



FLOOD-serv

D4.5 System Integration

Deliverable Report

Project acronym:	FLOOD-serv
Project full title:	Public FLOOD and Awareness SERVICE Emergency
Grant agreement no.:	693599
Responsible:	SIVECO
Contributors:	ANO, ANSWARE, CELLENT, DDNI
Document Reference:	D4.5
Dissemination Level:	<CO>
Version:	Final
Date:	29/11/2019



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 693599

Table of contents

Table of contents	2
List of figures	5
List of tables	6
List of abbreviations	7
1 Executive summary	8
2 Introduction	9
2.1 Objectives and Scope	9
2.2 Deliverable Type and Target Audience	9
2.3 Document Structure.....	11
3 Overview of the FLOOD-serv System and Its Architecture	12
3.1 The FLOOD-serv System	12
3.2 High Level Architecture	14
3.3 Extent of Integration	15
3.3.1 SSO Integration	16
3.3.2 Data Integration	16
3.3.3 UI Integration.....	16
4 FLOOD-Serv System Integration	17
4.1 Single Sign On and Role Mapping	17
4.1.1 SSO Protocol and Solution	17
4.1.1.1 SSO Protocol.....	17
4.1.1.2 SSO Solution	17
4.1.1.3 Role Mapping	17
4.1.1.4 SSO API.....	19
4.1.2 Emergency Management Console	19
4.1.3 Social Media Component.....	21
4.1.4 FLOOD-serv Semantic Wiki	23
4.1.5 Citizen Direct Feedback	23
4.1.6 Citizen Direct Feedback Mobile App	25
4.1.7 Territory Management System	25
4.2 Data Integration.....	25
4.2.1.1 FLOOD-serv Portal	25
4.2.1.2 Emergency Management Console	27
4.2.1.3 Social Media Component.....	29

4.2.1.4	Citizen Direct Feedback Component	30
4.2.1.5	Territory Management System	32
4.2.1.6	Sensors.....	33
4.2.1.6.1	Internal Sensors.....	33
4.2.1.6.2	External Sensors	36
5	Conclusions.....	41
6	References	42
	APPENDIX I: API Documentations.....	43
	Citizen Direct Feedback API	43
	Introduction	43
	SYNC Users	43
	Sync Users	44
	STATES	44
	Get a list of processes filtered by date of creation.....	44
	Get the list of attachments of a specific process.....	45
	Get the last version of file of a specific attachment.....	45
	Get the last version of file of a specific attachment (in base64).....	46
	Get list of movements of a process	46
	Update the status of a specific process	47
	Get a list of entities.....	47
	Get full details on a specific entity	48
	Base Data	48
	Available process states	48
	WEB REQUESTS	49
	Login User.....	49
	Report Issue.....	49
	Get Issue State.....	50
	Get Reported Issues.....	50
	Download Attachment.....	52
	6.1.1.1 Base Data	52
	6.1.1.1.1 Possible Values for “state”	52
	6.1.1.1.2 Possible values for “stateMessageld”	52
	Possible values for “resolutionMessageld”	53
	WEB MESSAGES.....	53
	Messages Received.....	53
	Messages Sent.....	54
	Create New Message	56

Territory Management System API	57
Introduction	57
Depth Analysis.....	58
Get a list of previous analysis filtered by date of creation.....	58
Create Analysis	59
EMC API	59

List of figures

Figure 1 : Title of figure *Error! Bookmark not defined.*

List of tables

Table 1 : Title of table *Error! Bookmark not defined.*

List of abbreviations

API	Application Programming Interface
CDF	Citizen Direct Feedback (component)
DOA	Description of Action
EMC	Emergency Management Console
OIDC	OpenID Connect
SAML 2.0	Security Assertion Markup Language version 2.0
SSO	Single Sign On
SW	Semantic Wiki (component)
TMS	Territory Monitoring Component
UI	User Interface

1 Executive summary

Executive summary is a term used in business for a short document that summarizes a longer report, proposal or group of related reports in such a way that readers can rapidly become acquainted with a large body of material without having to read it all. It will usually contain a brief statement of the problem or proposal covered in the major document(s), background information, concise analysis and main conclusions. It is intended as an aid to decision making by business managers. [1]

<This template for a “**Project Name** deliverable” contains the items that are necessary for quality and risk management. The project/WP leader can add his own information on subjects like “approach” and “planning”. There can be a separate project plan or the items can be part of a report. The items mentioned in this template have to be agreed upon and accepted at the start of a work package>

2 Introduction

2.1 Objectives and Scope

This document reports on the activities carried out in the FLOOD-serv Project, under Task 4.5 related to FLOOD-serv System Integration. Thus we present the System as a whole, its various components and modules and the process followed to achieve integration.

2.2 Deliverable Type and Target Audience

The Project Description of Action (DOA) states that deliverable D4.5 type is that of “Demonstrator”, and its dissemination level is Public. This is why the initial form of D4.5 was much shorter (not a Report) providing access information for the Portal and the entire FLOOD-serv System. However, we agree with reviewers’ assessment that in this way essential “more structured technical information on the developed system” is missing. Therefore, following the final review, we proceeded with making this deliverable a full report.

Reviewers’ requirements are quite specific about inclusion of detailed technical information. However, some detailed technical information may be confidential. Given that this deliverable is defined as Public, we present here public information. Where more detailed and confidential information is required (e.g. some technical requirements, API calls, etc.) that information is discussed generically in this report and made available fully in other revised confidential reports (which will be referenced in this document).

2.3 Changes in Response to Final Review Report

This report was modified in response to the Final Review Report. The following table describes how the issues raised were addressed in this report or elsewhere.

Reviewers’ observations	Explanations as to how observations are addressed
<p>The document consists of one-page links to the prototype platform, which in addition do not work. This form of reporting is not acceptable as the document does not contain more structured technical information on the developed system. Usual professional practice requires with the delivery of a software all the necessary documentation such as Technical Reference Manual, User Guide, etc...</p>	<p>The DOA defines deliverables D4.4 and D4.5 as of type “Other” and “Demonstrator” respectively, and not as Reports. Our team understood that what needed to be provided is the Portal /Integrated System themselves.</p> <p>Following your observations, we agree that the implementation of the Portal and the Integrated System need more complete documentation, so we proceeded with developing these deliverables into full reports. This report (D4.5) provides documentation about the FLOOD-serv System Integration, from a general overview, to a Component by Component discussion of SSO integration and Data integration.</p> <p>User Manuals were now added to component specific deliverables D3.3, D3.5,</p>

D4.5 System Integration

	D3.6, D3.7 and in the Portal related deliverable D4.4.
The delivered demonstrator of the integrated system is rather an empty shell than a complete prototype system. In fact, the access of an registered user to the FLOOD-serv on August 21st, 2019 allows displaying only a fragmented information from the testing period instead of a working prototype based on prepared scenarii, thoroughly validated by the flood operational services..	<p>Initially the Portal and Components were developed in a test environment. The initial deliverables (D3.3-D3.7) and D4.4-D4.5 pointed out to that environment. However, since that time, before the Piloting the Portal and Components were implemented in a production environment. The test environment is out of date, missing much content and some components are no longer available there.</p> <p>The Piloting deliverables (D5.2-D5.4) pointed correctly to the production environment access information.</p> <p>The updated access information is presented in Section 3.1, below. (Similarly, deliverables D3.3, 3.5-3.7, D4.4 were updated to reflect the changes in access information).</p> <p>For issues encountered with accessing the System, the FLOOD-serv Consortium was and is available to clarify and solve any difficulties. Please contact us and we will promptly respond if there are still any issues encountered.</p>
For cities Bilbao and Bratislava there was not possible displaying any map showing current or future weather/inundation situation.	Data reports for the Cities: Bilbao, Bratislava, Tulcea and Vilanova de Famalicao are available and fully functional. Given that data for those reports are collected usually from other sources (sensors or external data sources with sensor data, or whether data – as described in Section 4.2.6), it may happen that temporarily data may not be available if the sensor or system from which data is collected is down. Most of the time, however, these reports are available.
It is also impossible to test some developed functions (e.g. Social Media Component or Territory Monitoring System).	Access to Components (EMC, SMC, TMS, CDF) can be done by logging in the Portal with users which have user roles created for public administrators (e.g. Decision Maker, Facilitator, etc.). A list of test user names and passwords (which is confidential information not to be included in a public report) was sent separately to the Project Officer to be shared with Project Monitors.
Assess to public data is not indicated, what is	Data sources for the FLOOD-serv System, for

<p>in contradiction with D1.5 Data Management Plan – final Version and Open Research Data Pilot (ORDP) initiative the project has declared to follow.</p>	<p>data usually collected by the EMC Component (which acts as a project data repository) is described in Section 4.2, and more specifically in 4.2.6.</p> <p>Wherever the FLOOD-serv Portal presents data reports it indicates the source of that data.</p>
<p>As a result of 3-years project such a public demonstrator is in total disagreement with the Project main objectives</p>	<p>While we agree that we did not present all relevant information, and we are disappointed that the Project Monitors had difficulties accessing the FLOOD-serv System and its Components, we believe that the System has met the Project's Objectives.</p> <p>It should be noted that all Components are functional, and all user requirements elicited were implemented and Piloted. The Final Piloting Report (D5.4) shows that the at the end all issues, bugs detected and improvements suggested during piloting were corrected and all functionalities were/are in working order.</p>

2.4 Document Structure

In the next Chapter, no. 3, an overview of the FLOOD-serv System is given with the presentation of the High Level Architecture and the extent of integration of the system.

Chapter 4 discusses in detail the integration of each Component of the System under two main headings: Single Sign On Integration and Data Integration.

We draw conclusions in Chapter 5.

In APPENDIX I, the documentations of various components' APIs are provided.

3 Overview of the FLOOD-serv System and Its Architecture

3.1 Access Information

Access to the FLOOD-serv Integrated System is realized through the FLOOD-serv Portal which acts as a single entry point and SSO for the whole System.

The main Portal URL is: <https://floodserv.eu/>.

After the user chooses a City and a Language, the user is directed to the URL for the 5 instances of the Portal (corresponding to the five Pilot Cities):

Bilbao: <https://bilbao.floodserv.eu/>

Bratislava: <https://bratislava.floodserv.eu/>

Genova: <https://genova.floodserv.eu/>

Villa Nova de Famalicao: <https://vilanovafamalicao.floodserv.eu/>

Tulcea: <https://tulcea.floodserv.eu/>

Any Portal visitor can sign up (i.e. create an account) following the Sign Up procedure, described in the User Manual in D4.4, APPENDIX I, Section **Error! Reference source not found.** A self created account is given automatically the role of e-citizen.

As a citizen you will not have access to the other components (except for the Semantic Wiki).

To access the other components aimed at public administrators (employees of Pilot Cities), a set of test user names and passwords have been generated and sent to the Project Officer to be shared with Project Monitors.

3.2 The FLOOD-serv System

The FLOOD-serv System is a multilanguage, multicomponent, integrated system aimed at serving citizens and public administrators in various aspects of flood risk management. The components of the FLOOD-serv System are the following:

1. **The FLOOD-serv Portal:** is a portal dedicated to citizens of pilot and partner cities aiming to contribute to flood risk mitigation. It is conceived as a two-way information gateway; citizens can look up information about floods, or they can submit information to relevant public authorities in their city, and engage in a dialogue with them. The FLOOD-serv Portal offers news, multimedia galleries and data-based flood reports in each of the participating cities. It also contains the FLOOD-serv Semantic Wiki, providing to citizens and specialists, systematic information about floods. Citizens can also submit issues and information to public authorities by using the citizen involvement form . This Portal was developed under the FLOOD-serv Project, and it is the front end to a series of other information system components developed in the same project.
2. **Citizen Direct Feedback (CDF)** module provides a direct communication channel from the citizens to the respective local authorities, enabling them to more effectively inform and engage in dialogue with those local authorities of any flood related information, in terms of risks or prevention. CDF empowers the citizen by allowing

them to alert and discuss with local authorities of any potential flood risks or assess on prevention policies. CDF is a back office component for use by public employees of relevant emergency services in the pilot cities. The front office to CDF is implemented in the Citizen Involvement mobile app and in the Citizen Involvement Form which are designed for use by citizens. CDF is developed by ANO Software.

