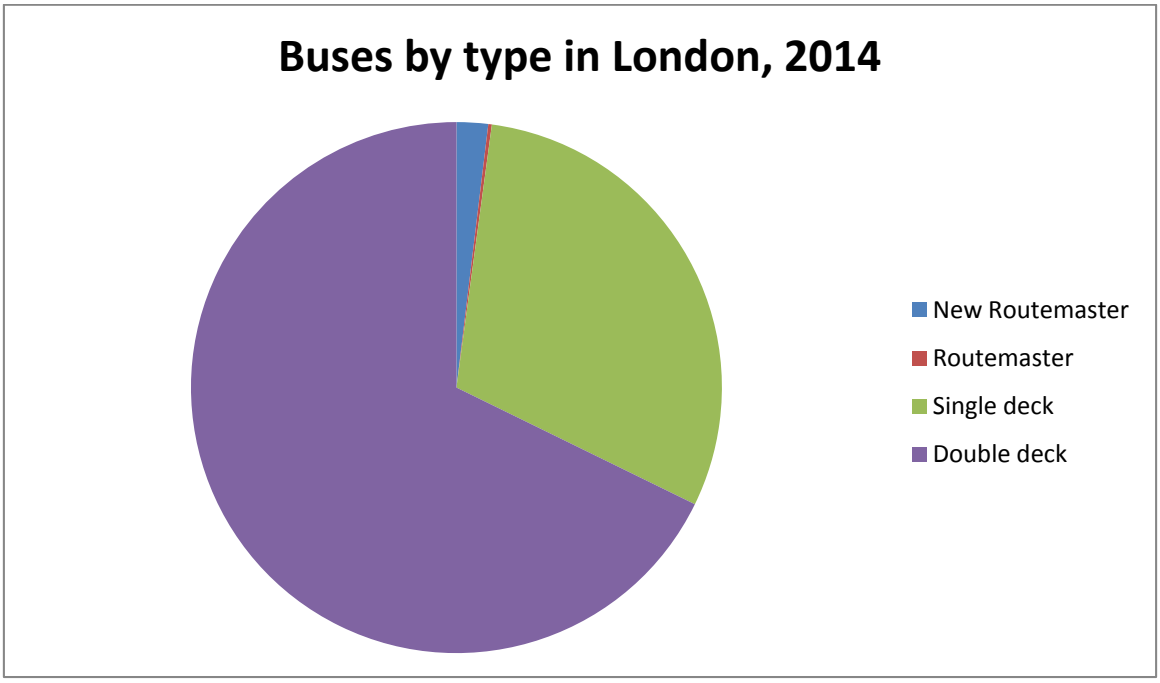
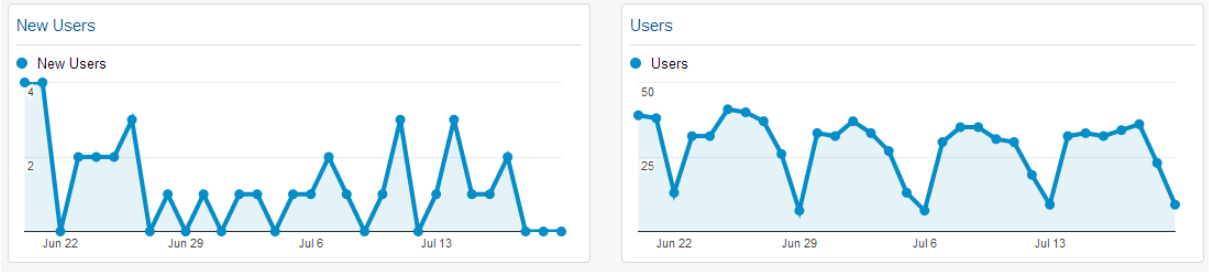


Figure 7: Pie Chart

2.2 Platform Analytics

The prototype provides means to track user views (on datasets), bounce rate, time spent on the site, etc. Tracking is implemented using Google analytics.



