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

Data- and Platform-as-a-Service

Deliverable 3.4

Native apps building platform

Date:	30 July 2015
Author(s):	Momchill Zarev
Dissemination level:	PU
WP:	3
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	July 22, 2015	Initial structure
0.2	July 23, 2015	Content and preparation for internal review
0.3	July 28, 2015	QA from Rick Moynihan and Amanda Smith
0.4	July 29, 2015	Amended version after QA
1.0	July 30. 2015	Adjustments and finalization

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. Within the DaPaaS research project we have developed this platform, more recently known as DataGraft (http://datagraft.net).

This deliverable focuses on the prototype for the native mobile apps that visualize data made available via the DataGraft Platform. The deliverable introduces the current prototype through a set of GUIs for various capabilities of mobile data visualization, with a particular focus on:

- Multi-platform support (native apps for major mobile platforms: iOS and Android);
- Geolocation data visualization (map);
- Apps building blocks (visualization widgets) based on research in D3.1;
- Analytics information of customer behaviour (track data views, time spent on the app, etc).

This document starts with a description of the native apps capabilities used by mobile apps to visualize data made available via DataGraft, and then continues with other UI screens and workflows related to the native app building process. The last section of document describes installation instructions for running the prototype.

Table of Contents

List of Figures

1 Introduction

This document represents supporting documentation for Deliverable D3.4 (Nature: Prototype) and addresses GUI requirements outlined in Deliverable D3.1[1]. The goals of this deliverable are to provide:

- Native apps for major mobile platforms: iOS and Android;
- Visualization widgets. The previous deliverables (D3.2 and D3.3) provide implementation of Web widgets. This deliverable focuses on mobile visualization mechanism to display and interact with data using native apps;
- Analytics information of customer behaviour (track data views, time spent on the app, etc);

Deliverable D3.1 outlined a set of requirements for visualization components, which reflect directly on apps UI. The tables below summarize how the current version of the prototype as of M21 addresses the requirements.

Table 1: 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	This deliverable targets iOS and Android native apps
UI-02	Support for bar and column charts	Bar chart widget implemented and described in section 2.1.4.
UI-03	Support for line and area charts	Line chart widget implemented and described in section 2.1.7.
UI-04	Support for pie charts	The pie chart widget is implemented and described in section 2.1.6.

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

2 UI Screens and Workflows

2.1 UI

Native apps provide more user friendly UI for the interaction for the data made available via the DataGraft Platform; they are more responsive and generally they provide better user experience. The prototype developed for this document includes the following UI elements:

- Better gestures control for easy navigation and data exploration;
- Side menus to access additional options and screens;
- Sliders for building filters, for easy data exploration / filtering;
- Different visualization widgets, including maps, bar charts, pie chart, etc.;
- Support for offline data exploration.

2.1.1 Side menus

Side menus are accessible from the 'hamburger' symbol on the top left side of the screen. Side menu provide additional navigation options without cluttering the valuable real estate of the screen.

In the example below, the Saltlux's PLUQI (Personalised and Localised Urban Quality Index), side menu gives option to switch between different settings in the app, such as main PLUQI index, rank of Cities in SK, or compare data for different cities.