3. **Emergency Management Console (EMC)** can be used as a data visualization or decision support system dedicated to employees of public authorities involved in flood emergency management. It receives and monitors relevant data from a variety of sources, internal to the FLOOD-serv project (from other components e.g. data submitted by citizens, or based on analysis of satellite pictures, etc) and from external sources (e.g. meteorological and sensor data). EMC generates various visualizations of data based on maps and charts, proposes response measures and tracks their evolution. The EMC receives a Crisis Snapshot of the emergency with the elements defining the crisis. From the Crisis Snapshot, the EMC analyses the situation of the crisis and propose a Crisis Action Plan (CAP) a list of operations or activities to be executed in a timing sequential order making use of the necessary resources to fight against floods. The tool includes a GIS component, based on PostGIS tools. It has a User Interface for the visualization and management of the crisis; and a BackEnd component with an Expert System based based on rules that generates the Crisis Action Plan of the current emergency. The User Interface is a responsive Web interface based on React framework and the mobile app is based on React JS. EMC is developed by Answare Tech, Spain.
4. **FLOOD-serv Semantic Wiki (SW)** is a semantic wiki containing general information and knowledge about floods and flood management but also specific and contextualized knowledge related to the FLOOD-serv Project. The SW is dedicated for use by both specialists on the one hand, and regular citizens on the other. The entries in the SW are written in a brief encyclopedic style of definition + further details. However we aim at keeping the entries rather short, and often the development of further details are made into other entries. The FLOOD-serv SW is based on Mediawiki and Semantic Mediawiki technologies. The implementation of the SW component was done by SIVECO Romania, one of the partners and coordinator of the FLOOD-serv Project.
5. **The Social Media Component (SMC)** aims to monitor and provide awareness citizen's concerns about flood related issues. It collects, monitors and analyses articles, opinions, posts about floods from various social media and Web sources, visualizes and structures them for further analysis. This provides an additional tool for flood management authorities to be aware of public concerns and sentiment. This assist in developing longterm strategies and communicate with the citizens efficiently through multichannel messaging in case of an emergency. This component was developed by CELLENT.
6. **Territory Monitoring System (TMS)** is an instrument for producing situational awareness and risk analysis within a geographical area by means of analysis of satellite and aerial pictures (from airplanes or drones). By analyzing successive images, its intelligent processing engine is able to generate reports about relevant change events related to flood occurrences or impacts, and is able to geographically localize them. The TMS component receives satellite and/or aerial pictures (taken by drones or airplanes) and is able to localize and orient them on a digital map of the area of concern. By analyzing successive pictures, taken at identified times, the TMS is able to recognize changes in the area and generate exactly localized events. The TMS intelligent processing engine is also able to classify the nature of the events which may be related directly to flood events: e.g. enlargement of surface covered by water

or the braking of dikes or dams, or related to flood risk: e.g. apparition new elements like buildings, building sites in at risk areas. The TMS is able to generate periodic reports (based on periodicity of input pictures collection) and alarms that can be used by themselves in the TMS Graphical User Interface or exported in a decision support or emergency management system like EMC. Data generated by TMS is sent for further analysis to the EMC. TMS is developed by ANO Software.

In addition to the above main components the following should be mentioned:

7. **Single Sign On module** based on Keycloak technology (further details in Section 4.1, below), broadly viewed as part of the Portal, but technologically different
8. **Sensor module** (see Section 4.2.6, below).

3.3 High Level Architecture

Figure 1 below displays the designed high level logical architecture of the FLOOD-serv System, as presented in D4.3, Section 2.2.

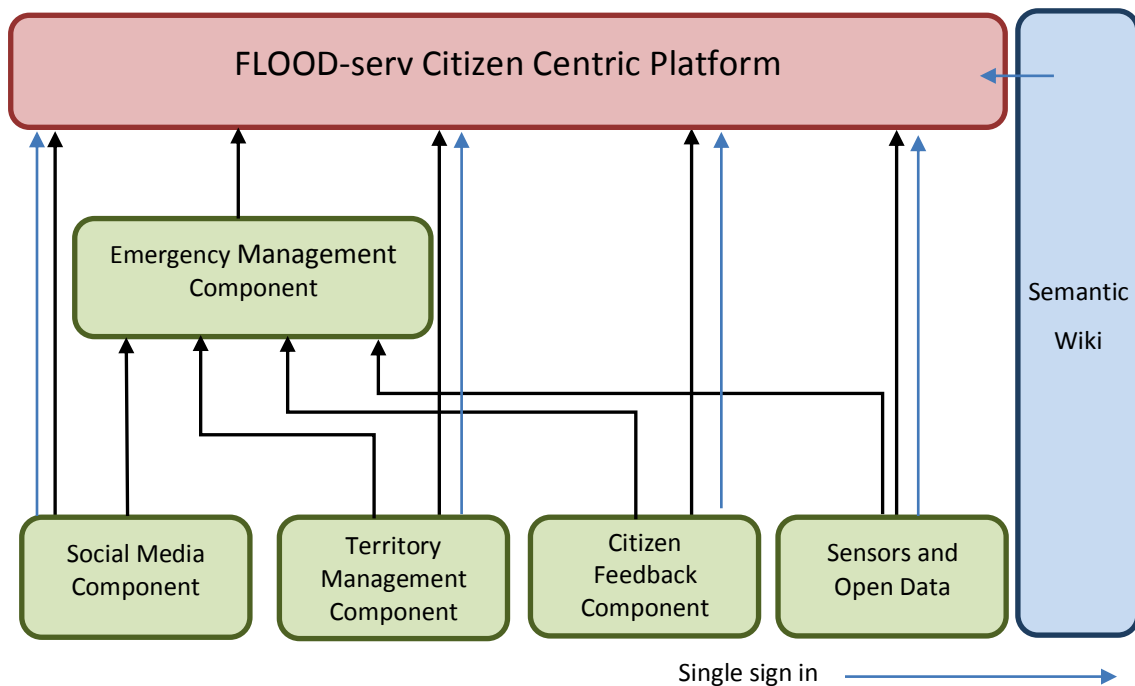


Figure 1: FLOOD-serv System Designed High Level Logical Architecture

Figure 2: Initial High Level Logical Architecture of the FLOOD-serv System

Initial high level logical architecture of the FLOOD-serv System was designed to allow for many interconnections between components, thus each component (except the Semantic Wiki) was seen as connecting to both the EMC and the Portal. This was due to not wanting to lock in interconnectivity design but have an initial design that is flexible and adaptable to project needs.

The updated final high level description of the FLOOD-serv System is presented in Figure 3, below. Here are the explanations to the few changes in architecture:

- In the implementation phase, more care and analysis was dedicated to which connections are needed to support the functional specifications of the System and its components, and only those interconnections were implemented. Thus the following changes intervened.
- The idea of connecting components to both the EMC and the Portal was dropped. Usually the EMC was seen as the System's data repository therefore in most cases where the Portal needed the same data it would take it from the EMC rather than directly from other components.
- An exception was CDF which is connecting directly to the Portal due to the fact that the Portal (through the Citizen Involvement Form) acts as a front office (citizen facing) interface to citizens.
- A user interface to TMS was implemented in the CDF component, therefore the TMS connects into CDF
- We now also included the CDF Mobile App and EMC Mobile App into the chart.

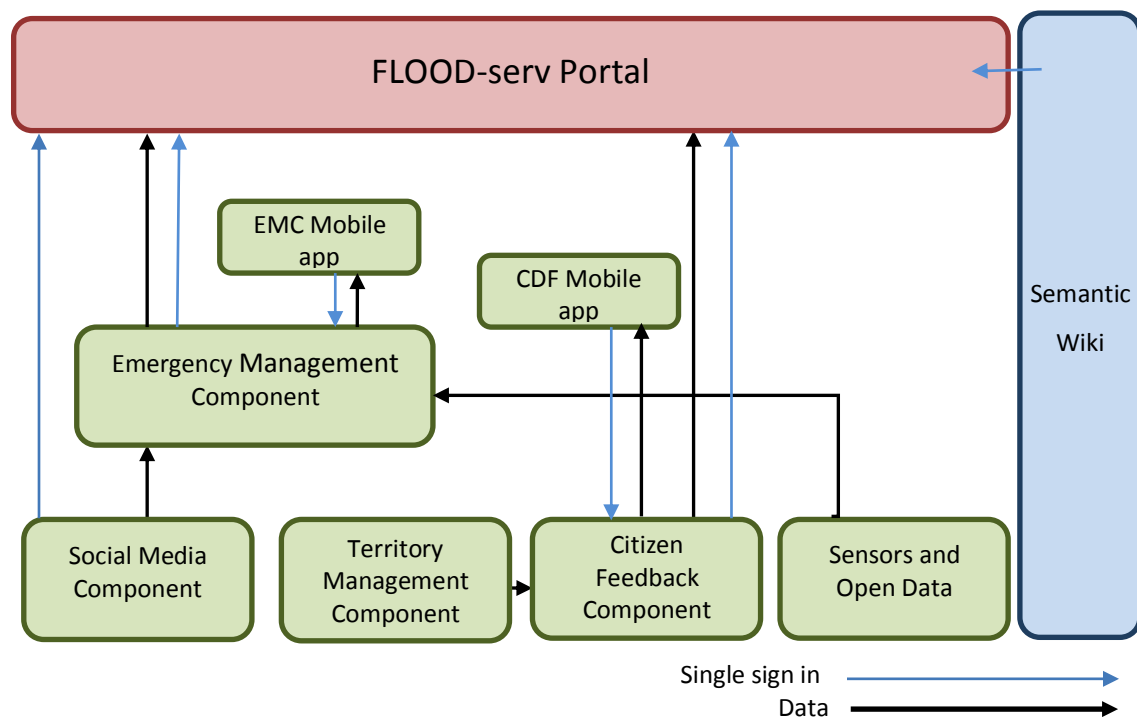


Figure 3: Final High Level Logical Architecture of the FLOOD-serv System

3.4 Extent of Integration

In the design and implementation stages we considered three aspects under which integration of Components could be carried out:

1. Single Sign On;
2. Data Integration; and
3. User Interface Integration.

3.4.1 SSO Integration

Single Sign On integration – represents the integration of components such that from a functional point of view the user logs in in one place and accesses all areas /Components of the system to which he should have access given his/her role. This implies a secure sign in and user management.

It was generally accepted that the FLOOD-serv System needed a Single Sign On system that, from a user and functional point of view, would simplify user access to the System and its Components.

The proposed SSO standard was Oauth 2.0, a popular open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.

3.4.2 Data Integration

Data Integration represents the interconnection between components such that they can send data from one to another. Data integration was achieved by means of Web services using data producing component's API. As explained in Section 3.3, above, which data interconnections were implemented depended on the functional requirements of the System and its Components. Data links are visible in Figure 3, above. Details about data integration at the level of each Component are given in Section 4.2, below.

3.4.3 UI Integration

UI integration refers to the extent to which a system's several components share the same interface or not. In the case of the FLOOD-serv System, once again such decision was subordinated to functional considerations. It was decided that there is no need for full UI integration for several reasons:

- The Components have a quite distinct identity, each dealing with different aspects of Flood Risk Management (FRM) (e.g. emergency situation management is quite different from social media analysis or from obtaining feedback/inputs from citizens);
- There were no specific use cases derived from the User Requirements which needed the user to go across components. While some extended use cases where the user may need to go from one component to another can be imagined (e.g. the user performs some social media analysis, then follows through in the EMC), it still makes sense to group the functionalities by functional area (social media analysis vs. emergency management) which is already done by the different identities of the Components.
- Most components are for use by public administrators, therefore more specialized users who can be trained to understand the distinct functions of different components, therefore a seamless integration of the interface is not required;

An exception to the above exists: the FLOOD-serv Portal, in the *Engage with public authorities* area, acts as a citizen facing user interface for the CDF Component. This functionality was dedicated to citizens (and not public administrators), therefore a stricter requirement of this integration between the two components being seamless existed.

4 FLOOD-Serv System Integration

4.1 Single Sign On and Role Mapping

4.1.1 SSO Protocol and Solution

4.1.1.1 SSO Protocol

As stated in the System Architecture deliverable (D4.3), the FLOOD-serv System uses the OAuth 2.0 protocol for SSO. OAuth 2.0 is an open protocol standard for authorization and access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords.

This protocol allows third-party applications to grant limited access to an HTTP service, either on behalf of a resource owner or by allowing the third-party application to obtain access on its own behalf. Access is requested by a client, it can be a website or a mobile application for example.

In addition the FLOOD-serv Technical partners decided to also use OpenID Connect (OIDC) which is an authentication layer on top of OAuth 2.0. OIDC allows clients to verify the identity of a user based on the authentication performed by an authorization server as well as to obtain profile information about the user in an interoperable and REST-like manner.

4.1.1.2 SSO Solution

The technology of choice for SSO, role mapping and user management was Keycloak which was installed, customized and configured by Siveco. While initially a PHP implementation of SSO was considered, it was decided that Keycloak is a more mature solution. This has triggered a technology change on the Portal side too. Keycloak is an open source identity and access management solution aimed at modern applications and services. Keycloak supports protocols like: OpenID Connect, OAuth 2.0, and SAML 2.0.