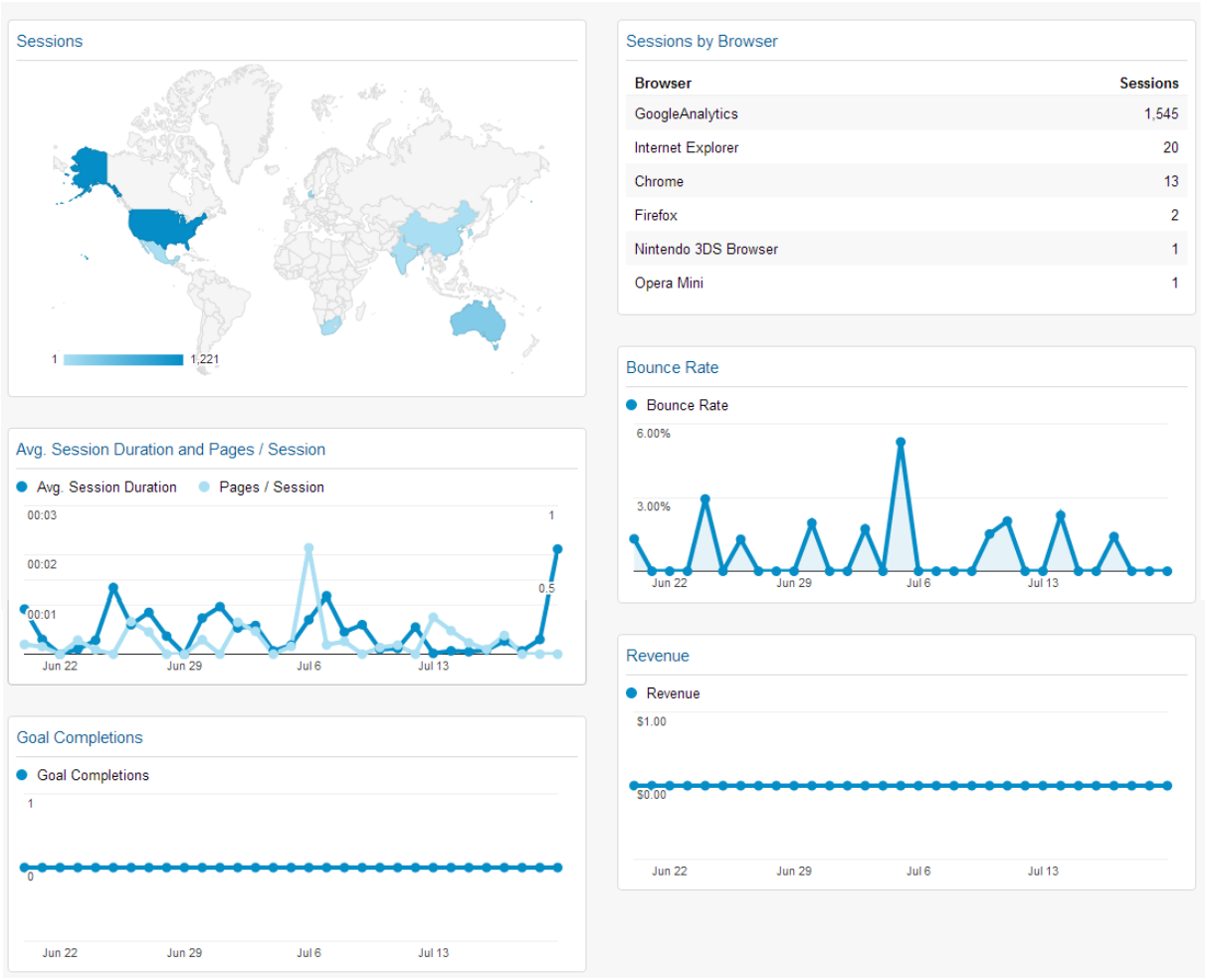
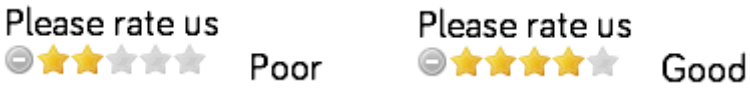


Figure 8: Analytics

2.3 Ratings

The prototype provides users with ability to rate datasets, by specifying one (lowest) to five (highest) star rating. Based on user rating, a list of the highest rated datasets available on the platform can be shown, therefore boosting better data content to the users.



