



Towards We-Government: Collective and participative approaches for
addressing local policy challenges
Grant Agreement number: 693514

Deliverable

D3.5

Final release of WeGovNow platform prototype

Project co-funded by the European Commission within H2020-EURO-2014-2015/H2020-EURO-6-2015		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Author List

Organisation	Name	Contact Information
UniTo	Alessio Antonini	antonini@di.unito.it
UniTo	Liliana Ardissono	ardissono@di.unito.it
LiquidFeedback	Jan Behrens	jan.behrens@flexiguided.de
UniTo	Guido Boella	boella@di.unito.it
UniTo	Luigi Di Caro	dicaro@di.unito.it
Infalia	Sotris Diplaris	sdip@infalia.com
Infalia	Elena Dougia	elena@infalia.com
UCL	Carles Boils Gisbert	c.gisbert@ucl.ac.uk
MfC	Louise Francis	l.francis@mappingforchange.org.uk
MfC	Alex Garulli	a.garulli@mappingforchange.org.uk
LiquidFeedback	Axel Kistner	axel.kistner@flexiguided.de
Empirica	Lutz Kubitschke	lutz.kubitschke@empirica.com
PoliTo	Luigi La Riccia	Luigi.lariccia@polito.it
UniTo	Maurizio Lucenteforte	lucenteforte@di.unito.it
UniTo	Lucia Lupi	lupi@di.unito.it
UniTo	Roberto Micalizio	micalizio@di.unito.it
Infalia	Alexandros Mokkas	mokkas@infalia.com
Infalia	Spiros Nikolopoulos	snik@infalia.com
LiquidFeedback	Andreas Nitsche	andreas.nitsche@flexiguided.de
Infalia	Akis Papadopoulos	spap@infalia.com
UniTo	Giovanna Petrone	petrone@di.unito.it
UniTo	Adriano Savoca	savoca@di.unito.it
UniTo	Claudio Schifanella	schi@di.unito.it
UniTo	Egidio Dansero	egidio.dansero@unito.it
UniTo	Cristina Baroglio	baroglio@di.unito.it
UniTo	Matteo Baldoni	baldoni@di.unito.it
UniTo	Elena Grassi	e.grassi@gmail.com
UniTo	Marino Segnan	segnan@di.unito.it
LiquidFeedback	Björn Swierczek	b.swierczek@flexiguided.de
Infalia	Ioannis Tsampoulatidis	itsam@infalia.com

PoliTo	Angioletta Voghera	angioletta.voghera@polito.it
Infalia	Stefanos Vrochidis	svro@infalia.com
UHei	Alexey Noskov	noskov@uni-heidelberg.de
UHei	Adam Rousell	adam.rousell@uni-heidelberg.de
Unito	Gianmarco Izzi	izzi@di.unito.it
Unito	Michele Graziadei	michele.graziadei@unito.it
Unito	Ugo Pagallo	Ugo.pagallo@unito.it
Infalia	Akis Papadopoulos	spap@infalia.com

Status, Abstract, Keywords, Statement of originality

Dissemination level:	Public
Deliverable No.	3.5
Leading Partner	UniTo
Participating Partners	LiquidFeedback, Infalia, UCL, Mapping for Change, Empirica
Contractual date of delivery:	31 Oct 2017
Actual date of delivery:	22 Dec 2017
Work Package:	WP3 – Agile Development of the WeGovNow platform
Type:	Report
Approval Status:	Final
Version:	1.0
Abstract This document describes the third release of WeGovNow platform prototype, extending the description of the second prototype (deliverable D3.4)	
Keywords Prototype, Platform, Functionalities	
Statement of originality The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any purpose. The user thereof uses the information at its sole risk and liability.	

History

Version	Date	Reason	Revised by
1.0	04.10.2017	Creation of the initial document structure	Alessio Antonini
	09.10.2017	TODOs	Alessio Antonini
	16.10.2017	Changelogs, section 2, component/integration tables, TMP placeholder in section 3	Alessio Antonini
	18.10.2017	OnToMap, OTM logger	Liliana Ardissono, Adriano Savoca
	19.10.2017	ProxyLogger, Torin's instance of WGN, Consolidate OTM	Alessio Antonini
	23.10.2017	Introduction of Section 4, TMP in section 5.7, section 4.2.3	Alessio Antonini
	23.10.2017	Section 5.6	Andreas Nitsche
	25.10.2017	Sections structure, LandingPage, Consolidating Contributions, Annex 8	Alessio Antonini
	25.10.2017	Section 4 OTM	Adriano Savoca, Liliana Ardissono
	25.10.2017	Section 4 UWUM and LiquidFeedback	Andreas Nitsche, Axel Kistner
	25.10.2017	Trusted Marketplace updates based on stakeholders meeting outcome	Spiros Nikolopoulos, Ioannis Tsampoulatis, Alex Mokkas, Akis Papadopoulos, Stefanos Vrochidis, Sotiris Diplaris, Elena Dougia
	25.10.2017	Trusted Marketplace scenarios	Sotiris Diplaris, Stefanos Vrochidis, Elena Dougia
	25.10.2017	ImproveMyCity section	Ioannis Tsampoulatis, Alex Mokkas, Elena Dougia, Spiros Nikolopoulos
	26.10.2017	Consolidating contributions, Section 2 structure, Conclusion topics, exported annex 1 and 2 to separate files	Alessio Antonini
	27.10.2017	Consolidating contributions, LandingPage deployment, monitoring and support	Alessio Antonini
	30.10.2017	CommunityMaps/GeoKey deployment	Alex Garulli
	31.10.2017	Consolidating contributions, 4.1 introduction, 6.12 interacting with areaViewer, Annex 9	Alessio Antonini

	2.11.2017	Conclusions subsection SaaS vs bundle, consolidating contributions	Alessio Antonini
	2.11.2017	Annex 1	Liliana Ardissono, Adriano Savoca
	3.11.2017	4.1, 4.2.2, 4.3.1, user data storage, 4.3.2 troubleshooting and support, revision of exhibit 6 and 7, 2.2 and introduction of 2.3	Alessio Antonini
1.1	9.11.2017	Moved future development in conclusions, fix future development 7.1, 7.2, 7.4	Alessio Antonini
1.2	9.11.2017	Geo-Spatial Data Repository for Quality Assessment was described: Changelogs, 2.1, 4.2.8, 6, 6.14,	Alexey Noskov, Adam Rousell
	10.11.2017	Consolidating contributions, sections 7.3, introduction to section 7, 6.9 InputMap V3 changelogs, 2.3 integration	Alessio Antonini
	10.11.2017	Trusted Marketplace description	Spiros Nikolopoulos, Ioannis Tsampoulatis, Alex Mokkas, Akis Papadopoulos, Stefanos Vrochidis, Sotiris Diplaris, Elena Dougia
	13.11.2017	FirstLife improvements, fixing exhibits style, consolidating contributions, fix table of contents	Alessio Antonini
1.3	13.11.2017	Revise introduction	Alessio Antonini, Lutz Kubitschke
	24.11.2017	Added content in 2., 5. (Exh-8), 5.2,	Andreas Nitsche
	27.11.2017	Revision	Claudio Schifanella
	28.11.2017	Added two screenshots in 5.6 (Exh. 34a and 34b)	Andreas Nitsche
	20.12.2017	Final revision	Claudio Schifanella
	19.03.2019	Revision of parts of deliverable concerning OnToMap, fixing typos	Liliana Ardissono, Gianmarco Izzi
	25.03.2019	6.13 updated. Section 7 added	Alexey Noskov
	26.03.2019	Final revision	Claudio Schifanella

Table of Contents

Author List	2
Status, Abstract, Keywords, Statement of originality	4
History	5
Changelogs	9
Executive Summary	9
1. Introduction	13
2. The modular nature of the WeGovNow platform	16
2.1. Platform instances	19
2.2. The development process of the WeGovNow platform	25
2.3. Approach towards software integration in WeGovNow	26
3. Description of the individual WeGovNow platform components	28
3.1. GeoKey & CommunityMaps	28
3.2. FirstLife	29
3.3. ImproveMyCity	30
3.4. LiquidFeedback	32
3.5. Trusted Marketplace	33
4. Setting up a WeGovNow platform instance	36
4.1. Setup of WeGovNow core	36
4.2. Setup of WeGovNow components	37
4.3. Monitoring and support	44
5. Level of integration achieved in relation to existing and newly developed components	47
5.1. LandingPage	50
5.2. Authentication Server	53
5.3. GeoKey and Community Maps	55
5.4. FirstLife	60
5.5. ImproveMyCity	64
5.6. LiquidFeedback	72
5.7. Enhanced User Profile Management (module of Trusted Marketplace)	77
5.8. Trusted Marketplace	79
5.8.1. Next steps towards pilot platforms implementation	81
6. Description of platform services	85

6.1.	Authentication Service	86
6.2.	Application Discovery Service	88
6.3.	Style Service	89
6.4.	NavigationBar Service	89
6.5.	User data storage	90
6.6.	Centralised User Profile	90
6.7.	Crowdsourced and Linked Open Data endpoint	92
6.8.	User's Activity Logger (OTM logger)	97
6.9.	InputMap (V3)	98
6.10.	TileServer	105
6.11.	Enhanced User Profile Management	106
6.12.	AreaView (V2)	108
6.13.	Geo-Spatial Data Repository (GSDR) of WeGovNow for Quality Assessment (V3)	112
7.	WeGovNow Data Quality Assessment Results	121
7.1.	Resulting Datasets	123
7.2.	Open Data Evaluation	125
7.3.	Evaluation of Open Street Map Data	129
7.4.	Ad-Hoc Quality Assessment of Tiled Raster Maps for Parco Dora	131
7.5.	Aggregated Data Quality	136
8.	Conclusions	140
8.1.	How to get WeGovNow	143
Annexes		144

Changelogs

The present document extends the report of the “Second release of WeGovNow platform prototype”, deliverable D3.4, updating the state of development of WeGovNow platform. The following list summarises the major changes:

Sections

1. Setting up WeGovNow platform (Section 4)
2. Extended conclusions

Environments

1. Deployment of final prototype <https://pt3.wegovnow.eu>
2. Deployment of pilot instances
 - a. <https://sandona.wegovnow.eu>
 - b. <https://torino.wegovnow.eu>
 - c. <https://southwark.wegovnow.eu>

New available and updated features

1. InputMap V3: has been revised in style and features to support edit and view only mode
2. AreaViewer V2: supports interactive messages with host applications, two different layout behaviours, source prioritization and temporal filters
3. TileServer: extended the endpoints and supporting local repositories of GeoJSON files
4. OTM Logger: support to API keys for GET requests
5. LoggerProxy endpoint to provide GET requests based on slippy tile notation
6. Geo-Spatial Data Repository (GSDR): a new web service of WeGovNow for data quality assessment

Platform instance settings

1. Updated Trusted Marketplace component in 3rd prototype installation
2. Re-added temporally unavailable GeoKey/CommunityMap in 2nd prototype installation

Options for further extensions of the platform

1. Area calendar module in FirstLife

Executive Summary

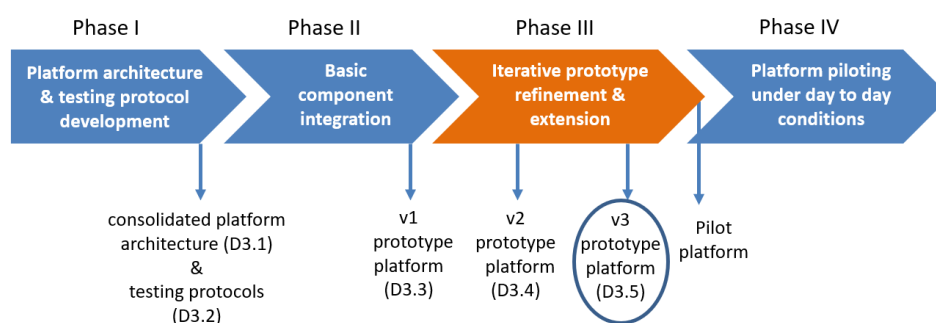
The present document extends the report of the “Second release of WeGovNow platform prototype”, deliverable D3.4, updating the state of development of the WeGovNow platform (see Changelogs for further details). To facilitate easy reading of the current document as a self-contained report information that has already been provided in the previous version is repeated to some extent. A change log has been added summarising the major changes achieved when compared with the previous version of the WeGovNow prototype.

In a nutshell, WeGovNow strives for integrating a set of innovative software applications into a unified citizen engagement platform. In doing so the project aims at overcoming current limitations of existing digital tools for citizen reporting, e-participation, and communication between the citizen and the public administration. To this end, a number of civic engagement applications that have already existed prior to the project are to be integrated into a single online platform together with software components to be newly developed.

With a view to facilitating the utilisation of the WeGovNow platform within different local contexts, individual implementation instances are to be configurable in a modular manner. Also, the new WeGovNow engagement platform is to be principally extendable with further applications which potentially may be desired to be added in future implementation instances. Such a multi-faceted set of basic requirements poses a number of challenges for the design of the WeGovNow platform, in particular when it comes to

- the development cycle and design process which is to involve a variety of stakeholders at the three pilot cities participating in WeGovNow,
- and the achievement of a coherent user experiences across a diverse range of software components to be integrated into a single online platform, including existing ones and newly developed ones.

Against this background, a multi staged development approach has been adopted for the purposes of WeGovNow. As graphically summarised in the schema below, four main phases can be discerned, starting with a thorough consolidation of the initial platform architecture (D3.1) and the development of suitable testing protocols (D3.2) to be utilised throughout the subsequent development phases.



This was followed by basic integration work (Phase II) mainly focussing on the seamless interoperation of those software components that have already existed prior to WeGovNow, e.g. in terms of a unified user management solution. The previous report (D3.4) presented a description of the second prototype of WeGovNow platform. The current report (D3.5) extends the previous report including new functionalities and ongoing development activities. The remaining duration of the

development phase (Phase III) will focus on the finalisation of the pilot version of the WeGovNow platform to be publicly operated under day-to-day conditions by the three pilot municipalities. To this end, the current v3 prototype will undergo further fine-tuning in collaboration with diverse range of stakeholders in three cities participating in the project. This will yield the final version of the fully tested platform which is then to be piloted under everyday conditions with large numbers of users (Phase IV).

The first release of the WeGovNow prototype platform is provided as a “working” environment for:

- Development purposes (<https://dev.wegovnow.eu>)
- Testing and presentation purposes (<https://sandbox.wegovnow.eu>)

To support the involvement of stakeholders at the three WeGovNow pilot sites during the next stage of platform development (Phase III) dedicated prototype implementation instances are provided to each of the three sites:

- San Donà di Piave (<https://sandona.wegovnow.eu>)
- Southwark (<https://southwark.wegovnow.eu>)
- Torino (<https://torino.wegovnow.eu>)

From a development angle, the WeGovNow platform prototype - as it currently stands - provides an environment of various internal services (core features) required to run a WeGovNow instance, including smooth interoperation of various stand-alone web components (WeGovNow components) integrated into the overall platform.

A local WeGovNow implementation instance can be configured to use any combination of modular components, enabling cross-component features according to the combination of active components.

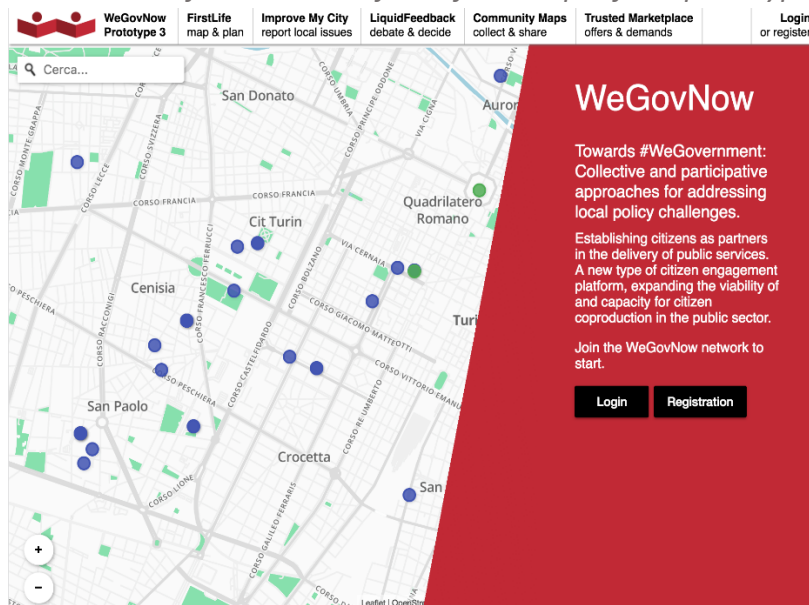
From the perspective of the end users, the current version of the prototype platform provides a number of functionalities through the integration of components that have existed prior to WeGovNow already, including:

- Map-based social networking functionalities realised by means of integrating FirstLife;
- Citizen reporting functionalities realised through the integration of ImproveMyCity;
- Opinion formation and voting functionalities realised through the integration of LiquidFeedback.
- Community mapping functionalities realised by means of integrating Community Map and GeoKey

During the process of their seamless integration into the WeGovNow platform, to a certain extent the existing components underwent further development work. Beyond this, new software components such as the so-called Trusted Marketplace and

a personal Dashboard were developed. Further to those functionalities immediately visible to the end users, a range of internal software services were developed as well. These include for instance a Style Service, an Application Discovery Service, a Unified Authentication System, a Centralised User Profile and a Centralised Activity Logger. Such internal software services do not only enable the seamless technical interoperation of the individual software components which together make up the overall WeGovNow platform but additional options for user action as well, e.g. though the combined utilisation of individual platform components by the user. Not at least, they enable straight forward customisation of local platform implementation instances. When it comes to interface integration, the approach adopted for the purposes of WeGovNow can be characterised as a gradual convergence toward a well-documented design framework (Material Design <https://material.io/>), a repository to collect common solutions to shared design issues, and a set of features supporting the propagation of user setups, customisations and elements among WeGovNow components. This approach provides a pragmatic solution to the need for providing a coherent user experience across the platform while at the same time taking into account that changing the interface design of existing components is much more complex and resource consuming (and thus costly) when compared with newly developed components. Also, the design approach adopted for the purpose of WeGovNow helps in keeping the platform as open as possible towards additional components that may be desired to be integrated after the ending of the project duration.

Screen shot of the web interface of the V3 platform prototype

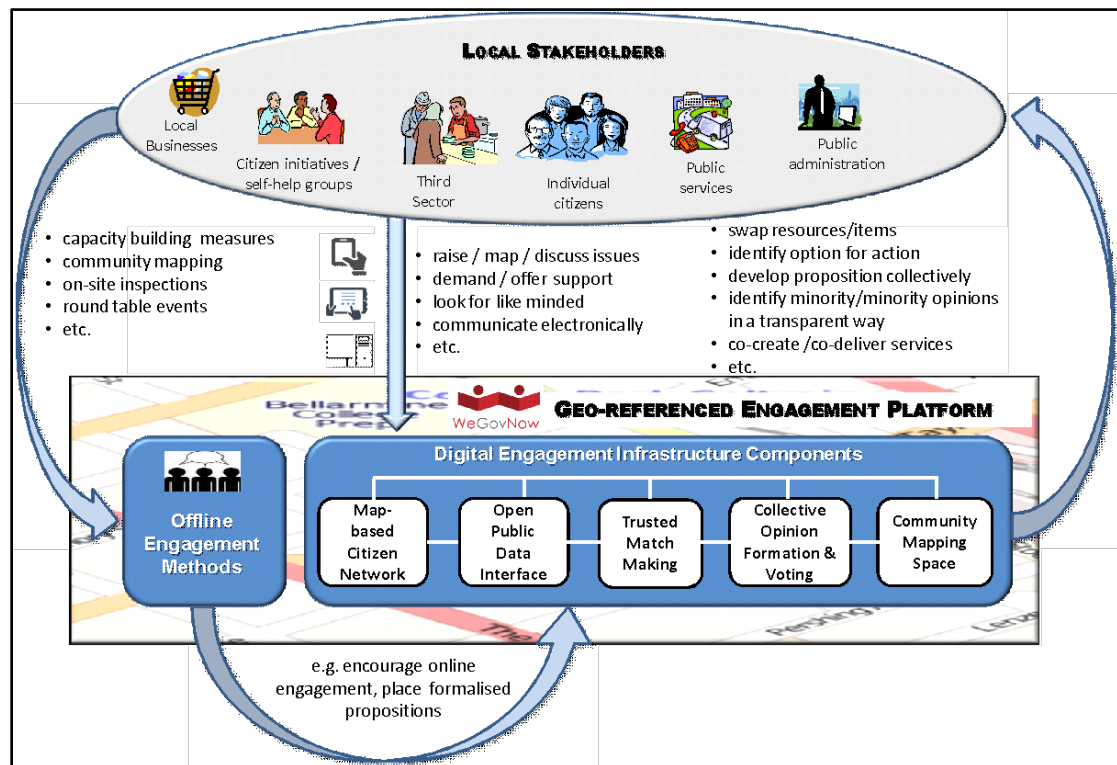


1. Introduction

The present document extends the report of the “Second release of WeGovNow platform prototype”, deliverable D3.4, updating the state of development of WeGovNow platform (see Changelogs for further details).

In a nutshell, WeGovNow strives for integrating a set of innovative software applications into a unified citizen engagement platform. In doing so the project aims at overcoming current limitations of existing digital tools for citizen reporting, e-participation, and communication between the citizen and the government as graphically summarised in Exhibit 1. To this end, a number of civic engagement applications that have existed already prior to the project are to be integrated into a single online platform together with various software components to be newly developed.

Exhibit 1: Graphical overview of the WeGovNow platform and its utilisation.



With a view to facilitating further utilisation of the WeGovNow platform within different local contexts, individual implementation instances are to be configurable in a modular manner. Also, the new WeGovNow engagement platform is to be principally extendable with further applications which potentially may be desired to be added in future implementation instances. Such a multi-faceted set of basic requirements poses several challenges for the design of the WeGovNow platform, in particular, when it comes to

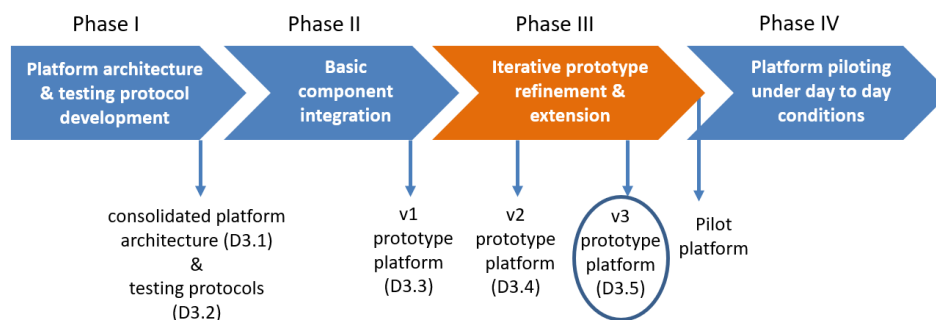
- the development cycle and design process which is to involve a variety of stakeholders at the three pilot cities participating in WeGovNow,

- and the achievement of a coherent user experiences across a diverse range of software components to be integrated into a single online platform, including those that have existed prior WeGovNow already and those that are newly being developed within the project.

Against this background, a multi staged development approach has been adopted for the purposes of WeGovNow. As graphically summarised in Exhibit 2, four main phases can be discerned, starting with a thorough consolidation of the initial platform architecture (D3.1) and the development of suitable testing protocols (D3.2) to be utilised throughout the subsequent development phases.

This was followed by basic integration work (Phase II) mainly focussing on the seamless interoperation of those software components that already existed prior to WeGovNow, e.g. in terms of a unified user management solution. The previous report (D3.4) is a paper-based description of the second prototype version of the WeGovNow platform. The current report (D3.5) extends the previous report including the new functionalities and ongoing development activities.

Exhibit 2: WeGovNow platform development phases.



During the remaining duration of the development phase (Phase III) work will focus on the finalisation of the pilot version of the WeGovNow platform to be publicly operated under day-to-day conditions by the three pilot municipalities. To this end, the final current prototype version (v3) will undergo further fine-tuning in an iterative manner, in collaboration with diverse range of stakeholders in three cities participating in the project. This will yield the final version of the fully tested platform which is then to be piloted under everyday conditions with large numbers of users (Phase IV).

The following Section 2 of this report provides an overview of the development work achieved so far. This is followed by a description of the functionalities provided by the third version of the WeGovNow platform prototype through the integration of existing software components into the overall platform (Section 3). Section 4 describes how to set up a new WeGovNow instance. Next, a description of the level of integration

achieved so far is provided when it comes to both, components that have existed prior WeGovNow already and those components that are specifically being developed for the purposes of WeGovNow (Section 5). Moreover, core platform features implemented as internal software services with the current prototype release are described (Section 6). Conclusions and options for further extensions of the platform end the document (Section 7).

2. The modular nature of the WeGovNow platform

As described in earlier deliverables, the first release of the WeGovNow platform prototype (v1 prototype) largely resulted from the integration of existing software solutions into the overall platform according to an architectural model developed during the start-up phase of the project (D3.1), under the premise of building a seamless environment specifically for:

- 1) user authentication
- 2) component navigation

The first prototype enabled users to log in from any component and navigate through the platform, keeping a user authentication session via single-sign-on (“auto login”) functionality implemented by each component.

From the perspective of integrating the various software components, the first prototype enabled to use the individual components’ APIs by means of an UWUM OAuth 2.0 token, generated by users’ authentication. Moreover, the first prototype enabled the implementation of functionalities within the WeGovNow ecosystem, based on the following patterns:

1. data integration (JSON/GeoJSON document exchange)
2. inclusion of dynamic html snippet
3. feature integration (HTML/CSS/JavaScript module integration)
4. integrated data based on the common logging system

From a development angle, the two subsequent WeGovNow platform prototype versions (v2 and v3) provided an environment of services (core features) required to run a WeGovNow instance, including a set of stand-alone web components (WeGovNow components) integrated into the overall platform. A local WeGovNow implementation instance can be configured to use any combination of modular components, enabling cross-component features according to the combination of active components.

According to the modular nature of the WeGovNow architecture, the challenge of integrating diverse software components within a single platform concerns multiple levels, i.e. integration of data, design and development of common services, functionalities and interfaces. Each of the individual software applications that have been integrated into the overall platform has its inner coherence in terms of data, functionalities and interfaces. However, in terms of WeGovNow platform components they have been harmonised with a view to add value beyond their implementation as standalone solutions. The remaining development work strand (Phase III) now focuses on final harmonisation work concerning the different layers - data, functionalities and interfaces.

From the perspective of the end users, WeGovNow platform provides the following core functional elements:

- Map-based social networking functionalities realised by means of integrating FirstLife. A documentation is available at:
 - o <http://tutorials.firstlife.org/> (Italian)
- Citizen reporting functionalities realised by the integration of ImproveMyCity. A documentation is available at:
 - o <https://www.youtube.com/watch?v=ONZd6yk0R2E&t=3s> (English)
 - o <http://www.improve-my-city.com/features> (English)
 - o <https://wegovnow.improve-my-city.com/handbook> (English)
- Opinion formation and voting functionalities realised by the integration of LiquidFeedback. A documentation is available at:
 - o <http://liquidfeedback.org/> (English)
 - o https://youtu.be/y0e9_-leRt8 (English)
 - o <http://www.interaktive-demokratie.org/files/downloads/LF-Information-Kit-EN.pdf> (English)
 - o <http://principles.liquidfeedback.org/> (English)
 - o <http://dev.liquidfeedback.org/trac/lf/> (English)
- Customisable community mapping functionalities realised by the integration of GeoKey and Community Maps. A tutorial for map users and a guide for administrators is available at:
 - o <http://help.communitymaps.org.uk/en/index.html>
 - o <http://geokey.org.uk/help/>
- Trusted Marketplace, first prototype (English),
 - o search, profile management and users' activities
 - o offering goods and services in the community

As mentioned earlier, the integration process encompassed newly developed software components and pre-existing ones which were further redeveloped within the project. These were augmented by a range of newly developed internal services which do not immediately become visible to the end user.

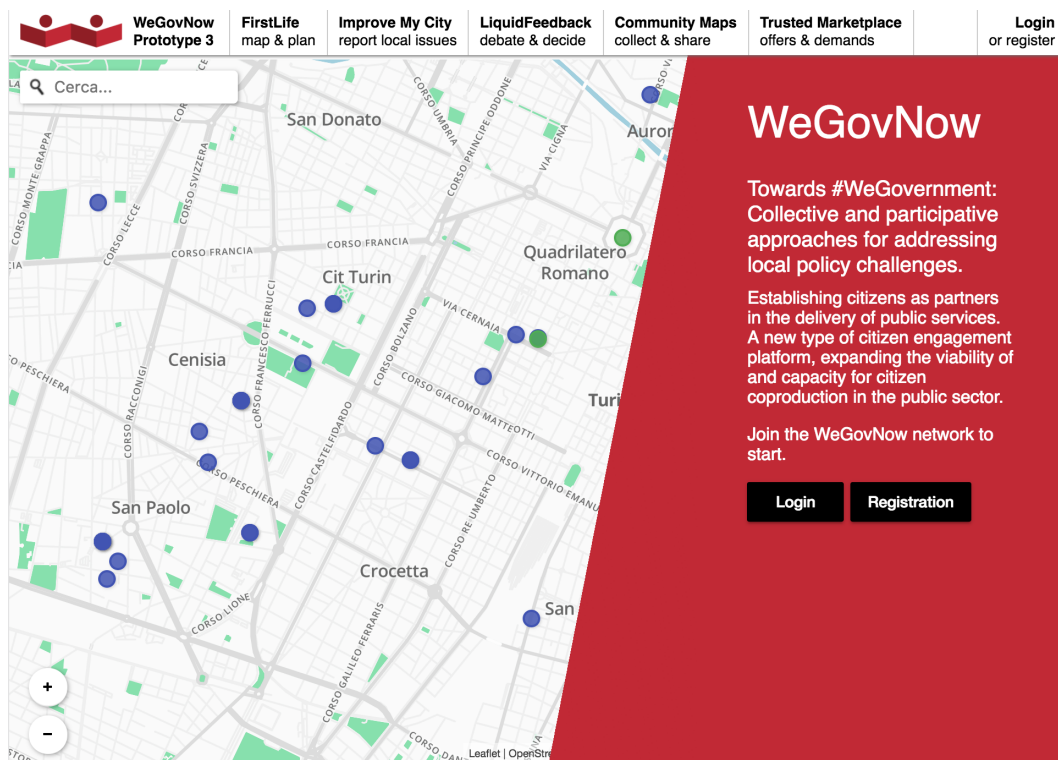
When it comes to interface integration in particular, the approach adopted for the purposes of WeGovNow can be characterised as a gradual convergence toward a well-documented design framework (Material Design <https://material.io/>), a repository to collect the solutions to shared design issues, and a set of features to support the propagation of user setups, customisations and elements among individual WeGovNow components.

Adopting a common design standard (Material Design) as a goal to be achieved by the end of the prototype development phase and continuous collaboration in addressing

common issues can be seen as the operational approach towards software integration adopted for the purposes of WeGovNow. This approach is a pragmatic solution to the need of providing a coherent platform in a development context within which changing the design of existing components is costly, time consuming and not as straight forward as for new components. Approaching the interface integration as a path and not as a requisite also enables the WeGovNow platform to remain principally open towards developers and companies who may be interested in contributing further platform components in the future, thereby avoiding unreasonable demands such as to re-design existing software as a mandatory condition.

At the same time, WeGovNow provides many services to support the integration of platform components, and the choice of Material Design was not at least made due to the availability of extensive documentations, examples and framework-based material supporting to the development of material interfaces in the most popular web languages and frameworks.

Exhibit 3: Screen shot of the landing page of WeGovNow platform final prototype.



From the point of view of the end users, the current release of the platform prototype is accessed through a common web interface which will be described in more detail later in this document (Section 5.1). Exhibit 3 provides a screenshot of the common entry point to the WeGovNow platform prototype as it is currently available to the users. It is expected that this version will undergo further fine-tuning according to the stakeholder input to be yielded by the various engagement activities conducted at the

three pilot sites during the final stage of the platform development phase of the project (Phase III).

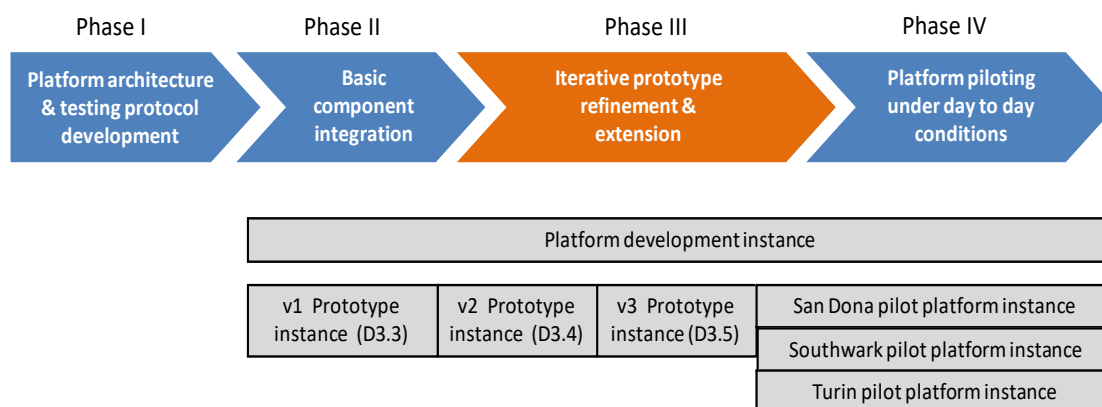
2.1. Platform instances

Different platform instances have been set up during the course of WP3 (Exhibit 4). A dedicated development environment (<https://dev.wegovnow.eu>) is meant to be constantly updated with the alfa version of the software, it is specifically required to test the communication between components and to develop integrations between components.

The 1st, 2nd and 3rd prototypes versions correspond to three milestones of the project and development life cycle. The platform versions are frozen “as they are”, to keep track of the evolution of WeGovNow platform.

Based on the 3rd and final prototype version, three different pilot instances of the platform have been set up as graphically summarised in Exhibit 4. Prior to the launching of the public validation trials, in each of the three pilot municipalities, the current pilot instances of the platform will be subject to customisation and fine-tuning towards local implementation requirements. The reminder of this subsection provides an overview of the settings implemented with each platform instance.

Exhibit 4: Overview of WeGovNow platform instances



1st prototype settings

The settings of the first prototype of WeGovNow platform (<https://pt1.wegovnow.eu>) includes all existing components:

- CommunityMaps <https://wegovnow-cm.geokey.org.uk/>
- GeoKey <https://wegovnow-gk.geokey.org.uk/admin/dashboard/>
- FirstLife <https://pt1.wegovnow.firstlife.org/>

- ImproveMyCity <https://pt1-imc.infalia.com/>
- LiquidFeedback <https://wegovnow-pt1.liquidfeedback.com/>

The entry point of the platform is the first prototype of LandingPage <https://pt1.wegovnow.eu/>.

2nd prototype settings

The installation of the second prototype (<https://pt2.wegovnow.eu>) of WeGovNow introduces a new component:

- Trusted Marketplace 1st prototype version

The first version of Trusted Marketplace does not provide full functionalities yet, but it is fully integrated in WeGovNow (see Section 5).

On the other hand, the second prototype does not include GeoKey/CommunityMap components, due to a technical problem preventing to run multiple instances of GeoKey that has emerged during the setup operation of the platform instance.

3rd prototype settings

The installation of the third and final prototype (<https://pt3.wegovnow.eu>) of WeGovNow (Exhibit 3) was setup with the following components and modules:

- AreaViewer <https://pt3.areaviewer.firstlife.org>
- FirstLife <https://pt3.wegovnow.firstlife.org>
- ImproveMyCity <https://pt3-imc.infalia.com>
- LandingPage <https://pt3.wegovnow.eu>
- LiquidFeedback <https://wegovnow-pt3.liquidfeedback.com>
- CommunityMaps <https://wegovnow-cm-pt3.geokey.org.uk>
- GeoKey <https://wegovnow-gk-pt3.geokey.org.uk/admin/dashboard/>
- LoggerProxy <https://loggerproxy-pt3.firstlife.org>
- OnToMapOnToMap <https://p3.api.OnToMap.eu>
- TileServer <https://tiles.firstlife.org>
- Trusted Marketplace <https://pt3-tmp.infalia.com>
- GSDR <https://wgn-pt3.gsdr.gq>

In this final prototype, ImproveMyCity, Trusted Marketplace and LiquidFeedback integrated both AreaViewer and InputMap in their own application. All components can retrieve Open Data from OnToMap (OTM) in order to populate their maps, and are providing users' activities to OTM logger, enabling the use of AreaViewer as consolidated map of WeGovNow.

San Donà di Piave pilot instance

The installation of the San Donà di Piave's instance of WeGovNow and final prototype (<https://sandona.wegovnow.eu>) of WeGovNow (Exhibit 5) was setup with the following components and modules:

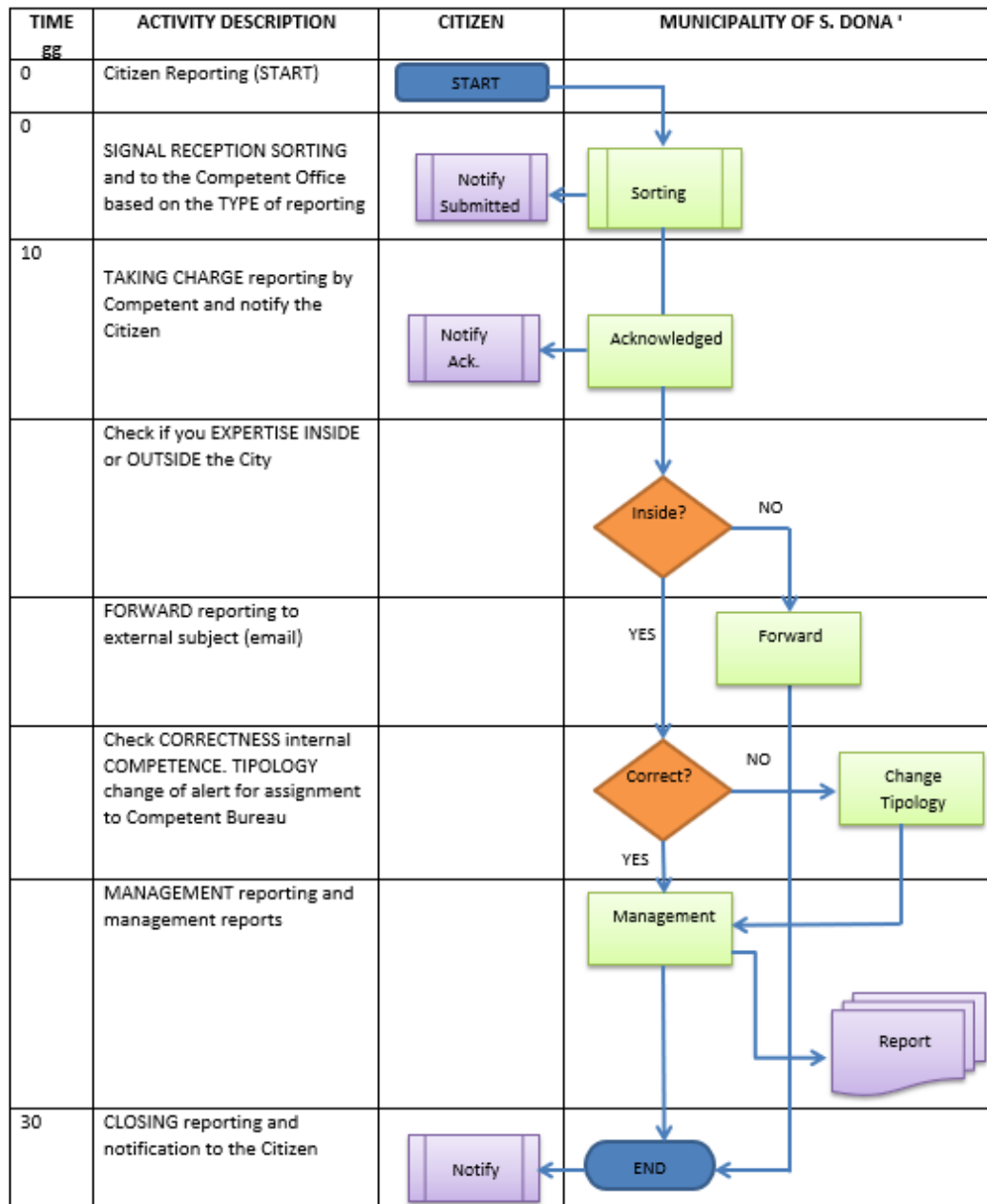
- AreaViewer <https://sandona.areaviewer.firstlife.org>
- FirstLife <https://sandona.wegovnow.firstlife.org>
- ImproveMyCity <https://sandona-imc.infalia.com>
- LandingPage <https://sandona.wegovnow.eu>
- LiquidFeedback <https://sandona.liquidfeedback.net>
- Community Maps <https://wegovnow-cm-sandona.geokey.org.uk/welcome>
- GeoKey <https://wegovnow-gk-sandona.geokey.org.uk/admin/dashboard/>
- LoggerProxy <https://loggerproxy-sandona.firstlife.org>
- OnToMap <https://sandona.api.OnToMap.eu>
- TileServer <https://tiles.firstlife.org>
- TrustedMarketplace <https://sandona-tmp.infalia.com>
- GSDR <https://wgn.gsdr.gq/sandona>

In general, this instance has already been customised in relation to setting interface colours, and the logo of the Municipality of San Donà di Piave. Moreover, ImproveMyCity tailored specifically to San Donà di Piave internal workflow (Exhibit 6). More specifically, the various departments of responsibility and their categories are set, the statuses of issues are defined and the accounts of the employees per department and category are created. The notification rules are also set, based on the Municipalities needs, per issue categories. Finally, the Italian translation strings of ImproveMyCity are merged to this instance and customised email templates (subject and body) for notification purposes are set. Customised rules are also applied per administrator group (e.g. which user groups are allowed to access the reports or moderate new issues). This instance will be further fine-tuned according to the stakeholders needs at pilot site level.

Exhibit 5: WeGovNow instance for San Donà di Piave pilot.



Exhibit 6: San Donà di Piave's internal workflow.



Torino pilot instance

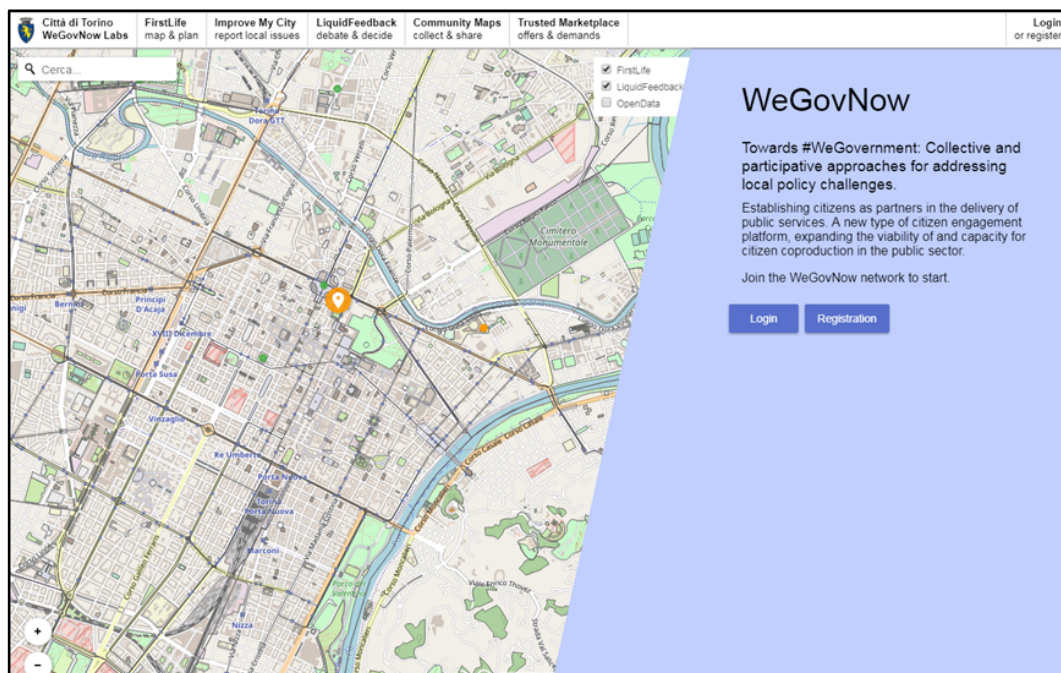
The installation of the Torino's instance of WeGovNow and final prototype (<https://torino.wegovnow.eu>) of WeGovNow (Exhibit 7) was setup with the following components and modules:

- AreaViewer <https://torino.areaviewer.firstlife.org>
- FirstLife <https://torino.wegovnow.firstlife.org>
- ImproveMyCity <https://torino-imc.infalia.com>
- LandingPage <https://torino.wegovnow.eu>
- LiquidFeedback <https://torino.liquidfeedback.net>

- Community Maps <https://wegovnow-cm-torino.geokey.org.uk>
- GeoKey <https://wegovnow-gk-torino.geokey.org.uk/admin/dashboard/>
- LoggerProxy <https://loggerproxy-torino.firstlife.org>
- OnToMap <https://torino.api.OnToMap.eu>
- TileServer <https://tiles.firstlife.org>
- TrustedMarketplace <https://torino-tmp.infalia.com>

In general, this instance has yet been customised in terms of setting interface colours, and the logo of the Municipality of Turin. Moreover, ImproveMyCity translated to Italian both for frontend and backend. Similar actions as San Dona instance will take place on Torino instance as well.

Exhibit 7: Torino instance of WeGovNow platform



Southwark pilot instance

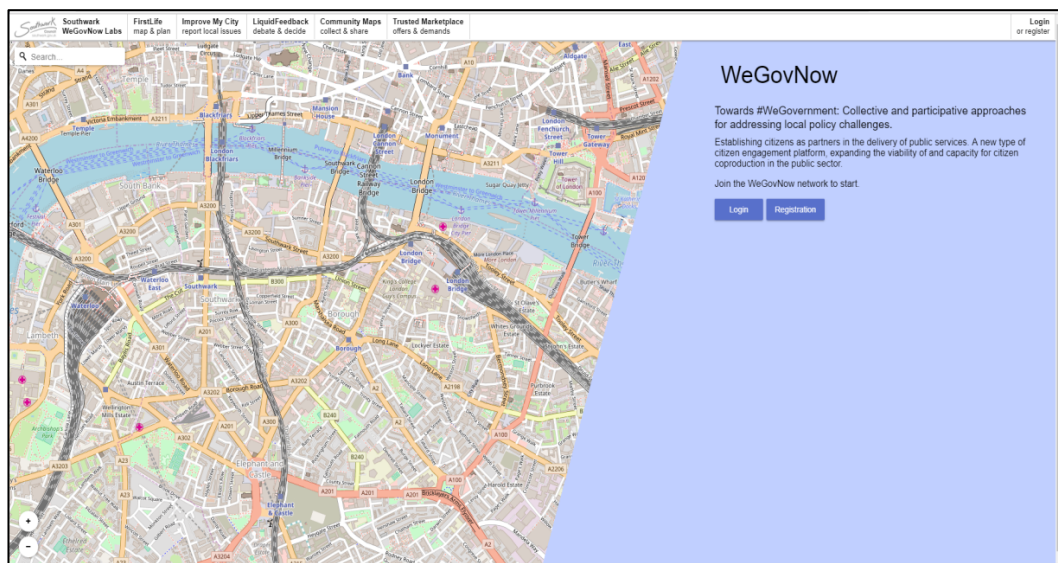
The installation of the Southwark's instance of WeGovNow and final prototype (<https://southwark.wegovnow.eu>) of WeGovNow (Exhibit 8) was setup with the following components and modules:

- AreaViewer <https://southwark.areaviewer.firstlife.org>
- FirstLife <https://southwark.wegovnow.firstlife.org>
- ImproveMyCity <https://southwark-imc.infalia.com>
- LandingPage <https://southwark.wegovnow.eu>
- LiquidFeedback <https://southwark.liquidfeedback.net>
- Community Maps <https://wegovnow-cm-southwark.geokey.org.uk>
- GeoKey <https://wegovnow-gk-southwark.geokey.org.uk/admin/dashboard/>

- LoggerProxy <https://loggerproxy-southwark.firstlife.org>
- OnToMap <https://southwark.api.OnToMap.eu>
- TileServer <https://tiles.firstlife.org>
- TrustedMarketplace <https://southwark-tmp.infalia.com>

In general, this instance has yet been customised in terms of setting interface colours, and the Southwark logo. Similar actions as San Dona instance will take place on Southwark instance as well.

Exhibit 8: Southwark instance of WeGovNow platform



2.2. The development process of the WeGovNow platform

At the current stage, the features described in the Consolidated WeGovNow System Architecture (deliverable D3.1) are available in the platform, though some minor last-minute requirements are currently being integrated within the software components concerned, e.g. in relation to the Trusted Market Place component (see 3.5). The WeGovNow architecture is consolidated and tested, as shown by the three subsequent prototype versions (v1, v2 and v3) and the three pilot instances *torino.wegovnow.eu*, *sandona.wegovnow.eu*, and *southwark.wegovnow.eu*. All further improvements will not result in new formal versions of WeGovNow but will be included via continuous integration or new versions of the single components.

In terms of Agile methodology, the development of WeGovNow required a number of sprints related to:

1. The release of the common modules
2. The setup of a new instance of the platform

In the first case, the structure of the organisation had a main role, a specific development team represented by the developer in charge of the task, and all other teams in testing the module.

In the latter case, the lead of sprints was given to UWUM development team (LiquidFeedback team), because:

- each task of each component required a feedback or activity by UWUM team
- UWUM setup is the actual bridge of WeGovNow components, and an effective monitoring system of the instance status
- UWUM incorporates most of the customer requirements: included components, colours, user verification policy and icon, provided by the municipalities

Summarising the sprints in WeGovNow development involving more than one of the technical teams, the following can be mentioned:

- UWUM first release
- UWUM final release
- InputMap first release
- InputMap third release
- AreaViewer second release

Currently, the development teams are involved in:

- UWUM user data storage
- Trusted Marketplace second release
- Community Maps/GeoKey social media integration

From the launching of the public validation trials in the three pilot municipalities onwards, the WeGovNow platform is expected to be gradually improved, based on the feedback received, in terms of overall look and feel and user experience, fix potentially emerging bugs and the like. Dedicated development sprints won't be required at that stage anymore, because all pilot platforms will remain stable in terms of functionalities offered to the users.

2.3. Approach towards software integration in WeGovNow

In technological regard, the WeGovNow platform environment is meant to be modular, extensible and principally open towards additional software components potentially desired to be added after the ending of the formal project duration. The general approach adopted towards integrating a given module into the WeGovNow overall platform therefore makes it extremely easy to plug in new software modules. Also, care was taken to enable deepen the level of integration of a given software component in an incremental manner as deemed useful.

Again, this general background, the minimum integration requirements are related to user authentication and security of communication, as discussed in Annex 2 “UWUM integration”. Further integration stages are related to:

1. The adoption of a set of WeGovNow core features (InputMap, AreaViewer, OnToMap Logger, etc.),
2. The on-demand bilateral (cross-component) integration with individual software components.

The first type of integration enables a coherent look and feel through across all platform components:

- a. the standardisation of common solutions, such as the map-based modules
- b. the harmonisation of the interfaces
- c. archiving information consistency with all other components, e.g. by referring to shared user information
- d. enabling cross-component features by sharing relevant information to through the common pool of user activities and open data

The goal of the second type of integration is to facilitate the implementation of the particular civic engagement services that WeGovNow platform should support, e.g. through cross-component navigation, information interlinking, combined features.

In conclusion, the incremental and pragmatic approach adopted by WeGovNow towards archiving the integration (and further development) of existing and newly developed civic engagement software modules is our answer to the need for building a flexible and modular platform for heterogeneous web-based technologies. From a sustainability perspective, this approach enables at the same time the flexible extension of the current WeGovNow platform with further software components in the future.

3. Description of the individual WeGovNow platform components

As mentioned earlier, technically speaking WeGovNow is an environment integrating various software components. In general, the capabilities provided by the overall platform are related to the use of individual components in conjunction with new possibilities provided by the smooth interworking of those components (cross-component utilisation). The following sections describe the features provided by individual components included in WeGovNow platform.

3.1. GeoKey & CommunityMaps

On one hand, GeoKey provides local communities with a web-based infrastructure to collect, share and discuss local knowledge. You can use it to setup your own mapping project with your community and to collect, visualise and analyse data using the tools of your choice.

Key Features

- **Setup up your data structures:** You decide what data your community should collect by setting up categories and attributes.
- **Decide who can access, contribute and moderate data:** Use user groups and data groupings — a predefined subset of all data contributed to the project — to define, which users can access, contribute and moderate data.
- **Add photos, videos and audio files to your contributions:** Enrich your contributions by uploading photos and videos to each of your contributions.
- **Discuss:** Comments on each contribution allows you and your community to discuss observations you have made, offer suggestions and links to other web-based material.
- **Connect your app using our API:** Use our public <http://geokey.org.uk/docs/> to build and connect applications for data collection, analysis and visualisation.

On the other hand, we have the web-map based application, CommunityMaps. This application allows users to collect new data, visualize the existing data, discuss and attach media to the contributions added to the projects previously created in GeoKey using the public REST API.

One real example could be that one municipality wants to do a survey to know where the inaccessible wheelchair local businesses are in the neighbourhood. The municipality will define the data structure on GeoKey (e.g. defining the name of business and type of business, selection criteria to assess the level of access etc.). The municipality will decide who can contribute to the project as well.

Once the municipality makes the project live, any citizen will be able to add contributions and comments to any of the contributions on the project. Indeed, users will be able to attach media files such as audio, video and pictures.

GeoKey has different extension which allow project administrators to import and export data. In this example, the municipality will be able to download the data in different formats (GeoJSON, KML or CSV), to upload data to existing projects, and additional functions which you can find on ExCiteS Group repository (<https://github.com/ExCiteS/>).

3.2. FirstLife

FirstLife is the newest software among the existing component of WeGovNow, and at current time is yet to be widely adopted outside pilots and projects. The maturity level of FirstLife is therefore inferior to others and it is only currently being finalised as service.

FirstLife addresses the need of a coordination/collaboration platform for urban activities and urban services. One of the strong aspects of FirstLife is to provide a working environment shared among the heterogeneous urban actors, such as institutions, organisations, charities, privates, etc. Moreover, FirstLife supports the self-organisation of citizens and informal groups.

Mapping, organization and documentation of distributed events

FirstLife supports a set of users' actions on distributed entities, such as events.

Users can add events on the map, linking them through a set of tags allowing to search for all the elements connected to a specific topic (for example: the name of a festival used as tag provide the possibility to create automatically a tag-map) or grouping all the events in an "initiative" that is a relation among entities. Different users can add new events on the map and decide to link them to a specific initiative, or contribute in describing and sharing real-time updates about planned, on-going or past events by adding a post in the entity card.

Users can create groups and sub-groups to manage in a light way coordination activity among the organizers and other people involved in the events (supporters, public, places owners, etc.). News and posts can be shared within each group, but keeping the information public for all the platform users.

Users can add stories to each event or group, sharing their experience about a specific event of the initiative in general, building a live and collaborative report of the events.

Users can easily retrieve all contents related to a specific initiative by selecting a period of interest on the timeline in the main interface and using FirstLife as an interactive archive of city activities.

Create groups to organize joint activities in multiple places

Citizen organizations operating in one or more neighbourhood of the city can create groups to coordinate and document the activities related to a specific project, such as managing a few urban gardens, with their schedules of gardening activities.

Each group can contain a list of places with their cards, events, news and stories. Each member of the group can add new content to the group (events or news) and visualize the map of the group.

General users, not members of the group, can explore the group content and choose to follow the group and receive notification about updates and new-shared content.

3.3. ImproveMyCity

Direct citizen-government communication & collaboration

ImproveMyCity enables residents to directly report, to their public administration, everyday local issues about their neighbourhood such as; discarded trash bins, faulty streetlights, and broken tiles on sidewalks, illegal advertising boards, but also ideas and suggestions.

The reported issues are automatically transmitted to the appropriate department in public administration to schedule their settlement more quickly and more efficient comparing to the tradition paperwork based procedures. Reporting is feasible through a friendly web based frontend that adopts a map-based visualisation, which makes reporting a user-friendly and intriguing process. The management and routing of incoming issues is performed through a backend administration infrastructure that serves as an integrated management system with easy to use interfaces.

Key features frontend

- Accurately positioned issues: By offering a map to facilitate citizens in determining the exact location of their issue with geocoding and reverse geocoding support
- Categorised on issue nature: By urging citizens to select one of the pre-specified categories reflecting the municipality departments and its internal organisation
- Easily submit new issues: By asking citizens to provide only the information necessary to locate and resolve the issue, such as title, description, location and category
- Photo enabled: By allowing to attach one or more images on the spot for describing the issue, easily with drag 'n drop
- Commented and voted: By offering the mechanisms to post comments or vote for issues that have been submitted by other citizens
- Filtering and search: By allowing to see only your own issues, or issues of specific categories or specific status or a combination of them

Key features backend (concerns the Municipality employees and administrators)

- Browse effectively: Issues are presented on the city map, as an ordered list but also in a single-issue page displaying the full set of submitted details. Only assigned issues are visible according to the user group that each officer belongs
- Track pending issues: Issues are automatically routed not only to the appropriate department but also to the inbox of the responsible officer. Detailed rules defined who gets notification and on what action
- Provide direct feedback: Provide written feedback to the citizens giving non-standard explanations for each specific case with a very friendly user interface that guides officers to take action
- Distributed responsibilities: Assign one or more officers per category and/or user group and split the administration effort across the municipality departments
- Monitor progress: Resolve issues and inform citizens by email automatically or through a progress indicator bar (Open - Acknowledged - Closed, etc.)
- Tailored to each authority/Municipality: Fully customize the system in terms of user rights, number and nature of categories, notification rules and localization settings. The hierarchy of the Municipality can be projected to the application no matter the complexity.

A typical example would be:

A citizen, on her way to work, realised that the pedestrian crossing lights next to the primary school are not synchronised correctly and more over they do not let enough time for children to cross the street safely. The citizen does not have time to go in person to the municipal police to submit an official complain, and she is afraid that this action will be very time consuming and very bureaucratic. Still she really needs to inform the officials about the issue because it is not an obvious problem and maybe no one else noticed that already. She realised the problem after many times of observing the children and the lights in that specific crossroad. Using ImproveMyCity to report the issue is very easy. She spots the road by typing the address and she fine tune the marker by zooming and drag to the correct place on map. She also uploads a photo of the pedestrian crossing and selects the Municipal Police category from the available ones. She submits the issue and immediately receives an informative email that her issue submitted successfully.

On the other hand, the Municipality employees who are set to get notifications when a new issue is submitted to their department, are aware of the new post. They read about it and unmoderated since it concerns a real problem and the status of the issue is set to acknowledged. The issue is now publicly available for other citizens to see thus promoting transparency. Moments later, the issue receives 15 upvotes and 2 comments by other citizens in the neighbourhood saying that they also are aware of the problem, but they didn't know where to report about. It proved to be an issue that concerned a lot of people but left unreported for months. All these votes, eventually

speed up the process of having the issue fixed in the next 2 days already. The status of the issue has changed in every step so that the citizens are aware of the progress. The problem is solved, and anyone could see how fast the Municipality responded.

3.4. LiquidFeedback

LiquidFeedback allows for opinion formation on a given issue with a discussion (deliberation) phase before the actual voting.

Citizens can start initiatives (proposals) and seek support among their fellow citizens. Other citizens can suggest improvements or start alternative initiatives. Considering nobody has enough time and knowledge for every issue, votes can be dynamically delegated by topic. Delegations are transitive and can be revoked at any time. Liquid Democracy is sometimes referred to as »Delegated« or »Proxy Voting«.

LiquidFeedback doesn't rely on a request commission and doesn't need a moderator. Instead, all participants gain equal rights in a scalable structured discussion process where it is ensured that minorities gain a fair share of representation and that even individuals may put up their proposals for discussion. The system is designed in such way that noisy minorities won't harm other minorities in the discussion process.

Predefined rules and timings ensure that plans on decision processes are made public in time. Decisions are made by recorded vote only, and all voting-relevant data in LiquidFeedback is made available to all participants in both human- and machine-readable form. This enables a transparent decision-making process and ensures that participants can verify the voting procedure.

LiquidFeedback doesn't ask predefined questions but encourages participants to suggest alternatives. A sophisticated voting system is facilitated to allow participants to express their opinions without necessity of tactical considerations. Its mathematical properties avoid vote-splitting and allow that similar proposals don't harm each other.

A typical example would be:

A citizen (initiator 1) suggests to allow BBQ in a given park and creates a new initiative in the dedicated subject area for this park. As there are no other initiatives on this matter, a new issue is automatically created. Other citizens support the idea. One citizen writes a suggestion and asks to consider the fire danger and to restrict the BBQ use to suitable areas. Some citizens share the concern and support the suggestion. Initiator 1 accepts the suggestions and uses the input map to mark areas in the park he thinks may be suitable.

Another citizen (initiator 2) likes the general idea but comes up with an alternative initiative. She suggests a picnic area with tables, benches and fixed fire places. She suggests a specific location in the park and marks it on the map. The alternative soon becomes as popular as the original initiative.

Another citizen suggests another location because that location would be closer to existing public restrooms. However, initiator 2, does not like the idea and therefore does not accept the suggestion. As a consequence, the citizen (now initiator 3) starts another alternative initiative and marks the proposed location on the map.

Over the next weeks all initiatives in this issue undergo a process of improvements by considering and eventually accepting suggestions. Citizens can support more than one initiative.

Only a few days before the foreseen discussion phase ends, a citizen (initiator 4) adds another alternative basically combining initiative 2 and 3 by suggesting picnic areas in both places.

Finally, all initiatives with the necessary support, e.g. 10 percent of the people interested in the issue, become voting options. The citizens can vote using a preferential order, starting with their favorite initiative. This way citizens can agree to a second choice without fearing this could harm their favorite. Likewise, citizens are not encouraged to tactical voting, e.g. exclusively voting in favor of the second choice just because they think their favorite has no chances.

LiquidFeedback's preferential voting (Clone Proof Schwartz Sequential Dropping) determines a collective preference.

3.5. Trusted Marketplace

The engagement activities conducted so far revealed useful inputs into the development of the WeGovNow pilot platform. When it comes to the Trusted Marketplace component in particular, strong concerns were voiced during a stakeholder workshop held on October 11 (M21) about the envisaged matching of demand and supply of voluntary personal support, e.g. when it comes to users offering help to older persons in activities of daily live. It was highlighted that any municipality operating the WeGovNow platform could be held legally liable for damages or fraudulent behaviour potentially occurring in the context of supportive activities/services mediated through the platform, even if these were delivered on a voluntary basis. Taking such a risk was considered unacceptable for the municipalities independent whether the platform was operated in a pilot setting during the project duration or afterwards in a mainstream setting, and even with a build-in reputation mechanism as it had been envisaged so far. The liability issue identified at that stage was assessed as representing a "show stopper" for the public piloting of the entire WeGovNow platform, and for its further mainstreaming beyond the project duration as well.

In response to this finding, it was decided to adapt the conceptual approach hitherto pursued in relation to the Trusted Marketplace. Generally speaking, it is to be avoided that a direct transaction between individual platform users is occurring through the WeGovNow Trusted Marketplace in a legally effective manner. To this end, the

WeGovNow Trusted Marketplace shall make a distinction between offerings concerning items (goods) and volunteering (services). When it comes to items, users shall be enabled to advertise freebie items to the WeGovNow user community. When it comes to personal support, local organisations, associations, small charities and timebanks shall be enabled to advertise opportunities for volunteering they offer in the community.

Trusted Marketplace in its current state, is composed of the following 3 main parts:

1. The **Enhanced User Profile** which contains:
 - a. Set of (non-required) fields to store basic user data:
 - i. Job, Education, Skills
 - ii. General interests (e.g. Photography)
 - iii. Areas of interests (geolocations)
 - b. Mechanism to link social media accounts (Facebook, Twitter, Google and LinkedIn) and let TMP to monitor user actions (e.g. when user posts a tweet, TMP collects its content). This feature is disabled by default and end-users could activate each account separately (or deactivate) through the enhanced user profile.
 - c. The personalised timeline which, under the hood, queries OnToMap Logger and actually presents all user actions from every WeGovNow component in a unified manner.
2. The **Offers & Demands** system which aims:
 - a. To allow registered users to post offers and demands
 - b. To allow associations / organisations / small charities to be promoted via Trusted Marketplace by posting a brief presentation of their purposes and their offered services (a mechanism to act as yellow pages moderated by the local authorities). This feature is yet to be implemented (see section 5.8.1 for more details)
3. The **matchmaking** system, which purpose is:
 - a. to make personalised suggestions the data used to feed the matchmaking system are collected by:
 - i. User actions/activities from each WeGovNow core component
 - ii. Textual content from linked social accounts (only when users have activated the social account linking from their enhanced profile)
 - iii. Custom data entered by users into their enhanced user profile to send notifications to registered users.
 - b. To notify users, at regular basis, about their personal actions/activities collected by each component. Users will be able to switch-off notifications from the Trusted Marketplace settings.

A typical example would be:

Citizen A is looking to collect used and new toys and clothes for the refugees' resettlement station that's being set up currently in the Municipality. He decides to post a demand in the Trusted Marketplace asking for these goods. After posting the demand, the matchmaking mechanism sends notification to users who live nearby, but also to other users that recently tweeted about this new settlement station. There's also an offer posted 3 days ago that gives away spare blankets. Since this offer is tagged with "clothing" label, the matchmaking mechanism notifies the user about this new demand. There are some comments exchanged asking for further details. The procedure went better than expected, so citizen A decides to keep his demand active one more week.

4. Setting up a WeGovNow platform instance

The setup of a WeGovNow platform instance is the setup of the core features of WeGovNow platform and to setup of the included components. In principle, a WeGovNow instance can be installed on the same environment, but for the sake of fast and easy management, because the heterogeneity of the requirements of the different components, and because there are no reasons to do so, all components and features will be provided “as service”. In this sense, the setup of a WeGovNow platform is in general, the setup of a logical or physical instance of each software.

In practice, the setup of a prototype or an instance is done as follows:

- 1) Identification of the included components, which provide a specific domain (entry point) to their instance (e.g. <https://torino.wegovnow.firstlife.org>), and the communication of the relative RSA public keys to be included in UWUM.
- 2) Collecting requirements about the styling and labelling of components: colours, logo, etc.
- 3) The setup of UWUM services: navigation, application discovery, style and authentication, the signature of each RSA public key to be used on server/server communication (UWUM <> component backend)
- 4) Testing fix of components configuration

The duration of the setup process is around two weeks, and could be considered a proper Agile sprint in terms of time, effort, goals, documentation and management.

4.1. Setup of WeGovNow core

In terms of functionalities, a WeGovNow instance could be reduced to the sets of core components, services and modules required to run WeGovNow:

- a. **LandingPage** is an independent component of WeGovNow, its setup follows the same pattern of any other component.
- b. **UWUM** is a module of LiquidFeedback providing the authorization and integration framework. Upon setup, the operator of UWUM, i.e. partner LiquidFeedback, becomes the certificate authority and announces the UWUM certificate along with the UWUM related URIs for a specific instance. All other component operators send certificate signing requests (CSRs) to the UWUM operator and inform the UWUM operator of the endpoint URIs to be used for their component. The installations for the pilot sites are customized to meet (or resemble) the municipalities’ corporate look and feel by deploying a color pair to all components using UWUM’s style endpoint. In addition, the UWUM navigation bar can be customized to display the municipality logo or coat of arms. During operation, UWUM provides all services using the endpoints described in the UWUM work report.
- c. **OnToMap** is backend component, not providing a front-end. OnToMap provides the other WeGovNow components with logging and data retrieval

services. For both functions OnToMap exploits a domain ontology that specifies an interlingua to support cross-application data logging and integration. The setup of OnToMap is described below.

- d. **TileServer** is a REST web service developed in NodeJS, it requires to be cloned from the repository and configured including the cartographical sources needed. TileServer does not need to be replicated, it can be used in any instance of WeGovNow.
- e. **InputMap** is a web application, to be setup and deployed it needs to be cloned from the repository and configured setting the map defaults (centre, zoom) and language.
- f. **AreaViewer** is a web application, to be setup and deployed it needs to be clone from the repository and configured setting the map defaults (centre, zoom), language, LoggerProxy endpoint (to get the OTM logs from the right environment).
- g. **LoggerProxy** is a REST endpoint developed in ExpressJS. It requires to be setup to read data from the right OTM logger endpoint, and to be given a token to have access to OTM logger GET endpoints.

In the definition of the consolidated architecture, we defined extra core services to enhance the overall user experience:

- h. **Enhanced User Management** is a module of TMP extending the user profile page setups with extra information fields such as social network profiles, skills and education, accessibility preferences and the user's personal data management.
- i. **Accessibility Preferences** is a TMP web module providing a wizard to setup basic accessibility parameters, which will be shared across WeGovNow components.

4.2. Setup of WeGovNow components

Setting up WeGovNow modules is strongly related to the specific technologies involved, but more importantly, it is related to the specific business processes implemented. In other words, to use a component effectively, it needs to be customised accordingly to the adoption context, for instance ImproveMyCity must be configured to manage the tickets in respect of the responsibility and accountability of the roles of the managing organisation. Therefore, these processes of setup WeGovNow components require a tight collaboration between technical teams and the adoption institutions (local authorities in case of the pilots).

4.2.1. Community Maps/GeoKey

GeoKey is the backend platform supporting the Community Maps component, both needs to be installed in the same server instance.

DNS records need to be set for both components and SSL certificates need to be generated for both domains, Apache SSL configuration file needs to be adjusted accordingly.

Community Maps

Community Maps component is the frontend element for Geokey GeoKey WGN platform therefore needs to be deployed after the GeoKey WGN platform has been set.

1. Community Maps repository needs to be cloned
2. NPM dependencies need to be installed
3. The config.js file need to be edited with URL used by
4. GeoKey platform, client's ID, also add optional Google Analytics details
5. In config.js file the *"platformConfig"* and *"uwumConfig"* variables need to be set accordingly with the URL's used by the WGN installation.
6. Finally, the production code needs to be built using Grunt.

GeoKey

In order to deploy a GeoKey WGN instance, three main installations are needed:

1. Geokey-commynitymaps extension.
2. Geokey-wegovnow extension.

GeoKey can be run on Python 2.7 only and required to install PostgreSQL and PostGIS along with other Python dependencies (those are listed on GeoKey Github)

1. Database and superuser database need to be created
2. Once the GeoKey repository has been cloned, all Python dependencies in the requirement.txt need to be installed
3. The settings.py file needs to be adjusted accordingly with new database and Super User.

Geokey-CommunityMaps extension

After installing Geokey-communitymaps, this needs to be added as an installed application in the Geokey settings.py file.

In settings.py file the custom Community Maps middleware needs to be added.

Template loaders need to be extended with a custom Community Maps.

A database migration needs to be run in order to add required Community Maps extension tables.

Geokey-WeGovNow extension

After installing Django-allauth-uwum and Geokey-wegovnow extensions, those which need to be added as installed applications in Geokey GeoKey settings.py file along with the material design support.

UWUM certificate needs to be generated, signed from UWUM authority and uploaded on the server.

In Geokey GeoKey settings.py file the custom UWUM middleware for requests and responses needs to be added and the UWUM provider settings accordingly with UWUM URL's need to be adjusted along with Geokey GeoKey redirection.

Once a user is created through UWUM registration, it can be upgraded to superuser Super User in Geokey GeoKey - if it needed.

4.2.2. FirstLife

FirstLife is crowd-based platform enabling users to share information about the city. Therefore, to setup a FirstLife instance requires:

1. to have a geographical dataset about the **urban entities**, provided to all other components through the TileServer endpoints
2. to have a definition of the “*things*” which could be attached to urban entities, in general these are **events**, **places** (in terms of activities), **news**, **extras** (insights, stories, reports) and **groups**
3. to define the related **category system**

These definitions are needed to build a shared environment for the city actors. In these regards, FirstLife setup is done by considering the common understanding of the city, through enquiring local stakeholders.

Technical setup

The technical setup of a new FirstLife instance requires the several operations at server and client side. Summarising, FirstLife backend supports multiple virtual instances.

The FirstLife backend does not require to run multiple instances, it requires the parameters of the virtual instance. In practice, the setups at backend side requires the definition of:

1. Domain: a virtual partition of FirstLife entities
2. Settings to use an OAuth 2.0 authentication server using the “Grant Code” flow
3. A category space for the supported entity types

FirstLife client is an actual instance (copy), including the specific setups of one instance:

1. The definition of the supported entities: properties, labels, icons, colours and placeholders

2. The required category space defined in the backend
3. The OAuth server endpoint and configurations
4. Defaults such as mapping start point, map tile servers, required authentication level to access to the main view

The operations of setting up both backend and client do not require more than 4 hours, including the endpoints, certificates, webserver, parameters and deployment of the changes.

All FirstLife instances are listed in the backend and frontend repositories, in configuration files.

4.2.3. [ImproveMyCity](#)

In WeGovNow project the version of ImproveMyCity (IMC) that is being used is the Joomla version. As such, it could be possible to deploy all IMC instances under a single setup with multiple installations. Instead of doing that, we decided to keep each instance completely separately. Although the overhead that creates such a decision in terms of complexity during the setup and maintenance is much higher, the benefits are multiple. First and foremost, portability. Having own hosted deployment environment for each instance (currently there are 7 instances running) means that there are no inter-dependencies and each instance could be transferred to any server in the cloud or even on-premises at each pilot site. Also, the database schema is kept tidy and future scaling becomes easier while it is easier to assign super users for decentralised administration in operation level.