4.1.1.3 Role Mapping

Role mapping planning took place in successive requirements documents under WP4 (based also on D3.1 and D3.2). In D4.1 a general mapping between roles and Components was produced in Chapter 4. We reproduce here in Table 1, below, a table from D4.1.

Table 1: User Roles and Components Access in the FLOOD-serv System

User Category	Component User Type	EMC	SMC	SW	TMS	CDF	FLOOD-serv Portal
		Non-institutional users	Unregistered user				
	E-Citizen					X	X
	Certified citizen/ observer					X	X

D4.5 System Integration

Institutional users	Decision Maker	X	X	X	X		X
	Facilitator (or platform operator)	X	X	X	X	X	X
	Flood Emergency Expert	X	X	X	X	X	X
	Emergency responder	X	X	X	X		X
	Content editor			X			X
	System Administrator		X	X	X	X	X

Further on, in D4.2 we presented more detailed tables showing user roles by each Component's functionalities. These remained the reference documents for role mapping implementation. In the case of the FLOOD-serv Portal, as some functionalities/requirements were detailed during implementation and updated matrix of this kind was produced and maintained (in Excel format). A presentation of the access rights to Portal functionalities by user roles is presented in Table 2, below. (Note the Excel table – available upon request – contains many more columns referring also to which functionalities are to be implemented for which Pilot city, plans to pilot in piloting cycles, etc.)

Table 2: User Roles and Access to FLOOD-serv Portal Functionalities

User stories name	Unregistered user	E-Citizen	Certified citizen/ observer	Decision Maker	Facilitator (or platform operator)	Flood Emergency Expert	Emergency responder	Content editor	System Administrator
View newsfeed	1	1	1	1	1	1	1	1	1
Add news	0	0	0	1	1	1	1	1	1
Edit news	0	0	0	1	1	1	1	1	1
Delete news	0	0	0	1	1	1	1	1	1
Share news	1	1	1	1	1	1	1	1	1
View general alarm level	1	1	1	1	1	1	1	1	1
View multimedia galleries	1	1	1	1	1	1	1	1	1
Create new gallery	0	0	0	1	1	1	1	1	1
Edit gallery details	0	0	0	1	1	1	1	1	1
Delete gallery	0	0	0	1	1	1	1	1	1
Upload picture or video into existing gallery	0	0	0	1	1	1	1	1	1
Edit picture or video related details	0	0	0	1	1	1	1	1	1
Delete picture or video from an existing gallery	0	0	0	1	1	1	1	1	1
Visualization of information/reports concerning floods	1	1	1	1	1	1	1	1	1
Communicate flood-related information to authorities	0	1	1	1	1	1	1	1	1
Two way messaging between citizen and public authorities.	0	1	1	1	1	1	1	1	1
View feedback status	0	1	1	1	1	1	1	1	1
View history of feedback submission	0	0	0	0	0	0	0	0	0
Component access	0	0	0	1	1	1	1	1	1
About This Portal	1	1	1	1	1	1	1	1	1
About FLOOD-serv project	1	1	1	1	1	1	1	1	1
About FLOOD-serv components	1	1	1	1	1	1	1	1	1
About Partner Organizations	1	1	1	1	1	1	1	1	1
About Consortium Organizations	1	1	1	1	1	1	1	1	1
Manage FLOOD-serv portal roles	0	0	0	0	0	0	0	0	1

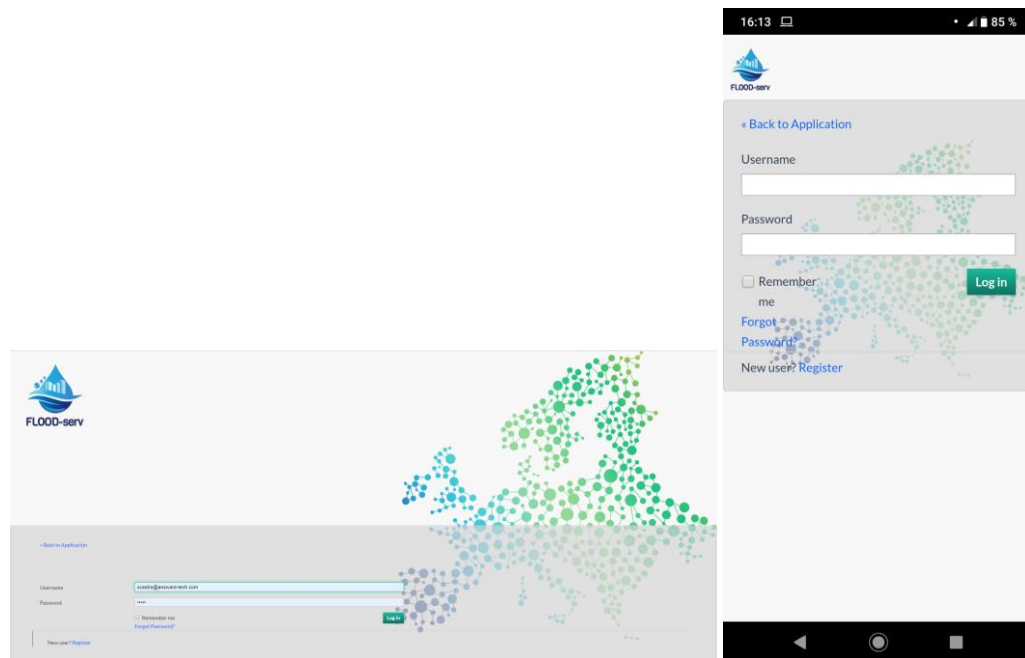
Log-in/ Single sign-on	1	1	1	1	1	1	1	1	1
Log out	0	1	1	1	1	1	1	1	1
Edit profile	1	1	1	1	1	1	1	1	1
View users list	0	0	0	0	0	0	0	0	1
Add user	0	0	0	0	0	0	0	0	1
Delete users	0	0	0	0	0	0	0	0	1
Inactivate users	0	0	0	0	0	0	0	0	1
Page 0	1	1	1	1	1	1	1	1	1
Home Page	1	1	1	1	1	1	1	1	1

4.1.1.4 SSO API

Documentation related to Keycloak API is available at: <https://www.keycloak.org/docs-api/8.0/rest-api/index.html>. The Documentation of API services was transmitted by Siveco to Technical Partners in charge of Components, for them to implement authentication and rights in their Components.

4.1.2 Emergency Management Console

The integration of the EMC with the SIVECO SSO was done through a Python module in which the different messages with the KeyCloak server were implemented through the OpenId protocol. Among these messages are obtain, renew and validate tokens plus obtain user information, their roles and their groups.



SIVECO authentication system integrated in the EMC web console (on the left) and in the EMC app mobile (on the right)

D4.5 System Integration

ID	Username	Email	Last Name	First Name	Actions
c2596e4-2226-4f30-b02b-25ef11a...	abrax	comp8@mailinator	Sitr	Alin	Edit Impersonate Delete
c2630282-8624-4095-a030-49c1866b...	adela1	adela.choborova@gmail.com	Choborová	Adela	Edit Impersonate Delete
b4300461-1154-4903-9464-c50b2171...	adminbratislava	adminbratislava@floodserv.eu	bratislava	admin	Edit Impersonate Delete
25c32d18-4278-428e-820f-5be23e4...	andrei	aqgzeanu@gmail.com	Ogrzeanu	Andrei	Edit Impersonate Delete
0507690-4695-4f68-9705-645e27163...	anna	anna.nadadyova@gmail.com	teste	Anna Nádadyová	Edit Impersonate Delete
a6880bc7-4483-4840-846c-75e0540...	ans.floodserv@answare-tech.com	ans.floodserv@answare-tech.com	DSS	DSS	Edit Impersonate Delete
0564f2f1-f4cf-473c-a1f5-682b2202f...	bleidner@yahoo.com	bleidner@yahoo.com	Koss	James	Edit Impersonate Delete
614b3838-a46c-4189-bc70-d81fb59...	borky247	boris.beho247@gmail.com	Bleho	Boris	Edit Impersonate Delete
586e5614-a7c3-4794-9c7a-ee5f6b...	br12	br12@gmail.com	teste	br12	Edit Impersonate Delete
06c741ec-59d8-4447-8e5f-f8f8251f...	bratislava	bratislava@floodserv.eu	bratislava	user	Edit Impersonate Delete
42f7244-e804-4299-bb45-40eb5b6c...	bratislava_certified_citizen	bratislava_certified_citizen@floods...			Edit Impersonate Delete
e83892c-1cb3-4119-8f5c-4c93295d...	bratislava_content_editor	bratislava_content_editor@floods...			Edit Impersonate Delete
a653d06c-42bd-4707-b76f-841587c...	bratislava_decision_maker	bratislava_decision_maker@floods...	maker	decision	Edit Impersonate Delete
24ab7377-f1b3-424e-a630-b3479e1...	bratislava_e-citizen	bratislava_e-citizen@floodserv.eu			Edit Impersonate Delete
35473207-19da-4184-9448-e44985c...	bratislava_emergency_expert	bratislava_emergency_expert@flood...	expert	emergency	Edit Impersonate Delete
921de5a1-918e-4bc2-a2ca-dbf0f18a...	bratislava_emergency_respondent	bratislava_emergency_respondent@...	responder	emergency	Edit Impersonate Delete
b1fab449-735c-43c7-8c03-44306e7...	bratislava_facilitator	bratislava_facilitator@floodserv.eu	facilitator	emergency	Edit Impersonate Delete
0e256449-2704-4376-8c34-39e98be...	busova	bb@zmail.sk	Busova	Bozerna	Edit Impersonate Delete
06e6274b-f688-407f-b48f-e1b3f6b...	bvuser	karol.kaimar@bviss.sk	User	BVIS	Edit Impersonate Delete
0245c37f-4c99-4068-8450-c71605...	cbrat	comp7@mailinator.com	Sitr	Georgi	Edit Impersonate Delete

Keycloak to create/modify/remove users and groups for each pilot instance

When new user groups are created in the Keycloak system, in the EMC these new user groups are automatically added.

New mission [X]

Title *

Description *

Group

- Firemen
- Policemen
- Ambulance
- Volunteers

Date *

November 13, 2019 10:00 AM

November 13, 2019 10:00 AM [X]

Add new task

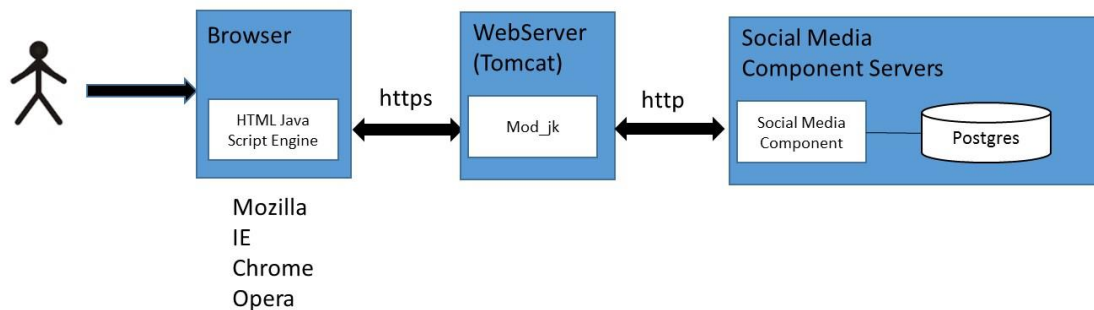
Area of interest *

Cancel Create

User groups created in the Keycloak and automatically added to the EMC

4.1.3 Social Media Component

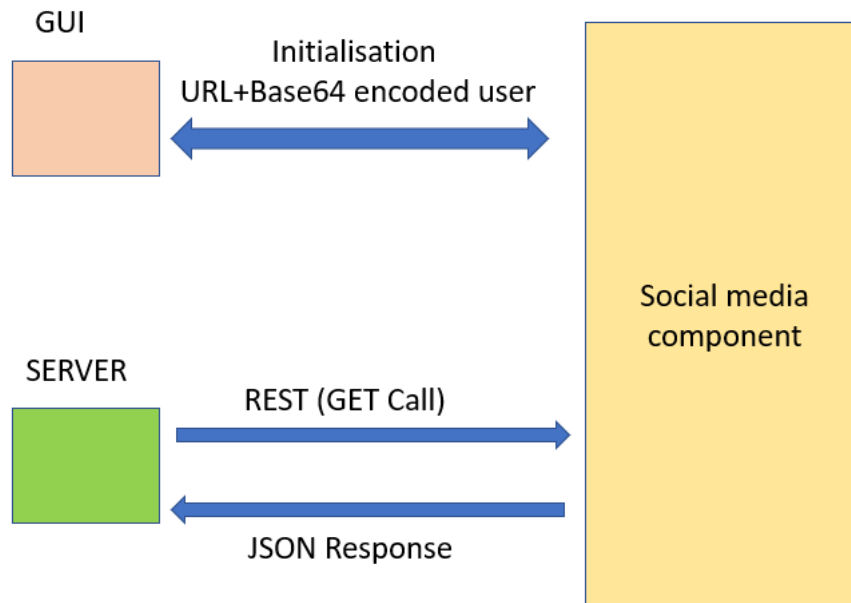
The software is web-based and as such does not require any installation on the user apart from a standard browser. The language of the GUI adjusts automatically following the browser language.