2.4 Browse Public Datasets

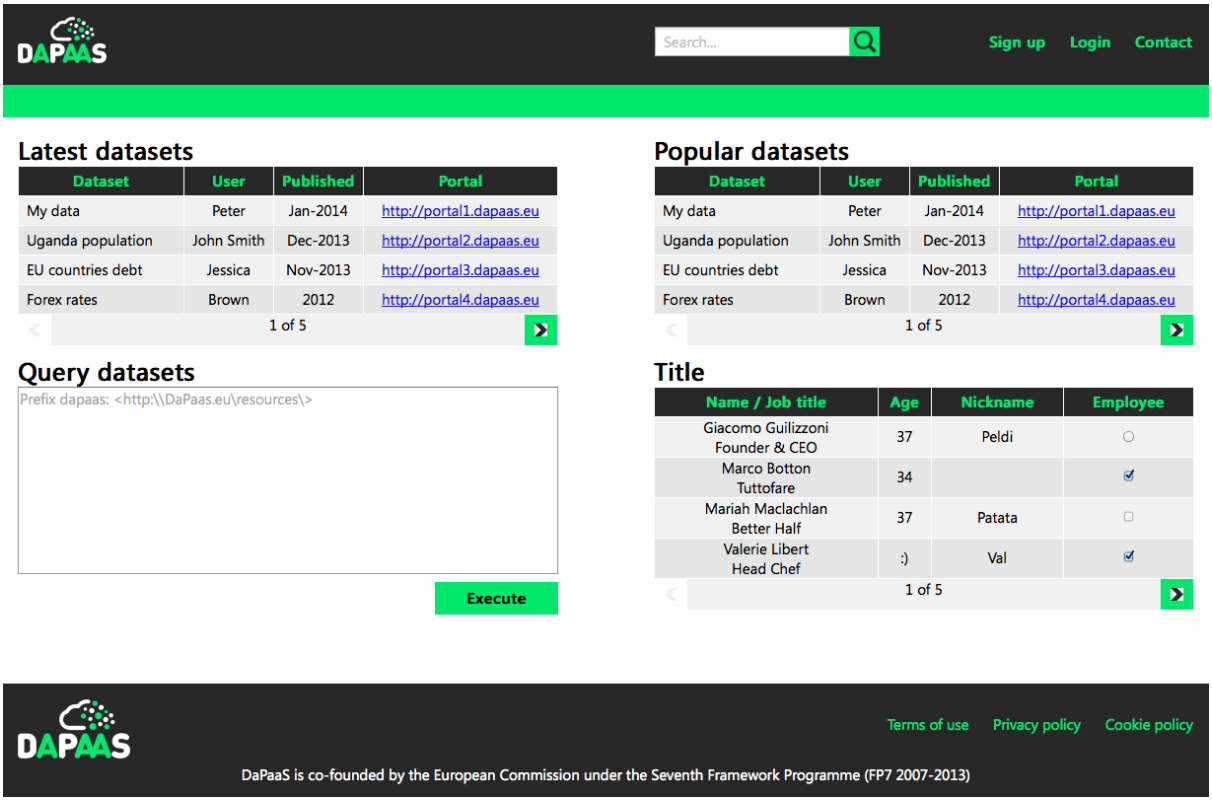


Figure 9: DaPaaS Platform homepage

The capability for a user to browse public data is on the homepage of the DaPaaS public portal. This screen provides the following functionality:

- View most popular / highest ranked datasets. Browse the list of most popular datasets, view data driven portals.
- View latest uploaded public datasets. Browse the list of latest datasets, view data driven portals.
- Search datasets based on keywords.

2.5 Sign-in / Sign-up

To be able to use most of the functionality of the DaPaaS platform, users need to be registered. The key roles supported by DaPaaS are described in Deliverable 3.1 and they define access level of the registered user, where next level accumulates permissions and functionalities of previous levels.

The first key role is End-User Data Consumer (Data Explorer) which can browse data on the platform, subscribe for notifications, and provide ratings. The next key role is Data Publisher which extends Data Consumer role with ability to publish data. Application developer has ability to publish applications beyond data publisher functionalities.

The sign-in/sign-up process supports several methods for registration and login using different authentication providers:

- Sign-in/Sign-up using Facebook account
- Sign-in/Sign-up using Google+ account
- Sign-in/Sign-up using username and password

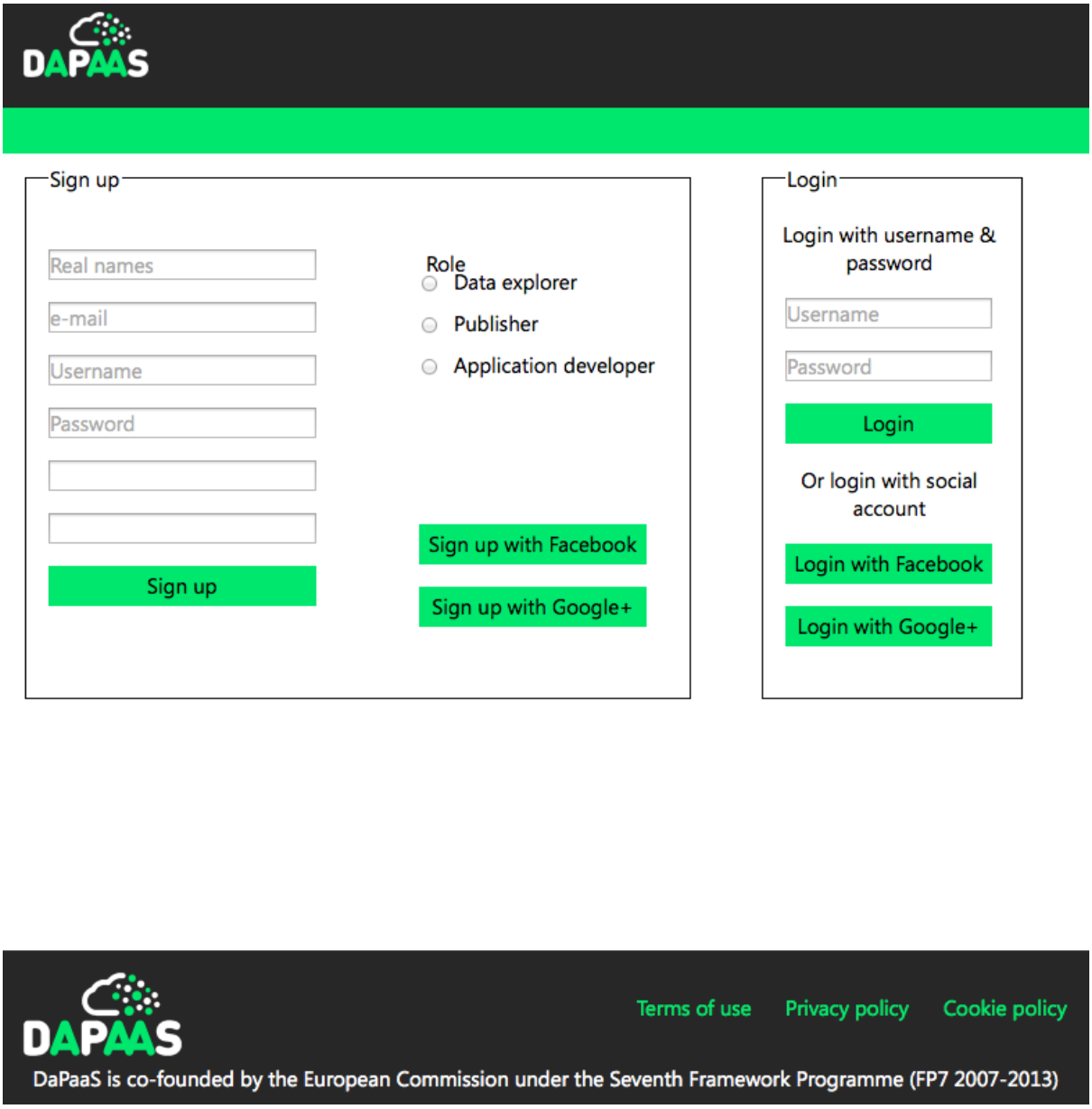


Figure 10: Login Screen

2.6 Manage Assets

After login, data publishers are redirected to manage their assets screen, where private and shared datasets are shown. This screen provides the following functionality:

- Browse their own and shared datasets
- Search datasets
- Filter datasets based on popularity or latest
- Edit the metadata for datasets and modify data-driven portal properties (widgets)
- Delete datasets
- Upload/propose modifications to shared datasets
- Create new datasets and upload data

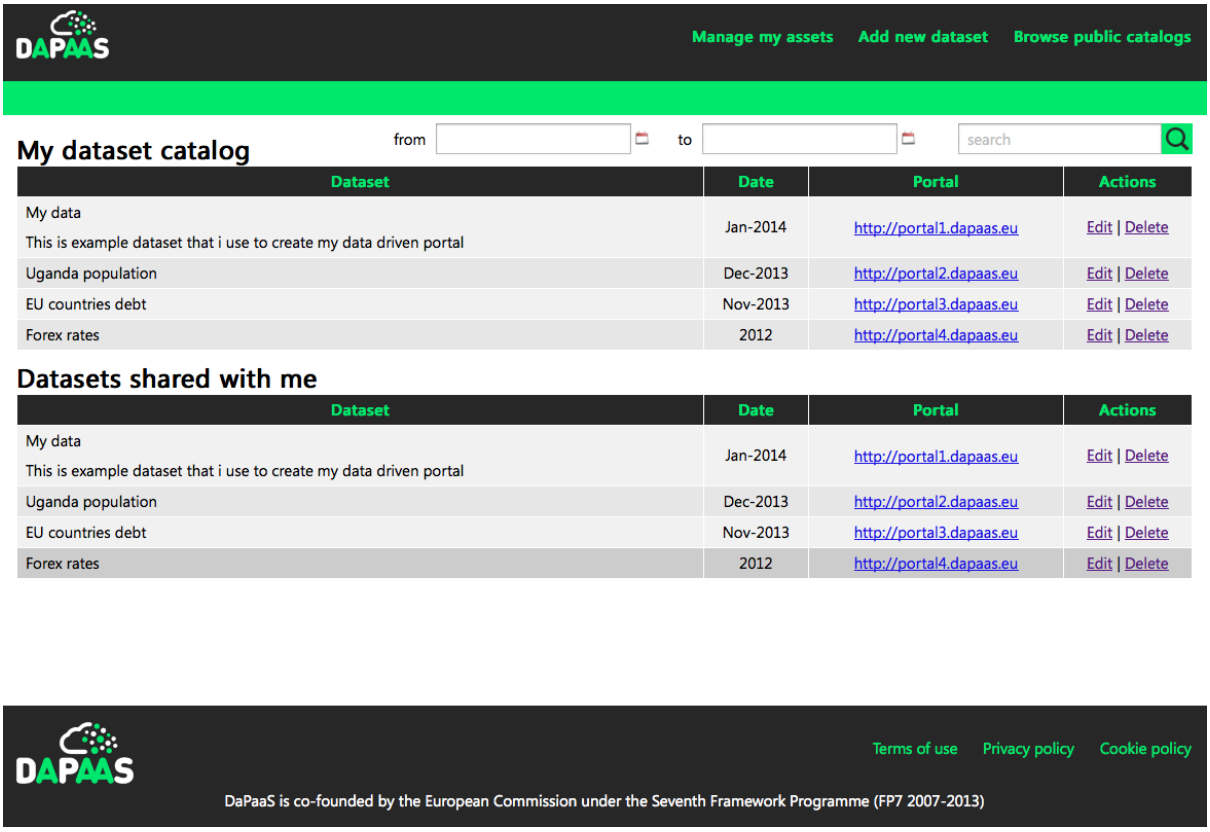


Figure 11: Manage my assets

2.7 Publish New Data

The data publishing process is implemented using wizard UI, where the publisher is lead one step after another to import their data into the platform. The process involves several steps:

- Step 1: Provide metadata. Metadata contains of information about the new dataset. This includes properties such as dataset name, description, source of data, original publisher, etc.
- Step 2: Upload data files and optional mapping. In the final version the system shall support processing data from the following files: CSV, XLS, XLSX, PDF, and RDF.
- Step 3: Explore data – search, browse, etc.
- Step 4: Interlink data with other data already available on DaPaaS platform.
- Step 5: Dataset permissions. The publisher can specify whether the dataset is publicly available or private data accessible to only predefined users.
- Step 6: Create data driven portal for the given dataset. Publishers can select widgets that will be included in the portal and configure each widget to use excerpt from datasets.