Since September 2017 (M20) all instances are transferred on same IT infrastructure for easier maintenance. They are hosted under a common Virtual Private Server (VPS) in Linode Cloud in London data-centre with Long-Term-Support OS. The steps to take for each instance are briefly the following:

1. Set an A record at INFALIA's DNS server under the domain infalia.com. The naming pattern is *<instance-name>-imc.infalia.com* (e.g. sandona-imc.infalia.com, pt3-imc.infalia.com). IPv6 is also set accordingly.
2. Setup the Apache configuration file in VPS (e.g. *sandona-imc.infalia.com.conf*)
3. Use LetsEncrypt free service to get certified SSL support
4. Set the signed UWUM certificate outside web-server (extra security measure)

The above steps are done manually (although for getting LetsEncrypt certificate the automated certbot script is used which handles most of the complexity). The remaining steps deal with Joomla installation itself and setting up all the extensions that are implemented for WeGovNow as explained in Section 5.5 (ImproveMyCity)

1. Create MySQL database and grant access to new database user (the new database user has access only to this specific database and used only for IMC)
2. Install latest stable Joomla setup from the official site (www.joomla.org)

3. Install all extensions that are tailored made for WeGovNow. Currently these extensions can be found at INFALIA's public Github repository, but the plan is to create a WeGovNow-Joomla package and make it available for free in Joomla Extension Directory (JED)
4. Install the IMC-WeGovNow Lite theme based on Bootstrap and Material Design
5. The Italian translation files are pulled from Transifex service (in general, the language files are synchronised automatically in every commit that is pushed in Github).

The above steps are done semi-automatically by executing a customised shell script that handles the pulling of all software packages, including Joomla core itself. The setup of the database and the automated installation of each extension and of the theme template are controlled by the shell script as well.

Finally, each extension needs to be manually set through the graphical interface as explained in Section 5.5. The instances for the pilot sites involves some extra steps.

1. Create user groups, departments, categories, rules, notification rules according to the internal workflow of each local authority. Also, special accounts with extra privileges for employees are created.
2. Set up the SMTP settings based on pilot sites requirements.

These are the major steps to be taken to deploy an ImproveMyCity instance for the WeGovNow platform. There are, of course, more technical details (such as how the certificate is created, how the Apache configuration files are set, etc.) but the overall procedure and workflow is covered. For full details, you can refer to INFALIA's Github repository.

4.2.4. [LandingPage](#)

LandingPage is a client web application, as such the setup and deployment require to clone the project and to setup the environment variables regarding:

- Colour schema based on material design palettes
- UWUM endpoint for the navigation bar and authentication server
- AreaViewer URL to be included
- OnToMap logger endpoint

The project requires to be provided by a web server such as Apache, Nginx or NodeJS (currently Apache).

As all other components, LandingPage requires the setup of a TLS certificate for each domain, to be included in UWUM configuration; a server-side service to retrieve the authentication token (as specified by OAuth 2.0 grant code flow), LandingPages relies on FirstLife backend to retrieve the authentication token.

4.2.5. LiquidFeedback

For each environment, separate LiquidFeedback Core databases and LiquidFeedback frontend instances are set up. Every instance is accessible using a dedicated host name with its own TLS certificate. Therefore, each instance needs to be configured in the domain name system (DNS). Environment specific settings, e.g. for a specific municipality, are done per instance.

For every instance, the respective OnToMap logger instance is configured and the LiquidFeedback specific event mapping is transferred to OnToMap through its API.

To keep operating costs within the project low, all LiquidFeedback instances related to WeGovNow share the same hardware and use a common Nginx reverse proxy, although an independent setup would be possible.

4.2.6. OnToMap

All OnToMap instances pertaining to pilot sites or prototypes are managed by one single running instance; the various instances are identified by the domain name used to access them. For instance, in the current set up the virtual instances referring to prototypes 2 and 3 are named as: p2.api.OnToMap.eu and p3.api.OnToMap.eu.

OnToMap stores the Open Data and the information about logged actions relative to different platform instances in the same datasources: Open Data are stored as Linked Data in a Triple Store (Parliament, which manages data as RDF triples). Logged activity data are stored in a no-SQL database (MongoDB). Both the Parliament service and the MongoDB DBMS serve multiple OnToMap instances and do not need to be replicated when a new platform instance is created. Within each datasource, the information items belonging to different pilot instances are virtually separated and can be identified by URL, as each data item has an URL including the domain name of the instance.

In order to set up a new instance, the server running OnToMap must be configured to accept requests sent to the domain name representing the new instance; this is accomplished by editing the DNS records on the domain name provider used for OnToMap.eu, and by updating the SSL/TLS certificate used by OnToMap accordingly.

OnToMap uses Let's Encrypt (<https://letsencrypt.org/>) as Certificate Authority, therefore the update/renewal of certificates is free of charge.

OnToMap relies on an instance of Nginx (<https://nginx.org/>) as a reverse proxy for TLS termination and client certificate verification; therefore, when a new instance/domain name is added, the Nginx configuration has to be updated accordingly in order to associate the correct Certificate Authority used for client certificate verification with each domain. After correctly updating the Nginx configuration, OnToMap can recognize the domain receiving the requests (i.e., the virtual instance invoked by the WeGovNow applications) and manage logger and open data separately for each

instance. Log data collected by applications are initially inexistent; they are pushed to the OTM Logger at platform runtime. Differently, the Open Data specific to the new instance (if any) have to be imported in the Triple store, after having been converted to RDF. RDF data can be loaded in the Triple store using the Parliament web interface (see Annex 1). Notice that, at data conversion time, it is possible to specify whether data items can be visible to all OnToMap instances, or they have only to be visible in a defined list of instances. Each data item visible to a specified list of instances must have an *“applicationInstance”* property set to the name of the instance, for each possible instance; e.g., *“applicationInstance=torino”*; *“applicationInstance=sandona”*, etc. The property is not set for the data items visible in all the platform instances.

4.2.7. Trusted Marketplace

Since September 2017 (M20) all instances are transferred on same IT infrastructure for easier maintenance. They are hosted under a common Virtual Private Server (VPS) in Linode Cloud in London data-centre with Long-Term-Support OS. The steps to take for each instance are briefly the following:

1. Set an A record at INFALIA’s DNS server under the domain infalia.com. The naming pattern is <instance-name>-tmp.infalia.com (e.g. sandona-tmp.infalia.com, pt3-tmp.infalia.com). IPv6 is also set accordingly.
2. Setup the Apache configuration file in VPS (e.g. sandona-tmp.infalia.com.conf)
3. Use LetsEncrypt free service to get certified SSL support
4. Set the signed UWUM certificate outside web-server (extra security measure)

The above steps are done manually (although for getting LetsEncrypt certificate the automated certbot script is used which handles most of the complexity). The remaining steps presented below:

1. **Clone Trusted Marketplace repository** (<https://github.com/infalia/tmp>) and install dependencies via composer
\$ composer update
2. **Configuration.** Rename the .env.example file to .env and edit the application’s configuration variables. APP_KEY variable should have been generated. If not, the command below creates it automatically.
\$ php artisan key:generate
3. **Public directory.** After Laravel installation (Trusted Marketplace is based on Laravel 5.5), the web server configuration follows. The file public/index.php serves as the front controller for all HTTP requests invoking Trusted Marketplace.
4. **Permissions.** Directories within the storage and the bootstrap/cache directories should be writable by the web server otherwise Laravel will function run as expected.
5. **Database.** After creating an empty Postgresql database the following commands should be executed to create the tables and seed the initial data.

```
$ php artisan migrate  
$ php artisan db:seed
```

4.2.8. Geo-Spatial Data Repository (GSDR) for Quality Assessment

All instances of Geo-Spatial Data Repository (GSDR) of WeGovNow for Quality Assessment (i.e., the latest release <https://wgn.gsdr.gq>, prototype 3 <https://wgn-pt3.gsdr.gq>, San Dona Di Piave instance <https://wgn.gsdr.gq/sandona>) are managed by a single instance of Apache web server configured to use multiple virtual hosts. GSDR deployment is a trivial procedure. First, Apache Rivet (an extension of Apache web server, <https://tcl.apache.org/rivet/>), Spatialite (a spatial extension of SQLite3 database, <https://www.gaia-gis.it/fossil/libspatialite/index>) and a few additional dependencies (e.g., tcllib, tclsqlite, tiles.cf, igis.tk) need to be installed and configured. Second, a folder containing required files (i.e., a predefined spatial database file, JS/CSS/PNG media files, *.rvt and *.tcl files) needs to be copied to a server.

GSDR uses Let's Encrypt (<https://letsencrypt.org/>) as Certificate Authority and Freenome DNS resolver (<http://www.freenom.com>), thus SSL Certificates and domain names are free of charge. Apache Rivet allows utilizing Apache MPM prefork module. This enables us to process multiple users' requests effectively in parallel non-blocking manner. Source code is managed by Fossil version control system (<https://www.fossil-scm.org>) and available for developers through CGI interface. This allows version control, ticketing, wiki documentation.

4.3. Monitoring and support

Given the structure of WeGovNow platform, all monitoring and support activities are direct responsibility of the technical teams for their own modules.

Thus, considering that one of the most valuable feature WeGovNow project offers is the availability of an open platform in which different components can be integrated by leveraging a common set of core services, the second level support (and monitoring) is delegated to the component owners, which are clearly best qualified for this task. For what concerns the first level monitoring and support, the priority is given to the local contacts, as they have a better perception of the needs and backgrounds of end users.

In order to smooth the inevitable issues resulting from the project complexity, we are exploring a set of pragmatic solutions, in particular:

- **cheatsheet of WeGovNow environments** (see Annex 8) for the technical teams including
 - endpoints of each component, module and feature
 - references to online resources and guides

- **vademecum** (see Annex 9) for each trial sites including
 - a short description of the applicative scenarios
 - referents of the municipality and other local facilitators
 - a short description of the role of each component in the applicative scenarios
 - technical referent for each component
 - references to online resources and guides

The goal of those simple tools is to address the practical challenges of working on and with WeGovNow, without introducing extra tools, repositories or other technological solutions.

4.3.1. Monitoring

Since all WeGovNow components are currently provided “*as service*” in all existing instances, the monitoring of WeGovNow is actually delegated to each technical team in regard to their own components. As described in deliverable D3.2, each component is based on different technologies, and each team has different consolidated practices in managing their own services. Therefore, the monitoring of WeGovNow instance is a shared activity, in which all partners are responsible, and each technical team is accountable for their own component, as it is in general for the development and continuous integration activities.

4.3.2. Troubleshooting and Support

The general architecture of a support system requires a single point of contact (1st level), a technical level (2nd level) and a local support. In the piloting of WeGovNow platform, the single point of contact is a responsibility of the local municipalities, and given the previous considerations about the structure of WeGovNow instances, the technical support is a responsibility shared among all technical teams.

WeGovNow as service formally provided by the local municipality falls in the general management of the municipality services. Therefore, the implementation of the single point of contact is strongly dependent on the local protocols and structures, making not realistic to define a common policy nor approach. On the other hand, considering the piloting of WeGovNow “engagement activities”, involving local facilitator, the creation of a **vademecum** (see Annex 9) is a pragmatic solution to provide the minimum information about the pilot scope and goals, and the role and use of each component under the scope of the pilot, and the direct contact to get the support, bypassing the official service support office.

As support of this approach, should the piloting of WeGovNow should be considered as part of the design of WeGovNow-based services, requiring a direct interaction between municipalities, citizens and technological providers, rather than the introduction of a new standard service.

5. Level of integration achieved in relation to existing and newly developed components

As mentioned earlier, the WeGovNow platform includes a number of software components that have existed prior of WeGovNow already, augmented with various components that have been specifically developed within the project. Exhibit 9 provides an overview of all software components implemented. As can be seen from the table, the current prototype includes all already existing features and some new features developed so far. Each component provides a set of features to users, and to other software components to enable new integrated features. A subset of modules provides the core features of the platform (WGN Core).

The status of the integration of individual components into the overall platform can be evaluated by the adoption of WeGovNow core features (the features required to let stand-alone components to be integrated in the platform):

- 1) UWUM integration, the adoption of the WeGovNow authentication server and the correct use of OAuth 2.0 protocol
- 2) Single-sign-on: "Auto login", check of user session and seamless login in the software components as the users switch from one component to another
- 3) APIs over UWUM token, the component APIs can be invoked using an UWUM token
- 4) NavigationBar service, the component is dynamically retrieving the NavigationBar as HTML or JSON from UWUM
- 5) AreaViewer integration, the map-based summary view is done using AreaViewer, which incorporates the information from the common logger
- 6) Log User's activities, the software component is integrated with the logger and is sending information about user's actions
- 7) InputMap integration, the software component collects user's location inputs using InputMap
- 8) User Profile Service, the software component uses stores and retrieves the common user information in UWUM
- 9) Material Design, the software component follows material design guidelines
- 10) Styling service, the software component retrieves the platform styles dynamically from UWUM
- 11) WCAG 2.0 AA, the software component is compliant with the guidelines about accessibility as prescribed by WCAG 2.0 level AA
- 12) Accessibility Setup, the software component retrieves dynamically the user accessibility setups from UWUM as they are populated in enhanced user profile in Trusted Marketplace

Exhibit 9: Software components, modules and features in WeGovNow prototypes.

Software Components	Modules	Core	Features	Pre	1	2	3
Community Map	Core		GeoKey data visualisation	X	X	*	X
			Contribution management	X	X	*	X
GeoKey	Core		Project management	X	X	*	X
			SocialMedia publishing		X	*	~
			Data import	X	X	*	X
FirstLife	InputMap	X	Map-based input		X	X	X
			Edit and view mode				X
			Geocoding				X
	TileServer	X	Vector Tile Server			X	X
	LoggerProxy		Tile-based queries on OTM logger				X
	Core		Custom Maps	X	X	X	X
			Collaborative Initiatives		X	X	X
			Crowdmapping	X	X	X	X
			Area calendar				
ImproveMyCity	Core		Issue reporting	X	X	X	X
			Issue management	X	X	X	X
			Organisation management	X	X	X	X
LandingPage	AreaView	X	Map-based view		~	X	X
			Priority of sources				X
			Data interval support				X

	User's Area		Data export				
			User's fast access				
LiquidFeedback	UWUM	X	Authentication		X	X	X
			Style		X	X	X
			Application discovery		~	X	X
			Dynamic client registration		X	X	X
			Navigation bar		X	X	X
			User profile & settings via API		~	X	X
	Core		Delegation management	X	X	X	X
			Collaborative proposals	X	X	X	X
			Voting	X	X	X	X
			Independence of "subject area memberships"		X	X	X
			Storage of geo data and geo-spatial indices and searches		X	X	X
OnToMap	Logger	X	Activity logger		X	X	X
	Core		Open data	X	X	X	X
			Semantic search	X	X	X	X
			Data integration			~	X
Trusted Marketplace	Enhanced User Profile Management	X	Accessibility preferences		~	X	X
			User profile settings		~	X	X
	Core		Activity timeline			X	X
			Reputation mechanism			~	#
			Offer & Demands			X	X

			Data export				
--	--	--	-------------	--	--	--	--

(~) almost ready (*) ready but not deployed (X) ready and deployed (#) deprecated

The level of integration of the existing components is currently restricted to the core features released so far and to the relevance of the core features in the mechanisms and purposes of the software components (see Exhibit 10).

Some components introduced by the WeGovNow Consolidated System Architecture (D3.1), including the so called Trusted Marketplace, are still under development.

Exhibit 10: Level of integration of WeGovNow software components

	LP	UWUM	PSW	GK	CM	FL	IMC	LF	TM*
UWUM integration	X	X		X	X	X	X	X	X
Single-sign-on/auto login	X	X		X	X	X	X	X	X
APIs over UWUM token	X	X		X	X	X	~	X	~
NavigationBar service	X	X		-	X	X	X	X	X
AreaView integration*	X	X	-	-	-	-	X	X	X
Log User's activities		~		*	*	X	X	X	X
InputMap integration	-	-	-	-	-	X	X	X	X
User Profile service		X						X	X
UserDataStorage								X	
Material Design	X	X			~	~	X	X	X
Styling service	~	X				~	~	X	~
WCAG 2.0 AA						~	X	~	~

(X): archived, (~): work in progress, (-) not applicable, (*) not released yet,

LP: LandingPage, UWUM: Unified WeGovNow User Management is provided by LF (includes Authentication Server, User and Application Management),

PWP: Profile and settings wizard is provided by TM, GK: GeoKey; CM: CommunityMap, FL: FirstLife, IMC: ImproveMyCity, LF: LiquidFeedback, TM: TrustedMarketplace

5.1. LandingPage

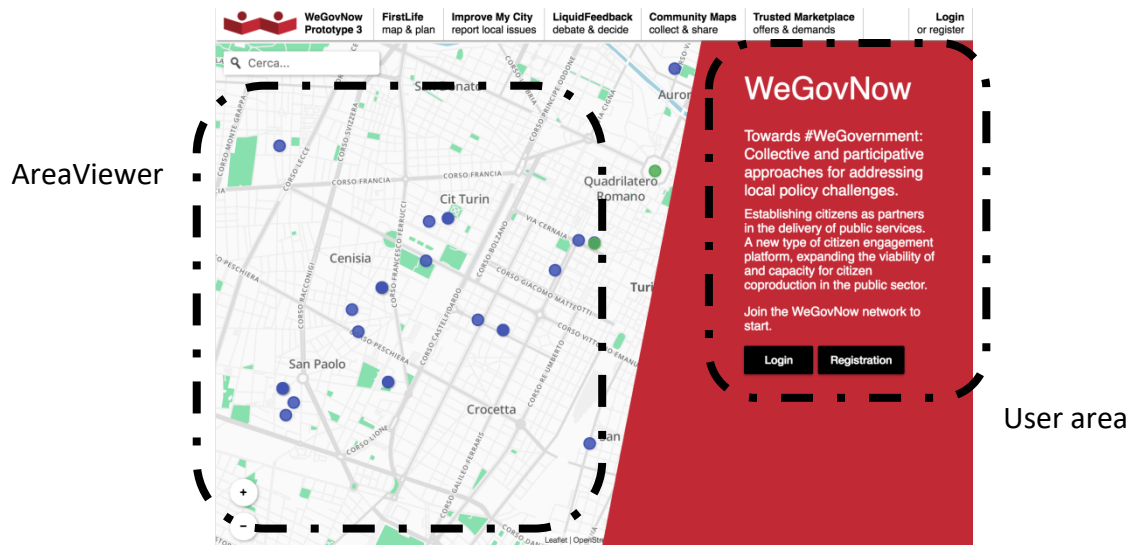
The LandingPage is the entry point of WeGovNow platforms. It has the role of presenting a summary of the status of WeGovNow platform. LandingPage is not an existing component, it was introduced during the development of the general architecture of WeGovNow (D3.1 Consolidated System Architecture).

The LandingPage has two main components (see Exhibit 11):

- AreaView**, a map-based view integrating information about the activities in the current instance of WeGovNow
- (to be implemented)* **UserArea**, a menu to collect direct links to user's last activities in the different WeGovNow components

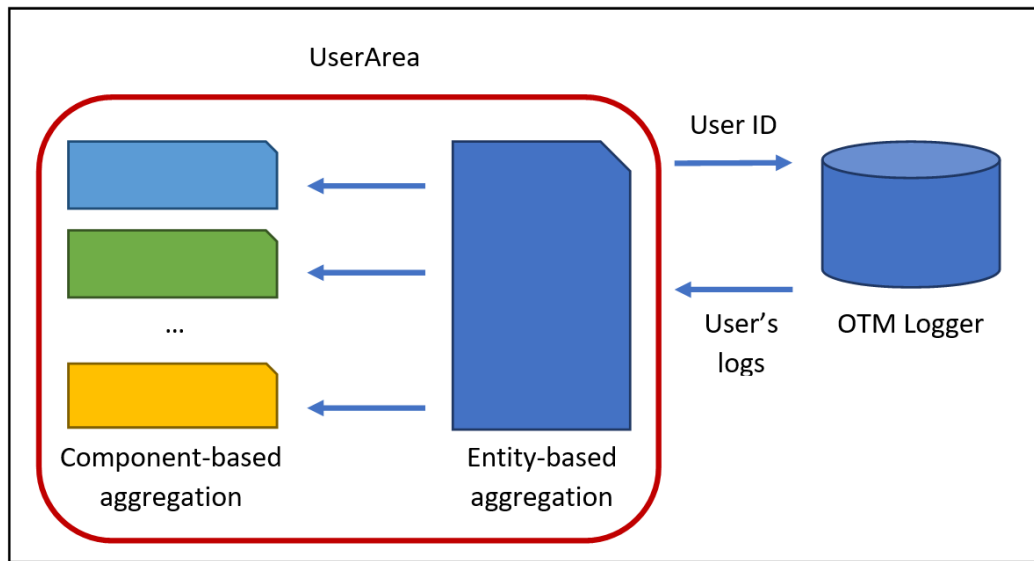
Furthermore, the LandingPage will include the styling personalisations requested for each instance of WeGovNow.

Exhibit 11: LandingPage includes the NavigationBar, AreaViewer (the map on the left) and UserArea to collect references to user's activities.



UserArea's purpose is to provide a direct access to latest entities which a logged in user interacted with. It retrieves logs of user's activities from OnToMap logger to build a list of entities structured by their origin component (Exhibit 12). The list of activities provides a direct access to the origin component (deep linking integration), highlighting if new updates are available.

Exhibit 12: OnToMap (OTM) logs of user's activities are first clustered by entity and then organised by the source of activities

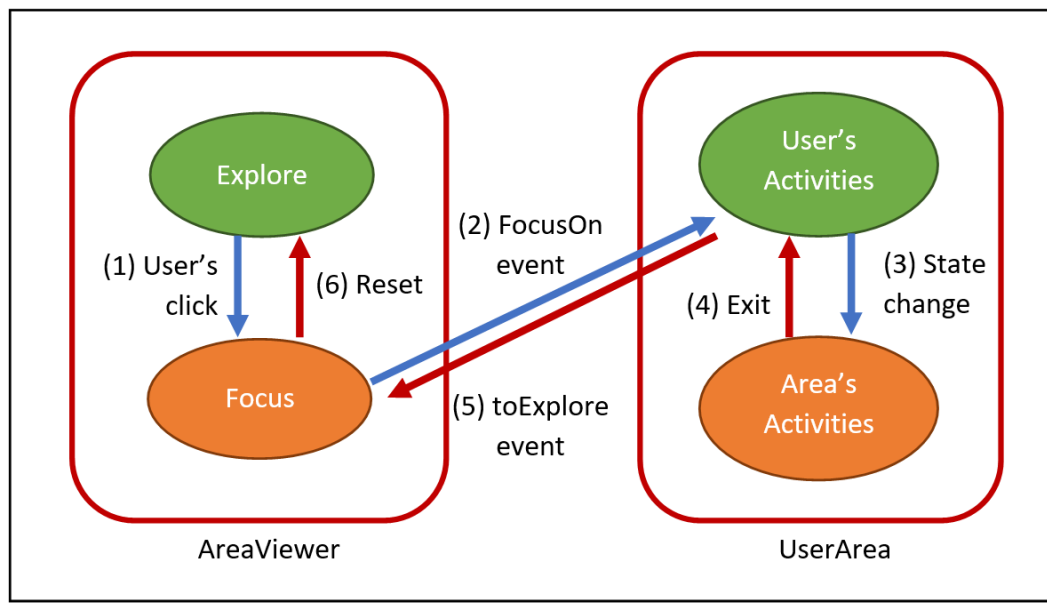


Interaction between AreaViewer and UserArea

By clicking on AreaViewer, a user's selects a "focus area" (Exhibit 13, step 1): the current area of interest of the user. Once selected an area, AreaViewer sends a message to LandingPage communicating, "focusOn" event, the details of the selected area and the list of included Point of Interests (POIs), representing the activities in that area (Exhibit 13, step 2).

In LandingPage, AreaViewer settings includes "interaction=false" preventing the display of the list of contents within AreaViewer (see section 6.12). The focus event will trigger a change of the UserArea replacing the user activities with the list of entries provided by AreaViewer (Exhibit 13, step 3). UserArea can trigger changes in AreaViewer, for instance sending "toExplore" message (Exhibit 13, step 5) to reset AreaViewer (Exhibit 13, step 6).

Exhibit 13 : An AreaViewer/UserArea interaction flow




LandingPage is implemented in Angular 4.x TypeScript.

5.2. Authentication Server

New users can register either by using an email address, existing social media credentials or local identity providers. The registration is valid for all components of a WeGovNow installation. In Exhibit 14 and Exhibit 15, login user interfaces.

The Authentication Server is provided by LiquidFeedback. The implementation shall also support user login using social media ID services. This allows participants to use already known credentials to access a WeGovNow platform - no need for another password. It is planned that Google ID and Facebook Login will be supported out of the box. The municipalities within the WeGovNow project are in the process of assessing the legal and political implications also taking the new General Data Protection Regulation of the European Union into account. Decisions to be taken by the municipalities may suggest minor re-adjustments of development priorities. Other ID providers could be added using an external login interface. Nevertheless, using social media ID services does not replace an appropriate accreditation process of the participants to ensure that no person can use multiple accounts to increase their voting weight (adherence to "one man – one vote").

For certain rights, e.g. voting privileges, a validation of an existing account may be necessary. A user can request his or her account to be validated. To ensure a proper accreditation process, the validation process for a given installation must be defined by the municipality or organization in charge of the WeGovNow installation.

Exhibit 14: UWUM login form.

WeGovNow Test Community	FirstLife map & plan	Improve My City report local issues	LiquidFeedback debate & decide	Community Maps collect & share	Login or register
-----------------------------------	--------------------------------	---	--	--	-----------------------------

🌐 文A


Login

Login name

Password

[NO ACCOUNT YET?](#) [FORGOT PASSWORD?](#) [FORGOT LOGIN NAME?](#)

About site [Use terms](#) [LiquidFeedback](#)

Exhibit 15: UWUM recovery password form.

WeGovNow Test Community	FirstLife map & plan	Improve My City report local issues	LiquidFeedback debate & decide	Community Maps collect & share	Login or register
-----------------------------------	--------------------------------	---	--	--	-----------------------------

🌐 文A

Forgot password?

Please enter your login name. You will receive an email with a link to reset your password.

Login name

[FORGOT LOGIN NAME?](#)

About site [Use terms](#) [LiquidFeedback](#)

The Authentication Server provides single-sign-on functionality for all other applications. The already implemented full-fledged OAuth 2.0 server implementation allows to share participant authorization information with other components of a participation solution, e.g. mapping or issue reporting components. Furthermore, the unified user management allows sharing of profile data and user settings across

different components of a participation solution. This allows a seamless integration of all components into a homogeneous platform. Participants can access all connected components without the need for multiple account registrations or multiple logins on different platforms. In turn, other applications can rely on LiquidFeedback as an identity provider, including a check whether an internet user has voting privileges in a given setup.

A full description of the implementation can be found in LiquidFeedback's Work report on Unified WeGovNow User Management (UWUM) development.

5.3. GeoKey and Community Maps

GeoKey provides a database-driven backend storage, together with a custom API that allows two main tasks namely interaction with data (data creation, editing, deleting) and the creation of projects which group data together. The latter is accessed via a web-based project management interface. In addition to this functionality, an API is also provided for user management, which is again enabled via a web-based frontend. The GeoKey architecture allows the data store to be accessed by any frontend application from the WeGovNow suite (web or mobile based) which can be customised making use of the APIs provided.

A flexible and stylish participatory mapping frontend, Community Maps can visualise data, compare information, and encourage conversation about the places which matter. Designed using the latest web development technologies, Community Maps offers a fast, reliable and intuitive interface. The display is clear, professional, and engaging for all screen types.

The Community Map applications tools has been developed to make use of the GeoKey public REST API and separation between GeoKey and Community Maps allows for a user interface for project management (more technical) and another for data collection/visualisation (intuitive and easy to use). This method hides the complexity of the technology behind the minimalistic and modern approach for end-user interaction. All information within Community Maps is stored in GeoKey, where the API enables storage and retrieval of data via secure SSL connection.

GeoKey

GeoKey is a web based platform for participatory mapping and is the connecting point between data collection on the one hand and data utilisation through the analysis and visualization on the other hand.

GeoKey allows project administrators (in WGN this relates primarily to municipalities but may also be applicable to key third sector organisations) to create bespoke maps in which they can define the data structure required within different categories of their choice. The type of data will differ depending on the category, symbolised by

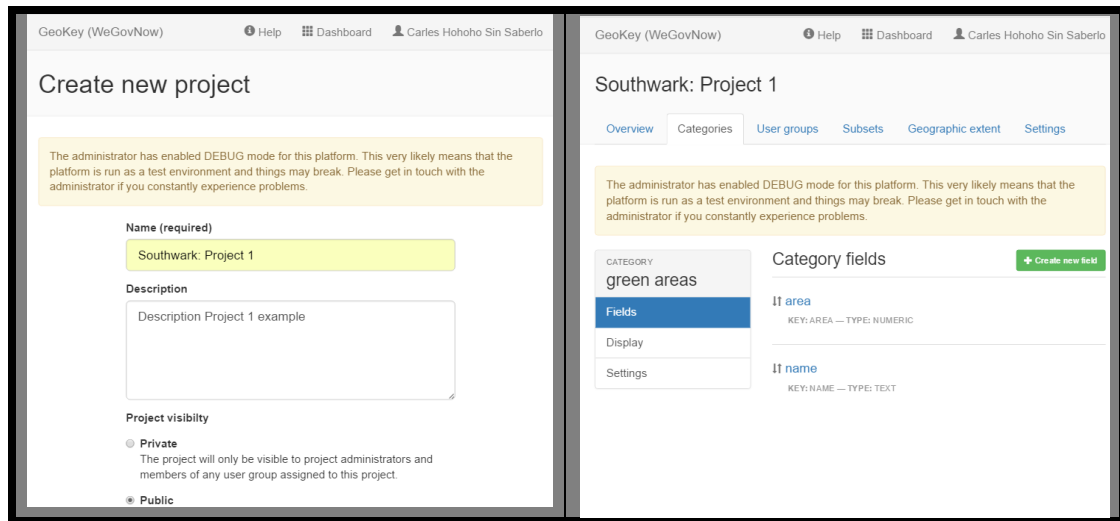
icons that can be uploaded, and the type of data (fields) should be predefined by the admins. The current field types available in GeoKey are:

- Text
- Numeric
- Date / Date & time / Time
- Selectbox
- Multiple select box

Thus, the following stages are required:

- Administrators create a project in GeoKey (see Exhibit 16). Once created, administrators can edit the project settings (see Exhibit 17)
- Within the project, administrators create a new collection that will appear on the map. They give this collection a name and (optionally) a description. You can think of this as the name being the equivalent of the table name in a database and the description a metadata description of what the table contains (see Exhibit 18)
- The administrators then give a structure ("columns") to the table – this can be ANYTHING they like – driven by what the users need. So, any combination of column types (date, number, lookup, multiple lookup, text and so forth) can be used.
- The geographic extent for an individual map is then set so that end users access the map at an extent that is relevant to local geography and context.
- Administrators can add additional settings to set the map permissions to any of the following:
 - All users can contribute. This includes anonymous contributions from users, who are not logged in.
 - All authenticated users. Users have to be logged in to contribute. Only members of contributor groups can contribute. Only users, who are members of a user groups that have been granted contribution permissions, can contribute.

Exhibit 16: Project creation and custom fields.

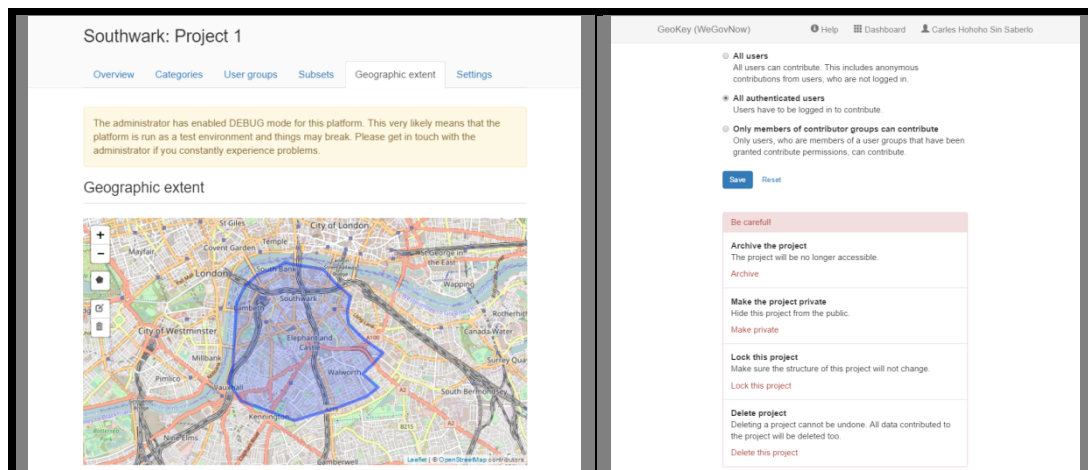


The left screenshot shows the 'Create new project' form. It includes a warning message about DEBUG mode, a 'Name (required)' field with the value 'Southwark: Project 1', a 'Description' field with the value 'Description Project 1 example', and 'Project visibility' options: 'Private' (selected) and 'Public'. The right screenshot shows the 'Southwark: Project 1' configuration page. It includes a 'Category fields' section with a 'Create new field' button. The 'Category' is 'green areas' and the 'Field' is 'area' (KEY: AREA — TYPE: NUMERIC). The 'name' field is 'name' (KEY: NAME — TYPE: TEXT).

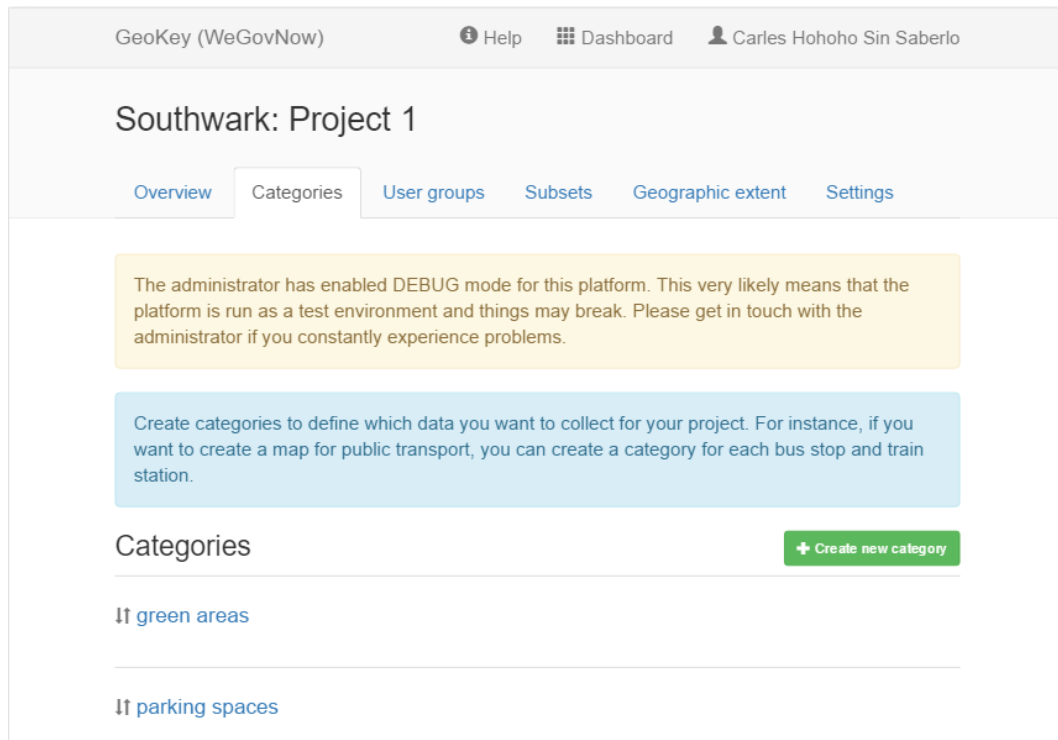
Each project thus has its own set of categories (corresponding to layers on a GIS map) and corresponding icons. The important thing is that the categories and the information they contain are defined by owners of the project (i.e. the users of the system) and created by the administrators to contain the information that the specific group of users' needs for their project. The variety of data that can be collected is illustrated by going to our live site: <https://communitymaps.org.uk/welcome>

This approach gives WGN users - whether municipalities, other organisations or local groups - the flexibility to create their own projects and decide what needs to be mapped as a group, allowing them to focus on local interests and priorities.

Exhibit 17 : Project geographic extent and settings.



The left screenshot shows the 'Southwark: Project 1' configuration page with the 'Geographic extent' tab selected. It displays a map of Southwark, London, with a blue polygon indicating the project's geographic extent. The right screenshot shows the 'Settings' tab, which includes a 'Be careful!' warning and a list of settings: 'Archive the project' (The project will be no longer accessible. Archive), 'Make the project private' (Hide this project from the public. Make private), 'Lock this project' (Make sure the structure of this project will not change. Lock this project), and 'Delete project' (Deleting a project cannot be undone. All data contributed to the project will be deleted too. Delete this project).

Exhibit 18 : GeoKey list of custom categories.

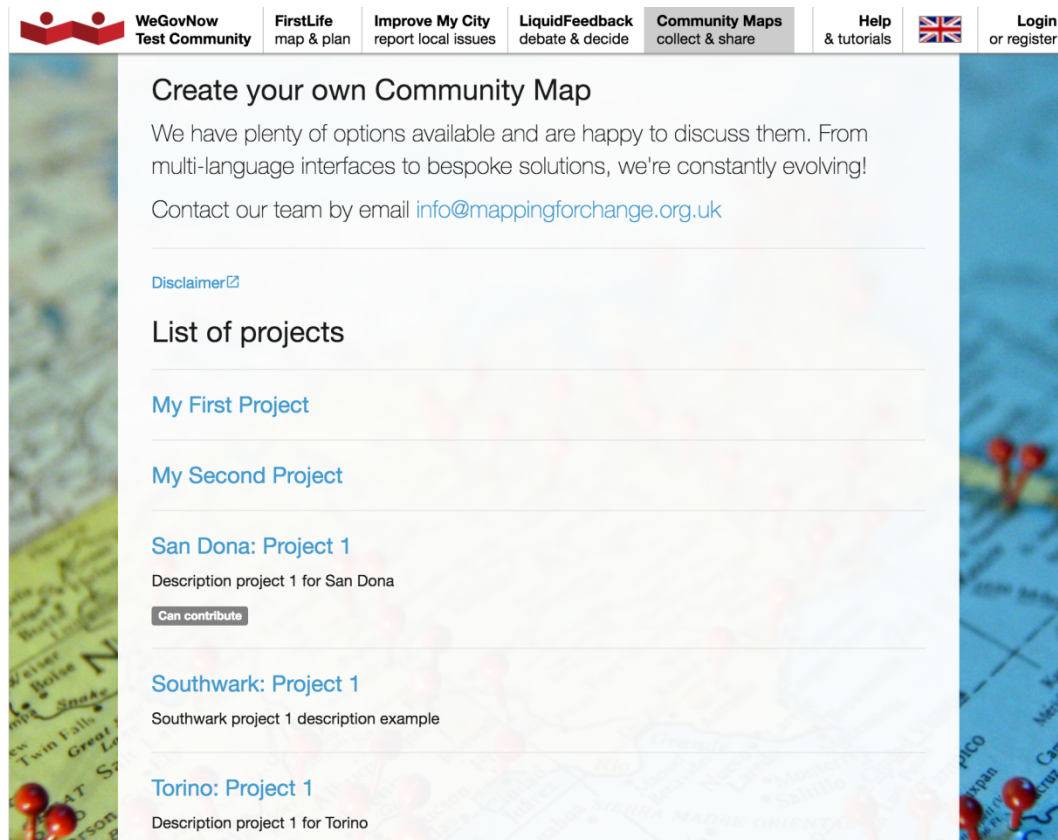
Community Maps

Once the project is live, users add data into the structure provided, using the Community Maps web mapping interface (see Exhibit 19 and

Exhibit 20). In Community Maps, users can add new data points (see Exhibit 21 and Exhibit 22), edit existing data or delete data they have contributed. They can also upload media and add comments to the data points.

There is the option of an administrator being able to moderate the data on the Community Maps site (via GeoKey) if required by the project. Further instructions on how to contribute to a project can be found here: <http://help.communitymaps.org.uk/en/add-new-contribution.html>

Exhibit 19 : List of projects on CM



Create your own Community Map

We have plenty of options available and are happy to discuss them. From multi-language interfaces to bespoke solutions, we're constantly evolving!

Contact our team by email info@mappingforchange.org.uk

[Disclaimer](#)

List of projects

[My First Project](#)

[My Second Project](#)

[San Dona: Project 1](#)

Description project 1 for San Dona

[Can contribute](#)

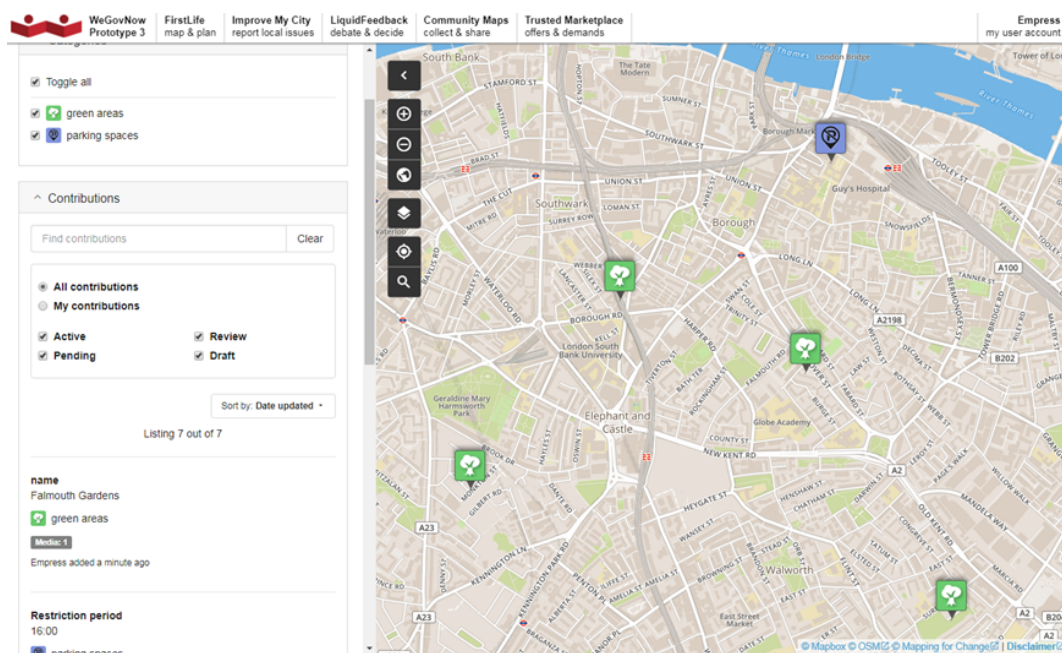
[Southwark: Project 1](#)

Southwark project 1 description example

[Torino: Project 1](#)

Description project 1 for Torino

Exhibit 20 : Display existing contributions for one project on CM.



WeGovNow Prototype 3

FirstLife map & plan

Improve My City report local issues

LiquidFeedback debate & decide

Community Maps collect & share

Trusted Marketplace offers & demands

Empress my user account

☒ Toggle all

☒ green areas

☒ parking spaces

Contributions

Find contributions Clear

☒ All contributions

☐ My contributions

☒ Active

☒ Pending

☒ Review

☒ Draft

Sort by: Date updated

Listing 7 out of 7

name

Falmouth Gardens

☒ green areas

Media: 1

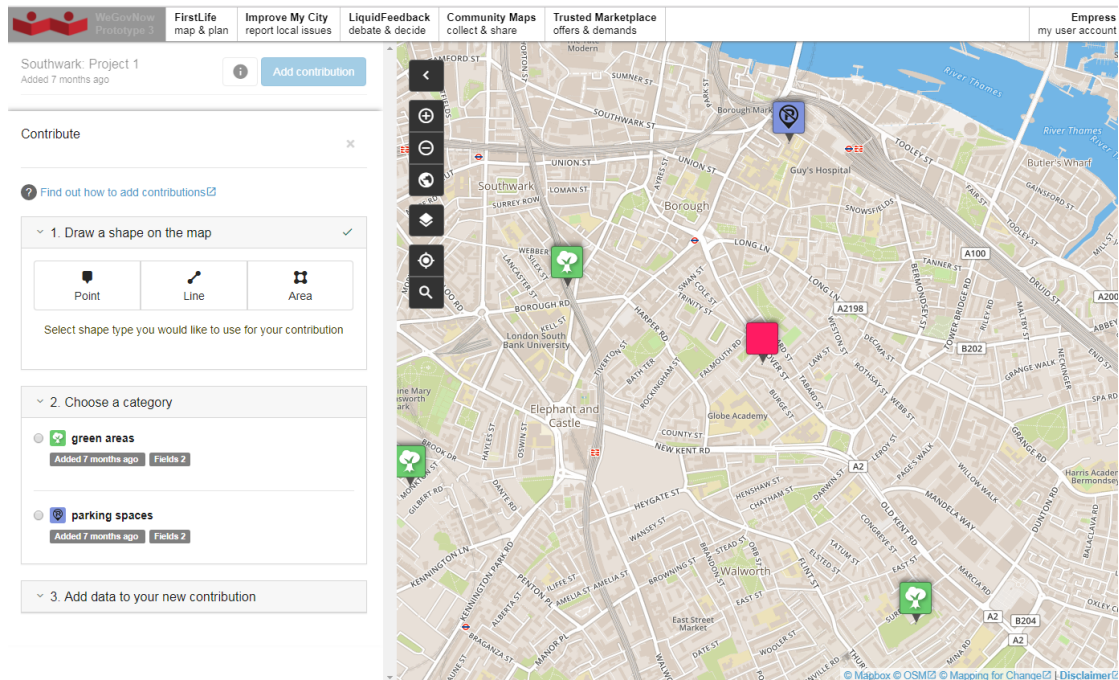
Empress added a minute ago

Restriction period

16:00

☒ parking spaces

Exhibit 21 : Add new contribution on CM. Geometry type definition, and categories.



Southwark: Project 1
Added 7 months ago

[Add contribution](#)

Contribute

[Find out how to add contributions](#)

1. Draw a shape on the map

Point Line Area

Select shape type you would like to use for your contribution

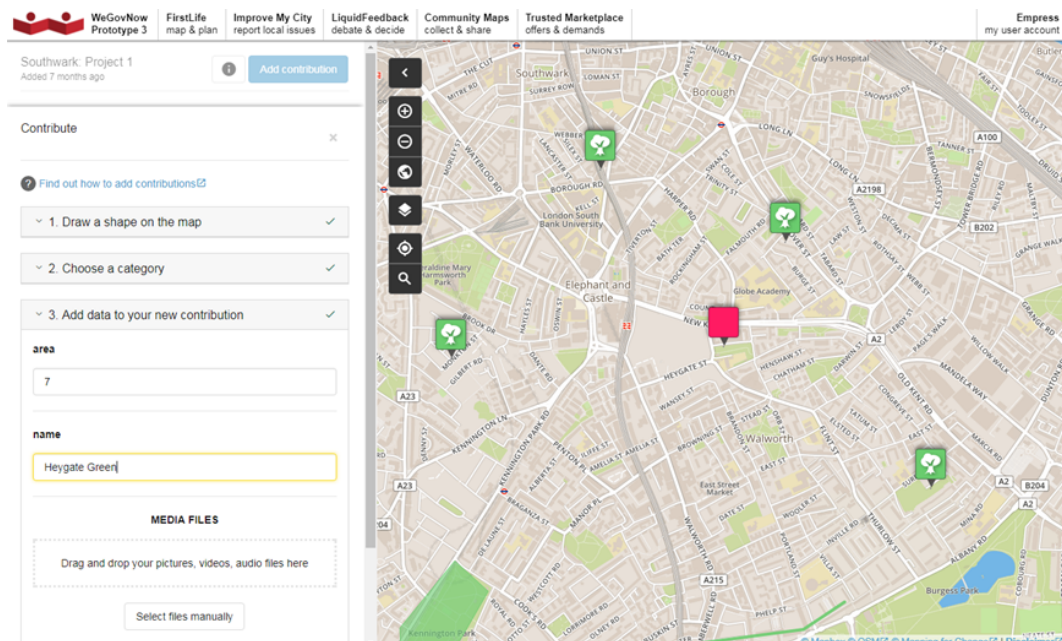
2. Choose a category

☒ green areas
Added 7 months ago Fields 2

☐ parking spaces
Added 7 months ago Fields 2

3. Add data to your new contribution

Exhibit 22 : Add new contribution on CM. Fill the fields for the chosen category.



Southwark: Project 1
Added 7 months ago

[Add contribution](#)

Contribute

[Find out how to add contributions](#)

1. Draw a shape on the map

2. Choose a category

3. Add data to your new contribution

area

7

name

Heygate Green

MEDIA FILES

Drag and drop your pictures, videos, audio files here

Select files manually

5.4. FirstLife

FirstLife is a map-based solution (see Exhibit 23, Exhibit 25) enabling users to interact with five types of entities: places, events, news, stories, and groups.

Each entity is an aggregation of geo-referenced contents related to a specific geographical unit or area (see Exhibit 24), at multiple scale: building, city block, neighbourhood, district, city, etc.

Exhibit 23 : FirstLife map-based view: pie charts representing clusters of markers.

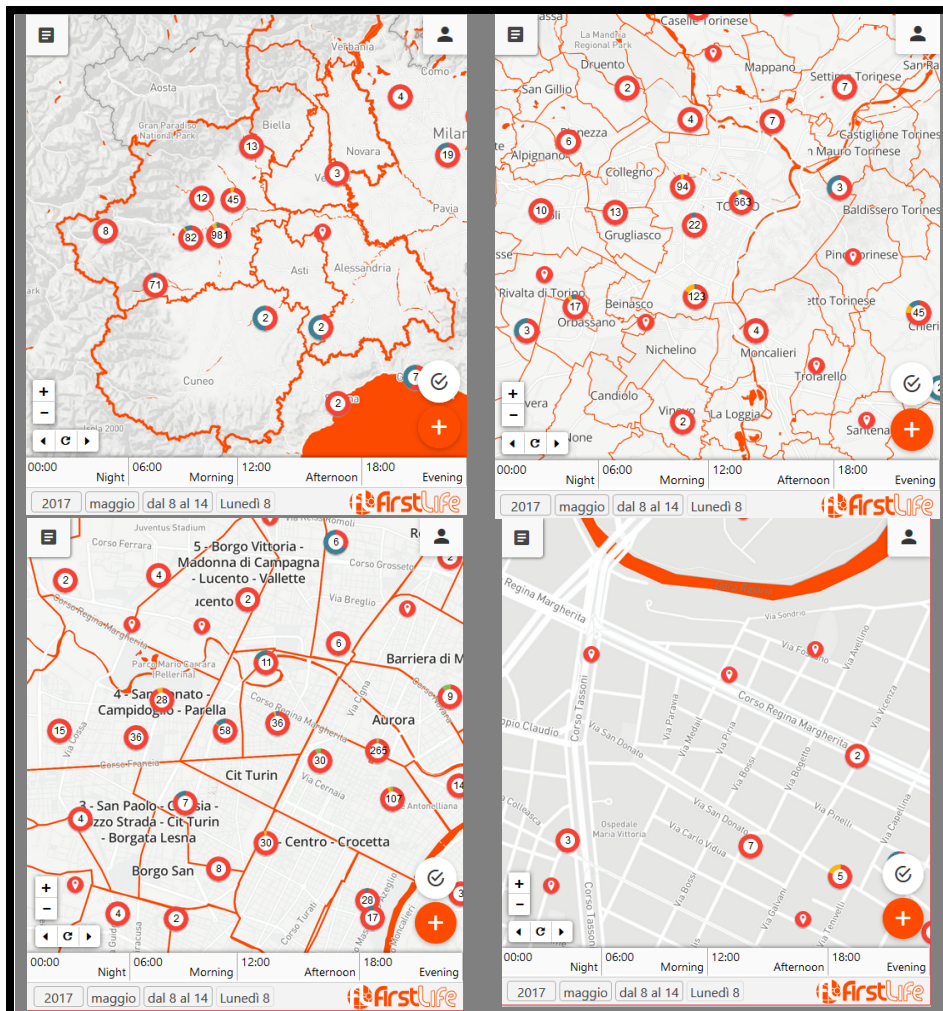


Exhibit 24 : FirstLife uses InputMap to collect location input from users.

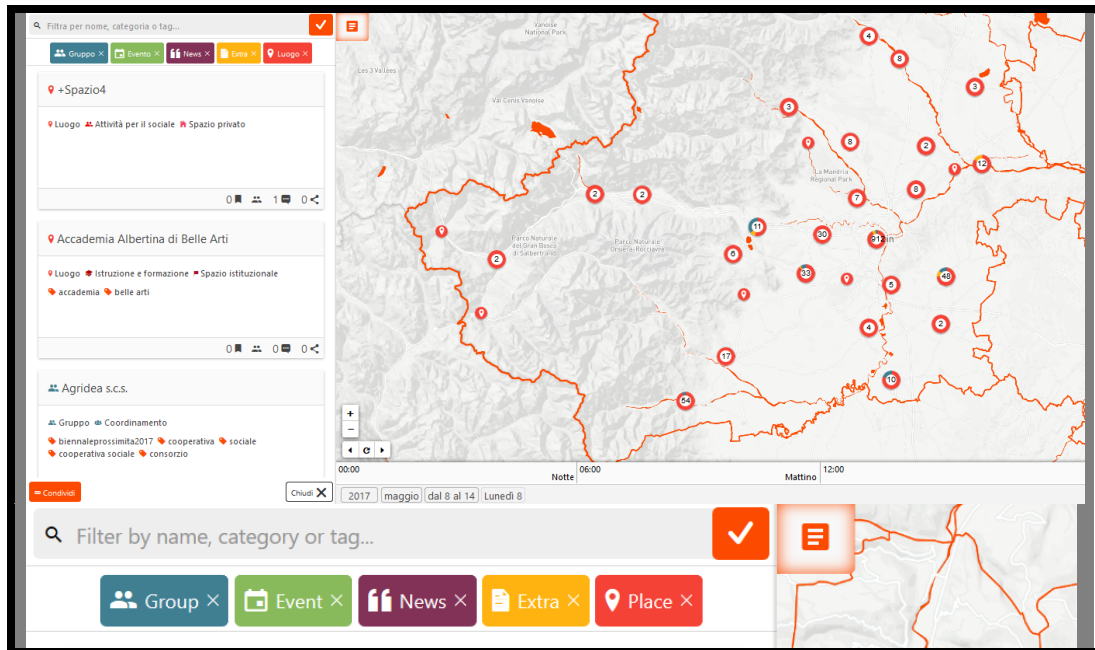


Users can:

- Explore existing contents by opening the entity cards clicking on the markers or selecting the element of interest in the wall
- Contribute to existing contents, by adding a new post or sub-entities to an existing card
- Add new entities on the map.

The integration of FirstLife in the WeGovNow prototype platform is implemented in terms of Angular 1.X release candidate 5 and Leaflet 1 release candidate. This version is currently being ported to Angular 2 and Leaflet stable versions.

Exhibit 25 : FirstLife provides a map-based view and wall to explore crowdsourced content. FirstLife implement a multi-dimension filtering system: time, tag, entity type and categories.



The development of FirstLife within the scope of WeGovNow was focused on three aspects:

1. To reach a level of maturity filling the gap with other WeGovNow components, providing comparable performances, including basic features of other social networks
2. To improve FirstLife accessibility as usability, considering that as map-based application there are strong limitations
3. To provide WeGovNow shared map-based modules.

In terms of maturity of FirstLife and readiness for the upcoming trials, the platform was completely rebuilt both backend and frontend to overcome performance issues of the map-based interface, and to enable a fast, richer and reliable configurability of the platform to support the pilots. Moreover, FirstLife was missing some basic features which are not innovative but highly expected from users, such as:

- Share and export of views and entities via email, permalink and iframe
- List-based view with alphabetical and time-based ordering features, full text search and filtering
- Calendar-based timeline, based on known granularities such as weeks, month, etc.
- Bottom-up relations between entities, new “initiative” feature

In terms of accessibility and usability, FirstLife html was extensively fixed introducing translation in attributes, and missing attributes such as “title” and “alt”. Many

elements and modules were completely redesigned to be more usable on mobile devices, to lower views complexity, and to solve counterintuitive mechanisms.

The principal role of FirstLife as technology was to provide the common map-based components. So far AreaViewer, LandingPage and InputMap are independent projects branched from FirstLife know-how, solutions and source code. Even though, these projects are not strictly components of FirstLife, they were part of FirstLife development tasks, and the reason to not integrate them as part of FirstLife was specifically to enable their use within all other WeGovNow components.

5.5. ImproveMyCity

ImproveMyCity can stand both, with or without a map. However, using a map-based UI to display the issues gives a more intuitive user experience to the end users and a better overview when browsing issues. Users can easily select at any time - if they prefer - a list or card based representation of the issues by clicking the appropriate button. The information on both views is the same. But card-display focuses mostly on photos. In any case, depending of personal preference different layout modes are supported. Different views are displayed by Exhibit 26: on the left is the list view and on the right, is the card view.

Each issue contains further details and more importantly, the citizens are able to see the timeline of each issue. When it is submitted (and by whom according to the settings), who (either responsible employee or Department name), when and what action has been taken as shown in Exhibit 27. Submitting a new issue by registered users is a very quick and easy procedure, as displayed in the form in Exhibit 28.

It should be noted that users are able to edit their own issues ONLY if the status of their issue has not yet changed by the administrators. This allows a time window to correct typos, add new photos, etc. The business logic (e.g. when an issue is editable, when it becomes public, whether commenting is allowed or when comments are kept privately between issuers and Municipality, who gets notified and when and many other actions and rules) is set graphically in the backend / administration side.

Exhibit 26 : Browsing on existing issues. Different views, same content and information. Google Maps are replaced by AreaViewer (OSM based) in the final prototype.

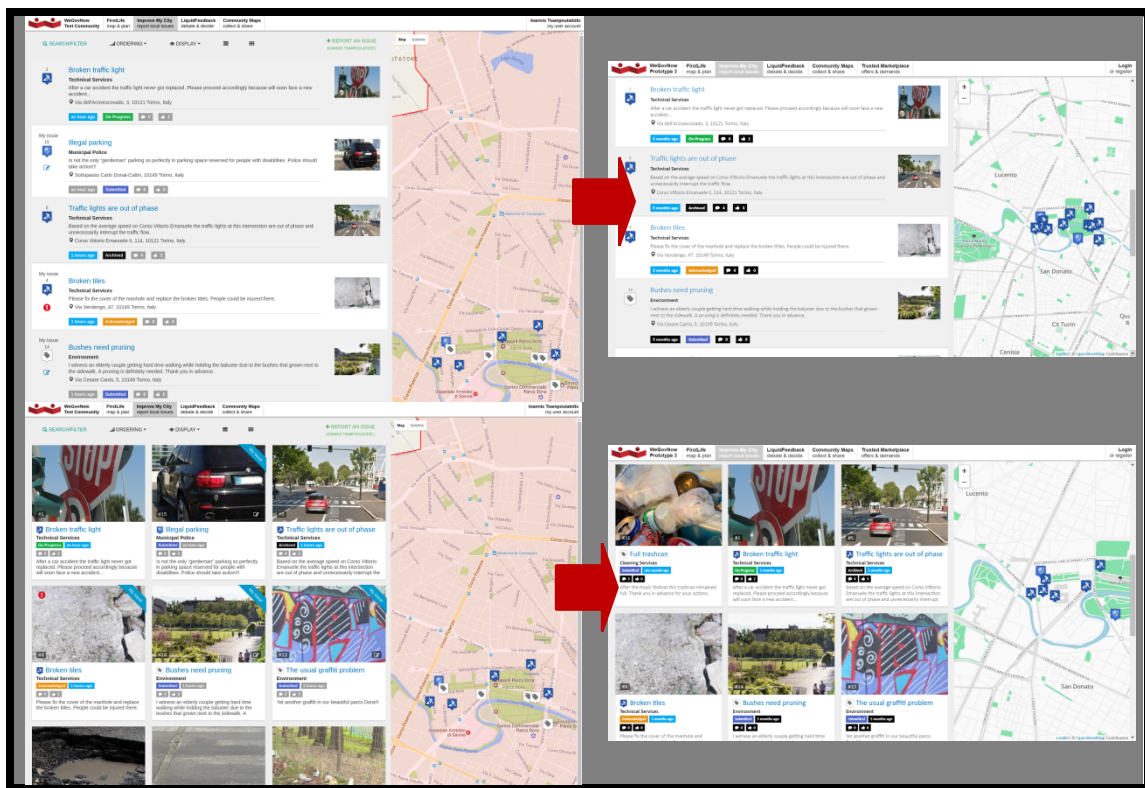
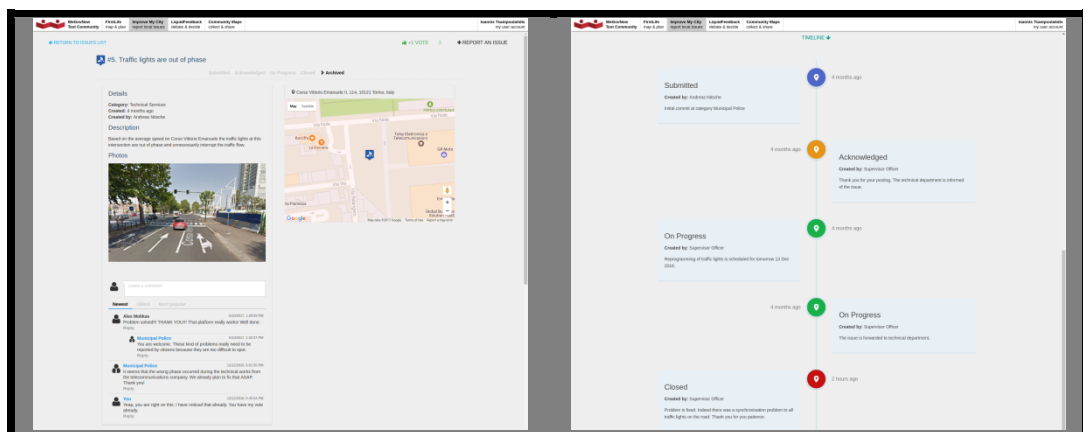


Exhibit 27 : Issue details (showing also commenting and voting functionality) and detailed timeline that promotes transparency and direct citizens – Municipality interaction. Google Maps is replaced with OSM based map



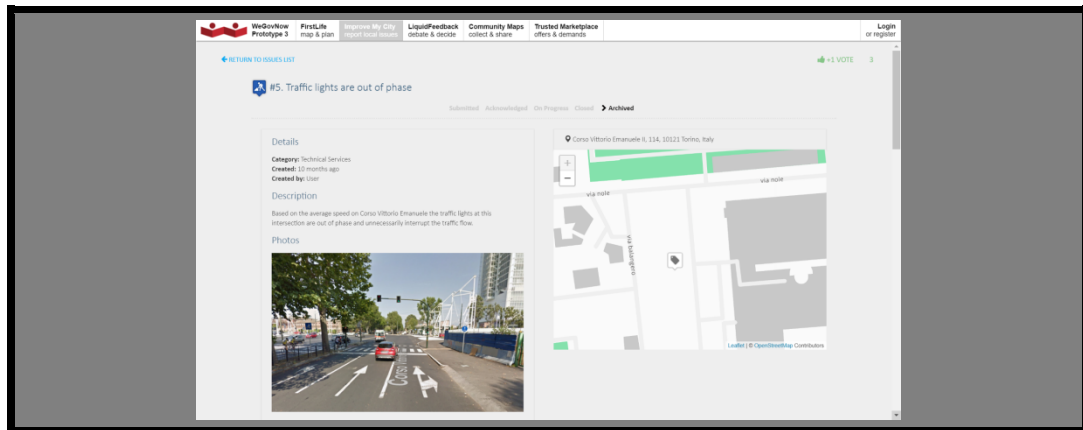
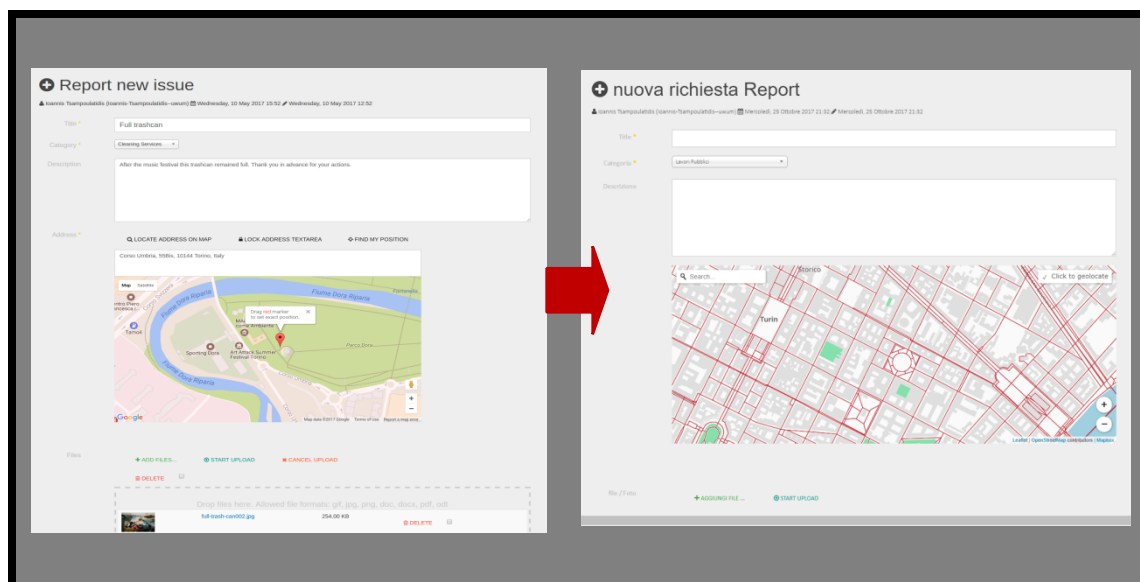


Exhibit 28 : Reporting a new issue in WeGovNow with ImproveMyCity. GoogleMaps are removed and in the final prototype, InputMap integrated and is being used to post end edit issues.



ImproveMyCity is not a monolithic application. It is a highly modular application that can expand or reduce the complexity and features on demand. More specifically, it is composed of:

- Core component
- Map module
- Filtering module
- Search plugin
- Notifications plugin
- Categories extra fields plugin
- Reporting expansion
- Workflows expansion

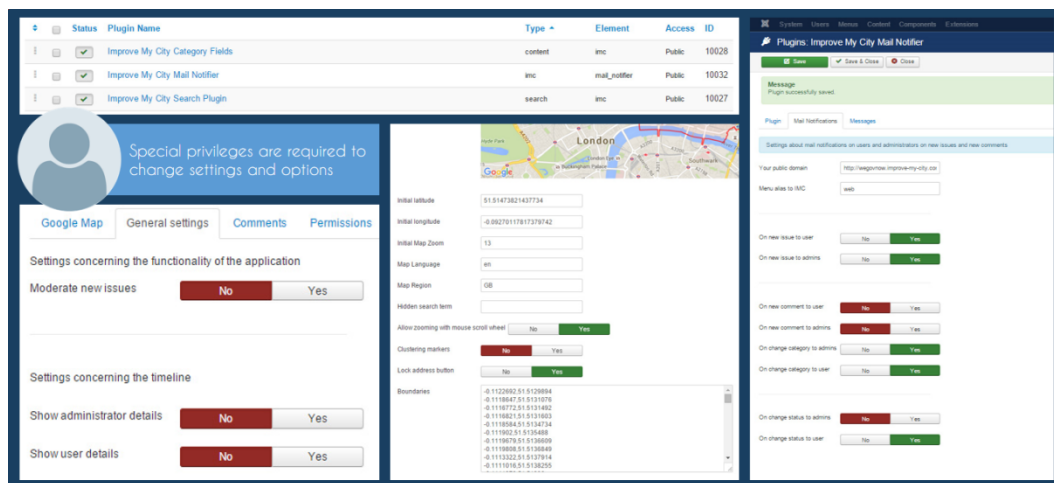
Plus the following which are developed specifically for WeGovNow

- UWUM plugin (connects to UWUM OAuth2.0)
- WeGovNow NavigationBar module (displays the common navigation bar)
- WeGovNow IMC Lite template (the Bootstrap based Material Design theme)
- OTM notifier plugin (listens to triggered actions and sends logs to OnToMap Logger)
- AreaViewer module (to use AreaViewer in any template position)
- InputMap custom field (to allow using InputMap as field to any form)

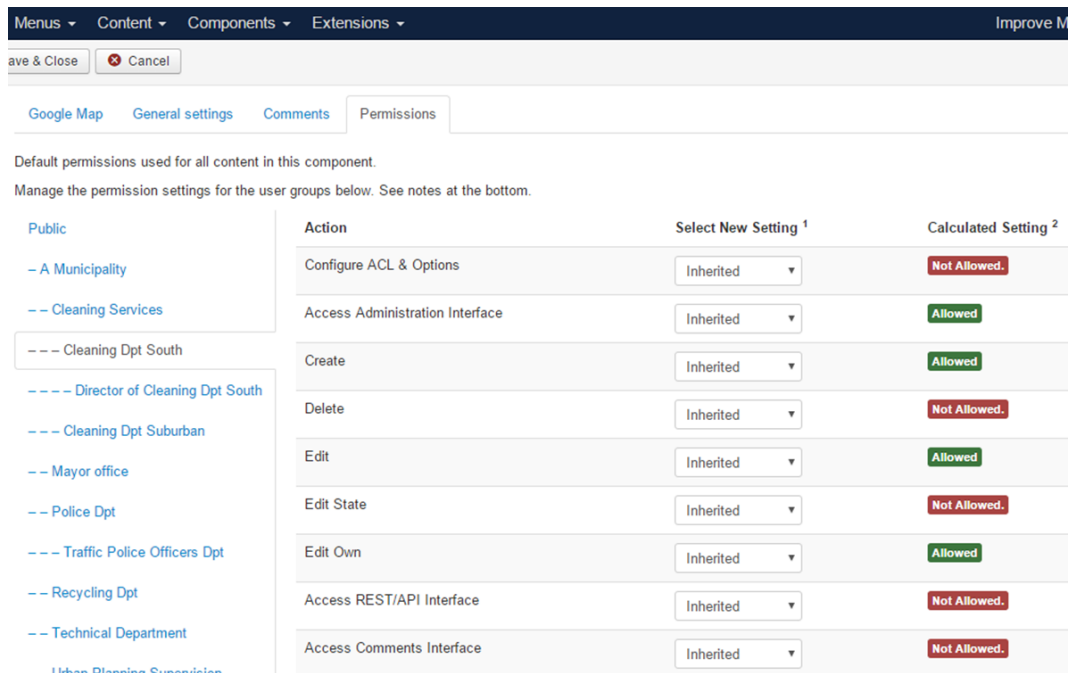
Each and every module/plugin has its own settings and preferences, but they operate and communicate with each other flawlessly. The following exhibit (Exhibit 29) shows some of the settings that are set graphically by the administrator.

A municipality's organisational diagram is easily imported no matter how complex it is. For each department, office and generally any hierarchical level, it is feasible to define permissions. Each employee belongs to one or more groups/departments and inherits its permissions. Each department (e.g. Technical Department) can have one or more corresponding categories (roads, parks, etc.) and its assigned officers get notified automatically according to the rules. All these parameters are already set for San Dona instance, and Torino and Southwark will follow.

Exhibit 29 : ImproveMyCity is highly modular and parametric to cover different Municipality needs according to their workflow and complexity.



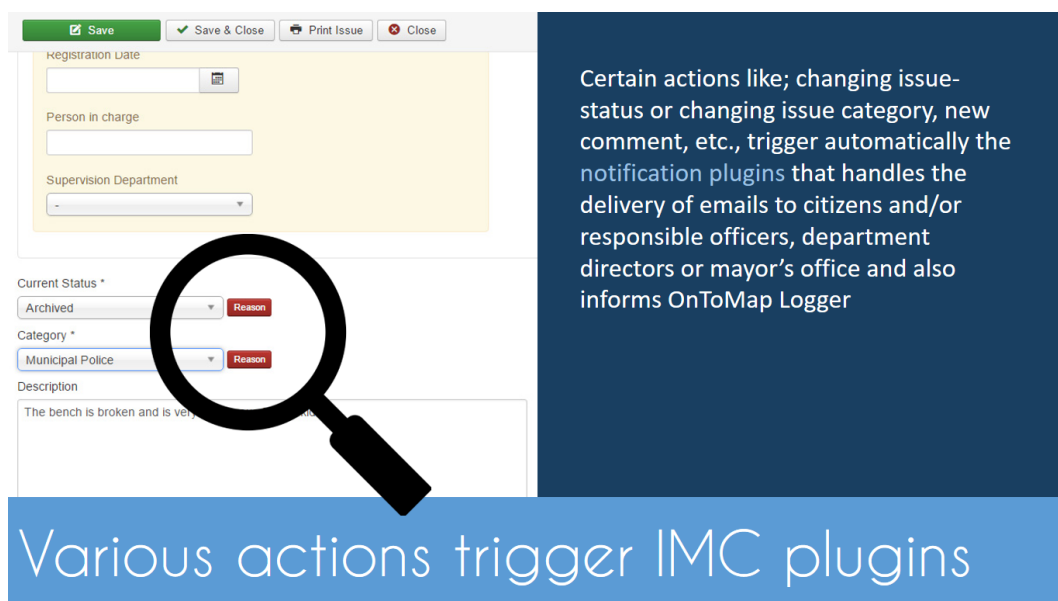
*Exhibit 30 : ImproveMyCity permissions according to the Municipality Hierarchy.
“Who does what, who sees what”.*



Action	Select New Setting ¹	Calculated Setting ²
Configure ACL & Options	Inherited	Not Allowed.
Access Administration Interface	Inherited	Allowed
Create	Inherited	Allowed
Delete	Inherited	Not Allowed.
Edit	Inherited	Allowed
Edit State	Inherited	Not Allowed.
Edit Own	Inherited	Allowed
Access REST/API Interface	Inherited	Not Allowed.
Access Comments Interface	Inherited	Not Allowed.