D4.5 System Integration

The SMC can be called standalone directly through a browser with a URL. In this case username / password will be required.

The SMC is also ready for integration with other components using two integration mechanisms, direct call (URL) and REST-call. This means the SMC can be called from any other component of the FLOODSERV universe, but also from third party applications such as an eGov portal of a city. This approach has been chosen to be as flexible as possible.



The GUI initialization is used to enable SSO.

It requires a weblink for direct calling the GUI

The username needs to be added in the HTTP Header of the request. In this case the username / password entry in the SMC is not necessary and the SMC proceeds directly to the first screen of the application. By adding the username in the request the SMC invokes the role assigned to the specific user in the SMC access control database.

The composition of the URL is as follows:

[https://socialmedia.cellent.at/socialmedia-client/login?id=\[base64 encoded username from the SMC\]](https://socialmedia.cellent.at/socialmedia-client/login?id=[base64 encoded username from the SMC])

Base64 has been chosen, because it is particularly prevalent on the Web (and therefore a standard way to include such information in an URL). It is used to include binary assets inside textual assets such as HTML and CSS files. Base64 is designed to carry data stored in binary formats across channels that only support text content

Base64 is a group of binary-to-text encoding schemes that represent binary data in an ASCII string format by translating it into a radix-64 representation. Each Base64 digit represents exactly 6 bits of data. Three 8-bit bytes (i.e., a total of 24 bits) can therefore be represented by four 6-bit Base64 digits.

Those can be read by the SMC and the checked against the SMC database. The downside is that the login data are not encrypted, but this is considered less important since the service to be accessed does not contain confidential data (the raw data are public anyway)

4.1.4 FLOOD-serv Semantic Wiki

The FLOOD-serv Semantic Wiki Component, based on Mediawiki and Semantic Mediawiki technologies, was integrated with Keycloak for Single Sign On and user management purposes, using the Keycloak API. When a user tries to enter the Semantic Wiki this component calls Keycloak to find out user's information and valid log in token. If the user is logged in the SW checks against its local list of users and gives access to the corresponding local user. If the user is new (logs in the SW for the first time) then a corresponding user is created locally and given access to the SW.

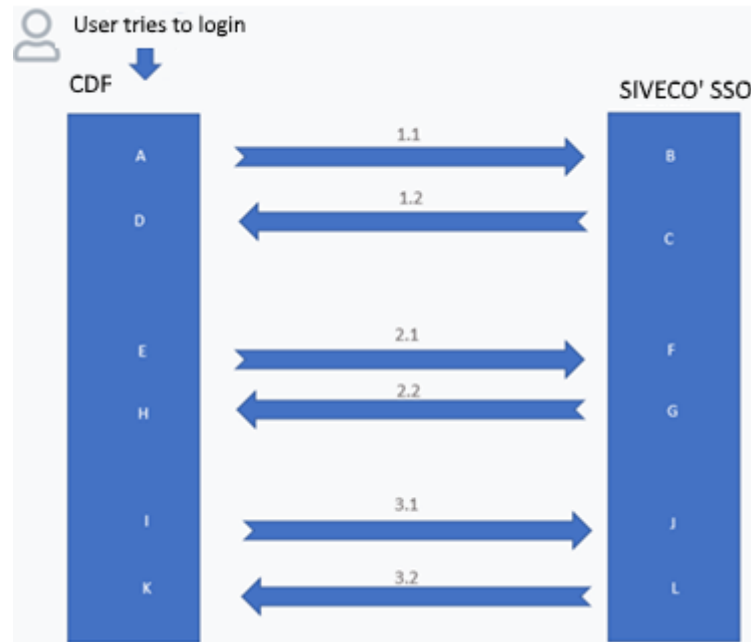
Managing user roles for SW is simple given the business decision that all registered users should have access to all content of the SW while unregistered users should have only view access. As such, the SW needs only to know if the user is registered in Keycloak and give similar access to all roles except unregistered users.

Additional synchronization between the SW on the one side, and Keycloak and the Portal, on the other side, needed to be made in terms of language preference synchronization.

4.1.5 Citizen Direct Feedback

The CDF backoffice was integrated with the SIVECO SSO through a dedicated module developed in JAVA and fully integrated with Spring Security (the security layer in the CDF technological stack). This was achieved using the OpenID protocol for the exchange of messages, using the serviced API.

When a user tries to login, these are the messages exchanged between CDF and the SIVECO'S Keycloak:



INITIAL REQUEST (1)

- 1.1 – Redirect to SIVECO to authenticate or get authentication already done
- 1.2 – Returns “code”

Firstly, when the user tries to enter CDF, it will be redirected to the authentication page of the authentication server, i.e. the SIVECO'S Keycloak. If this user is already authenticated, the portal immediately returns the code so that the CDF can continue the entire local authentication process. If the user is not yet authenticated to SIVECO's then normal authentication will be requested and then the code will be returned to the requesting application.

GET TOKEN (2)

- 2.1 – Request to get token based on the returned code
- 2.2 – Returns tokens

Once the code has been obtained, we must obtain the token from its returned code.

GET BASE DATA (3)

- 3.1 – Get Base Data
- 3.2 – Returns Base Data

After obtaining the token we can invoke the various web services. One is the web service that returns basic user information, used for the session injection within CDF.

In terms of syncing the user and group database between CDF and SIVECO' SSO, the CDF uses a backend procedure to keep its database up to speed, acting as slave and thus treating the portal as master. Thus, operators for the CDF backoffice are always first created in the portal.

4.1.6 Citizen Direct Feedback Mobile App

The CDF mobile app was integrated with the SIVCO SSO through a dedicated module developed in JAVA. This was achieved using the OpenID protocol for the exchange of messages, using the serviced API. The same logic used in the CDF Backoffice was applied with a difference: the CDF App also uses the keycloak method to create a new user, when a citizen registers through the app.

4.1.7 Territory Management System

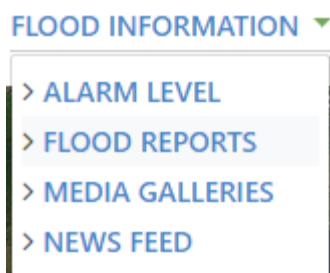
The TMS was integrated with the SIVCO SSO through a dedicated module developed in JAVA and fully integrated with Spring Security (the security layer in the CDF technological stack). This was achieved using the OpenID protocol for the exchange of messages, using the serviced API. The same logic/approach from the CDF was used – check sub chapter 4.1.6.

4.2 Data Integration

4.2.1 FLOOD-serv Portal


The FLOOD-serv Portal consumes data from two main sources: EMC and CDF.

The Portal consumes data from the EMC in order to produce a series of data reports aimed at informing citizens about floods. By going in the Main Menu/Flood Information/Flood reports or Main Menu/Flood Information/Alarm Level, the user can access such information:



An example of Alarm Level is shown in Figure 4, below.

D4.5 System Integration



Rain's flood
Start:28/08/2019 12:11:50 End:
ORANGE- Flood alert II
Flood alert level II is declared a) when the water level of water flow defined in the flood plan is reached and when the water l...
art 11 of Act No. 7/2010 Coll. on protection against floods as amended

[Read more](#)

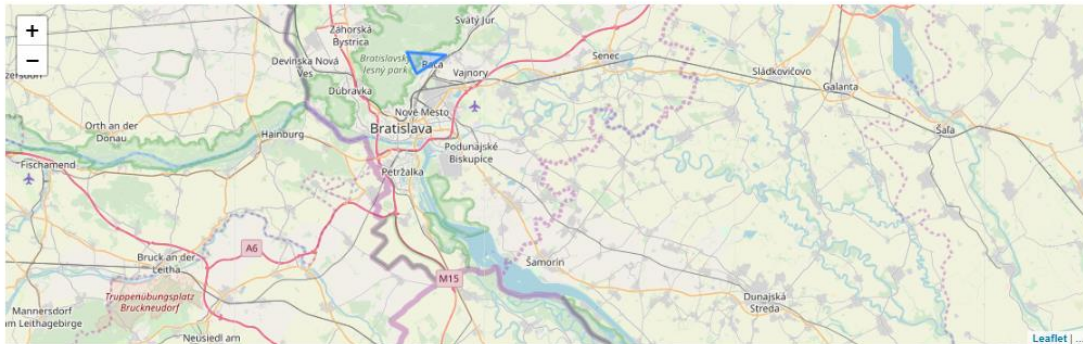
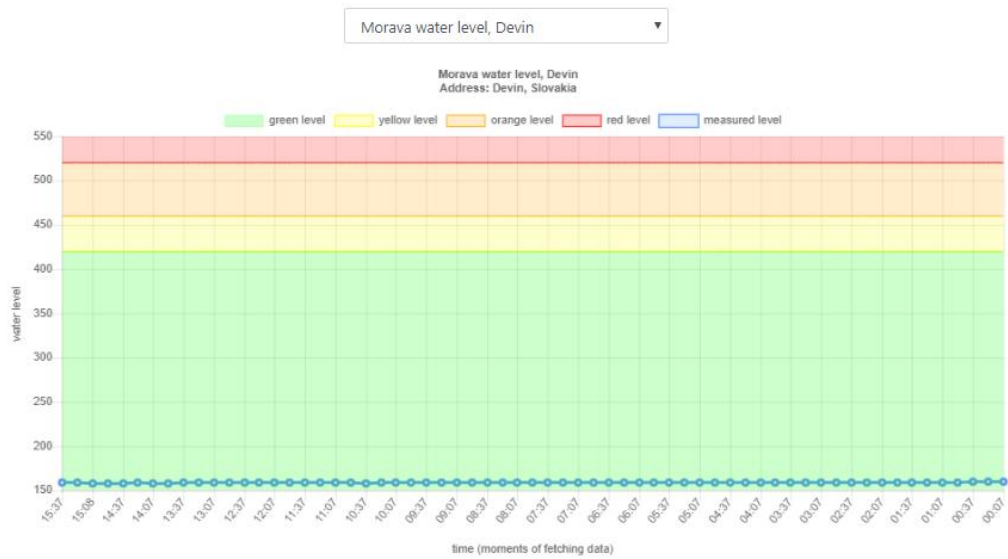


Figure 4: Example of Alarm Level Published on the Portal

Examples of data reports are shown in Figure 5 and Figure 6 below.



This chart presents today's measurements water level in centimeters.

Data source: Slovak Hydrometeorological Institute (SHMU)



Figure 5: Example of Water Level Report

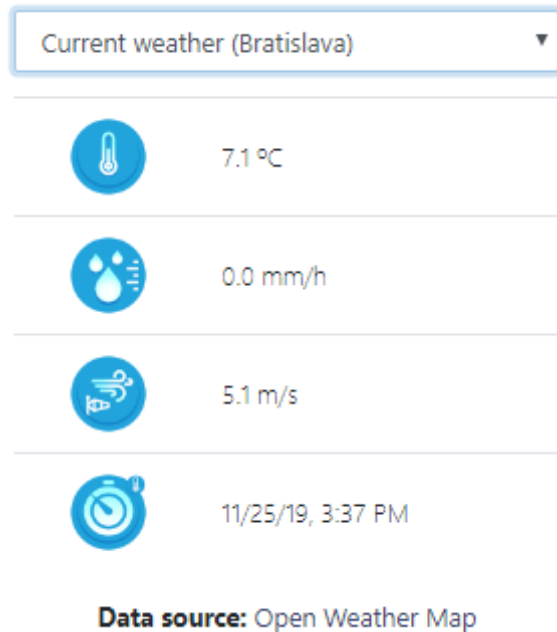


Figure 6: Example of Current Weather Report

To produce these reports the Portal accesses the EMC API, described in 4.2.2, below.

The FLOOD-serv Portal acts also as a citizen facing interface of the CDF Component in relation to citizen users. Thus the section [ENGAGE WITH PUBLIC AUTHORITIES](#) contains the Citizen Involvement Form where citizens can signal issues to public authorities and engage in dialogue with them. To allow citizens to submit information in the Citizen Involvement Form, generate the Submission History report, or Write Messages, the Portal accesses the CDF API, as presented in D4.5 APPENDIX I.

4.2.2 Emergency Management Console

EMC consumes data from the Social Media Component (SMC). Specifically, the data consumed from the SMC are the postings belonged to a specific region of interest.