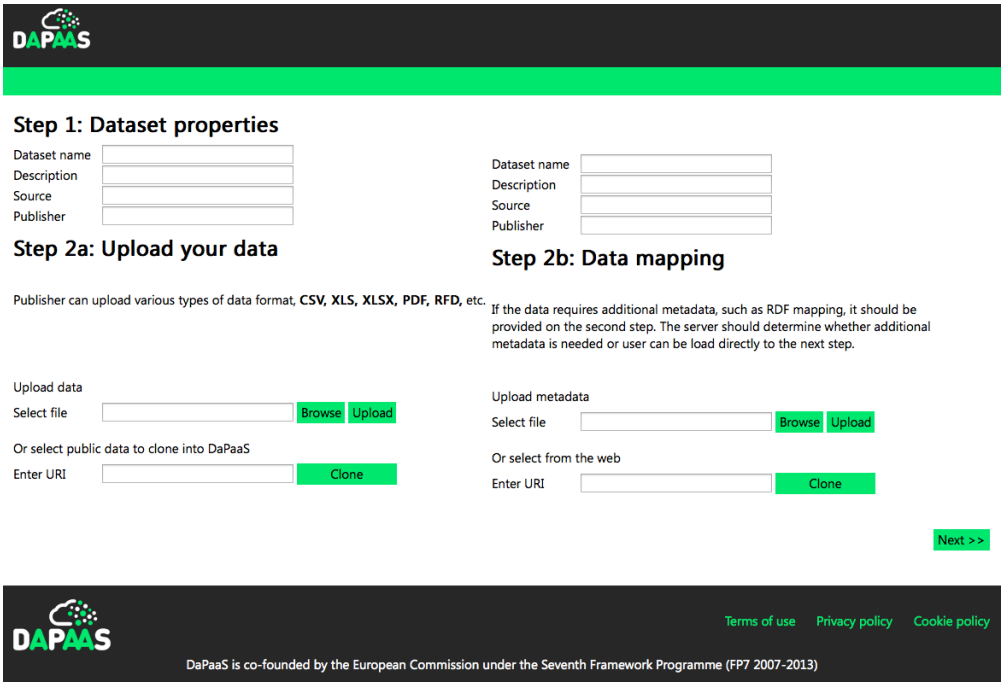


Figure 12: Add new dataset wizard

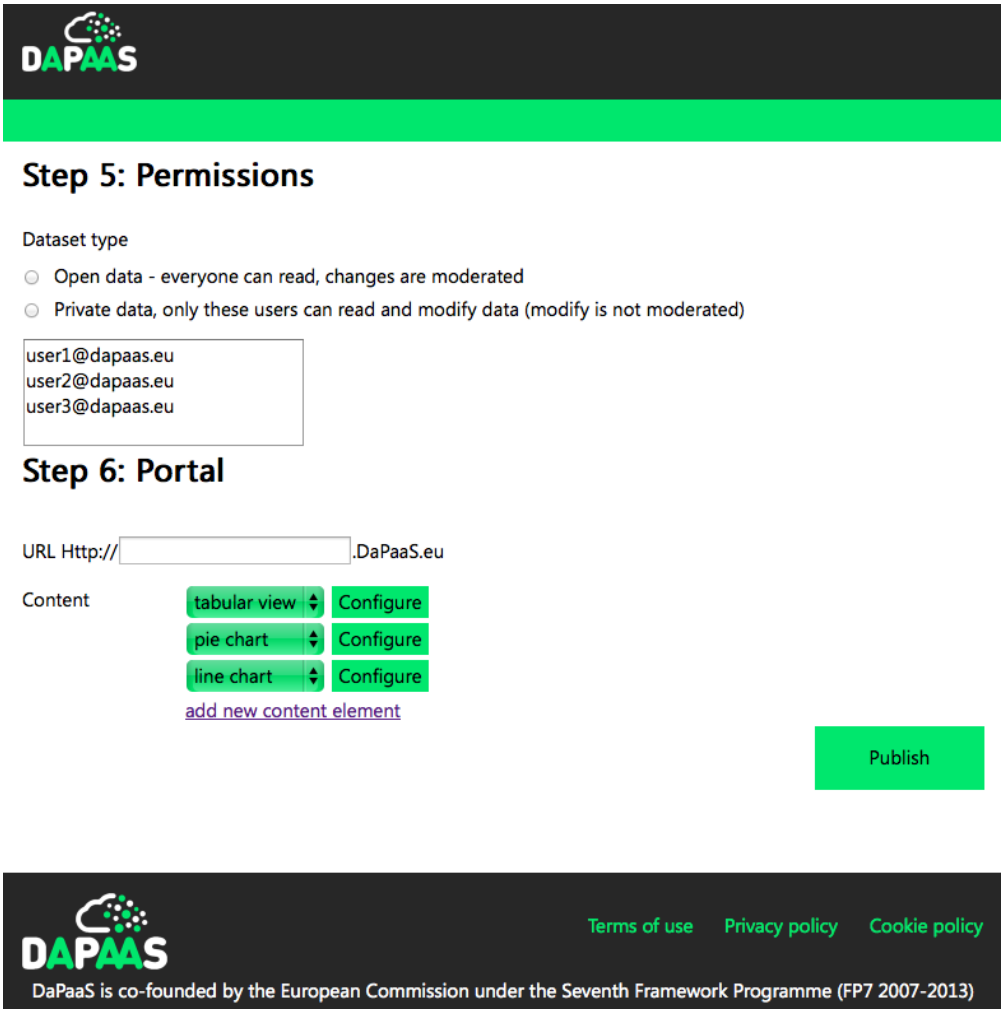