Administrator employees in ImproveMyCity are facing a very simplified User Interface without hassles and complexities. They see only the issues that concern them and the only actions that they take is to change a dropdown menu and type a descriptive response. All others (notifications, logs, updates, etc.) are handled automatically by the application.

Exhibit 31 : ImproveMyCity triggers most actions automatically reducing the effort of employees to the bare minimum.



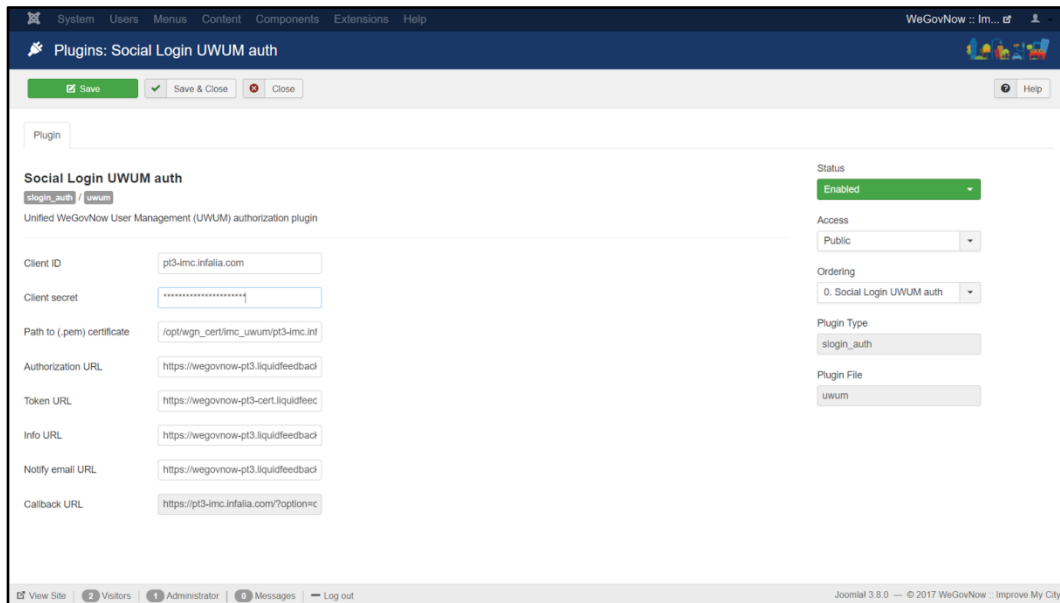
Certain actions like; changing issue-status or changing issue category, new comment, etc., trigger automatically the notification plugins that handles the delivery of emails to citizens and/or responsible officers, department directors or mayor's office and also informs OnToMap Logger

Various actions trigger IMC plugins

For a detailed explanation of the backend/administrator side of ImproveMyCity, visit <https://wegovnow.improve-my-city.com/handbook/administration/general.html>

Keeping the same modular design approach, every new module and plugin follows the official Joomla guidelines and best practices. For example, the path to certificate (pem file), callback URL, client iD, and every other parameter needed to communicate and integrate UWUM are all set graphically in the backend as depicted in the following Exhibit 32, making the setup a very friendly process.

Exhibit 32 : Login UWUM auth plugin has its own GUI settings.



The real added value is that we provide to the thousands of power-users of the Joomla open source community, ready solutions that they could use immediately to their web applications and be connected to UWUM authentication server, send actions to OnToMap Logger, display the unified WeGovNow Navigation Bar and use AreaViewer and InputMap and thus become part, as future external components, of the WeGovNow platform.

All these Joomla compatible plugins, modules and fields could be installed the same way as any other free or commercial Joomla extension. The next step is to package and make them available to Joomla Extension Directory (JED) and become available to million of users that are using the Joomla CMS. The benefits are two-fold. On one hand we promote WeGovNow to a huge community, on the other hand we give out-of-the-box technical solution for easy integration. The latter would also help to further extend WeGovNow even when the project is over.

ImproveMyCity: Actions taken towards accessibility

ImproveMyCity on final prototype also includes accessibility fixes and updates, based on suggestions and comments that reported by Funka. Some of the major

implementations towards a more accessible user experience that took place in ImproveMyCity are.

Conformance to WCAG2.0 AA guidelines

Conformance to WCAG2.0 level AA is a time-consuming procedure and demands a lot of effort. It concerns not only the core IMC but also all IMC modules, plugins, fields and of course the theme. The result of this effort is to pass most of the accessibility audits as set by Lighthouse 2.3.0 of the Google Chrome development tools with a total score of 94/100. Still there are more to be done, such as colour contrast and correct the dynamic filtering inputs and checkboxes. Conformance and best practices appliance is an ongoing work that will continue. At the time of this writing (M21) the following seven audits are passed:

Elements Use Attributes Correctly

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- [accesskey] values are unique.
- <audio> elements contain a <track> element with [kind="captions"].
- Image elements have [alt] attributes.
- <input type="image"> elements have [alt] text.
- No element has a [tabindex] value greater than 0.
- Cells in a <table> element that use the [headers] attribute only refer to other cells of that same table.
- <th> elements and elements with [role="columnheader"/"rowheader"] have data cells they describe.

ARIA Attributes Follow Best Practices

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- [aria-*) attributes match their roles.
- [role]s have all required [aria-*) attributes.
- Elements with [role] that require specific children [role]s, are present.
- [role]s are contained by their required parent element.
- [role] values are valid.
- [aria-*) attributes have valid values.
- [aria-*) attributes are valid and not misspelled.

Elements Describe Contents Well

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- The page contains a heading, skip link, or landmark region.
- Document has a <title> element.
- <frame> or <iframe> elements have a title.
- Form elements have associated labels.
- Presentational <table> elements avoid using <th>, <caption> or the [summary] attribute.
- <object> elements have [alt] text.

- `<video>` elements contain a `<track>` element with `[kind="captions"]`.
- `<video>` elements contain a `<track>` element with `[kind="description"]`.

Colour Contrast Is Satisfactory

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- Background and foreground colours have a sufficient contrast ratio.

Elements Are Well Structured

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- `<dl>`'s contain only properly-ordered `<dt>` and `<dd>` groups, `<script>` or `<template>` elements.
- Definition list items are wrapped in `<dl>` elements.
- `[id]` attributes on the page are unique.
- Lists contain only `` elements and script supporting elements (`<script>` and `<template>`).
- List items (``) are contained within `` or `` parent elements.

Page Specifies Valid Language

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- `<html>` element has a `[lang]` attribute.
- `<html>` element has a valid value for its `[lang]` attribute.
- `[lang]` attributes have a valid value.

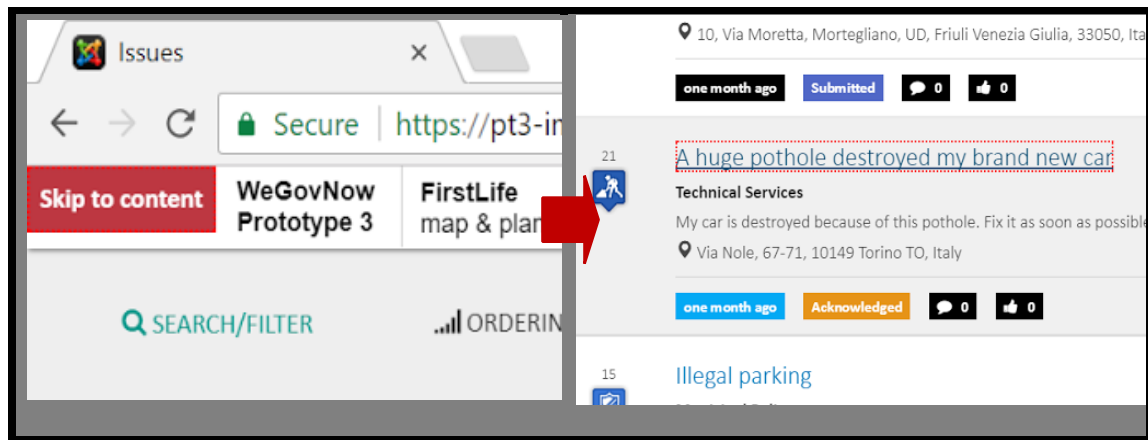
Meta Tags Used Properly

- Screen readers and other assistive technologies require annotations to understand otherwise ambiguous content.
- The document does not use `<meta http-equiv="refresh">`.

Make ImproveMyCity keyboard friendly

Applying some of the best practices of Design4All approach, ImproveMyCity has become more friendly for people using only the keyboard for browsing. The following Exhibit 33 depicts the “Skip-to-content” link that appears when user presses the TAB key in the keyboard. Also, links are outlined with a highlighted ring to help visually impaired users to focus better.

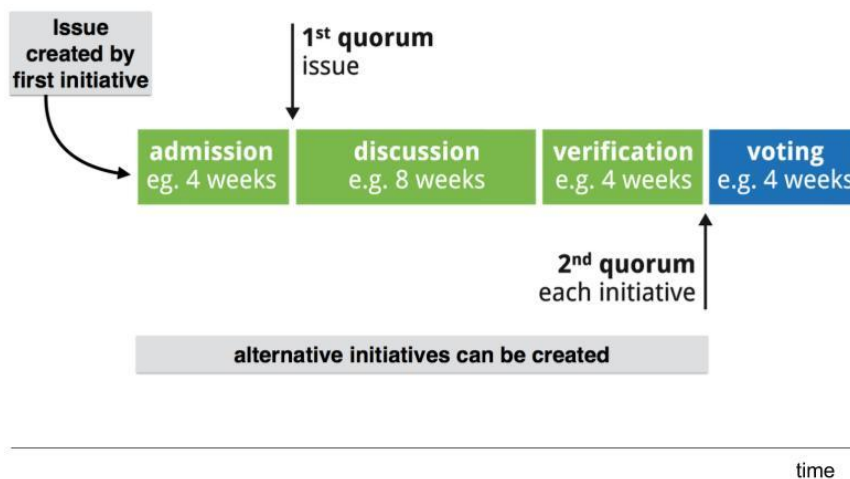
Exhibit 33 : Applying best practices for accessibility in ImproveMyCity.



5.6. LiquidFeedback

The overall function of LiquidFeedback within WeGovNow is opinion formation consisting of a deliberation process and a voting phase to determine a collective preference (Exhibit 34).

Exhibit 34 : The LiquidFeedback proposition development process.



Copyright 2017 Interaktive Demokratie e. V.

Citizens can start initiatives (proposals) and seek support among their fellow citizens using the input map provided by FirstLife (partner UniTo) and the new WYSIWYG editor (rich text editor) of LiquidFeedback (Exhibit 35). Other citizens can suggest improvements or start alternative initiatives.

Predefined rules and timings ensure that plans on decision processes are made public in time. Decisions are made by recorded vote only, and all voting-relevant data in LiquidFeedback is made available to all participants in both human- and machine-readable form. This enables a transparent decision-making process and ensures that participants can verify the voting procedure.

Exhibit 35 : Creating a New Initiative in LiquidFeedback with WYSIWYG Editor and WeGovNow InputMap.

Add another park bench

Enter your proposal and/or reasons:

B I U ↻ Ⓢ T₁ T₂ T₃ ✖ " ≡ ≡ ≡ ≡ ≡ ≡ ≡ ⇌ ↺ ↻

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin ac erat volutpat, pretium lorem quis, molestie leo. Cras laoreet ligula sit amet
ligula blandit, eget luctus metus ultricies. Nam ac ligula velit. Quisque semper malesuada elementum.

• Nulla facilisi.
• Mauris turpis purus,
• semper sit amet leo et,
• congue egestas diam.

Aenean nec tortor nulla. Aenean risus neque, gravida nec elementum et, varius eu urna. Mauris lacinia ipsum interdum, finibus dui eu,
fringilla odio. Mauris euismod neque arcu, non consequat ex pretium at. Quisque tempus nisi eget lorem pulvinar mollis. Integer gravida
nunc at dolor tincidunt gravida. Vivamus tempor non arcu vitae scelerisque. Nam tristique ante eu est commodo elementum.

You can change your text again anytime during admission and discussion phase

In the context of WeGovNow, LiquidFeedback already added geospatial functionality and published a core update as a prerequisite for the integration (see “Work report on pgLatLon in Appendix 6.5, an alternative to PostGIS” and “Second work report on extending the LiquidFeedback Core” in the appendix). The adoption of material design is in progress.


For prototype 2, LiquidFeedback focused on finalizing the integration features and check the feasibility. The input map was integrated, a new rich text editor was added, and LiquidFeedback's UWUM server was extended with an application discovery endpoint providing access to the list of both static and dynamically registered applications. The finalization of the frontend will be the central aspect of prototype 3.

For prototype 3, LiquidFeedback finished adopting material design. However, the layout is still subject to minor changes, e.g. based on feedback from experts, such as partner Funky for accessibility, and end users' comments.

All previous developments regarding functionality were incorporated in the LiquidFeedback Core Release 4.0.0 which was published under the MIT/X11 open source license on 2017-09-29. The release includes:

1. **Geospatial support.** Units, areas, initiatives, suggestions and member data structures have been extended to support storage of related geospatial data using the PostgreSQL extension pgLatLon.
2. **Dynamic admission quorum.** The previous issue admission system with quora based on subject area memberships has been replaced. The enhanced issue admission system achieves a better user experience as quora are based automatically on the actual number of active users without the need for users to manually choose subject area memberships. Optionally, the dynamic adaption of the quorum can also take the actual number of open issues into account. This can prevent LiquidFeedback from being flooded with too many issues in discussion phase at the same time and thus improves the ability to handle very large participation groups. More information about the underlying algorithms has been published in the Liquid Democracy Journal.
3. **Absolute issue and initiative quora.** The possibility to demand a configurable absolute number of supporters has been added in addition to the quora based on user activity.
4. **Support for integrated OAuth 2.0 server.** Data structures for a fully fledged OAuth 2.0 compatible authentication server have been added. This enables single-sign-on solutions for third party applications.
5. **Usage of PostgreSQL's JSONB datatype to store member profile information and settings.** The storage of member profile information and settings has been updated to use JSON documents. Therefore, it is possible to use arbitrary member profile fields depending on the needs of different application scenarios.
6. **Revised snapshot system.** The snapshot system has been revised. The possibility to store intermediate snapshots for a defined period of time has been added. As a prerequisite for the dynamic admission quorum, snapshots may contain more than one issue now. Additionally, required supporter counts are stored per issue.
7. **Extended event logging.** The event logging has been extended to record most user and system activities. This allows creation of more comprehensive timelines.
8. **Revised system to store information about accepted terms of use.** An explicit storage facility for information about accepted terms of use or similar legal rules has been added.

Exhibit 36 : Area view in LiquidFeedback (example from the Torino instance).



Cosa posso fare qui?

Voglio imparare di più su LiquidFeedback

- [discussione strutturata](#)
- [4 fasi di una decisione](#)
- [delega del voto](#)
- [voto di preferenza](#)

Utenti registrati (14)

ALL UNITS IN TUTTE LE FASI

Parco Dora Parchi Proposta civica #3

Discussione (9 giorni 16:53:21 rimasti)

i4: Panchine verdi
by Luigi Sanasi
2 sostenitori

Parco Dora Parchi Proposta civica #1

Verifica (11 giorni 21:40:17 rimasti)

i1: Le panchine dovranno essere dipinte del colore blu che identifica la città
by Andreas Nitsche
6 sostenitori

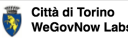




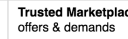

i2: Le panchine potranno essere colorate in diversi modi, tramite il contributo di artisti locali e un concorso di idee
by Axel Kistner
3 sostenitori

i6: Creiamo sedute alternative e non panchine
by Federica Ebanista
3 sostenitori

Parco Dora Parchi Proposta civica #2

Exhibit 37 : Issue view of finished issue in LiquidFeedback (example from the Torino instance).

Secure | <https://torino.liquidfeedback.net/issue/show/9.html>

» [Parco Dora](#) » [Parchi](#) » Project selection #9

Project selection #9

Projects up to 100.000 euro

- ✓ 1. Ammissione (raggiunto)
- ✓ 2. Discussione (completato)
- ✓ 3. Verifica (raggiunto)
- ✓ 4. votazione (completato)

Completato con un vincitore

Cosa posso fare qui?

Questa tematica è chiusa

Vontanti

Iniziativa alternativa

i16: Classical concerts in Parks
by Andreas Nitsche
Raggiunto >50/100: 3 Si (100%), 0 No (0%), 0 Astensioni (0%)

Iniziativa alternativa in confronto con il vincitore:

i19: Vernisages in schools
by Federica Ebanista

i17: Art fairs on Thursday afternoons
by Alberto Rudellat

i15: Changing Exhibitions
by Claudio Schifanella

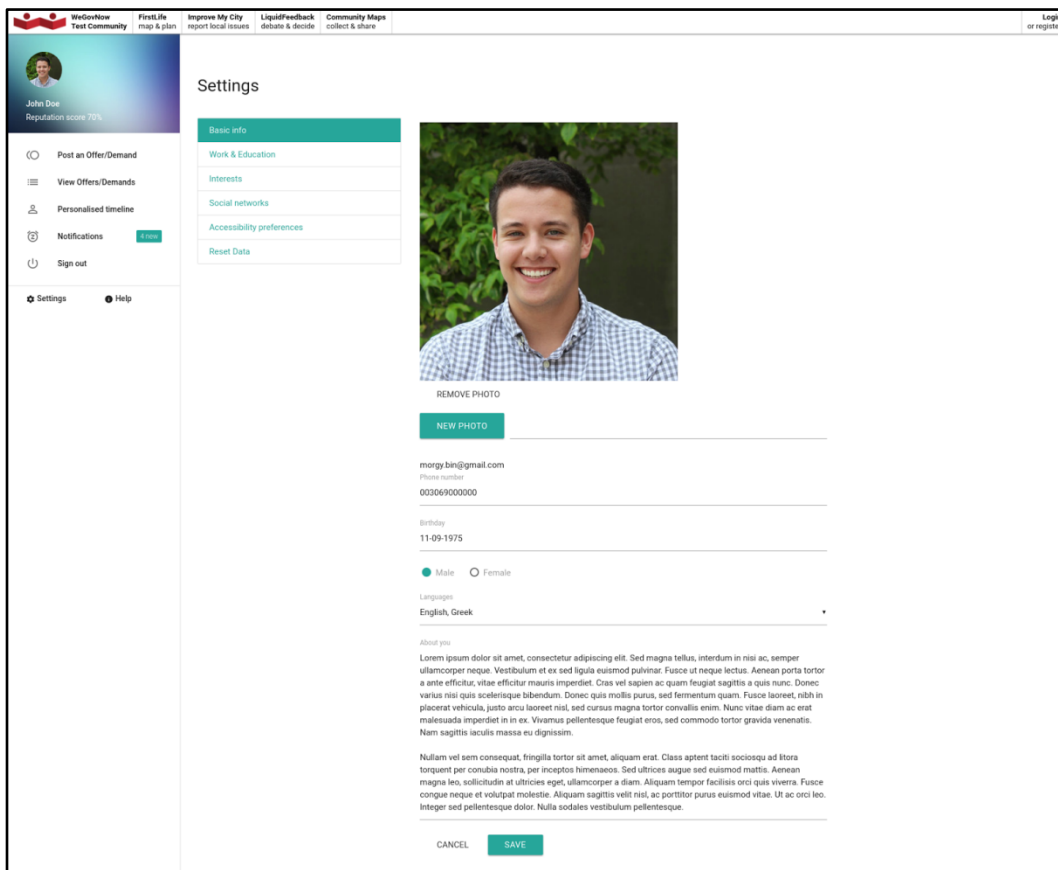
i18: Authors reading from their books
by Andreas Nitsche

Dettagli

5.7. Enhanced User Profile Management (module of Trusted Marketplace)

Enhanced User Profile Management is currently part of the Trusted Marketplace (TM) component and its purpose is to provide a single entry-point to user's preferences and other data (such as user interests) across WeGovNow platform. Enhanced User Profile Management extends the profile settings in UWUM (see Exhibit 38), including the possibility to edit all information stored about the user by all WeGovNow components. Moreover, it connects to social networks on user consent in order to get extra information that could help to better matchmaking suggestions.

Exhibit 38 : The Enhanced User Profile Management page, extending the UWUM profile settings.



The screenshot displays the 'Settings' page for a user named John Doe. The page layout includes a top navigation bar with links like 'WeGovNow', 'FirstLife', 'Improve My City', 'LiquidFeedback', and 'Community Maps'. On the left, a sidebar contains navigation options such as 'Post an Offer/Demand', 'View Offers/Demands', 'Personalised timeline', 'Notifications', 'Sign out', 'Settings', and 'Help'. The main content area is titled 'Settings' and features a list of settings categories: 'Basic info', 'Work & Education', 'Interests', 'Social networks', 'Accessibility preferences', and 'Reset Data'. A large profile picture of John Doe is shown, with 'REMOVE PHOTO' and 'NEW PHOTO' buttons below it. Below the photo, the user's email 'morgan.ban@gmail.com', phone number '003069000000', and birthday '11-09-1975' are listed. The user's gender is set to 'Male' and languages to 'English, Greek'. A 'SAVE' button is at the bottom right.

User Management, besides basic info, it includes other sections such as:

- Work & Education (see Exhibit 39)
- Personal interests (see Exhibit 40)
- Social network accounts (see Exhibit 41)

It should be noted that under the hood, the system provides more than 60000 predefined values which are selected automatically while user types at certain fields. Users are also able to type free text.

Exhibit 39 : Section of user's "work and education".

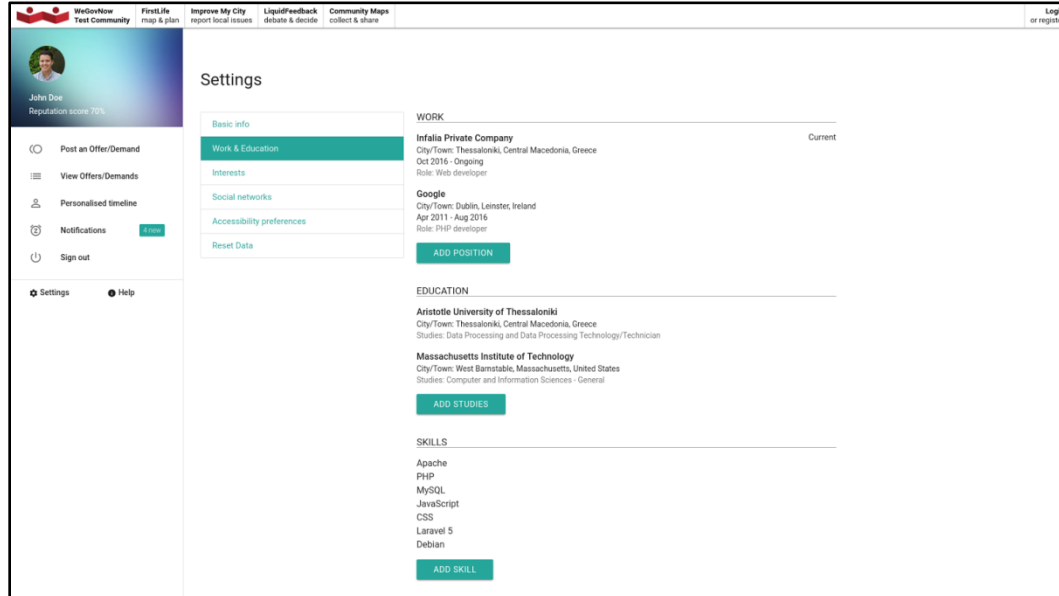
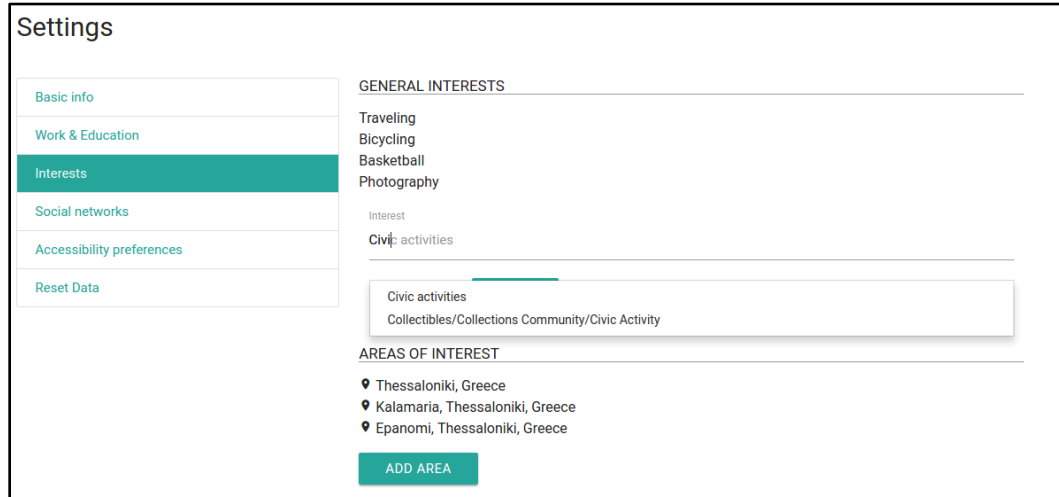


Exhibit 40 : Section of user's interests.

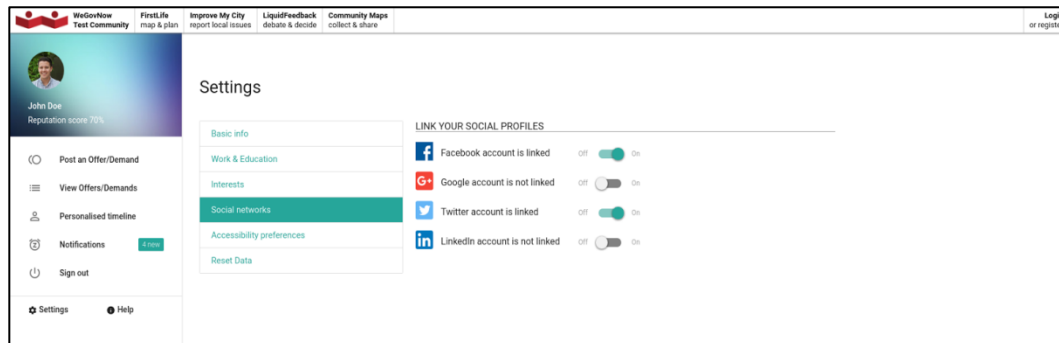


Social Networks account

Through the Enhanced User Management users are able to link into their user profile various social networks as depicted in Exhibit 41.

The purpose of linking social accounts is to collect more information about the users automatically and thus enhance the matchmaking mechanism. At any time, users are able to unlink their accounts.

Exhibit 41 : Link to social accounts.



Social networks linking could be enabled or disabled per implementation instance.

5.8. Trusted Marketplace

The Trusted Marketplace is designed and implemented from scratch according to the requirements and needs of the pilot use cases. Besides the matchmaking of users-to-events and users-to-users and the handling of personalised notifications (based on the match-making suggestions), Trusted Marketplace also incorporates features that are not available by the other core components. These features include:

- Enhanced user profile management
- Mechanism to handle demands and offers.
- Graphical representation of the personalised timeline by aggregating OnToMap Logger user actions/activities in a friendly interface.
- A friendly dashboard for easy overview of all the above.

The Trusted Marketplace matchmaking engine receives input from three different sources.

1. Indirectly from any other core component (including the “Offers & Demands”, part of Trusted Marketplace)
2. Directly from the users through the “Dashboard” and their personal profile, preferences and interests that are explicitly declared by them.
3. By social networks, on user’s consent, through the “Social Media Linker & Collector” of Trusted Marketplace.

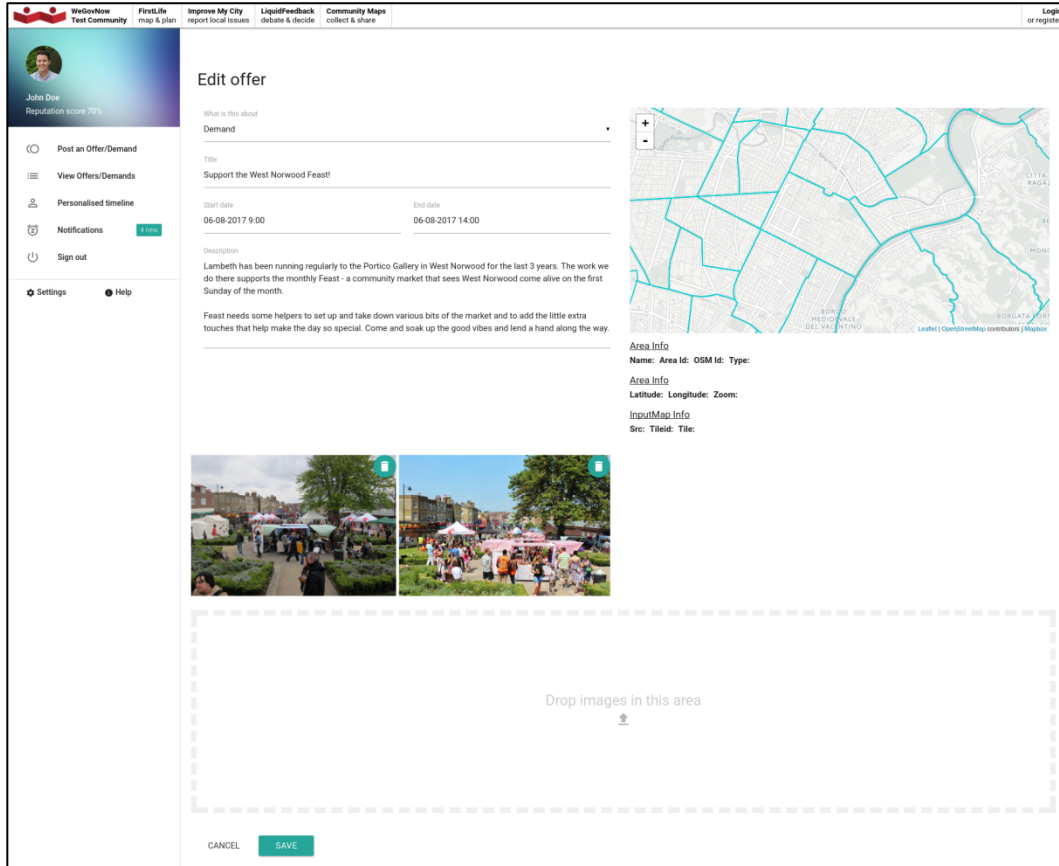
Based on the above, the Trusted Marketplace will notify users on actions and events that will match their preferences and interests.

The other major role of Trusted Marketplace is the implementation of “Offers & Demands”. The following Exhibit 42 depicts its integration with InputMap when users are posting a new offer.

In addition, Trusted MarketPlace is linked with OnToMap. The following Exhibit (Exhibit 43) displays the personalized timeline of user actions as pulled from

OnToMap. Currently, user actions of TM are pushed to OnToMap, and aggregated actions from every component are displayed inside TrustedMarket (pull actions from OnToMap).

Exhibit 42 : Trusted Marketplace integrates InputMap already

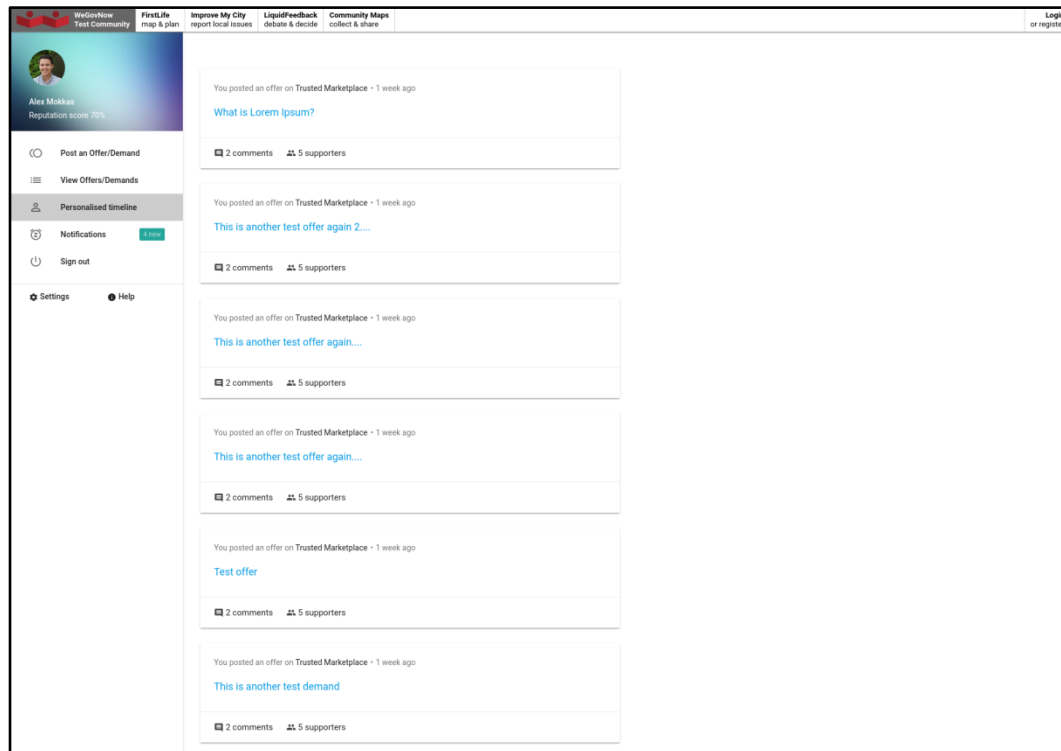


The screenshot displays the 'Edit offer' page in the WeGovNow platform. The interface includes a top navigation bar with links like 'WeGovNow Test Community', 'FirstLife map & plan', 'Improve My City report local issues', 'LiquidFeedback debate & decide', 'Community Maps collect & share', and 'Login or register'. A left sidebar shows a user profile for 'John Doe' with a 'Reputation score 70%' and navigation options: 'Post an Offer/Demand', 'View Offers/Demands', 'Personalised timeline', 'Notifications', 'Sign out', 'Settings', and 'Help'.

The main content area is titled 'Edit offer' and shows a 'Demand' type. The title is 'Support the West Norwood Feast!'. The start date is '06-08-2017 9:00' and the end date is '06-08-2017 14:00'. The description states: 'Lambeth has been running regularly to the Portico Gallery in West Norwood for the last 3 years. The work we do there supports the monthly Feast - a community market that sees West Norwood come alive on the first Sunday of the month. Feast needs some helpers to set up and take down various bits of the market and to add the little extra touches that help make the day so special. Come and soak up the good vibes and lend a hand along the way.'

On the right, there is a map showing the location of the market. Below the map, there are fields for 'Area Info' (Name, Area Id, OSM Id, Type) and 'InputMap Info' (Latitude, Longitude, Zoom, Src, TileId, Title). At the bottom, there are two images of the market and a large dashed box with the text 'Drop images in this area' and a small icon of a folder. At the very bottom, there are 'CANCEL' and 'SAVE' buttons.

Exhibit 43 : Personalised Timeline: Trusted Marketplace integrates with OnToMap



5.8.1. Next steps towards pilot platforms implementation

The next major step concerning Trusted Marketplace towards pilot platform implementation (due to Feb 2018) is:

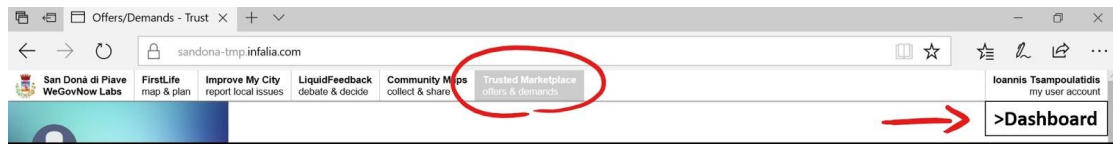
*To separate “Offers & Demands” from the rest features of TMP and actually create **a new component** based on i) the Enhanced User Profile and ii) the Matchmaking mechanism under the name “WeGovNow Dashboard” or simply “Dashboard”.*

The separation will make clearer the purpose of the Trusted Marketplace and also will allow users to access their profile under a common WeGovNow dashboard for all core components.

Trusted Marketplace and Dashboard should be treated as separate components:

- **Trusted Marketplace** will contain
 - Offers & Demands
- **Dashboard** will contain:
 - The enhanced user profile
 - The matchmaking mechanism

Exhibit 44: Updates on Navigation Bar to support the Dashboard for the pilot platforms.



The Trusted Marketplace will appear in the Navigation Bar, while the Dashboard should be accessible as dropdown item under the “my user account” button on the right corner. The proposed position in the menu makes sense since the Dashboard will be accessible only to registered users.

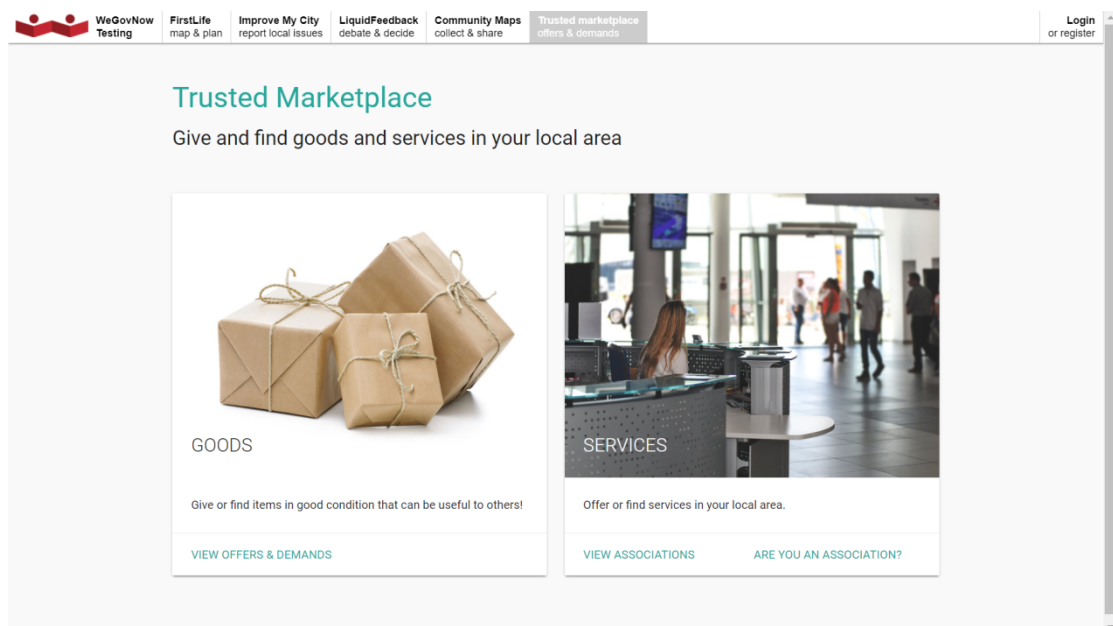
Trusted Marketplace new layout and functionality

Based on London stakeholders meeting (M21) outcomes, it has been decided to let “Offers & Demands” to handle only goods (free items). Offering and demanding services (such as volunteering work) should be handled by external authorised associations since it is not possible to be controlled by the local authorities.

The home page of Marketplace should always contain clear information of its purpose. There will be two big buttons (or page divided in two equal parts). The first button is named “Goods” and the second button is named “Services”. A disclaimer, terms of use and further legal details could be defined per pilot platform.

The following Exhibit 45, depicts the development instance of the new layout of Trusted Marketplace as it is currently envisaged.

Exhibit 45: The development instance of the new layout of TMP (to be finalised by the time of piloting phase based on stakeholders meeting outcome).



Marketplace for Goods (free items)

This is very similar to the existing (Prototype 3) version of offers & demands, having the same interface but without the user profile (which will become WeGovNow Dashboard).

Users do not need to be logged to see the existing offers/demands. Only registered users are allowed to post new offers/demands (and edit or delete their own posts). An offer/demand is removed by the system automatically when date is due (notification is sent to the owner) or when the owner deletes it.

All offers/demands are published only after moderation. Moderation will be a minimal backend interface, listing all offers/demands next to a radio button (publish/unpublish) per record and be accessible only by the local authority moderator(s).

Marketplace for Services

“Services” button will lead to a page quite like ImproveMyCity’s interface (e.g. items list on the left, map on the right).

It will contain the list of approved (after moderation) associations, organisations, timebanks, small charities, and others, that support volunteering work, etc.

The map, will display markers denoting their address (if any).

Each record contains:

- Title
- Register number
- Description
- Photo(s)
- Contact details
- Link to external site
- Tags / Labels
- Attached Documents

Association, Organisations, Charities, etc, will register themselves. The local authorities will be responsible to approve their registration in order for the new post to become publicly available. To verify a new entry, the local authorities will have the possibility to check the registration number or alternatively to check the attached documents (e.g. in the case of small charities that might not have registration number). Moreover, for convenience, an automated notification will inform the moderators that a new registration is posted.

Concluding, the next actions to be implemented for the pilot platforms, in order to support the above plans, are the following:

1. Separate enhanced user profile & matchmaking (dashboard) from offers & demands (trusted marketplace)

2. Split offers & demands in i) “Goods” and ii) “Services”
3. Implement the mechanism to allow associations and organisations to post and list their services in Marketplace.
4. Implement the moderation mechanism for both “goods” and “services” (the backend administration interface)
5. Redesign the UI to match all the above

6. Description of platform services

The core features of WeGovNow are meant to support the integration between components. A set of core features is provided by an extension of stand-alone components, while other core features required the definition of new modules. In this regard, as indicated in Exhibit 9 and Exhibit 10, the first prototype of WeGovNow does not include some new core features which are currently being developed.

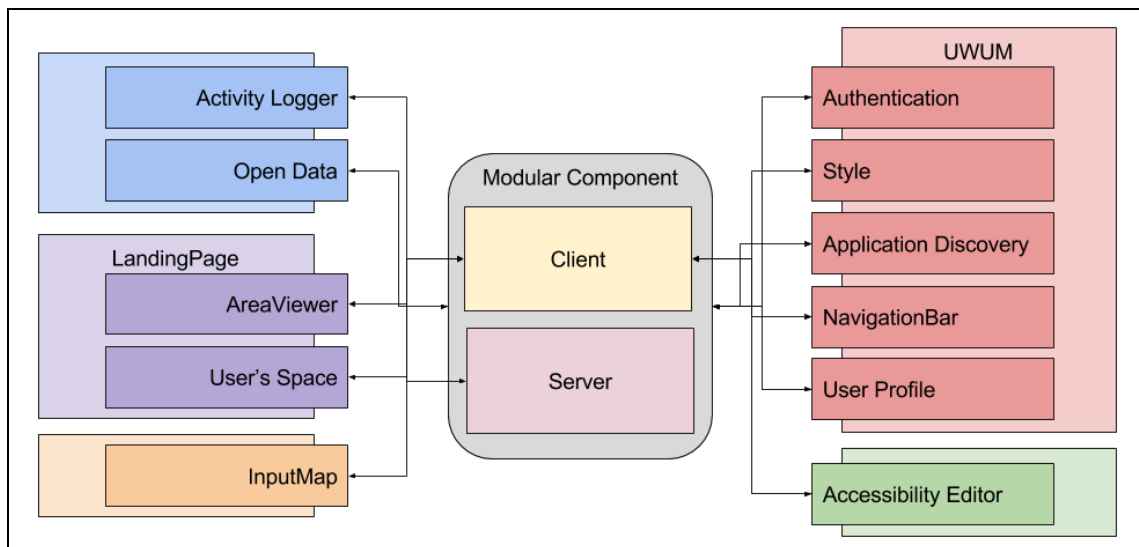
The WeGovNow core thus provides features essential for realising the seamless integration of individual WeGovNow components:

- **Unified Authentication System:** provides functionalities concerning user registration, authentication and authorization, as well as single sign-on.
- **Application Discovery Service:** is an API service providing the list and details of currently available components in a WeGovNow instance.
- **Style Service:** is an API service providing WeGovNow style sheets dynamically to the components, it is used to retrieve the instance customization such as colours, fonts, etc.
- **NavigationBar:** is an API service providing the description or the HTML source of WeGovNow navigation bar, including the button tabs to the current available components and the reference to the user profile.
- **User data storage:** LiquidFeedback provides a data store for global user settings and public user profile information which is accessible to all registered WeGovNow components for storage and retrieval.
- **Centralised Activity Logger:** provides centralised data logging within the WeGovNow platform and data integration aimed at integrating the knowledge about users and about geographical data shared in the platform. Data integration is performed by translating information items, expressed in the terminology used by the specific front end applications of the wegovnow platform, to a unified format, defined by the OnToMap Ontology. In this way, data can be reused cross-application.
- **Linked Open Data and Crowdsourced Data endpoint:** is a semantic endpoint to retrieve the Linked Open data generated from the Open Data of the municipalities and the user activities within a WeGovNow instance.
- **InputMap:** is an embeddable web map to collect spatial input (point based references) and references to existing entities in OnToMap.
- **AreaView:** is an embeddable web map to visualise summary information extracted from OnToMap (see section 5.1).
- **TileServer:** a vector tile server providing geographical entities from official open data and OpenStreetMap, in protobuf format (PBF).
- **Enhanced User Profile Management:** is a module of Trusted Marketplace, extending user preferences with skills, education, social network accounts and the management of user's personal data

- **Geo-Spatial Data Repository (GSDR) of WeGovNow for Quality Assessment:** is a web service for quality assessment/improvement of data provided and generated by the WeGovNow platform.

As mentioned earlier, the core features are provided by existing and new components specifically developed for WeGovNow.

Exhibit 46 : WeGovNow core features provide the environment to integrate modular components accordingly to the specific needs of each WeGovNow instance.



Core features are used by modular components (see Exhibit 46) to:

- Establish connections with other WeGovNow components
- Provide cross-component features
- Unify the appearance to the rest of a WeGovNow instance
- Synchronise components

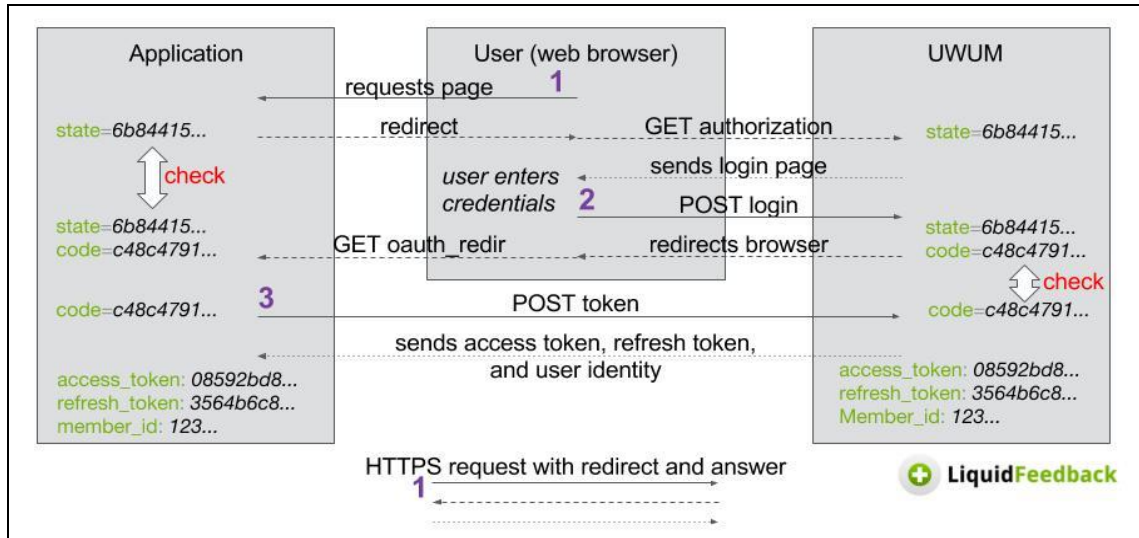
As any other modular component, core features themselves are optional as far as their non-essential functionalities are concerned: if a core feature is not fully enabled in a specific instance of WeGovNow, only its critical functionalities will be available. In the following subsections, the WeGovNow core is described in more detail.

6.1. Authentication Service

For reasons of interoperability and security, WeGovNow aims to create an implementation that is fully compliant with the OAuth 2.0 Authorization Framework as described in RFC 6749, but extended in such way that it allows for secure user authentication following the OAuth 2.0 Authorization Code flow (see Exhibit 47). Special security considerations were taken into account, for instance client identity verification (through X.509 certificates) to repel authorisation code substitution

attacks. For further security considerations refer to Consolidated System Architecture D3.1, Section 2 and Section 5.

Exhibit 47 : OAuth 2.0 Authorization Code flow.



RFC 6749 defines several roles (“authorisation server”, “client”, and “resource server”). The UWUM component as implemented by LiquidFeedback takes the role of the “authorisation server”. Other WeGovNow components will take the role of “clients” but may also act as “resource server” for other components. This allows other components to interact with each other, while UWUM is responsible for user authentication and authorisation.

To register a new client, it is required to address the UWUM integration checklist (see annex 2, Integration checklist). UWUM provides two methods of client registration:

- registering clients through the municipality (or their technical administration) or an organisation running an installation of WeGovNow,
- registration of any other (“dynamic”) client on a per-user basis by each user who wishes to use that client to access WeGovNow (machine accessibility).

Manual client registration by the municipality is only suitable for those clients that are known at the time of deployment of a WeGovNow instance.

Currently, it is possible to use UWUM with an invite code, in the future it shall be possible to create non-verified accounts using an email registration, Google, Facebook or OAuth 2.0 compliant authentication server. Non-verified accounts must not have any voting right or be counted in any other quantification until they are verified.

6.2. Application Discovery Service

The Application Discovery Service provides information about the available software modules in a WeGovNow instance, as defined in the configuration of the platform. The Application Discovery Service role is to enable the dynamic configuration of active WeGovNow software components: to enable cross-components features accordingly to the availability of other services, to provide navigation active components can “discover” other enabled components.

New components can be included in a WGN instance and in WGN environment in general following the Integration Checklist (see Annex 2).

The Application Discovery Service is available via a REST API endpoint for authenticated clients (Exhibit 48). For technical details and future development of the Application Discovery Service see Annex 2 to D3.1 WeGovNow Consolidated Architecture.

An application discovery endpoint has been added for prototype 2.

Exhibit 48 : Application discovery.

JSON	Raw Data	Headers
Save	Copy	Filter JSON
<pre> result: 0: base_url: "https://wegovnow-pt3.liquidfeedback.com/" cert_common_name: "wegovnow-pt3.liquidfeedback.com" manifest_url: "https://wegovnow-pt3.liquidfeedback.com/api/1/info" name: "LiquidFeedback" type: "system" 1: base_url: "https://pt3-imc.infalia.com/" cert_common_name: "pt3-imc.infalia.com" name: "Improve My City" type: "system" 2: base_url: "https://pt3.wegovnow.eu/" cert_common_name: "pt3.wegovnow.eu" name: "Landing Page" type: "system" 3: base_url: "https://pt3-tmp.infalia.com/" cert_common_name: "pt3-tmp.infalia.com" name: "Trusted Marketplace" type: "system" 4: base_url: "https://pt3.wegovnow.firstlife.org/" cert_common_name: "pt3.firstlife.org" name: "FirstLife" type: "system" 5: cert_common_name: "wegovnow-gk-pt3.geokey.org.uk" name: "Community Maps" type: "system" </pre>		

Considering the extent of the existing WeGovNow applications, using application discovery might create more effort for every component owner than possible benefit. However, the existing UWUM endpoint for application discovery returns a list of applications along with their registered base URLs and provides a method for application discovery in WeGovNow systems with a growing number of applications.

Dynamically registered applications (dynamic clients for OAuth 2.0) are fully supported (registration with X.509 TLS client certificate authentication or DNS TXT records, see Section 2.4.2 of the UWUM Work Report from December 12, 2016). These dynamically registered applications are included in the list of registered applications returned by the application discovery endpoint for the currently logged-in user. Additional information on provided protocols and services may either be retrieved through a yet-to-be-defined well-known URL constructed from each application's base URL, included in future versions of the UWUM database, and/or collected by other applications such as OnToMap.

6.3. Style Service

An instance of WeGovNow includes style customization, for instance colours of a municipality, fonts, icons, etc. The style service was introduced as part of UWUM (Unified WeGovNow User Management) specifically to provide the style customization of the current instance of WeGovNow.

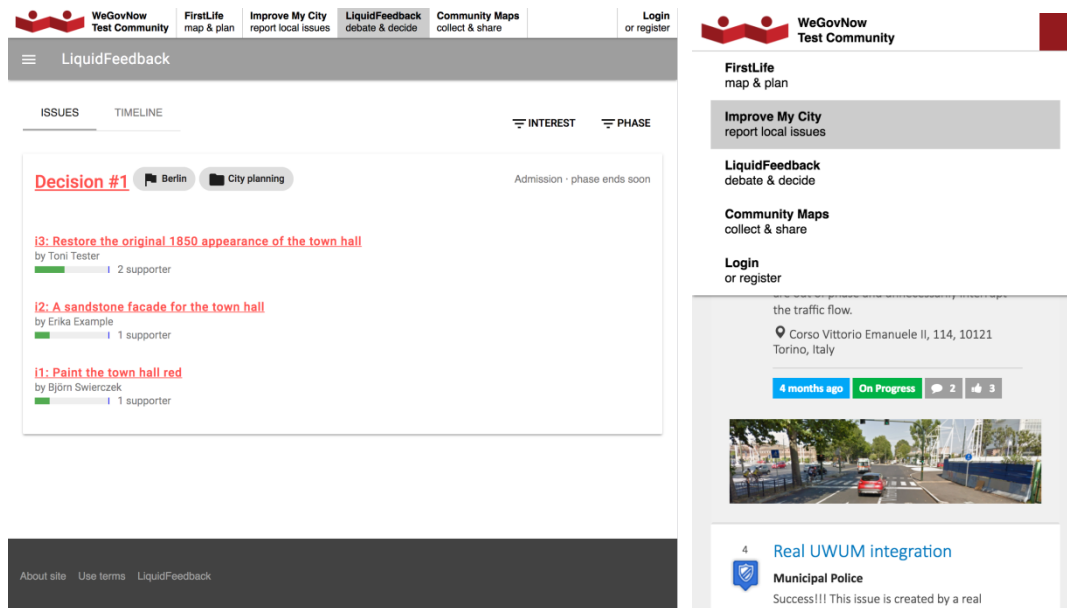
The Style Service already provides a material design colour theme. As of now only LiquidFeedback honours the colour theme. It planned that all other components also honour the colour theme. Future versions may also include font and other style information.

For technical details and further development of the Style Service see Annex 2 to D3.1 WeGovNow Consolidated System Architecture.

6.4. NavigationBar Service

To integrate all WeGovNow applications in such way that they look and feel like a single application, all WeGovNow applications share a common WeGovNow navigation bar (see Exhibit 49).

Exhibit 49 : NavigationBar is dynamically retrieved from UWUM with the current setup of the platform instance. There is also a responsive version depicted on the right



The navigation endpoint of the UWUM server returns this navigation bar to be included by each WeGovNow application. This way, modifications to the navigation bar can be made at a central place without the need to change every single application. Either a login button or the user name with a link to a user page (where logout is possible) is included in the navigation bar, depending on whether an access token is provided when calling the endpoint.

6.5. User data storage

A key factor to provide a good overall user experience is to keep consistent user information across WeGovNow components: to use the latest information regardless to where user provided in the profile settings or in a component setting.

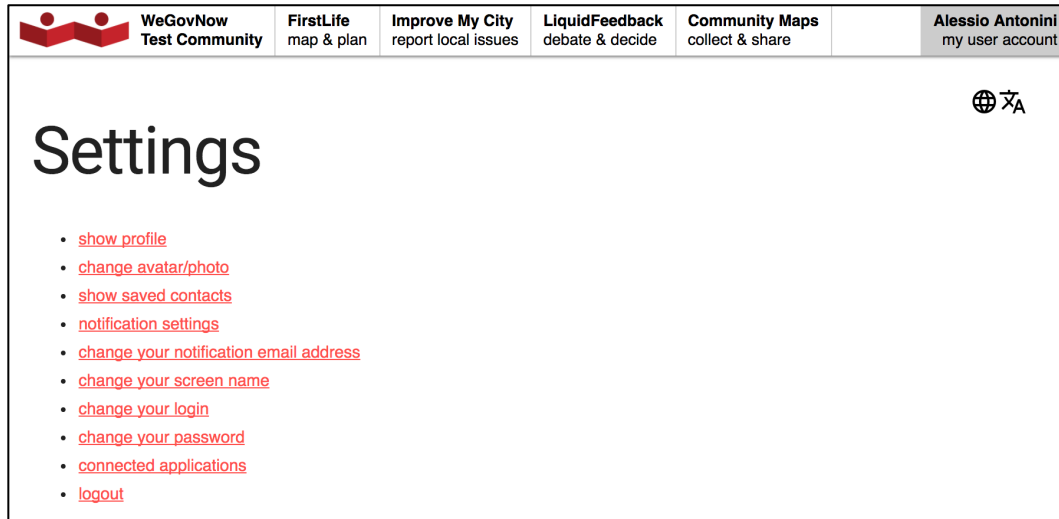
In order to enable the sharing of user information, UWUM provides a **user data storage**, which is the “single point of truth” about user’s information. For instance, user data storage can be used to keep and update the email address for notifications, the preferred language, the accessibility preferences.

6.6. Centralised User Profile

The WeGovNow platform will provide a centralised user profile repository to its components where user profile information of common use among the platform are collected. The centralised user profile will be an extension of user profile settings of UWUM component. Currently, the user settings in UWUM enable users to manage their personal data, authentication details and notification settings. The user settings

are accessible only through LiquidFeedback user settings (see Exhibit 50), TrustedMarketplace component will provide an extended interface to the user settings (see section 6.11).

Exhibit 50 : UWUM user settings.



The centralised user profile has the following main functions:

- Single synchronisation point for common user information (such as firstname, lastname, displayname, email, etc.) avoiding the multiple request of user profile data;
- Propagating the updates of users' profile across WeGovNow platform (Exhibit 46) avoiding inconsistent information;
- Increasing the integration of WeGovNow components providing a mechanism to share user's information platform wide.

The user profile data fields and their type will be configurable per installation. These data fields can be of standard types such as text, number, image, location or complex JSON data fields. Standard type fields will be automatically editable by UWUM's built in editor but can also be updated using the API. Complex JSON data fields need to be updated using the API by the corresponding component(s) using such fields.

The management of the user profile will be extended in TrustedMarketplace including the reference to the information about the user stored in UWUM, the reference to user settings in each application, the accessibility settings (see next section), and global setups about the management of user information in WeGovNow.

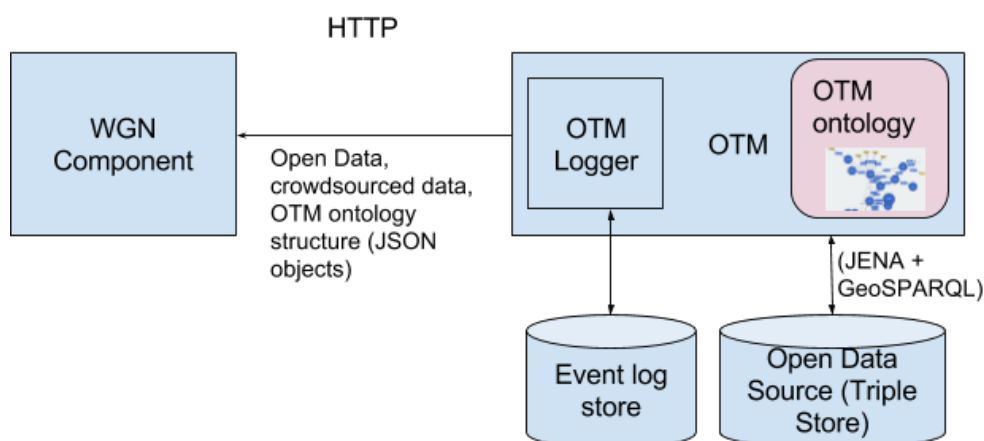
6.7. Crowdsourced and Linked Open Data endpoint

The OnToMap endpoint supports data integration within the WeGovNow platform. The idea is that of having a single point of access to the geographical information available in an instance of the platform, in a unified format and representation language that enables the applications to retrieve and present shared data regardless of their origin. Specifically:

1. OnToMap supports the integration of heterogeneous Open Data sources, which are stored into the platform in a Linked Data format supporting the semantic exploration of information (see below). We have acquired the Open Data from Torino (enriching them with Open Data provided by the Geoportale of Piedmont Region), San Dona' Di Piave and London Southwark, concerning services available in the towns, transportation systems, facilities, and the like. In the next future, further Open Data from the three cities will be added; e.g., aggregated statistical data concerning the economic tissue in San Donà di Piave.
2. OnToMap supports the integration of data shared within the platform by the WeGovNow applications, which can publish the users' activities concerning the creation, revision, etc., of data items through the OnToMap Logger. As this type of information is based on the observation of events concerning user activities, it is stored in a log as a set of events mapped to the common terminology provided by OnToMap. This aspect is described in the section about the OnToMap Logger.

The integration of heterogeneous data and their representation in a common format is based on a semantic knowledge representation layer that reconciles data representational and conceptual heterogeneity.

Exhibit 51 : OnToMap architecture.



The OnToMap Ontology (see Exhibit 51) is a knowledge representation layer that makes it possible to:

- Integrate heterogeneous Open Data and manage it as Linked Data. As reported in <http://linkeddata.org/>, “Linked Data” is about using the Web to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods." Heterogeneous data integration has been traditionally carried out by exploiting domain ontologies representing the types of information to be described at the conceptual level¹.
- Provide a dictionary, shared in the platform, for mapping the concepts used in the domain conceptualizations adopted by the WeGovNow applications to a unified terminology for cross-application data sharing.
- Describe semantic relations among information items to express spatial relations, different levels of abstraction in the description of entities and thematic relations. This approach enables a semantic and geospatial exploration of the information space.

The OnToMap Ontology is defined using the OWL Web Ontology Language <https://www.w3.org/OWL/>. We have developed the ontology by importing standard ontologies for the representation of geographical data (e.g., the definition of SpatialObject with geometry and feature) and by taking into account the categories of information available in the Open Data which were provided to us by the Municipalities participating in the WeGovNow project. The introduction of high-level concepts and of subclasses enabled us to map Open Data to ontology concepts at different granularity levels, depending on the degree of specificity provided from the Open Data sources. For instance, the Open Data from Torino city distinguish several types of businesses (bars, restaurants, etc.). Differently, the Open Data from San Dona' di Piave describe some types of businesses in a more general way (e.g., Logistics, Manufacturing, etc.). We thus defined concepts at different levels of specificity and accommodated the Open Data accordingly. Notice also that the semantic representation of relations among ontology concepts enabled us to relate different types of concepts, including initiatives, issues and other information categories handled by some WeGovNow applications, such as LiquidFeedback, which have geographical objects as their objects. While this type of information is not available in the Open Data provided by the Municipalities, it concerns the logging of user activities in the platform instances.

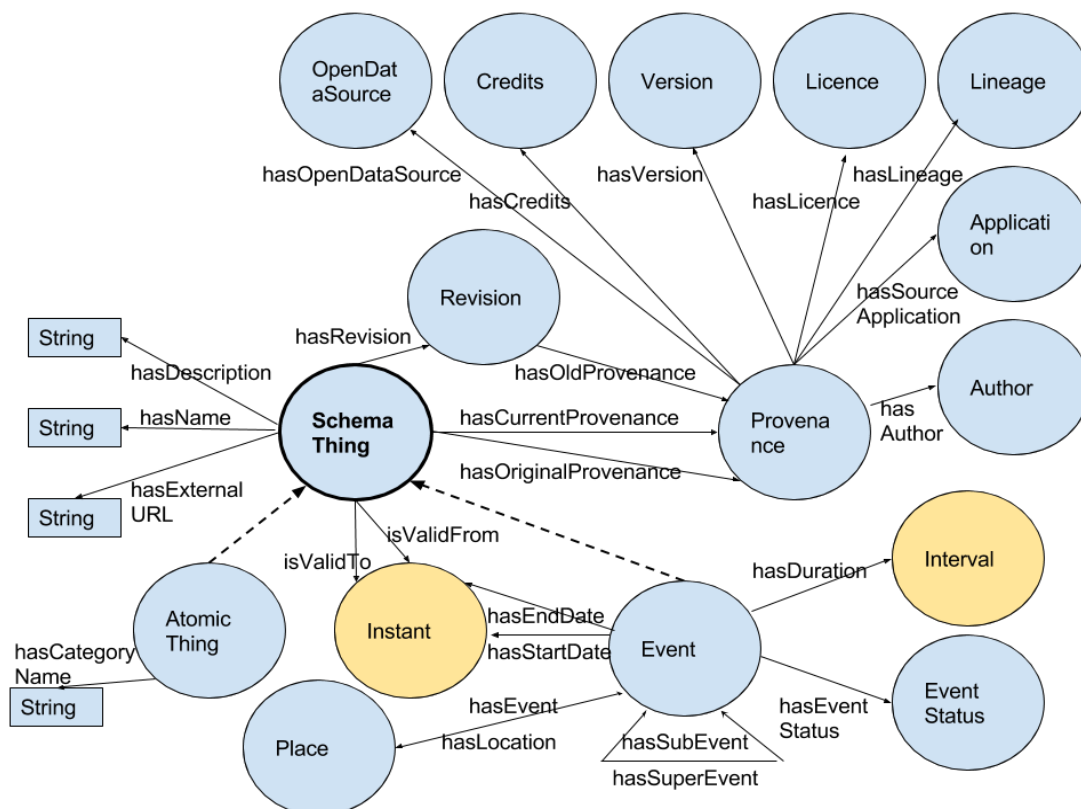
The latest version of the OnToMap Ontology can be downloaded in OWL format at the following URL: <http://OnToMap.eu/ontology/>. The ontology includes about 100 concepts.

¹ For instance, see (i) Fonseca, F. T. “Using ontologies for geographic information integration” Transactions in GIS, 2002, and (ii) K. Janowicz, S. Scheider, T. Pehle, and G. Hart. 2012: Geospatial Semantics and Linked Spatiotemporal Data – Past, Present, and Future. *Semantic Web - On linked spatiotemporal data and geo-ontologies*

The following Exhibit 52 shows a portion of the current version of the OnToMap ontology, focusing on concept “SchemaThing”, the root of the taxonomy concerning geographical information.

“SchemaThing” specifies the main data structure for geographical information items. It is related to concept “Revision”, in order to comply with the possibility of having multiple versions of data, modified at different times by different users, and to concept “Provenance” in order to support the retrieval of the provenance of the current version of data. In turn, each revision of a geographical data item is related to its own provenance, so that the revision history of data can be reconstructed.

Exhibit 52 : Portion of the OnToMap Ontology.



The OnToMap Ontology supports a more or less tightly coupled data integration, in order to allow both the mapping of stable domain conceptualizations, based on a fixed set of data categories (e.g., FirstLife’s one), as well as the integration of dynamic conceptualizations, similar to tag-based systems. “SchemaThing” has the following sub concepts:

- “AtomicThing”, which represents geographical information based on dynamic data categories, such as those used in GeoKey/CommunityMaps. It has a “hasCategoryName” property, which can be used to associate information items to categories, modelled as tags. This approach supports a free type of data tagging,

which does not require consistency checks when adding new categories to a domain conceptualization.

- Concepts for the representation of geographic information based on stable domain conceptualizations, which can be mapped to OnToMap Ontology:
 - “Event”, which represents events, associated to their temporal extension (see the relations with concept “Instant”, which models temporal points), and to the “Places” in which they are held.
 - “Place”, which represents all the geographical items having a geometry representing their geographical extension.

Regarding this work, POLITO has collected, analysed and prepared for acquisition in OnToMap the Open Data provided by the cities of Torino, San Dona’ di Piave, and Southwark. UNITO has carried out the design and development of the Software (OWL ontology, OnToMap component with Logger).

Representation and storage of Open Data

The Open Data integrated in OnToMap is mapped to the ontology concepts and it is represented as Linked Data, as RDF (Resource Description Framework²) triples. This makes it possible to exploit standard tools for browsing the ontology and for retrieving geographical information items.

As shown in Exhibit 51, OnToMap stores Linked Data in the Open Data Store: this is a triple store, i.e., a specialized tool for managing and querying RDF datasets. The current implementation of OnToMap exploits the Parliament triple store³, which supports GeoSPARQL queries⁴.

OnToMap exploits the Apache Jena ontology API⁵ to perform GeoSPARQL queries on ontology concepts. The result of a query is a set of RDF triples representing geographical information items. This set of triples is translated to an external GeoJSON format to make data directly usable by WeGovNow applications for map-based presentation.

The mappings between the data categories defined in the Open Data Sources and the ontology concepts are aimed at translating external data items to sets of RDF triples to be loaded in OnToMap Open Data Store. As Open Data sources typically use standard data representations, open source mapping tools can be found for doing this work without defining ad-hoc software. For instance, for converting shape files to RDF data we use GeoTriples (<https://github.com/LinkedEOData/GeoTriples>)

² <https://www.w3.org/RDF/>

³ <http://parliament.semwebcentral.org/>

⁴ <http://www.opengeospatial.org/standards/geosparql>

⁵ <https://jena.apache.org/documentation/ontology/>

Data retrieval API

The interaction between OnToMap and the other WeGovNow applications, regarding data retrieval and semantic data representation, is managed through the OnToMap Logger API specified at <https://OnToMap.eu/>, which returns results as a JSON object.

The API can be used to navigate the common data structure defined by the OnToMap Ontology and to retrieve the Open Data provided by the OnToMap Open Data Store, as well as the data crowdsourced by WeGovNow applications, in the unified terminology and format provided by the OnToMap Ontology. Specifically:

- The API supports the exploration of the concepts and relations stored in the OnToMap Ontology. It makes it possible to retrieve the root concept of the ontology, the list of concepts, the concepts related to a specific concept C, and so forth. This type of information makes it possible to discover how the ontology is shaped (concepts and relations) directly from the application, via http. It thus represents an entrypoint useful to retrieve metadata about the knowledge that is represented. While the structure of the ontology could obviously be analysed by downloading the owl document describing it, this might be error prone (suppose that the ontology has been changed but this was not clearly reported in the configuration information). In contrast, the interactive entrypoint offered by OnToMap makes it possible to provide detailed and up-to-date information about the ontology structure in use.
- The API can also be used to retrieve information about the instances of a concept C, which can be expressed either in the terminology of the WeGovNow application (in which case OnToMap translates C to the corresponding concept of the OnToMap Ontology), or directly in the OnToMap Ontology format. Depending on the filters used in the query, the retrieved instances can belong to the Open Data and/or to the data crowdsourced by the WeGovNow applications: the information items retrieved by applying the Open Data filter are GeoJSON objects describing geographical information (specific restaurants, etc.); the information items retrieved by applying the crowdsourcing filter are both geographic items and instances of other types of information, such as issues, pushed to the OnToMap Logger by the WeGovNow frontend applications (e.g., LiquidFeedback, ImproveMyCity). The API supports filtering the information items to be retrieved by geographical area and/or temporal validity, and/or data source (e.g., Open Data belonging to a certain source, crowdsourced data). The information items returned by the API are, by default, the most recent versions of the data available to OnToMap, and they are represented as GeoJSON “Features”⁶.

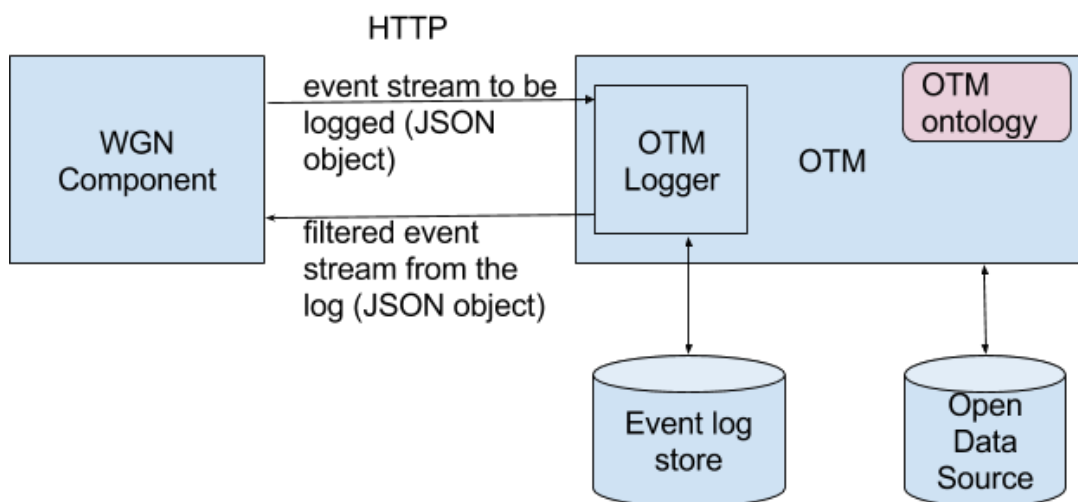
The data retrieval API is fully developed and tested with the WeGovNow applications.