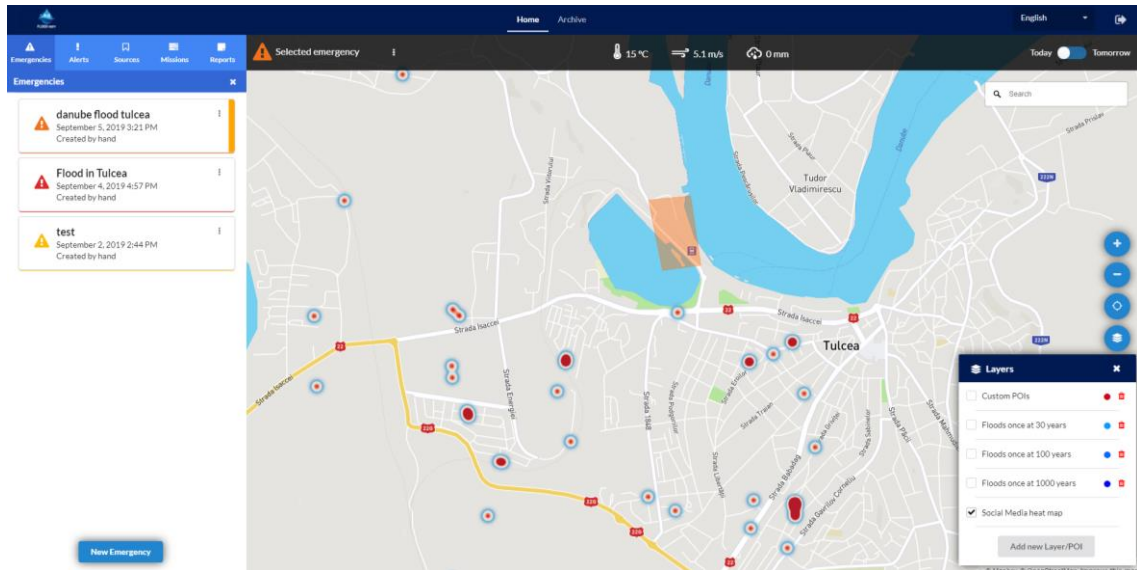


Figure 7: Heat map with the social postings (In layers)

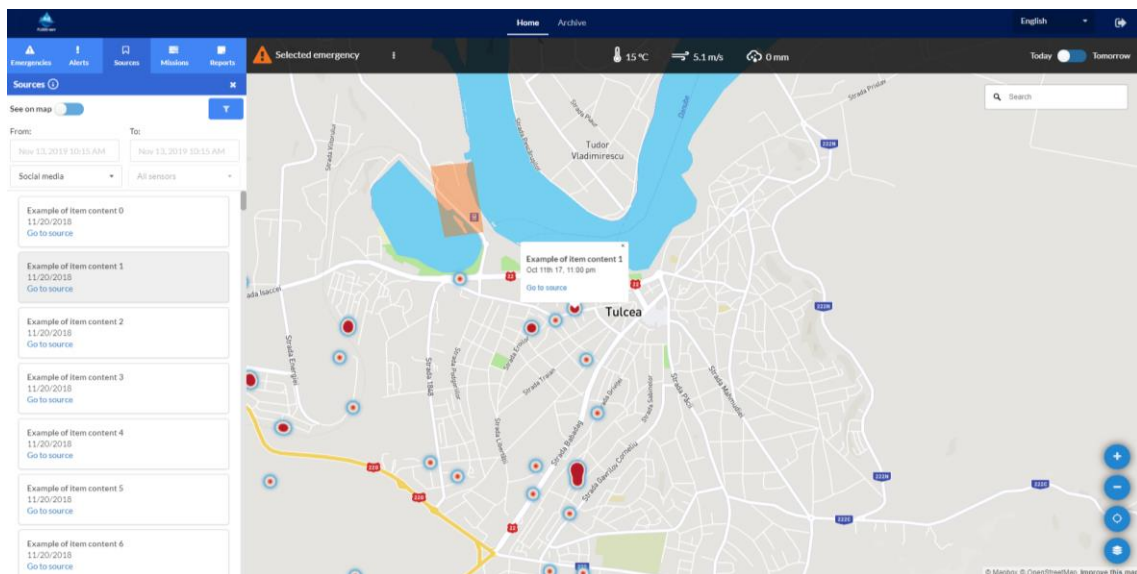


Figure 8: Social postings as data source

Nevertheless, after Cellent withdrawn, we lost the permission to access to such data. From such moment, we decided to simulate such input in the EMC. So in the current version of the EMC, the social map is a simulation.

On the other hand, the EMC also acts as data producer for the portal. The data provided by the EMC to the portal is:

- Sensor data
- Data related to flood situation

- A list of flood warnings
- Past flood events
- Reports sent by the Emergency Responder
- Missions sent to the ER
- Flood management plan
- Social media content

You can check the EMC API in the Appendix I.

4.2.3 Social Media Component


The SMC delivers data only, it does not consume data from other components of the FLOODSERV system.


The SMC integrates a variety of incoming data (messages) for further analysis, namely

- a) Social media (Facebook, Twitter)
Typically social media sites of the cities, first responders or politicians
- b) Newspapers (RSS)
- c) Opinion Maps
They allow a geo-located response and can be embedded in any website or social media channel.
- d) Questionnaires
They allow a structured response. As is the case with the opinion maps they can also be embedded in any website or social media channel

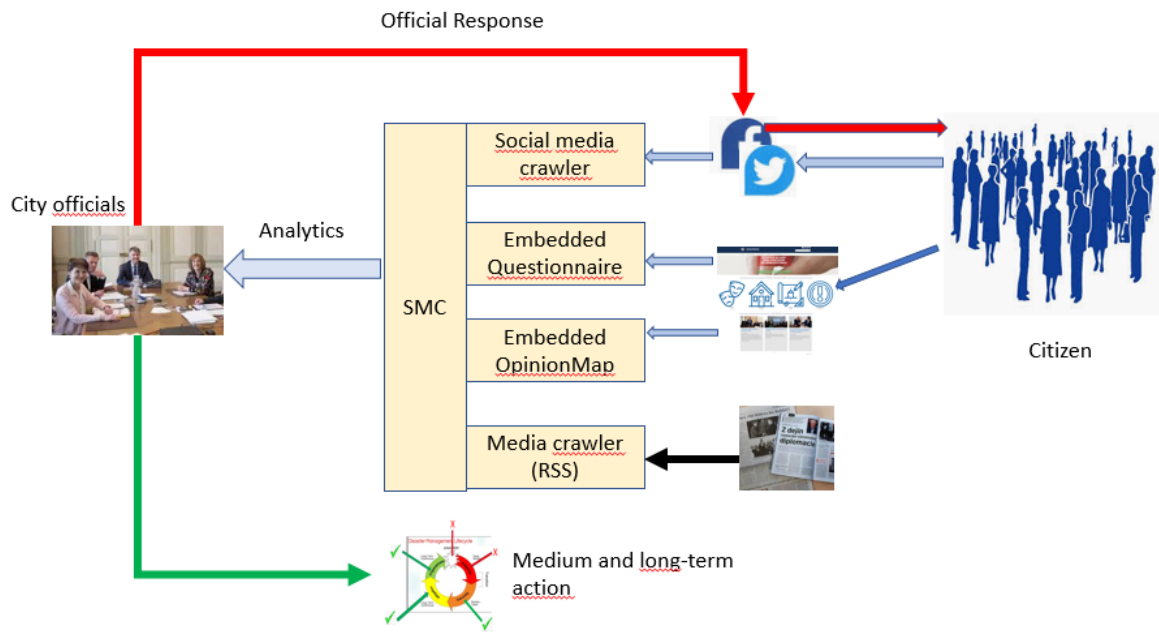
Both c) and d) are complementing the CDF, the EMC and the related mobile applications.

The picture below shows how the SMC is embedded in the overall citizen communication for flood risks.

 Reactions and comments are collected from various channels and aggregated for analysis.

 The city officials analyze and subsequently respond directly to social media. The response could be for example communication on any medium or long-term action planned and implemented

 Actual implementation resulting from the analysis



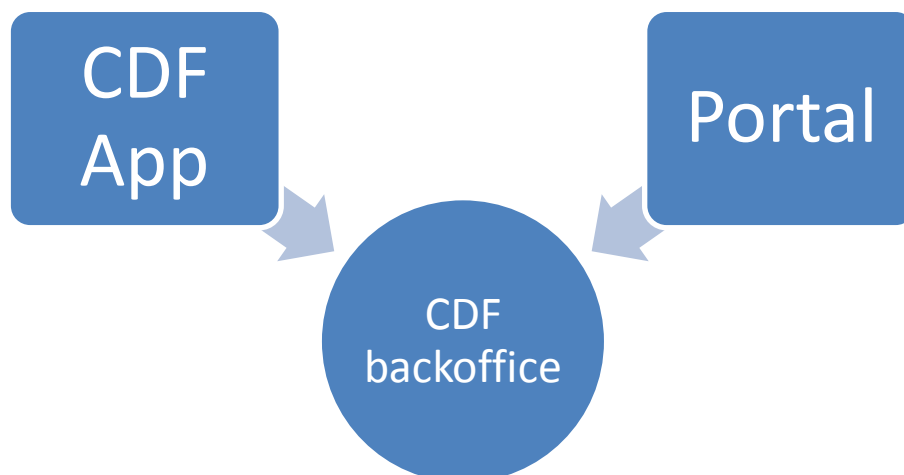
Technically this refers to the retrieval of data stored in the SMC database from other software components.

Data retrieval is based on REST and the respective output is a JSON string with all relevant parameters of a posting.

REST stands for “Representational State Transfer”. It is a set of rules that developers follow when they create their API. One of these rules states that you should be able to get a piece of data (called a resource) when you link to a specific URL. Each URL is called a request while the data sent back to you is called a response.

4.2.4 Citizen Direct Feedback Component

The CDF, as a two-way communication component, acts as producer and a received of data between the portal and the CDF App.



The following links are examples for Bilbao. As shown in the attached file with the full description of the CDF API, the CDF has a different instance per pilot, so each pilot's API has its own link.

As input, the CDF database receives:

1. Issues reported by Citizens and other stakeholders, in the Portal and CDF mobile application:

<https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/report> - check Chapter 4.2 of the CDF API Documentation.

2. Messages created in the Portal, also via integration API:

<https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/createNewWebMessage> - check Chapter 5.3 of the CDF API Documentation.

As output, the CDF component sends:

1. Feedback on the reported issues and full details of the reported issues and created processes (upon approved issues);

a. This allows to integrate the component with external systems. In the case of Genova a integration with the SIG system is planned:

For all the issues reported: <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/getIssues> – check Chapter 4.4 of the CDF API Documentation.

For the processes created: (the approved issues): <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/records/getProcessesByDate/{date}>

2. Messages created in the Backoffice for a specific citizen to the portal;

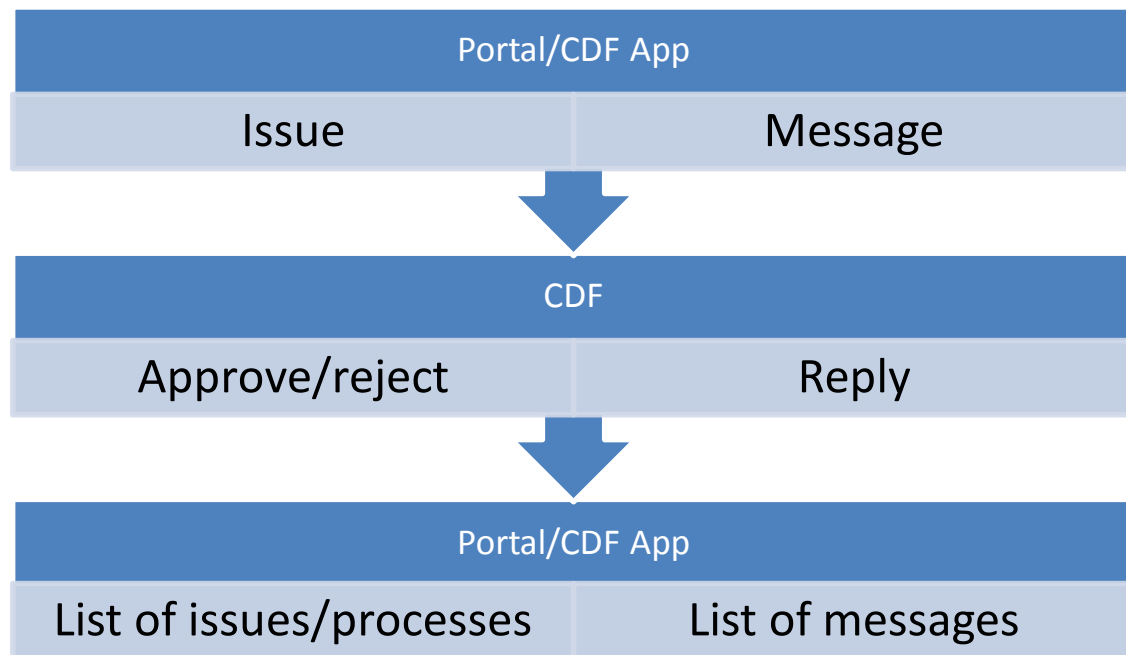
For sent messages: <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/getSentWebMessagesByUser> – check Chapter 5.1 of the CDF API Documentation.