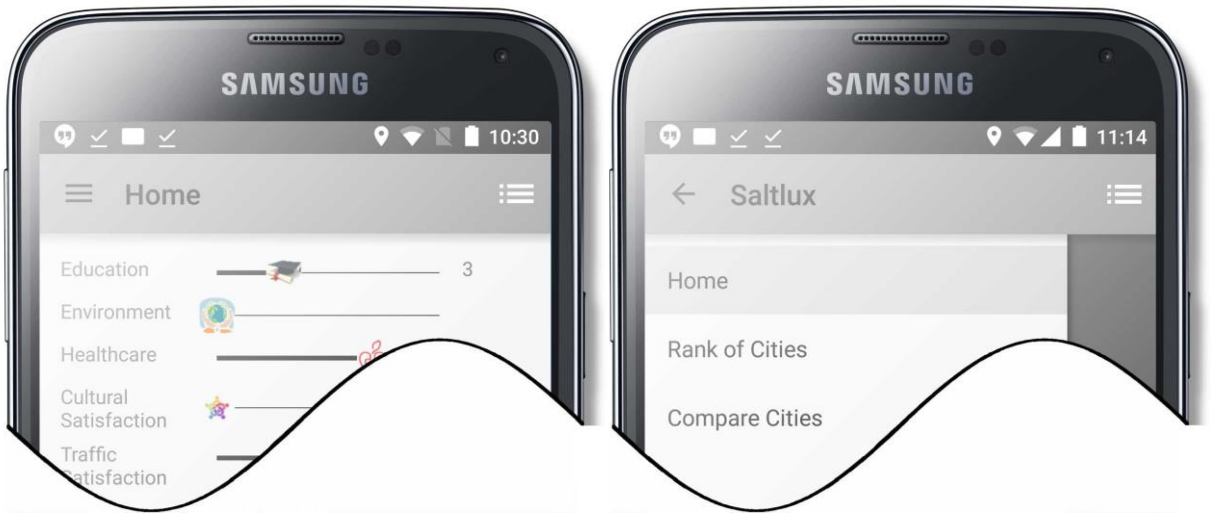


Figure 1: Side menus

2.1.2 Filters

Filters provide means to manage data subsets and interact with the data visualization.

Filters are implemented as a set of sliders, where each slider give weight to a particular value and filter data according to slider position.

In the PLUQI example, filters are defining the weight for each parameter from the PLUQI dataset: education index, environment index, healthcare index, cultural and traffic satisfaction indexes.

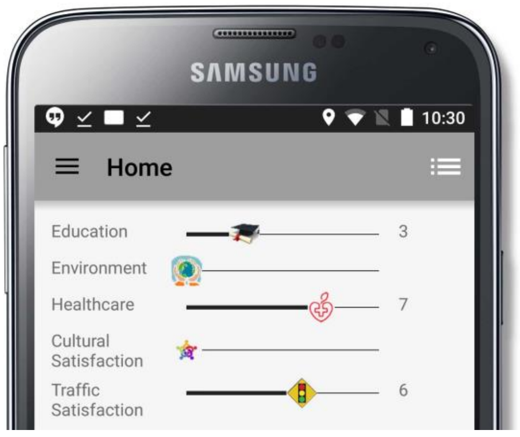


Figure 2: Filters

2.1.3 Geo data view

The Map view displays geo-location information using a maps provider such as Google maps or Apple maps. The view allows user interaction with the map, by zooming, panning and clicking, letting users interactively explore the data at each location.

Geo-data is shown as points of interest (POI) which are clickable allowing the user to obtain additional information about each POI.