⁶ See <http://geojson.org/geojson-spec.html#feature-objects>.

6.8. User's Activity Logger (OTM logger)

UniTo has developed the OnToMap Logger. The OnToMap Logger is a centralised collector of the history of actions performed by WeGovNow users by interacting with the WeGovNow applications. The goal is that of achieving a unified perspective on user behaviour and of providing a unified view on the data shared in the WeGovNow platform, including the Open Data managed by OnToMap, in order to enable WeGovNow applications to retrieve information collected about geographical objects, initiatives, issues, and so forth. The OnToMap Logger enables WeGovNow applications to push streams of events to be logged, and to retrieve filtered log information; e.g., the activities performed by a certain user in all the WeGovNow applications on a certain date.

Exhibit 53 : Interaction between a WeGovNow (WGN) component and the OnToMap(OTM) Logger.



For user activity collection, the OnToMap Logger offers an API that allows WeGovNow applications to push user activity events, represented as JSON objects; see Exhibit 53. Events can be pushed synchronously (one by one, as soon as the user actions happen), or asynchronously (pushing a list of events to be stored). The action descriptions specify user activities on data items; e.g., creating, updating, removing, commenting some data objects.

The Logger APIs for logging and for data retrieval are fully developed.

Only authorized WeGovNow applications must be able to use the Logger APIs for storing and retrieving data and events. For enforcing this requirement, the OnToMap logger makes use of X.509 client certificates, which are used for authorization and authentication of WeGovNow applications. In order to connect to the OnToMap Logger, every application must forward a Certificate Signing Request to

LiquidFeedback that will result in a client certificate signed by LiquidFeedback that will have to be included in every request made to the OnToMap Logger APIs.

Since the use of client certificates is not feasible in some circumstances (e.g. requests sent directly from a web page), a token-based authentication method has been implemented for obtaining logger events and open data. The access token can be included as a request parameter, and it can be obtained only by authorized WeGovNow applications.

The OnToMap Logger APIs allow WeGovNow applications to retrieve lists of user action events, sorted from the most recent to the least recent. It is possible to use and combine different filters, to generate list of events which correspond to the specified filter(s); e.g., to retrieve all actions of a specific user in a specific timeframe. It is possible to filter events by the user performing the action, application in which the action was performed, user activity type, interval of time in which the action was performed.

Since, as previously described, the OnToMap Logger is also used to provide a unified view on the data shared among the WeGovNow applications by users; it is possible to filter the event data by some properties of the shared data items associated with the events. This results in the Data Retrieval APIs described in the previous section. Moreover, in addition to the previously cited filters, it is possible to filter the event data by geographical area in which the shared data items are located, the concept of which the shared data items are instances and the unique identifier associated with every shared data item.

The OnToMap Logger documentation can be found at the following URL: <https://OnToMap.eu/>. This documentation reports the specification of the APIs, and examples of usage with sample results.

6.9. InputMap (V3)

To enhance integration between WeGovNow components, the consolidated system architecture (deliverable D3.1) introduced a unified location map-based module to input spatial coordinates within WeGovNow components (Exhibit 54), improving the overall look and feel by adopting a common solution platform wide, the overall integration link components' entities to a common geographical data source.

InputMap is a web map module connecting users input on a map (point-based reference) with entities of the geographical data source provided by an ad hoc tile server integrating official open data and OpenStreetMap entities of the project areas (Exhibit 55). InputMap can be used by all components in replacement of common map modules used to collect point-based locations (latitude and longitude), without any specific technical requirements. The second version of InputMap introduces new parameters and features, and a general restyling based on Material Design.

Exhibit 54: InputMap overlaps cartography with the geographical entities available in WeGovNow. User's input is enhanced with references to the selected entity.

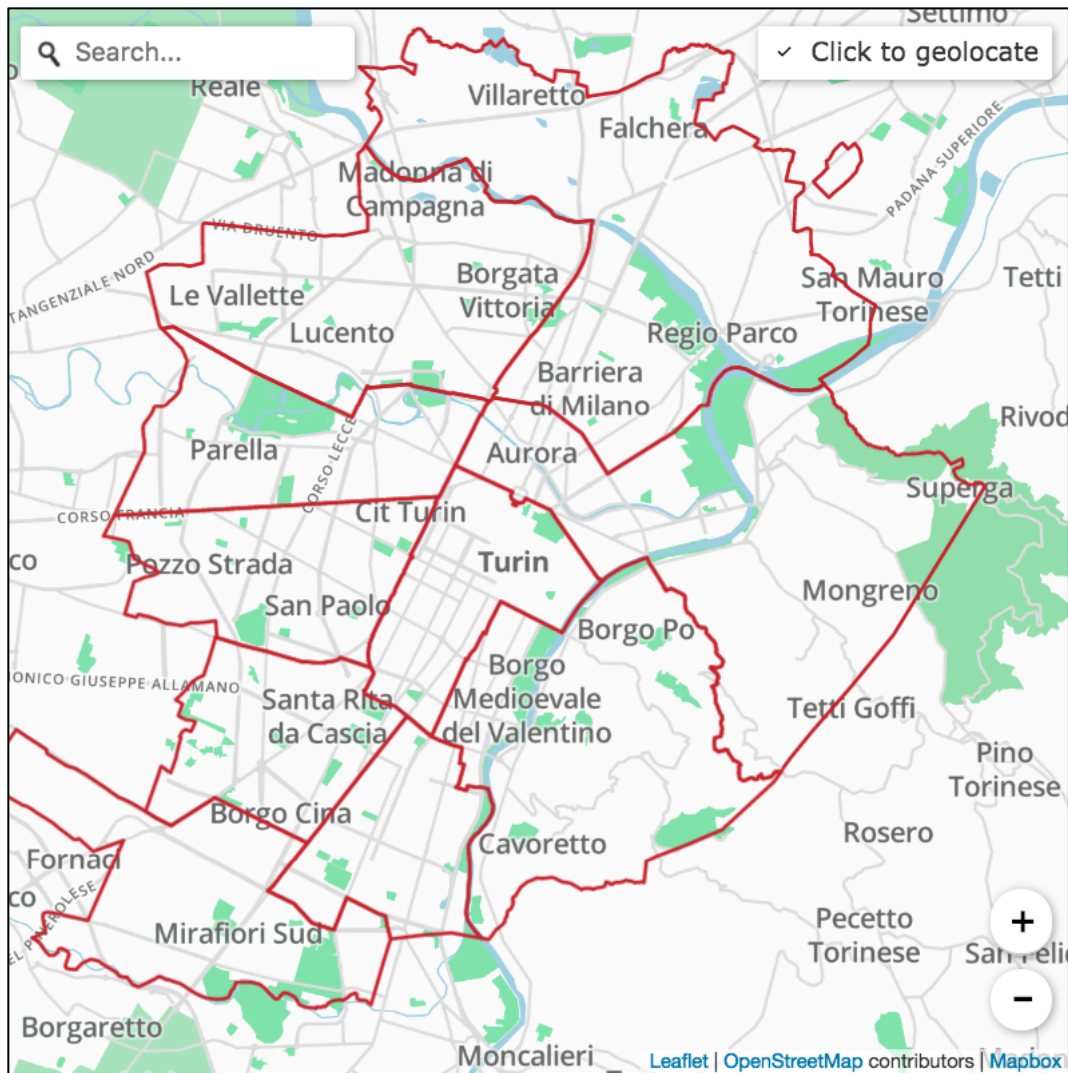
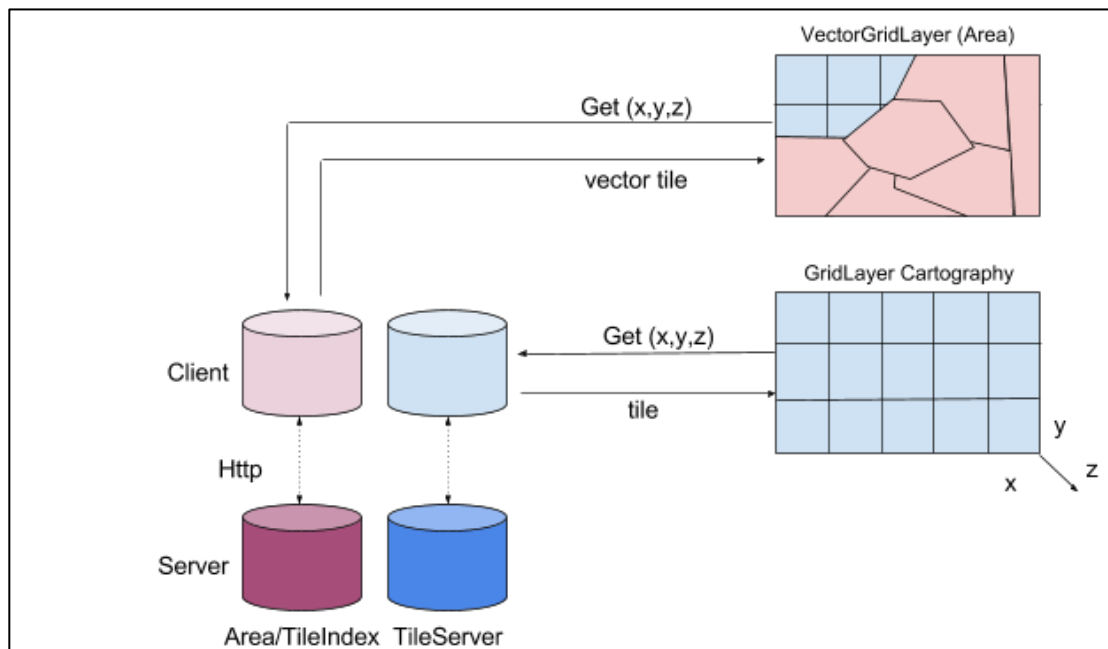
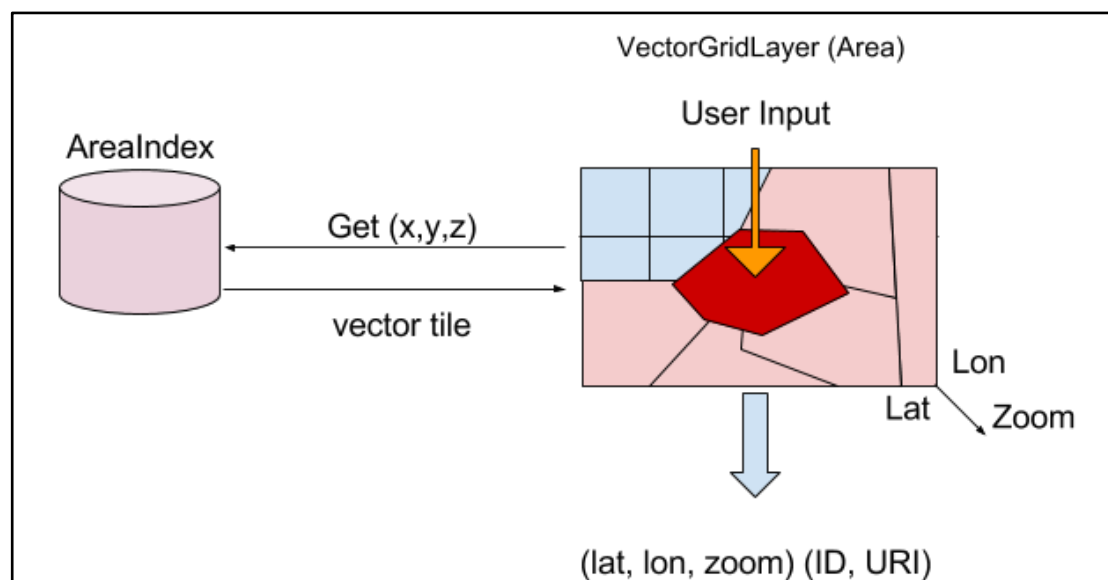


Exhibit 55 : InputMap uses a cartographic layer and a vector layer to the interactive layer. The interactive layer is a GeoJSON data layer provided by a specific tile server integrating municipalities open data and OpenStreetMap entities.



InputMap is based on Leaflet and Leaflet VectorGrid plugin, to render the data provided by a tile server specifically developed to provide the scale-wise the geographical data available in WeGovNow. InputMap collects user's location input (click on the map, see Exhibit 56) and sends information to the "hosting" application, the component embedding InputMap.

Exhibit 56: InputMap concept: user's input is combined with geographical information retrived from WeGovNow tile server (AreaIndex).



To address new requirements coming from the integration of InputMap in LiquidFeedback and ImproveMyCity, the third version of InputMap introduced state management, InputMap can now be used to:

1. input a new location,
2. edit an existing location,
3. show a location

This change enables the use the same module in (creation and update) forms and in presentation views. InputMap mode can be chosen via “*state*” parameter (<https://gitlab.di.unito.it/firstlife/inputmap>).

The output of InputMap are:

- **Latitude** and Longitude
- **Zoom level**, current zoom of the map
- **Areald**, identifier of the selected area (if applicable)
- **TileId**, hash encode of tile
- **OSM Id**, identifier of the geographical entity (if applicable)
- **Name** of the geographical entity
- **Type** of the geographical entity
- **Address** provided by the reverse geocoding service of OpenStreetMap Nominatim⁷
- **Display name** of the address provided by the reverse geocoding

InputMap can be used even when there are not geographical sources available. The extra information about zoom level in combination with latitude and longitude can be used to infer the reference later on, as new geographical sources are available. Tile id is specifically meant to provide a reference to an area, to support the area-based aggregation provided by AreaViewer.

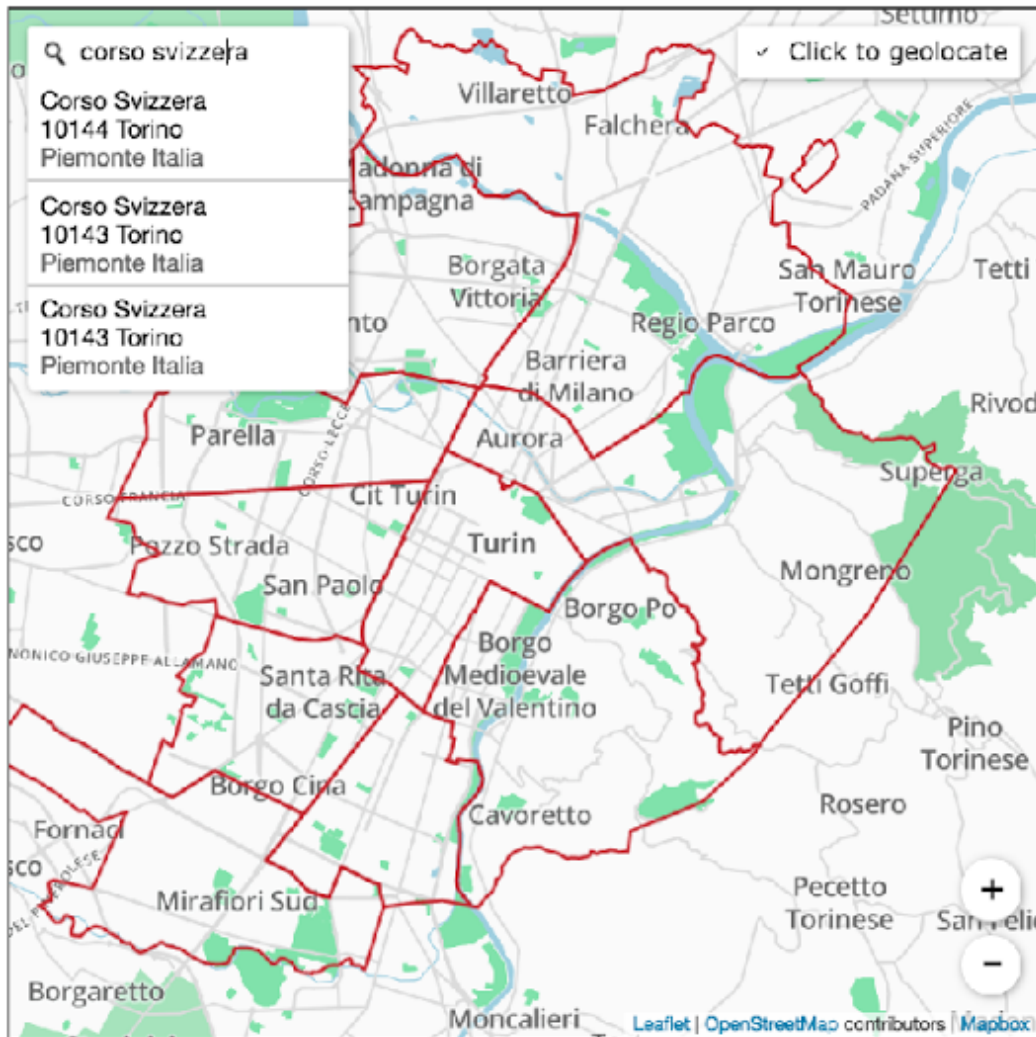
InputMap can be included as embed within any web application or websites as replacement of other web maps. Through InputMap, users can interact with the geographical entities within WeGovNow collected from OpenStreetMap or Open Data, and select a direct reference between them and the hosting application contents.

V2 of InputMap introduces a geocoding service connected to OpenStreetMap Nominatim web service. The geocoding service is used to support address-based geolocation via search bar, and to enrich the point-based (input on map) location with the nearest address (reverse geocoding).

The search bar enables users to search location by address, by click on results users can locate the map (Exhibit 57).

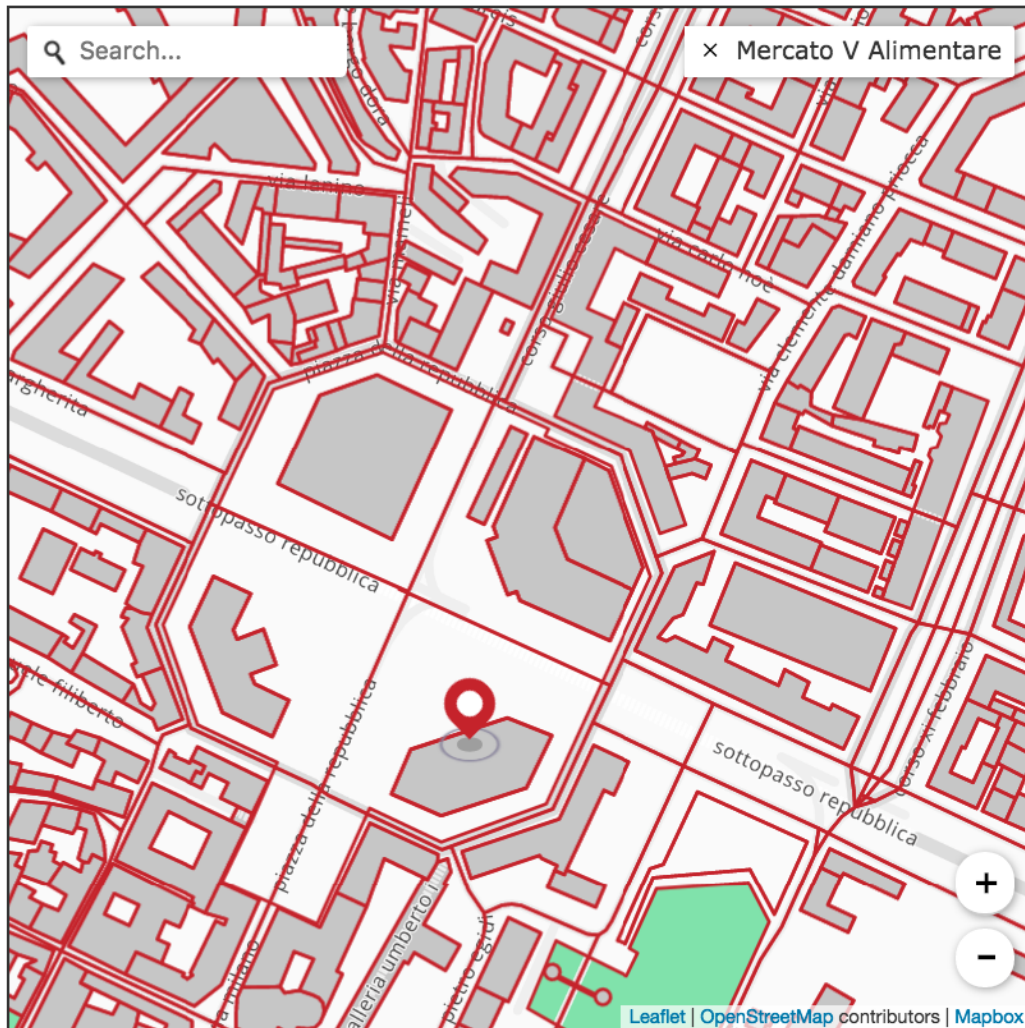
⁷ <https://nominatim.openstreetmap.org/>

Exhibit 57: InputMap V2 introduce a search bar (top left corner) enabling geolocation by address. This service is based on OpenStreetMap Nominatim search API.



Relying on Nominatim APIs, V3 InputMap introduces the use of reverse geocoding of coordinates, to enrich user's input retrieving the most "close" address to their input on map. The result of reverse geocoding is included in the overall result, and could be used in case the TileServer does not contain information regarding specific coordinates. The result is a resilient system which prioritise the information contained in official open data, but in case non-covered areas or scales, it is able to rely on OpenStreetMap crowd data.

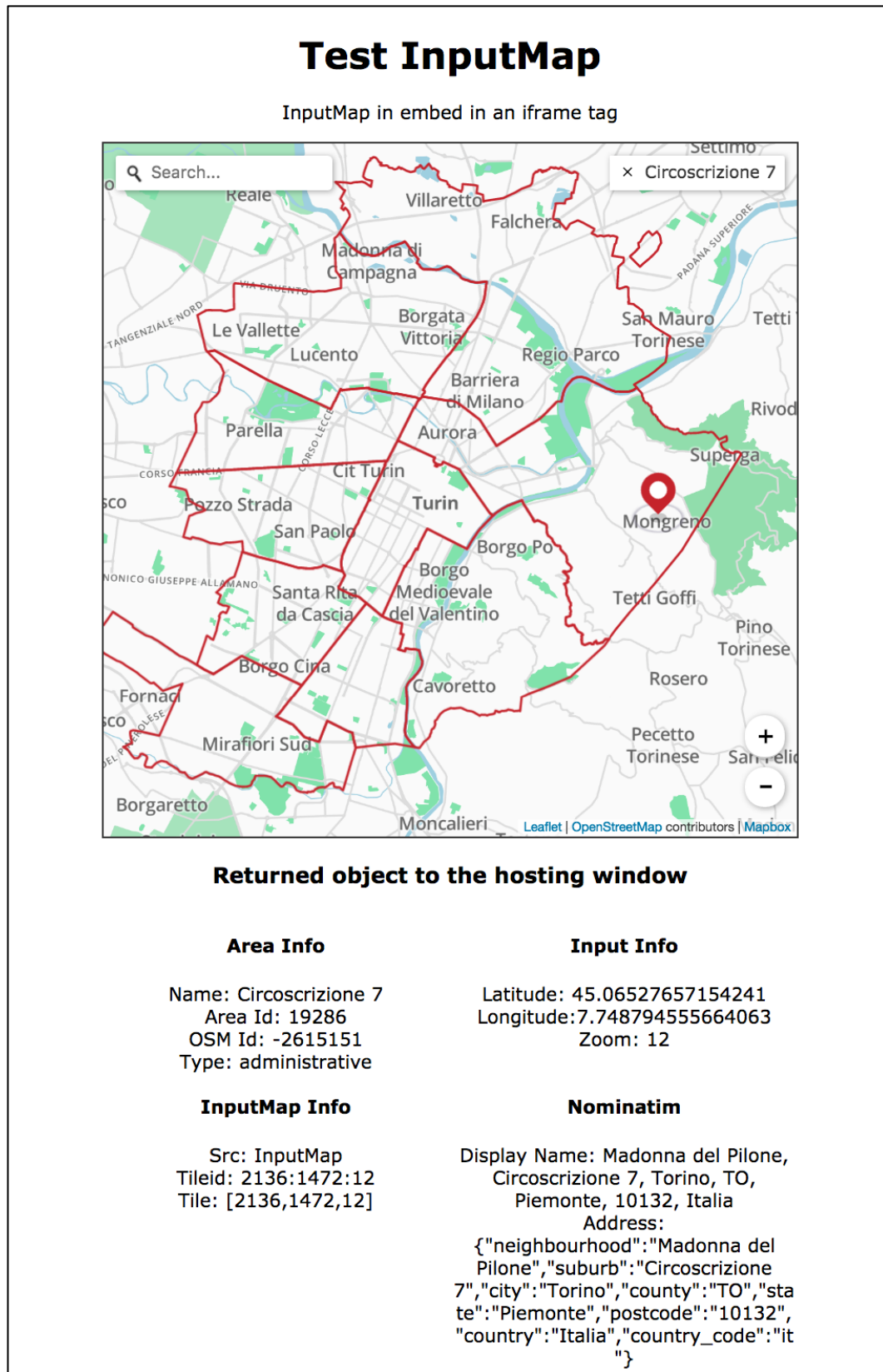
Exhibit 58: InputMap enable users to input different entities according to the selected zoom. E.g. it makes possible to refer explicitly to city district or building.



InputMap V2 introduced a label box introducing a tooltip text about how to use inputmap and the current selected location. The language of the tooltip is dynamically chose considering the agent language (browser), or it could be passed as parameter “lang” invoking InputMap. The label box consents to cancel the current selection by clicking “x” button (Exhibit 58). The cancel action sends a “null” message to the hosting application.

To support integration of InputMap, a test page is available at <https://inputmap.firstlife.org/test>. The test page shows InputMap result messages sent to hosting applications triggered by users’ interactions (Exhibit 59).

Exhibit 59: InputMap test page shows InputMap messages corresponding to user's interactions with InputMap.



6.10. TileServer

InputMap and AreaViewer require to access to a common source of geographical data to render an interactive layer on a web map. In order to provide a fast and standard access to web map components, a tile server has been developed and deployed within WeGovNow platform (TileServer).

TileServer provides geographical entities in vector tile format 2.1 (<https://www.mapbox.com/vector-tiles/specification/>), a open source format widely used to provide vector tiles. TileServer provides other features to support the interconnection between WeGovNow components' entities and the common geographical datasource included in TileServer. Specifically, TileServer provides the following endpoints:

- **/areas/:id** returns the detail of a specific area
- **/area/:z/:lon/:lat** returns the ID of the related area
- **/areas/content/:id** returns a feature collection of contained areas
- **/tile/:z/:x/:y** returns a vector tile in PBF format⁸

Currently, TileServer datasource include the official open data and OpenStreetMap layers of the municipality of Turin.

The end point of TileServer can be found at <https://tiles.firstlife.org/>, source code and documentation can be found at the project repository <https://gitlab.di.unito.it/firstlife/tileserver/>. TileServer is an ExpressJS application, using libraries and tools developed by Mapbox and Leaflet communities.

ProxyLogger

AreaViewer requires to access to OnToMap logs client side by querying for slippy tile notation⁹ in GeoJSON standard¹⁰ encoded in Protocol Buffer (PBF) format¹¹. ProxyLogger is an ExpressJS REST endpoint which converts slippy tile [x,y,z] in boundingbox query, supported by OnToMap logger, and then converts the result in GeoJSON and PBF.

Currently, ProxyLogger exposes one method:

- GET /tile is mapped to GET /events endpoint in OnToMap logger APIs

Therefore, there is an instance of ProxyLogger endpoint for each OnToMap API instance:

- loggerproxy-pt2.firstlife.org > p2.api.OnToMap.eu
- loggerproxy-pt3.firstlife.org > p3.api.OnToMap.eu
- loggerproxy-sandona.firstlife.org > sandona.api.OnToMap.eu

⁸ <https://developers.google.com/protocol-buffers/docs/encoding>

⁹ http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames

¹⁰ <https://tools.ietf.org/html/rfc7946>

¹¹ <https://developers.google.com/protocol-buffers/>

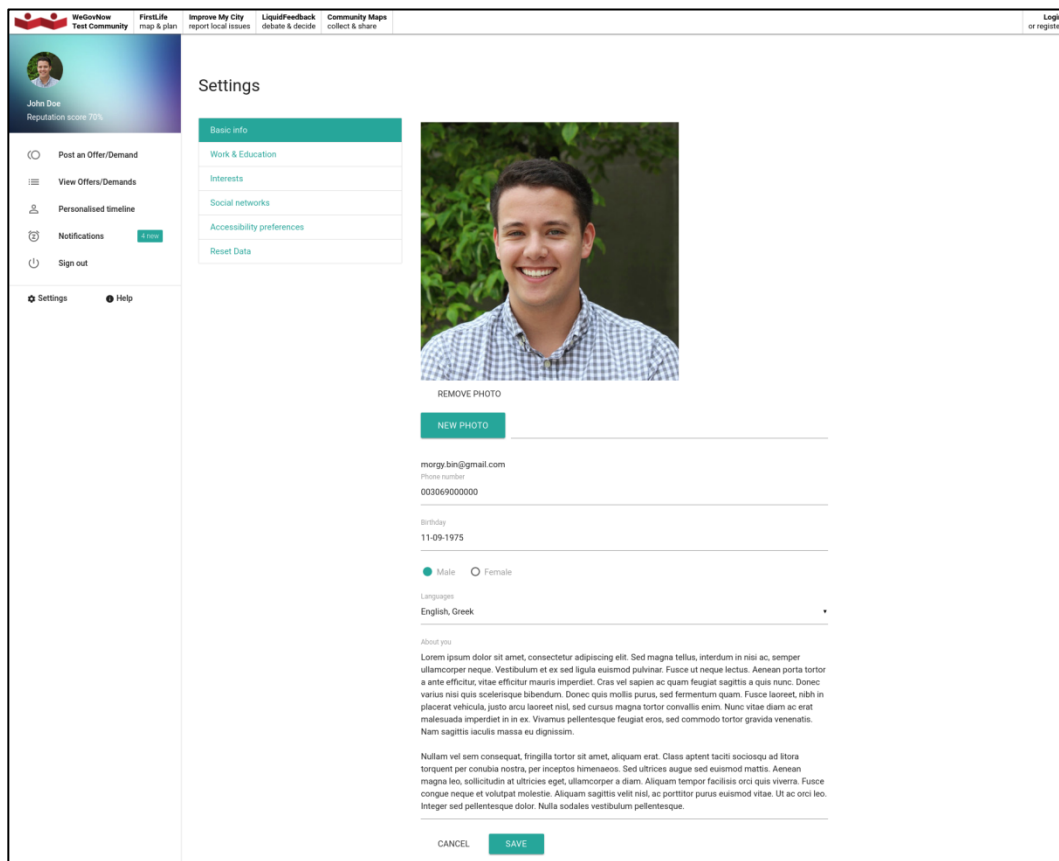
- loggerproxy-torino.firstlife.org > torino.api.OnToMap.eu
- loggerproxy-southwark.firstlife.org > southwark.api.OnToMap.eu

ProxyLogger is not meant to be used in production mode, but as support for client web modules during the development and testing phases. In the late phase of the project, ProxyLogger endpoints will be integrated in OnToMap if needed.

6.11. Enhanced User Profile Management

Enhanced User Profile Management is currently part of the Trusted Marketplace (TM) component and its purpose is to provide a single entry-point to user's preferences and other data (such as user interests) across WeGovNow platform. Enhanced User Profile Management extends the profile settings in UWUM (see Exhibit 60), including the possibility to edit all information stored about the user by all WeGovNow components. Moreover, it connects to social networks on user consent in order to get extra information that could help to better matchmaking suggestions.

Exhibit 60: The Enhanced User Profile Management page, extending the UWUM profile settings.



User Management, besides basic info, it includes other sections such as:

- Work & Education (see Exhibit 61)
- Personal interests (see Exhibit 62)

- Social network accounts (see Exhibit 63)

It should be noted that under the hood, the system provides more than 60000 predefined values which are selected automatically while user types at certain fields. Users are also able to type free text.

Exhibit 61: Section of user's "work and education".

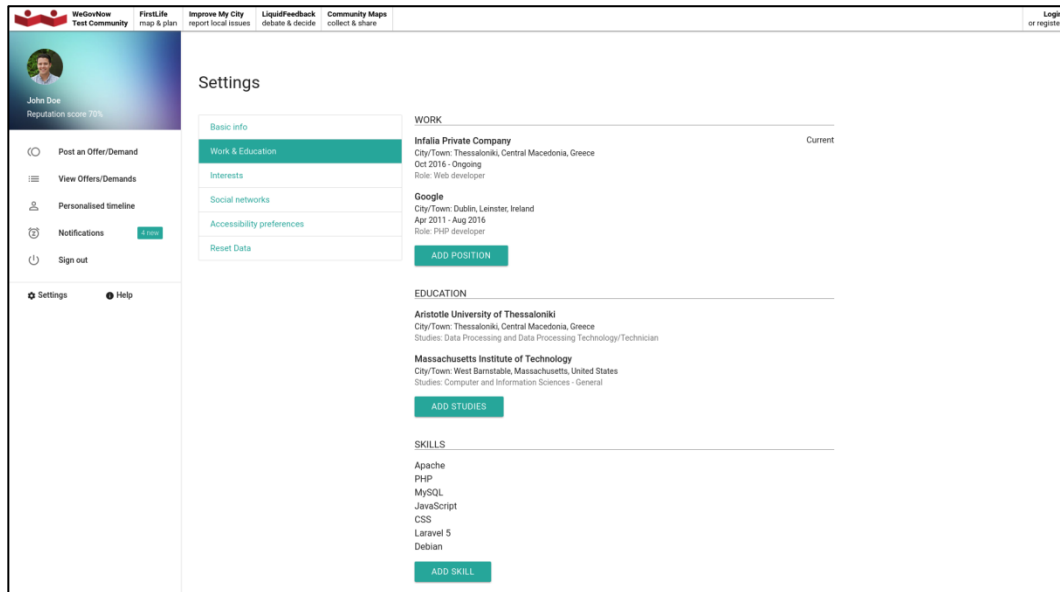
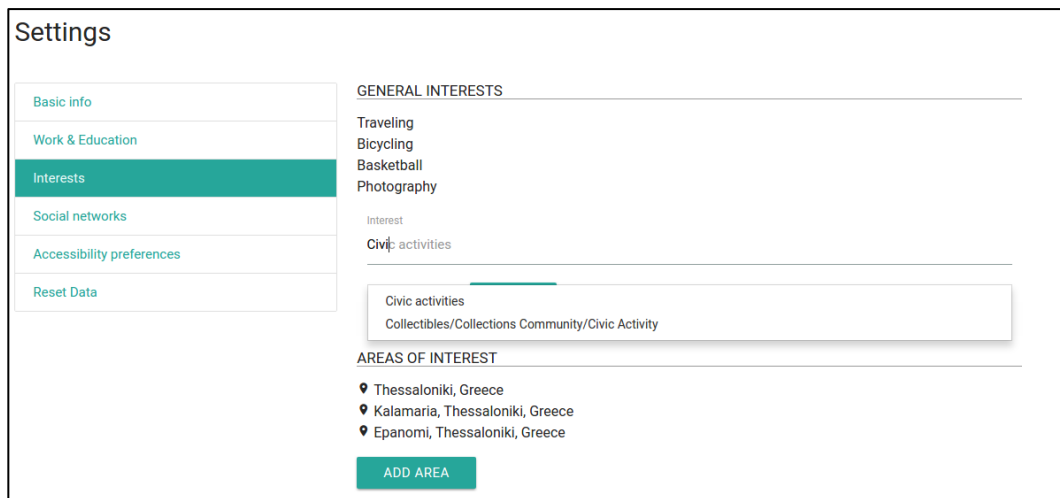


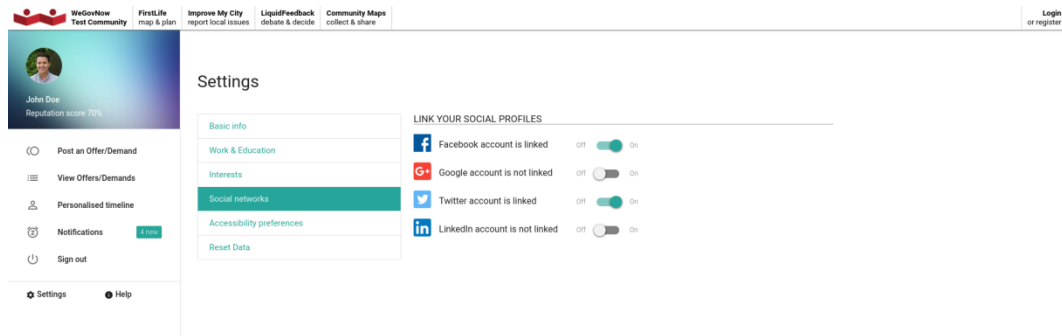
Exhibit 62: Section of user's interests.



Social Networks account

Through the Enhanced User Management users are able to link to their user profile various social networks as depicted in Exhibit 63.

Exhibit 63: Link social accounts.



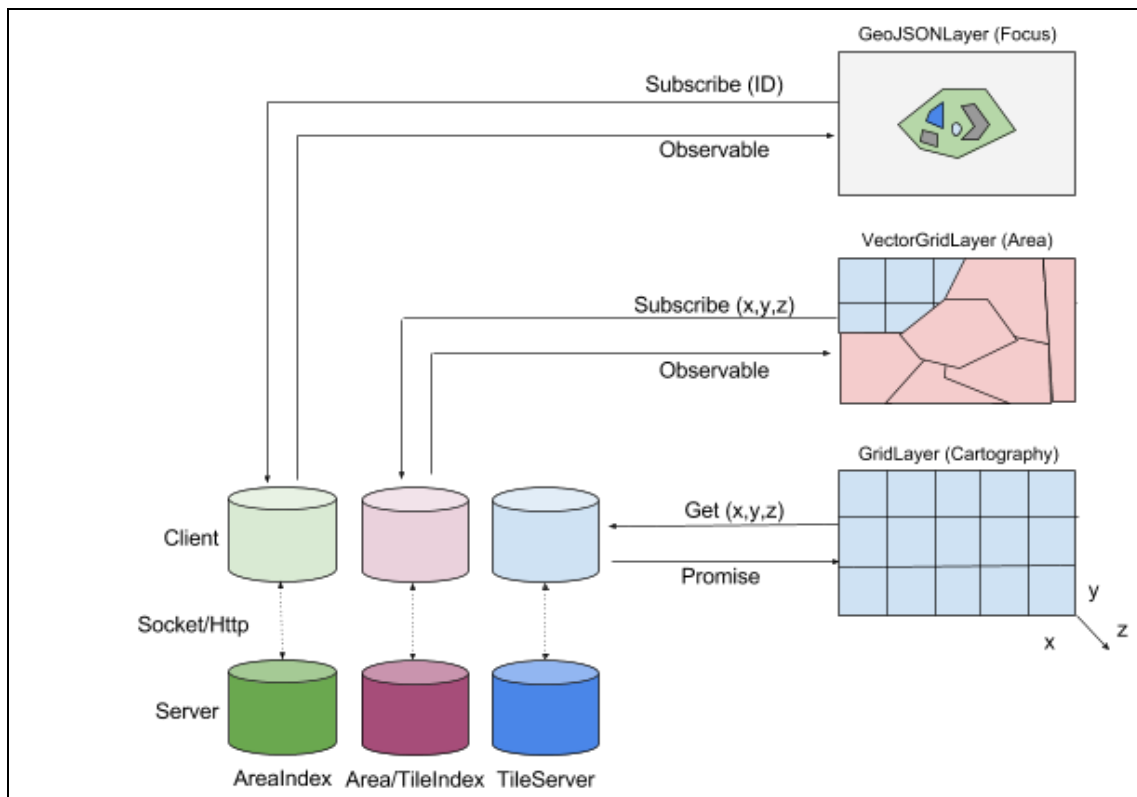
The purpose of linking social accounts is to collect more information about the users automatically and thus enhance the matchmaking mechanism. At any time, users are able to unlink their accounts.

6.12. AreaViewer (V2)

The AreaViewer is a web map based module providing a view of WeGovNow aggregated data based on OnToMap. It works in combination with the InputMap component, exploiting the explicit relations between application data of WeGovNow components and OnToMap entities.

The purposes of the AreaViewer are enhancing the overall look and feel of WeGovNow platform and providing a coherent visualisation of the status of WeGovNow instances across components. AreaViewer presents information from the overall WeGovNow platform extracted from users' activities via the OnToMap logger interface. The information provides will be a summary represented as area-based clusters (aggregations based on geographical entities), see Exhibit 64.

Exhibit 64: AreaViewer is composed by a static cartographic layer, a dynamic geographical layer presenting aggregated data in area-based clusters and a focus layer presenting the details of area-based clusters.

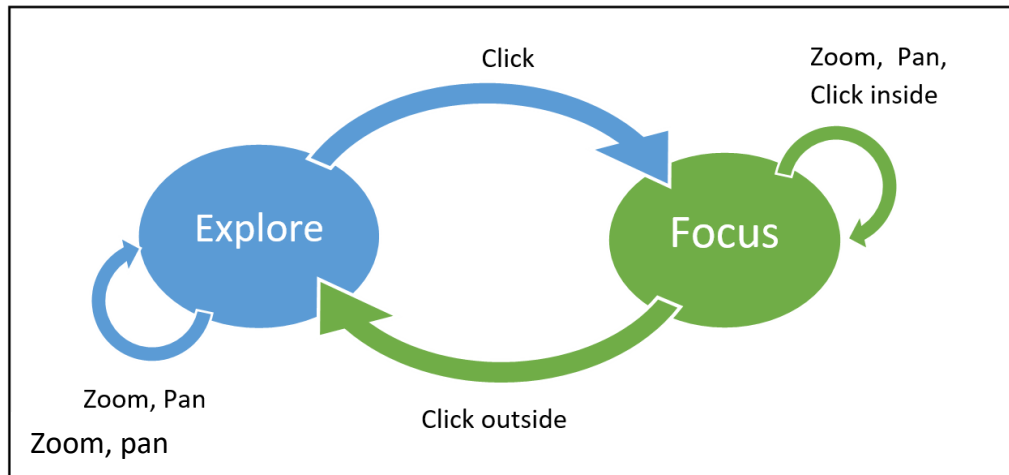


The AreaViewer is a component of the LandingPage that can be included in any WeGovNow component through iFrame (embed) and be controlled via url. Currently, the following parameters are available:

- 1) **C**, hash encoding on the initialization latitude, longitude (map centre) and zoom level (latitude:longitude:zoom) of the map.
- 2) **Date**, ISO format of the day, month, year of the retrieved data (default current day)
- 3) **Contrast**, selection of the low/high contrast style
- 4) **Lang**, interface language

AreaViewer has two main status: “explore” and “focus” (Exhibit 65). In explore status it is possible to pan and zoom the map, changing the view port, and to filter entities according to the source components. Clicking will cause to switch to focus status. The focus will highlight the clicked area, listing only the entities within it, and hiding all point of interest (POIs) not related to the specific area. In focus state, a click inside the focus area will cause to focus on deeply on a new area, a click outside the focus area will cause to go back to explore mode, restoring the previous viewport. A click on one the listed entities present in the focus area, will cause a redirection to the entity details within the belonging component by using the deep-linking bilateral integration method.

Exhibit 65: AreaViewer status automata: explore mode will change to focus on a specific area by click, it goes back to explore mode by click outside.



AreaViewer is based on Leafletjs technologies, a famous open source JavaScript library for web maps. It is developed in JavaScript ECMA6 and released under MIT license. AreaViewer source code and documentation can be found in the project repository (<https://gitlab.di.unito.it/firstlife/areaviewer>).

Interacting with areaViewer

areaViewer sends messages about its current state, and expects messages to trigger state changes. For instance, it is possible to switch the current language sending a message `areaViewer.setLang` with params `{lang:'it'}`.

Triggering a change in current state can be archived triggering a JavaScript event.

Getting state changes (reading the current state)

- `areaViewer.focusOn` entering of focus state
- `areaViewer.explore` entering in explorer state
- `areaViewer.position` at map viewport change

Triggering state changes (changing the current state)

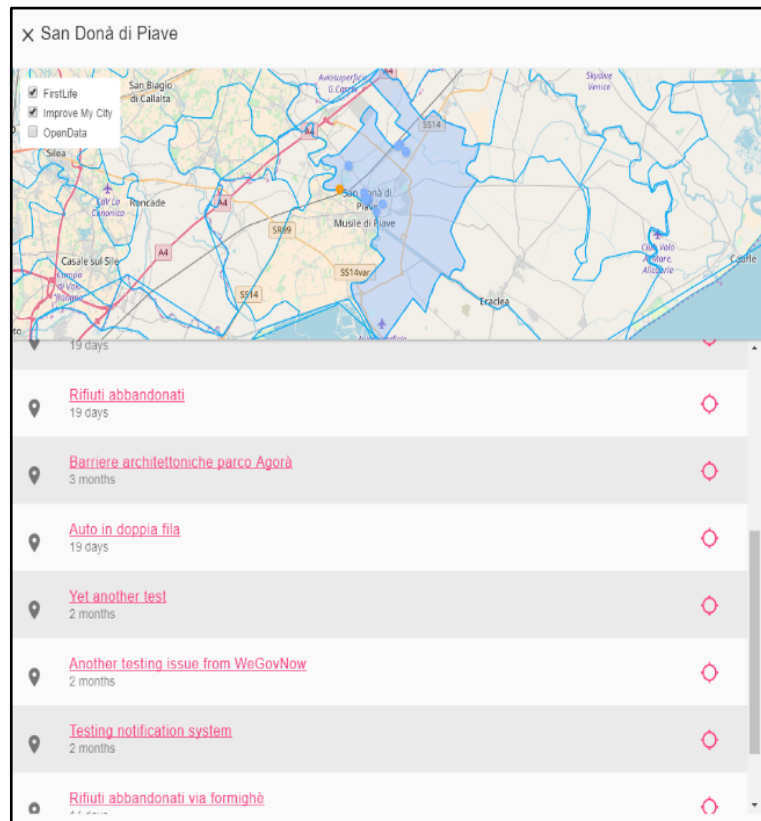
- `areaViewer.setView` to change map viewport
- `areaViewer.setBounds` to change map viewport
- `areaViewer.setContrast` to change map base layer
- `areaViewer.setLanguage` to change language available languages
- `areaViewer.setDate` to change time interval of the events query
- `areaViewer.setPriority` to change the rendering of events
- `areaViewer.focusTo` to enter focus mode on area id
- `areaViewer.toExplore` to exit from focus mode

- `areaViewer.setInteractive` to change focus behaviour enabling/disabling the layout change and content rendering

Exhibit 66: AreaViewer explor state: POIs representing users' activities from OTM logger.



Exhibit 67: Focus mode: user's click triggers a focus on the selected area, POIs within the focus area are left in evidence. POIs outside the focus area (gray circles) are reduced in size and left as background information.



6.13. Geo-Spatial Data Repository (GSDR) of WeGovNow for Quality Assessment (V3)

The third version of Geo-Spatial Data Repository (GSDR) of WeGovNow for Quality Assessment was released. This web service is developed for assessment, improvement and representation quality of data utilized, provided, harvested and generated by WeGovNow.

GSDR consists of the five main components. First component is event triggered tasks. The tasks are designed to automatically derive data from the various sources (i.e., OpenStreetMap, OnToMap). The tasks are triggered either by time (OpenStreetMap is updated once in three months) or as soon as source data are changed. The tasks download required data and provide these data to a second component for further processing. The second component is designed for data quality assessment and improvement. Result of assessment is harvested by a spatial database as well as the data itself. The data and result of the quality assessment can be derived through a public API. Additionally, the API enables WeGovNow components (i.e., FirstLife, LiquidFeedback, ImproveMyCity and Community Maps) to implement quick data check and users' contribution improvement. Fifth component is a front-end displaying

quality assessment results (mainly, through a set of interactive maps) and interactive examples for developers demonstrating ways of the public API use.

The two first components are relied on the igis.tk open source tools collection. The tool `geojson2spline` is designed to convert GeoJSON data to Spatialite format. `tilelog2sqlite` allows us to convert OpenStreetMap tile access log to the GSDR's spatial database. The third tool (`wbi` – Weighted Blocks Indicator) is developed to assess quality of the spatial data.

The public API of GSDR is designed to provide access to data stored in the spatial database and quick quality assessment and improvement. The API converts GET requests to SQL code and response data in GeoJSON or JSON format (see Exhibit 68). GET requests encode SQL code. The semantic of proper requests is described in detail in the documentation available by the following link <https://wgn.gsdr.gq/docex.rvt#api>.

Exhibit 68: Example of a proper request with correspondent SQL code and GeoJSON response

URL	SQL	GeoJSON Response
<pre>api.tcl? SELECT=__printf,__q__ __p__s__c__p__s__c__ __p__s__q__,19,col,row, __&AS=id,wbi,__AsGeoJ SON,geomcent,__&FR OM=wbidata&JOIN=tile s&ON=tile&__e__=tiles. rowid&WHERE=name& __e__=__q__SanDona.O NM.PSI__q__&AND=__W ithin,geomcent,__Build Mbr,12.562845701477 203,45.627366163691 335,12.570345157882 84,45.6313050536553 8,__,__</pre>	<pre>SELECT printf('%s,%s, %s',19,col,row) AS id,wbi,AsGeoJSON(geo mcent) FROM wbidata JOIN tiles ON tile = tiles.rowid WHERE name = 'SanDona.ONM.PSI' AND Within(geomcent,Buil dMbr(12.5628457014 77203,45.6273661636 91335,12.5703451578 8284,45.63130505365 538))</pre>	<pre>{ "type": "FeatureCollection", "crs": { "type": "name", "name": "EPSG:4326" }, "features": [{ "type": "Feature", "id": "19,280440,187292", "geometry": { "type": "Point", "coordinates": [12.5632095336914,45.631085485 9241]}, "properties": { "wbi": 2} }, { "type": "Feature", "id": "19,280440,187293", "geometry": { "type": "Point", "coordinates": [12.5632095336914,45.630605329 41472]}, "properties": { "wbi": 1} } ... </pre>

Currently API supports six types of functions: (1) [SQL core functions](#), (2) [SQL aggregate functions](#), (3) [SQLite extention-functions.c](#), (4) [Spatial functions](#), (5) Tiles [Common Framework functions](#), (6) Custom functions (e.g., levenshtein distance calculation).

A set of interactive examples assisting developers to write correct URLs to the GSDR's API was prepared (see https://wgn.gsdr.gq/docex.rvt#quality_examples). The examples are split into three categories. First, basic examples of simple SQL requests allow developers to understand semantics of the requests and derive correspondent SQL code or raw response. Second, a number of examples demonstrating text input improvement was prepared.

Data quality indicators were calculated for San Donà di Piave area. Quality of OpenStreetMap and "Public Sector Information" data was assessed. Results are

available as a set of web interactive maps. Frontend is designed in responsive manner. Web pages were tested on desktop and mobile devices. Web maps utilize the described earlier public API. The API was designed to be universal and extremely flexible, that allows us to use the same API for quick quality assessment and improvement tasks, for providing of spatial data for interactive maps, and for calculating statistics required for setting up colours of mapped features, etc. Such flexibility has a disadvantage; users may easily write a request taking long time for execution. To prevent it, maximal duration of a request processing is limited by two seconds (longer requests are rejected). A number of techniques are proposed and applied by GDRS (interactive maps and examples for developers) to prevent the described long-time-requests problem. For spatial requests, area should be limited by bounding box or tile definition. For large areas data should be aggregated by tiles and tile layers should be used in a web application instead of regular bounding box requested layers. Some SQL functions (e.g., levenshtein Distance function) are required longer time. To decrease time of the processing number of elements should be limited by a tile or a bounding box. Two types of tile coordinates are supported by the API: regular tile coordinated and quadkey coordinates. Quadkey coordinates enable us to easily aggregate features or request results for different zoom levels. GSDR provides a set of functions designed for the tile-based processing. Concrete examples are described by the following web page: <https://wgn.gsdr.gq/docex.rvt>.

The main page of the GSDR's web site (see Exhibit 69) consists of the links to various maps, examples and documentation. Links are decorated as icons, this allows us to represent the user all significant parts of the web site graphically on one page.

Exhibit 69: Home page with links

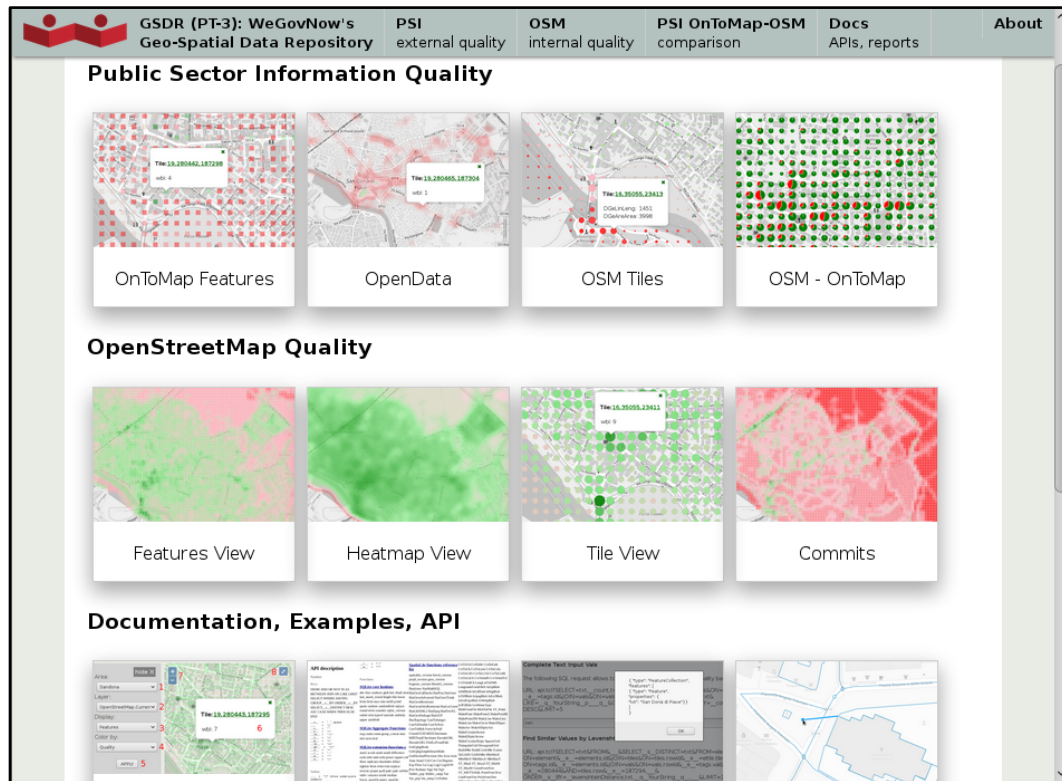


Exhibit 70: Components of GSDR's interface

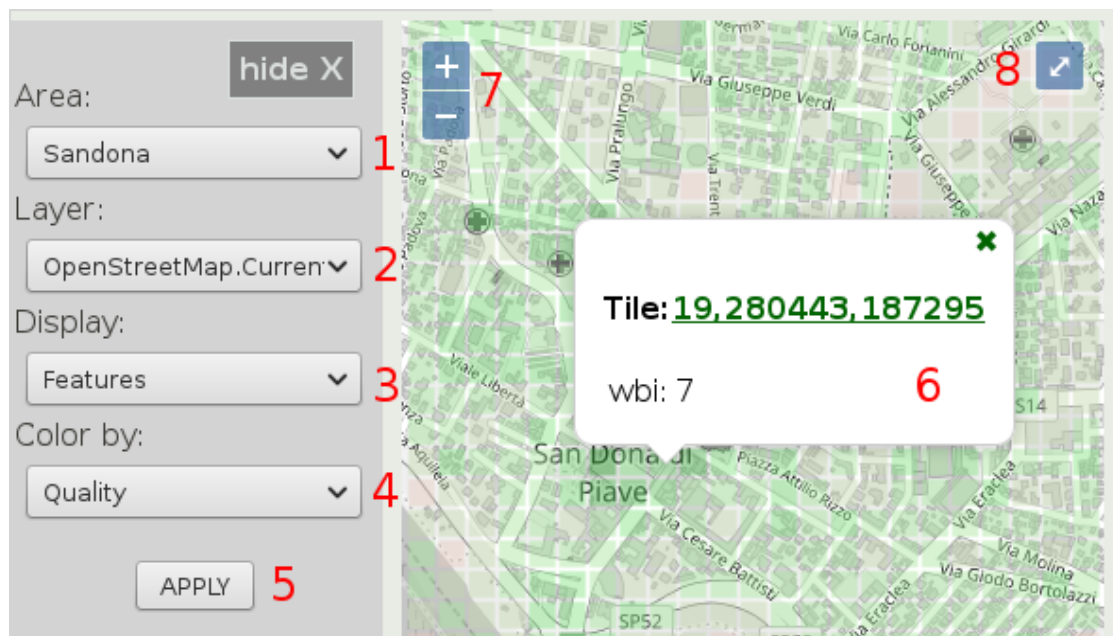


Exhibit 70 describes the frontend elements of GSDR. In the exhibit:

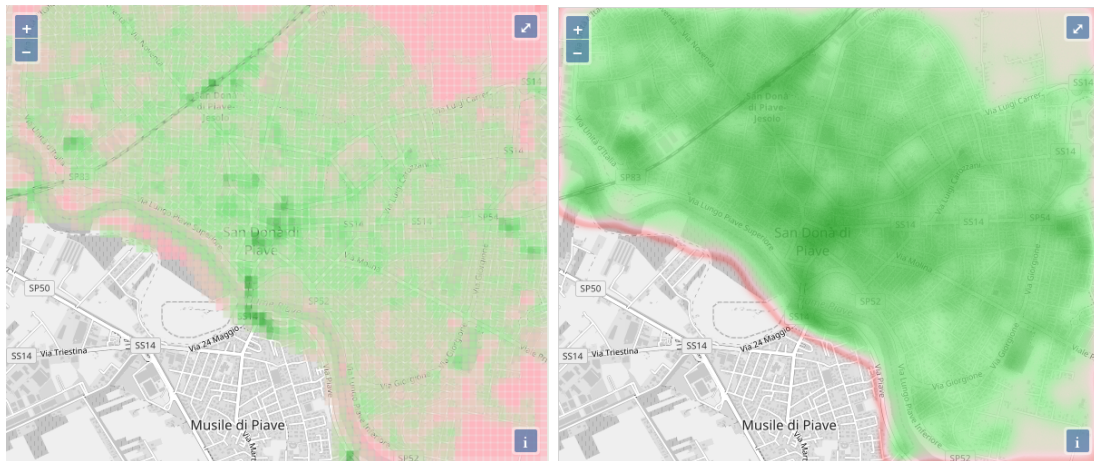
1. Select a pilot site. Currently, the service covers three pilot sites: San Dona di Piave (Venice, Italy), Turin (Italy) and Southwark (London, UK). Moreover, OpenStreetMap data and related quality indicators are available for the Heidelberg area (Germany).

2. One of the following data layers is available.
 - OpenStreetMap.Current - review of OSM current data for a pilot site area.
 - OpenStreetMap.PublicSectorInformation - review of public sector information for a pilot site extracted from OSM data.
 - OnToMap.PublicSectorInformation - review of public sector information provided by OnToMap.
 - OpenData.PublicSectorInformation - review of public sector information provided by OpenData (Municipalities).
 - OpenStreetMap-OnToMap.PublicSectorInformation - comparison of OSM (PSI extraction) vs. OTM PSI data.
 - OpenStreetMap-OpenData.PublicSectorInformation - comparison of OSM (PSI extraction) vs. OpenData PSI.
3. Map visualization types:
 - Features - represents data as separate square features.
 - Heatmap - represents data as a field continuous information.
 - Tiles - represents data by tiles (features summarize data for current zoom level+3).
4. Select a data attribute to display. The following attributes are available:
 - Quality - aggregated data quality of a layer.
 - DGeNpt - a number of points (Data Geometry Number of Points).
 - DGeLinNumb - a number of lines (Data Geometry Number of Lines).
 - DGeLinLeng - lengths of lines (Data Geometry Line Lengths).
 - DGeAreNumb - a number of polygons (Data Geometry Areas Number).
 - DGeAreLing - a length of polygons' boundaries.
 - DGeAreArea - area of polygons.
 - DAt - a number of attributes.
 - MUh - a number of users' hits (Metadata User Hits).
 - MCoNco - contributors' activity, number of contributors (Metadata Contributors Number of Contributors).
 - MCoNcs - contributors' activity, number of changesets (Metadata Contributors Number of Changesets).
 - MTIAve - an average version of features (Metadata Time Average Version)
 - MTIAti - an average integer timestamp (Metadata Time Average Timestamp)
5. Apply button - update map with respect to form parameters.
6. A popup window - emerging by clicking on a map.

7. Zooming control.
8. Full screen and reset map rotation. A map could be rotated by Shift+LeftMouse on a desktop computer, or by two fingers on a touchscreen.

In Exhibit 71 and Exhibit 72, interactive maps of OSM data quality (SDQI) and source parameters are presented. Currently, three types of views are supported: feature, heatmap and tile views. In order to represent a parameter in the feature and heatmap views, classic bounding box request to a server is applied. For the tile view, requests are implemented separately for each tile visible on a map. This approach is very scalable, because it aggregates requested data according to a tile and, thus, it is applicable to all zoom levels.

Exhibit 71: SDQI in the feature and heatmap view. Bottom: area of polygons (left), number of changesets (right). Red and green represent minimal and maximal values, correspondingly. Zoom level 14. San Donà di Piave, Italy. Components of GSDR's interface



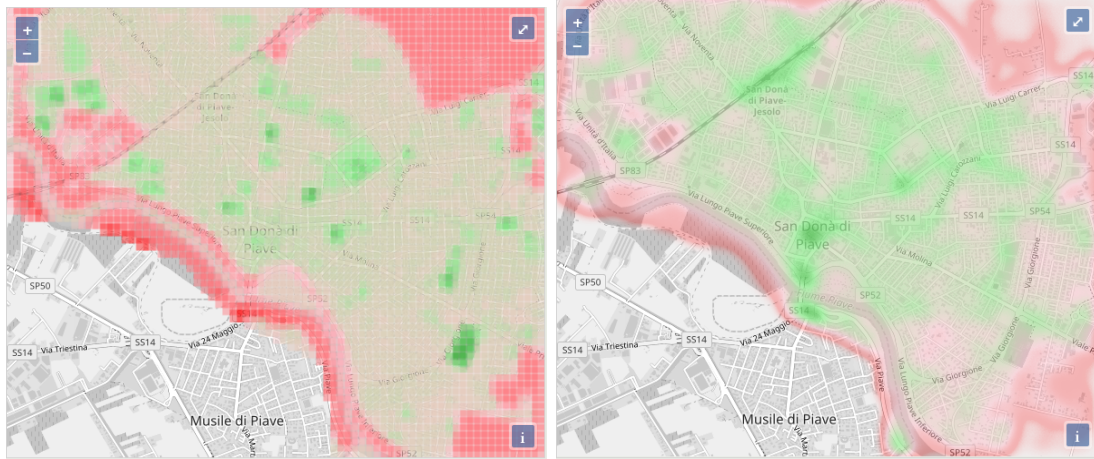
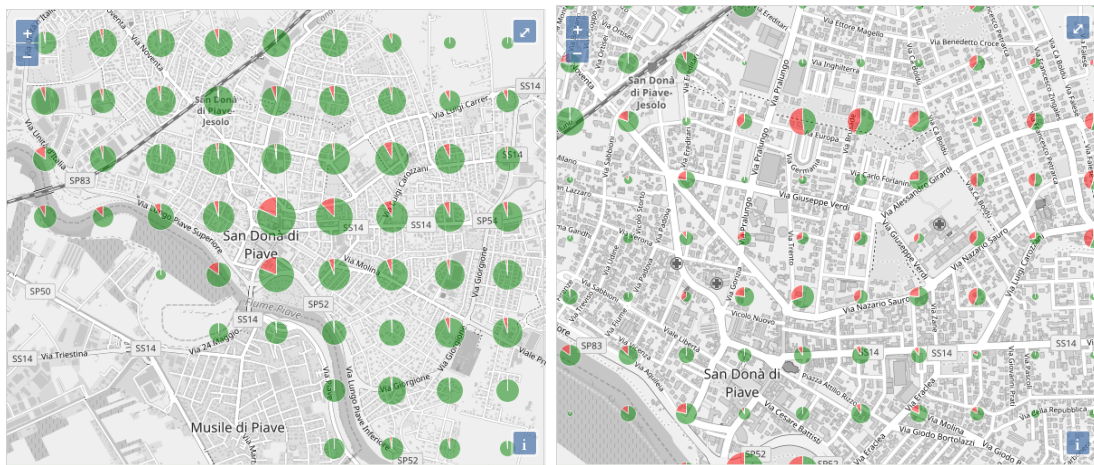
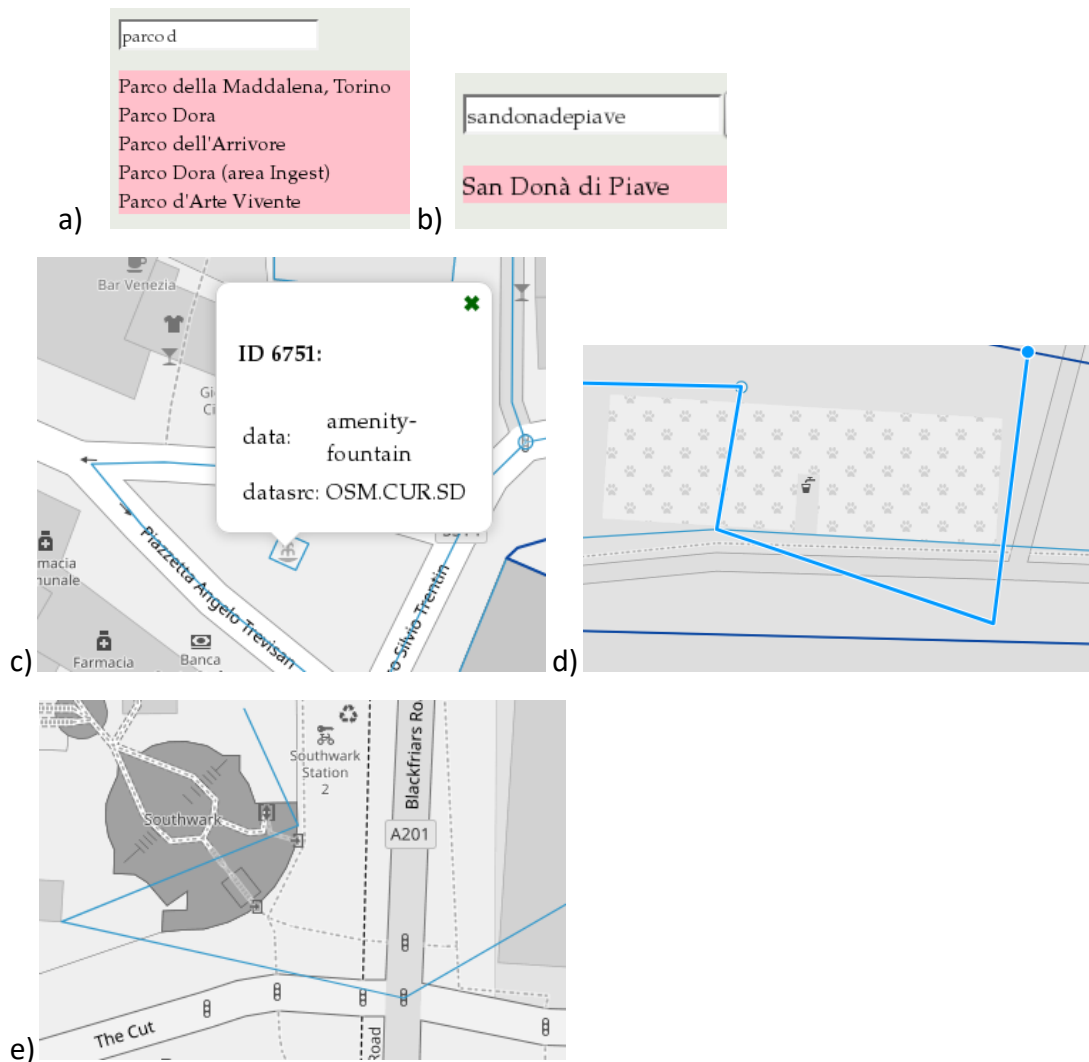


Exhibit 72: Comparison of OSM (red) and PSI (green) data. Left (zoom level 14): data quality, size represents the sum of quality classes. Right (zoom level 15): area of polygons, size represents the sum of areas. San Donà di Piave, Italy



The data quality assessment results are available in form of the archived data and GSDR's interactive web pages. Moreover, we use the results for embeddable instances of GSDR. In general, such instances consist of API and a database comprising only data with acceptable quality (low-quality data is filtered out). Embeddable GSDR enables to implement the following functionality for users' input improvement: auto completing, spell checking, object picking, client- and server-side snapping. Exhibit 73 illustrates this functionality.

Exhibit 73: Users input improvement using an embeddable GSDR instance: a) auto completing (Turin), b) spell checking (San Dona), c) object picking (San Dona), d) client-side snapping (Turin), e) server-side snapping (Southwark).



Examples of pilot sites instances are accessible by the following links:

https://gsdr.gq/docex_sd.html

https://gsdr.gq/docex_tr.html

https://gsdr.gq/docex_sw.html

A detailed description of embeddable GSDR is provided in [EmQ]. Moreover, concrete instructions allowing deployment of an instance of GSDR and all required software and data are provided by the following links:

<https://wgn.gsdr.gq/emgsdrwgn.tar.gz>

https://wgn.gsdr.gq/instructions_emgsdr_centos7.txt

Furthermore, all data and source code was archived using Zenodo repository:

<https://doi.org/10.5281/zenodo.2454150>

<https://doi.org/10.5281/zenodo.2455025>

The development version of libraries and scripts are contributed to the following projects: <http://igis.tk> and <http://tiles.cf>.

7. WeGovNow Data Quality Assessment Results

Data quality assessment of urban data is a batch activity procedure aiming at providing quality contextual data to WGN users. The assessment should answer the question “is the source good enough to support users’ activities?”. In other words, the cartographic information and open data provided by WGN should not mislead users in their evaluations but should, instead, support the collaboration and cooperation within the platform.

WGN platform handles the following data sources:

- OpenStreetMap (OSM) data,
- Public Sector Information (PSI) and open data,
- FisiLife tiled maps data.

Each of the different types of data is further analyzed below. OpenStreetMap (OSM) is utilized mainly as a base map for WeGovNow applications. Public Sector Information (PSI) and open data are provided through OnToMap. Original open dataset files have been assessed as well. Point, polyline and polygon features related to a city infrastructure (e.g., bike lanes, urban parks, hospitals, etc.) are provided as PSI datasets. WeGovNow’s applications widely utilize PSI data. It allows users to select geospatial objects on an interactive map, derive and manipulate with attribute information (e.g., address of a cinema stored as an attribute), search (e.g., find a shop by its name), etc. PSI data could be retrieved by OnToMap’s API. An application of WGN may derive PSI data by this API using different parameters (e.g., bounding box, language, etc.). The datasets are provided in GeoJSON format. Higher level manipulations may be implemented on a side of a specific application. Objects of city infrastructure are provided as point, polygon and polyline features. Currently, the following data types are provided: urban parks, schools, stores, markets, bike lanes, restaurants, museums, places of worship, monuments, libraries, health social services, hospitals, drug stores, law enforcement objects, art galleries, clubs, etc.

Earlier, it has been decided to implement a web service for data quality assessment and improvement. Geo-Spatial Data Repository (GSDR, <https://wgn.gsdr.gq>) has been developed for WeGovNow project needs. It provides the following functionality:

- Intrinsic data quality assessment,
- Comparable data quality assessment,
- Presenting data quality assessment results in interactive maps and auto-generated report,
- Public APIs for quick data check and retrieval.

Ground truth reference datasets are not available in the frame of the project. Despite this, data quality could be evaluated and improved by the methods mentioned above. Intrinsic quality assessment allows defining data quality of OSM data. Additionally, the service is designed for detecting errors in PSI and WGN apps’ data. Quality and

imperfections of PSI data are defined by the comparison with OSM data with known quality calculated intrinsically.

The core component of GSDR is a database. The main harvesting data types are as follows: (1) OpenStreetMap (OSM) covering pilot sites, (2) original PSI data files (in the ESRI shape-file format), (3) PSI data provided by OnToMap, (4) Tiled Raster Maps utilized by WeGovNow and provided by the FirstLife's infrastructure.

OSM is a very dynamic map, thus OSM data is automatically updated every three months. This functionality is supported by the aforementioned web service.

In general, a process of quality evaluation may be described as follows. First, quality of OSM data is defined intrinsically at the first step. Next, quality of PSI data is evaluated using OSM data with known quality. It should be mentioned that many objects provided by the PSI datasets of OnToMap are presented in OSM.