Figure 13: Dataset permissions settings

2.8 Contact Form

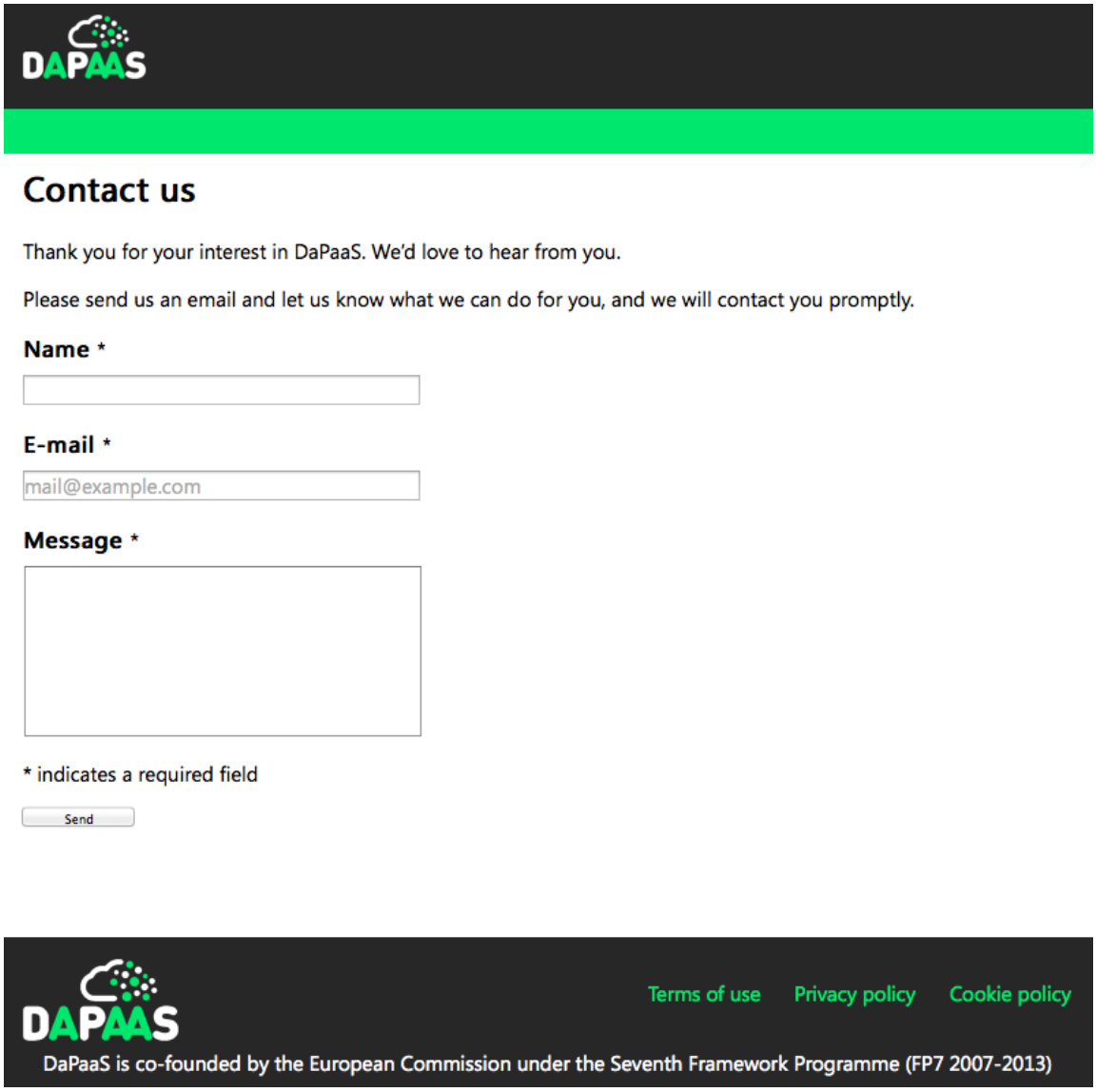
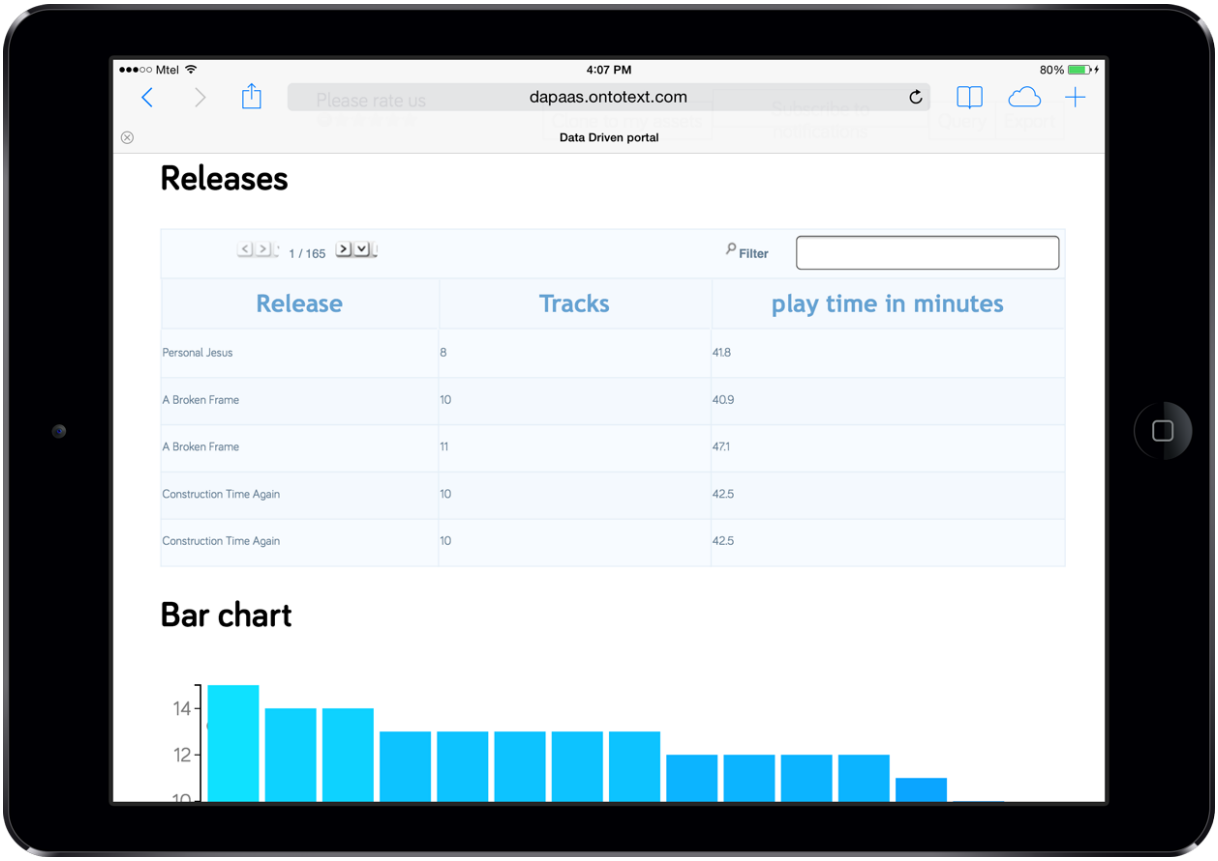