For received messages: <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/getSentWebMessagesByUser>

– check Chapter 5.2 of the CDF API Documentation.

3. Broadcast messages to the mobile application;

<https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/floodserv/createNewWebMessage>



The API of the CDF also allows to export/access the following data.

- **User/Entities**
 - <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/persons/> getEntities – check Chapter 3.7 of the CDF API Documentation.
- **Processes**
 - <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/records/getProcessesByDate/{date}> - check Chapter 3.1 of the CDF API Documentation.
- **Attachments of the Process**
 - https://bilbao-floodserv-saas.ano.pt/bilbao/pilot_contextroot_name/services/api/attachments/getAttachments/{id} - check Chapter 3.2 of the CDF API Documentation.
- **Movements of the Process**
 - <https://bilbao-floodserv-saas.ano.pt/bilbao/services/api/movements/getMovementsByProcess/{id}> - check Chapter 3.5 of the CDF API Documentation.

For the rest, of the API's methods, please consult the CDF API Documentation.

4.2.5 Territory Management System

The TMS receives images as input and exports the results for the PORTAL, EMC and other components through its API – Please consult Territory Management System API document.

The method to calculate depth is available in the API and returns the calculated depth of the available image. A method to list all the previous analysis is also available.

4.2.6 Sensors and External Data

Sensor data used in the FLOOD-serv System is of two general kinds.

1. Internal Sensors: Sensors implemented during the FLOOD-serv Project by members of the Consortium (namely DDNI and BSK);
2. External Sensors: Sensor data from other sources (usually already in a database and served through web services).

Both types of sensor data are consumed by the EMC, the first by accessing directly the data from the sensor server, the second by accessing the data using the API services of external data providers.

4.2.6.1 Internal Sensors

DDNI purchased a sensor that was mounted on a concrete foundation support installed on the bank of the Danube cliff, near the institute pontoon.

The information transmitted by the sensors is received on a server installed at the DDNI headquarters. The server is sending the collected data in txt format on a ftp site (file transfer protocol) from where Answare can further received it.

The whole system is configured as follows:

The system for monitoring the level of water and temperatures in the well, hereinafter referred to as the HidroMon or HidroMon system, ensuring centralization over the Internet or VPN network, can be made available in case of water temperature in drilling, for humidity and temperature in the AMR module box. They can be centralized from maximum 10,000 measurement points (drilling).

The HidroMon system is composed of:

1. Measuring points (holes) containing:
 - a. A level sensor that measures the pressure of the pressure column above it and the water temperature.
 - b. An AMR module (Automatic meter reading - Automatic sensor reader automatically) that provides:
 - i. periodic reading of the level sensor; it reads: the pressure of the water column and the measuring range of the sensor and high water is calculated above the sensor; based on the height of the water column and the distance from the water, measured the installation, calculates the current distance of the power of the well (water area, in meters:
 - ii. water temperature, in Celsius degree;
 - iii. water member, Na (m)
 - iv. water temperature, Ta (° C);
 - v. internal humidity (inside the housing), Good (%);
 - vi. internal temperature (inside the housing), Ti (° C);
 - vii. battery voltage change, Vbat (V)
2. HidroMon server maintenance program tracking functions: Retrieving the archives from the AMR module and saving them in logs. The program creates, in the DataPm folder, for each measurement point, to provide a file with the name of the given module of the AMR. In this folder he creates the folders: Alarms, Config, Logs and Reports.

Each AMR module is connected to GPRS on the daily Hidromon server, at the current time for data transmission is recorded in the archives.

Technical specification for the Datalogger are as follow:

Table 3: Sensor Technical Specifications

Technical specifications	
Modem GSM/GPRS	Quadband, frequency 850/900/1800/1900 MHz
SIM Card	Supports SIM card GPRS / GSM data
Antenna	Cable antenna, planar antenna
Transmission	M2M protocol, GSM / GPRS data, transmission to the FTP server
Housing	Stainless steel (316L), shock resistance, vibration and high humidity
Antenna connector	FME (male), optional adapter
Transfer Interface and connection to the PC	USB standard , optional wireless , RS232 , radio
Power supply	alkaline batteries 2 X 1.5V , optional adapter for 220V or solar panel
Operating temperature	-30°C 85°C
Mechanical protection	Standard IP67 with closed protective cover and sensor connected, water and dust, IP 68 for 24 hours water and dust
External translator connection	input signal 4....20 mA, RS485 , RS232, SDI-12, Modbus, CAN
Data memory	up to 500,000 sets of measured values, non-volatile, data remains stored even without battery, each measured value is correlated with time and date
Server	database management, online data visualization, online and offline configuration
Registered measures	liquid level (pressure), liquid temperature, humidity and internal temperature protection housing, battery voltage, signal power, memory space, data transfer date
Data export format	ASCII, CSV, XLS, TSV
Data interrogation	Automatic data interrogation, transmission at preset intervals
Transmitted data access security	2 separate levels with username and password
Alarms	local / remote configuration, transmission via SMS or email. Two configurable thresholds, maximum and minimum
Configuration	size recording interval, data transmission interval, system name, location, two modes of calibration (groundwater measurement, watercourse measurement) ,definition of alarm levels with warning

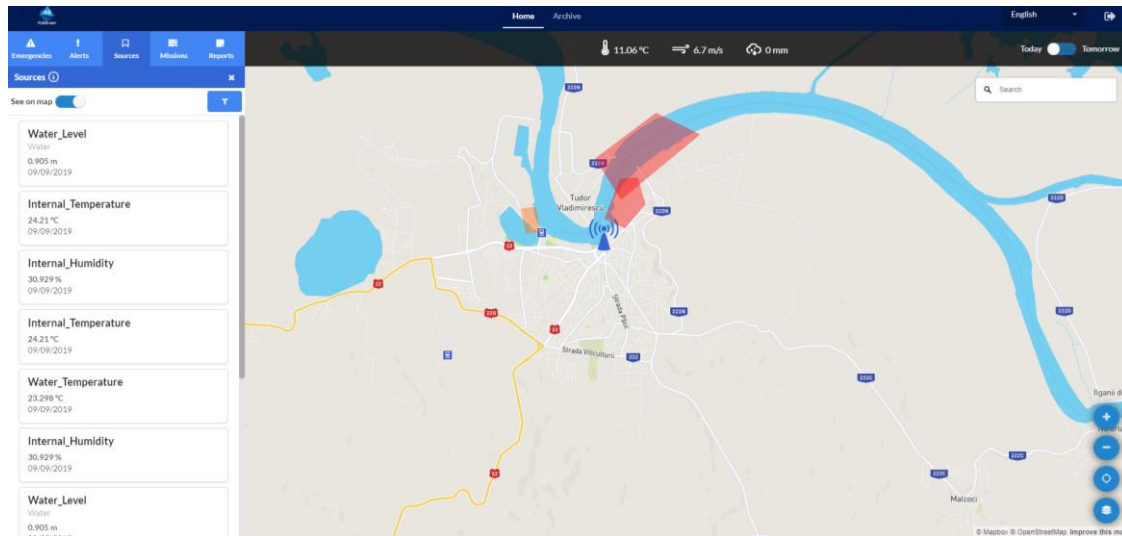
D4.5 System Integration

Other standard specifications	Diameter 55 mm, length 600 mm; Data transfer possible without stopping the recording and without removing it from the installation location;Autonomie baterie pana la 5 ani la o inregistrare per parametru pe zi si transmisie date la 24 ore; remote configuration and modification of parameters;Continuous temperature and humidity monitoring in the software; Software available for parameter configuration, data transfer, offline or online graphical and table visualization ;Optional software module for data transfer from the server in custom applications Optional data acquisition software and programming / configuration (OS WIN7, WIN8, WIN10) with graphical and table representation
Technical specifications for level and temperature transducer	
Pressure measurement	
Pressure measurement field	0 ...250 Mh20, 0...25 bar , special or custom domains
Precision	0.03% from the measured domain (FS)
Resolution	0.001 m
Compensation of atmospheric pressure	realised in the transmitter
Resoponse time	< 1ms from 10...90%
Over pressure	3bar/ 3 X FS (≥ 3 bar)/3 X FS
Temperature measurement	
Temperature measurement range	-5...60°C
Precision	0.5°C
Resolution	0.1°C
General specifications	
Operating temperature/storage	-5...+80°C/-30°C....+80°C
Mechanical protection	IP68
Output signal	RS485, Modbus, optional SDI-12, RS232, CAN , 4..20 Ma
Outercase	stainless steel (316L)
Data transfer cable and support	Standard PUR , optional FEP, PE, up to 400 m
Calibration certificate	optional
Standrads and certifications	
EN 60068-2-6	Vibrations level : 10gh (4....2000Hz/ ± 10 mmpp)
EN 60068-2-27	Shocks level : 100g (pulse duration 6 ms)

In the case of Tulcea the EMC uses data coming from:

[DDNI](#): Danube Delta National Institute for Research and Development

- The data consumed from them are related to:
- water level of Danube river in Tulcea
- Internal temperature of Danube river in Tulcea
- Internal humidity of Danube river in Tulcea



Sensors data integrated in the EMC for Tulcea

Using this information, the different rules implemented in the EMC DSS for the situation of today¹ in Tulcea are:

- If the hourly measured water level (Tulcea) < 220 cm
- If the hourly measured water level (Tulcea) is inside [220, 410) cm
- If the hourly measured water level (Tulcea) is inside [410, 440) cm
- If the hourly measured water level (Tulcea) => 440 cm

4.2.6.2 External Sensors and Data

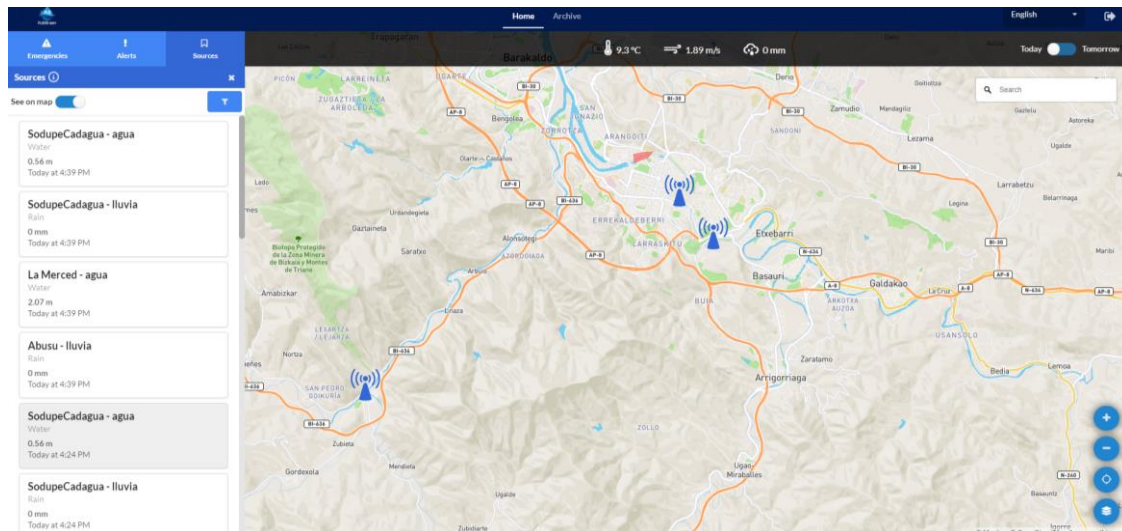
The EMC consumes sensor data coming from different sources.

In the case of Bilbao, the EMC integrates sensor data coming from the open data portal [GeoEuskadi](#), which is a partner of the EFAS System and satisfies the EU INSPIRE Directive and all the National Directives to publish geo-referenced data as part of the “Spatial Data Infrastructure”.