The following publications provide the detailed methodology designed for the WeGovNow project:

- [BFS] Noskov A. and Zipf A., "Backend and Frontend Strategies for Deployment of WebGIS Services", Proc. SPIE 10773, Sixth International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2018), 107730I (6 August 2018); Internal-Link DOI: 10.1117/12.2322831
- [PoWQ] Noskov, A.: Smart City WebGIS Applications: Proof of Work Concept for High-Level Quality-of-Service Assurance, ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci., IV-4/W7, 99-106, DOI: 10.5194/isprs-annals-IV-4-W7-99-2018, 2018.
- [RTiles] Noskov A., "Computer Vision Approaches for Big Geo-Spatial Data: Quality Assessment of Raster Tiled Web Maps for Smart City Solutions" Proceedings 7th International Conference on Cartography and GIS, Vol. 1, Editors: T. Bandrova & M. Konecny, Publisher: Bulgarian Cartographic Association, ISSN: 1314-0604, pp. 296-305, June 18-23 2018 DOI: 10.5281/zenodo.1346671
- [QConc] Noskov A., Zipf A. and Rousell A. "Data Quality Concept for e-Government Web-Map Based Services" Proceedings 7th International Conference on Cartography and GIS, Vol. 1, Editors: T. Bandrova & M. Konecny, Publisher: Bulgarian Cartographic Association, ISSN: 1314-0604, pp. 306-315, June 18-23 2018 Internal-Link External-Link DOI: 10.5281/zenodo.1314215
- [OSMMod] Noskov, A., Grinberger, A. Y., Papapesios, N., Rousell, A., Troilo, R., & Zipf, A. (2019), "Modelling and Assessing Spatial Big Data: Use Cases of the OpenStreetMap Full-History Dump." In A. Voghera, & L. La Riccia (Eds.), Spatial Planning in the Big Data Revolution (pp. 16-44). Hershey, PA: IGI Global. DOI:10.4018/978-1-5225-7927-4.ch002

- [EmQ] Noskov A. and Zipf A.: Open-data-driven embeddable quality management services for map-based web applications. Big Earth Data. Taylor & Frances DOI:10.1080/20964471.2019.1592077

In [BFS] main principles behind GSDR are described. [PoWQ] introduces a methodology for assuring the quality of web applications. [RTiles] provides solutions for automatic quality assessment of raster web-map tiles (RWMT) designed for WeGovNow. [QConc] presents a concept of data quality utilized in the project. [OSMMod] introduces a novel approach to OSM data evaluation. [EmQ] summarizes the WeGovNow data quality related activity.

7.1. Resulting Datasets

The resulting datasets have been achieved using the Zenodo service:

<https://doi.org/10.5281/zenodo.2454150>

Exhibit 74 shows a list of the resulting files. The *.sqlite files are Spatialite GSDR databases. *.csv.gz are comma-separated value compressed text files. They mainly provide content dumps of the full GSDR database's tables. *.linestats.txt are line statistics resulting files. linestats.html provides the line statistics in human-readable HTML format. *.poly provide boundaries of pilot sites suitable for various OSM tools. *.osm.bz2 are OSM data clipped from the OSM data dump. *.json.gz are compressed GeoJSON files providing masks utilized for the extraction of OSM (sdqimask.json.gz), and OnToMap's PSI (sdqipsimask.json.gz) data for GSDR pilot sites' instances. ontomapdata.tar.gz archives data layers provided by OnToMap. opendata_sandona.zip, opendata_southwark.rar, and opendata_turin.rar comprise original (without modifications) Open Data in the ESRI shape-file format. wgntilelog.sqlite.gz is a database comprises OSM tile hits data utilized for quality assessment.

Exhibit 74: Archiving data: data files information.

#	File name	MD5 sum	File size
1	corpts.csv.gz	3865626c9ea29968292484fb206da4b1	40.1 Mb
2	edpotmdist.csv.gz	f73c07bed9af6739ae9db25c746e548d	6.8 Mb
3	elements.csv.gz	b8beca49911f8d6145409a5af6f1ae47	25.1 Mb
4	eltile.csv.gz	4f977a80deef8ac0fe50deef39320edc	5.1 Mb
5	gsdr_wgn_sd.sqlite	c44ccee39d13635a5cc74e2616e0c9ad	22.3 Mb
6	gsdr_wgn_sw.sqlite	109b925b8a75f13d2e461f1c76702a43	119.8 Mb

7	gsdr_wgn_tr.sqlite	be1f4e884f7e6c9e847adbd89db1efe a	71.2 Mb
8	hd.linestats.txt	822e2f76023ffea8caaa371fe9a8736a	428 B
9	hd.osm.bz2	f18b837a6111dba48fde92cdef966c2 e	12 Mb
10	hd.poly	c885be83c702450c2d54d2d0460bccc 0	145 B
11	keys.csv.gz	ccf7eb4cdb38e8a49550909bfde9439 8	20 Kb
12	linestats.html	e6309a4d625488bc9ef1999ba4a1fbf 3	11 Kb
13	nocorpts.csv.gz	ab4a08619f81d6bad0508199513c54f a	3.1 Mb
14	ontomapdata.tar.gz	6f388e92f541d0573a99b7292a06f40 1	8.7 Mb
15	opendata_sandona.zip	09bd68eabc476b07d1bf81b6c994af5 d	357 Kb
16	opendata_southwark.r ar	cb2ed202374bdb194a66ddf153d9bb 71	4.9 Mb
17	opendata_turin.rar	47f5e0216c2f67a57b24b7c83aae0f58	4.1 Mb
18	osmrast_ptsnum.csv.gz	e18b3d4ba49acaafaf8b30f18f175b4f	1.6 Mb
19	osmrastpts.csv.gz	3eb8d2a3d09ce0a50f8b75c812c9d21 1	2.4 Gb
20	qwgnsgdata.sqlite.gz	aa7fb169091d4614b465370dfd5e69f c	836.3 Mb
21	sandona.poly	b732e21887670c171b8e175f981b12 8e	2 Kb
22	sd.linestats.txt	d54de1a26cabf40930a2b562da1cd5b 7	418 B
23	sd.osm.bz2	e959f35c59c210cfa880608959f7c0ed	2.2 Mb
24	sdqidata.csv.gz	1b9258bb95b1686eeff1fa0d7c41268 7	5.8 Mb
25	sdqimask.json.gz	0f3c9b125e30419cffab57207bcbb772	132 Kb
26	sdqipsimask.json.gz	14e0e2249aff8a8cf78ffeb93b87d7e1	99 Kb
27	southwark.poly	51c1290a55bc8cbab86b119c68e26c4 b	150 B
28	sw.linestats.txt	b591bd22b8ddae07e5709b1705e3fd 5f	430 B
29	sw.osm.bz2	92004647668d496977eb4b8ea446ee a0	12.1 Mb
30	tags.csv.gz	1c90d903f437cd7fc2263e3ae98b173 4	16.6 Mb

31	tiles.csv.gz	3165f5d0a53c87d343a79cb235822f61	4.2 Mb
32	tilesizes.csv.gz	71f961b960a22789034f32140ae41ea3	2.9 Mb
33	tr.linestats.txt	ff147a233ccd1e53f53d5d28259a1976	426 B
34	tr.osm.bz2	100800c7305153f57cfb805fcca6e874	6 Mb
35	turin.poly	f86130b138e50766b1c34e839b0f4d72	2 Kb
36	vals.csv.gz	3ab1a1eb1bbbacf3070b323534f9f769	2.7 Mb
37	wgntilelog.sqlite.gz	e22b116d70fbfa01484b937725ed6376	460.1 Mb

7.2. Open Data Evaluation

An Equidistant points (EDP) based approach to the Open Data evaluation is described in [EmQ]. The central principle is as follows. EDP are calculated and saved to a separate data layer. For this, point data are taken unchanged. EDP reside on line features. Polygon objects are treated as line features; for this, boundaries are processed. We calculated EDP for two data types: OSM and OnToMap. For OSM, a one-meter interval between points is applied. In order to reduce a time for further processing, a three-meter interval is used for the OnToMap data.

We calculated the shortest distances between points of two datasets. It allows us to define an average shortest distance from an OnToMap feature to OSM objects. This approach enables to mark outstanding (in a negative sense) data layers. Notice that all points, nodes, and vertices are included into EDP datasets. Moreover, EDP facilitates evaluation of the correctness on the import of Open Data into OnToMap. For this, we prepare one-meter EDP for both datasets and detect Open Data EDP without correspondences in OnToMap.

For a comparative evaluation of the OnToMap data we define the shortest distance from corresponding points to the closest OSM feature (i.e., point, line or boundary). Then, for each layer, we aggregate a set of parameters: number of EDP, minimal, maximal, average, median values, and standard deviation of shortest distances. In order to speed up the processing, instead of OSM features we use corresponding one-meter EDP, while for OnToMap three-meter EDP are calculated.

EDP results enable us to evaluate the OnToMap. For this, we compare OnToMap data with source Open Data and, then, with OSM. The OnToMap and Open Data comparison are based on one-meter EDP. *Exhibit 76* provides the resulting EDP statistics. Point integer coordinates in the UTM projection were considered. For each coordinate pair, we calculated the number of points that belong to a pair. If both data

types (OnToMap and Open Data) are linked to a coordinate pair, this pair is marked as correct and, thus, appeared in corpts.csv.gz. Otherwise, it is considered incorrect and stored in nocorpts.csv.gz.

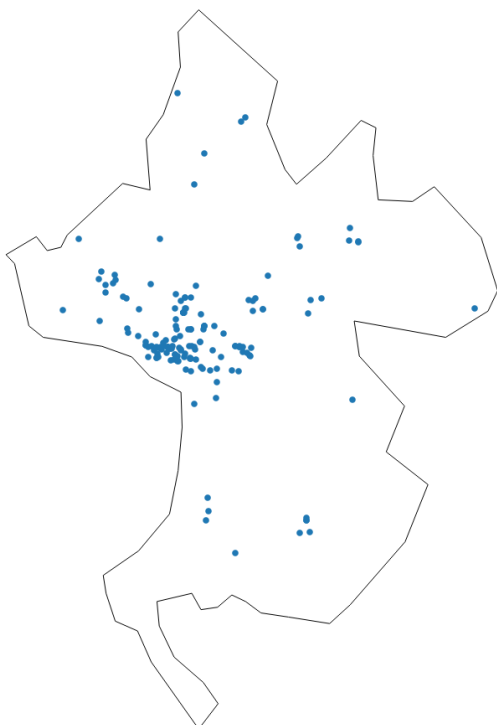
Several inferences can be concluded from the resulting EDP data. First, one can notice that Southwark EDP comprise very few points (52) without correspondences. That indicates that almost all data features from the source Open Data appear almost unchanged in the OnToMap database. Second, the San Dona di Piave EDP datasets comprise up to 160 points without correspondences (points appeared in OnToMap database only). few data sets are not included in the Open Data layers transferred to us. Third, the Turin EDP data have massive points without correspondences; both single Open Data and OnToMap points are marked. It means that many data features were ruled out during the data selecting. Moreover, there are OnToMap points without Open Data pairs. It indicates that several source Open Data layers are not included in the dataset provided for the evaluation.

In *Exhibit 75*, points without correspondences are depicted. Number of points with and without pairs are as follows (number of pair points, without correspondences, ratio, OnToMap single points, and Open Data single points):

- Southwark: 1250867, 52, 0.004%, 26, 26;
- San Dona di Piave: 67994, 160, 0.2%, 160, 0;
- Turin: 1411687, 18803, 1.3%, 3888, 14915.

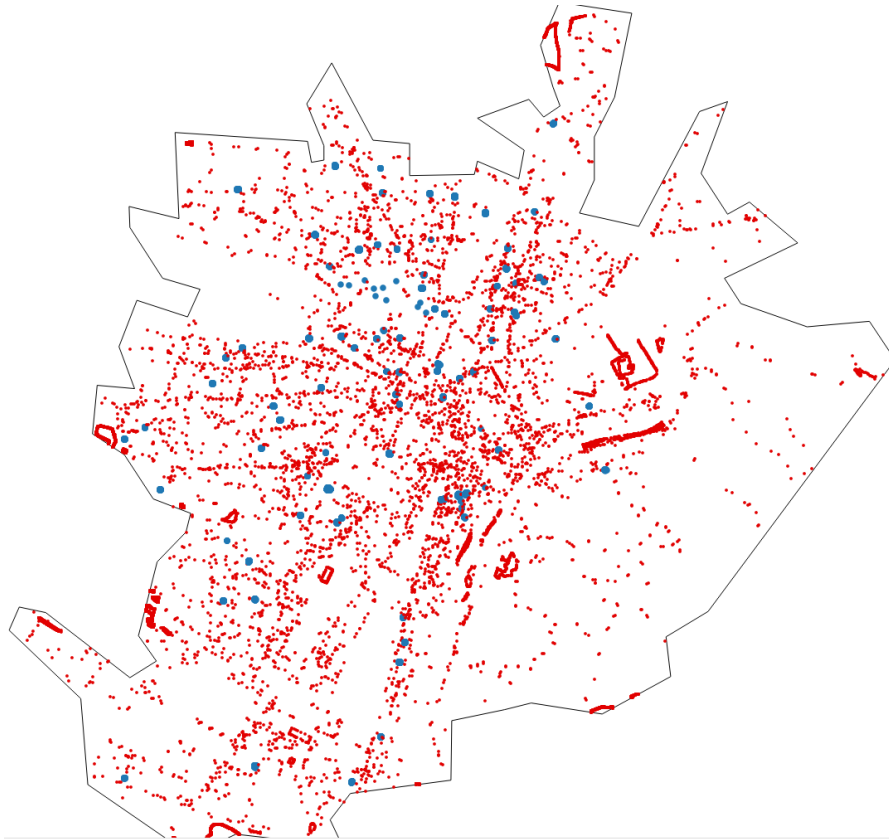
Exhibit 75: EDP without correspondences: blue – OnToMap, red -OpenData

San Dona



Southwark





Turin

EDP data are utilized for the evaluation and filtering out of the OnToMap data against OSM features. The shortest distances from the three-meter OnToMap EDP to the one-meter OSM EDP are calculated (see the `edpotmdist.csv.gz` resulting file). The results are presented in *Exhibit 76* according the pilot sites and layer names. A complete list of the OnToMap's layer names is as follows:

- San Dona di Piave (25 layers): Accommodation, ArtGallery, BicyclePath, Business, Cafe, Cinema, Club, CulturalCenter, FinancialService, FoodAndAccommodation,HealthSocialService, InformationCommunicationService, Leisure, Library, Logistics, Manufacturing, Monument, Museum, PowerDistribution, ProfessionalScientificTechnicalActivity, RealEstateActivity, Restaurant, School, Store, WaterAndWasteDisposal;
- Turin: Cinema, Drugstore, HealthSocialService, Hospital, LawEnforcement, Leisure, PlaceOfWorship, School, StreetMarket, UrbanPark;
- Southwark: BicyclePath, Highway, HistoricalCenter, Monument, ParkingLot, PlaceOfWorship, School, TrainStation, UrbanPark.

Exhibit 76: Shortest distances from three-meter OnToMap EDP to one-meter OSM EDP. Values bigger than the values defined by the rule-of-thumb are marked by a bold font.

Data	Number	Min	Max	Avg	Median	Stdev
PSI.OTM.SD.ArtGallery	1	3.7	3.7	3.7	3.7	0
PSI.OTM.SD.BicyclePath	22327	0	29.9	2.6	1.9	3
PSI.OTM.SD.Business	724	0	29.8	3.1	2.2	3.5
PSI.OTM.SD.Cafe	85	0	15.1	2.9	2.2	2.9
PSI.OTM.SD.Cinema	3	3.1	5.2	4.4	4.8	1.1
PSI.OTM.SD.Club	20	0.4	7.6	3.7	3.5	2.2
PSI.OTM.SD.CulturalCenter	1	1.3	1.3	1.3	1.3	0
PSI.OTM.SD.FinancialService	175	0	15.6	2.8	2.1	2.3
PSI.OTM.SD.FoodAndAccom	181	0.1	19	2.8	2.1	2.8
PSI.OTM.SD.HealthSocialService	94	0.3	17	4.4	3.8	3
PSI.OTM.SD.InformationCom	107	0.2	29	4.2	2.6	4.6
PSI.OTM.SD.Leisure	63	0.1	11.7	3.3	2.8	2.3
PSI.OTM.SD.Library	1	7.1	7.1	7.1	7.1	0
PSI.OTM.SD.Logistics	67	0.2	12.2	3.2	2.7	2.2
PSI.OTM.SD.Manufacturing	262	0.2	24.6	4.2	3.2	3.5
PSI.OTM.SD.Monument	23	0.3	16.3	6.3	5.9	4.2
PSI.OTM.SD.ProfessionalScienti	191	0.2	29.8	4	2.5	5.1
PSI.OTM.SD.RealEstateActivity	996	0	29.8	3.8	2.8	4.1
PSI.OTM.SD.Restaurant	58	0.2	17.1	2.7	2.1	2.6
PSI.OTM.SD.School	65	0.2	11.2	4	3.7	2.7
PSI.OTM.SD.Store	1110	0.1	29.8	3.4	2.5	3.4
PSI.OTM.SD.WaterAndWasteDis	9	0.7	29.8	6.7	4.3	8.9
PSI.OTM.SW.BicyclePath	1014	0	16.5	2.3	1.8	2.1
PSI.OTM.SW.Highway	286418	0	29.9	4.1	2.4	4.9
PSI.OTM.SW.HistoricalCenter	36191	0	29.9	3.5	2	4.4
PSI.OTM.SW.ParkingLot	80533	0	12.5	2.1	1.7	1.6
PSI.OTM.SW.School	390	0.1	26.6	7.3	5.3	6

PSI.OTM.SW.TrainStation	14	0.4	8.8	2.6	1.3	2.7
PSI.OTM.SW.UrbanPark	26496	0	29.7	2.4	1.7	2.6
PSI.OTM.TR.Cinema	61	0.1	5.5	2.3	2.2	1.3
PSI.OTM.TR.Drugstore	225	0.1	9.6	1.2	1	1
PSI.OTM.TR.Hospital	6468	0	29.9	4	1.9	5.3
PSI.OTM.TR.LawEnforcement	66	0.1	6.1	1.4	1.1	1.3
PSI.OTM.TR.PlaceOfWorship	8443	0	29.9	3.7	1.7	5
PSI.OTM.TR.School	61099	0	29.9	4.2	1.8	6
PSI.OTM.TR.StreetMarket	7123	0	29.1	2.4	1.7	2.4
PSI.OTM.TR.UrbanPark	421023	0	29.9	3.9	2.7	4.2

We have carried out a detailed visual and statistical analysis of the datasets and defined the first rule of thumb. The rule says that layers comprise more than 500 points with average, median and standard deviation more than 4, 2 and 4, correspondingly, should be excluded from the further processing, because they are inconsistent with the OSM features. In *Exhibit 76* values meeting the criteria are marked by a bold font. According to the table, "PSI.OTM.SW.Highway" is a low-quality layer which should not be utilized. OSM highway data provided by GSDR.GQ can be used instead.

7.3. Evaluation of Open Street Map Data

In [OSMMod], we have described a novel methodology for the evaluation of OSM data. Further, we provide essential results which are described in detail in [EmQ]. We have introduced the scores allowing users to compare OSM data in pilot sites. According to the resulting scores, the pilot sites are ordered from the lower to higher quality as follows: San Dona di Piave (19), Turin (39), and Southwark (67). The scores facilitate further PSI data quality assessment. In *Exhibit 77*, the quantity of characters of OSM data according to the string classes is provided. The scores are calculated according to the normalized values displayed in parentheses.

Exhibit 77: Line Statistics Results according to the string classes.

Attribute name	San Dona	Southwark	Turin
lines	458375	2661966	1473814
chars	34,919,699	168,024,430	94,605,273

	(2.369E+01)	(1.140E+02)	(6.419E+01)
sblank	1,413,536 (9.591E-01)	8,810,992 (5.978E+00)	4,729,122 (3.209E+00)
atrs	19,365,896 (1.314E+01)	93,092,114 (6.316E+01)	53,381,869 (3.622E+01)
noatrs	14,140,267 (9.594E+00)	66,121,324 (4.486E+01)	36,494,282 (2.476E+01)
stags	1,697,083 (1.151E+00)	10,096,675 (6.851E+00)	5,327,011 (3.614E+00)
ctags	49,580 (3.364E-02)	1414,824 (2.815E-01)	360,556 (2.446E-01)
slashmore	817,592 (5.547E-01)	4,494,286 (3.049E+00)	2,226,518 (1.511E+00)
more	49,579 (3.364E-02)	414,823 (2.815E-01)	360,555 (2.446E-01)
mblanks1	1,724,456 (1.170E+00)	8,335,417 (5.656E+00)	4,623,310 (3.137E+00)
noatrsaz	8,077,521 (5.481E+00)	34,029,882 (2.309E+01)	18,973,022 (1.287E+01)
dquotes	3,448,912 (2.340E+00)	16,670,834 (1.131E+01)	9,246,620 (6.274E+00)
atrsblanks	19,724 (1.338E-02)	1,132,696 (7.685E-01)	343,010 (2.327E-01)
atrs09	12,046,340 (8.174E+00)	46,094,568 (3.128E+01)	26,494,417 (1.798E+01)
atrsaz	2,362,232 (1.603E+00)	20,745,112 (1.408E+01)	12,315,721 (8.356E+00)
atrsAZ	402,052 (2.728E-01)	3,294,365 (2.235E+00)	1,892,739 (1.284E+00)
atrspunct	1,084,071 (7.356E-01)	5,151,912 (3.496E+00)	3,088,311 (2.095E+00)

atrsgraph	2,565 (1.740E-03)	2,627 (1.782E-03)	1,051 (7.131E-04)
SCORE	19	67	39

7.4. Ad-Hoc Quality Assessment of Tiled Raster Maps for Parco Dora

In the Deliverable 1.2, we have introduced an approach to the entropy-based assessment of tiled raster web-maps. OpenStreetMap, Google Maps, and Bing Maps have been examined. It has been disclosed that all maps provide incomplete data. Therefore, FirstLife has started to use another web map source for the Turin pilot site:

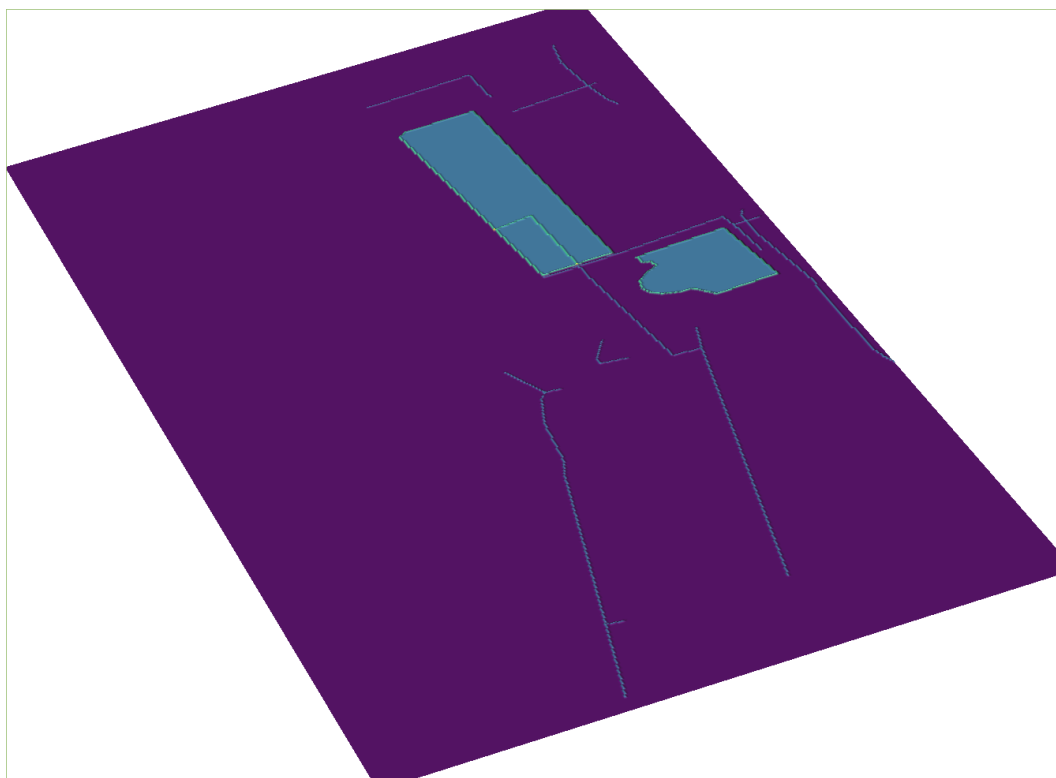
<http://geoportale.comune.torino.it/web/> or http://geomap.reteunitaria.piemonte.it/ws/siccms/coto-01/wmsg01/wms_sicc01_dati_di_base

This source has been involved in the assessment. Blurred features' entropy is utilized for comparable assessment which enables to calculate the amount of information provided by a map and compare maps. Results of calculations are described further. The approach is implemented as a part of GSDR.

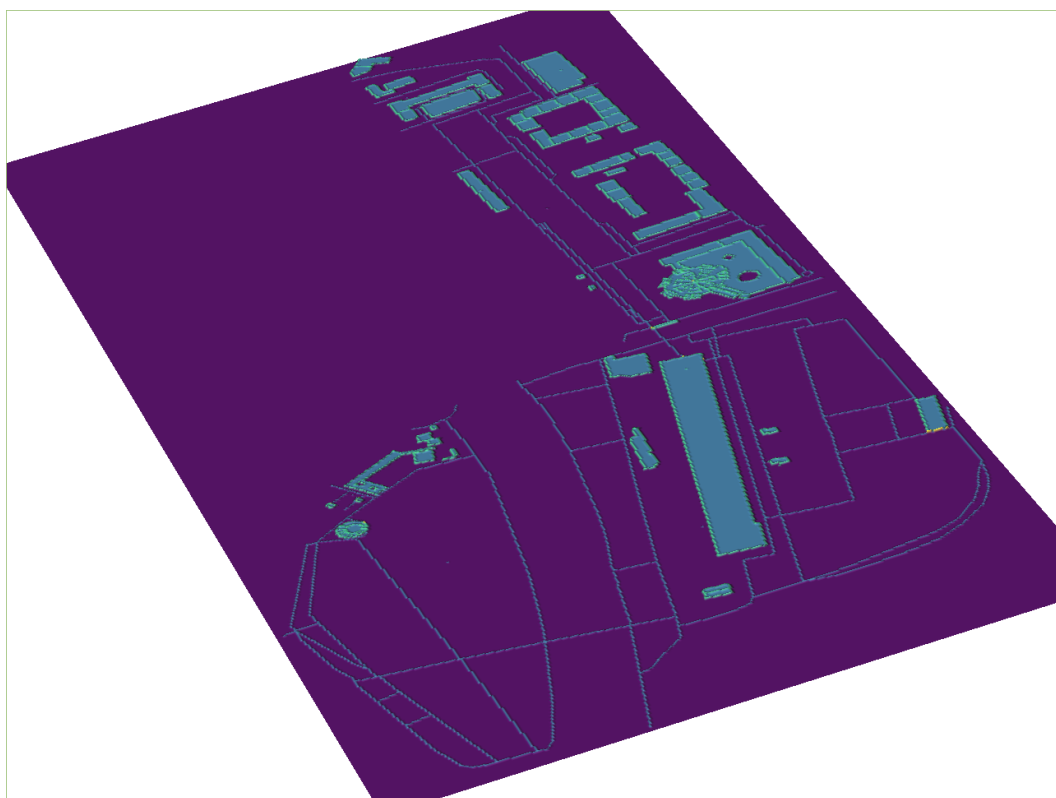
Parco Dora is a pilot area related to several scenarios. Thus, this area has been considered in detail. Using blurred features' entropy, we calculate raster tile comparable completeness provided by popular map services and local map provider for Dara Park's area. In addition to the analysis carried out in January 2017, we conducted an evaluation in January 2019. The results show map updates and allow us to compare maps from different sources. The following three exhibits demonstrate the results. *Exhibit 78* depicts 3D models of extracted information. *Exhibit 79* shows contour line maps of the blurred features' entropy.

Exhibit 78: 3D models of extracted information represented by the blurred features' model (January 2019).

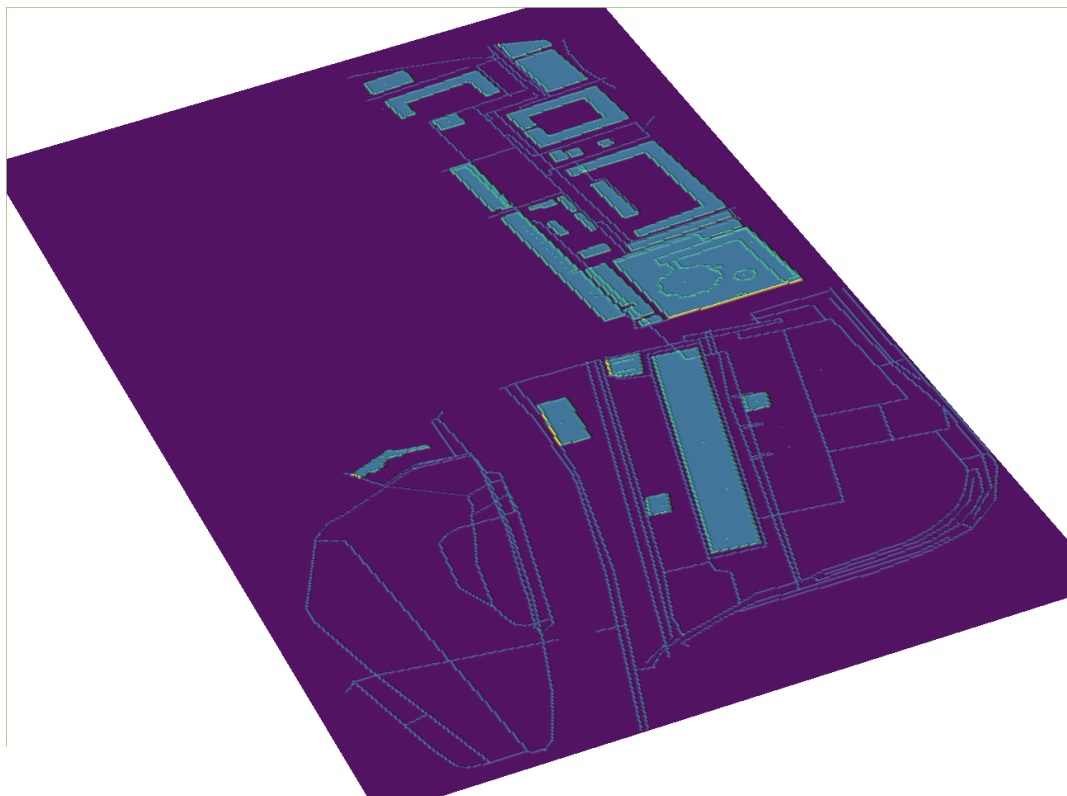
Bing Maps



Google Maps



OSM



Local Data Provider

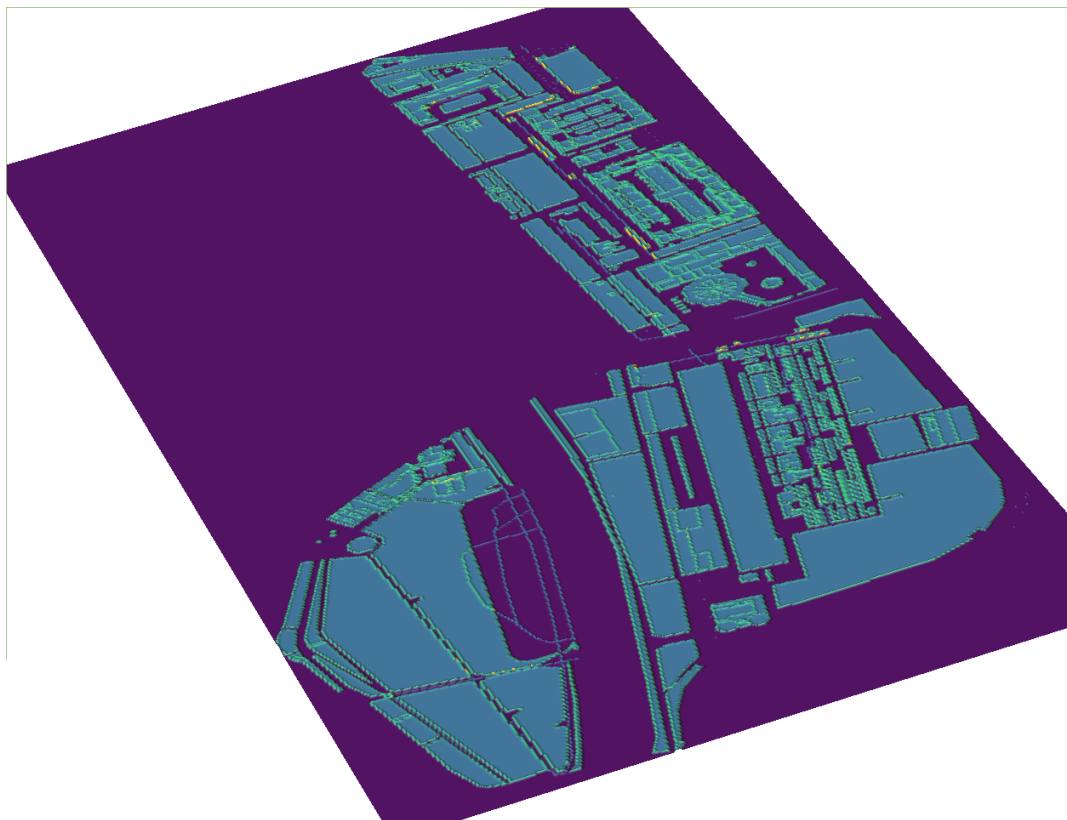
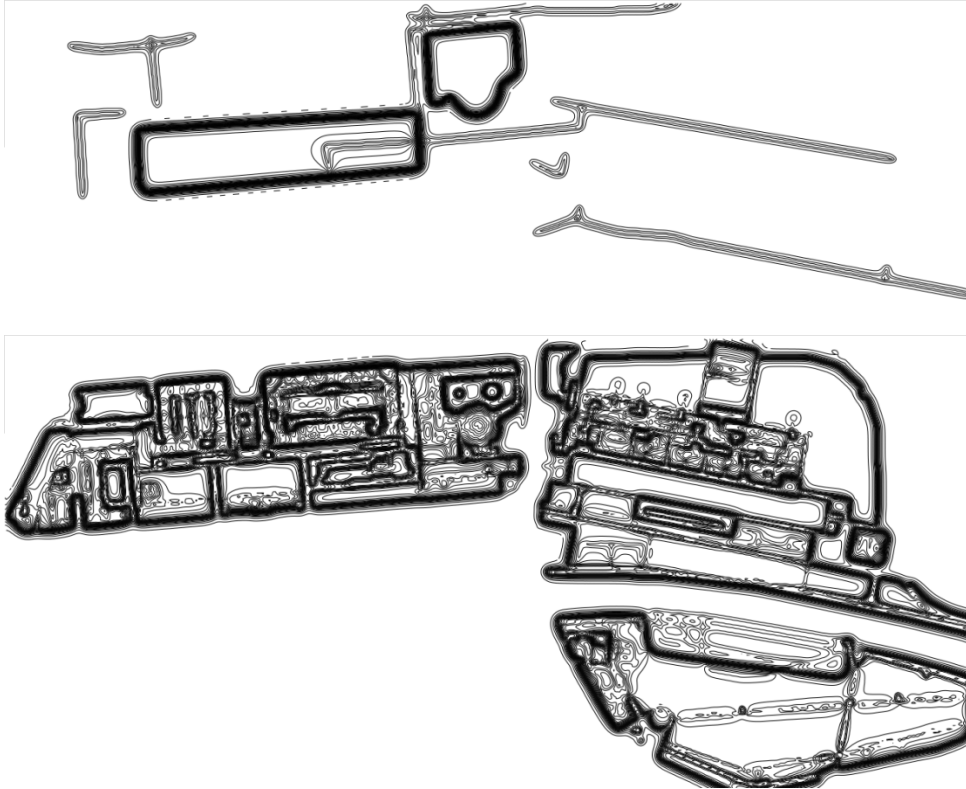


Exhibit 79: Contour maps of the blurred features' model (January 2019). From top to bottom: bing, google, osm and local provider.



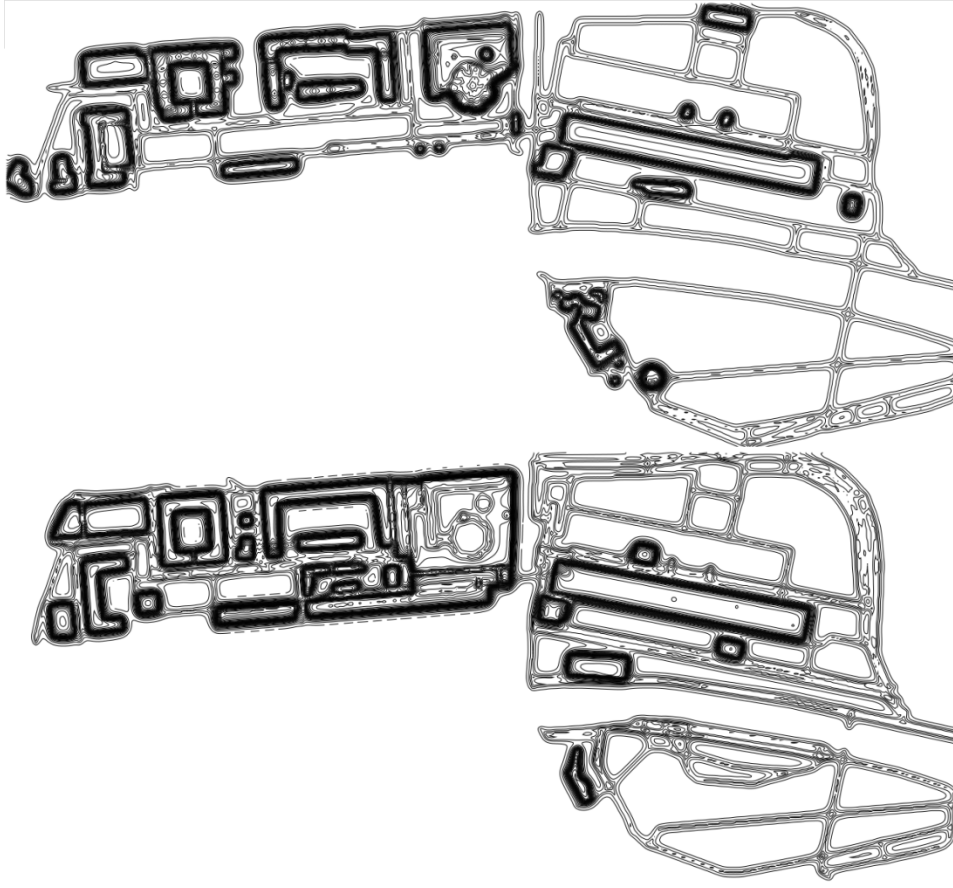


Exhibit 80: Lengths of blurred features' model contour lines (in meters).

	Bing 2017	Bing 2019	Google 2017	Google 2019	OSM 2017	OSM 2019	Turin (local provider) 2019
Overall	39825	62396	146618	163151	150372	206018	279677
2017- 2019,%		36		10		27	
% of Turin	14	22	52	58	54	74	
Polygon	10110	32357	91254	101990	79540	117939	257473
2017- 2019,%		69		11		33	
% of Turin	4	13	35	40	31	46	
Boundary	8264	24230	30829	34194	38465	53437	86129
2017- 2019,%		66		10		28	
% of Turin	10	28	36	40	45	62	
Polyline	28451	29272	88585	88558	76250	99390	48726
2017- 2019,%		3		0		23	
% of Turin	58	60	182	182	156	204	
Point	0	0	515	1044	1831	2779	7581
2017- 2019,%		0		51		34	

% of Turin	0	0	7	14	24	37
------------	---	---	---	----	----	----

According to *Exhibit 80*, the local provider delivers maps with the highest quantity of information. The statistics are broken down to polygon, boundary, polyline and point features. Notice that OSM provides much more information of line features (almost twice). Nevertheless, the local provider's maps contain approximately 25% more information than OSM. Google provides 42% less information, while, Bing comprises only 14% percent in comparison to the local provider. OSM is the most updated map; the information quantity has been increased by 27%. The local provider maps were not evaluated in 2017. From the conducted analysis, one can conclude that choosing the local map provider for the FirstLife's Turin instance was a correct decision.

7.5. Aggregated Data Quality

Simplified Data Quality Indicator (SDQI) is a mock data quality indicator implemented in the frame of GSDR [QConc]. It was designed to evaluate prospective of aggregated data quality indicators and their visualization and utilization. The idea if the SDQI is very simple. A set of parameters are defined. Each parameter represents an essential aspect of data or metadata. Moreover, a lower value indicates lower fitness-of-use and vise versa. The following parameters were defined: number of points, number of lines, length of lines, number of polygons, length of polygons' boundaries, area of polygons, number of attributes. The mentioned parameters could be calculated for almost any vector spatial data. In addition, the following parameters were calculated for OSM data: number of tile hits (represents an OSM tile popularity, it is delivered by OSM planet portal), number of contributors, number of changesets, average version, average timestamp. SDQI is calculated as follows:

$$SDQI = \frac{\left(\sum_{i=0}^n \frac{v_i - \min v_i}{(\max v_i - \min v_i)/10} \right) - \min c}{(\max c - \min c)/10}$$

where, i is a parameter's index, $\min v_i$ and $\max v_i$ are minimal and maximal values of a parameter, correspondingly. $\min c$ and $\max c$ are minimal and maximal class numbers. One can mention that data are split into 10 classes. A lower value indicates lower data quality and vise versa. SDQI is calculated for OSM tile rectangle areas in zoom level 19.

SDQI is calculated for every tile of the pilot sites. SDQI of OSM and PSI data is provided for all pilot sites. PSI's SDQI does not cover OSM specific parameters (number of hits, contributors, changesets, canny points, average version, and timestamps). It is calculated for the comparison of the PSI sources: OSM, OnToMap and Open Data.

As mentioned in [RTiles], files sizes of PNG raster tiled web maps of OSM and Google Maps are compared. File sizes were obtained without actual downloading of PNG files; they have been retrieved from headers provided by a web server. In general,

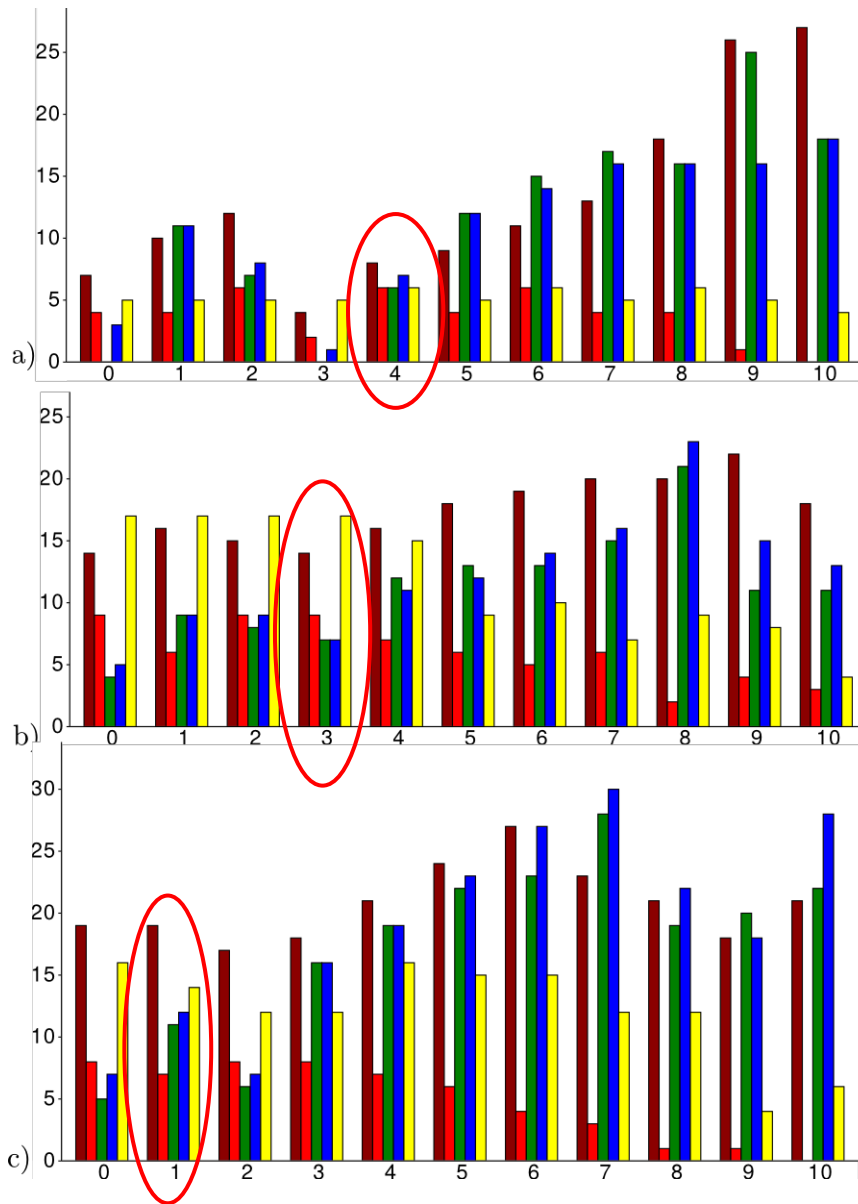
bigger tile size means that it provides a more quantity of information.

According to the harvested size statistics, empty tiles (i.e., tiles filled by only one color) have the following sizes: OSM - 81, Google Maps - 215. Further, we use the difference of these values as a "gap" value ($215 - 81 = 134$). In order to prevent the influence of the PNG encoding model chosen by tile providers, we normalize the OSM - GoogleMaps tile size differences using the gap. Tiles' SDQI groups the resulting statistics of the file-size differences.

In order to define conformance quality level of OSM data, we analyze tile file size according to the SDQI classes. *Exhibit 81* illustrates the results. In the figure, the following parameters are depicted (Y-axis): standard deviation (dark red bars), minimal (red), median (green), average (blue), and maximal (yellow) values of OSM-GoogleMaps file-size differences. All parameters are calculated using the following equation: $(\text{osm_size} - \text{googlemaps_size}) / \text{gap}$. Moreover, minimal and maximal values are divided by 10.

Using the bar charts, in [EmQ], we have specified a rule-of-thumb: a minimally acceptable data quality level is bigger than SDQI class with either minimal or average value more than two of standard deviation, minimal or maximal values, while these two values are not equal zero. According to *Exhibit 81* and the rules-of-thumb, an acceptable data quality level for pilot sites are as follows: (a) San Dona di Piave - 4, (b) Turin - 3, (c) Southwark - 1.

Exhibit 81: File-size statistics (Y-axis) according to the SDQI classes (X-axis). a) San Dona di Piave, b) Turin, c) Southwark. Standard deviation (dark red), minimal (red), median (green), average (blue) and maximal (yellow) values.

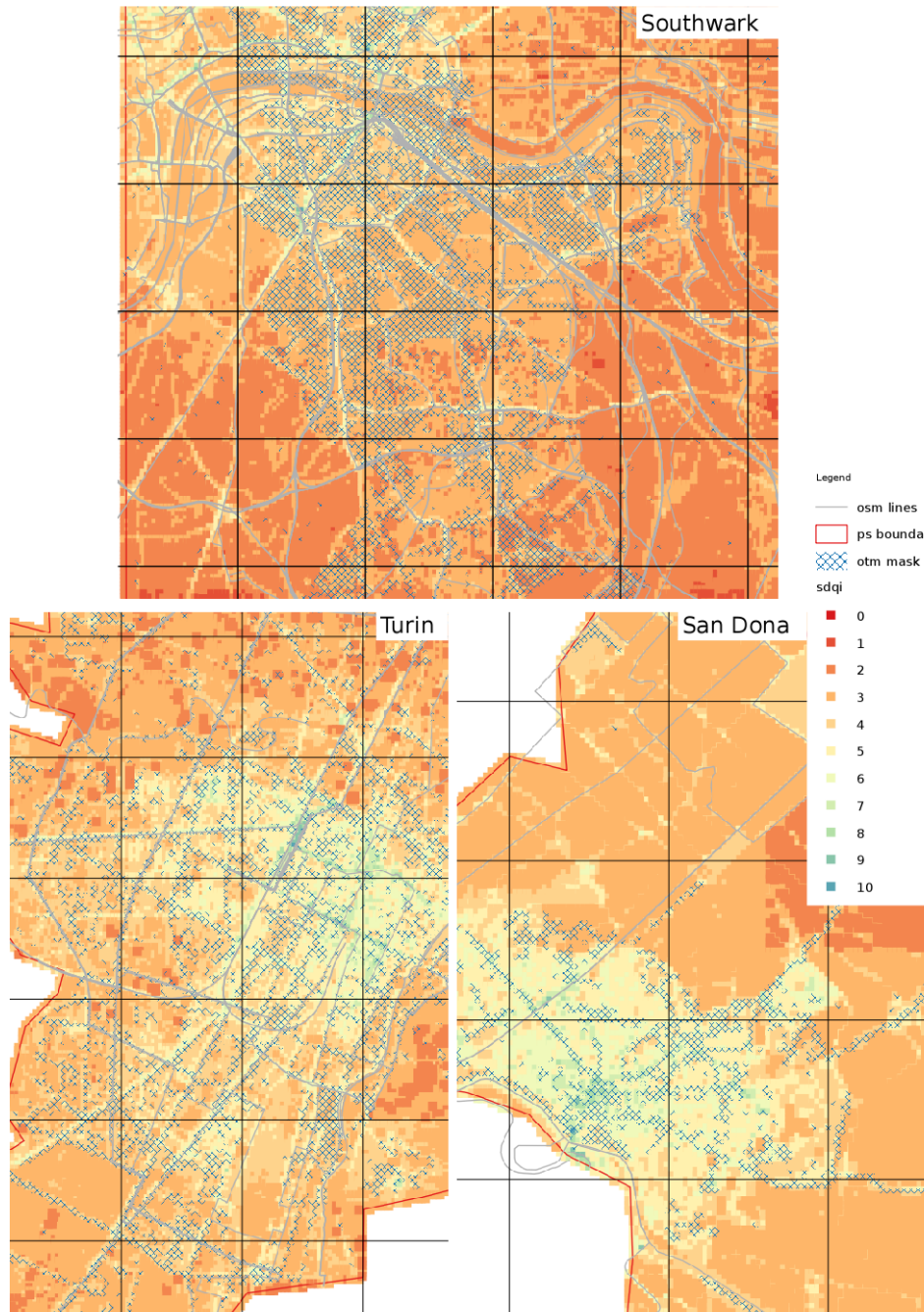


Another rule-of-thumb allows defining minimally acceptable SDQI level for PSI data using the OSM, PSI OnToMap, and PSI OSM SDQI results (see [EmQ]). The rule says that PSI OnToMap SDQI should be more than PSI OSM SDQI for tiles with OSM SDQI more or equal a minimally acceptable data quality level. Moreover, PSI OnToMap SDQI level should have "+2" class in comparison to PSI OSM SDQI for tiles with OSM SDQI less a minimally acceptable data quality level. The rule can be formulated as follows:

$$Q_{psiotm} - Q_{psiosm} \geq -1, \text{ when } Q_{curosm} \geq Q_{minosm}$$

$$Q_{psiotm} - Q_{psiosm} > 2, \text{ when } Q_{curosm} < Q_{minosm}.$$

Exhibit 82: SDQI Results. Color background is OSM SDQI levels. Hatching areas are masks used for the OnToMap data selection.



Tiles corresponding to the described rule are marked in *Exhibit 81* as an OTM mask. The masks depict areas used for the selection PSI OnToMap data for the resulting databases of the pilot sites. In *Exhibit 82*, OSM SDQI classes are displayed by color background. For each pilot site, the rule-of-thumb specifies masks for the selection of OSM data masks. One can notice that significant area of the San Dona di Piave and

Turin sites have SDQI level below a minimally acceptable quality level. Almost whole area of Southwark pilot site is covered by tiles with SDQI level greater than minimally acceptable quality levels. These results are utilized for the preparation of embeddable instances of GSDR for each pilot site (for details see [EmQ]). Moreover, PSI masks represent areas with an acceptable quality of PSI data.

8. Conclusions

On the road to the final prototype of WeGovNow platform was filled by many challenges, we had to overcome in order to design, build and deploy WeGovNow platform. The production-ready version of WeGovNow platform passed three major milestones in terms of three prototype versions of the platform. In general, the purpose of releasing three prototypes is to consolidate, check and test the improvements made by the different development teams involved in WeGovNow project. Specifically, each prototype addresses one phase of the life-cycle of the development process.

The scope of the 1st prototype of WeGovNow platform was to be a proof of concept of the integration of the existing components of WeGovNow, in the technical framework of the platform (as described in “Consolidated System Architecture” D3.1).

The current 2nd prototype of WeGovNow was useful to stress new modules and features we designed and developed within WeGovNow project (new components such as the Landing Page and Trusted Marketplace).

Finally, the 3rd and final prototype includes new components and the bilateral integrations between components, and it will be used for testing focused on the integration of the platform.

Setting up and running multiple instances of WeGovNow platform is part of the technical testing of the platform itself. So far, we managed to solve various minor issues related to the extension of WeGovNow components to support the general architecture, such as CORS errors¹², miss use of https certificates and the like.

In particular, in order to run multiple instances of WeGovNow platform, each development team had to develop a methodology to provide:

- Data isolation
- Instance-based security settings
- Instance-based endpoints
- Version consistence

As emblematic example of the usefulness of testing multiple deployment, the issue on GeoKey/CM present in pt2 has been solved in PT3.

¹² https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS

During the definition of The WeGovNow Consolidated System Architecture (deliverable D3.1), the technical teams of WeGovNow identified a set of requirements to support the integration of WeGovNow components.

The engagement activities of stakeholders at the trial sites and the requirement elicitation process brought us to the definition of a set of general needs, preliminary requirements, daily-based use cases, and expectations based on the stakeholders' intuition of WeGovNow technologies, summarised in deliverable D2.3, Annex II. The collected inputs had been assessed by technical teams, under the light of the scope of the project, of the availability of time and resources, of the existing components and their workflows, resulting in a preliminary list of indications for further development of WeGovNow platform. This list will be completed during the public validation trials.

The assessment (see Annex 6 "Extending WeGovNow platform"), was supported by interviews with the goal of:

1. extend service scenarios from a specific application example to a general pattern of use
2. mapping the fragmented daily use cases in steps of full service processes, involving multiple actors in wide-range of time
3. collecting the missing information about the context and work procedures, important for the adoption of the platform
4. extracting the underlying workflows from the service patterns
5. clustering similar workflows in common we-government processes
6. mapping workflows' steps with components, to highlight missing features and required extensions
7. validation of workflows and corresponding platform functionalities with local municipalities willing to participate

The result of the assessment lead to clusters of requirements (see Annex 6 "Extending WeGovNow platform"):

- Transversal requirements for the adoption of WeGovNow platform by local municipalities,
- Functional requirements to implement the applicative scenarios through WeGovNow components

The release of the final prototype version of the WeGovNow platform represents the basis for setting up the three pilot instances to be publicly operated by each of the three pilot municipalities for validation purposes during the next project year (2018). From the activities described above, a potentially useful "nice to have" feature (an area based calendar) has emerged that is considered to be implemented later in the project, thereby relying on hands-on experiences gained with the pilot platform during the validation trails.

This specific request does not strictly fall within WeGovNow scope because:

1. the use cases referred to coordination within groups and not between organisation, which could be done through internal systems or many web services
2. the booking of resources is managed by the manager of the resource and must be integrated within existing systems and procedures

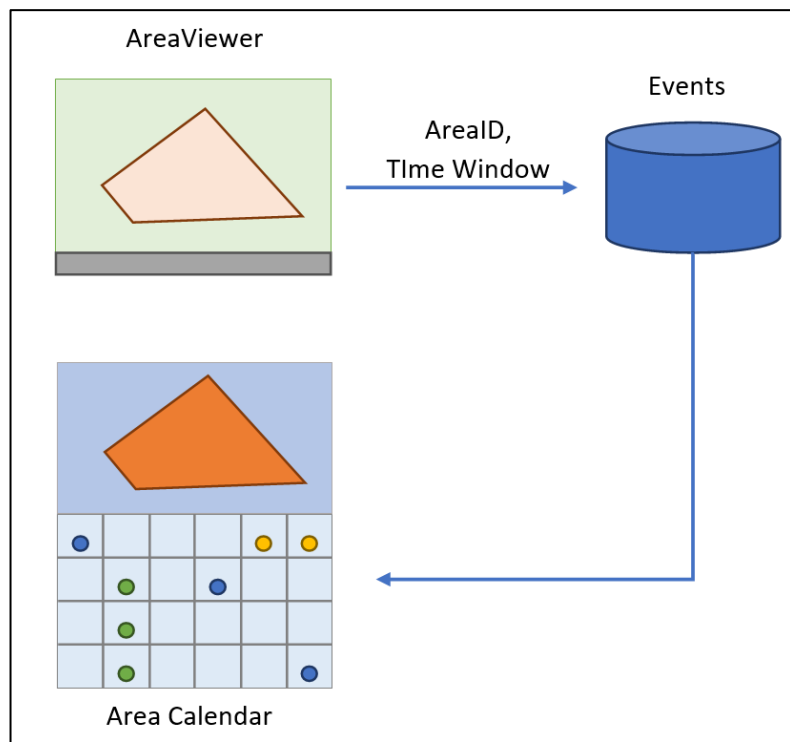
This hypothetical module, AreaCalendar, could provide a global calendar of the area, shared by all actors and groups acting on a specific area (Exhibit 83). Area calendar could be a tool to coordinate the scheduling, search and document public activities, accessible via FirstLife.

Technically, area calendar could be a web module of FirstLife querying FirstLife APIs in search of temporal events of a specific area, in a given time window defined by users. To represent a wider range of urban activities, FirstLife is currently being extended to support periodicity expression, such as:

- Every Friday between 15th of July and end of October
- Every 12 AM between Monday and Friday, Thursday excluded
- The first week of the month

FirstLife temporal extension is extensively described in “Temporal indexing of urban entities”, Annex 7.

Exhibit 83: Concept of AreaCalendar: user will select an area of interest from FirstLife, then go to AreaCalendar to explore the local activities on a calendar view.



The most likely option is to extend FirstLife's calendar exploiting its events and creating a direct link to and AreaViewer-based module such as LandingPage. A second option is to extend LandingPage exploiting OnToMap logger, specifically the existing area-based view of logs. A third option is to extend the Trusted Marketplace. This module has been evaluated not strictly essential, because there are alternative ways to have access to the information therefore it has not been developed as part of the final prototype of WeGovNow, but it could be developed later in the project relying on the evidence coming from the piloting of the platform, with a view to include it for the further mainstreaming of WeGovNow beyond the ending of the pilot duration.

8.1. How to get WeGovNow

The WeGovNow project is made up of various software components that were developed to enable running WeGovNow as an integrated online platform. All software components developed / extended within the project are available as open source solutions for downloading. The "How to get WeGovNow" section of the project website provides all information to support those interested in implementing and / or further developing WeGovNow open source software components. All information and repositories about the civic participation platforms, as well as all integration services are available at the address <https://wegovnow.eu/how-to-get-wegovnow.html>. It contains all details on:

- WeGovNow civic participation software components
 - LiquidFeedback
 - ImproveMyCity
 - FirstLife
 - Offers and Requests and Trusted Marketplace
 - Community Maps
- WeGovNow integration software components
 - Unified WeGovNow User Management
 - OnToMap Logger
 - Navigation Bar
 - OpenStreetMap Joomla Module for WeGovNow
 - Data Quality Management System
 - InputMap
 - Landing Page
 - AreaViewer
 - Tile Server

Annexes

List of Annexes:

1. Logger Endpoint
2. UWUM integration
3. LiquidFeedback work report
4. UWUM work report
5. PgLatLon work report
6. Extending WeGovNow platform
7. Temporal indexing of Urban Entities
8. WeGovNow Environment Cheat Sheet
9. WeGovNow Pilot Vademecum Template

Annex 1

OnToMap Logger and Open Data Management

OnToMap Logger and Open Data Management

The OnToMap Logger has two primary roles in the WeGovNow platform:

- *Centralised logging of user activities within the WeGovNow platform.* The OnToMap Logger is a centralised collector of the history of actions performed by WeGovNow users by interacting with the WeGovNow applications. The goal is that of achieving a unified perspective on user behaviour, e.g., to support cross-application personalization. Within the WeGovNow platform, applications push their log data to a single component (the OnToMap Logger), which merges the information managing a unified history of user activities. In order to effectively support information retrieval, centralised logging has to be coupled with data integration. In fact, as applications can adopt different terminologies, the user activities they log might seem to refer to different types of information even when it is not the case. This leads to the second role of the Logger, described in the following.
- *Data integration across WeGovNow applications.* The Logger provides a unified view on the data shared in the WeGovNow platform, including the Open Data managed by OnToMap, in order to enable WeGovNow applications to retrieve information collected about geographical objects, initiatives, issues, and so forth to populate their maps. As applications can use diverse conceptual models for representing geographical information, heterogeneous data descriptions have to be reconciled. The OnToMap Logger addresses this issue by exploiting the semantic representation of geographical information provided by **OnToMap Ontology**, that is used as an interlingua among WeGovNow applications. The ontology formally represents the concepts in which the data shared among the WeGovNow applications can be classified: the data categories defined by an application can be mapped to the concepts of the ontology, at application integration time, to define the rules for translating the information provided by the application to the common ontology format.

The OnToMap and OTM Logger software can freely be downloaded from <https://ontomap.gitlab.io/wegovnow/>

1 Open Data conversion and import

OnToMap stores public Open Data that can be exploited by the WeGovNow front-end applications and visualized in the Area View, e.g., in order to integrate OpenStreet data with richer descriptions of geographical items in the cities. OnToMap stores Open Data in a Triple Store as Linked Data, in order to allow semantic queries on the data, starting from the domain representation provided by the OnToMap Ontology. In order to import new Open Data, it must be mapped to the appropriate concepts and properties from the OnToMap ontology and it must be converted to RDF (Resource Description Framework), which is the standard format of Linked Data. The mapping and conversion process will produce a RDF dataset, which will be imported in the Triple Store of OnTomap and made available through the OnToMap Data Retrieval API. It is thus expected that each front-end application that needs to visualize some Open Data from the set of information provided by the

Municipality invokes the OnToMap Data Retrieval API; the application will receive the data in GeoJSON format, ready to be visualized on the geographical map.

The first step for importing new data consists in the mapping: the new dataset has to be associated to one or more concepts from the ontology, and an analysis of the properties of the new data items has to be performed in order to associate them to the properties from the ontology and, if necessary, to exclude the properties that are considered as irrelevant. At the moment, mapping new data to the ontology can only be performed manually, by analyzing the new data and matching it to the ontology. As a matter of fact, this task can be hardly automatized because mappings have to be done after having interpreted the meaning of data and of its properties. Most data is represented as shapefiles, or as relational data. Therefore, starting from the metadata of the tables describing data, the mappings can be defined.

1.1 Mapping example

The following image shows an example of the attribute data table of a shapefile:

Exhibit 1. Extract from an attribute table of a shapefile

image	type	days_of_op	hours_of_o	max_stay	no_return	area_m2	specificat	zone	street	featid
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sun	0000-0700; 1000...	40	120	26.61	Aecom	J	WALWORTH ROAD	4516
http://maps.southwark.gov.uk/layer/loadingonly.3333	Short stay	Mon - Sat	0700 - 1900	20	120	24.45	Aecom	B	Melford Road	4533
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40		26.46	Aecom	GR	Grange Road	4520
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40		35.28	Aecom	GR	Grange Road	4519
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sun	0700 - 1900	40	120	29.17	Aecom	B	SOUTHWARK PA...	3258
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40	120	18.01	Aecom	B	BOMBAY STREET	3257
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40	120	16.28	Aecom	B	BLUE ANCHOR L...	3255
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40	120	17.68	Aecom	B	AMBROSE STREET	2654
http://maps.southwark.gov.uk/layer/loadingonly.3333	Short stay	Mon - Sat	0700 - 1900	20	120	87.27	Aecom	B	ANCHOR STREET	2651
http://maps.southwark.gov.uk/layer/loadingonly.3333	Short stay	Mon - Fri	0700 - 1900	30	240	47.73	Aecom	B	LORDSHIP LANE	1365
http://maps.southwark.gov.uk/layer/loadingonly.3333	Short stay	Mon - Fri	0700 - 1900	30	240	136.38	Aecom	B	LORDSHIP LANE	1310
http://maps.southwark.gov.uk/layer/loadingonly.3333	Short stay	Mon - Fri	0700 - 1900	30	240	18.40	Aecom	B	LORDSHIP LANE	1264
http://maps.southwark.gov.uk/layer/loadingonly.3333	Loading only	Mon - Sat	0700 - 1900	40	60	23.46	Aecom	B	WHATELEY ROAD	1231
http://maps.southwark.gov.uk/layer/loadingonly.3333	Electric vehicle re...	Mon - Sat	0700 - 1900			11.71	Aecom	B	DANBY STREET	158
http://maps.southwark.gov.uk/layer/loadingonly.3333	Coach (short stay)	Mon - Fri	0800 - 1000, 150...			169.22	Non standard siz...	B	Townley Road	4544

The table shown in Exhibit 1 belongs to a shapefile containing data about parking bays in Southwark. The OnToMap ontology already contains a suitable concept (ParkingLot), therefore it is not necessary to add a new concept to the ontology.

The relevant attributes from the source shapefile are “street”, “featid”, “type” and “image”; they can be mapped to the following ontology properties:

“street” -> “hasAddress”

“featid” -> “hasID”

`"type" -> "hasCharacterization"`

`"image" -> "hasURL"`

Once the mapping has been performed, the actual conversion process can take place. The data must use EPSG:4326 as coordinate reference system; if necessary, GIS software (e.g. QGIS) can be used to convert the reference system of the data to EPSG:4326.

The tool used so far for this task is GeoTriples (<https://github.com/LinkedEOData/GeoTriples>), which supports spatially-enabled relational databases (e.g. PostGIS), ESRI shapefiles and XML, GML, KML, JSON, GeoJSON and CSV documents as input.

GeoTriples provides a GUI which can be used to read the input data and generate a RML mapping, which specifies the ontology concept/properties to which the input data and its properties are mapped. The RML mapping generated by GeoTriples must then be edited to reflect the mappings from the first step and to add additional properties not included in the input data (e.g. a label, provenance, etc.). If the new data has to be available only to specific OnToMap instances, a "applicationInstance" property has to be added to the mapping; this property may have one or more values from a predetermined set, which at the moment includes "torino", "sandona" and "southwark" referring to the three pilot OnToMap instances. The last step consists in setting the RDF vocabulary to GeoSPARQL and the reference system to 4326.

At this point, GeoTriples is able to correctly convert the input data into RDF; the output is a text file containing the selected RDF serialization of the input data. This file is then ready to be imported in a Triple Store. Currently, OnToMap doesn't provide any API for loading new datasets, therefore the new data has to be imported from the Triple Store itself. Parliament (<http://parliament.semwebcentral.org/>), the Triple Store used in OnToMap, provides a web interface for the import of new RDF data.

2 API access control

The OnToMap API uses X.509 client certificates (signed by LiquidFeedback) for authenticating client applications. This authentication method guarantees the recognition of the identity of the client applications in a reliable way; however, it is not trivial to implement it in certain cases (e.g. in the case of AJAX requests performed directly from web pages). For this reason, a token-based authentication method has been implemented; this authentication method allows applications to get data from the logger and data retrieval APIs without submitting a client certificate with each request.

A token can be obtained by submitting a GET request, including the application dev/test certificate, to https://api.ontomap.eu/api/v1/access_token; a JSON object containing the token will be included in the response. If a token must be replaced (in case it is lost or compromised), a POST request, including the application dev/test certificate, to https://api.ontomap.eu/api/v1/access_token/renew has to be submitted; the previous token will be invalidated and the new token will be included in the response.

Once a token has been obtained, it can be used as the value of the “token” request parameter. An example of the correct use of a token follows:

```
GET https://api.ontomap.eu/api/v1/logger/events?token=abc12345
```

3 Logger API

3.1 Workflow

In order to use the OnToMap Logger APIs to publish user activity events, client applications must perform some preparatory steps, which are listed below:

- 1 Obtain a valid client certificate from LiquidFeedback;
- 2 Generate a valid mapping associating the application concepts and properties to those from the OnToMap ontology;
- 3 Use the OnToMap Mapping APIs to submit the mappings;
- 4 (optional) obtain an access token for read-only access to the logger data, without using certificates.

3.2 Concepts/properties mapping

Since the data submitted to the logger by the applications uses the application specific terminology, mapping rules are necessary in order to translate the data to the OnToMap terminology and, consequently, to ensure that the logging data from different applications uses a common terminology.

The mapping rules must be inserted in a JSON document, which must be submitted to OnToMap through the OnToMap API. The mapping rules consist in a list of concept mappings, which associate each concept of the terminology used by the application to a concept of the OnToMap terminology. For each concept it is possible to include a list of property mappings, each one associating a property in the app terminology to one in the OTM terminology.

All of the concept and property mappings must refer to concepts and properties in the OnToMap ontology.

When a property represents a measure, it is possible to include its measure unit, in order to avoid ambiguities.

When new mapping rules are submitted, any previous rules associated to the application performing the request are overwritten.

An example of valid mapping rules is reported below:

```
{
  "mappings": [
    {
```

```

    "app_concept": "ChildCare",
    "ontomap_concept": "Kindergarten",
    "properties": [
      {
        "app_property": "denominazione",
        "ontomap_property": "hasName"
      },
      {
        "app_property": "indirizzo",
        "ontomap_property": "hasAddress"
      },
      {
        "app_property": "telefono",
        "ontomap_property": "hasPhoneNumber"
      },
      {
        "app_property": "retta_mensile",
        "ontomap_property": "hasMonthlyRate",
        "unit": "EUR"
      }
    ]
  },
  {
    "app_concept": "Ristoranti",
    "ontomap_concept": "Restaurant"
  }
]
}

```

The mappings can be submitted to OnToMap by invoking the API described in the OnToMap documentation: https://ontomap.eu/#api-Mapping_APis-InsertMappingRules.

3.3 Event Logging

User activity events are represented as JSON objects including some mandatory properties (user performing the action, timestamp of the action, type of action) and some optional information. All the event properties are listed below.

Exhibit 2. Name, type and description of all properties used by the OnToMap Logger events

actor	Int	Required. The UWUM ID of the user performing the action.
timestamp	Long	Required. The Unix timestamp (in ms) of the action.
activity_type	String	Required. The user activity type.
activity_objects	Array	An Array of GeoJSON Features . The feature properties must include a hasType field (the concept to which the feature belongs) and an external_url field (deep link to the

		<p>feature). The remaining properties (as well as the feature concept) are expressed in the terminology of the client application, and must be included in the mapping previously sent to the logger by the application.</p> <p>The properties can contain a <code>additionalProperties</code> object: every field contained in this object will not be translated and stored as-is.</p> <p>The same event must not contain multiple <code>activity_objects</code> having the same <code>external_url</code>.</p>
references	Array	<p>An Array of objects.</p> <p>Every object contains two fields: <code>external_url</code> (String) and <code>application</code> (String), referring to a specific feature managed by a specific application.</p>
visibility_details	Array	<p>An Array of objects.</p> <p>Every object contains two fields: <code>external_url</code> (String) and <code>hidden</code> (Boolean).</p> <p>When the <code>visibility_details</code> field is included, the features corresponding to the <code>external_urls</code> included (that must be managed by the application sending the event) are hidden/shown from the data retrieval results, based on the corresponding <code>hidden</code> value.</p> <p>The <code>visibility_details</code> field is used when the action to be logged implies a change of visibility for some features (for example, if a feature is deleted or made private/public).</p> <p>The visibility change of a feature is applied only if the timestamp of the event is greater than or equal to the timestamp of the last logged event containing a visibility change for that feature.</p>
details	Object	<p>An Object that can include further details regarding the user action to be logged. This field is stored as-is.</p>

The full documentation for the event logging API can be found at https://ontomap.eu/#api-Logger_APIs-InsertEvents.