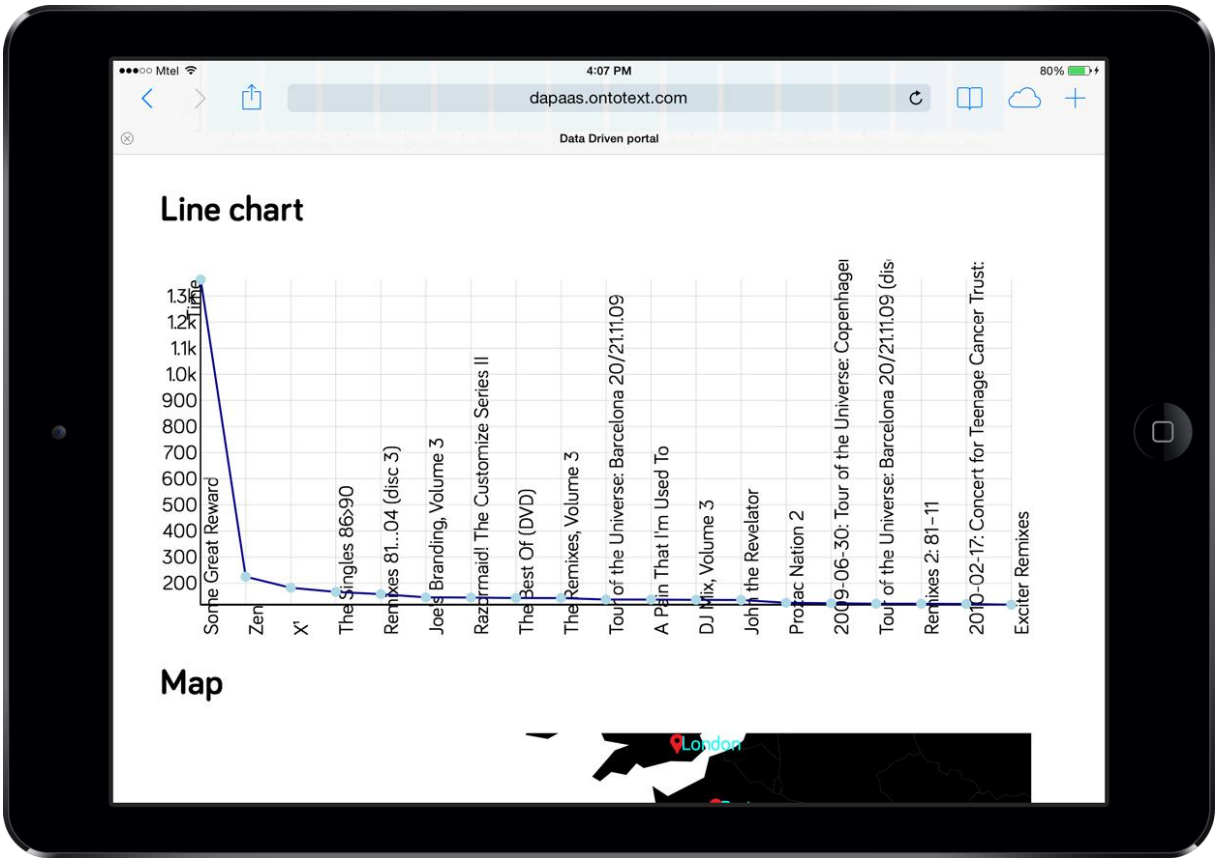
Figure 14: Contact form

Contact form is a standard way to receive user feedback. The platform provides means to manage user requests for contact and respond back.