The data consumed in Bilbao are:

- Water level of the river in Sodupe
- Water level of the river in Bilbao
- Rain level in Sodupe
- Rain level in Bilbao

¹ To implement the rules in Tulcea we have followed the recommendations made by DDNI



Sensors data integrated in the EMC for Bilbao

Using this information, the different rules implemented in the EMC Decision Support System (DSS) for the situation of today² in Bilbao are:

- If the hourly measured rain (Nervion-Ibaizabal|Cadagua-Gueñes) is < 15 l/m² Then Activate the green emergency level
- If the hourly measured water level (Nervion-Ibaizabal) < 4 m. Then Activate the green emergency level
- If the hourly measured water level (Cadagua-Gueñes) < 1.60 m. Then Activate the green emergency level
- If the hourly measured rain (Nervion-Ibaizabal|Cadagua-Gueñes) is inside $[15, 30)$ l/m² Then Activate the yellow emergency level
- If the hourly measured water level (Nervion-Ibaizabal) is inside $[4, 4.3)$ m. Then Activate the yellow emergency level
- If the hourly measured water level (Cadagua-Gueñes) is inside $[1.60, 1.90)$ m. Then Activate the yellow emergency level
- If the hourly measured rain (Nervion-Ibaizabal|Cadagua-Gueñes) is inside $[30, 60)$ l/m² Then Activate the orange emergency level
- If the hourly measured water level (Nervion-Ibaizabal) is inside $[4.3, 4.5)$ m. Then Activate the orange emergency level
- If the hourly measured water level (Cadagua-Gueñes) is inside $[1.90, 2.30)$ m. Then Activate the orange emergency level
- If the hourly measured rain (Nervion-Ibaizabal|Cadagua-Gueñes) $=> 60$ l/m² Then Activate the red emergency level
- If the hourly measured water level (Nervion-Ibaizabal) $=> 4.5$ m. Then Activate the red emergency level
- If the hourly measured water level (Cadagua-Gueñes) $=> 2.30$ m. Then Activate the red emergency level

And for tomorrow¹:

² To implement these rules we are use the official document: <https://1drv.ms/b/s!ApsUzqT-3ft-01fcOYL7uL-7GR6?e=dWnjml>

D4.5 System Integration

- If the daily forecasted rain (Nervion-Ibaizabal|Cadagua-Gueñes) is < 15 l/m² Then Activate the green emergency level
- If the daily forecasted rain (Nervion-Ibaizabal|Cadagua-Gueñes) is inside $[15, 30)$ l/m² Then Activate the yellow emergency level
- If the daily forecasted rain (Nervion-Ibaizabal|Cadagua-Gueñes) is inside $[30, 60)$ l/m² Then Activate the orange emergency level
- If the daily forecasted rain (Nervion-Ibaizabal|Cadagua-Gueñes) $\Rightarrow 60$ l/m² Then Activate the red emergency level

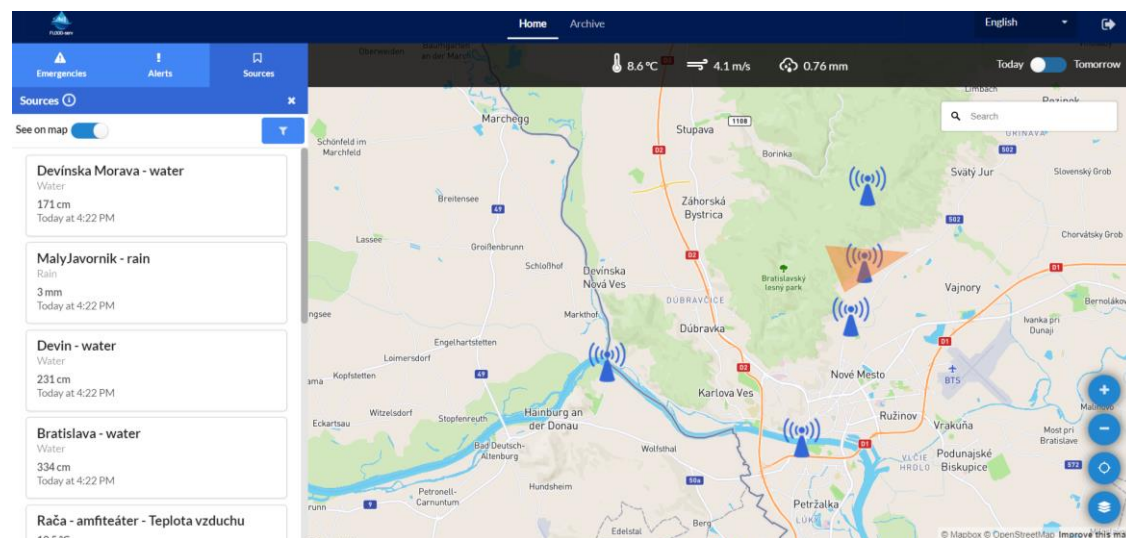
For the rest of pilots, unfortunately there are not any open data sources which can be integrated directly in the EMC. Nevertheless, each pilot city has provided different official web pages which can be used to extract the published data to be integrated in the EMC.

In the case of Bratislava the EMC uses data coming from:

- [SHMU](#): The Slovak Hydrometeorological Institute, which is a data provider of the EFAS System.
- BSK municipality

The data consumed in Bratislava are related to:

- Water level of the Morava river in Devin
- Water level of the Danube river in Bratislava
- Rain level in Raca



Sensors data integrated in the EMC for Bratislava

Using this information, the different rules implemented in the EMC DSS for the situation of today in BSK³ are:

- If the hourly measured water level (Danube) $< 6,5$ m. Then Activate the green emergency level

³ To implement the rules in BSK we have followed the recommendation made by BSK and Exdwarf

D4.5 System Integration

- If the hourly measured water level (Morava) < 4,2 m. Then Activate the green emergency level
- If the hourly measured water level (Danube) is inside [6.5, 7.5) Then Activate the yellow emergency level
- If the hourly measured water level (Morava) is inside [4.2, 4.6) Then Activate the yellow emergency level
- If the hourly measured water level (Danube) is inside [7.5, 8.5) Then Activate the yellow emergency level
- If the hourly measured water level (Morava) is inside [4.6, 5.2) Then Activate the yellow emergency level
- If the hourly measured water level (Danube) > 8,5 m Then Activate the red emergency level
- If the hourly measured water level (river) > 5,2 m Then Activate the red emergency level
- If the hourly measured rain (Raca) < 5 l/m² Then Activate the green emergency level
- If the hourly measured rain (Raca) is inside [5, 20) l/m² Then Activate the yellow emergency level
- If the hourly measured rain (Raca) is inside [20, 50) l/m² Then Activate the orange emergency level
- If the hourly measured rain (Raca) => 50 l/m² Then Activate the red emergency level

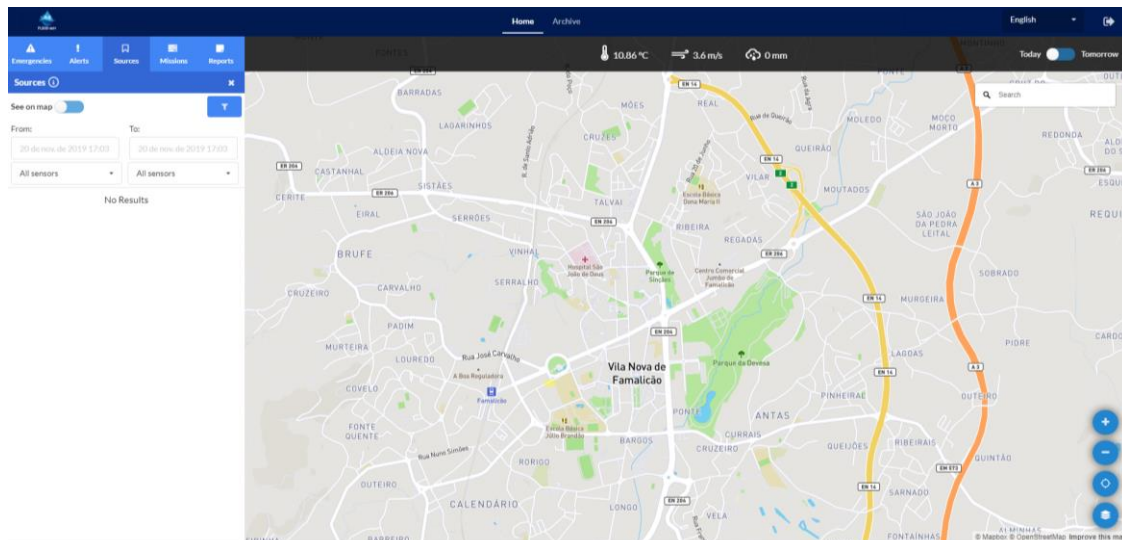
And for tomorrow³:

- If the daily forecasted rain (Raca) < 5 l/m² Then Activate the green emergency level
- If the daily forecasted rain (Raca) is inside [5, 20) l/m² Then Activate the yellow emergency level
- If the daily forecasted rain (Raca) is inside [20, 50) l/m² Then Activate the orange emergency level
- If the daily forecasted rain (Raca) => 50 l/m² Then Activate the red emergency level

In the case of Vila Nova the EMC uses data coming from:

- [IPMA](#): The Portuguese Institute of the Sea and the Atmosphere, which is a partner of the EFAS System. In this case, there are not sensors available. Instead of sensor data, we are using the official information about the daily flood situation in Vila Nova provided by IPMA portal. Therefore, the rules implemented in the EMC DSS for this pilot are the following:
 - If Flood situation == green Then Activate the green emergency level
 - If Flood situation == yellow Then Activate the green emergency level
 - If Flood situation == orange Then Activate the orange emergency level
 - If Flood situation == red Then Activate the red emergency level

D4.5 System Integration



There are not sensor data integrated in the EMC for Tulcea

In [this excel](#) it can be checked the different sources from where the EMC obtains the data as well as the kind of sensor, sensors location, unit of measurement, etc.

5 Conclusions

This document presented and documented the Integration of the FLOOD-serv System. The main content is grouped in Chapters 3 and 4. In Chapter 3 we gave an overview of the FLOOD-serv System and its high level architecture. In Chapter 4 we present in detail the integration of components under two aspects: SSO and Role Mapping, and Data Integration. To achieve SSO and Role Mapping integration, Keycloak technology and OpenID Connect and OAuth 2.0 protocol standards were used. For data integration components access each other (depending on data needs dictated by functional requirements) usually based on APIs.

The Integration of the FLOOD-serv system posed some technical and logistical problems for the FLOOD-serv Consortium, particularly the Technical Partners. Bilateral and multilateral coordination efforts needed to be made by partners at both technical and business level. However, the challenges were met by the Consortium and integration work was finalized to be ready for Piloting.

6 References

APPENDIX I: API Documentations

Appendix includes additional information (if applicable) at the end of the deliverable. If more than one is necessary, "Appendices" are listed separately. They support the text, although they include less important information (graphics, tables, images, questionnaires, etc.) that the reader may refer to if he wants.

Citizen Direct Feedback API

Introduction

The following chapters identify the methods present in the three main areas of the CDF API. For the URL, each pilot has its own CDF instance:

<https://bilbao-floodserv-saas.ano.pt/>

<https://bratislava-floodserv-saas.ano.pt/>

<https://genova-floodserv-saas.ano.pt/>

<https://tulcea-floodserv-saas.ano.pt/>

<https://vnfamalicao-floodserv-saas.ano.pt/>

For the API link, they obey the same logic:

`https://{pilot_instance_name}/{pilot_contextroot_name}/services/api/records/`

Pilot	{pilot_instance_name}	{pilot_contextroot_name}
Bilbao	bilbao	bilbao
Bratislava	bratislava	bratislava
Genova	genova	genova
Tulcea	tulcea	tulcea
Vila Nova de Famalicão	vnfamalicao	vnfamalicao

For example, for the **GetProcessesByDate** of the **STATES** API, for Genova the link is:

<https://genova-floodserv-saas.ano.pt/genova/services/api/records/getProcessesByDate/>

SYNC Users

]
--	---

Get the list of attachments of a specific process

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/attachments/
Path	getAttachments/{id}
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => process identifier
Return	[{ "id": <attachment identifier>, "creationDate": "<date created in milliseconds>", "name": "<file name>" }]

Get the last version of file of a specific attachment

Link	https://{pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/attachments/
Path	getFile/{id}
Method	GET
Produces	application/octet-stream
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => attachment identifier
Return	The file

Get the last version of file of a specific attachment (in base64)

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/attachments/
Path	getFileBase64/{id}
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => attachment identifier
Return	{ "data": "<base64 encoded file content>" }

Get list of movements of a process

Link	https://{pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/movements/
Path	getMovementsByProcess/{id}
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => process identifier
Return	[{ "id": <movement identifier>, "number": <movement number>, "creationDate": "<date created in milliseconds>",&br/> "userOrigin": "<origin user>",&br/> "userDestination": "<destination user>",&br/> "departmentOrigin": "<origin department>",&br/> "departmentDestination": "<destination department>",&br/> "resolutionDate": "<resolution date in milliseconds>",&br/> "resolutionDescription": "<resolution description>" },

]
--	---

Update the status of a specific process

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/records/
Path	updateState/{id}/{status}
Method	POST
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => process identifier status => the new process state
Return	A boolean indicating success or failure

Get a list of entities

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/persons/
Path	getEntities
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Return	[{ "id": <entity identifier>, "name": "<entity name>",&br/> "number": "<entity number>" }, ...]