An example invocation of OnToMap Logger to store an event describing a user action is reported in the following:

POST <https://api.ontomap.eu/api/v1/logger/events>

Request body:

```
{
  "event_list": [
    {
      "actor": 12345,
      "timestamp": 1485338648883,
```

```

"activity_type": "object_created",
"activity_objects": [
  {
    "type": "Feature",
    "geometry": {
      "type": "Point",
      "coordinates": [
        7.643340826034546,
        45.07558142970864
      ]
    },
    "properties": {
      "hasType": "Asili",
      "external_url": "http://appl.wegovnow.eu/Asili/1",
      "denominazione": "Asilo 1"
    }
  }
],
"references": [
  {
    "application": "appl.wegovnow.eu",
    "external_url": "http://appl.wegovnow.eu/Asili/22"
  }
],
"visibility_details": [
  {
    "external_url": "http://appl.wegovnow.eu/Asili/56",
    "hidden": true
  }
],
"details": {}
}
]
}

```

3.4 Event retrieval

OnToMap provides an API which returns user activity events stored in the logger. The events are returned in reverse chronological order (from the most to the least recent one). In order to support more specific queries, the API supports several filters which can be combined to return only events meeting certain conditions. The following table lists all the filters implemented so far:

Exhibit 3. Name, type and description of all the filters supported by the OnToMap Logger

actor	Int	If specified, this filter selects the events related to actions performed by the user with the specified UWUM ID.
application	String	If specified, the filter selects the events related to the specified application.

activity_type	String	If specified, it selects the events related to actions of the specified type.
concept	String	If specified, it selects the events which contain activity_objects belonging to the specified concept (OnToMap terminology).
external_url	String	If specified, it selects the events which contain the activity_object having the specified external_url.
reference_concept	String	If specified, it selects the events which refer to activity_objects belonging to the specified concept (OnToMap terminology).
reference_external_url	String	If specified, it selects the events which refer to the activity_object having the specified external_url.
subconcepts	Boolean	If set to true and concept or reference_concept are specified, this filter selects the events which contain or refer to some activity_objects belonging to the specified concept and its subconcepts. The filter has effect only if concept or reference_concept are specified. Default value: false
start_time	Long	If specified, it selects the events having a timestamp greater than or equal to the specified timestamp (Unix timestamp in ms).
end_time	Long	If specified, it selects the events having a timestamp less than or equal to the specified timestamp (Unix timestamp in ms).
boundingbox	String	A string representing the coordinates of the North-East and South-West points of a bounding box (using EPSG:4326 as CRS). If specified, this filter selects the events including or referring to any activity_objects located in the specified bounding box. String format: NE_lng,NE_lat,SW_lng,SW_lat

Also in this case, we report a sample invocation and the returned data items:

```
GET
https://api.ontomap.eu/api/v1/logger/events?boundingbox=7.7,44.9,7.5,45.1
{
  "event_list": [
    {
      "application": "appl.wegovnow.eu",
```



```

    "actor": 67890,
    "timestamp": 14967425487654,
    "activity_type": "profile_updated",
  },
  {
    "application": "app2.wegovnow.eu",
    "actor": 12345,
    "timestamp": 1485338648883,
    "activity_type": "object_created",
    "activity_objects": [
      {
        "type": "Feature",
        "geometry": {
          "type": "Point",
          "coordinates": [
            7.643340826034546,
            45.07558142970864
          ]
        },
        "properties": {
          "hasType": "Asili",
          "external_url": "http://appl.wegovnow.eu/Asili/1",
          "nome": "Asilo 1"
        }
      }
    ],
    "references": [
      {
        "application": "appl.wegovnow.eu",
        "external_url": "http://appl.wegovnow.eu/Asili/22"
      }
    ],
    "visibility_details": [
      {
        "external_url": "http://appl.wegovnow.eu/Asili/56",
        "hidden": true
      }
    ],
    "details": {
      "action_id": "76b66b1a-d846-4a56-8bfb-5aa5b4a3227f"
    }
  },
  {...}
]
}

```

3.5 Request Idempotency

Since the events don't contain a unique ID, the logger will store and list an event multiple times if it is submitted more than once. This could be an issue in case of network problems: if an application submitting an event doesn't receive any response from OnToMap, there are two possible cases: 1) the request didn't reach OnToMap at all, or 2) the request was processed successfully but the

response didn't reach the client application. It would be safe to resubmit the request only in the first case, however the client application has no way to verify whether the request was received or not by OnToMap.

In order to mitigate this problem, the concept of idempotent requests has been introduced. To make use of idempotent requests, it is sufficient to provide a "Idempotency-Key: <key>" request header to the request used to submit events. The key is a client-generated random string; good examples of possible keys are V4 UUIDs.

If a request contains an idempotency key, OnToMap will not save the contained events when a request with the same key has been sent previously; it is then safe to resubmit requests when using idempotency keys.

4 Rate limiting

In order to ensure its availability even under high request load, OnToMap relies on the Nginx web server for request rate limiting.

As described in its documentation, Nginx can be configured to serve requests at a maximum fixed rate ("limit_req_zone" and "limit_req" directives). Normally, requests exceeding the set rate are dropped, unless the "burst" parameter is set. When the "burst" parameter is set, Nginx makes it possible to store exceeding requests into a queue which is used as a buffer to manage the queries in a delayed mode, respecting the maximum rate. When the queue is full, any additional requests are rejected with a 429 error.

At the moment, the maximum request rate set for OnToMap is 20 requests for second, with a queue length of 200 requests.

Annex 2

UWUM integration

UWUM Integration

UWUM API endpoints

The UWUM endpoints for OAuth2 and the integration framework are available at the following URLs:

<https://wegovnow-pt2.liquidfeedback.com/api/1/authorization>

<https://wegovnow-pt2.liquidfeedback.com/api/1/token>

<https://wegovnow-pt2.liquidfeedback.com/api/1/validate>

<https://wegovnow-pt2.liquidfeedback.com/api/1/navigation>

<https://wegovnow-pt2.liquidfeedback.com/api/1/style>

<https://wegovnow-pt2.liquidfeedback.com/api/1/application> (formerly "client")

UWUM scopes

Currently the following access scopes can be granted through UWUM (based on the access levels of LiquidFeedback):

- “authentication” authenticate the current user by reading its
 - o static ID (id)
 - o current screen name (name)
- “identification” identify the current user by reading its unique ident string (identification). Automatically implies scope "authentication"
- “notify_email” read the notification email address of the current user (notify_email)
- “read_contents” read any user generated content (w/o authorship, ratings and votes)
- “read_authors” read the author names of user generated content (author's static ID and screen name)
- “read_ratings” read rating scores by other users
- “read_identities” read the identities of other users (identification)
- “read_profiles” read the profiles of other users (e.g. phone number, self-description)
- “post” post new content
- “rate” rate user generated content (e.g. thumbs up/down, "+1", support an initiative, rate a suggestion)
- “vote” finally vote for/against user generated content in a decision (i.e. vote on an issue)
- “profile” read profile data of current user (e.g. phone number, self-description, ...)
- “settings” read current user's settings (e.g. notification settings, display contrast, ...)
- “update_name” modify user's screen name (name)
- “update_notify_email” modify user's notification email address (notify_email)
- “update_profile” modify profile data (e.g. phone number, self-description, ...)
- “update_settings” modify user settings (e.g. notification settings, display contrast, ...)

Any of these scopes can be suffixed with "_detached" to request the scope for usage without the need for the user to be logged in. This should only be used when it is really needed.

X.509 certificate for client identification

To create a trustworthy relationship between applications using UWUM and the central UWUM component, we will use X.509 certificates. Therefore, any official WeGovNow client is required to provide a valid X.509 certificate with each request made to the central UWUM service. For this purpose we kindly ask all technical partners to provide X.509 certificate signing requests to be signed by the UWUM Certificate Authority.

For more information on X.509 certificates and signing requests, please refer to the documentation of your preferred TLS software such as LibreSSL or OpenSSL.

Integration Checklist

We will support all technical partners during UWUM integration. We defined a number of steps we like to take together with every technical partner. In the following list, the term "client application" refers to the application to be integrated with UWUM:

1. **Availability of application via IPv4.** The client application is available via a defined URL using IPv4.
2. **Availability of application via IPv6.** The client application is also available using IPv6.
3. **Serving via HTTPS.** The client application service is encrypted via HTTPS.
4. **Publicly trusted X.509 certificate for end users.** The client application server provides a publicly trusted X.509 certificate.
5. **OAuth2 redirection endpoint defined.** The URL of the OAuth2 redirection endpoint of the client application has been determined and submitted to LiquidFeedback (FlexiGuided GmbH).
6. **Certificate signing request for UWUM.** A private key for accessing the UWUM API has been created. A corresponding certificate signing request (CSR) has been submitted to LiquidFeedback (FlexiGuided GmbH).
7. **Certificate signed by UWUM Certificate Authority.** A signed certificate for the client application has been sent back to the technical partner.
8. **Successful X.509 secured connection.** The client application has successfully established a secured connection with the UWUM server, e.g. using LiquidFeedback's /info API endpoint.
9. **Authorization endpoint access.** The client application can redirect an end user to the UWUM authorization endpoint.

10. **Authorization endpoint error response handling.** The client application is capable of receiving authorization errors through its OAuth2 endpoint.
11. **Authorization endpoint error display.** The client application is able to display authorization error messages (see 10) to the end user.
12. **Successful authorization request and user identification.** The client application made a successful authorization request, received a UWUM access token, and determined the end user ID.
13. **Using access token for API calls to other components.** The client application has successfully performed a LiquidFeedback API call (e.g. to the /info API endpoint) using a previously obtained UWUM access token.
14. **Accepting access token from other components.** The client application (acting as resource server) provides at least one API call which accepts a UWUM access token for authorization.
15. **Access token verification.** The client application (acting as resource server) is capable of verifying the validity and scope of a UWUM access token passed from another component (see 14).
16. **Access token verification errors.** The client application (acting as resource server) is capable of handling error responses during validation of UWUM tokens (see 15).
17. **Accepting access tokens as "Authorization" header.** In conformance with RFC 6750 (Bearer Token Usage), the client application (acting as resource server) accepts UWUM access tokens through the HTTP request header field "Authorization".
18. **Cross-origin resource sharing (CORS).** The client application (acting as resource server) allows cross-origin resource sharing (CORS). See also <https://www.w3.org/TR/cors/>.
19. **HTTP Strict Transport Security (HSTS).** The client application ensures secure access by using HTTP Strict Transport Security (HSTS) according to RFC 6797.
20. **Cross-application navigation.** The UWUM navigation bar has been successfully integrated into the client application.

Usage of scopes / screen name

When acting as WeGovNow UWUM client without data exchange with other WeGovNow applications, you will only need to request the "authentication" or the "identification" scope to be able to identify the user. These scopes allow to retrieve some user related information (i.e. numerical ID, identification string, screen name) to identify the user. When using the "authentication" or the "identification" scope, the response of the /api/1/token endpoint can optionally include an additional data structure providing member information. To request this optional "member" data structure, you need to set the parameter "include_member" to 1 or "true". Using the "identification" scope with the parameter "include_member" set to true, the response to the /api/1/token endpoint could look like as follows:

```
{
  "access_token": "UFYPzKrz7JHIKATI",
  "expires_in": 3600,
  "refresh_token": "5QM8OL7AbdabXusG",
  "token_type": "bearer",
  "member_id": 123,
  "member": {
    "id": 123,
    "name": "Johnny",
    "identification": "John Doe"
  }
}
```

The field "id" of the "member" object contains the static numerical ID of the user (equal to "member_id", i.e. redundant), the field "name" contains the screen name chosen by the user, the field "identification" contains the identification string set by the authority which identified the user as unique and eligible to use the WeGovNow application. In future, there may be more fields according to the upcoming specification of the /api/1/member endpoint of LiquidFeedback.

The parameter "include_member" can also be used at the /api/1/validate and the /api/1/info endpoints.

When acting as WeGovNow application using user related data or services of other WeGovNow applications, you will need to request the appropriate scopes from UWUM for the types of actions you want to perform with other WeGovNow applications (e.g. if you want to post new content to other applications, request the scope "post"; if you want to rate content in other applications request the scope "rate"; ...)

When acting as WeGovNow resource server (i.e. when offering user related data or services to other WeGovNow applications) you need to check (via the /api/1/validate endpoint) the scopes of the access token you received from the requesting application (e.g. if another application tries to post content for a user, check for scope "post"; if another application tries to rate content, check for the scope "rate").

Handling of updated user related data / user's email addresses

When a WeGovNow application wants to send notification emails to users, it is not adequate to retrieve the email address only once from UWUM as the notification email can be changed by the user at any time. Such a change needs to be reflected by all applications using this email address. Therefore, an application needs to retrieve the current notification email address **directly** before using it, in fact again before every usage.

For that purpose, the newly introduced API endpoint GET /api/1/notify_email can be used (using an access token with the "notify_email" scope). To be able to retrieve the email address while the user is not currently logged in, it will be necessary to request the "notify_email_detached" scope when identifying the user and to store the received refresh token permanently. The suffix "_detached" requests a scope for detached usage, i.e. for usage even after the user logs out. Please note, when exchanging a refresh token for an access token after the user has been logged out, you must explicitly request the "*_detached" scope(s) you need, e.g. "notify_email_detached" using the scope parameter of the /api/1/token endpoint.

Similar situations can occur related to other member properties stored in one application but used in another one, e.g. the screen name. But these seem not to be as critical as to avoid using an outdated email address. Such properties could be cached for a limited time before retrieving them again from the application storing this property.

Sustainability, unregistered third-party clients and the future

Following these rules, even a complete new (non-registered, third-party) application can easily make use of the WeGovNow infrastructure. The application can request certain scopes from UWUM (which can be granted or declined by the user) and use the appropriate services of all other WeGovNow applications. Using the upcoming application and service discovery, this is also possible vice versa.

Scopes vs. User Privileges

NB: The scope does NOT grant a privilege to a user, it just means an application can trigger an action within the scope *if* the user is authorized to perform the action. Example voting: an application needs the scope "vote" to cast a vote on behalf of the user but casting a vote will only work if the user has the necessary voting privileges. You can think of this as a matrix of scopes and user privileges or (alternatively) as a logical AND conjunction. Scopes control that an application does not misuse user privileges: while the trusted WeGovNow applications can request certain scopes without user interaction, a non-trusted third-party application/client would trigger a request for a confirmation by the user "Do you want to allow application X to cast votes on your behalf? [yes, one time / yes, permanently]" (compare to permissions for third party Twitter/Facebook clients and Android permissions).

Annex 3

LiquidFeedback work report

Second work report on extending the LiquidFeedback Core

Jan Behrens, Andreas Nitsche, Björn Swierczek

April 19, 2017

© 2017 FlexiGuided GmbH, Berlin

1 Overview on tasks

1.1 Open tasks listed in work report from June 6, 2016

The first work report on extending the LiquidFeedback Core (dated June 6, 2016)¹ listed the following three remaining tasks:

1. Adjustment of the background counting program (“lf_update”) to reflect the implemented changes for the new issue admission process.
2. Improvements to geospatial indexing and radial searches.
3. A complete dependency and license review of all WeGovNow software components.

The first task of adjusting the background counting program “lf_update” has been completed (see section 3). As for the second task, the geospatial indexing method has been completely revised with publication of pgLatLon (see “Work report on ‘pgLatLon’, an alternative to PostGIS”² from August 19, 2016). pgLatLon’s geospatial index also contains support for radial searches using PostgreSQL’s GiST interface. These and other features are also documented in pgLatLon’s online documentation, which can be downloaded at its project page³. The results of the third task (dependency and license review) have already been documented in Deliverable D1.2 (“Consolidated conceptual & methodological framework v2”) and will not be part of this work report.

¹included as Appendix 3.5 to Deliverable D3.1, pages 171 through 190

²included as Appendix 7.1 to Deliverable D3.1, pages 212 through 227

³<http://www.public-software-group.org/pgLatLon>

1.2 Additional tasks

1.2.1 Further adjustments to LiquidFeedback's issue admission mechanism

At the time of writing the previous work report (June 6, 2016), LiquidFeedback's draft for an issue admission mechanism not depending on "subject area membership"⁴ was based on admitting the most supported issue within a given time frame. Apart from difficulties with visualization, further considerations revealed that such an approach lacks the ability to respond to dynamic changes in the activity of the participants or their demand to discuss several issues at once while no issues are being discussed at other times.

Nonetheless, LiquidFeedback's previous issue admission system is unsuitable for integration with WeGovNow (due to the "subject area memberships" that are infeasible in the context of WeGovNow). Therefore, the new issue admission system has been modified further, including necessary changes to the LiquidFeedback Core. The final issue admission system is described in section 2 of this work report.

1.2.2 Adding necessary data structures for the Unified WeGovNow User Management (UWUM)

The necessary data structures for UWUM had to be added to the LiquidFeedback Core. The main work on the Core has already been done by end of August 2016, when the UWUM work report⁵ was published. Section 5 below will give a list of all UWUM related modifications to the LiquidFeedback Core.

1.2.3 Event logging

As part of LiquidFeedback's API implementation, data structures and triggers had to be created in the LiquidFeedback Core SQL schema. A list of all logged events is given in section 4.

1.2.4 Providing update capabilities for LiquidFeedback 3

LiquidFeedback provides a smooth upgrade path from version 1.0.0 through the most recent published version 3.2.2. Automatic SQL scripts convert the database of one LiquidFeedback version to the next higher version. This enables users of

⁴See section 2.1 of the first work report on extending the LiquidFeedback Core.

⁵included as Appendix 2.1.2 to Deliverable D3.1, pages 124 through 151

LiquidFeedback (e.g. municipalities or organizations using the software) to benefit from future updates. In order to keep this upgrade path intact, all modifications to the LiquidFeedback Core database schema were condensed into an upgrade file (“update script”) that may be executed on a LiquidFeedback v3.2.2 database, which then gets automatically updated to incorporate all changes made for WeGovNow. The work done to create the update script is discussed in section 6.

2 Modifications to LiquidFeedback’s issue admission system

Until LiquidFeedback version 3.2.2, user generated issues required to pass a certain supporter quorum in order to be further discussed. The quorum (e.g. 10%) used a reference population that was determined through so-called “memberships” in subject areas.⁶ While this approach is limited in regard to scalability,⁷ it also does not integrate well with the architecture of WeGovNow and potentially other integrated systems developed in the future.

After considering different approaches, including the approaches described in [8] and [9], a simple yet powerful mechanism was created to control the admission of issues for the discussion phase in LiquidFeedback, which will be described in the following three subsections.

2.1 Issue quorum based on active participants

The first measure to modify the issue admission mechanism is to base the existing supporter quorum on a population that is not calculated based on the “member-

⁶See section 4.9 (pages 71 and 72) in “The Principles of LiquidFeedback”, ISBN 978–3–00–044795–2.

⁷A minority exceeding the supporter quorum may create a flood of issues, see “The Evolution of Proportional Representation in LiquidFeedback” in “The Liquid Democracy Journal on electronic participation, collective moderation, and voting systems”, Issue 1. ISSN 2198–9532. http://www.liquid-democracy-journal.org/issue/1/The_Liquid_Democracy_Journal-Issue001-04-The_evolution_of_proportional_representation_in_LiquidFeedback.html

⁸“A Finite Discourse Space for an Infinite Number Of Participants” in “The Liquid Democracy Journal on electronic participation, collective moderation, and voting systems”, Issue 4. ISSN 2198–9532. http://www.liquid-democracy-journal.org/issue/4/The_Liquid_Democracy_Journal-Issue004-02-A_Finite_Discourse_Space_for_an_Infinite_Number_of_Participants.html

⁹“LiquidFeedback 2.x discontinued, LiquidFeedback 3.1 released, LiquidFeedback 4 development started”, December 14, 2015, on the LiquidFeedback developer mailinglist. <http://dev.liquidfeedback.org/pipermail/main/2015-December/000532.html>

ship” in subject areas (as described in [6]), but rather on *all* active participants¹⁰. This avoids the need of users to decide (and declare through LiquidFeedback’s user interface) in which subject areas they plan to engage in.

Naturally, using the total count of active participants as reference for the supporter quorum results in a higher barrier for new initiatives. Therefore, the issue quorum should be set to a very low value (or may even be entirely disabled) to also allow small minorities to engage in the proposition development and decision making process of LiquidFeedback. At the same time, additional measures are required to ensure managability for the participants when also possibly very small minorities are able to influence the agenda. These measures are described in the next subsection 2.2.

2.2 Issue limiter

The newly developed LiquidFeedback issue limiter ensures that the number of open issues¹¹ in a subject area does not grow unboundedly by using the number of open issues as a feedback for dynamically adjusting the supporter count required for admission of a new issue. For considerations on democratic fairness of such an approach, especially in regard to protection of minorities, refer to [8].

The basic principle is that increasing the number of open issues by a given absolute count increases the required supporter count by a certain (constant) factor. In turn, issues that are closed (e.g. because of finally having been voted upon) reduce the required supporter count by the same factor. This results in an exponential correlation between the number of open issues and the currently required supporter count to let a new issue pass to discussion phase.

Using S to denote the required supporter count, B_0 to denote the desired supporter count when no issues are open, and n as the number of open issues, the relation can (simplified) be described as follows:

$$S = B_0 \cdot f_1^n$$

with $f_1 \in \mathbb{R}^+$ being a configurable factor. In order to simplify configuration, the formula can also be expressed as:

$$S = B_N \cdot f_N^{\frac{n}{N}-1}$$

with an arbitrary $N \in \mathbb{N}$, $f_N = f_1^N$, and $B_N = B_0 \cdot f_N$. In this case, S is the actual required supporter count, B_N is the required supporter count if N

¹⁰This includes all participants which (a) are eligible to vote in a given unit and (b) have logged in recently, i.e. within a configurable time in the recent past.

¹¹In this context, we use the term “open issues” to refer to all issues that have been accepted for discussion but not been closed yet, i.e. all issues that are in discussion, verification, or voting phase.

issues were open, and f_N is a factor (or divisor) by which the supporter count is modified if N more (or less, respectively) issues are open.

The variables B_N , f_N , and N correspond to the following new columns of the area (subject area) table in the LiquidFeedback Core SQL schema¹²:

B_N	quorum_standard
f_N	quorum_factor
N	quorum_issues

The described approach doesn't yet take into account that different issues may have different runtimes. Counterintuitively, open issues that have a shorter runtime should be weighted more (i.e. a shorter runtime of an open issue should increase the required supporter count) because an equilibrium of N open issues that have a short runtime require more interactions of the participants than N open issues with a longer runtime.

Taking different runtimes into account, the number S of required supporters calculates as follows:

$$S = B_N \cdot f_N^{\frac{n^*}{N}-1}$$

with

$$n^* = \sum_i \left(\frac{d_i}{D} \right)^{-a}$$

where d_i is the total runtime of an issue i after admission for discussion phase (i.e. discussion time + verification time + voting time), D is a reference runtime (e.g. runtime of a default policy), and $a \in [0, 1]$ is an exponent selecting how much the runtimes of different issues are taken into account.

While the exponential relationship between open issues and the required supporter count doesn't need to take the total number of active participants into account, the implementation of the issue limiter still allows to use a relative base quorum (Q_N) instead of an absolute number of required supporters (B_N). In this case, $B_N = Q_N \cdot M$, where M is the total number of active¹⁰ participants.

LiquidFeedback's SQL schema¹² names all the configurable variables as follows:

B_N	quorum_standard (if quorum_den is NULL)
f_N	quorum_factor
N	quorum_issues
D	quorum_time
a	quorum_exponent
Q_N	quorum_standard divided by quorum_den

¹²Refer to the definition of the area table in the LiquidFeedback Core SQL schema file: http://www.public-software-group.org/mercurial/liquid_feedback_core/file/5855ff9e5c8f/core.sql#l636

Refer to the LiquidFeedback Core SQL schema file (particularly the definition of the “area_quorum” view on lines 3164 through 3200)¹³ for the actual implementation.

2.3 Resulting quorum

An issue passing from admission to discussion phase must fulfill both the demands stated in subsection 2.1 and subsection 2.2. Therefore, the required supporter count is always the maximum of the issue quorum (using the total number of active participants as a reference) and the supporter count determined by the issue limiter as described in subsection 2.2.

3 Vote and supporter counting

As already noted in subsection 1.1, LiquidFeedback’s background counting program (“lf_update”) had to be adjusted to reflect the implemented changes for the new issue admission process. In addition to work on this program, the snapshot mechanism¹⁴ needed further changes to be capable of documenting the set of active¹⁰ participants at a particular point in time (the number of active participants is required by the issue admission system described in section 2 of this work report).

4 Event logging

The LiquidFeedback Core has been extended to additionally log the following events:

- `suggestion_removed`: A suggestion has been deleted.
- `member_activated`: A participant has activated his or her account for the first time.
- `member_removed`: An account has been permanently disabled or removed.
- `member_active`: The “active” status of a participant changed. (A participant is marked inactive when he or she hasn’t log in for a configurable length of time.)

¹³http://www.public-software-group.org/mercurial/liquid_feedback_core/file/5855ff9e5c8f/core.sql#l3164

¹⁴See section 5.3 (pages 116 and 117) in “The Principles of LiquidFeedback”, ISBN 978-3-00-044795-2.

- `member_name_updated`: A participant changed his or her screen name.
- `member_profile_updated`: A participant updated his or her profile.
- `member_image_updated`: A participant updated his or her images.
- `interest`: A participant added or removed interest in an issue.
- `initiator`: A participant was added to or removed from the list of initiators of an initiative.
- `support`: A participant supported an initiative or removed his or her support.
- `support_updated`: A participant updated his or her supported revision to the most recent revision of an initiative.
- `suggestion_rated`: A suggestion has been rated by a participant.
- `delegation`: A delegation has been set or removed.
- `contact`: A contact has been published or unpublished.

These events complement the previously existing events:

- `issue_state_changed`: An issue passed from admission phase to discussion phase, from discussion phase to verification phase, from verification phase to voting phase, or got closed.
- `initiative_created_in_new_issue`: A new issue was created.
- `initiative_created_in_existing_issue`: A new initiative was created in an existing issue.
- `initiative_revoked`: An initiator revoked his or her initiative.
- `new_draft_created`: The initiative text has been updated.
- `suggestion_created`: A new suggestion for an initiative was added.

The above logging functionality has been implemented using triggers. Due to other existing triggers (including referential triggers), special care had to be taken in regard to the trigger firing order. PostgreSQL fires triggers in the following order:

1. BEFORE (regular trigger)
2. BEFORE (constraint trigger)
3. AFTER (constraint trigger)
4. AFTER (regular trigger)

Implementing the above event logging triggers as regular AFTER triggers solved conflicts with other triggers. However, additional checks had to be implemented in some event logging triggers to avoid event logging due to referential cascades; e.g. deletion of old issues should not cause logging of revoked delegations for those issues.¹⁵

5 UWUM

The following tables¹⁶ have been added to the LiquidFeedback Core SQL schema in order to store UWUM related data:

- `system_application`: to store OAuth 2.0 clients known to the operator of a LiquidFeedback/WeGovNow platform
- `system_application_redirect_uri`: to store additional (non-default) redirect URIs for the OAuth 2.0 flow
- `dynamic_application_scope`: to store dynamically registered OAuth 2.0 clients with their scope
- `member_application`: to store privileges granted by participants to particular OAuth 2.0 clients
- `token`: to store issued OAuth 2.0 tokens and authorization codes
- `token_scope`: to temporarily store additional sets of scopes for authorization codes when multiple different scopes were requested during the Authorization Code flow¹⁷

¹⁵For an example, refer to the `TG.OP='DELETE'` part of the `write_event_delegation()` trigger in the LiquidFeedback Core SQL schema file on lines 2372 through 2428, which requires that the corresponding organizational unit, subject area, or issue has not been deleted by the time of execution of the logging trigger (in which case the deletion was not a user action but the consequence of a referential constraint with `ON DELETE CASCADE` clause).

¹⁶See lines 356 through 483 of the LiquidFeedback Core SQL schema: http://www.public-software-group.org/mercurial/liquid_feedback_core/file/5855ff9e5c8f/core.sql#1356

¹⁷an extension to the OAuth 2.0 standard; refer to subsection 2.13 of the UWUM work report⁵

Additional triggers¹⁸ ensure that tokens are automatically invalidated and/or reduced in regard to their OAuth 2.0 scope when a participant logs out.¹⁹

Apart from the automatically generated PRIMARY KEY and UNIQUE indices,²⁰ additional indices were created where applicable to speed up database operations.

6 Update capabilities from LiquidFeedback 3

In order to enable users of previous LiquidFeedback versions to incorporate the implemented features, an update script has been created that converts a LiquidFeedback 3.2.2 database to the new format.

Because certain internal structures of LiquidFeedback were modified (such as the snapshot system or parts of the session management), the process of creating the update script had to be reviewed manually (with the help of automatic “diff” tools). The resulting update script²¹ has a size of more than 140 kilobytes and contains:

- 98 “ALTER TABLE” statements, consisting of
 - 29 “ALTER TABLE ... ADD COLUMN” statements,
 - 18 “ALTER TABLE ... DROP COLUMN” statements,
 - 1 “ALTER TABLE ... ALTER COLUMN” statement,
 - 30 “ALTER TABLE ... ADD CONSTRAINT”²² statements,
 - 14 “ALTER TABLE ... DROP CONSTRAINT” statements,
 - 4 “ALTER TABLE ... RENAME TO” statements,
 - 2 “ALTER TABLE DISABLE/ENABLE TRIGGER” statements,
- 15 “CREATE TABLE” statements,
- 8 “DROP TABLE” statements,

¹⁸See lines 2509 through 2584 of the LiquidFeedback Core SQL schema: http://www.public-software-group.org/mercurial/liquid_feedback_core/file/5855ff9e5c8f/core.sql#l2509

¹⁹See also subsection 2.9 on binding the lifetime of access and refresh tokens to a user’s web session by default in the UWUM work report⁵.

²⁰Note that PostgreSQL automatically creates indices for all PRIMARY KEY and UNIQUE constraints in a table, but not on columns covered by FOREIGN KEY constraints.

²¹http://www.public-software-group.org/mercurial/liquid_feedback_core/file/5855ff9e5c8f/update/core-update.v3.2.2-v4.0.0.sql

²²including ADD PRIMARY KEY, ADD FOREIGN KEY, and ADD UNIQUE statements

- 25 “CREATE INDEX” statements,
- 3 “DROP INDEX” statements,
- 22 “CREATE (OR REPLACE) VIEW” statements,
- 9 “DROP VIEW” statements,
- 4 “CREATE (OR REPLACE) RULE” statements,
- 35 “CREATE (OR REPLACE) FUNCTION” statements,
- and 145 “COMMENT ON” statements.

7 Publication of results

The developed changesets (including the update script as mentioned in section 6) have been submitted to the Public Software Group e. V on March 30, 2017 and were published by the Public Software Group in the source code repository of LiquidFeedback Core²³ under the terms of the MIT-License²⁴ on March 30, 2017.

© 2017 FlexiGuided GmbH, Berlin

²³http://www.public-software-group.org/mercurial/liquid_feedback_core/rev/5855ff9e5c8f

²⁴http://www.public-software-group.org/mercurial/liquid_feedback_core/file/d88fd3ae32d2/LICENSE

Annex 4

UWUM work report

Work report on Unified WeGovNow User Management (UWUM) development

Jan Behrens, Axel Kistner, Andreas Nitsche, Björn Swierczek

2016-12-12

© 2016 FlexiGuided GmbH, Berlin

1 Presentation of UWUM in Berlin

A first draft of UWUM has been presented in the kick-off meeting “Connecting The Bits” on April 14, 2016 in Berlin. The overall idea was to build a single-sign-on (SSO) solution on OAuth 2.0’s Authorization Code¹ flow.

For access tokens, the use of bearer tokens² was proposed. Furthermore, it was agreed on that TLS is to be used to secure all communication between UWUM and other components.

In addition to single-sign-on, UWUM’s capabilities were planned to include:

- a style endpoint, which allows applications to retrieve style information (e.g. a color scheme),
- a navigation endpoint, which allows applications to incorporate a common navigation bar into their user interfaces, and
- a service discovery endpoint, which allows applications to retrieve a list of other applications within the system and their capabilities/protocols.

This way, WeGovNow is designed to be a modular system that may be extended with different services which are all connected through UWUM.

It was agreed that UWUM will be implemented by LiquidFeedback such that it is possible to use synergetic effects between the necessary creation of an API for LiquidFeedback and the newly created features required by UWUM.

¹See <https://tools.ietf.org/html/rfc6749#section-1.3.1> for a short overview on the Authorization Code flow and <https://tools.ietf.org/html/rfc6749#section-4.1> for a detailed description.

²<https://tools.ietf.org/html/rfc6750>

2 Authentication and Authorization

For reasons of interoperability and security, we aimed to create an implementation that is fully compliant with RFC 6749.³ In this section, the extensions necessary in addition to that document will be explained below. All functionality has been implemented by the time of publishing this work report except where otherwise noted.

2.1 Roles

RFC 6749 defines several roles in subsection 1.1.⁴ The UWUM component as implemented by LiquidFeedback takes the role of the “authorization server”. Other WeGovNow components will take the role of “clients” but may also act as “resource server” for other components.

2.2 Choice of protocol flow

UWUM requires the Authorization Code flow¹ for secure user authentication, i.e. when used for single-sign-on (SSO). (Note that subsection 10.16 in RFC 6749 explains why the Implicit flow⁵ as defined by OAuth 2.0 is *not* suitable for secure user authentication.⁶)

The Implicit flow⁵ is still supported for clients which only require authorization but do not rely on secure user authentication (e.g. pure JavaScript clients which access other components but do not store themselves any resources which would need to be protected by SSO).

2.3 Types of clients

RFC 6749 distinguishes between “confidential clients” (which are capable of secure client authentication, e.g. by maintaining confidentiality of their client credentials) and “public clients” (which are incapable of secure client authentication). UWUM requires all clients which use OAuth 2.0’s Authorization Code¹ flow (and thus receive long-lasting refresh tokens) to be capable of secure authentication; i.e. every use of the token endpoint (see subsections 2.7 and 2.8) will require client authentication (except when an access token scope downgrade

³<https://tools.ietf.org/html/rfc6749>

⁴<https://tools.ietf.org/html/rfc6749#section-1.1>

⁵See <https://tools.ietf.org/html/rfc6749#section-1.3.2> for a short overview on the Implicit flow and <https://tools.ietf.org/html/rfc6749#section-4.2> for a detailed description.

⁶<https://tools.ietf.org/html/rfc6749#section-10.16>

is performed, see subsection 2.14). The use of “public clients” is only supported for those clients which utilize the Implicit⁵ flow because these clients will not handle any long-lasting tokens.

2.4 Client registration

Client registration is mentioned in section 2 of RFC 6749, even though the standard explicitly states that “the means through which the client registers with the authorization server are beyond the scope of [the] specification”.⁵⁶ UWUM provides two methods of client registration:

- registering clients through the municipality (or their technical administration) or an organization running a particular installation of WeGovNow,
- registration of any other (“dynamic”) client on a per-user basis by each user who wishes to use that client to access WeGovNow (machine accessibility).

These two registration methods are described in the following two subsections respectively.

2.4.1 Clients approved by the municipality

Clients approved by the municipality authenticate through TLS (X.509) certificates which are signed by the municipality or a certificate authority acting on their behalf. For example, the operator of the UWUM server could issue a certificate to the operator of each respective client. Furthermore, the operator of the UWUM server configures a list of automatically granted access scopes⁷ for the particular client (not every client has the same automatically granted access scopes, e.g. some clients might not require voting rights). Any other access scope may be granted on a per-user basis by the respective end-user or be disallowed by the municipality for a particular client (through white or black lists).

This results in the following information being stored per client:

- name of client,
- OAuth 2.0 client identifier (`client_id`),
- redirect URI(s)⁸,
- common name (CN) of the TLS certificate,

⁷<https://tools.ietf.org/html/rfc6749#section-3.3>

⁸See <https://tools.ietf.org/html/rfc6749#section-3.1.2> for redirection URIs. One redirect URI is the default redirect URI, other redirect URIs may be selected through the `redirect_uri` parameter, see: <https://tools.ietf.org/html/rfc6749#section-4.1.1>

- automatically granted scopes⁷,
- white list of scopes (optional),
- black list of scopes (optional, i.e. may be empty).

2.4.2 Dynamic clients

For the sake of machine accessibility, it would be nice to allow unregistered clients. Unfortunately, OAuth 2.0 requires some sort of client registration (at least) for the following security reasons:

- allowing capability to authenticate a client,⁹ in order
 - to avoid refresh token abuse by a third party in case of accidentally exposed refresh tokens,¹⁰
 - to avoid authorization code abuse (which could expose access and refresh tokens to a malicious 3rd party) in case of exposed authorization codes,¹¹
- restriction of choice of the redirect URI¹², in order
 - to avoid redirection URI manipulation,¹³
 - to avoid open redirector attacks.¹⁴

In order to be able to provide an open platform, however, it should still be possible to use clients which have not been explicitly approved by the operator of the WeGovNow platform. Assuming there will be more than one WeGovNow installation (e.g. run by different municipalities, each operating their own system), this is necessary in order to enable third parties to provide generic clients that can be used by *any* WeGovNow platform, even those not known to the operator of the client.

Consequently, registration of these clients should happen dynamically without further human interaction.¹⁵ This requires to automatically establish a channel

⁹See <https://tools.ietf.org/html/rfc6749#section-2.3> and <https://tools.ietf.org/html/rfc6749#section-10.1>

¹⁰<https://tools.ietf.org/html/rfc6749#section-10.4>

¹¹<https://tools.ietf.org/html/rfc6749#section-10.5>

¹²<https://tools.ietf.org/html/rfc6749#section-3.1.2>

¹³<https://tools.ietf.org/html/rfc6749#section-10.6>

¹⁴<https://tools.ietf.org/html/rfc6749#section-10.15>

¹⁵We assume that every user of WeGovNow is legally entitled to use any client of his or her choice to access his or her data and to perform actions. In cases where a particular operator of LiquidFeedback (e.g. a municipality) wants to decline this right, the use of dynamic clients could be disabled.

of trust between the client and the UWUM server through secure authentication. UWUM relies on the following mechanism to archive secure authentication of a dynamic client:

- a dynamic client is only referenced by its domain, and
- at the choice of each client, registration is performed either
 - by adding a certain entry to the domain's DNS zone¹⁶ or
 - temporarily through a REST API call to the UWUM server with a client-side TLS (X.509) certificate issued to the respective domain and signed by a publicly trusted certificate authority (e.g. "Let's Encrypt")¹⁷.

Taking into account that it cannot be outruled that TLS certificates could accidentally be exposed to a malicious 3rd party and considering that there might be at least one publicly trusted CA which is vulnerable to a state-level attack,¹⁸ we restrict the redirection URI¹² to the following static path on the web server's root level:

`/liquidfeedback_client_redirection_endpoint`

This repels any attempts of "authorization code redirection URI manipulation" as explaiend in subsection 10.6 of section 10 ("Security considerations") of RFC 6749 ("The OAuth 2.0 Authorization Framework")¹³ even in cases where dynamic client registration could be forged.

Any client that cannot follow the above redirection URI convention must be registered by the municipality or organization running a particular installation of WeGovNow (see subsection 2.4.1).

As an additional security mechanism, the dynamic registration is always done for a set of access token scopes⁷ to be used with a particular OAuth 2.0 flow. Thus a client's redirection endpoint registered for the Authorization Code flow cannot be used by the Implicit flow or vice versa unless the registration is broadened accordingly.

¹⁶A TXT DNS resource record needs to be added to the subdomain "_liquidfeedback_client" of the respective domain which must include a so-called magic string (namely "dynamic_client_v1") as first entry.

¹⁷The operator of LiquidFeedback is therefore required to decide on a list of trusted CA's. Many operating systems already ship with such a list of root certificates.

¹⁸Note that similar security considerations also apply to DNS and the risk of DNS cache poisoning or similar attack vectors. This could, however, be fixed by DNSSEC such that future versions of UWUM might lift the described restrictions for domains which are cryptographically secured.

The operator (e.g. a municipality) may still decide to disallow the use of non-approved (dynamic) clients completely. This would, however, limit machine accessibility and render the platform less open for extensions and unforeseen use cases. An appropriate configuration option will be provided which can also be used to limit the access token scope of dynamic clients (using a white or black list).

Unless dynamic clients are entirely disabled, an additional security warning will be displayed to the user when authorizing such a client. The user will be requested to verify that:

- the client domain is trustworthy,
- the client domain is used to host a legit application to access LiquidFeedback,
- the spelling of the domain name (whose client is going to be authorized) is correct,
- the granted scope of access (access token scope) is intended by the user.

Clients which want to avoid these warnings must be approved by the municipality or organization that is operating the LiquidFeedback system (see subsection 2.4.1).

2.5 Access token types

As previously mentioned, bearer tokens² as defined in RFC 6750 will be used as access tokens. Therefore, the access token type (`"token_type"`)¹⁹ returned by UWUM is always set to `"bearer"`.

2.6 Access token scopes

The following set of generic²⁰ access token scopes⁷ has been specified:

authentication: Authenticate the current user by reading its unique static ID and current screen name.

¹⁹<https://tools.ietf.org/html/rfc6749#section-7.1>

²⁰Application specific scopes could be introduced if they turn out to be necessary in the future. It would also be thinkable for dynamic clients acting as a resource server to provide a set of application specific scopes as part of their registration. Further security analysis would be required for such an extension. See also subsection 5.8 for considerations on generic versus application specific scopes.

identification: Identify the current user by reading its unique identification string. Automatically implies scope "authentication".

notify_email: Read the notification e-mail address of the current user.

read_contents: Read any user generated content (without authorship, ratings and votes).

read_authors: Read the author names of user generated content (author's static ID and screen name).

read_ratings: Read ratings (see scope "rate" below) by other users.

read_identities: Read the identities (identification strings) of other users.

read_profiles: Read the profiles of other users (e.g. phone number, self-description, etc).

post: Post new content.

rate: Rate user generated content (e.g. thumbs up/down, "+1", support an initiative, rate a suggestion).

vote: Finally vote for/against user generated content in a decision (e.g. vote on an issue in LiquidFeedback)

profile: Read profile data of current user (e.g. phone number, self-description, etc).

settings: Read current user's settings (e.g. notification settings, display contrast, etc).

update_name: Modify user's screen name.

update_notify_email: Modify user's notification e-mail address.

update_profile: Modify profile data (e.g. phone number, self-description, etc).

update_settings: Modify user settings (e.g. notification settings, display contrast, etc).

Note that any of these scopes can also be suffixed with "_detached" to request the scope for usage also when the user is not logged in (which will be explained in subsection 2.9).

2.7 User authentication (single-sign-on)

OAuth 2.0 by itself is not suitable for user authentication. Both the Authorization Code flow¹ and the Implicit flow⁵ can be extended to provide user authentication and thus allow to implement a single-sign-on (SSO) system. Because the Implicit flow would require additional security mechanisms to be implemented at client side (where bad implementations result in security vulnerabilities),⁶ UWUM extends the Authorization Code flow for the purpose of implementing an SSO solution as described in the following.

In order to protect against authorization code substitution attacks, the UWUM server checks the OAuth 2.0 client identity before accepting an authorization code.²¹ This is both a requirement stated in subsection 4.1.3 of RFC 6749 (“The OAuth 2.0 Authorization Framework”)²² and a recommended countermeasure to avoid authorization code substitution attacks in subsection 4.4.1.13 of RFC 6819 (“OAuth 2.0 Threat Model and Security Considerations”)²³.

The Access Token Response²⁴ of the OAuth 2.0 Authorization Code flow gets extended with the field “member_id” which returns the LiquidFeedback member ID of the signed-in user. OAuth 2.0 clients not aware of this extension are requested to ignore this field as stated in subsection 5.1 of RFC 6749.²⁵ Nonetheless, these clients may still pass the returned access token to the validate endpoint (see next section) in order to determine the member_id of the user who has logged in.

2.8 Endpoints

RFC 6749 defines two endpoint URLs at the authorization server side: the “authorization endpoint”²⁶ and the “token endpoint”²⁷. These are defined as follows:

- https://server_name/api/1/authorization (GET)
- https://server_name/api/1/token (POST)²⁸

Note that a base path may be appended to the *server_name* component if applicable.

²¹Note that, if the client is authenticating with the UWUM server, the *client_id* parameter can be omitted by the client when accessing the token endpoint (see next footnote).

²²<https://tools.ietf.org/html/rfc6749#section-4.1.3>

²³<https://tools.ietf.org/html/rfc6819#section-4.4.1.13>

²⁴<https://tools.ietf.org/html/rfc6749#section-4.1.4>

²⁵<https://tools.ietf.org/html/rfc6749#section-5.1>

²⁶<https://tools.ietf.org/html/rfc6749#section-3.1>

²⁷<https://tools.ietf.org/html/rfc6749#section-3.2>

²⁸The server name for the token endpoint may differ for those requests where TLS client certificates are used. See subsection 5.2 for explanation.

RFC 6749 does not specify any method for a resource server to “ensure that an access token presented to it by a given client was issued to that client by the authorization server”.²⁹ Therefore, an additional validation endpoint has to be specified:

- `https://server_name/api/1/validate` (POST)

The validation endpoint does not require any parameters except the access token (bearer token) to be passed using the mechanisms described in section 2 of RFC 6750.³⁰ It returns a JSON object with the following fields:

- `scope`: a space separated list of scopes⁷ associated with the access token (with any “_detached” suffix stripped off, see next subsection 2.9),
- `member_id`: an integer set to the id of the user who logged in,
- `logged_in`: a boolean set to false if the user has meanwhile logged out.

Note that the scope of an access token may change when the user logs out. This is explained in the following subsection 2.9. Subsection 2.12 will pick up the issue of user logout again.

There may be situations where an OAuth 2.0 client wants to check whether a user is currently logged in without actually forcing the user’s web browser to perform a login if no user was logged in. To provide this functionality, a 4th endpoint (also out of scope of the OAuth 2.0 specification) is added at the authorization server side:

- `https://server_name/api/1/session` (POST)

This “session” endpoint can be accessed directly by a user’s web browser (through a script performing a CORS³¹ HTTP request with credentials). Its usage is further explained in subsection 2.10.

2.9 Binding lifetime of access and refresh tokens to a users web session by default

Access tokens have an expiry time after which they will be invalidated.³² In addition to the maximum access token lifetime returned in the Access Token Response,²⁴ UWUM additionally limits the lifetime of both access tokens and

²⁹See section 10.3 of the RFC: <https://tools.ietf.org/html/rfc6749#section-10.3>

³⁰<https://tools.ietf.org/html/rfc6750#section-2>

³¹Cross-origin resource sharing, see <https://www.w3.org/TR/cors/>

³²<https://tools.ietf.org/html/rfc6749#section-5.1>

refresh tokens to the user's web session at the LiquidFeedback (UWUM) server by default (i.e. if the user logs out, the access tokens and refresh tokens will be immediately invalidated).

Some clients, however, require access longer than the user's login session. For this purpose, access token scopes⁷ with the suffix “_detached” may be requested (e.g. “vote_detached” instead of “vote”). Whether an application may request these scopes (as well as which scopes may be requested for detached access) depends on the configuration for the particular client, or – in case of dynamic clients – on the configuration for all dynamic clients. An access or refresh token that contains only detached scopes will not be invalidated on user logout. Access tokens, however, will still be invalidated when their expiry time (as denoted by the “expires_in” field in the Access Token Response²⁴) has elapsed, in which case a refresh token must be used to obtain a new access token. Access and refresh tokens which contain both detached and non-detached scopes will only have their non-detached scopes removed on user logout instead of being invalidated completely.

Other than the behavior described above, the “_detached” scopes behave as any other scope for the authorization²⁶ and token²⁷ endpoint. Only the validation endpoint (“api/1/validate”) will strip the suffix “_detached” from the scope field in its response because it doesn't matter for a validating resource server whether a scope has been granted detached from a web session or not.³³

Even if the token lifetime is bound to the web session (i.e. when only non-detached scopes are requested), a user's logged in web browser may still automatically re-authorize the client whenever he or she is logged in at UWUM and visits the client's website. If such a client was authorized by the user, the permission can be revoked by the user at any time using a designated configuration dialog provided by the UWUM server.

2.10 Checking user login without triggering a login

An interactive UWUM client application may want to determine whether a user is logged in without actually triggering a login. OAuth 2.0 does not provide such a mechanism on its own.³⁴ UWUM therefore provides an additional “session” endpoint (https://server_name/api/1/session, see subsection 2.8) to allow

³³Neither RFC 6749 nor RFC 6750 are violated because the authorization and token endpoint treat detached scopes like any other scope and a validation endpoint is not covered by these RFCs.

³⁴Also, extending the authorization endpoint by accepting a “prompt” parameter as done by OpenID Connect is not feasible for user-registered clients because non-logged-in users could be redirected to malicious clients registered by other users, making the system susceptible to open redirector phishing attacks. See subsection 5.5

web applications to gather information about the current login status of a user without actually triggering any (interactive) login or permission grant procedure. This endpoint is directly accessed by the user's web browser through an XMLHttpRequest (XHR) call while setting the "withCredentials" option of the XMLHttpRequest object to true.

The call does not need any parameters and should not have any additional request headers set³⁵. It returns a JSON object with the "member_id" attribute set to the ID of the current user (or to null if there is no logged-in user or if a user-registered client is not authorized to obtain the login status). Since the request is done by the user's web browser, the answer is *not* authoritative for the UWUM client and must only be used as a hint. **A returned user ID MUST still be confirmed via the regular OAuth 2.0 procedure using the authorization endpoint!** In this case, the authorization endpoint will not show a login window (because the user is already logged in).³⁶

2.11 Caching the login state

A successful user authentication could be cached in the session store of the UWUM client (usually at the web server side in conjunction with a cookie). This, however, can create confusion for the user because he or she might show up as being logged into the system after having logged out or vice versa. A possible solution is to use the "session" endpoint as discussed in the previous subsection 2.10 through a JavaScript which then notifies the server side of the UWUM client by redirecting the web browser if a reconfirmation of the user's login status is necessary.

In either case, UWUM clients should reconfirm that the user has not logged out at least immediately before any state changing request (e.g. posting, rating, voting, etc.) by using the validation endpoint (see subsection 2.8). This check cannot be done directly by the web browser due to security reasons (as also explained in the previous subsection 2.10).

³⁵Not setting additional request headers avoids CORS pre-flight requests, see <https://www.w3.org/TR/2014/REC-cors-20140116/#cross-origin-request-with-preflight-0>

³⁶There is a chance for a race-condition if the user simultaneously logs out. This could be solved by returning an authorization code through a CORS call. However, implementation of such a protocol is out of scope for WeGovNow and would require further security analysis.

2.12 Logout

2.12.1 Checking for logout

As explained in subsection 2.9, an access or refresh token is automatically invalidated on logout if only non-detached scopes have been requested. For all other cases, the “logged_in” boolean field returned by the validation endpoint (see subsection 2.8) may be used to detect a logout by the user.

The “session” endpoint (as further explained in subsection 2.10) may also be used to check whether a user might have logged out (without consuming much resources on the server-side of the UWUM client).³⁷ Note, however, that a request from the web browser to the session endpoint is not suitable for the UWUM client application to validate that a user is really logged in or to securely confirm that his or her session has really ended (see subsection 2.10).

2.12.2 Performing logout

Depending on design criteria, logout could be performed either

- through a direct link in the UWUM navigation bar or
- through a link in the UWUM navigation bar which leads to a user page where there is a second link for the actual logout procedure.

Technical implementation requirements differ for these two cases. In the first case, the logout is performed in the context of any UWUM client; while in the second case, the final logout link or button can be displayed in the context of a web page returned by the UWUM server (which is a different origin). Due to protection against cross-site-request-forgery (CSRF), an appropriate access token or dedicated logout token would need to be part of the link in the first case (the case of using a direct link for logout). In this case, an appropriate OAuth 2.0 access token scope would need to be added to avoid unwanted exposure of the logout token (or an access token with respective scope).

A decision on this issue has not been taken yet; user interface design considerations and technical security considerations should determine which of the discussed two approaches is more suitable. Also refer to subsection 5.10, which discusses certain design limitations due to privilege separation.

³⁷A future extension of UWUM could also allow UWUM clients (or their JavaScript components at the web browser side) to issue a request which is held open by the UWUM server for a set amount of time in order to allow pushing a change of the user’s login status just-in-time (see also subsection 5.4).

2.13 Requesting several access token scopes at once

To avoid unnecessary delays, a client may (as an extension to RFC 6749) request several access token scopes⁷ (i.e. sets of access ranges) at once by using the parameters “scope1”, “scope2”, etc. in the Authorization Request. The corresponding result parameter “access_token” will have “1”, “2”, etc. appended to its name (e.g. “access_token1” etc.). Note that counting must start with “1”. It is, however, allowed to include an optional non-numbered “scope” parameter in addition to “scope1”, “scope2”, etc. The result parameters “token_type” and “expires_in” are never numbered or duplicated due to size limitations in the Implicit flow (maximum URL length) but always relate to all returned access tokens.

The described behavior of this subsection is not part of OAuth 2.0. Using this extension is entirely optional for the client.

2.14 Downgrading access token scopes

As an extension to RFC 6749, the token endpoint has been extended in such a way that it can be used to downgrade access token scopes. This feature is important for meta-APIs because according to RFC 6749, the only way to obtain a new access token without the user’s web browser is to provide a refresh token to the token endpoint.³⁸ Refresh tokens, however, are bound to a particular client and must not be shared by the client with any other party but the authorization server.³⁹

A meta-API might receive an access token with a broader scope⁷ than the scope necessary for calls made by the meta-API provider to another resource server. Using a greater scope than necessary for calls to resource servers, however, weakens the overall security of the system. In order to allow meta-API providers to downgrade the scope prior to using the access token, the token endpoint²⁷ accepts the string “access_token” as value for the “grant_type” parameter, which will tell the UWUM server that an access token (and not an authorization code or refresh token) is being presented to receive a new access token with a downgraded scope. The access token has to be provided according to the rules stated in section 2 of RFC 6750,³⁰ and one or more scopes must be requested through the “scope”, “scope1”, etc. parameters (see subsection 2.13 for details on requesting several scopes at once). Client authentication is not required. The old access token with the broader scope will not be invalidated and may still be used in future requests (e.g. to receive another access token with a different scope).

³⁸<https://tools.ietf.org/html/rfc6749#section-6>

³⁹<https://tools.ietf.org/html/rfc6749#section-10.4>

For security reasons, downgrading an access token scope will never extend the token lifetime, i.e. the returned access token will have the same remaining maximum lifetime than the access token presented to the token endpoint.⁴⁰

2.15 Additional measures to prevent refresh token abuse

Conforming with section 10.4 of RFC 6749,⁴¹ the UWUM server (LiquidFeedback) ensures that refresh tokens are bound to the client they have been issued to. As also suggested in subsection 10.4 of RFC 6749, further means to restrict refresh token abuse are implemented. Refresh tokens are replaced periodically and using a refresh token invalidates the corresponding scope⁷ of all other previously issued refresh tokens, with the exception that refresh tokens which are still bound to a logged in user are unaffected.⁴² An additional grace period avoids problems due to race conditions or aborted connections. This approach is similar to the example given in subsection 10.4 of RFC 6749 while being resistant against accidental race-conditions or connection aborts and allowing for a more flexible usage (e.g. different subsystems of the same client may store different refresh tokens independently).

2.16 Required CORS support for resource servers

Because RFC 6750 requires bearer tokens² to be accepted through the HTTP header “Authorization”,⁴³ and because the “Authorization” header is not in the list of “simple response headers” as defined by the W3C recommendation on cross-origin resource sharing,⁴⁴ it is inevitable for all resource servers to support cross-origin resource sharing (CORS) with the respective “Access-Control-Allow-Headers” option⁴⁵ set to be able to fulfill the requirements of RFC 6750. Every UWUM component acting as a resource server should therefore enable and configure CORS accordingly. See <https://www.w3.org/TR/cors/> for details.

⁴⁰This is the reason why client authentication would not grant any extra security here and thusly can be omitted.

⁴¹<https://tools.ietf.org/html/rfc6749#section-10.4>

⁴²This is implemented by downgrading “_detached” scopes to their corresponding non-detached scopes.

⁴³<https://tools.ietf.org/html/rfc6750#section-2.1>

⁴⁴<https://www.w3.org/TR/cors/#terminology>

⁴⁵<https://www.w3.org/TR/cors/#access-control-allow-headers-response-header>

2.17 HSTS

We recommend to use HTTP Strict Transport Security (HSTS)⁴⁶ for all WeGovNow components to increase security.

3 Additional endpoints for integration

Beyond user authentication and authorization, three more API endpoints are being defined for backend and UI integration:

- a “navigation” endpoint to incorporate a navigation bar,
- a “style” endpoint to retrieve style information, and
- a “client” endpoint for application and service discovery.

Prototypes for the navigation and style endpoint have been implemented; the client endpoint for application and service discovery is currently only a stub.

3.1 Navigation endpoint

In order to integrate all WeGovNow applications in such a way that they look and feel like a single application, all WeGovNow applications share a common navigation bar. The “navigation” endpoint of the UWUM server returns this navigation bar to be included by each WeGovNow application. This way, modifications to the navigation bar can be made at a central place without the need to change every single application.

Either a login button or the user name with a link to a user page (where logout is possible) is included in the navigation bar, depending on whether an access token is provided when calling the endpoint.⁴⁷ For the login button, an alternative URL may be provided by the caller of the navigation endpoint. This login URL may either be the authorization endpoint of the UWUM server with an appropriate “state” HTTP GET parameter included (note that the value must be percent-encoded⁴⁸) or an URL provided by the UWUM client which initiates the OAuth 2.0 authorization and authentication procedure as described in section 2 of this document. Alternatively a unique placeholder (e.g. a GUID)

⁴⁶<https://tools.ietf.org/html/rfc6797>

⁴⁷Also a dynamic popup menu is thinkable. However, issues with JavaScript and privilege separation in case of animated submenus according to Material Design require further consideration. Refer to subsection 5.10 in that matter.

⁴⁸<https://tools.ietf.org/html/rfc3986#section-2.1>

can be passed as login URL to allow caching of the rendered navigation bar and replacing the login URL locally at the UWUM client.⁴⁹

Whether a structured JSON document or a pre-rendered HTML snippet is returned can be selected by another parameter passed to the navigation endpoint. A pre-rendered HTML snippet may be either returned encapsulated in a JSON response or raw for usage with the HTML5 include tag.

When the “client_id” parameter is provided to the navigation endpoint, the corresponding client tab gets highlighted (or marked as active in case of the structured JSON document response).

It is planned to collapse the navigation bar on small screens. This feature might interfere with application specific menus; refer to subsection 5.12 for that matter.

3.2 Style endpoint

The style endpoint provides basic color definitions for a primary and an accent color as 8-bit RGB triplet to be able to customize the unified visual look of all WeGovNow applications for a particular installation by central configuration. Additional colors can be derived from these two base colors. If the UWUM server gets configured with colors from the Material Design color palette, the corresponding Material Design color name of the primary and the accent color is also provided.

3.3 Endpoint for application and service discovery

The endpoint “client” is supposed to return a list of all system applications and, if an access token is provided, a list of all registered dynamic clients for the corresponding user. Implementation of this endpoint will require storing the base URL of all system applications at the UWUM server.

Further discussion with OntoMap is required for specification and implementation of this endpoint.

4 Test platform

A test platform has been created in mid September to start integration with the other consortium partners.

⁴⁹Note that the characters “<”, “>”, “&”, as well as the quotation mark character should be avoided in a placeholder string because these characters would get HTML entity encoded as described in subsection 8.1.4 of the HTML5 standard, see: <https://www.w3.org/TR/html5/syntax.html#character-references>

4.1 Benchmarks

The following benchmarks for integration have been defined, whose fulfillments have been published in the weekly status reports.

Client URLs established A client application (resource server and/or relying party) has been installed and its base URL and redirection endpoint⁵⁰ has been communicated to the consortium.

SSL key and certificate for end-users A private key and a publicly trusted SSL certificate has been created for the end-user web interface and SSL connections to that interface have been successfully tested.

Certificate signing request (CSR) for UWUM API A private key for accessing the UWUM API and a corresponding certificate signing request (CSR) has been created and submitted to FlexiGuided GmbH (LiquidFeedback).

SSL certificate for UWUM API A signed certificate for the UWUM API client key has been sent back to the consortium member and their client application has successfully established a secured connection with the UWUM server.

Authorization endpoint accessed The client application can redirect an end-user to the UWUM authorization endpoint.⁵¹

Authorization endpoint error response handling The client application is capable of receiving authorization errors⁵² through its redirection endpoint⁵⁰ and displaying it to the end-user.

Access token request (including end-user identification) The client application has successfully received an authorization code and identified the end-user through an access token request.⁵³

Access token request error handling The client application is capable of properly processing errors during the access token request.⁵⁴

Using access tokens for API calls to other components The client application has successfully used an access token to perform a LiquidFeedback API call.

⁵⁰<https://tools.ietf.org/html/rfc6749#section-3.1.2>

⁵¹<https://tools.ietf.org/html/rfc6749#section-4.1.1>

⁵²<https://tools.ietf.org/html/rfc6749#section-4.1.2.1>

⁵³<https://tools.ietf.org/html/rfc6749#section-4.1.3>

⁵⁴<https://tools.ietf.org/html/rfc6749#section-5.2>

Access token verification The client application is capable of verifying the validity and scope of an access token.

Accepting access tokens from other components The client application provides at least one API call where an access token is used for authorization.

Accepting access tokens as “Authorization” header In conformance with RFC 6750 (Bearer Token Usage), the client application (resource server) accepts access tokens through the authorization request header field.⁵⁵

Cross-origin resource sharing The client application allows cross-origin resource sharing (CORS) as described in subsection 2.16 of this document.

Cross-application navigation The UWUM navigation bar has been successfully integrated into the client application.

IPv6 IPv6 capabilities have been tested.

5 Technical challenges

In this section, we will describe obstacles encountered during implementation and during integration with the consortium partners as well as respective solutions.

5.1 Third party clients (non-registered clients vs. dynamic registration)

OAuth 2.0 demands client registration but does not specify how such client registration is to be implemented.

“Before initiating the protocol, the client registers with the authorization server. The means through which the client registers with the authorization server are beyond the scope of this specification but typically involve end-user interaction with an HTML registration form.”⁵⁶

Manual client registration, however, is only suitable for a service-centered approach where a software provides only a single service (e.g. Facebook, Google, Twitter, etc). An open source solution, however, could be installed at several sites by different service providers. It is therefore not sufficient to register a client

⁵⁵<https://tools.ietf.org/html/rfc6750#section-2.1>

⁵⁶Ed. D. Hardt: The OAuth 2.0 Authorization Framework, October 2012. Section 2 (Client Registration), <https://tools.ietf.org/html/rfc6749#section-2>

at a single service provider if this client shall be usable for any service provider using the UWUM server software.

One possible solution would be the creation of a central (i.e. world-wide) UWUM client registry. Such central client registry, however, could be a single point of failure and would empower a central authority to control usage of the UWUM protocol (e.g. it would be possible to block certain clients). We consider this approach contrary to the concepts of open source and open data.

Therefore, we implemented a dynamic client registration protocol that keeps implementational complexity at a minimum while providing good security properties which outperforms many other solutions for client registration due to requiring direct access to the DNS zone of the domain (for adding a TXT record) or credentials (a publicly trusted TLS certificate with corresponding key) that should be accessible only by the domain owner. Dynamic client registration is described in subsection 2.4.2 of this document.

5.2 TLS client side certificates and web browser behavior

Web server software often offers three different settings for handling TLS client certificates:

- client-side certificates disabled,
- optional client-side certificate,
- mandatory client-side certificate.

Often these settings can be made only on a per-domain basis (i.e. for each virtual host). Furthermore, enabling client-side certificates (even if set to “optional”) will cause web browsers to show up a dialoge when accessing pages on that domain.

For these reasons, a separate hostname has to be used for API endpoints when a TLS client-side certificate is to be provided (which affects the token endpoint²⁷). The UWUM server will have to provide a configuration endpoint where dynamic clients may retrieve a deviant domain for the token endpoint; and dynamic UWUM clients (see subsection 2.4.2) will have to query this configuration endpoint prior to using the token endpoint).

5.3 Multi-domain certificates

TLS certificates may be issued for more than one domain using the “Subject Alternative Name” (SAN) extension. The current implementation of Liquid-Feedback, however, relies on an HTTP reverse proxy to include the distinguished

name (DN)⁵⁷ of the certificate in a designated HTTP header. Some reverse proxy software, namely “NGINX” which is recommended for use with LiquidFeedback, does not properly support transmitting a domain list from the SAN extension. In case of “NGINX”, header line folding⁵⁸ is used to pass multiple domain names from a TLS certificate to the respective backend (e.g. LiquidFeedback). Header line folding, however, has recently been deprecated by RFC 7230,⁵⁹ and it is not supported by LiquidFeedback (and not even by “NGINX” for incoming requests). The problem of header line folding in the context of multi-domain TLS certificates has also been discussed in the “NGINX” issue tracker under ticket #857.⁶⁰ The issue is currently not classified as bug⁶¹ and it is unclear when a patch will be incorporated into the software.

For the technical difficulties explained above, we refrained from supporting multi-domain certificates at this stage. In case of UWUM clients approved by the municipality or operator of LiquidFeedback, this shouldn’t be a problem anyway because the certificate authority will be under the control of the operator, such that it is easy to create a certificate using the DN/CN property. For dynamically registered clients, an alternative mechanism using DNS TXT records is available (see subsection 2.4.2).

If multi-domain certificates are supported in the future, it is vital that the token endpoint requires the “client_id” parameter to be set for all clients authenticating with such a multi-domain certificate. This way, code substitution attacks²³ can be repelled. (Note that RFC 6749 requires the “client_id” parameter to be set only if the client is not authenticating with the authorization server,²² but this does not work for multi-domain certificates.)

5.4 Outdated logins

While a successful OAuth 2.0 authorization procedure (using the Authorization Code flow¹) can be used to confirm that a user is logged in at the particular time of the Access Token Response²⁴, an UWUM client obviously can’t assume that the login will be still valid at any later time.

UWUM currently provides two methods to check if a user has logged out; these are explained in subsection 2.12.1 of this work report. Considerations in regard to purposeful caching of the user’s login status are found in subsection 2.11.

Even if no caching of the login status is performed, there is still the possibility that a user opens WeGovNow with two different browser windows or browser tabs.

⁵⁷The DN contains a single domain as CN (common name).

⁵⁸<https://tools.ietf.org/html/rfc2616#section-2.2>

⁵⁹<https://tools.ietf.org/html/rfc7230#appendix-A.2>

⁶⁰<https://trac.nginx.org/nginx/ticket/857>

⁶¹<https://trac.nginx.org/nginx/ticket/857#comment:2>

He or she might then log out in one window and afterwards switch to the other window where the logout has not been noticed yet, which creates confusion for the user. A possible solution is to regularly check if the user has logged out by utilizing the cross-origin-resource-sharing (CORS) XML-HttpRequest (XHR) as explained in subsection 2.10.

Regular requests to detect logouts, however, cause unnecessary resource consumption for all involved components. A better approach would be to have a permanent TCP connection between the web browser and the UWUM server (or alternatively between the UWUM client and the UWUM server if at the same time there is a permanent connection between the web browser and the UWUM client). There are different technologies thinkable for this approach. One method is to keep an XML-HttpRequest open for a set amount of time during which the server is capable of sending a message directly to the web browser. (The request has to be repeated after timeout or after a message has been received, whichever happens first.) Another technique would be to use WebSockets. None of these additional techniques have been implemented yet.

5.5 Susceptibility to open redirector phishing attacks when allowing login checks through web browser redirection

Subsection 2.10 mentioned that an interactive UWUM client application may want to determine whether a user is logged in without actually triggering a login. OAuth 2.0 does not provide such a mechanism on its own, and our research concluded that any form of redirection-based mechanism for providing this functionality⁶² would be susceptible to open redirector phishing attacks as described in subsection 4.2.4 of RFC 6819 (“OAuth 2.0 Threat Model and Security Considerations”)⁶³ as long as third parties are capable of registering a malicious client with a corresponding redirection URI¹² that is under the control of the third party.

The previously mentioned subsection of the threat model and security considerations document (RFC 6819) suggests client registration with redirect URI registration (and avoiding redirects to any non-registered redirect URI)⁶⁴ as only countermeasure for this threat. However, this countermeasure only works when manual client registration (and manual approval through the operator of the UWUM server) is mandatory. It particularly fails if dynamic client registration (e.g. as described in subsections 2.4.2 and 5.1 of this work report) is allowed.

⁶²e.g. accepting a “prompt” parameter as done by OpenID Connect, see http://openid.net/specs/openid-connect-core-1_0.html#AuthRequest

⁶³<https://tools.ietf.org/html/rfc6819#section-4.2.4>

⁶⁴<https://tools.ietf.org/html/rfc6819#section-5.2.3.5>

Luckily, the technique of cross-origin resource sharing (CORS) allowed for the development of an alternative to the redirect-based approach. Subsection 2.10 explains the mechanism.

5.6 Handling of updated user related data (e.g. user's e-mail addresses)

When a WeGovNow application wants to send notification e-mails to users, it is not adequate to retrieve the e-mail address only once from UWUM as the notification e-mail can be changed by the user at any time. Such a change needs to be reflected by all applications using this e-mail address. Therefore an application needs to retrieve the current notification e-mail address *directly* before using it, in fact again before every usage.

For that purpose, another API endpoint `/api/1/notify_email` (GET) can be used (using an access token with the "notify_email" scope). To be able to retrieve the e-mail address while the user is not currently logged in, it will be necessary to request the "notify_email_detached" scope when identifying the user and to store the received refresh token permanently. The suffix "_detached" requests a scope for detached usage, i.e. for usage even after the user logs out.⁶⁵

Similar situations can occur related to other member properties stored in one application but used in another one, e.g. the screen name. But these seem not to be as critical as to avoid using an outdated e-mail address. Such properties could be cached for a limited time before retrieving them again from the application storing this property.

5.7 Race conditions with refresh token rotation

As suggested in subsection 10.4 of RFC 6749,⁴¹ refresh token rotation is employed to provide better security properties (e.g. in case of exposed refresh tokens and client certificates, or in case of the existence of a single compromised certificate authority which would render authentication of dynamic clients insecure).

Unfortunately, RFC 6749 does not specify how old refresh tokens are invalidated. Section 6 of RFC 6749 only says that⁶⁶

- the authorization server MAY issue a new refresh token, in which case
- the client MUST discard the old refresh token and replace it with the new refresh token, and

⁶⁵Note that when exchanging a refresh token for an access token after the user has been logged out, an UWUM client must also explicitly request the "*_detached" scope(s) it needs, e.g. "notify_email_detached" using the scope parameter of the `/api/1/token` endpoint.

⁶⁶<https://tools.ietf.org/html/rfc6749#section-6>

- the authorization server MAY revoke the old refresh token.

Always revoking the old refresh token after transmission can have a bad effect on system stability, considering that responses might be interrupted. Furthermore, multiple backends of an UWUM client could simultaneously access the token endpoint. Such legit accesses by two legit backends of the same client would need to be distinguished from accesses by a legit client and a malicious third party who obtained a copy of a refresh token.

Subsection 2.15 explains the mechanisms employed by the UWUM server to mitigate the risk of refresh token abuse while solving the problems stated above.

5.8 Creating a set of suitable access token scopes

A useful set of access token scopes⁷ is a vital aspect of privilege separation. From a security point of view, scopes should be as fine-graded as possible, particularly there should be different scopes for different applications (e.g. an application that wishes to rate user contributions in application X does not need an access token that allows to rate user contributions in application Y). Extensibility, on the other hand, would be complicated if access token scopes always refer to a single application (i.e. a single resource server in this context). Furthermore, it is a goal that the WeGovNow platform looks and feels like a single integrated application. When users grant access scopes to third party clients, such application-based scopes would be difficult to understand for the user, which by itself can have bad influence on the overall system security.

We therefore decided to provide a set of generic access token scopes as listed in subsection 2.6. For future extensions, see footnote 20.

5.9 Misconceptions regarding scopes vs. user privileges

Scopes must not be mistaken for user privileges. I.e. a scope does *not* grant a privilege to a user; it just means an application can trigger an action within the scope *if* the user is authorized to perform the action. For example, an application needs the scope “vote” to cast a vote on behalf of the user but casting a vote will only work if the user has the necessary voting privileges.

Programmers of UWUM clients must keep these differences in mind and execute an action only if both the scope and the users privileges are sufficient for the respective action.

5.10 JavaScript integration and privilege separation

Dynamically sharing JavaScript code between UWUM clients or between the UWUM server and an UWUM client violates privilege separation because it would enable one component to execute code in the security context of another origin. For example, one application 'A' could send a harmful JavaScript to be included in a web page returned by another application 'B' which then discloses the session cookie for application 'B' to application 'A'.

For this reason, the common navigation bar as returned by the navigation endpoint (see subsection 3.1) currently does not include any JavaScript code. UWUM clients may therefore even consider to sanitize the returned HTML code in such a way that any JavaScript is removed or rejected.

Interface design decisions, however, might suggest to use JavaScript for the navigation bar. Material design, for example, requires popup-menus to be animated, which cannot be done with CSS alone. Another reason for JavaScript might be dynamic modifications of the navigation bar (e.g. collapsing the navigation bar to a menu icon) depending on the screen size or the device of the user. Also other integration techniques might suggest the use of JavaScript.

An alternative to dynamically provided JavaScripts by the UWUM server would be a common library to be included locally by each WeGovNow component. Whenever this library is updated, administrators of each component can look over it before incorporating it. While this approach provides proper privilege separation, its downside would be the administrative overhead.

At least in regard to the navigation bar, it would eventually need to be decided whether

- there will be no JavaScript used by the navigation bar,
- the UWUM server will dynamically return JavaScript code for the navigation bar, or
- each WeGovNow component needs to include a pre-distributed JavaScript.

5.11 Logout through navigation bar

The common WeGovNow navigation bar (as returned by the "navigation" endpoint, see subsection 3.1) should also include a possibility to logout. Due to protection against cross-site-request-forgery (CSRF) and because the navigation bar will be included in responses from different web servers (different "origins"), a simple logout link does not work. Subsection 2.12.2 deals with different approaches to this problem.

5.12 Collapsing navigation bar and application menu

In case of mobile devices, it may be desirable to collapse the navigation bar to a single menu icon displayed in the corner of the screen. Despite the technical problems in regard to JavaScript (which are discussed in subsection 5.10), there is also a challenge in regard to a potentially existent second menu bar which is provided by the particular application currently selected.

It could be difficult for the user if two menu icons are being displayed (i.e. a meta-menu, which covers the entries of the navigation bar, and an application specific menu). A potential solution could be to combine both menus into a single one. In this case, however, the considerations of subsection 5.10 still apply.

5.13 UWUM clients without user interface

In addition to UWUM clients having a user interface, there are also WeGovNow applications thinkable which do not have any (end-)user interface. This includes both meta-API providers as well as other service components. In the context of WeGovNow, one meta-API provider could be OntoMap.