2.9 Cross Platform Delivery

Data browsing and data driven portals are viewable across screens, supporting different resolutions, desktop browsers, tablets and mobile browsers, as depicted below.





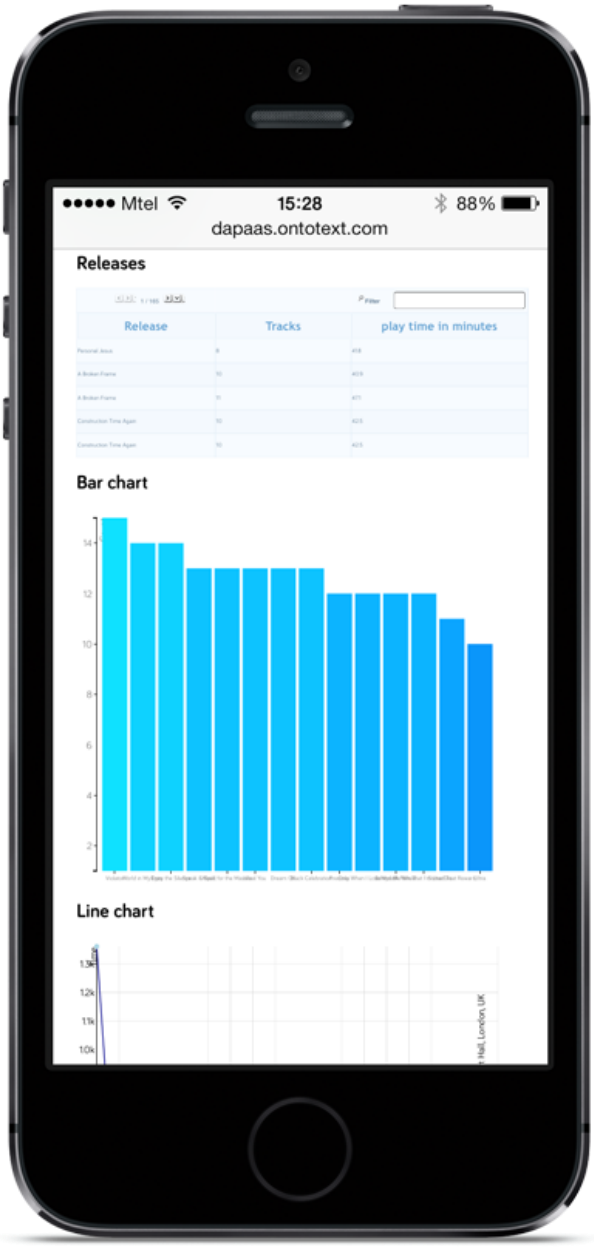


Figure 15: Cross-platform access to the DaPaaS Platform via graphical interfaces.

3 Installation Instructions

The UX layer runs as standard Web application and is distributed as a Web application ARchive (WAR) file, which is uploaded on compliant Web application servers. The following instruction assumes Tomcat Web application server running on Linux environments.

3.1 Prerequisites

- OS supporting Java and Tomcat container
- Java runtime environment 1.6+
- Tomcat application server 7+

3.2 Obtaining the Source Code

The source code can be obtained from DaPaaS project GITHUB repository located at:

`https://github.com/dapaas/ui-prototype`

The following command retrieves source code from the repository:

```
# git clone https://github.com/dapaas/ui-prototype
```

3.3 Compiling the Source Code

The source code can be compiled using ant build system, or any IDE that support building WAR, such as Eclipse or similar.

To build package from ant, navigate to the project source and execute from command line:

```
# ant war
```

To build from Eclipse, import project into eclipse and chose auto-build project, or manually select from project menu “Build project”.

3.4 Configuration

Configuration options are located in `<source>/WebContent/WEB_INF/web.xml`. This xml file contains several context properties as described below:

- googlePlusClientId – application ID used for Google+ authentication
- googlePlusClientSecret – API key used for Google+ authentication
- additionalPermissions – permissions for accessing user profile. Should contain at least access to profile and email
- FBClientId – application ID used for Facebook authentication
- FBClientSecret – API key used for Facebook authentication

3.5 Deployment

The war file with compiled code and configuration should be deployed using standard application contained means. For Tomcat this means using Tomcat manager to deploy war file, or if enabled hot deploy, copy war file to tomcat/webapps/demo folder.

4 Summary and Outlook

This document provided an overview of the current prototype of UX Layer of the DaPaaS platform, outlined a set of screens and workflows for the UX Layer from the perspective of users, and the visualization widgets the platform supports, as of M9. Following the technology evaluation during prototype development, the following remarks are to be considered for the next version of the UX Layer implementation, with a particular focus on the Platform Layer:

- The UI should provide means for visual data exploration (for graph data in RDF). After uploading and RDF-izing data the user should be able to browse/explore RDF data with visual tools. The followings are tools-implementations currently considered for this task: Visual RDF[2], D3 graph map[3], LOD Live[4], Payola[5].
- In order to allow users to work with UI without technical knowledge of SPARQL queries, it would be beneficial to provide visual query builder as alternative of direct SPARQL query writing.

Besides the visual support for RDF data exploration/querying, GUIs for DaPaaS components will be built once the components become available. Specifically, GUI components addressing requirements AD07 and IO-06 will be considered.

[2] http://graves.cl/visualRDF/
[3] http://nylen.tv/d3-process-map/graph.php?dataset=les-mis
[4] http://en.lodlive.it/
[5] https://payola.github.io/Payola/