Get full details on a specific entity

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/persons/
Path	getCompleteEntity/{id}
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw=="
Parameters from path	id => entity identifier
Return	<pre>{ "id": "<entity identifier>", "name": "<entity name>", "number": "<entity number>", "email": "<entity e-mail>", "phoneNumber": "<entity phone number>", "birthday": "<entity birth date in milliseconds>", "address": { "id": "<address identifier>", "street": "<street>", "district": "<district>", "county": "<county>", "town": "<town>", "postalCode": "<postalCode>" } }</pre>

Base Data

Available process states

- R - Registry
- P - Pending
- A - Archived
- UA - Unarchived
- S - In follow-up
- DR- Draft

WEB REQUESTS

Login User

Link	http://195.82.131.198/oauth2_server/public/index.php
Path	api/login
Tip	POST
Parameter from body	email* password*
Return	"token_type" "expires_in " "access_token

Report Issue

Link	https:// {pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/floodserv/
Path	report
Method	POST
Consumes	multipart/form-data
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZ1dHVyZWVvYw==" token => the Oauth2 access token
Parameters from body	title => title of the issue description => description of the issue latitude => latitude (location) longitude => longitude (location) attachments => the images and videos, as a list of multipart attachments named "attachment1", "attachment2", etc.
Return	{ "newId": "<internal ID of the created issue/request>", "state": "SENT" }


```
"description": "<issue description>",
"dateSent": "<send date (number of milliseconds since January
1, 1970, 00:00:00)>"
},
"state": {
  "state": "<state of the request>",
  "processNumberDisplay": "<created process number display>",
  "stateMessageId": "<string ID of the state of the request for
Android>",
  "resolutionMessageId": "<string iD of the resolution for
Android>",
  "resolutionDate": "<resolution date (number of milliseconds
since January 1, 1970, 00:00:00)>"
},
"location": {
  "latitude": "<latitude>",
  "longitude": "<longitude>"
},
"attachments": {
  "count": "<number of attachments>",
  "attachments": [
    {
      "id": "<internal ID of the attachment>",
      "originalName": "<original file name>",
      "size": "<file size (bytes)>",
      "dateSent": "<send date (number of milliseconds since
January 1, 1970, 00:00:00)>"
    },
    ...
  ]
},
...
]
```

Download Attachment

Link	https://{pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/floodserv/
Path	getAttachment/{id}
Method	GET
Produces	application/octet-stream
Parameters from headers	Authorization => "Basic VVNFUldTOmZ1dHVyZWVvYw==" token => the Oauth2 access token
Parameters from query	Id => internal ID of the attachment
Return	The file

6.1.1.1 Base Data

6.1.1.1.1 Possible Values for "state"

- DRAFT
- SENT
- PREPARATION
- ACCEPTED
- ACCEPTED ARCHIVED
- ACCEPTED DEFERRED
- ACCEPTED REJECTED
- ACCEPTED DEFERRED ARCHIVED
- ACCEPTED REJECTED ARCHIVED
- ERROR

6.1.1.1.2 Possible values for "stateMessageId"

- issue_state_full_DRAFT
- issue_state_full_SENT
- issue_state_full_PREPARATION
- issue_state_full_NOTACCEPTED
- issue_state_full_ACCEPTED
- issue_state_full_ACCEPTED_ARCHIVED
- issue_state_full_ACCEPTED_DEFERRED
- issue_state_full_ACCEPTED_REJECTED
- issue_state_full_ACCEPTED_DEFERRED_ARCHIVED

- issue_state_full_ACCEPTED_REJECTED_ARCHIVED
- issue_state_full_ERROR

Possible values for “resolutionMessageId”

- issue_resolution_d
- issue_resolution_r

WEB MESSAGES

Messages Received

Link	<code>https://{pilot_instance_name}-floodserv-saas.ano.pt/{pilot_contextroot_name}/services/api/floodserv/getSentWebMessagesByUser</code>
Path	<code>api/floodserv/getSentWebMessagesByUser</code>
Type	GET
Parameters from headers	token
Return	JSON (application/json) <pre>{ "count": 2, "list": [{ "saveEnabled": true, "id": 1447, "entryDate": "11-06-2019 14:17:12", "type": "NOR", "wmePrioridade": 0, "title": "test", "details": "test_details", "from": "USER", "to": "APP", "haveAttachs": "NO", "dataOrigin": "GSE_R4", "read": false }] }</pre>

```
    },  
    {  
      "saveEnabled": true,  
      "id": 1446,  
      "entryDate": "07-06-2019 17:03:29",  
      "type": "NOR",  
      "wmePrioridade": 0,  
      "title": "OK",  
      "details": "OK",  
      "from": "USER",  
      "to": "APP",  
      "viewDate": "07-06-2019 17:04:05",  
      "haveAttachs": "NO",  
      "read": true  
    }  
  ]  
}
```

I/O:

@GET

@Path("/getSentWebMessagesByUser")

@Produces(MediaType.APPLICATION_JSON)

PaginationModel<WebMessage> getSentMessages(
 @HeaderParam("token") String token,
 @DefaultValue("0") @QueryParam("offset") Integer offset,
 @DefaultValue("10") @QueryParam("limit") Integer limit,
 @DefaultValue("-entryDate") @QueryParam("orderBy") String orderBy,
 @QueryParam("filter") String filter
);

Messages Sent

Link	https://{pilot_instance_name}-floodserv-saas.ano.pt /{pilot_contextroot_name}/services/api/floodserv/getSentWebMessagesByUser
------	--

Path	api/floodserv/getSentWebMessagesByUser
Type	GET
Parameters from headers	token
Return	<p>JSON (application/json)</p> <pre> { "count": 2, "list": [{ "saveEnabled": true, "id": 1447, "entryDate": "11-06-2019 14:17:12", "type": "NOR", "wmePrioridade": 0, "title": "test", "details": "test_details", "from": "USER", "to": "APP", "haveAttachs": "NO", "dataOrigin": "GSE_R4", "read": false }, { "saveEnabled": true, "id": 1446, "entryDate": "07-06-2019 17:03:29", "type": "NOR", "wmePrioridade": 0, "title": "OK", "details": "OK", "from": "USER", "to": "APP", "viewDate": "07-06-2019 17:04:05", "haveAttachs": "NO", </pre>

	<pre> "read": true }] }</pre>
--	---

Note: The list can be filter as the following example: https://{pilot_instance_name}-floodserv-saas.ano.pt/{pilot_contextroot_name}/services/api/floodserv/getSentWebMessagesByUser?offset=0&limit=1.

I/O:

@GET

@Path("/getSentWebMessagesByUser")

@Produces(MediaType.APPLICATION_JSON)

PaginationModel<WebMessage> getSentMessages(
 @HeaderParam("token") String token,
 @DefaultValue("0") @QueryParam("offset") Integer offset,
 @DefaultValue("10") @QueryParam("limit") Integer limit,
 @DefaultValue("-entryDate") @QueryParam("orderBy") String orderBy,
 @QueryParam("filter") String filter
);

Create New Message

Link	https://{pilot_instance_name}-floodserv-saas.ano.pt/{pilot_contextroot_name}/services/api/floodserv/createNewWebMessage
Path	api/floodserv/createNewWebMessage
Type	POST
Parameters from headers	token
Parameters from body	JSON (application/json) { "title" : "test", "details" : "test_details" }
Return	JSON (application/json) {


```
"saveEnabled": true,  
"id": 1447,  
"entryDate": "11-06-2019 14:17:12",  
"type": "NOR",  
"wmePrioridade": 0,  
"title": "test",  
"details": "test_details",  
"from": "USER",  
"to": "APP",  
"haveAttachs": "NO",  
"dataOrigin": "GSE_R4",  
"read": false  
}
```

I/O:

@POST

@Path("/createNewWebMessage")

@Consumes(MediaType.APPLICATION_JSON)

@Produces(MediaType.APPLICATION_JSON)

WebMessage createNewWebMessage(
 @HeaderParam("token") String token,
 WebMessage wmsg);

Territory Management System API

Introduction

The following chapters identify the methods present in the three main areas of the TMS API. For the URL, each pilot has its own TMS instance:

<https://bilbao-floodserv-saas.ano.pt/>

<https://bratislava-floodserv-saas.ano.pt/>

<https://genova-floodserv-saas.ano.pt/>

<https://tulcea-floodserv-saas.ano.pt/>

<https://vnfamalicao-floodserv-saas.ano.pt/>

For the API link, they obey the same logic:

`https://{pilot_instance_name}/{pilot_contextroot_name}/services/api/records/`

Pilot	{pilot_instance_name}	{pilot_contextroot_name}
Bilbao	bilbao	bilbao
Bratislava	bratislava	bratislava
Genova	genova	genova
Tulcea	tulcea	tulcea
Vila Nova de Famalicão	vnfamalicao	vnfamalicao

Depth Analysis

Get a list of previous analysis filtered by date of creation

Link	<code>https://{pilot_instance_name}-floodserv-saas.ano.pt/{pilot_contextroot_name}/services/api/floodserv/</code>
Path	<code>getDepthByDate/{date}</code>
Method	GET
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZsb29kc2VydjEyMw==" token => the Oauth2 access token
Parameters from path	date => date in milliseconds
Return	[{ "id": <identifier>, "depth": <depth>, "creationDate": "<date created in milliseconds>",&br/> "processedBy": "<username>,"

	<pre> }, ...] </pre>
--	-----------------------

Create Analysis

Link	https:// saas.ano.pt/{pilot_instance_name}-floodserv- saas.ano.pt/{pilot_contextroot_name}/services/api/floodserv/
Path	calculateDepth
Method	POST
Consumes	multipart/form-data
Produces	application/json
Parameters from headers	Authorization => "Basic VVNFUldTOmZ1dHVyZWRvYw==" token => the Oauth2 access token
Parameters from body	image => image
Return	{ "newId": "<internal ID of the created analysis >", "depth": "<calculated-depth>" }

EMC API

Regarding to the EMC API, we have different URLs to each API pilot. Below you can find the data related to Bilbao pilot (although the API is the same for all pilots). The IP is https://backend.{pilot_name}.floodserv.answare-tech.com. Pilot name could be:

1. bilbao
2. bratislava
3. tulcea
4. vilanova

Using Bilbao as example, the endpoints of the EMC are:

Sensor data:

Sensor list: <https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/sensors/>

Sensor data: https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/sensors/{sensor_id}/data/today/

Data related to flood situation:

D4.5 System Integration

its Points of Interest (PoIs): <https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/custompoi/?type=School> (the types are Hospital, School, other)

its daily measured weather

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/weather/>

Its daily weather forecast for next days:

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/weather/forecast/>

Its daily flood situation: <https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/emergencies/current/> (now returns an array of emergencies)

A list of flood warnings:

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/warning/>

Past flood events:

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/emergencies/>

Reports sent by the Emergency Responder (ER):

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/reports/> (now returns reports associated to active emergencies)

Parameters:

confirmed: optional. Values: 1: returns only confirmed reports of active emergencies. 0: returns only unconfirmed reports of active emergencies.

emergency_id: optional: returns the missions associated to the emergency with this id.

Example of use <https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/reports/?confirmed=1>

Missions sent to the ER:

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/missions/> (now returns missions associated to active emergencies)

Parameters:

confirmed: optional. Values: 1: returns only confirmed missions of active emergencies. 0: returns only unconfirmed missions of active emergencies.

emergency_id: optional: returns the missions associated to the emergency with this id.

Example of use https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/missions/?emergency_id=1&confirmed=0

Flood management plan:

<https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/documents/> .

https://backend.bilbao.floodserv.answare-tech.com/api/emc/v1.0/documents/{document_id}

Social media content:

<https://socialmedia.cellent.at/fupol-services/rest/public/Campaign/findPosts?source=OpinionMap&latitudeFrom=43.2&longitude>

D4.5 System Integration

From=-3&latitudeTo=43.35&longitudeTo=-2.5 (in order to access to this data you need to request access to Peter.Sonntagbauer@cellent.at)

All the data related to regions of interest and point of interest produced by the EMC follow the geoJSON format.

For each pilot, there is a mandatory header: the Authorization Bearer <API KEY>

Pilot	API KEY
Bilbao	a06d2358cf995a595a16faed142304f9fd2f024e
Bratislava	0209bb403da8d885547d9f34419ad9066e096be8
Tulcea	138e8f3abac7ff233eaf02f798a5f9cc3b15284c
Vilanova	0a879320eb07e2dc1b270e78bc3b02b77ff1f2cf