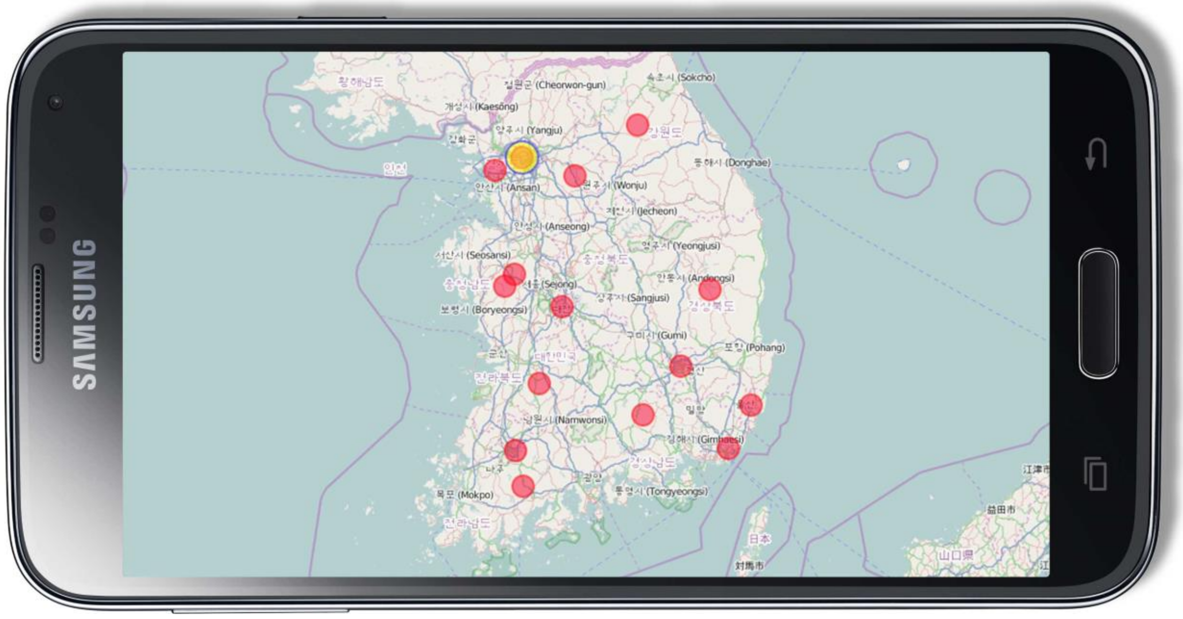


Figure 3: Map view

2.1.4 Bar chart view

Bar charts can be used to show comparison among categories. Categorical data is grouped into discrete groups, such as months of the year, age group, and cities. In the example below, a bar chart has been used to show the PLUQI index by administrative divisions (cities) in South Korea.

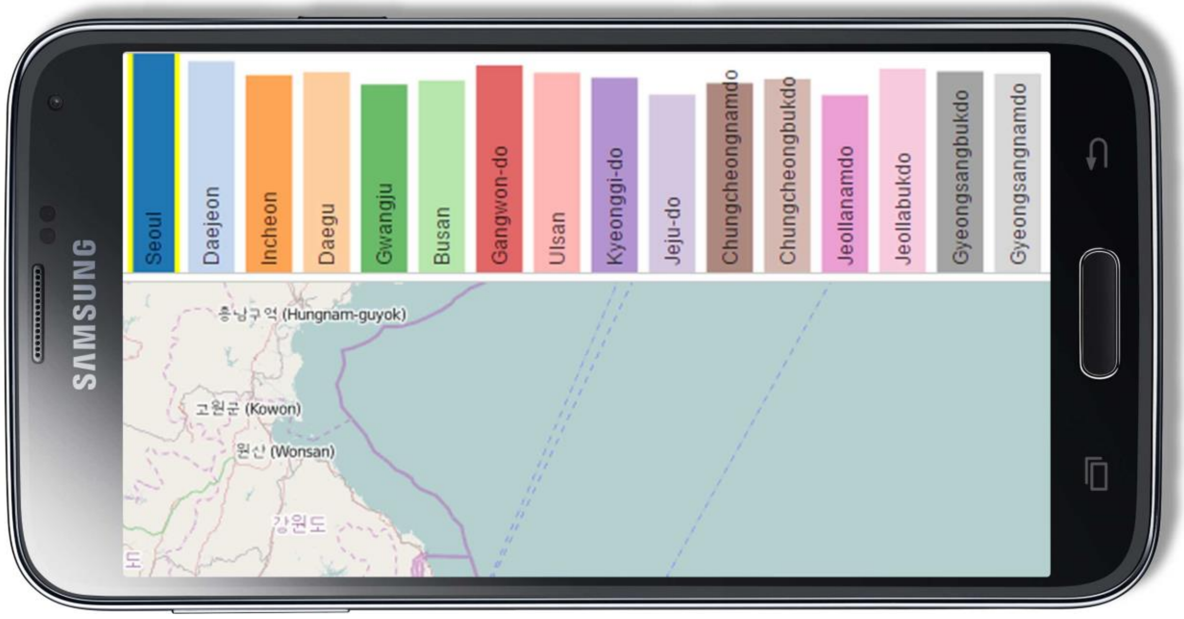


Figure 4: Bar char view

2.1.5 Multi dimension data view

Mobile apps allows one to compare several sets of data, as a multidimensional data view. This view can be used also to compare several datasets, assuming they have similar structure. The example below shows comparison of statistical data between major cities in South Korea. Each city has a separate dataset with information about education, crime rate, healthcare, etc. The view shows 3 major cities compared by all indexes.

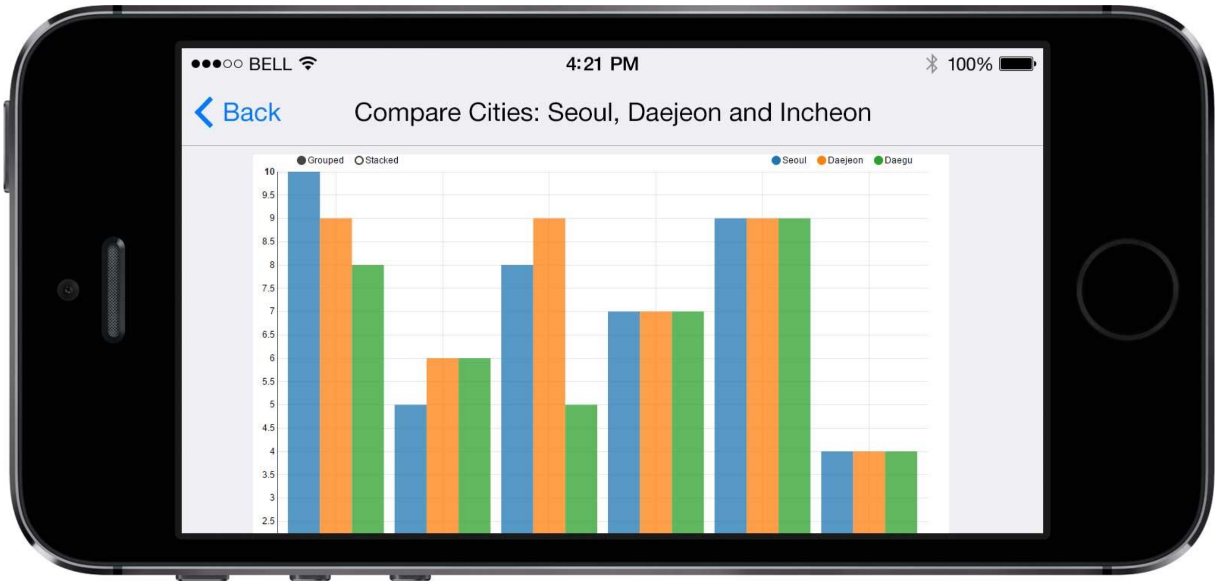


Figure 5: Line Multi dimension data view

2.1.6 Pie chart view

A pie chart is divided into slices to illustrate numerical proportion between components, which are usually of the same type. This view can provide information on how much each component takes from the whole (as a percentage).
The example demonstrates a visualization of crimes by city in South Korea as a count of crimes. It is very easy to spot that almost half of the crimes happened in the biggest city – Seoul.

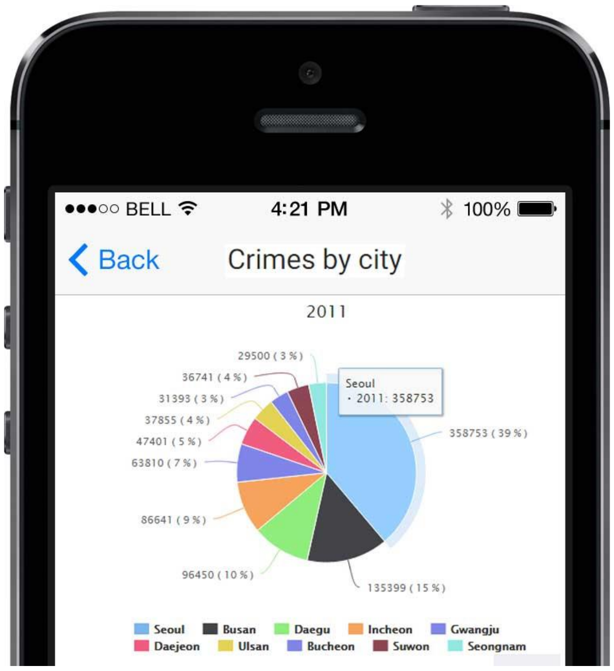


Figure 6: Pie Chart view

2.1.7 Line chart view

Line charts can be used to monitor trends and compare data. In the example below, the line chart shows comparison of crime rate data for different cities in South Korea.

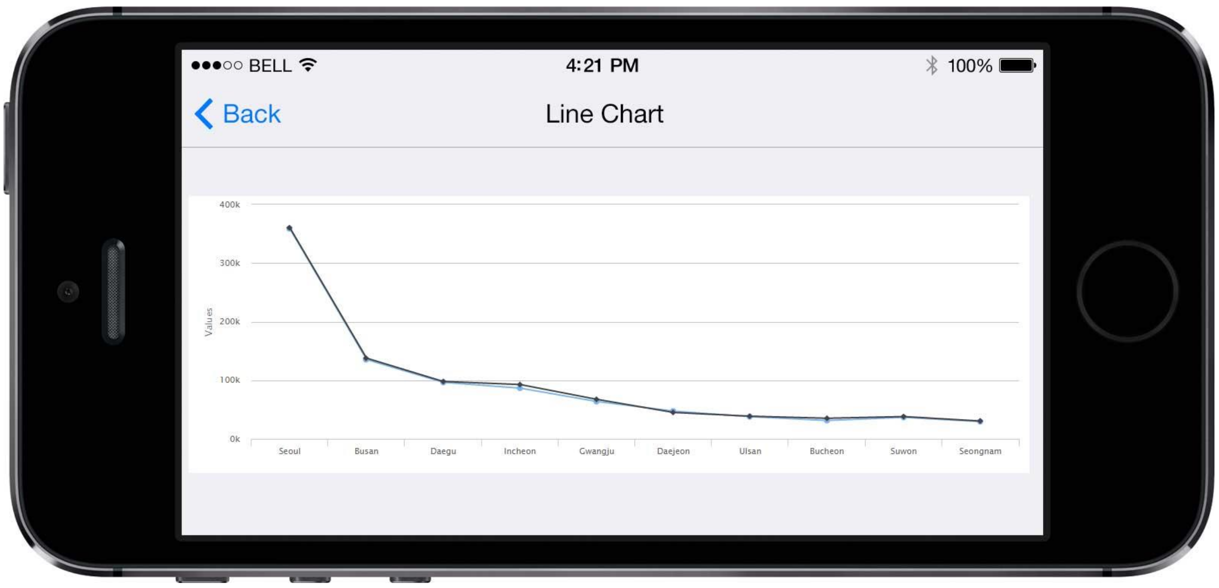


Figure 7: Line Chart view

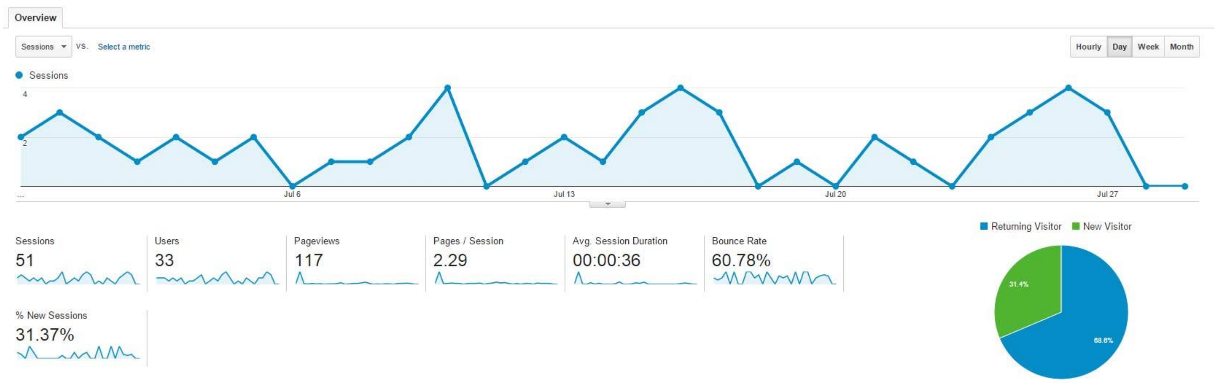
2.2 Offline support

Applications are capable of caching data and running in an offline mode, without actual connection to the Internet. This feature makes native apps suitable for demonstration and allows exploration of dataset when you travel on a plane, or other areas without connectivity.

2.3 Analytics

Applications utilize Google Analytics SDK for providing analytics statistics. The metrics provides by google analytics include:

- active users per hour of day / day / week / month;
- active sessions, which is a metric for application usage;
- how much time average user spend on application;
- how much views user visited and etc.



Sessions	Users	Pageviews	Pages / Session	Avg. Session Duration	Bounce Rate	% New Sessions
51	33	117	2.29	00:00:36	60.78%	31.37%

Figure 8: Analytics

3 Build and Installation Instructions

The native mobile applications are built using appropriate tools for iOS and Android development: XCode for iOS and Android Development Tools for Android.

3.1 Build and Installation Instructions for Android

3.1.1 Prerequisites

- OS supporting Java and Android Development Tools
- Java runtime environment 1.6+
- Android Developer Tools v24+ or Android Studio 1.2+

3.1.2 Obtaining the Source Code

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

```
https://github.com/dapaas/PLUQI/tree/master/Android
```

The following command retrieves source code from the repository:

```
# git clone https://github.com/dapaas/PLUQI
```

3.1.3 Compiling the Source Code

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

To build from Eclipse, import project into eclipse and chose auto-build project, or manually select from project menu “Build project”. To export Android APK, chose from Android Tools menu, Export Signed Package.

3.1.4 Deployment

The APK file with compiled code and configuration should be deployed using standard android application deploy tools:

- adb command from Android SDK tools: adb install <APK file>;
- upload APK file on web server and navigate from Android phone to the web page containing APK file;
- send APK file as an email attachment and open email from the phone.

3.2 Build and Installation Instructions for iOS

3.2.1 Prerequisites

- Mac OS X 10+
- Apple Developer Tools (Xcode)

3.2.2 Obtaining the Source Code

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

```
https://github.com/dapaas/PLUQI/tree/master/iOS
```

The following command retrieves source code from the repository:

```
# git clone https://github.com/dapaas/PLUQI
```

3.2.3 Compiling the Source Code

The source code can be compiled using xcode build system, or Xcode IDE. The following command build project using command line tools:

```
# xcodebuild –sdk iphoneos clean build
```

To build from Xcode IDE, open project (double click on project file) and select from menu Build – Build Project.

3.2.4 Deployment

The iOS application package (IPA) with compiled code should be deployed using standard iOS deployment mechanism:

- iTunes. To install IPA using iTunes, double click on IPA file, then select app for install on device and sync with device;
- using IPA repository, such as TestFlight;
- Build and install from Xcode. This is the easiest way to install application – connect the device to computer and select Run app on device.

4 Summary

This document provided an overview of the current prototype of Native Apps built in the DataGraft platform, and outlined a set of screens and workflows to build and use the apps from the perspective of publishers. Additionally we have documented each of the mobile visualization widgets the platform supports, as of M21.