The current UWUM specification enables the development of meta-APIs because access tokens are not bound to a particular UWUM client and can be downgraded in regard to their access token scope (of which the latter is important for security, see subsection 2.14). Thus, a meta-API can simply require its callers to provide a valid access token which then can either be used directly or downgraded for further requests performed by the meta-API provider to other resource servers.

Nonetheless, the mechanisms described in this work report still require privileges that are bound to a particular user. For UWUM clients requiring access privileges that are not tied to a particular user (e.g. clients which aggregate data of all users and publish that information), the Client Credentials Grant⁶⁷ should be implemented.

5.14 Client authentication for resource servers

While UWUM enables (a) its clients to authenticate users and (b) resource servers to verify user authorization (both explained in section 2), it does not enable resource servers to authenticate clients. Such client authentication might be required by applications that want to establish a trusted channel to another application independently of user authorization.⁶⁸

⁶⁷<https://tools.ietf.org/html/rfc6749#section-4.4>

⁶⁸An example could be OntoMap logging actions executed at other applications (which are then reported to OntoMap by the respective application with client authentication enabled).

Unfortunately, neither OAuth 2.0 nor UWUM enable applications to verify the identity of another application. Even if OAuth 2.0 uses client authentication for a variety of reasons⁶⁹ for the authorization endpoint²⁶, it doesn't provide such an authentication method to other applications. Extending the OAuth 2.0 work flow in this matter (e.g. by returning the `client_id` when an access token is presented to the validation endpoint⁷⁰) would rise some issues:

- Tying an access token (a bearer token² in case of UWUM) to a particular client does not make sense in case of applications that behave both as an OAuth 2.0 resource server and as a client (e.g. meta-API providers or applications which provide an API and have to perform further API calls to complete a requested action). Also, client impersonation would be possible. To give an example: if a received access token is tied to a particular client A, and if application A uses this access token to perform an action at application B, then application B would be able to impersonate application A. Furthermore, application B couldn't use the access token to authenticate as application B when performing further requests at the API of another application C.
- Using a custom scope to identify the origin of a request (e.g. a scope "I_am_appX") would also enable client impersonation (e.g. any application who receives an access token with the scope "I_am_appX" could then impersonate application X). An alternative could be to use scopes that reflect better the particular action to be performed, e.g. a scope "write_appXs_log_at_appY". It is self-evident that this would increase the number of scopes drastically (possibly quadratically), which, in turn, might create a maintenance/configuration mess. Other than that, there is another problem with using scopes for client authentication: following RFC 6750, there can only be one bearer token per request.² If a client needs to use a received access token for an API call at another component, then this access token could not be used to authenticate that client because it won't have the necessary scope. One possible solution could be to allow adding scopes to an existing access token or extend RFC 6750 in such a way that multiple access tokens could be used per request. All those solutions, however, go far beyond OAuth 2.0 and would require extra implementation work for all consortium partners. In the end, the created solution wouldn't be OAuth 2.0 anymore.

⁶⁹See beginning of subsection 2.4.2 of this report.

⁷⁰See subsection 2.8 for an explanation of the validation endpoint.

The straight-forward way of authenticating clients is to use the existing mechanism already employed by all UWUM clients: TLS client-side certificates. This, however, requires TLS client certificate checking by each resource server that needs to authenticate other clients.

© 2016 FlexiGuided GmbH, Berlin

Annex 5

PgLatLon work report

Work report on “pgLatLon”, an alternative to PostGIS

Jan Behrens, Andreas Nitsche

2016-08-19

© 2016 FlexiGuided GmbH, Berlin

1 Previous work

As explained in our work report dated June 6, 2016, one created extension for the LiquidFeedback backend (“LiquidFeedback Core”) was the feature of geo-tagging user input as well as allowing geographic searches (geospatial indexing). A common solution for geospatial data processing with PostgreSQL is PostGIS (see <http://postgis.net/>). Using PostGIS to fulfill that task, however, was ruled out because of the following reasons:

- PostGIS would introduce a long chain of software dependencies.
- Many of PostGIS’ features were unnecessary for LiquidFeedback’s needs. At the same time, the number of functions that actually support the WGS-84 spheroid is very limited in PostGIS.
- Viral/incompatible licensing.

Therefore, a lightweight alternative for database indexing and distance measurement (based on a 2-dimensional Taylor series approximation¹ and PostgreSQL’s existing geometric data types²) was implemented. The associated results have been published on May 29, 2016 under the terms of the MIT-License.³

¹See page 11 of work report June 6, 2016.

²See pages 11 through 13 of work report June 6, 2016.

³See subsection 2.2.13 of work report June 6, 2016.

1.1 Limitations of the implementation as of June 6, 2016

1. While the implementation as of June 6, 2016 was sufficient for local geospatial data processing, it wasn't capable of performing correct calculations for large-area applications (e.g. spanning multiple continents).⁴
2. A particular side-effect of the Taylor-based approach were reversal-symmetry anomalies that could confuse the user of a system when comparing different views on the same dataset.⁵
3. Another drawback was the necessity to include a certain syntax in all geospatial queries (see lines 318 through 339, lines 524 through 544, and lines 606 through 625 in the patched `core.sql` file⁶).
4. Nearest neighbor searches could only be performed iteratively.⁷

As we will show in this document, all previous limitations could meanwhile be lifted by implementing a GiST-based index using fractal curves.

1.2 Ideas presented in the work report June 6, 2016

The work report of June 6, 2016 already contained an outlook in regard to possible improvements on page 20:

Keeping future applications in mind, it would be beneficial to improve the implemented methods for geospatial indexing and radial searches by providing an alternative formula for distance calculation on the WGS-84 ellipsoid that could serve as a compromise between the complexity of Vincenty's formulae and the simplicity of local Taylor approximation. Utilizing the nearest neighbor search capabilities of PostgreSQLs GiST indexing framework would be another improvement.

⁴Examples for the accuracy have been given in subsection 2.2.8 of work report June 6, 2016 and will be revisited in this work report, subsection 2.1.4

⁵See subsection 2.2.9 of work report June 6, 2016.

⁶http://www.public-software-group.org/mercurial/liquid_feedback_core/file/3e28fd842354/core.sql

⁷See subsection 2.2.11 of work report June 6, 2016.

2 Implemented improvements

2.1 A new formula for approximating distances between two points on the WGS-84 spheroid

The idea of developing a new algorithm for distance calculation has been pursued and successfully implemented.

2.1.1 The general idea

The general idea to quickly calculate distances on the WGS-84 spheroid is to calculate the exact coordinates of two points in a three dimensional (Euclidean) coordinate system first. Then, the exact tunnel distance can be calculated using the Pythagorean theorem. After the tunnel distance has been determined, a sphere model of earth (with radius $r = \frac{2a+b}{3}$) is used⁸ to transform a tunnel distance into a distance on the surface of earth.

$$s_{\text{surface}} = 2r \cdot \arcsin\left(\frac{s_{\text{tunnel}}}{2r}\right)$$

The error for small distances tends towards zero. (Note that for $s_{\text{tunnel}} = 0$, the derivative of the above function is 1, and therefore $s_{\text{surface}} \approx s_{\text{tunnel}}$.) For medium distances, the above formula serves as a good approximation for the surface distance on the WGS-84 spheroid. In case of antipodal points, however, this method becomes numerically unstable,⁹ which is why some modifications are necessary as explained in the following subsection 2.1.2.

2.1.2 Using antipodal points for huge distances

For nearly antipodal points, the distance between two points \mathfrak{A} and \mathfrak{B} can be approximated by calculating a third point $\overline{\mathfrak{B}}$ on earth by reflecting \mathfrak{B} through the earth’s center. Then the method described in subsection 2.1.1 can be used to calculate a distance $\overline{s}_{\text{surface}}$ between \mathfrak{A} and $\overline{\mathfrak{B}}$ (i.e. between \mathfrak{A} and the reflection of \mathfrak{B}). The approximate distance between the original points \mathfrak{A} and \mathfrak{B} then calculates as $s_{\text{surface}} = \pi r - \overline{s}_{\text{surface}}$.

Cross-fading is done for points that are neither close to another nor nearly antipodal. This ensures monotonic behavior of the algorithm. For details, we refer to the corresponding source code that has meanwhile been published by the Public Software Group e. V.

⁸With a and b being the semi-major and semi-minor axis of the WGS-84 reference spheroid.

⁹There is a numerical instability because the tunnel distance is almost constant for all nearly antipodal points on a sphere. Furthermore, the tunnel distance between two antipodal points on the spheroid varies and is not always $\leq 2r$.

2.1.3 Costs of trigonometric calculations

The algorithm described in subsections 2.1.1 and 2.1.2 requires 12 trigonometric operations, 5 square root operations, and up to six if-clauses¹⁰ (see source code). For points that are close to one another, some of these operations can be omitted.

Tests have shown that the previously used Taylor-based approach doesn't have any significant speed advantages when implementing both functions in the programming language “C”. The Taylor-based approach still has advantages when trigonometric functions are called in an interpreted language such as PostgreSQL's built-in PL/pgSQL language.¹¹ By switching to C, however, this advantage vanishes if the pre-calculated Taylor coefficients must be read from a PostgreSQL record-type data structure.¹²

The speed of the new distance calculation function is approximately the same as the old Taylor-based formula while the new formula has a higher accuracy and also works across continents and across the poles.

2.1.4 Comparing the accuracy with the truncated Taylor series

The work report from June 6, 2016 presented exemplary errors when calculating distances between the following locations on earth:

Location	Sym.	Latitude	Longitude
Berlin (Brandenburg Gate)	B	52.5162746 N	13.3777040 E
London (Big Ben)	L	51.5007292 N	0.1246254 W
Paris (Eiffel Tower)	P	48.8583701 N	2.2944813 W
Rome (Colosseum)	R	41.8902102 N	12.4922309 W
Moscow (Red Square, Mausoleum)	M	55.7537117 N	37.6198846 E
Longyearbyen (harbour)	X	78.2289157 N	15.5994530 W

The Taylor-based approach resulted in the following relative errors:

¹⁰Counting the fabs() function as if-clause.

¹¹Due to the overhead of function calls.

¹²The PostgreSQL source code contains a comment in backend/executor/execQual.c explaining that the C functions “GetAttributeByName” and “GetAttributeByNum” are slow due to a “typcache” lookup on each call.

Error in ‰	To B	To L	To P	To R	To M	To X
From B (Berlin)	0	+1.40	+1.41	+1.96	+4.42	−6.95
From L (London)	+1.39	0	+0.15	+0.08	+11.09	−3.97
From P (Paris)	+1.29	−0.21	0	+0.09	+10.72	−3.94
From R (Rome)	+0.63	−1.23	−0.66	0	+10.58	−3.03
From M (Moscow)	+4.48	+11.20	+11.15	+13.26	0	−2.01
From X (Longy.)	+1.82	+1.34	+1.55	+2.25	+11.65	0

The new antipodal cross-fading method (as described in subsections 2.1.1 and 2.1.2) results in the following reduced errors:

Error in ‰ ¹³	To B	To L	To P	To R	To M	To X
From B (Berlin)	0	0.006	0.008	0.022	0.017	0.056
From L (London)	0.006	0	< 0.001	0.005	0.042	0.051
From P (Paris)	0.008	< 0.001	0	0.003	0.050	0.056
From R (Rome)	0.022	0.005	0.003	0	0.084	0.065
From M (Moscow)	0.017	0.042	0.050	0.084	0	0.070
From X (Longy.)	0.056	0.051	0.056	0.065	0.070	0

While there is no formal proof, tests have shown that the maximum error of the new method is always less than 2 ‰ (0.2 %) for random points on earth.¹⁴ Therefore, the first limitation mentioned in section 1.1 gets lifted when using the new formula for calculating distances between points on earth.

2.1.5 Reversal symmetry

Also the second limitation mentioned in section 1.1 gets lifted by using the new formula: as already visible in the last table of the previous subsection 2.1.4, the error of the new distance function is not only smaller but also symmetric (e.g. the error when calculating the distance from London to Moscow is the same error as calculating the distance from Moscow to London). This is an improvement when comparing the method to the Taylor-based approach, which doesn't behave symmetrically (see corresponding tables in the previous subsection).

To ensure that not even floating point errors can yield to different results when swapping the origin and destination point, our implementation orders the latitude and longitude values prior to calculation (see source code for details).

¹³All errors in this table have a positive sign.

¹⁴This is less than the maximum error when earth would be completely modeled as a sphere.

2.2 Utilizing PostgreSQL’s Generalized Search Tree (GiST) framework

While the improvements described in section 2.1 can lift the first and second limitation mentioned in section 1.1, a mere formula for calculating distances cannot improve the syntax constraints (third issue) or allow nearest neighbor searches without the incremental workaround (fourth issue in section 1.1).

For these reasons, we considered using the GiST interface of PostgreSQL to provide an integrated solution for geospatial indexing.¹⁵

2.2.1 A short introduction to GiST

For the remainder of this section 2.2, we assert that the reader be familiar with the general structure of GiST indices. A short introduction to GiST can be found at the following URLs:

- <http://gist.cs.berkeley.edu/>
- <http://db.cs.berkeley.edu/papers/vldb95-gist.pdf>
- <https://www.postgresql.org/docs/9.5/static/gist-intro.html>

An implementation of a nearest neighbor search capable GiST index in PostgreSQL requires the implementation of the following 8 support functions: “consistent”, “union”, “compress”, “decompress”, “penalty”, “picksplit”, “same”, “distance”.¹⁶ The structure of the index differs depending on the particular implementations chosen for these functions.

After discussing the key structure for keys stored in the index (based on a space-filling fractal curve) in subsection 2.2.2, a short overview on the particular implementation of each of the 8 GiST support functions will be given in subsection 2.2.3.

2.2.2 Using a space-filling curve

While both PostgreSQL and PostGIS use GiST-based R-tree indices for geometric (and, in case of PostGIS, geographical) data types, an R-tree based implementation didn’t appear to be the best choice for geospatial indexing.¹⁷ Instead, we

¹⁵This was already proposed in section 3.3 (“Remaining tasks”) of work report June 6, 2016 as well as footnote 45 on page 16 of the same document.

¹⁶<https://www.postgresql.org/docs/9.5/static/gist-extensibility.html>

¹⁷An R-tree usually has overlapping nodes and requires to store bounding boxes for each node. For these two reasons, additional working memory would be consumed which, in turn, can be a problem for caching indices on huge datasets in main memory.

used an index based on a space-filling fractal curve. For the fractal, the Lebesgue curve¹⁸ was chosen for reasons of implementational simplicity.

Following the shape of a Lebesgue curve, a single point on earth can be represented by a bit string where the latitude and longitude bits are interspersed:

$$\phi_0, \lambda_0, \phi_1, \lambda_1, \phi_2, \lambda_2, \phi_3, \dots, \phi_{27}, \lambda_{27}$$

with

$$\phi_n \in \{0, 1\}, \lambda_n \in \{0, 1\}$$

This representation consumes 56 bits including the zeroth bits of latitude and longitude because $2 \cdot (27 + 1) = 56$. The worst resolution of the index is $40,000 \text{ km} / 2^{28} \approx 14.9 \text{ cm}$ at the equator.¹⁹

The leaves of the tree always contain 56 bit-long bit strings (7 bytes). The internal nodes of the search tree, however, need to cover a range of leaves. For this reason, truncated bit strings need to be stored in the nodes. This is encoded by appending a length information (requires 6 bits to store a length from 0 to 56 inclusive) to the key.

In practice, the total key size for single points is rounded up to 8 bytes for reasons of memory alignment and polymorphic functions (refer to the source code for details).

Beside storing simple points, the index shall also be capable of handling more complex objects that cover an area bigger than a singular point (e.g. paths, polygons, etc). To store such kinds of objects (and to allow index lookups on those entries), a center point of a bounding circle and its radius is included in the index keys. The center point can be encoded by interspersing the latitude and longitude bits as explained above. However, the radius of the bounding circle (i.e. the information on the size of the indexed object) must also be interspersed with the bits for latitude and longitude of the center of the bounding circle. The following scheme can be used:

$$\rho_0, \phi_0, \rho_1, \lambda_0, \rho_2, \phi_1, \rho_3, \lambda_1, \rho_4, \phi_2, \rho_5, \dots, \rho_{54}, \phi_{27}, \rho_{55}, \lambda_{27}, \rho_{56}$$

As the exact radius of the bounding circle doesn't need to be stored precisely,²⁰ we can restrict the choice of all ρ_n in such way that only one bit can be

¹⁸https://en.wikipedia.org/wiki/Z-order_curve

¹⁹Note that the resolution of the index is not limiting the resolution of the indexed data because PostgreSQL can recheck whether a condition is satisfied using the table data rather than the index data.

²⁰A larger value for the radius can be stored because PostgreSQL can recheck whether a condition is satisfied when querying the index. See “consistent” support function on page 8 and the associated footnote 24.

set to 1 while all other bits are 0. This allows to perform a logarithmic compression of ρ by simply storing an index n of that ρ_n which is 1.²¹ If $n = 57$, then all $\rho_n = 0$. Since $n \in \{0, 1, 2, \dots, 57\}$, appending a single byte to the index key is sufficient to consider the dimensions of an object referenced in the index tree. The total key size results in 9 bytes when indexing geographic objects with a size not always equal to zero (i.e. circles, paths, polygons, etc).

Object type	Key size of leaf nodes ²²	Key size of internal nodes
Points	7 bytes	8 bytes
Boxes	8 bytes	9 bytes
Circles	8 bytes	9 bytes
Paths	8 bytes	9 bytes
Polygons	8 bytes	9 bytes

Using the space-filling fractal as well as the logarithmic compression results in a maximum key size of 9 bytes, which is considerably smaller than storing the position and dimension of bounding boxes as done in R-trees.

2.2.3 Implementation of the 8 support functions

The following list gives an overview on the implementation of the necessary GiST support functions. For details, please refer to the source code.

consistent PostgreSQL allows three return values: “no”²³, “maybe”²⁴, or “yes”²⁵. Because the index key format (as explained in subsection 2.2.2) is lossy²⁶, only “no” and “maybe” are returned while “yes” is never returned.²⁷

For equality queries, the query datum is simply converted to a key, and then “maybe” is returned if and only if the key of the query datum is equal to the key stored in the tree.

²¹Increasing n by 1 corresponds to reducing the area of the bounding circle by 50 %, hence dividing the radius by $\sqrt{2}$. The special value of $n = 0$ is used for all bounding circles with a radius greater than a given reference length (which is in the same order of magnitude as the radius of earth).

²²PostgreSQL does not allow to specify different key sizes for leaf nodes and internal nodes unless using variable size data types. To reduce computational and implementational overhead, leaf nodes use the format for internal nodes. Hence the values in this column are not applicable for the implementation.

²³Returning false.

²⁴Returning true with “recheck” set to true.

²⁵Returning true with “recheck” set to false.

²⁶The resolution can be as coarse as 14.9 cm at the equator.

²⁷Note that while it still would be possible to return “yes” in certain cases, the implementational and computational overhead doesn’t seem to justify the necessary extra calculations.

For testing whether a point is contained in a bounding box, the key stored in the tree is converted to a bounding box, and then “maybe” is returned when this bounding box overlaps with the query bounding box.

For all other “overlap” queries,²⁸ a minimal distance between the query object (e.g. a circle for radial searches) and a bounding box²⁹ for the center point of the bounding circle of the indexed object is calculated. The maximum size of the indexed object (which is stored in the key using logarithmic compression) is subtracted from that value. If the value is smaller than or equal to zero, “maybe” is returned.

In all other cases, “no” is returned.

union Combining two keys is performed by simply truncating the bit strings of the keys (at the right side) to have the same length and then further truncating any differing bits from the right side of the strings.

compress The compress function converts geographic objects to the index key structure as defined in subsection 2.2.2.

decompress The decompress function is not used (and simply returns its argument) because the index support functions work directly on the stored key data format.

penalty The penalty function returns the number of truncated bits when creating a union of an original key with a key to be inserted. This roughly corresponds to returning the additionally covered area in case of R-trees.

picksplit The picksplit function decides which entries are moved to a new page when the size of the page grows too big, and which entries are kept. In case of R-trees, it is possible that entries are both kept and copied to a new page (overlapping bounding boxes). The fractal curve described in subsection 2.2.2 does not require overlapping index pages though. Instead, a union of all keys in the page is calculated. Then one bit is appended to the bit string of the union (this bit is set to 0 or 1). Those keys that are covered by one key (e.g. the union appended by 0) stay on the old page, and those keys that are covered by the other key (e.g. the union appended by 1) are moved to the new page. If the bit string of the union of all keys has already a maximum length (i.e. if no bit can be appended to the bit string), then a trivial split is performed where half of the entries are selected arbitrarily to be moved to a new page.

²⁸Operator “&&” in SQL.

²⁹Estimating the possible location of the center point, calculated from the (lossy) index key.

same The “same” support function simply tests two keys for equality. In our implementation, this check can be performed binary (using memcmp) because all unused bits are always initialized.

distance The distance function returns a minimal distance between the query object and the bounding box²⁹ of the center point of the bounding circle of the indexed object. Then the maximum size of the indexed object (which is stored using logarithmic compression) is subtracted. (This is basically the same mechanism described for the “consistent” function.)

The result is sanitized in such way that only zero or positive finite values are returned because PostgreSQL internally uses $-\infty$ and $+\infty$ for other purposes, such as handling NULL values.³⁰

2.2.4 Handling empty objects

Special considerations have been taken to handle empty objects in the index (objects which do not contain any point and thus do not overlap with any other geographic object or even themselves). These could be circles with a negative radius or a set of polygons that is empty.

In order to process empty objects (which are distinct from NULL), the “compress” function must return a special key (the “empty key”) that is not matching any location on earth. Furthermore, another special key (the “universal key”) covering both empty and non-empty geographic objects must be returned by the “union” function when combining keys of empty and non-empty objects.

The two special keys “empty key” and “universal key” are implemented without additional space consumption by storing a magic value in the byte that is normally storing the logarithmically compressed object size. For the details of the implementation, refer to the source code.

2.3 Adding new data types and necessary operators

In addition to providing the previously explained key format and associated GiST support functions, actual implementations for the desired operators had to be created. Most importantly, that is:

- the overlap operator ($\&\&$),
- the distance operator ($\<->$).

For most other operators such as the “contains” ($\@>$)³¹ and the “contained in”

³⁰<https://www.postgresql.org/docs/9.5/static/gist-extensibility.html> warns about returning $\pm\infty$.

³¹In older PostgreSQL versions: \sim (still used by PostGIS).

(<@)³² operator, the same index strategy as used for the overlap operator (&&) can be applied.³³ Only the operator functions (and necessary geometric operations) would need to be implemented for adding these operators.

The following data types have been implemented:

- `epoint`: a point on earth,
- `ebox`: a latitude/longitude range (e.g. describing a viewport for a map),
- `ecircle`: a filled circle (used in conjunction with the distance operator for radial searches with fixed radius),
- `eclass`: a collection of points, paths, (filled) polygons, and outlines (i.e. closed paths / non-filled polygons).

Particularly for polygons, certain geometric algorithms had to be implemented such as a “point in polygon” algorithm based on ray tracing³⁴ or geometric projections to determine the shortest distance between two polygons or a point and a polygon. As a result, the distance operator <-> works with high precision and does not use bounding boxes for an approximation of the distance to clusters/polygons. (Note: PostGIS, in contrast, uses inaccurate bounding boxes here.)

Currently not all combinations of operators and/or data types are implemented. This is ongoing work; for details refer to the published reference documentation.

2.4 Treatment of the poles and the 180th meridian

The distance calculation algorithm as explained in section 2.1 works properly both at the poles and the 180th meridian. The new algorithm automatically considers shortest paths crossing the north or south pole when calculating distances (e.g. between Iceland and Alaska). No special handling is required here.

The fractal index as explained in section 2.2 has a slight drop of performance near the poles because of the singularity in the coordinate system (at the poles all possible longitudes from -180 to +180 degrees are covered). The accuracy, however, is not affected by the index.

Whenever clusters are used (which may contain paths, polygons, or outlines of polygons), there is a drop in accuracy next to the poles that depends on

³²In older PostgreSQL versions: @ (still used by PostGIS).

³³If an object is contained in another object, those two objects always overlap.

³⁴https://en.wikipedia.org/wiki/Point_in_polygon

the maximum distance between the vertices of the polygon. This is because projections are performed in Euclidean space.

In order to avoid ambiguities in regard to eastern/western orientation of edges of paths, polygons, or outlines of polygons, each entry in a cluster may only cover a longitude range of less than 180 degrees. Areas that cover a wider longitude range are still possible by splitting them into multiple polygons.

3 Comparison with PostGIS

As PostGIS is only available under the terms of a viral license that is incompatible with certain other open source licenses,³⁵ using PostGIS is not an option for the LiquidFeedback project.³⁶ However, we will compare our created solution with PostGIS for informational reasons.

3.1 Constraints in PostGIS when treating the earth as a spheroid

PostGIS does not support global coordinates for a wide range of functions. To name a few examples:

- The `ST_Overlaps` function is not defined for the geographic data type.
- The `<->` operator returns true distances only between values of the geometry data type (which doesn't work globally) but uses “sphere distance” for the geography data type (which can describe global coordinates).
- The `&&` operator uses bounding boxes for determining whether two objects overlap (i.e. the operator may return true even if two objects do not overlap). This is an even bigger error than just modeling earth as a sphere.

Our created extension, in contrast, always uses spheroid coordinates for all operations.³⁷ Distances are measured in such way that all local operations honor the flattening of earth, even if the data type and/or table column stores global coordinates. Only for long-range distances, the error can grow up to 0.2 %, which is still better than modeling earth as a sphere.

³⁵Refer to page 9 of work report June 6, 2016.

³⁶LiquidFeedback aims to provide users and developers maximum freedom and accessibility to the source code also when combining it with any other software components. The GNU General Public License is contrary to these goals. See also: <http://www.public-software-group.org/licenses>

³⁷The only exception is when internally performing projections on a polygon, path, or outline in order to determine the shortest distance to a cluster (see section 2.4). The error, however, approaches zero in most practical cases.

3.2 OpenGIS “Simple Features Specification for SQL”

PostGIS provides many functions such as `ST.Distance`, `ST.MakePoint`, etc. as defined in standards by the OpenGIS consortium as well as ISO 19125. Implementing these functions as well as data storage formats that follow the OpenGIS geometry class hierarchy is possible but a matter of time and resources (see also following section 4.2 on compatibility considerations regarding additional data storage formats). It should be noted that PostGIS does *not* properly implement many of the functions when using spheroid coordinates that work across several continents, i.e. globally. This has already been elaborated in the previous subsection 3.1.

4 Compatibility considerations

4.1 Standard compliance

GIS standards may be followed on the SQL level or on the level of data representation when querying a database through REST API calls.

As already mentioned in section 3.2, extending our geospatial extension for PostgreSQL to include functions such as `ST.Distance`, `ST.MakePoint`, and associated data types on the SQL level is possible but a matter of time and resources (and budget).

Beside these economic concerns, it is currently not planned in the WeGovNow consortium to provide any form of SQL interface to third party components. Instead, a REST based API has been proposed (and is already implemented by GeoKey, for example). Therefore, compatibility must be ensured on the level of data representation when querying WeGovNow software components through REST API calls.

As we will see in the following section 4.2 on additional data storage formats, PostgreSQL enables a programmer to include any desirable data format with little overhead. Therefore, it is easily possible to make a REST API interface standard compliant with any kind of geospatial data exchange format including those which follow the OpenGIS geometry class hierarchy or any other scheme like GeoJSON (which is used by UCL's GeoKey).

4.2 Additional data storage formats

It is desirable to also support more complex geographic data formats such as GeoJSON or XML based documents, which may contain meta data. Whenever

needed, these formats can be stored directly in a corresponding JSONB, XML³⁸, or even TEXT table column.³⁹ All spatial indexing functions and operators as presented in section 2 of this paper, may still be used for these data formats even if the data types do not support lossless round-trip conversions. This is possible because PostgreSQL supports indexes on expressions (see chapter 11.7 of the PostgreSQL 9.5 manual⁴⁰). An application simply needs to provide a one-way mapping function from the stored geographic data type to one of the data types supported by the index.

4.3 Using a different spheroid

Also other reference spheroids than WGS-84 are thinkable by adjusting the corresponding constants in the C code. However, as the WeGovNow project depends on a compatible standard for coordinates anyway, it would be advisable to use WGS-84 as a common standard.

5 Publication of results

The developed software has been submitted to the Public Software Group e. V on August 18, 2016 and was published by the Public Software Group in the source code repository of LiquidFeedback Core⁴¹ under the terms of the MIT-License⁴² on August 18, 2016.

The source code has been modularized so that it is easy for other consortium partners to incorporate it, if desired. All previous work on LiquidFeedback in regard to spatial data was adjusted in order to use the new geospatial indexing module.

³⁸Support for the XML data type requires PostgreSQL to be compiled with the “--with-libxml” configuration option.

³⁹Such data types may be extended by adding validity checks using so-called “domains”. See PostgreSQL’s documentation on the CREATE DOMAIN command: <https://www.postgresql.org/docs/9.5/static/sql-createdomain.html>.

⁴⁰<https://www.postgresql.org/docs/9.5/static/indexes-expressional.html>

⁴¹http://www.public-software-group.org/mercurial/liquid_feedback_core/rev/96ee2db56bec

⁴²http://www.public-software-group.org/mercurial/liquid_feedback_core/file/96ee2db56bec/LICENSE

6 Conclusion & Outlook

It has been shown that PostGIS' spatial indexing functions can be replaced by a more lightweight approach within reasonable time. Previous limitations (see section 1.1) could be lifted. A usable implementation has been published as open source software under a permissive (non-viral) license.

While certain data representation formats are not yet supported by the new extension, these could easily be integrated with the current implementation (refer to section 4 for that matter).⁴³

© 2016 FlexiGuided GmbH, Berlin

⁴³Note, however, that other data structures (such as bézier curves etc.) would require further data type implementations.

Annex 6

Extending WeGovNow platform

Extending WeGovNow platform

Requirements and features to finalise WeGovNow platform in supporting service scenarios

WP3 Technical report

Authors: Alessio Antonini, Lucia Lupi – University of Turin

Version: 1.1

Date: 03/07/2017

History

Version	Date	Reason	Revised by
1.0	22.06.2017	First document version	Alessio Antonini
	26.06.2017	Workflows	Lucia Lupi, Alessio Antonini
	27.07.2017	Exhibit 2 and 3	Alessio Antonini
1.1	03.08.2017	Captions, revision of section 1 and section 3	Alessio Antonini
	08.08.2017	Revision of section 1	Alessio Antonini, Lucia Lupi

Summary

The following report is an assessment and analysis of service scenarios, use cases and preliminary requirements collected within the scope of the project, consolidated by joint work with local municipalities and technical teams. The key point of the analysis is to highlight the processes behind the scenarios, which interconnects the “day in a life” use cases to a process of progressive integration of the platform in the city life synthesized in four types of general workflows. Furthermore, the report presents the main requirements for the adoption of the platform by the local stakeholders and municipalities, completed by a brief assessment of the technical feasibility within the technological framework of the project. Finally, considering the key action points of workflows, it is presented a possible implementation to support scenarios using in the current WeGovNow platform.

Table of Contents

Summary	1
1 Introduction	3
1.1 Addressing the service scenarios	4
1.2 Evaluating requirements	5
2 Adoption of WeGovNow platform	9
2.1 Interoperability with existing tools	9
2.2 Connecting with existing authentication services	10
2.3 Unified access to user guides	10
2.4 Single helpdesk	10
2.5 Organisation account	10
2.6 Custom monitoring of entities	11
2.7 Importing data	11
2.8 Exporting data	11
3 Requested features and integrations	12
3.1 Co-design for public administration-driven projects	12
Phase 1: Call for participation	13
Phase 2: Co-design activities (online and offline)	13
Phase 3: Discussion and selection of proposals	14
Phase 4 Call for contributions & call for tender	14
Phase 5. Follow up about the project	15
3.2 Co-management of public spaces, commons and shared resources	15
Phase 1: mapping of public spaces/buildings + urban commons	16
Phase 2: Groups formation, development of civic proposals, and selection of proposals	16
Phase 3: co-management of spaces, monitoring and reporting.	17
3.3 Coordination of private initiatives facilitated by the Public administration	17
Phase 1: Collaborative building of the business/managing model	18
Phase 2: Local network activation (Food system for san Donà)	18
Phase 3: Inter-organizational coordination in using the common space	18
Phase 4: Network consolidation and growing	18
Phase 5: periodical call for ideas/projects	19
3.4 Co-production of services between public and private sector	19
Phase 1: Mapping of public offices/facilities and sharing of public information about one or more sectors of city services	20
Phase 2: Collaborative monitoring and data integration (institutional and crowdsourced)	20
Phase 3: Aggregation of information about local activities, initiatives and projects independently implemented by local stakeholders	21
Phase 4: Participatory process to define new targets and services	21
Phase 5: Co-production of services between public and private sector	22
Conclusions	23

1 Introduction

WeGovNow platform should be considered primarily a digital environment for civic technologies, oriented to the we-government of urban resources such as commons, policies, projects among local stakeholders and citizens. The development of WeGovNow platform can be summarised as follows:

- a) the development of the **environment services**, providing the basic functionalities to enable the orchestration among peer components
- b) the development of **new components, modules and features** to be integrated in one or more components
- c) the development of **integrations between components** supporting the applicative workflows.

The platform development is organised accordingly with three milestones corresponding to the completion of each step.

1st prototype (released in May 2017) enables users to switch between components and to use their functionalities with the same account. Users can login using a common navigation bar and switch between components keeping their session.

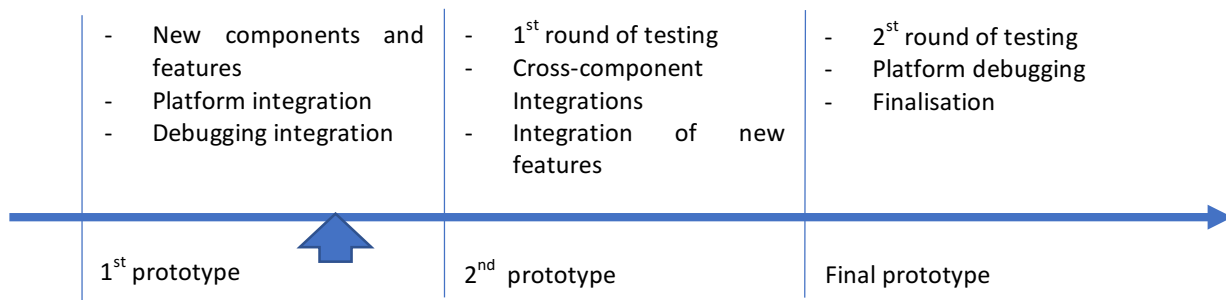
2nd prototype (released in July 2017) will include the new component Trusted Marketplace, OntoMap logger and LandingPage. Users will be enabled to explore their activities within the WeGovNow platform from the TrustedMarketplace and the LandingPage, thanks to the integration of all components with the OntoMap logger.

Final prototype (expected for the end of October 2017) will provide integrations between components in terms of content and features deep linking. The integrations will be specifically designed to support the “Service Scenarios” provided by the Municipalities of the trial sites, developed in workshops involving local stakeholders.

The development of WeGovNow platform as described is supported by engagement activities hosted in the three trial sites of the project, and by rounds of expert testing. The outcome of first year and half of engagement activities, where reported as “Service Scenarios” providing inputs for the development of the new components and new features, and other accessory solutions to support the use of the platform, accordingly to the vision of local stakeholders. The expert testing produced specific reports for each component, with the aim of improving their accessibility aspects.

Between the 2nd and final prototypes and between the final prototype and the first release of the platform, expert testing will be focused on both accessibility and usability of new components and of the integrated platform. Following, the use by end users will provide further input to polish the integration and enhance the overall usability of the platform with the goal of increasing the success rate of adoption in the local contexts of trial sites.

Exhibit 1. Current development status and aims.



Currently, the platform development activities are focused on the development of new components and on the finalisation of the platform environment (between the 1st and 2nd prototypes), see Exhibit 1. In this point in time, the development activities are focused on delivering new components and features, and on incorporating the relevant inputs as features of the new components (under development) and as integrations to be developed within the final prototype.

1.1 ADDRESSING THE SERVICE SCENARIOS

Service scenarios are collections of general needs, preliminary requirements, daily-based use cases, and intuition about how municipalities envisage the use of the platform in providing services to and with the citizens, summarised in deliverable D2.3, Annex II. The collected inputs had been assessed by technical teams, under the light of the scope of the project, of the availability of time and resources, of the existing components and their workflows, resulting in a list of indications for further development of WeGovNow platform.

The process of assessing service scenarios and specification of requirements has been developed as follows:

1. extend service scenarios from a specific application example to a general pattern of use
2. mapping the fragmented daily use cases in steps of full service processes, involving multiple actors in wide-range of time
3. collecting the missing information about the context and work procedures, important for the adoption of the platform
4. extracting the underlying workflows from the service patterns
5. clustering similar workflows in common we-government processes
6. mapping workflows' steps with components, to highlight missing features and required extensions
7. validation of workflows and corresponding platform functionalities with a subset of local municipalities (up to now with Turin and San Donà di Piave)

The assessment process required to build a deeper understanding of applicative context in terms of **workflows, scope of activities, goals, key steps**, which brought the analysis presented in the following Section 3. The collected service scenarios are used as starting point for further discussion between development teams and the municipalities (as representatives of local stakeholders involved in the platform design). The direct communication between stakeholders and development teams is resulting effective in terms of consolidating the applicative scenarios, and defining the next development tasks.

1.2 EVALUATING REQUIREMENTS

Service scenarios are partial in terms of both understanding of WeGovNow functionalities and in terms of description of services. Consequently, use cases and requirements presented within service scenarios have to be considered as **general indications** of how the platform should behave, and how the platform will be adopted. Considering those limitations, requirements were analysed using the following criterion:

- a) **the underlying workflows**, extending the scenarios to cover the full life-cycle of the service to expose hidden requirements, to prioritise the tasks and to find common solutions among different scenarios and trial sites
- b) **alternative implementations** of use cases, exploiting the existing mechanisms of the consolidated components of WeGovNow
- c) **the scope of WeGovNow**, what falls in the field of we-government and help in defining what is out of scope for the platform development
- d) **existing tools** used by or available to stakeholders and to end users for ancillary or complementary activities.

The analysis highlighted the following classes of requirements:

- 1) **External requirements**, expectations regarding the dissemination and communication of the platform toward end users to involve them in the platform
- 2) **Unspecific non-functional requirements**, about general expectation toward the platform as technological tool
- 3) **Specific non-functional requirements**, about overall setup of the platform and the internal coherence of components
- 4) **Specific functional requirements**, based on a specific implementation of scenarios.

External requirements (1), such as the platform must be promoted and used by a large public or the municipal staff should be trained in using the platform, cannot be addressed within the scope of developing the platform. Unspecific non-functional requirements (2), such as usability, accountability, safety or transparency of data, cover basic principles which are best practices and constraints already addressed. Both types could be considered as indications for the communication of the platform: what users may be concerned of, strong points of a high quality technological platform, etc.

Specific non-function requirements (3) are important to support the daily use of the platform, improving the overall perception of the platform as “one” and the consistent use of different features. This class of requirements should be addressed in late stage, to finalise WeGovNow, by working on improving the visual coherence of components in terms of colours, position and recognisability of recurrent elements, type of interaction for similar features, etc.

Specific functional requirements (4) are what should be addressed in the current stage of development: inputs for the new components (2nd prototype) and input for integrations (final prototype). A relevant set of specific functional requirements are cross scenario and are aimed to support the adoption of the platform in their work environment, within the current standards for software tools that trial sites are already using. Following the existence of a set of adoption requirements, we explored the issues regarding the adoption of WeGovNow and extracted a small set of specific requirements presented in section 2.

As mentioned, specific functional requirements of components had been evaluated considering:

- i) Schema of general workflows and scope of components involved phase by phase in the service scenario, implemented by the consolidated technologies
- ii) Possible alternatives to use cases, based on existing mechanisms
- iii) Use of existing collateral tools available
- iv) Cost/benefit assessment in terms of strict relation of the requirement with the implementation of the scenario
- v) Number of occurrences of requirements in scenarios and steps of workflows
- vi) Number of expected end users and stakeholders interested to a specific feature
- vii) Technical feasibility within the project technological framework

The outcome of the analysis of service scenarios and requirements is to focus the development effort on supporting the adoption of the platform and on supporting its use by end-users. Considering Exhibit 2, the development of WeGovNow platform will focus on addressing:

- Functional requirements specific to a component or to the platform as whole
- Non-functional requirements related to the platform in general

Furthermore, each single development teams will address the non-functional requirements related to their components if possible, along with the development of new features and integrations with other components.

Exhibit 2. Grid of covered requirements considering their classification of functional/non-functional in relation to their scope: specific to the platform, specific to a component or unspecific. In green the focus of the development activities, in blue the requirements that will be addressed if possible case by case, in red the class of requirement which will be not addressed.

	Specific to the platform	Specific to a component	Unspecific
Functional	<ul style="list-style-type: none"> • Export data • Notifications Organisations 	<ul style="list-style-type: none"> • Import data • Rules of public calls • Search for requests 	<ul style="list-style-type: none"> • Usable • Reliable • Intuitive
Non-functional	<ul style="list-style-type: none"> • Documentation • Standards • Support users 	<ul style="list-style-type: none"> • Usability • Accessibility • Look and feel 	<ul style="list-style-type: none"> • Communication • Advertisement • Training

Considering the generalisation of service scenarios and the output of the requirement elicitation process, it is possible to outline “specific roles” for each component, within the foreseen use of WeGovNow platform in the service scenarios. If the requirements about the platform in general will support the platform adoption, the functional requirements specific to components could be interpret oriented to support their “roles” in the workflow, integrating features and creating paths between components. The roles of each component could be described as follows (see Exhibit 3, actions):

- GeoKey/CommunityMap
 - Surveys (collecting, and presenting summary information)
 - Present survey results (importing surveys)
 - Making surveys (importing setup, exporting results)
 - Project plans
 - Including a plan
 - Alternative plans (plans related to the same call)
- FirstLife
 - Group localisation
 - Documentation of activities
 - Events and Calendars
 - Dissemination of initiatives / news
 - Exploring data (location and time)
 - Crown mapping
 - Monitoring and coordination
- ImproveMyCity
 - Reporting issues related to projects
 - Monitoring implementation of projects
 - Communication with local authorities
- LiquidFeedback
 - Proposals formation
 - Contributing to projects and proposals
 - Decision making
 - Submitting proposals to public calls
- Trusted Marketplace
 - Search for topics and collaborations
 - Active dissemination of initiatives, calls, etc. (recommendation)
 - Exporting / monitoring activities

In our opinion, the next tasks leading to the final prototypes should be aimed specifically to the extension of existing components, to support their roles and to support the adoption of the platform in general (see Exhibit 3, new features and integrations):

- GeoKey
 - Importing plans
 - Importing surveys
- FirstLife/OntoMap
 - Exporting data
 - Exploring data
 - Area calendar
- ImproveMyCity
 - Zone to units
 - Boundaries
 - Public / private management
 - Private reports
- LiquidFeedback

- Support to public calls (description)
- TrustedMarketplace/OntoMap
 - Exporting data
 - Search
 - Recommendation

Exhibit 3. Table summary of the actions, integrations and new features for each component, involved in the workflows.

	CM/GK	FL/OTM	IMC	LF	TM/OTM
Actions	- surveys -group map -group plans -sharing plans	- local groups - events -crowd mapping - monitoring - dissemination	- reporting - accountability	- proposals - contributin g - decision - rules	- collaborations - monitoring - dissemination - group making
New Features	- importing plans - Importing surveys	-exporting data -exploring data -area calendar	- zone to units - boundaries -public / private management - private reports	-call description	- exporting data - search - recommendation
Integrations	-to initiative	- all initiatives - all plans - all activities	- from map	-group plans - from map -from calendar	- notifications - initiatives - groups - initiatives

2 Adoption of WeGovNow platform

The features and requirements in common among trial sites and service scenarios deserve particular consideration in terms of cost/benefit evaluation of the development effort. Moreover, features and requirements about the adoption of WeGovNow platform are critical for the success of the project itself. Therefore, our assessment is to prioritise adoption requirements without reservation, considering the project scope and time/resource frame.

The adoption of the platform results to be related to:

- a) **Interoperability** with existing tools: the possibility to interconnect WeGovNow platform with existing software through the adoption of standards.
- b) **Connecting** with existing authentication services used by other local services.
- c) Unify the mechanisms to **access to guides and FAQs** of each specific component
- d) Single **helpdesk** for WeGovNow platform
- e) **Organisation** accounts, users from an organisation should sign their contents as their organisation sector (e.g. Dept. of social services of the municipality of...)
- f) Custom **monitoring of entities**, the possibility to set up **notification preferences** at entity level to receive emails about the updates.
- g) **Import** project data, from csv loading both metadata and records
- h) **Exporting** datasets, a form where to specify time interval, location, topics to export data platform wise, with additional setups regarding the inclusion of comments or other related contents.

In our opinion, supporting those requests can be archived within the platform framework and scope, nevertheless we should be prioritised by considering also specific functional requirements, in particular the **minimal set of requirements to enable the implementation of service scenarios**.

2.1 INTEROPERABILITY WITH EXISTING TOOLS

Interoperability with existing tool is interconnected with task 3.7 “Interlinking with existing systems at trial sites and technical support”. The approach adopted in WeGovNow documented in “Consolidated WeGovNow system architecture”, deliverable D3.3, addresses this requirement by the adoption of standards:

1. REST APIs
2. GeoJSON format
3. RDF triplestore

If the adopted solutions are sufficient to address this requirement, should be verified by assessing a possible integration with a small set of existing services. This will be done by UniTo with the support from the trial sites for the identification of candidate services, and by all technical partners for the assessment.

In our opinion, this requirement should be resolved by demonstrating that integration is possible and relatively cheap in terms of development, and by addressing other following requirements such as integration with authentication systems, data import, data export, without requiring any other specific development.

2.2 CONNECTING WITH EXISTING AUTHENTICATION SERVICES

Currently, WeGovNow has its own native authentication server (UWUM) and enables the use of two more from social media platforms (Google+ and Facebook). On the other hand, municipalities and big organisations have already their own authentication systems, for end users and for their employees. The requirement is to connect WeGovNow with local authentication systems, in order to enable end users to use the same “official” account they use for all other local services.

This requirement is a derivation of the system interoperability and therefore related to task 3.7. UniTo will enquiry the trial sites in order to assess what are the local authentication providers used in public services, what technology they use and if and how we could archive an integration.

We began by assessing the case the municipality of Turin, they have a system called “FaciliTo”, which is part of “Sistema Piemonte”: a Shibboleth 2.0 authentication service. In the beginning of 2018, FaciliTo will close to new registrations but currently, it counts more than 100.000 users (40.000 active in the last year). From the beginning of the next year FaciliTo will be sided with SPID (the Italian ID), which can require the ID verification by a public officer of the user.

SPID is based on OAuth 2.0, but the technical documentation is in Italian only. We are currently trying a direct integration of SPID with FirstLife authentication server. If successful, we will support the integration of SPID within UWUM. Moreover, SPID could be applied in San Donà di Piave too.

2.3 UNIFIED ACCESS TO USER GUIDES

Each component has its own documentation and method to provide it: in line, as pdf, in a web site, etc. This requirement is not about building a unified documentation, but to provide documentation in a unified and reliable way to end users: a common mechanism to access to the documentation, a method which does not change regardless the component.

A possible approach is to extend the component configuration with a link to a support web resource (it could be the pdf, a web site, etc.) The introduction of this reference could be used to extend the navigation bar, to introduce a contextual help button in the right corner to show the support resources of the current component.

2.4 SINGLE HELPDESK

A WeGovNow instance should have one single helpdesk for all components. This requirement is supported by the best practices and guidelines, the platform helpdesk has to provide a single first level which will report the technical issues to single technical teams if the issue if needed.

This requirement is also included in task 3.7 “Interlinking with existing systems at trial sites and technical support”.

2.5 ORGANISATION ACCOUNT

This requirement comes by municipalities but it could be extended to formal organisations, it comes from the consolidated custom of publishing contents “under the umbrella” of an institutional account in structured organisations. It is critical for formal organisations to 1) give a proper frame to their employee contents, and to 2) control the information flow. From the perspective of employees, it is critical to 3) be protected “under the veil” of the institutions avoiding personal

responsibility. The expected outcome is to have contents posted by organisations, supporting the internal structure of the organisation and the traceability of contents within the organisation (no shared accounts).

Currently structures among WeGovNow components, there are multiple approaches to organisation that could be use. In our opinion, we should identify the most suitable one and design a solution based on the native mechanism of one of the consolidated components. The most suited one is LiquidFeedback since it provides authentication functionalities, but also Trusted Marketplace could provide this feature as part of the user setup.

2.6 CUSTOM MONITORING OF ENTITIES

From the perspective of the local authority, it is very important to manage key steps or key information such as a public call. An effective management requires to: 1) monitor the evolution of specific contents such as a project proposal, 2) keep the quality of notification low in number and focused the enable an effective use of the platform by municipality staff. Summarising, the notification system should support both digests about locations or themes and punctual updates related to specific entities.

Each component includes its own notification system, but Trusted Marketplace in particular could support end users in managing the overall notification setup exploiting UWUM user's profile settings. To support notifications at entity level we could plug a notification system to OntoMap logger: as long components send users' activities to logger with explicit reference to entities it is possible to build cross-component fine grain notifications, assuming the consistency of references between components. To support the consistency of references WeGovNow components can rely on linked on data provided by OntoMap, on InputMap and on other components APIs to consolidate the references.

2.7 IMPORTING DATA

Even though WeGovNow instances for the trial sites will provide a relevant set of local linked open data, the platforms should support the autonomous intervention of municipalities in importing new datasets: open data, projects plans, proposals, etc. The municipalities wish to be enabled to load data sets in a semi-automated way from in the most common format of spreadsheet (.csv, .xls).

The most type of data are geospatial datasets, supported by GeoKey, FirstLife and OntoMap. The choice should be done considering the very different type of information and features provided by the platform. If the point of automated import is to input a project plan or surveys, then the choice should fall on GeoKey, if the goal is to collect contributions, initiatives and other contents then it should be done via FirstLife. About importing open data, it is not realistic at this point of time to provide automated tools for ontology alignment.

2.8 EXPORTING DATA

In order to monitor and report about project and activities, it is expected to be enabled to use the filtering capabilities of WeGovNow to extract datasets, for instance about specific topics, time frame, locations, etc. The export of data should be executed by organisation and municipality staff. Currently, component data can be retrieved via API but about platform wise data there is not a solution ready. Nevertheless, querying OntoMap linked data or from OntoMap logger could be a starting point if we manage to build a crawler which to resolve deep links.

3 Requested features and integrations

The service scenarios indicate a vision about the use of the platform, providing a hypothetical use in a day of the platform. On the other hand, the analysis of collected applicative scenarios made evident that their implementations require an extensive use of the platform for months or years, far beyond the presented use cases. Therefore, in order to build a deep understanding of the scenarios, we introduced a workflow analysis, to highlight the key steps of each scenarios and define a sequence of phases to reach the integration of the WeGovNow platform in the trial sites environments, searching also for common patterns.

Service scenarios are described also in relation to very specific situations and groups of stakeholders, but it is possible to find common patterns among different scenarios and extend or generalise them in order to represent the potential application of WeGovNow platform as a continuous support tool for municipality and local stakeholders in a wide applicative context.

The analysis of service scenarios, completed by some additional interviews to clarify unwritten points and aspects, lead us to rethink them as services progressively structured and used in the trial sites contexts described by synthetic workflows.

The results are four main workflows of WeGovNow platform:

1. Co-design for Public Administration-driven projects
2. Co-management of local resources between local stakeholders and Public Administration
3. Co-production of services between public and private sector
4. Coordination of private initiatives facilitated by the Public Administration

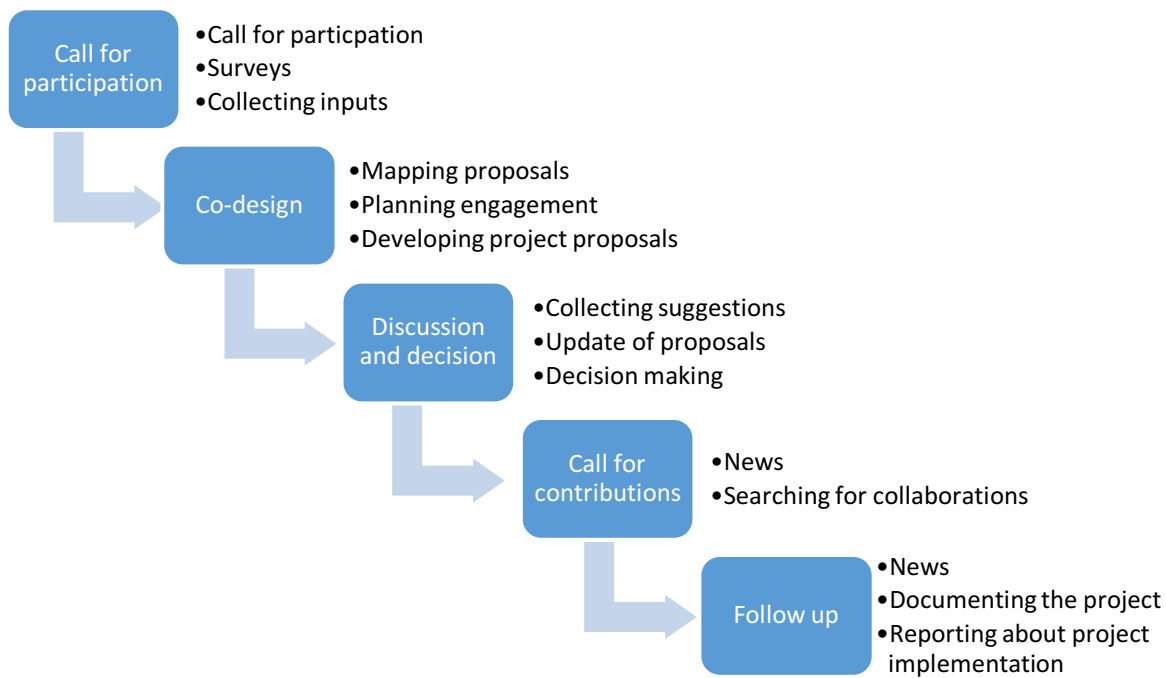
Following, the workflows descriptions and their key steps.

3.1 CO-DESIGN FOR PUBLIC ADMINISTRATION-DRIVEN PROJECTS

The workflow of a co-design process usually can be defined by a public administration promoting or starting a participatory process aimed to involve citizens and local organizations in expressing their needs and priorities, suggestions, proposals about the transformation of a specific area or the implementation of a new service. The engagement path can include offline activities such as seminars or workshops to be integrated with a wide participation through the online platform or can be organized online only.

The main goal is to share the responsibilities of the decision making process and design plans or interventions responding to the local needs and expectation.

The co-design workflow is a generalisation of the following scenarios: “Co-design of an area for teenagers and young people” where the objective is to design a physical transformations of a Parco Dora sector; the “Iron valley – cultural open air museum” aimed to involve citizens in design the future activities and build the museum heritage; the “Highway scheme scoping exercise” oriented to perform a community need analysis and a collection of inputs from local stakeholder for a public plan.



Phase 1: Call for participation

The Municipality define the guidelines for the project to be developed and launch a call for participation in the design process opened to citizens, organizations, local business, etc. operating close to the project area or interested to the project theme.

a) FirstLife – public information about the call

An officer of the department promoting the project publish the news of the call on FirstLife to inform citizens about this new public initiative. Citizens can explore the map, find the news and contribute by sharing additional contents.

b) Geo-key / Community Maps - Context analysis and publication of open/public data in thematic maps.

If the department in charge of the project already have local surveys or one of the task is to make a detailed context analysis or there is a preliminary plan, Geokey/Community maps could be used to provide to citizens an intuitive visualization of all the available information regarding the project area with different thematisms: environment, local services, population, etc.

c) Liquid feedback – rules of participation in the process

The office in charge of the project open a new session on Liquid Feedback dedicated to the project in order to clearly communicate how to participate online and offline (methods, events, timing, access, evaluation of proposal, etc), start thematic discussions about the project, collect users' inputs in a preliminary phase.

Phase 2: Co-design activities (online and offline)

Workshops organized by one or more departments of the municipality in collaboration with local organizations, and online activities to extend the offline participation. Workshops can be organized

in one or more cycle, progressively oriented to define the functions and then components and design of the area.

a) FirstLife: collaborative documentation of offline activities and integration of online activities.

- Planning of participatory workshops by using events
- Documentation of offline meetings by using extra, posts, and other entities
- Upload of reports of offline meetings
- Online co-design space (collection of proposals, suggestions, comments)
- Integration of data regarding the project area with proposals open for discussion within georeferenced groups
- Collection of proposals regarding the entire project area or related to a sub-area
- Collection of proposals related to different topics
- Search of a proposal by tags

b) Geokey-community maps: group maps corresponding to group proposal

Every group participating to the workshop can make its own map, drawing the elements of the proposals directly on the map by using drawing tools, and engaging other people to contribute. Proposals including activities or services can indicate the area where they can be implemented.

Phase 3: Discussion and selection of proposals

Participants to offline and online activities can discuss about proposal to be developed into a final project and select their favourite option.

a) Liquid feedback

- Public discussion about the main proposal developed during the offline activities
- Integration of new contributes in that proposals
- Versioning and evolution of the proposal over time until the voting phase
- Collection of suggestions, comments, alternatives
- Selection of a preliminary project through an online vote

A critical/open point is how to differentiate people (and proposals) involved in both, offline and online activities, and people contributing online only.

Phase 4 Call for contributions & call for tender

The municipality organize a call for tender based on the final project designed by the technical officers of the municipality and a call for contribution opened to local supplier or business or professional that want to sponsor one or more component of the project.

a) FirstLife

News about the updates related to the call for tender

Coordination groups for citizens that want to contribute in place making

b) Trusted market place

Call for contribution addressed to local business and professionals or people interested in contributing by sharing resources and expertise to build outdoor furniture, decorations, and additional components.

Phase 5. Follow up about the project

Works start and participants to the process can stay up to date about what it is happening in the project area and contribute to monitor issues.

a) FirstLife

News, groups, events, stories involving the administration and local stakeholders

b) Improve my city

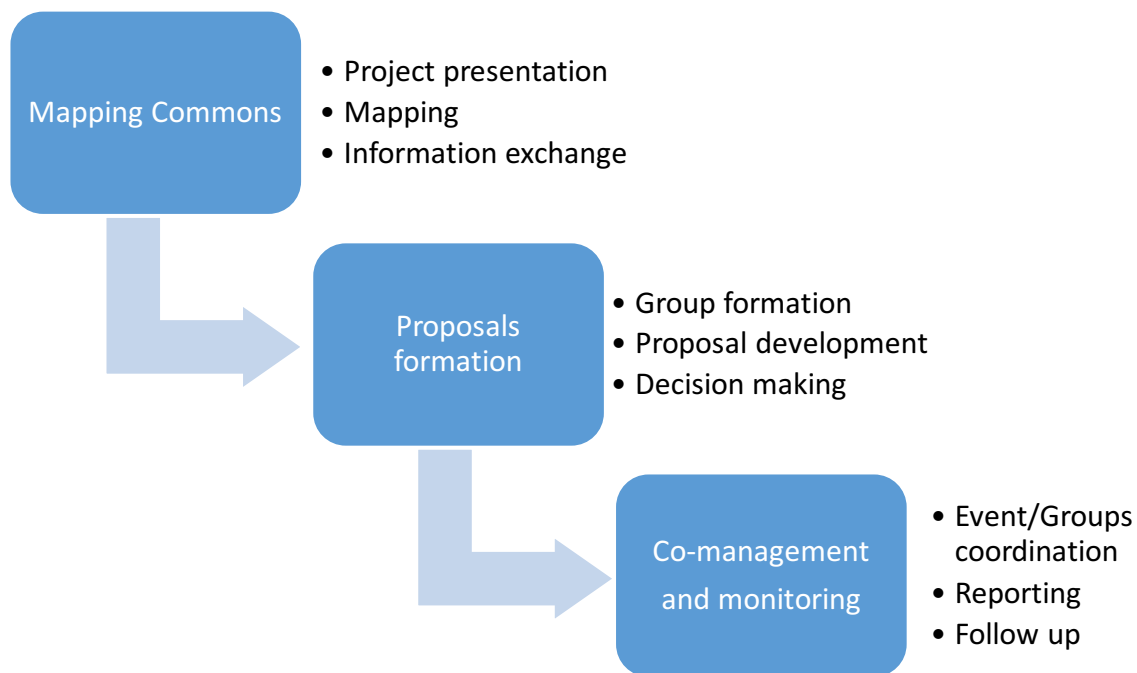
To collect local report within the perimeter of the project area.

c) OntoMap/Trusted Market Place

The Municipal staff want to find all data related to a specific topic, area, user or class of users, time interval and using it for internal or institutional reports in an aggregated way throughout the process.

3.2 CO-MANAGEMENT OF PUBLIC SPACES, COMMONS AND SHARED RESOURCES

The workflow of the co-management of public spaces, commons, or shared resources in the urban context involve the Public Administration in sharing the responsibility of managing them by supporting the civic proposals of citizens and local organizations. The main goal is to offer a clear framework allowing citizens in contributing, self-organizing their activities and extend their network. At the same time, the monitoring of how community resourced are used is crucial for PA. The workflow of the coordination of private initiatives is a generalisation of the following scenarios: “Regulation of common goods: Hortus Conclusus” and “Cleaner Greener Safer engagement map”. The second scenario cover only the first part of the entire workflow to have an interactive map of past project and proposals.



Phase 1: mapping of public spaces/buildings + urban commons

a) Geo-key / community maps: PA data

The Municipality create a new project on Geo-key to map the public buildings and spaces available for citizens and groups proposals. The Staff of the PA can decide the structure of the project, categories, material, type of descriptions, and all field useful to represent this kind of information. In case of dataset already prepared they can be imported on Geo-key structuring automatically the project.

Analytic representation of places + complete informative cards.

b) FirstLife: participatory mapping based on flexible entities (no structured field).

Citizens and local association, all platform users in general, can map new places that they consider urban common and suggest that they could be activated or regenerated.

Sharing of knowledge about places and their perception.

c) OntoMap:

The Municipality, but also local organizations, want to see an aggregation of heterogeneous data associated to the same spatial entities on both platform and to switch from a crowdsourcing environment to an institutional one.

Phase 2: Groups formation, development of civic proposals, and selection of proposals

a) FirstLife: geo-referenced public groups

Groups can be used to propose an idea of reuse of public spaces or urban commons. User can see the proposal on the map, stay up date on news coming from the PA or about territorial events on the chosen space, read contents and suggestions shared by other contents, access on Liquid Feedback.

b) Liquid feedback: collaborative development of proposals and voting

PA: can set a new session on LF dedicated to the collection of proposals for a specific space, or a type of agreement (such as pact of collaboration or others) or a class of spaces (such as primary schools in the district 5). Pa can set the rules for this new session: interval of time to collect proposals

needing a municipal support, characteristics of the proposals, and evaluation criteria for each proposal and explicit the relation between online voting and administrative attribution of the space. Citizens and Associations can:

- Add a new proposal (initiatives) and develop it with others, integrating suggestions and reviews.
- Support an existing proposal joining the initiatives
- Vote for the preferred proposals.

Phase 3: co-management of spaces, monitoring and reporting.

a) FirstLife

Events, Groups coordination, News, Sharing of experiences, Place calendar, Storyline of the project.

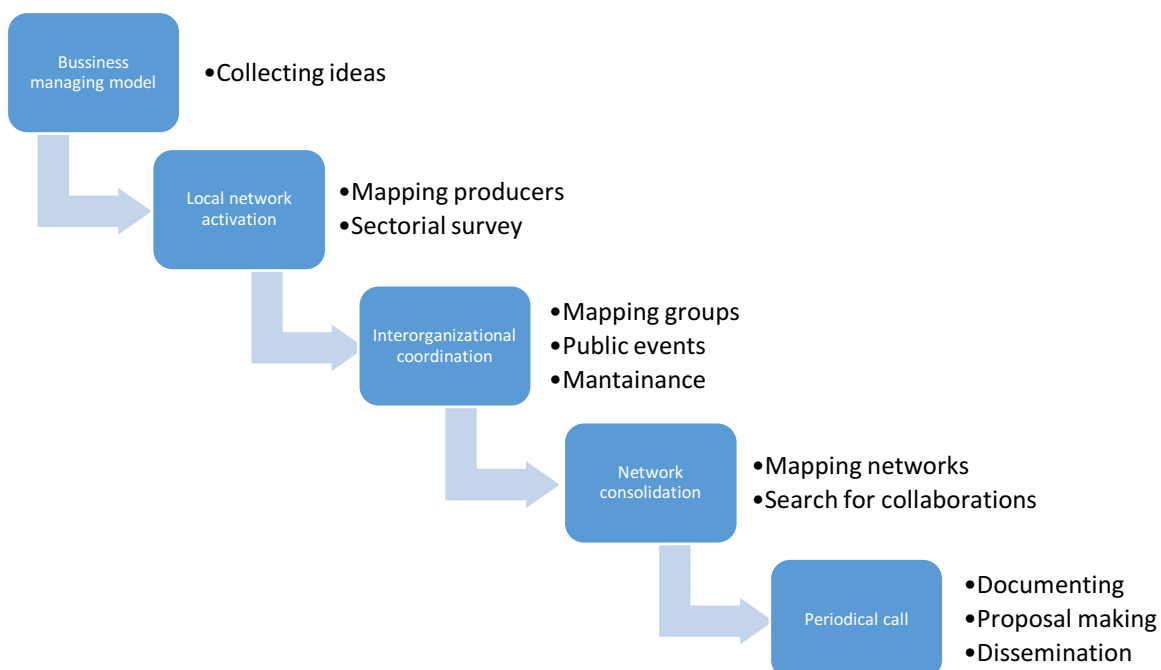
b) Improve my city

Report of issues and malfunctioning in the building or area used by the groups managing the area. Sending of request for using other spaces and receiving authorizations.

3.3 COORDINATION OF PRIVATE INITIATIVES FACILITATED BY THE PUBLIC ADMINISTRATION

The workflow of the coordination of private initiatives in a public space or in a private space with public functions involve the Public Administration as facilitator of the aggregation of resources and/or people to implement project and initiatives for the local development, in an economic and social perspective. The main goal is to support companies and private organizations managing or operating in the same space to cooperate and open new opportunities for the local communities through regeneration projects and/or business projects.

The workflow of the coordination of private initiatives is a generalisation of the following scenarios: “The Wine Cellar as a local innovative business network” and “Dora Shopping Mall and IPERCOOP”.



In San Donà di Piave, the applicative scenario is focused on the Wine cellar, a building under rehabilitation that will host local food producers for commercial, educational and recreational activities. The main goal of the project is to define a new collaborative business models for the stakeholder that are going to use that space and opened it to the community.

In Turin, the applicative scenario involve as main actors the company managing the Dora Shopping Mall and the Ipercoop, the supermarket inside, characterized by a growing attention toward the community through initiatives and projects hosted in their public spaces.

The second scenario cover only the last part of the entire workflow defined by the first one.

Both can be extended to the other spaces managed by private actors that want to contribute in community activities oriented to social innovation and experiment new economic and governance models in a protected environment.

Phase 1: Collaborative building of the business/managing model

The public administration involve the companies selected to use the public space in starting a democratic process to collaboratively design and decide the business models to be implemented within the producers network in the local hub.

a) Liquid feedback

Initial collection of proposals, development of the most supported ideas, evaluation of alternatives and concurrent proposals, versioning and evolution of the main idea.

Phase 2: Local network activation (Food system for san Donà)

a) FirstLife

Mapping of producers (production and selling spaces, stories, updates about services, etc.)

b) Geo Key/community map

Thematic maps of a single producers or of a group of producers to communicate sectorial information.

Phase 3: Inter-organizational coordination in using the common space

a) FirstLife

Public groups and events where everyone can read what is happening and what is planned in a specific area, explore which events and activities had been organised in the past.

(A reservation system is out of scope, also because areas often have been already assigned and the management of spaces can be supported also by other existing digital/analogic tools)

b) Improve my city

Integrate the reporting system connected with the municipality offices with a reporting system active for the area under private management and its maintenance office to offer a unified reporting system to citizens and intervene according to the office in charge for each zone.

Phase 4: Network consolidation and growing

a) FirstLife

Activation of producers in mapping their network of suppliers and distributors, local partnerships and initiatives where they are involved, inviting the related prospect users to enter on the WeGovNow platform and use it to consolidate and extend the network. They can enrich the map by using groups, events, but also collaborative and distributed stories about products and services.

b) Trusted market-place

Research of collaboration, volunteers, partnership for projects calls, resources and services. Visualization of the information divided per theme, topic, keywords, and area or time interval. For the Public administration is an information tool to find information about activities needing help and to build better program for funds and policies to support the local development starting from the economic and social ecosystem.

Phase 5: periodical call for ideas/projects

a) Trusted marketplace

Publication of calls for ideas to organize temporary activities in the local hub involving citizens and local associations and disseminations to users interested in a specific topic or active in related activities.

b) Liquid Feedback

In complex call, use of the platform to build integrated proposals by integrating initial proposals with suggestion coming from new groups and users.
If needed, selection of best proposals.

c) FirstLife

Publication of news and updates on the map.
Collaborative and multi-perspective documentation of activities and projects hosted in the local hub with their impact and collaboration on the territory with schools, public and private organizations, community groups, etc. Archive of past, present and planned activities, exploration by keyword or name of the event/project.

3.4 CO-PRODUCTION OF SERVICES BETWEEN PUBLIC AND PRIVATE SECTOR

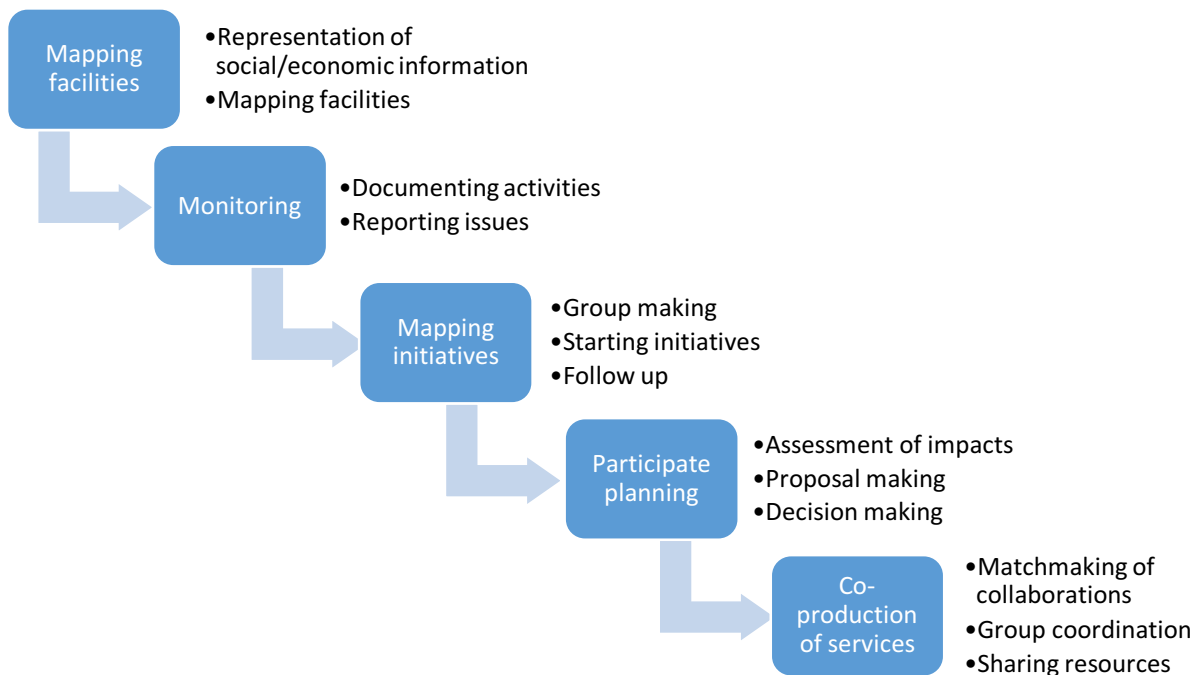
The workflow of the co-production of services between public and private sectors is a complex scenario where the Public Administration has the role of initiator of a process aimed to create an integrated system of services in a specific or crossing domain. The main goal is to involve privates, businesses, professionals, and the third sector in sharing knowledge and skills in a common digital space to cooperate in real world, consolidate their networks and work synergistically with public offices.

The workflow of co-production of services is a generalisation of the following service scenarios: "Collaborative services in an ageing community".

In San Donà di Piave, the applicative scenario is focused on social service having as target groups the ageing population, but also unoccupied people, low-income inhabitants, migrants, etc.

The same process could be transposed in other branch of city services, such as the ones dedicated to education, job placement, environmental services, etc.

Considering the WeGovNow platform as a tool to support an extensive engagement path, the implementation of this kind of scenario can take months, especially for the phase 3 and 5.



Phase 1: Mapping of public offices/facilities and sharing of public information about one or more sectors of city services

The Municipal staff use the platform to make available a set of public information currently fragmented in multiple portals, web pages, tools and municipal archives.

In particular:

- Representation of aggregated data of the social and economic context including demographic, economic and social information (existing in digital archives) associated to specific areas, such as neighbourhoods or suburbs.
- Mapping of public offices, facilities and services with all related information: location, opening times, target population, type of services, application procedures, etc. (For example social services for the ageing and frail population).

a) Geo-key / community maps

Set of thematic maps representing the social and economic context build on the basis of existing surveys and institutional information.

b) FirstLife

mapping of public facilities through modular and flexible cards, updated over time.

Phase 2: Collaborative monitoring and data integration (institutional and crowdsourced)

Supported by communication, engagement and training offline/online activities, citizens can contribute to the preliminary mapping of public facilities by sharing additional contents on the map objects and use the platform to request a timely intervention in critical situations.

The results could be:

- The integration of published data with feedbacks of platform users (previously trained in using the platform)

- A periodic update of official and crowdsourced information
- The creation of a dynamic archive of evolving services available for all the platform users, and not only the policy makers and administrators.
- An online and fast reporting system not focused on complaints but on request of intervention to the municipal offices in critical situations concerning, for instance, the elders and frail community.

a) FirstLife

- collection of comments, posts, additional contents in crowdsourcing
- consultation of data associated to every interval of time on the map by using spatial and temporal filters.

b) Improve my city

Collection of reports and request of intervention with different level of priority and connected to the municipal offices involved in providing social services.

These activities start as phase 2, but then they are supported by the platform in the following phases too, in order to produce an iterative assessment of available services and impact of public actions/interventions at territorial level.

Phase 3: Aggregation of information about local activities, initiatives and projects independently implemented by local stakeholders

Private institutions, professionals, business, third sector organizations use the platform

- to map their operational areas and share information about the services they provides.
- to document local initiatives, projects and events implemented in the municipality
- to represent their collaboration/territorial networks

a) FirstLife:

- group maps visible to all platform users,
- initiatives as bottom up relations among map entities,
- events collaboratively described through texts, images, links, etc.

Phase 4: Participatory process to define new targets and services

Once the platform is integrated in the city life as a tool for the municipal administration and local stakeholders, it could be used to periodically launch a public discussion on strategic goals for the local communities and joint program for social services or other cross-domain topics.

a) Liquid feedback:

- Assessment of available services and impact of previous initiatives and projects
- Online discussions about visions and strategic goals for the community, with a prioritization of related programs to implement them.
- Collaborative building of proposals within the implementation programs.
- Opinion formation about proposals and alternatives with their constraints and opportunities.

- Selection of proposals with a wide support from local stakeholders to be implemented year after year.
- Publication of results of the decision-making process.

Phase 5: Co-production of services between public and private sector

Design and implementation of new services based on the cooperation between two or more stakeholders, supported by the local administration. Create local development starting from social needs.

a) FirstLife:

- geo-referenced groups as entry point for requests, discussions, coordinate local activities among distributed operators or sub-groups
- initiatives to represent the local organizations involved in providing specific services in a project framework
- events and stories to switch from offline activities to an online collaborative storytelling of the city system in the social services domain.
- bulletin boards of groups and organizations showing activities, projects, events to which they collaborated or promoted in a selected interval of time.

b) Trusted marketplace:

- matching of users/organizations publishing requests of collaborations/resources with other users/organizations that can offer them;
- search of requests on temporal, spatial and thematic basis;
- updates on the status of request;
- collection skills/resources cards for local organizations, integrated with the organization's bulletin board.

Conclusions

From the perspective of the development of the platform we have to conclude that the requests of new features falling within the scope of the project are far less predominant than expected. On the other hand, sensibility of stakeholders and municipalities is strongly focused on the cost of introducing a new tool in their activities, and on meeting the expectation of their end users.

There are multiple reasons for those results:

- 1) Municipalities are directly responsible of the engagement of end users and of their internal staff in the project, therefore it is critical that the platform does not jeopardise their credibility nor the introduction of the platform within their procedures is welcomed as an extra burden.
- 2) The integration of consolidated civic technologies provides already consolidated methodologies, lessening the need to develop new solutions.
- 3) Building a platform from stand-alone components makes the construction of a uniform identity of the platform very costly and complicated. Those difficulties are very evident also for the municipalities and stakeholders, driving their attention toward the problem of uniformity, interoperability and identity of the platform.

As further consequence of WeGovNow architecture, the extension of consolidated components is most of the case too costly for the benefits that could be brought to the project. A wise approach is rather to scout for existing solutions or develop new modules to cover the requests of new features. This is not surprising: building a new software for a specific need is much more convenient and effective than extending a consolidated software: assessing the technical and theoretical feasibility, extending the source code, making regression testing, etc. Furthermore, WeGovNow architecture was specifically developed to support a cost-efficient integration of new components.

About functional requirements, until the platform is not going to be used in real condition is unlikely to receive specific inputs. Not being grounded on the use of the platform and of the single components, the engagement activities major contributions was about the “vision” of WeGovNow and enabling technologies which provide the opportunity to rethink stakeholders’ activities. As collateral results, we found evidence of other features which currently do not fall in the scope of the project, but that could be easily integrated in future development of the platform.

In conclusion, we expect from the piloting the platform much more specific inputs about integrations and accessory functionalities as support of the daily use, that will eventually result in minor but essential improvements of the platform in during the last year of the project, and on precious documentation for polishing WeGovNow as professional service for future exploitation.

Annex 7

Temporal indexing of Urban Entities

TEMPORAL INDEXING OF URBAN ENTITIES: BUILDING A COLLECTIVE CALENDAR OF CITY ACTIVITIES

2017-01-30

Department of Computer Science
University of Turin, Italy

In this document, we present a general framework to address the temporal dimensions of urban entities. It highlights the features and issues related to the interpretation of times and timings of entities. The analysis is followed by the definition, implementation and experimentation of a temporal indexing system, in the perspective of the building a web-based global calendar for city activities.

1 INTRODUCTION

Nothing is forever but still most of the web applications do not address dynamics. In the context of applications for cities, such as online urban communities, dynamics cannot be overlooked but should be supported as being the foundation of coordination, collaboration and planning among local actors. Urban entities are in fact tightly bounded to city dynamics: to social and cultural phenomenon and to city services and activities coexisting in the same space. In this regard, there are specific issues related times (the life cycle of entities) and timings (the operational times of entities) in the city.

Specifically, times are addressed by almost every commercial technology enabling application to support the stratification of information during time. In particular, most of the major applications for supporting activities in the city enables time queries, time views of data but timings (recurrences and times of activities) are still not supported at the city scale.

For instance, OpenStreetMap (OSM) is the most popular and effective web volunteering geographical information system (VGI), collecting contributions around the globe. Nevertheless, the representation of geographical entities is static and this represents a problem in civic social networks based on interactive maps where entities are accessible through a time window.

One dimension of time is the distinction between a place's times (when it starts and end to be) and the timing of the activities it hosts. For instance, there are stable entities providing their functions with a specific timing, such as openings and vacations, entities do not stop to be but the timing where they operate is much more relevant in the way we interact with them (is a closed store a resource?). The timing of activities has a strong impact of the conception of what and when activities can be done and therefore it is not a dimension which can be overlooked and flattened to entity existence. Moreover, there ephemeral entities which are even physically present with a certain timing in different places such as market stalls moving around the city or recurrent events such as music or religious festivals.

In this perspective, the collection of city entities is a unique collective global “calendar” defined by all local actors and, at the same time, the foundation of their activities. In this regard, in order to enable technology to represent this global calendar with the times and timings of city entities we need to address issues related to representation and querying of timings (performance wise), and to user interaction with such huge set of data. In particular, the proposed framework addresses:

1. Which are the relevant properties of temporal representation?
2. Which is the nature of the entities
3. How to extend common databases to represent and query timings directly at database level (without relying on software business logics) in a scenario of a global set of entries
4. How to support querying a global calendar of all city entities and activities by heterogeneous users

The approach proposed in this document starts from an analysis of urban entities as social artefacts. Urban entities are the results of a network of activities at local level, involving heterogeneous groups of actors. The time dimensions of urban entities are therefore interrelated with actions and services, processes at urban level as part of the common playground among local actors. From this perspective, urban entities are commons (Ostrom, 2015) shared by institutions, citizens and any other local actor, and in general involved in the orchestration of urban activities, urban planning and part of the mutual understanding of the urban space.

Thus, the entanglement of urban entities in any other aspect of city life makes the definition of a commons representation of the dynamic feature of urban entities itself a challenging issue. In fact, from a theoretical perspective, the multi-dimensionality of temporal features of entities it is not a new issue in the database research community. The best example of consolidated work is TSQL2 (Snodgrass, 2012), it is a consensus extension of SQL for temporal structures meant to address time in a very general way. On the other hand, real applications require enterprise solutions supported by most used database. Just to provide an analogy, in the field of geographical information systems libraries such as PostGIS successfully enabled developer to work with geographical entities using a widely used database such as Postgres. In particular, in the case of PostGIS what really matters were not accuracy or performances of the library but rather its accessibility to developers. On the contrary, even if TSQL2 can be considered a mature approach (1993) based on a strong theoretical framework with the backup of the major experts of the field it still just a theoretical work.

Following this lesson, this contribution does not provide a theoretical framework but a light-weighted ready-to-use fast solution which can be implemented in the most used databases and ORMs based on standard languages.

The rest of the contribution analyses the scenario highlighting challenges and existing approaches. Section 3 presents the hybridization of CRON language for job schedules used in operative systems and ICAL standard for events, its translation in database properties and query language. Section 4 shows applications of such language and discuss its limitations. Section 5 presents the experimentation results and the future extensions. Conclusions ends the document.

2 SCENARIO

Representing urban entities is commonly reduced to geographical features. This simplification leads to a static representation, a snapshot, nevertheless it is a suitable solution in most of the common applicative cases. For instance, the many geographical tools for web applications such as LeafletJS and OpenLayer used to collect geographical features do not support temporal information of any kind. Furthermore, technologies

are being updated to dynamic layers of data streams such as environmental sensors, still avoiding to address the issue of temporal features of urban entities.

Supporting collaboration, cooperation and planning requires the idea of an evolving world in which the dynamics in term of timing and coordination play a very important role in the success of initiatives. For instance, the scheduling of the occupation of a public square prevents having a strike, a concert and a sport event at the same place at the same time.

Urban entities as any social artefacts (Searle, 1995) (Ferraris, 2013) have a lifecycle. In this regard, along with the classical challenges of dynamic systems there are specific issue related to observability of such entities. In particular, non-formal entities (not formally defined by local authorities with constitutive actions) being dependent to the establishment of practices have a very fuzzy beginnings and ends. For instance, it is possible can count several important phases of a urban entity such a business centre such as the project, the first contract with a company, the beginning of the construction, the opening, but none of all can be considered the beginning of the place itself.

Furthermore, urban entities are commonly providers of services at local level. The temporal availability of service availability adds an extra temporal dimension in addressing dynamic entities, in particular in case of temporal queries. For instance, the relevance of a closed pharmacy can vary according with the use of the information, is a user looking to the commodities in our neighbourhood or we are in time of need? Moreover, past entities are not just not existing anymore but leaving a trace as impact on the perception of what can or cannot be. On the other side, planned entities which are known to be are “potential” presence on the territory affecting greatly the overall coordination. For instance, public infrastructures have a huge impact on the city landscape even before being realised such as an underground line which result on changing the expansion directives of cities, real estate market, and public/private investments years before the actual opening.

Summarising, addressing the dynamic of urban entities leads to specific issues related to the very nature of social artefacts and urban dynamics:

1. Fuzzy timings, the impossibility to establish a change of phase in an entity lifecycle
2. Worktime of urban services, urban entities are not always available or operating but they still exist
3. Traces of past activities and entities, what was matters
4. Planning the future, potential entities have their own impact even bigger than what they will actually have

Relating to the dynamics of urban entities, there are many other issues related to keep track of changes of shape and location, change of functions of urban entities, etc. All of them require the support of temporal features.

3 MODELLING URBAN ENTITIES

About the fuzzy timing, the establishment of entity phases are mostly due by institutional actions or by convention. In general, there is not a way to catch the exact beginning and the end of social artefact because the very nature of social artefacts. In fact, we can address this issue only by relying on evidences provided by users. In this regards, technology can support the emergency of social artefacts widening the collective awareness about such entities. Still, urban entities as social artefacts should be managed considering that there is not a “right” or “rightful” representation because social artefacts are just not fully objective (Ferraris, 2013).

Moreover, the city is made of an infinity of social entities which are not always shared by the most of the citizens or significant for their activities. For instance, community places are of this kind, they have special meanings and uses which are established through community practice only, such as a spot in a public park used to host the neighbourhood party. We address as urban entities only what is significant and relevant for most of the citizens. This very simple description does not lead to a computable mechanism to “catch” urban entities from a sea of information such as social media, but in our opinion makes this issue falling in the field of crowd sourcing.

Given this epistemic assumption, we consider citizen as our platform users the sensors for collecting urban entities. In order to collect urban entities from users’ contribution we establish a set of entities at the granularity we wish to catch, for instance events, groups, places.

3.1 INTERMITTENT ENTITIES

Services provided by urban entities may not be always available. This may not be true in general but urban entities are commons and the interactions with urban entities falls in the problem of commoning (the use of commons in coordination of local actors) (Bollier, 2014). Therefore, urban entities seen as common resources to be used in an orchestration require to take into consideration the specific limitations of resource availability.

In more simple words, the working time of services in the city has a major impact on our daily organization of activities. Access to the dynamic of urban entities can be approached: 1) at visualization level (opening time in google maps); 2) considering the timing of the availability of services as a relevance metric to filter urban entity.

The first approach considers the service availability as a property of the entity (e.g. opening time). This approach is implemented with is a dynamic visualization of the timing, for instance, the opening time of a store can be presented as a simple “open” or “close” according with the current date. In particular, it solves the problem of representing service timing in a static setup (snapshot) thanks to an assumption: the current time is what users are interested in, for instance see Figure 1.

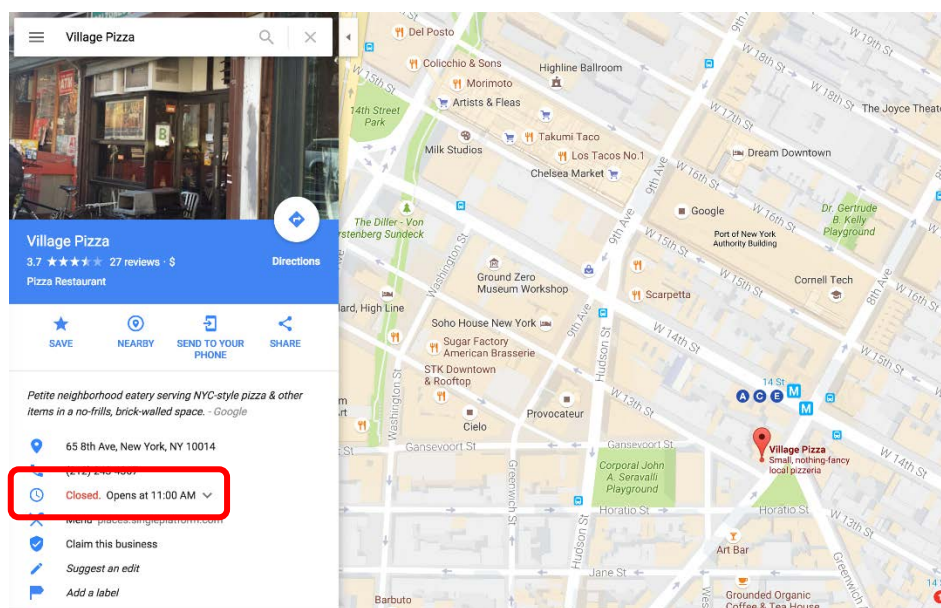


FIGURE 1. GOOGLE MAPS KEEPS THE OPENING TIME BUT THE INFORMATION IS PRESENTED AS AN EVALUATION, CONSIDERING THE USER’S CURRENT TIME.

The second approach is based on the necessity to keep the quantity of presented information low, therefore if a service is not available it should not be a “suggested” to users. In particular, this approach works along with ranking algorithm based on distance, preferences, popularity introducing one objective parameters in ranking (and filtering) results. If results are excluded by time queries, we can talk about a dynamic representation supporting time-based views.

The two approaches encode two complementary ideas: a scope and a point of view. Given a specific query, the scope (time window) can be narrow or broad accordingly to the quantity of information keeping the visualisation informative and manageable at the same time. Within a time window, the possible points of view are multiple. The choice of the point of view can be transparent to users, like in Figure 1, or can be bounded to the temporal scope of the view. For instance, routing application provide the possibility to customise a departure time and date, see Figure 2.

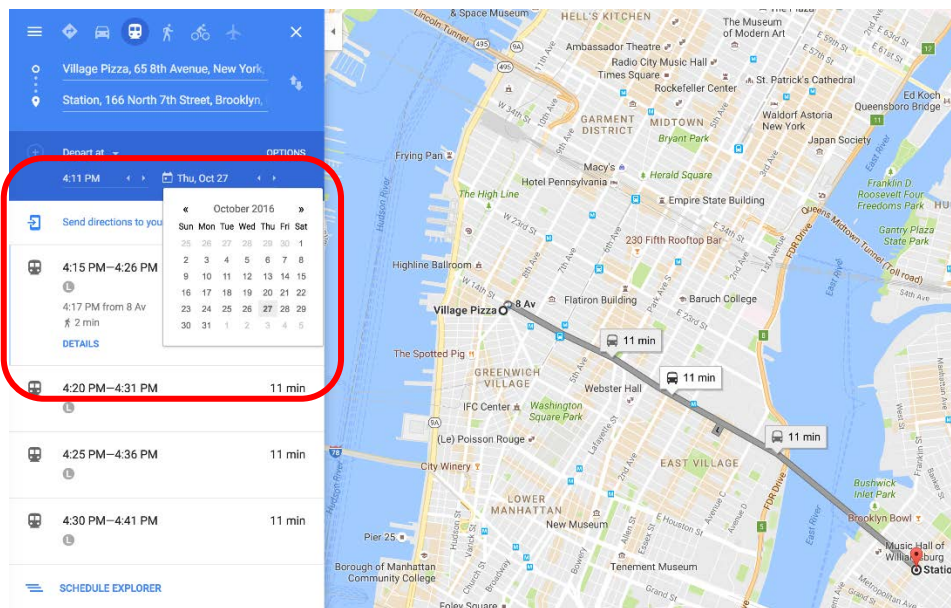


FIGURE 2. ROUTING FEATURE PROVIDE TO USER THE POSSIBILITY TO CUSTOMISE THE DEPARTING OR THE ARRIVING TIME, THIS INFORMATION IS USED FOR INSTANCE TO EVALUATE TRANSPORTATION OPTIONS.

Summarising, there are at least two aspects which need to be considered in order to evaluate temporal entities: the scope is needed to filter entities according to their valid time; the user’s temporal point of view is required in order to evaluate the timings. All applications supporting temporal features implement a theory of scope and/or temporal point of view as explicit or implicit assumptions.

3.2 EPHEMERAL ENTITIES

A new kind of complexity is given by planned or intermittent entities: ephemeral entities. Planned entities are very important in the city organization, let us consider festivals for instance which with a short duration of few days requires months of organization involving many local actors in many locations. Intermittent entities are those entities “happening” with regularity such as markets. The impact of intermittent entities involves the organization but also: the timing when provided services are available and the unavailability of other services which are replaced by the intermittent entities. For instance, street markets and parking slots are mutually exclusive.

The extreme consequence is stressed by the phenomenon of ephemeral cities: giant settlements appearing and disappearing with an interval of years in case of religious or cultural events (Kehoe, 2011). In this cases

entire cities with their system of infrastructures and services are planned, organised, implemented, and finally dismantled for existing only for a short period of time but influencing an area for years.

The representation of ephemeral entities can be done with digital maps. In particular, digital maps can support planning and documentation of ephemeral entities if including a temporal support. Digital maps can work as exploration tools for what was in the past and what will or can be in the future.

What matters specifically in case of ephemeral entities are the connections among local activities and areas as complement of planned urban entities. For instance, the difference between a street market stall and common grocery store is not the timing of provided service but the orchestration of activities before and after required to setup the service such as setup the market stalls and clearing the market area after, in addition to the mutual exclusivity use of the same area with other services.

Ephemeral entities are an extreme case of urban entities: some urban entities are planned, other are incidental but all of them have a live cycle connected to urban activities. Since urban entities are the results of initiatives, and can trigger many other initiatives at different timing, in order to collect effectively urban entities, it is required to extend the efforts to the related activities. In other words, the granularity of entities in an urban calendar should be fine grained at level of activities.

3.3 ASSUMPTIONS ON TIMES AND TIMINGS

Time is one but the temporal features can be many: the duration of an event, the time when the event was planned, the time when the event become public. In addressing time, there are issue related to time semantic, how we should evaluate the things we know about an entity if we look at it from a different time or if we consider at a different scale. In the context of urban entities, which are the relevant temporal features? What are the relevant questions about the duration and the repetitions of urban entities?

The concept of timing in real applications, time repetitions, is bounded to the concept of time granularity: time deltas between repetitions. Time granularity is a very complex field: theoretically is the study of relations between class of time intervals. A calendar is a specific setup of interval classes and relations. For instance, we can count many “official” calendars across the world such as the western calendar or the Chinese calendar), school calendars, work calendars with differences in term of durations (work week of 5 rather than 7 days), general alignment (beginning of the year), or specific granularity levels (“semesters” of 4 months in school calendars).

Time granularity is being already addressed (Snodgrass, 2012) making possible to define new calendars and mapping between calendars. In a context of urban activities everyone refers to the national plus some common notions of festivities and work time (weekly holidays). In this context, a general framework is not strictly required instead a pragmatic approach can be implemented with enterprise technology rather requiring the implementation of theoretical a language such as TSLQ2.

A pragmatic approach does not save us from the problem of time semantics, in particular when comes to representing information at different granularity. In fact, timed information semantically dependent to the granularity of their time description. For instance, a salary amount can be per month, week or year. In any case, a change of time granularity (in information query) may or may not affect the outcome value.

This phenomenon is called *telicity* and it is mostly applied in studying the semantics of verbs and verbal phrases. In temporal database, *telicity* influences the transformation record values according with the change of time granularity (Khatri, Snodgrass, & Terenziani, 2009). The semantics of time queries is dependent to *telicity* of entities. In our context, the “value” that we are referring to is the validity of a result is a time query.

For instance, an opening time of from 8:00 to 16:00 from Monday to Friday defines the availability of a service in term of day of the week and hours. The timing is given by the combination of two expressions defined at different granularity: day and hours. Considering a query about the opening time on Monday and on Sunday, in the latter case we can safely say that it our shop is closed but in case of Monday we cannot be certain since during the day there will be both open and close hours. In fact, the granularity of the combination of the two timing expressions is hours: a shop is opened for 8 hours each 16 hours with two exemptions each 5 openings. On the other hand, in case of a query for a specific minute such as Monday at 8:59 we don't have any problem scaling down the granularity.

This example shows how tricky timing in term of query resolution, even trivial case may rise unexpected complexity. Moreover, it highlights that the telicity has an impact of properties of timed entities. In particular in case of the previous example, we see different semantics according with store being liquid (Bettini, Wang, & Jajodia, 1998) or not in respect of its opening time. Liquidity is the property of preserving value changing the scale upward and or downward. In the previous example was easy to see as opening time have downward but not upward liquidity.

This issue cannot be solved with a data schema, since schemas do not represent semantic of information cannot be solved with a priori logic if we do not fix the domain of application and certainly we cannot ask users to define semantic of time and timing features (Bettini, Wang, & Jajodia, 1998). Furthermore, in case of punctual time information (timestamps) we may have other properties such as persistence: an entity lasts unless something changes.

In our context, the conservative position is assuming telicity for each entity and timing expression, in other words we are not sure when we may assume upward or downward liquidity or persistence: scaling entities upward or downward or persisting entities in case of punctual time information (now is open) in the future may mislead users introducing false positives.

Summarising, a pragmatic stand oriented to real-life generalist applications should consider the issue of telicity or the lack of information carefully, designing a coherent set of assumptions about persistence, upward and downward liquidity and other possible point-based or interval-based semantics.

In this regards, our approach is to exploit users' input (on a timeline) about the relevant time window (and granularity). In fact, since the biggest issue is filtering information, we assume avoid false positive presenting only the entities whose timing descriptions are defined in the input granularity. This may rise in hiding important information but since the timeline is interactive, users can still access to information by switching the time granularity.

3.4 TIME FEATURES AND TIME-BASED TECHNOLOGIES

Entities may have multiple time dimensions. For instance, the classical dimensions in databases are between transaction time and valid time: the first is the time when an information becomes available in a database, the latter is the time in which the information is true. Transaction time is mostly punctual and it assume the persistence of knowledge: knowledge is preserved until it changes. Valid time are commonly expressed by a couple of dates (start and end dates). The semantic of valid time is usually implemented by business logic field by field. In commercial databases up to today there is no generic support for time repetitions. In few rare cases it is possible to express a time schedule: repetition of punctual times and not of time intervals.

Even so, there are few very handy web applications supporting times and timings such as web calendars or events platforms. In those cases, the solutions are protected by their companies. What we can know is that the few open source solutions rely on business logics and data visualization in a context of very limited

quantity of information (a user's events on a calendar are not much). In particular, the visualisation of web calendar comes in help highlighting a specific portion of time. This enables many possible heuristics such as dereferencing periodicity in a chain of single events in a reasonable wide time window.

Considering the goal of building a global calendar of millions of events in a territory such as a city, we seek a production-ready solutions meant to support numbers in real case scenarios. The most used formalism to express recurrences in software and database are the CRON expressions, regular expressions meant to be fasted computed with low resources by a job scheduler.

The complexity of cases in real life requires an expressivity much deeper than what may be requested by technical software such as job schedulers. In this sense, ICalendar standard represent the consensus about a common language for calendar, defining entities, recurrence expressions, exception, alerts, time zones, durations and other specific features of real life applications.

3.5 CRON EXPRESSIONS

CRON is a family of time-based job schedulers used in Unix operating systems. CRON jobs are a set of CRON expressions representing when a task should be triggered. The timing is expressed by a regular expression (Wikipedia, n.d.) composed of 5 or more substrings defining a set of time points at each time granularity.

```
# _____ min (0 - 59)
# | _____ hour (0 - 23)
# | | _____ day of month (1 - 31)
# | | | _____ month (1 - 12)
# | | | | _____ day of week (0 - 6) (0 to 6 are
# | | | | | Sunday to Saturday, or use names)
# | | | | |
# | | | | |
# | | | | |
# * * * * * command to execute
```

For instance, a command that should be executed each day at 3 a.m., such as a database dump can be expressed as follows:

```
0 3 * * * # every day at 3:00
```

A job that should be executed once a week (every Saturday at 3:00) such check-up of the file system can be expressed as:

```
0 3 * * 6 # every Saturday at 3:00
```

A CRON expressions can represent multiple recurrences, for instance at 3 a.m. and 1 p.m. during workdays on odd months:

```
0 3,13 * 1/2 1-5      # from Monday to Friday at 3:00
                        # and at 13:00 of Jan, Mar, May,
                        # Jul, Sep, Nov
```

In particular, the syntax allow special symbols (*, / -) representing respectively: no restriction or always (*), concatenation (3,13 at 3h and 13h), increment of a number n (e.g. /2 every 2 occurrences) and interval (i.e 1-5 from Monday to Friday).

Notice that the third and last field are both at the same granularity since week days are days of the month too, therefore they can be used together but in case of overlapping the result is not to be duplicated but the union of the two sets. For instance, considering * * 1/2 * 1-5 says each 2 days starting from the first day of the month and every day from Monday to Friday, the result will be every day from Monday to Friday plus all odd Saturdays and Sundays. If one of the two is defined, then the other is ignored.

There exist many versions of CRON expressions extending the semantic. For instance, we considered Quartz CRON (Quartz) (Software AG, n.d.) which is a Java implementation of CRON. Quartz introduces two extra fields: seconds and years and more symbols (L # W ?). In particular a Quartz expression can represent schedules like:

```
0 15 10 ? * 6#3      # at 10:15 on the third Friday of
                        # every month
0 15 10 L * ?        # at 10:15 of the last day of
                        # every month
0 15 10 * * ? 2016    # at 10:15 of everyday of 2016
```

Quartz expressivity does not come free, in particular:

- L , $\#$ and W symbols standing cannot be resolved without providing a specific month and year
- $?$ symbol is used to disable “day of the week” or “day of the month” alternatively. Otherwise in Quartz the use of both at the same time may lead to unwanted results.

Quartz notations can actually catch many scenarios we may see in real life. But in order to evaluate a Quartz expression it is required a specific time (a month or a year) because we can’t say which what is the last day or the third Thursday of a month in general, but we need to point a month and year (e.g. May 2016). A Quartz expression cannot be evaluated by its definition but it requires to be resolved with its extensive representation.

In general, CRON expressions are limited to point-based time information, in other words it is not possible to represent durations in any of CRON variants. For instance, it is possible to represent the opening time but not the duration (e.g. from 8:00 until 16:00, 8 hours). Moreover, CRON expressions do not support exceptions such as not during Christmas holidays (from 25th of Dec to 6th of Jan).

In order to represent durations two CRON expressions are required, one for the starting time and one for the ending time. This semantic cannot be represented straightforward by a database schema and it requires a storage procedure to define how to combine the two records.

3.6 ICALENDAR STANDARD

ICalendar is a file format and standard notation for representing events (Oracle Corporation, n.d.). ICalendar is adopted as intercommunication language among mostly enterprise products and web services such as Google Calendar, iCal, IBM Lotus Notes, Microsoft Outlook, Mozilla Thunderbird, etc.

ICalendar define several entities definitions: VCALENDAR (calendar), and entities VEVENT (events), VTODO (tasks), and VJOURNAL (journal entries). ICalendar is relevant because it takes into account a decade of experience of the web (1998 updated in the 2009 and still improved) and industrial requirements of the major software developer.

Concerning the issue of time and timing we can specifically focus on RRULE, the attribute of ICalendar describing repetitions. RRULE value (RECUR) can express repetitions of time intervals and exceptions too. Still, RECUR expressions are much more expressive but there are CRON expressions which cannot be encoded as RECUR, for instance:

```
0 0/2 * 1/3 *          # at 00:00 every two hours
                        # every day of each three months
                        # starting in Jan
```

While CRON expressions are designed to be implemented in job scheduler, ICalendar does not look so, in particular there are not standard libraries and the specifics are only partially implemented by the supporting software, making ICalendar mostly an interexchange format rather than an operational language.

4 A DATABASE-ORIENTED SOLUTION FOR TIMES AND TIMINGS

A real world scenario of millions of events suggests the necessity of a solution comparable to common information storage and retrieval in database. Specifically, we seek to represent time and timing as records and be able to query such records in real time, fast and without extra burden for the DBMS and without relying on post processing results crossing field values: we wish to store time and timing as records and be able to query using the same expressivity using relational selection only.

In order to archive this result, we require a compromise in terms of expressivity of the language for times and timings. On step of our approach is to decompose complex expressions beforehand in order to keep the querying simple. The second is to use the same language in both representing and querying time and timing expressions, in order to implement queries with a simple selection operator.

4.1 FROM POINT-BASED CRON EXPRESSIONS TO INTERVAL-BASED CRON MASKS

From a computational perspective, the most convenient starting point is standard CRON expressions, but: can we actually express intervals with CRON masks? As argued previously, CRON expressions are not meant to express intervals, but repetitions of point-base times.

Considering the following CRON expression:

0 8-16 * * 1-5

It does not express “from 8:00 to 16:00 from Monday to Friday” but a set of points in time “every hour at 00 minutes from 8 to 16, from Monday to Friday”. In particular, CRON expressions represent quantized time points in 5 granularities, sections of a CRON expression. Following the previous example, we have at each minute 0 of each hour between 8 and 16 included, each day between Monday and Friday included: $1 \times 8 \times 5 = 40$ moments each week.

The intervals between two moments are given by the granularity of the section. Following, from moments it is possible to infer intervals, for instance 0 minutes is equivalent to the set [0,1) because it is not possible to express a moment between 0 and 1 minutes. As consequence we can consider quantized intervals rather than just moments. Following, the same example we have 1 minute each hour between 8 to 16 each day from Monday to Friday, $1 \times 8 \times 5 = 40$ minutes each week.

In this sense, the * symbol indicates the full interval (60 minutes, 24 hours, one month, one week, one year) according with the position, and the semantics is given by the intersection between the 5 sections.

CRON expressions can converted in 5-tuples of bit masks which are widely supported in commercial databases. The conversion of CRON expressions in bit masks is trivial since standard CRON are regular expressions and each section of a CRON expression indicates a set of a finite domain (minutes, hours, etc.). Following an example:

```
# ┌────────── minutes (1...0 1th bit at 1 of 60 bits)
# │ ┌────────── hour (0010...0 - third bit at 1 of 24 bits)
# │ │ ┌────────── day of month (all 0 of 31 bits)
# │ │ │ ┌────────── month (1010... - odd bits at 1 of 12 bits)
# │ │ │ │ ┌────────── day of week (0...10 6th bit at 1 of 7 bits)
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ │
# │ │ │ │ │
# 0 3 * 0/2 6          #every Saturday at 3:00 of odd months
# m h d M w
```

The * symbol is encoded as a set of 0 as special case. Now, we need a way to check the matching between a set of CRON expressions and a query (also a CRON expression). We encoded CRON expressions as a set of 5 bit masks (CRON), using a bit for each. It is possible to query a table of CRON (CRON table) using bitwise operators, given a CRON as query. Given a query CRON $q = (q_m, q_h, q_d, q_M, q_w)$ and a generic CRON $g = (g_m, g_h, g_d, g_M, g_w)$, the comparison operator O is defined as follows:

$$q \text{ O } g = \&\&(!q_i \mid \mid !g_i \mid \mid q_i \& g_i)$$

With $\&\&$ indicating the conjunction of the disjunction ($\mid \mid$) of the check between q and g columns, and $\&$ the bitwise and. In words, a CRON matches iff the conjunction of the check of each column is true. Each column is matched with the corresponding query section with a disjunction of the following conditions: the CRON has *, the query has * or the bitwise and between the two bitmasks is true (the intervals overlaps). Following some examples with generic bitmasks:

```
q = 00101 00000 10101 00101 11111      # query

c1 = 00000 10001 00000 00001 00000
    true  true  true  true  true          # q O c1 = true

c2 = 00000 00000 11111 00000 00000
    true  true  true  true  true          # q O c2 = true

c3 = 00000 00000 01000 00000 00000
    true  true  false true  true          # q O c3 = false

c4 = 11111 01100 01000 11000 00110
    true  true  false false true          # q O c4 = false
```

The column match is true iff one or both masks are 0 or they have at least one bit at 1 in the same position (e.g. $c4$ do not share any bit at 1 with 1 in the 4th column). In words, O is an extension of Allen's overlap relations o between intervals to sets of intervals, it is true iff there are overlaps between all sets of intervals at each granularity. see the following example.

```
----      -----      # query q
--  --      -----      # c1 O q = true
--              --      # c2 O q = false
              -----      # c3 O q = true
```

Why is it possible to query CRON without explicit dates? A CRON expression is an intentional definition of a class of intervals but a CRON mask is an extensional definition of a class of intervals. Encoding CRON expressions in bit masks makes explicit a set description of the class of intervals defined by CRONs enabling querying operations. A CRON is still a class definition, without casting a CRON within an explicit time we can't

calculate for instance all intervals. In particular, we don't have a fixed number of days or Mondays in a month, but until we can express a query with a CRON it is possible to avoid casting CRON.

As previously described, CRON expressions are not meant for representing nothing more than job schedule; we need to extend their CRON expressivity in order to partially cover ICalendar cases.

4.2 EXTENDING CRON EXPRESSIVITY

Confronting ICalendar with the software supporting it, it is easy to see that not all possible scenarios are actually implemented and used but there are some basic features which can be found in all calendar applications:

1. Date constraints
2. Granularity
3. Exceptions
4. Occurrences
5. Cardinal recurrences

In order to extend the expressivity of CRON we had to enrich the notation introducing new fields in the CRON table. On the other hand, some notations can be actually addressed simply by encoding the CRON as bit masks.

4.3 DATE CONSTRAINTS

CRON as defined do not address times but only timings, there is not a way (in standard CRON) to express a valid time as explicit dates. Moreover, CRON extensions like Quartz support years and seconds, consenting the representation of date constraints but it did not consider year field in CRON. The reason is that regardless of the specific field a CRON needs a boundary in order to be encoded as bit masks: an open set cannot be converted in a fix length bit mask. In order to add a column for years we need to define an upper granularity such as decades, century or millennia. Summarising, we can only push the problem but in the end we still need to handle an open set to fix a constraint.

In order to constrain recurrences is sufficient to introduce a valid from and valid to fields. It is a standard solution in database schema which does not affect performance, in particular it can be a very handy solution to pre filter records and setup a distributed architecture.

Moreover, the introduction of date constraints may solve some issues related to real life cases, for instance *"18:00 of Monday to 8:00 of Friday"* can be easily solved this way. It does not look like but this simple example cannot be represented with a single CRON because hours constraints should not be applied to each day of the interval but respectively 18:00 to Monday (18:00 – 23:59) and 8:00 to Friday (0:00 – 8:00). The related CRON will look like the following:

```
* 18-8 * * 1-5      # from 18:00 of Monday to 8:00 of Friday?
```

This expression makes no sense because of the 18-8 interval, but even without this issue it is still not representable. For instance, we can consider the following revised version

```
* 8-18 * * 1-5      # from 8:00 of Monday to 18:00 of Friday?
```

The semantic of this CRON is not what we expect, in fact the right interpretation is the following: from 8 to 18 (10 hours) each day from Monday to Friday. This example requires 3 standard CRON to be expressed:

```
* 18-0 * * 1      # from 8:00 to 23:59 of Monday
*   * * * 2-4     # from Tuesday to Thursday
* 0-8 * * 5       # from 0:00 to 8:00 of Friday
```

With the introduction of the valid from and valid to we can simply represent the previous example as follows:

```
* * * * 1-5 (2016/10/17@18:00) (2016/10/21@8:00)
```

Still, there is a difference between the two solutions, the first one does not require an explicit date, in the second case we use a date constrain to bind a generic interval description. In the second case the CRON semantics is wrong but thanks to the validity time the result is as expected.

Validity time can be optional (null by default) as it is implemented in some applications like Google Calendar. Since we do not need to cast all possible occurrences it does not endanger performances or rise memory issues.

4.4 GRANULARITY

CRON can represent quantized intervals only, therefore in case of intermediate intervals the only solution is approximation. For instance, if we encode the timing “8:30 – 12:00, 14:00 – 18:30” using the comma operator we have the same issue of previous examples because we can’t represent intervals of half hour:

```
0-30 8-12,14-18 * * *      # 8:00 – 8:30, 9:00 – 9:30...
```

In general, the maximum approximation is the granularity itself, in case of the previous example we should drop the constraints in the minute section and approximate the ending time to next hour:

```
* 8-12,14-19 * * * # 8:00 – 12:00, 14:00 – 19:00
```

In order to use CRON an indexing system the approximations may not be a problem, but in some scenarios we may require extra sensibility. In this regard, there are two possible strategies:

1. Introducing an intermediate granularity such as day sections (“*morning*”, “*afternoon*”, “*evening*”, “*night*”) and the relative column
2. Combining two granularities in one single column, for instance minutes and hours can be replaced both with half-hours (48 bits)

The two strategies can be combined in order to keep timing representations short. Considering the half-hour granularity, the previous example can be represented because the quantized intervals become of 30 minutes.

4.5 EXCEPTIONS

A very common case are recurrences with exceptions, in particular with generic exceptions (not just date restrictions), such as holydays. It is interesting to notice that those kinds of exceptions can be expressed as CRON, for instance:

```
* * 25-26 12 *           # from 25 to 26 of December
                           # Christmas holidays
```

The solution we developed introduces the notation of negative CRON, an extra field indicating if the CRON should include or exclude the related event.

```
false * * 25-26 12 *     # Christmas holidays
true  * 8-18 * * 1-6      # Openings 8-18 Mon-Sat
```

In order to use positive and negative CRON expressions it is required to define how to handle the cases where both are included in the results (for instance Monday 25 Dec at 12:00). A simple semantics can include an event if the conjunction of the control flags of the returned CRON is true.

4.6 OCCURRENCES

In commercial applications is common to possibly define an occurrence number as limit for repetitions. For instance, each two days for 10 times. This can be handy in case of a fixed set of events scheduled at the same time.

CRON does not support counters but counters are syntactic sugar, explicating the occurrences and indicating an end date is equivalent. Therefore, in this case it is possible to support this feature through a pre-processing phase, which it does not increase the overall complexity of the system.

4.7 CARDINAL RECURRENCES

Some recurrences are defined by cardinality, for instance *“each second Sunday of the month”*. Those cases are supported by ICalendar and Quartz but not by standard CRON. In particular, cardinal recurrences are necessary in order to connect days of the week and months, because there is no way to know the exact day of the month in general. In other cases, this notation is syntactic sugar, therefore it can be addressed via pre-processing inputs.

In order to address this feature, it is required a new column for the cardinality of day of the week (5 bits because 5 is maximum of occurrences of a day of the week in a month). Given the previous example in Quartz notation (# indicates the cardinality):

```
* * * * 0#2          # second Sunday of each month
0 0 0 0 1000000 01000    # equivalent bit masks
```

Extending the day of the week with cardinality stresses the use of `*` symbol in queries. In particular we can consider the following query:

```
0 0 0 0 1000000 00000    # q = Sunday

0 0 0 0 1000000 01000    # c1 = second Sunday of the month
t t t t true      true    # q O c1 = true
```

The problem seems to be the lousy semantic of `O` but with the same operator we can express two different queries: “*all Sundays*” and “*some Sundays*”. In order to explicit the “*all*” quantifier the query should be implemented as follows:

```
0 0 0 0 1000000 00000    # q1 = some Sundays
0 0 0 0 1000000 11111    # q1 = all Sundays

0 0 0 0 1000000 01000    # c1 = second Sunday of the month
t t t t true true        # q1 O c1 = true
t t t t true false       # q2 O c1 = false
```

In particular the use of the symbol `*` in a query or in entries causes to skip the check, but the full set mask indicates the full interval as requirement.

4.8 LIMITATIONS

There are cases we cannot address with CRON without generating false positive or false negative results. In particular, we cannot address:

- The last operator “*L*” and the negative cardinals (e.g. -1 is equivalent to L, -2 means second to last, etc.)
- The weekday operator “*W*” indicating the closest weekday to a given day of the month

Those operators require to evaluate expressions in a given date. Specifically, we do not know how many Mondays there are in a month (4 or 5?) therefore, we may cannot encode the last or negative cardinals in a bit mask. Moreover, the weekday requires also to know the positions on weeks in a month in order to calculate the proximity.

They may exist particular applicative scenarios strongly requiring those operators but among the examples we collected so far we did not find a case which cannot be converted in the supported notation yet.

5 EXPERIMENTS

Since this contribution wishes to address real life applications, in order to stress the expressivity of the system, we collected a dataset of real timings from Porta Palazzo market square in the city of Turin, Italy. Encoding the dataset, we exposed limitations we addressed in the previous section. Following, we stressed the system performance with random generated entries and queries.

5.1 CORRECTNESS

We collected so far 233 timing descriptions over a total of a hundred of market stalls. Most of them where very simple but in some cases we found exceptions, seasonal timings, cardinal occurrences and alternate days.

A sampling the most representative cases:

7:00 - 14:00 Mon - Fri

7:00 - 19:30 Sat

Common opening time.

8:00-13:00, 15:30-19:30 Tue - Fri

7:00 - 19:30 Sat

Closing at lunch time.

14:00-19:00 Sat from 1st Aug to 15th Oct

This is an exception to the normal opening during summer time.

8:30 - 13:00, 15:30 - 19:30 Mon-Tue, Thu-Fri

8:30 - 13:00 Wed

8:30 - 19:30 Sat

Half day opening on Wednesday.

9:00 - 13:00, 15:30 - 19:15 Tue - Fri

Closing at 19:15 requires approximation to 19:30.

9:00 - 12:30, 15:30 - 19:15 Mon-Tue, Thu-Sat

Closed on Wednesday and approximation of the closing time.

8:30 - 13:00, 15:00 - 19:00 Mon-Fri, Sat/2

Opening on Saturday only each two weeks.

5.2 PERFORMANCES

Tests were run on a document-based database (MongoDB) over more than one million random generated entries. The tests were conducted increasing the number of queries by tenfold recording total, maximum and average time.

1.023.944 ENTRIES			
NUM. OF RUNS	Max time (ms)	Total time (ms)	Average time (ms)
1	2	2	2
10	2	4	0,4
100	2	8	0,08
1.000	3	44	0,044
10.000	3	265	0,0265
100.000	3	2380	0,0238
1.000.000	3	21675	0,021675
10.000.000	5	212000	0,0212

TABLE 1. RANDOM QUERIES EXECUTED OVER A DATABASE OF 1.023.944 ENTRIES, THE AVERAGE TIME IS CLOSE TO ZERO (LESS THAN 0,1 MS).

Following, we repeated tests but firing sets of 10, 100, 1000 and 10000 concurrent queries.

1.023.944 ENTRIES – BATCH OF 10 CONCURRENT QUERIES			
NUM. OF RUNS	Max time (ms)	Total time (ms)	Average time (ms)
1	2	78	7,8
10	2	76	0,76
100	3	333	0,333
1.000	2	1766	0,1766
10.000	2	13799	0,13799
100.000	2	130416	0,130416

1.023.944 ENTRIES – BATCH OF 100 CONCURRENT QUERIES			
NUM. OF RUNS	Max time (ms)	Total time (ms)	Average time (ms)
1	2	3582	35,82
10	2	5919	5,919
100	2	19524	1,9524
1.000	2	127193	1,27193
10.000	2	1203622	1,203622
100.000	6	12437346	1,2437346

1.023.944 ENTRIES – BATCH OF 1.000 CONCURRENT QUERIES			
NUM. OF RUNS	Max time (ms)	Total time (ms)	Average time (ms)
1	2	118111	118,111
10	2	317724	31,7724
100	3	1442373	14,42373
1.000	5	12295172	12,295172

10.000	5	115261069	11,5261069
--------	---	-----------	------------

1.023.944 ENTRIES – BATCH OF 10.000 CONCURRENT QUERIES

NUM. OF RUNS	Max time (ms)	Total time (ms)	Average time (ms)
1	5	16643359	1664,3359
10	3	26489348	26,489348
100	4	137238709	137,238709
1.000	5	1350002410	135,000241

TABLE 2. RESULTS OF THE EXPERIMENTS WITH BATCH OF CONCURRENT QUERIES, WE CAN SAFELY STATE THAT IN THE WORSE SCENARIO WITH 10.000 QUERIES WE HAVE WITH MULTIPLE AN AVERAGE TIME OF LESS THAN 0,2 SECONDS.

Tests were performed on a VM with 2 cores and 2 gb of RAM only without any database tuning. The overall result is in worst scenario less than 0,2 seconds in case of 10.000 concurrent queries. As far as we know, there are no available results for comparison but in term of web services and considering a common workload of 100 concurrent queries the time is irrelevant (less than 0,2 milliseconds) considering an average time of 70 ms for very fast cached calls such as Google Maps's API.

6 CONCLUSIONS AND FUTURE APPLICATIONS

This document addresses the technical issues of representing times and timings of urban entities in order to build a collective calendar of city activities. In particular, the premises of the proposed solution are: an applicative scenario in which there is not a user-based partition of entries such as in calendar applications; an indexing system to compute the relevant entries should not require business logic to run at query time; the solution should be able to represent real cases; the solution performances with a common environment setup should not hindrance the overall performances.

The developed solution is based on the most common formalism for timings, CRON expression, and integrated in order to support time intervals, valid time and the most common features supported in the standard format for calendar for software applications ICalendar.

Along with the discussion of the specific issues of representing timing according with the premises, we introduced a set of solutions, workaround and limitations in the scope of the presented solution. Despite limitations, the solution was proved to be able to represent real cases of timing collected from real commercial activities in a dynamic environment such as market squares.

Following, through a set of experiments with random entries and random queries we tested the solution in a low capacity environment such as a common web server. The experiments where organised as cycles of batch queries fired concurrently. In a common scenario with 100 concurrent access the queries where solved in a fraction of millisecond. Under heavy requests (10.000 concurrent queries) the system archived a maximum an average time of 0,135 seconds without any caching system or database tuning.

In conclusion, the presented solution can be used to represent and retrieve high number of entries with a common database without relying on post processing results. The costs in term of expressivity and approximation compared with the real world examples are shown to be very reasonable. Moreover, it was

discussed a set of possible workaround in order to extend the expressivity or compress the representation according with the application demands.

7 REFERENCES

- Bettini, C., Wang, X. S., & Jajodia, S. (1998). Temporal semantic assumptions and their use in databases. *IEEE Transactions on Knowledge and Data Engineering*, 10(2), p. 277-296.
- Bollier, D. (2014). *Think Like a Commoner*. Gabriola Island, Canada: New Society Publishers.
- Ferraris, M. (2013). *Documentality: Why it is necessary to leave traces*. Fordham University Press.
- Kehoe, K. L. (2011). *Burning Man was better next year: "a phenomenology of community identity in the Black Rock counterculture*. California State University, Communication Studies. Sacramento: Diss.
- Khatri, V., Snodgrass, R. T., & Terenziani, P. (2009). Telic distinction in temporal databases. In *Encyclopedia of database systems* (p. 2911-2914). Springer US.
- Oracle Corporation. (s.d.). *ICalendar*. Tratto da ietf.org: <https://tools.ietf.org/html/rfc5545>
- Ostrom, E. (2015). *Governing the commons*. Cambridge university press.
- Searle, J. R. (1995). *The construction of social reality*. Simon and Schuster.
- Snodgrass, R. T. (2012). *The TSQL2 temporal query language* (Springer Science & Business Media ed., Vol. 330). Springer.
- Software AG. (s.d.). Tratto il giorno 2016 da Quartz Job Scheduler: <http://www.quartz-scheduler.org/>
- Wikipedia. (s.d.). *Cron*. Tratto il giorno November 2016 da Wikipedia: <https://en.wikipedia.org/wiki/Cron>

Annex 8

WeGovNow Environments Cheat Sheet



Cheatsheet Environments

Dev - dev.wegovnow.eu

FirstLife - wegovnow.firstlife.org
Improve My City - wegovnow.infalia.com
Liquid Feedback - wegovnow.liquidfeedback.com
Community Maps - wegovnow-cm.geokey.org.uk
Trusted Marketplace - tmp.infalia.com
Ontomap - dev.api.ontomap.eu
Inputmap - inputmap.firstlife.org
Area Viewer - areaviewer.firstlife.org
Logger Proxy - loggerproxy.firstlife.org
Tile Server - tiles.firstlife.org

Pt1 - pt1.wegovnow.eu

FirstLife - pt1.wegovnow.firstlife.org
Improve My City - pt1-imc.infalia.com
Liquid Feedback - wegovnow-pt1.liquidfeedback.com
Community Maps - wegovnow-cm.geokey.org.uk
Inputmap - inputmap-pt1.firstlife.org
Tile Server - tiles.firstlife.org

Pt2 - pt2.wegovnow.eu

FirstLife - pt2.wegovnow.firstlife.org
Improve My City - pt2-imc.infalia.com
Liquid Feedback - wegovnow-pt2.liquidfeedback.com
Trusted Marketplace - pt2-tmp.infalia.com
Ontomap - p2.api.ontomap.eu
Inputmap - inputmap-pt2.firstlife.org
Logger Proxy - loggerproxy-pt2.firstlife.org
Tile Server - tiles.firstlife.org

Pt3 - pt3.wegovnow.eu

FirstLife - pt3.wegovnow.firstlife.org
Improve My City - pt3-imc.infalia.com
Liquid Feedback - wegovnow-pt3.liquidfeedback.com
Trusted Marketplace - pt3-tmp.infalia.com
Ontomap - p3.api.ontomap.eu
Inputmap - pt3.inputmap.firstlife.org
Area Viewer - pt3.areaviewer.firstlife.org
Logger Proxy - loggerproxy-pt3.firstlife.org
Tile Server - tiles.firstlife.org

San Donà di Piave - sandona.wegovnow.eu

FirstLife - sandona.wegovnow.firstlife.org
Improve My City - sandona-imc.infalia.com
Liquid Feedback - sandona.liquidfeedback.net
Trusted Marketplace - sandona-tmp.infalia.com
Ontomap - sandona.api.ontomap.eu
Area Viewer - sandona.areaviewer.firstlife.org
Logger Proxy - loggerproxy-sandona.firstlife.org
Tile Server - tiles.firstlife.org

Southwark - southwark.wegovnow.eu

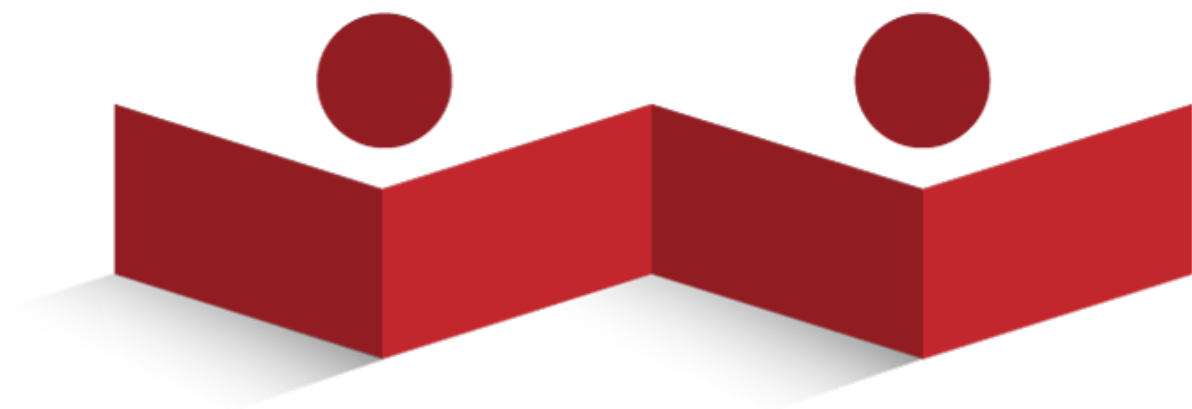
FirstLife - southwark.wegovnow.firstlife.org
Improve My City - southwark-imc.infalia.com
Liquid Feedback - southwark.liquidfeedback.net
Trusted Marketplace - southwark-tmp.infalia.com
Ontomap - southwark.api.ontomap.eu
Area Viewer - southwark.areaviewer.firstlife.org
Logger Proxy - loggerproxy-southwark.firstlife.org
Tile Server - tiles.firstlife.org

Torino - torino.wegovnow.eu

FirstLife - torino.wegovnow.firstlife.org
Improve My City - torino-imc.infalia.com
Liquid Feedback - torino.liquidfeedback.net
Trusted Marketplace - torino-tmp.infalia.com
Ontomap - torino.api.ontomap.eu
Area Viewer - torino.areaviewer.firstlife.org
Logger Proxy - loggerproxy-torino.firstlife.org
Tile Server - tiles.firstlife.org

Annex 9

WeGovNow Pilot Vademecum Model



WeGovNow

San Donà di Piave Information Manual

Pattern 1

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referent: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Name Surname

phone number: 011/00000000

e-mail: name@example.org

Pattern 2



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referent: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Name Surname

phone number: **011/00000000**

e-mail: **name@example.org**

Pattern 3

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referent: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Name Surname

phone number: **011/00000000**

e-mail: **name@example.org**



FirstLife

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Developers: "Lorem ipsum dolor sit amet, consectetur adipiscing

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Problem Solving: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

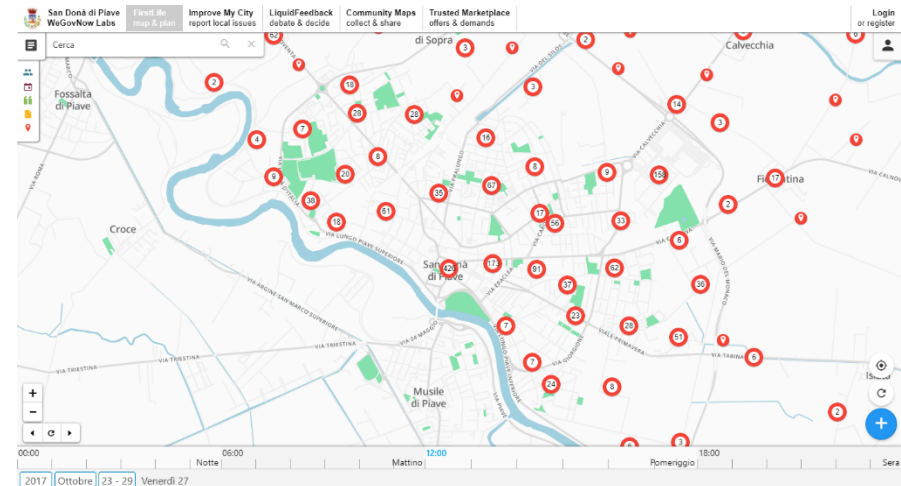
Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Tutorial and Guide:

tutorial.example.org



Improve My City

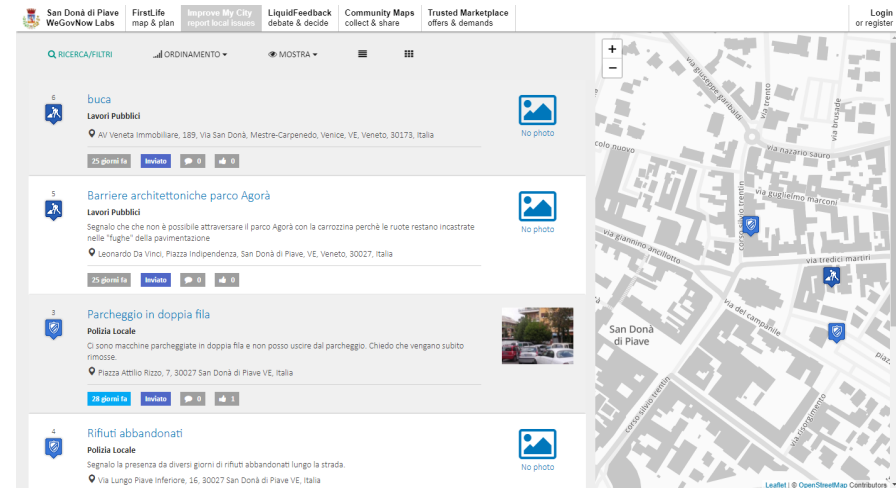
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Developers: "Lorem ipsum dolor sit amet, consectetur adipiscing

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**



Problem Solving: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Tutorial and Guide:

tutorial.example.org

Liquid Feedback



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut aute irure dolor in reprehenderit in voluptate velit esse cillum nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Developers: "Lorem ipsum dolor sit amet, consectetur adipiscing

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Problem Solving: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

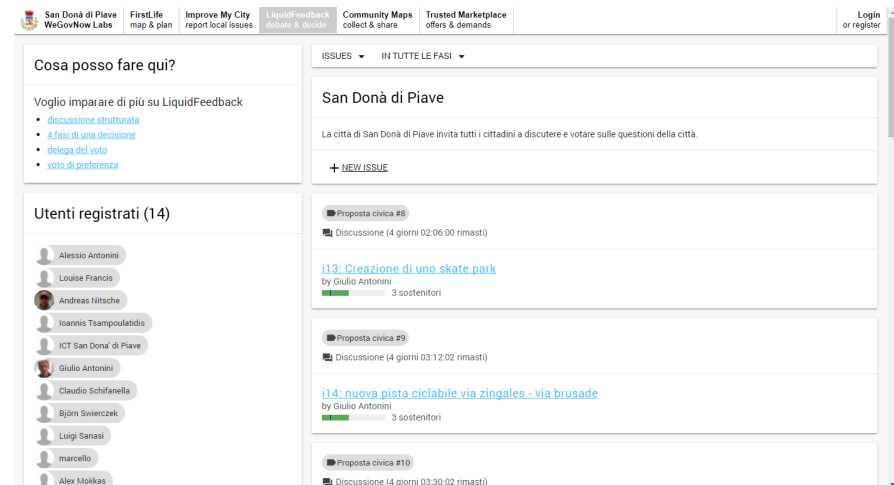
Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Tutorial and Guide:

tutorial.example.org



Community Maps



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Developers: "Lorem ipsum dolor sit amet, consectetur adipiscing

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Problem Solving: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

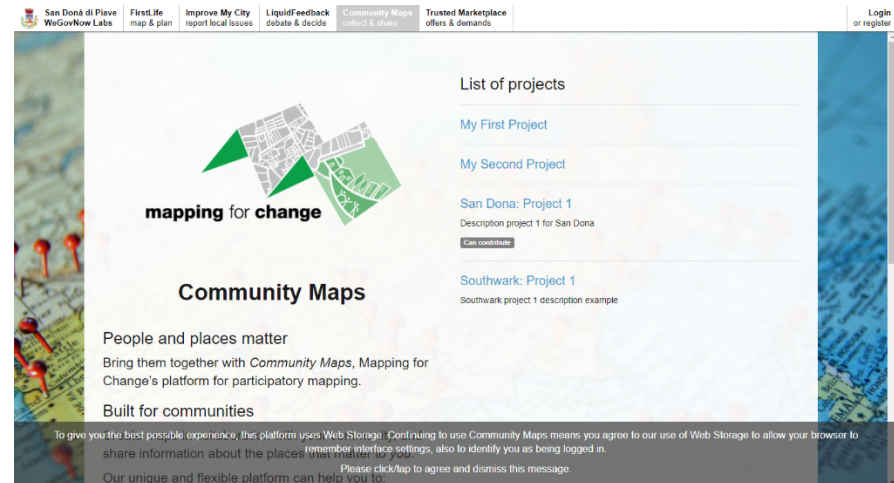
Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Tutorial and Guide:

tutorial.example.org



Trusted MarketPlace



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat laborum."

Developers: "Lorem ipsum dolor sit amet, consectetur adipiscing

Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Problem Solving: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

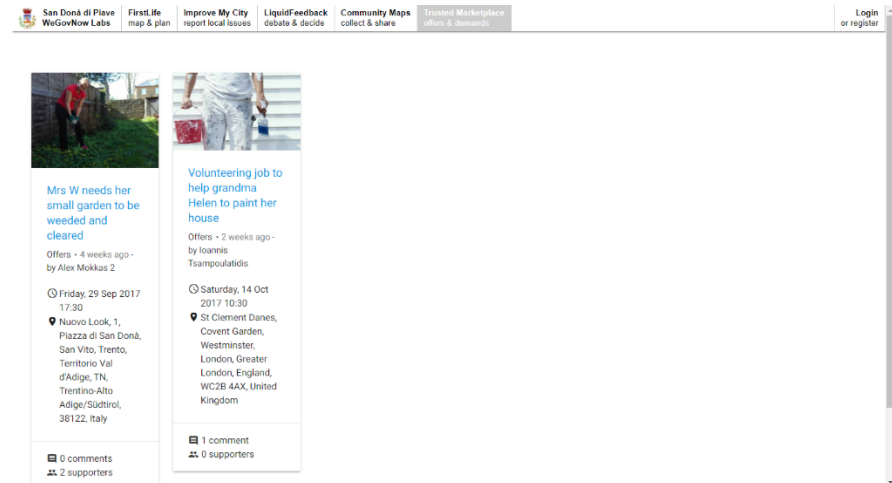
Name Surname

Phone Number: **011/00000000**

e-mail: **name@example.org**

Tutorial and Guide:

tutorial.example.org



Partners

Institutions:



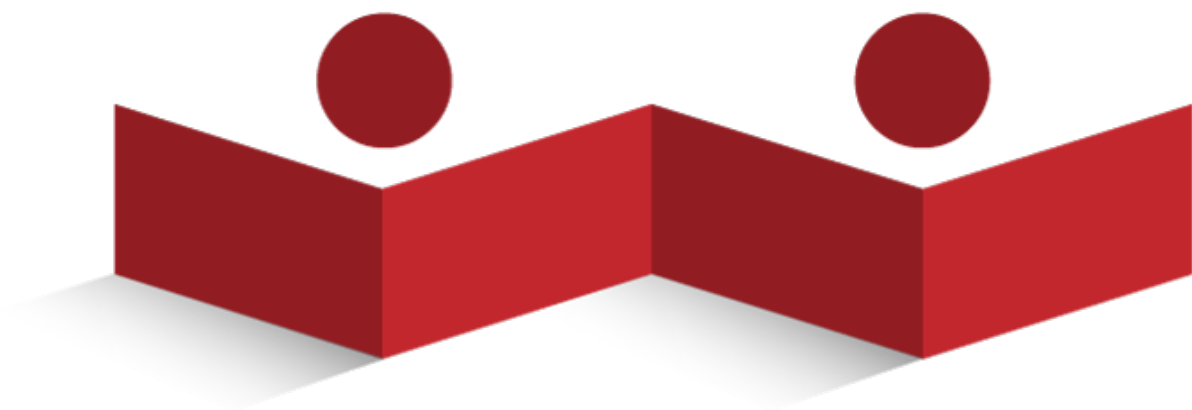
UNIVERSITA
DEGLI STUDI
DI TORINO

Organizations:



Croce Rossa Italiana

Company:



WeGovNow

San Donà di Piave Informazioni

Scenario 1

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referente: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Scenario 2



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referente: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Scenario 3

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut

Referente: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**



FirstLife

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod temp. Ut . Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Sviluppatori: "Lorem ipsum dolor sit amet, consectetur

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Assistenza: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

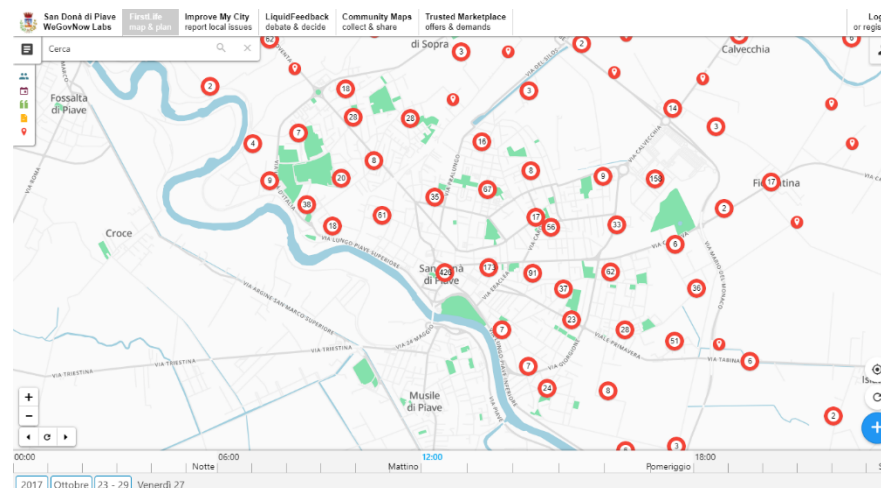
Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Guida all'uso:

tutorial.example.org



Improve My City

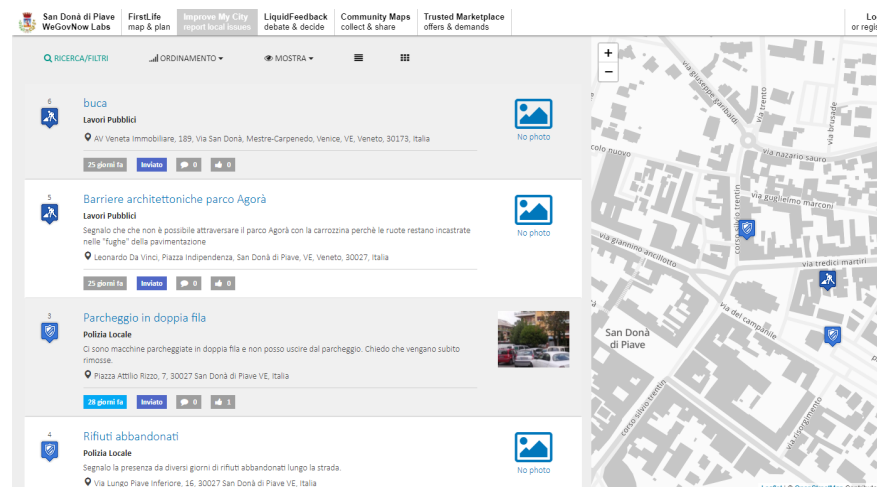
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut aute irure dolor in reprehenderit in voluptate velit esse cillum non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Sviluppatori: "Lorem ipsum dolor sit amet, consectetur

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**



Assistenza: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Guida all'uso:

tutorial.example.org

Liquid Feedback



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut aute irure dolor in reprehenderit in voluptate velit esse cillum nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Sviluppatori: "Lorem ipsum dolor sit amet, consectetur

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Assistenza: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

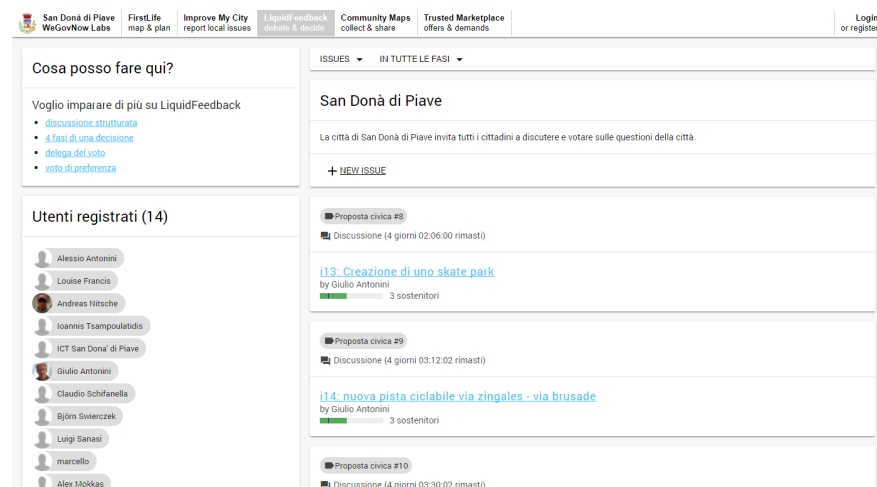
Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Guida all'uso:

tutorial.example.org



Community Maps

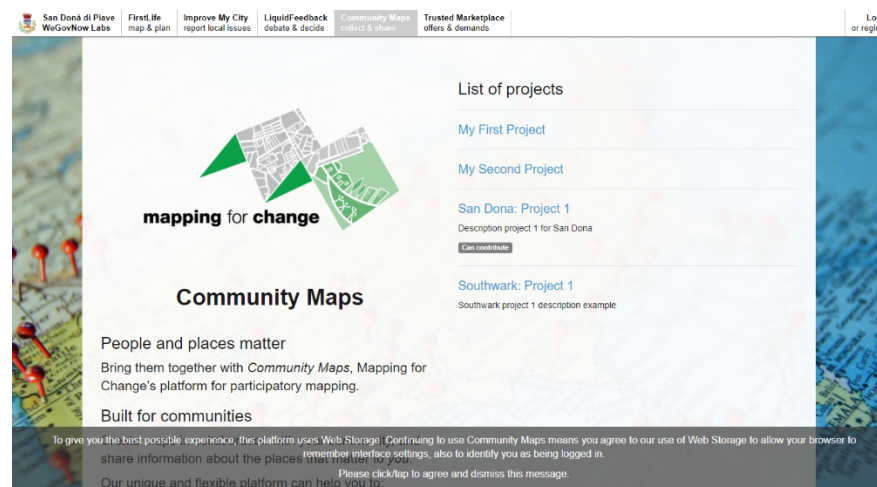
"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut aute irure dolor in reprehenderit in voluptate velit esse cillum, sunt in culpa qui officia deserunt mollit anim id est laborum."

Sviluppatori: "Lorem ipsum dolor sit amet, consectetur

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**



Assistenza: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Guida all'uso:

tutorial.example.org

Trusted MarketPlace



"Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum."

Sviluppatori: "Lorem ipsum dolor sit amet, consectetur

Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Assistenza: "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

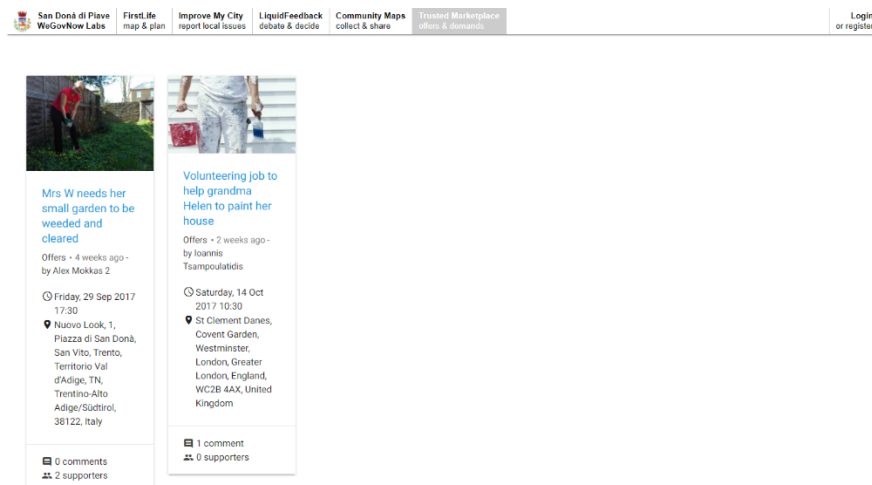
Nome Cognome

numero telefono: **011/00000000**

e-mail: **nome.cognome@prova.org**

Guida all'uso:

tutorial.example.org



Partners

Istituzioni:



UNIVERSITA
DEGLI STUDI
DI TORINO

Organizzazioni:



Croce Rossa Italiana

Imprese:
