D4.1 Guidelines for the structure of the overall development framework

Prepared by Gabriel Ruiz, Lesly Houndolé, Vincent Roch (CREM) and Antoine Widmer, Daniel Hunacek (HES-SO Valais)

Reviewed by Sara Fritz, Lukas Kranzl (TU Wien)

2017-06-29

Funded by the Horizon 2020 programme of the European Union

Project Information

Project name	**Hotmaps** – Heating and Cooling Open Source Tool for Mapping and Planning of Energy Systems
Grant agreement number	723677
Project duration	2016-2020
Project coordinator	Lukas Kranzl Technische Universität Wien (TU Wien), Institute of Energy Systems and Electrical Drives, Energy Economics Group (EEG) Gusshausstrasse 25-29/370-3 A-1040 Wien / Vienna, Austria Phone: +43 1 58801 370351 E-Mail: kranzl@eeg.tuwien.ac.at **info@hotmaps-project.eu** www.eeg.tuwien.ac.at www.hotmaps-project.eu
Lead author of this report	Gabriel Ruiz Centre de Recherches Energétiques et Municipales (CREM) +41 27 721 25 40 Gabriel.ruiz@crem.ch

Legal notice

The Hotmaps project

The EU-funded project Hotmaps aims at designing a toolbox to support public authorities, energy agencies and urban planners in strategic heating and cooling planning on local, regional and national levels, and in line with EU policies.

In addition to guidelines and handbooks on how to carry out strategic heating and cooling (H&C) planning, Hotmaps will provide the first H&C planning software that is

- **User-driven**: developed in close collaboration with 7 European pilot areas.
- **Open source**: the developed tool and all related modules will run without requiring any other commercial tool or software. Use of and access to Source Code is subject to Open Source License.
- **EU-28 compatible**: the tool will be applicable for cities in all 28 EU Member States.

The consortium behind

Scientific partners

Pilot areas for developing and testing the tool

1 Introduction

Amongst the 7 work packages (WP's) of the Hotmaps project, WP 4 is dedicated to the implementation of the open source tool with the objective to provide the user-friendly open source toolbox Hotmaps. This requires the implementation of a system environment for the open source tool from single energy system analysis modules and to define and implement a user interface and GIS-based visualizations for the user interaction. Thus, based on the user and the planning task, single or multiple modules have to be (sequentially) triggered. Additional required data has to be loaded from the data store and the results need to be visualized in an interactive, GIS-based map. Furthermore the testing of toolbox is initiated via the definition of beta versions for testing and step 1 of the testing procedure, namely automated testing.

The present report called "Guidelines for the structure of the overall development framework" represents the first deliverable of the WP4 due by the end of project's Month 9. This report provides a screening and definition of the IT related framework set-up and choices that could be implemented after this first project reporting period. It covers topics ranging from the management system in place and the retained development approach, up to the proposed hosting solutions and tool licencing, going through the selected technologies and tool architecture, as well as the chosen infrastructure and development frameworks. A point is also made on the remaining elements to be still set-up.

2 WP4 management

2.1 SCRUM-meetings

The retained agile development approach follows the SCRUM method (see chapter 3). Its application asks for a regular review of the development performed within a set timeframe (the so called "sprint"). The timeframe chosen for the Hotmaps project is of 3 weeks. Practically, a web conference is set up every third Thursday, from 10h30 to 12h.

As these meetings are happening on regular basis and involve the pilot areas (users' representative) as well as the institutes involved in the toolbox development, it was decided to extend the goal of the SCRUM meeting by starting with general project management topics. The followed agenda is set as follows:

- Organizational part for the whole consortium (all, 5 min): led by TUW and dedicated to discussing next steps, open questions and the progress of the project as a whole (like topics to follow-up or coming deliverables).
- SCRUM meeting – part I (all, 15 min): led by TUW and where the updates of the IT-development process are demonstrated for the Pilot areas. For the time being, it has been initiated that each Pilot area has a dedicated SCRUM meeting to present a specific energy concern of its territory. On a later stage, feedback from using the toolbox prototype will be collected from the Pilot areas.
- SCRUM meeting – part II (IT-Development team and all who are interested to participate, <60 min): led by HES-SO (SCRUM master) and dedicated to IT specific

progress and issues. Attendance of Pilot areas is welcome, although not required. During this part of the meeting, every institute involved in IT activities gives a feedback on its progress or encountered issues. CREM is in charge of the sprint review - a review of the achieved and coming programming activities. This sprint review is performed using Taiga software (the retained project management system for Agile development). The other IT activities are also discussed, like the features specifications, infrastructure set up, demonstration of the progress made on the user interface, etc.

In short, these SCRUM meetings are actually the regular follow-up meeting of the project. Furthermore, they give the pilot areas the opportunity to provide continuous feedback on the ongoing developments, on the features definition and on the application modules later in the project (part of the Agile approach).

2.2 CREM – HES bilateral meeting

In between the SCRUM meetings, the WP4 core team, i.e. CREM and HES-SO, meets on a regular basis to ensure a proper coordination of the software development activities:

- Physical meeting 1 week after the SCRUM meeting from 9h to 11h. The goal is to address management issues related to WP4, as well as to review and get aligned on each component of the IT project specific (infrastructure, architecture, data base structure, interface, testing etc.).
- Alignment meeting (by phone or web conference) the day before SCRUM meeting at 16h. The purpose it to coordinate the input to Part II of the SCRUM meeting.

2.3 CREM – HES daily SCRUM-meeting

As required by the Agile approach, the IT developers are holding a short alignment meeting 10 min at 10 am every day. Done by phone or web conference, the purpose is to inform all on the planned work of the day, and to ask for support in case a blocking issue is encountered.

This way of acting fosters the mutual support and facilitates the attribution of the works to be carried out. Indeed, supported by the Taiga project management system, each day developers can reallocate the programming tasks in coherence with each other's availability and competencies.

2.4 Working session with other partners

On top of the regularly scheduled meeting presented above, specific working sessions are hold on demand. These can happen to further discuss an issue raised by a partner during the SCRUM meeting or to get aligned with the project leader TUW.

In this respect, several working sessions (through web conference) are carried out with TUW regarding the toolbox features definition, the consistency of input/output and the comparison of indicators, the user storyboard (mock-up of the toolbox), and overall link between WP4 and other WP's.

3 Development approach

3.1 Agile methodology

3.1.1 Introduction

The IT developments are driven by the Agile approach which consists in “breaking product development work into small increments that minimize the amount of up-front planning and design. Iterations are short time frames (time boxes) that typically last from one to four weeks. Each iteration involves a cross-functional team working in all functions: planning, analysis, design, coding, unit testing, and acceptance testing. At the end of the iteration a working product is demonstrated to stakeholders. This minimizes overall risk and allows the product to adapt to changes quickly. An iteration might not add enough functionality to warrant a market release, but the goal is to have an available release (with minimal bugs) at the end of each iteration. Multiple iterations might be required to release a product or new features” (source: https://en.wikipedia.org/wiki/Agile_software_development, June 2017).

The key principles of the adopted Agile development method can be summarized as follows:

- Release often, resulting from incremental and iterative development works (ref. to the above mentioned "time boxes").
- Get real user feedback, meaning involving the users from an earlier stage and throughout the whole tool development.
- Focus on value rather than output, which allow to reset priorities according to the received users' feedback (and validated by the product owner (TUW)).
- Experiment a lot with product, so it reflects users wishes and requirements

3.1.2 SCRUM & sprint, taiga and development of features according to priorisation from owner

To integrate the Agile methodology within the project, we defined sprint of 3 weeks punctuated by a SCRUM meeting (see Chapter 2). The goal of the SCRUM meeting is to update each partner on what was done during the last sprint, to test freshly done features to verify they are in line with users’ requirements, to discuss current technical or organisational problems and at last make sure each partner knows where to focus for the next 3 weeks.

Tasks for next sprints are prioritized based on the product owner. Everyone can see at anytime what is currently on focus in Taiga, (https://taiga.io) a visual tool for management of SCRUM based projects.

3.2 Coding

3.2.1 Documentation

The Hotmaps code related to the toolbox and the webservices will be documented. Documentation allows for a better understanding of the code, and how the available features have been implemented. Code documentation is also key when seeking for an open source tool to be supported by a wide development community.

As coding activities also involves other WP's and project partners, and also thinking for the future development contributions, CREM will publish a Documentation guideline during the project to align any contributors to related best practices.

3.2.2 Git-flow

In the development process, "Git-flow" was implemented to manage different version of the source code. Git-flow is a workflow for development with the Git version control system. It manages how the Hotmaps repository, commits and branches are structured. Essentially, Git-flow is a set of procedures that every team member has to follow in order to come to a managed software development process. A dedicated tutorial was released by CREM and is available in Annex 0.

3.2.3 Testing

Unit testing was defined in this project as a "must-have". Unit testing represents a software development process in which features that are parts of an application, called units, are individually and independently tested for proper operation. Unit testing can be done manually but is often automated by using continuous integration (CI).

The great benefit to unit testing is that the earlier a problem is identified, the fewer compound errors occur. A compound error is one that doesn't seem to break anything at first, but eventually conflicts with something down the line and results in a problem. CI is therefore implemented in order to achieve automated testing.

There are two programming language used to develop the Hotmaps toolbox: python and javascript. For testing purposes, python developers can use **unittest -** a python framework for unit testing. Javacript developers can use **karma -** a javascript framework for unit testing. CI will be done by using the software Jenkins.

3.3 From user needs to features development

3.3.1 Features Templates

In order to describe all the features of the Hotmaps toolbox in a standard and common way, templates have been created.

These features come from the compiled end user needs generated in the deliverable "User stories and priorisation", an output from WP6. This report reflects the specification and needs collected from the Pilot areas and classified according to their development priorities.

Descriptions through these templates encompass, for each feature:

- Its inputs
- Its outputs
- The required data
- The required calculation modules

This ensures the technical feasibility of the features in terms of project partner competences and data availability.

Once completed, these features will serve as requirement specifications for the IT developers.

For convenient sharing purposes with Hotmaps partners, these templates have been created with the help of GoogleSheet documents and associated scripts.

A guidelines document has been created which describes all the templates, their usage and access to them (see Annexe 8.2).

3.3.2 User interface storyboard (next steps)

As described in chapter 3.3.1, two parallel works have been carried to define all the features that will be part of the Hotmaps toolbox:

1. Feature coming from the user needs (Deliverable of the task "Specification and priorisation of the user needs" of WP6)
2. Their standard description through templates to ensure their technical feasibility

At this stage, if the development of the individually prioritized features could start, an overall picture is however missing to ensure that each developed features will be properly integrated into the toolbox. Next step is then to describe all the relations between these features in order to ensure their compatibility and their coherent integration in the Hotmaps toolbox, while ensuring an efficient user experience.

This exercise is related to prototyping the Hotmpas User Interface (UI) by creating a wireframe (storyboard) that will be refined and completed iteratively over time. The Hotmaps UI Storyboard creation will be done through the help of dedicated software like the Balsamiq software, specifically dedicated to create mockups of the features.

These UI definition activities should also be managed following SCRUM and sprints, but where their validation should always happen one sprint ahead of the related feature programming activity.

4 Infrastructure and development framework

4.1 Servers

During the development phase of the project, the server is hosted at HES-SO Valais-Wallis on a virtual machine running Ubuntu 16.04 xenial x64. All the different services / servers needed by the project will be hosted as virtual containers on Docker. The advantage of doing so is to offer an easy to deploy application containing multiple servers in one place. The only constraint is to have Docker installed and an infrastructure capable of running the different services altogether. For now, the available system is set-up with 8 CPU and 8 GB of RAM with 100 GB disk space that can be increased if needed.

At the end of the project, the whole infrastructure will be migrated to another server that will be hosted by Energy Cities. The migration process will be facilitated as Docker can be run on any system and the configuration of the different services doesn't need any modification.

4.2 Docker

Docker is virtualization software that offers an additional layer of abstraction. This gives the ability to run multiple services or servers on different containers. Each container encapsulates the service or server and prevents unwanted access from each other. To make the containers communicate Docker allows to configure the network between each of them. This way the database can be for instance separated from the web server without using two different machines.

Docker version 17.04 for linux is currently in use.

Docker is currently running seven services:

- the web server of the toolbox "toolbox-client" developed by CREM & HES-SO
- the main web service "toolbox-backend" developed by HES-SO & CREM
- the database developed by HES-SO & CREM
- the computation module "building h-c" developed by TUW
- the computation module "building h-c default EU28" developed by e-think
- the computation module "waste heat application" developed by ISI
- the computation module "res-potential" developed by EURAC

4.3 Github/GitLab

4.3.1 Data

To host data files that are collected, GIT repositories are chosen as they offers a good amount of storage size and an easy setup. To avoid taking responsibility to maintain the infrastructure of Git, an online hosting on GitLab was chosen which offers 10 GB storage for free. This first set up allows starting to use Git and thus provides data for the whole consortium immediately. Then if the amount of storage size needed to be increased, GitLab can be installed on our server and thus have unlimited storage. There will be one repository for each partner. The structure of the different repositories will be defined at a later stage.

4.3.2 Code

GitHub was selected to host the code related to the project. GitHub offers free public repositories for open source projects, up to 1 GB per repository. There is one repository for each service, except the database:

- toolbox-client
- HotMaps-Toolbox (toolbox-backend)
- HotMaps-res-potential
- HotMaps-building_h-c
- HotMaps-building_h-c_default_EU28

- HotMaps-Waste-heat-application

Each repository contains one directory containing the code, one containing the docker startup scripts and one for the documentation.

GitHub is also used to host files used to build Docker images. However, the built images are stored on Docker Hub. The advantage of Docker Hub is that it allows to automatically building images on push to GitHub.

5 IT Architecture and technologies

5.1 Architecture scheme

The architecture of the toolbox is quite complex at first sight. There are multiple web services working as computation modules, a main web service, a database, some data repositories and a web application. To make this work altogether it was decided to use Docker (see Chapter 4) as abstraction layer so that the solution can easily be downloaded and run on any server.

In order to meet the needs of allowing the user to use his own data privately and to be compliant with the Data Management Plan (thus data confidentiality issues), two approaches are proposed in Hotmaps, the online & local toolbox, as well as the hybrid configuration of the toolbox, described in the following sections.

5.1.1 Online & local toolbox

The first one aims at offering the toolbox as both an online (see Figure 1) and offline software (see Figure 2). The online toolbox is the platform with public data, that will be hosted by Energy Cities at the end of the project and that anyone can use to plan with public data. The second one, the offline platform is made available to anyone who wants to plan with both private and public data.

In order to allow the user to import sensitive data, the solution gives the opportunity to add a private git repository in the data warehouse that will be made available in the toolbox. To use the offline platform, anyone can download the project from GitHub and build the application. The application can then be used locally and no private data is shared to the toolbox across the web.

This architecture schema is considered as must-have, as it represents an efficient implementation way for ensuring a compliant sensitivity data treatment while minimizing the risks related to complex developments. Furthermore, this architecture is the basis for the second possible architecture, meaning, the toolbox can still be upgraded to correspond to the other imagined architecture (if time allows).

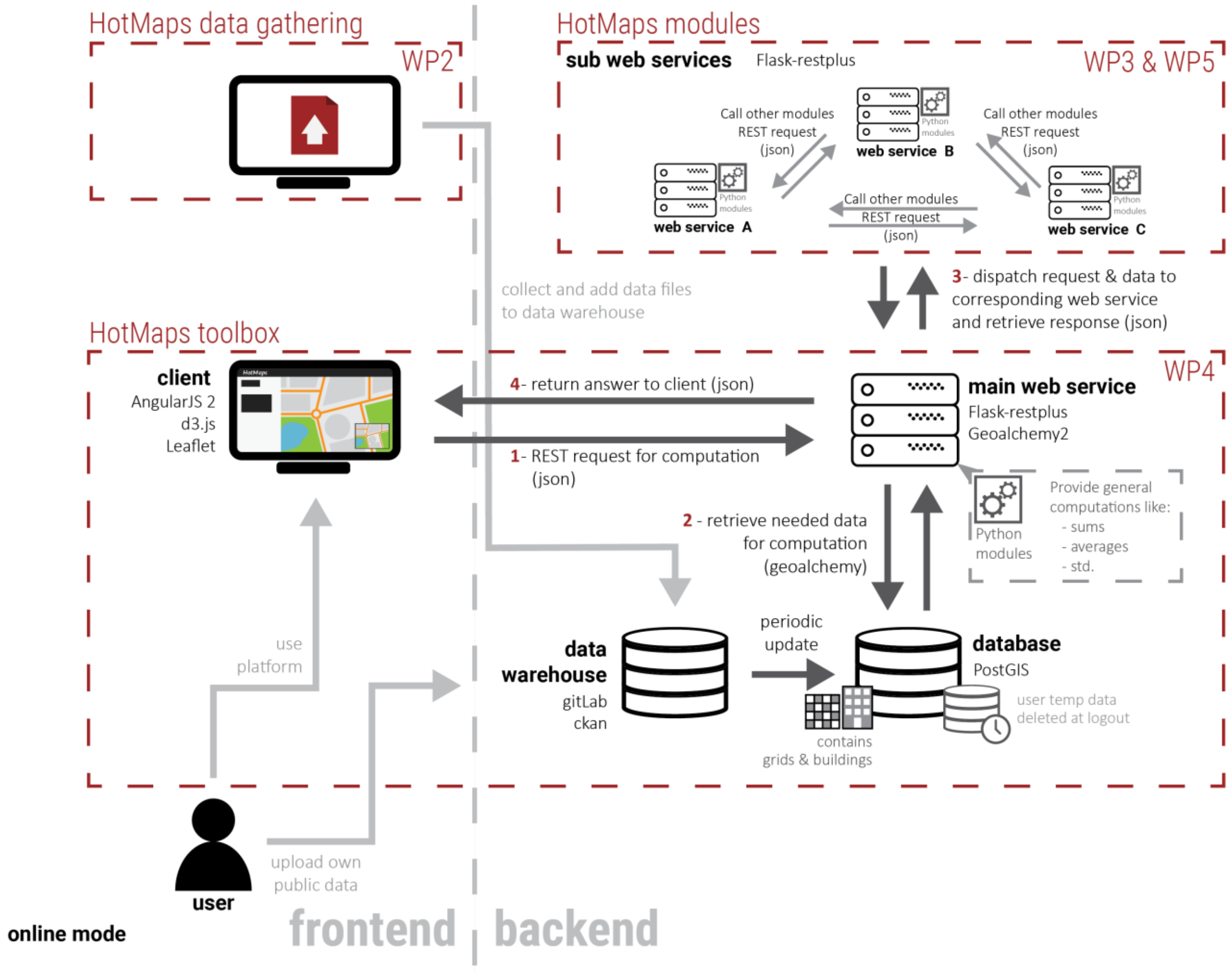

HotMaps data gathering
WP2
HotMaps modules
sub web services
Flask-restplus
WP3 & WP5
Call other modules REST request (json)
web service A
web service B
web service C
Python modules
Call other modules REST request (json)
Call other modules REST request (json)
collect and add data files to data warehouse
3- dispatch request & data to corresponding web service and retrieve response (json)
HotMaps toolbox
WP4
client
AngularJS 2
d3.js
Leaflet
4- return answer to client (json)
1- REST request for computation (json)
main web service
Flask-restplus
Geoalchemy2
Python modules
Provide general computations like:
- sums
- averages
- std.
2 - retrieve needed data for computation (geoalchemy)
use platform
data warehouse
gitLab
ckan
periodic update
database
PostGIS
user temp data deleted at logout
contains grids & buildings
user
upload own public data
online mode
frontend
backend

Figure 1: Online mode architecture of the Hotmaps tool

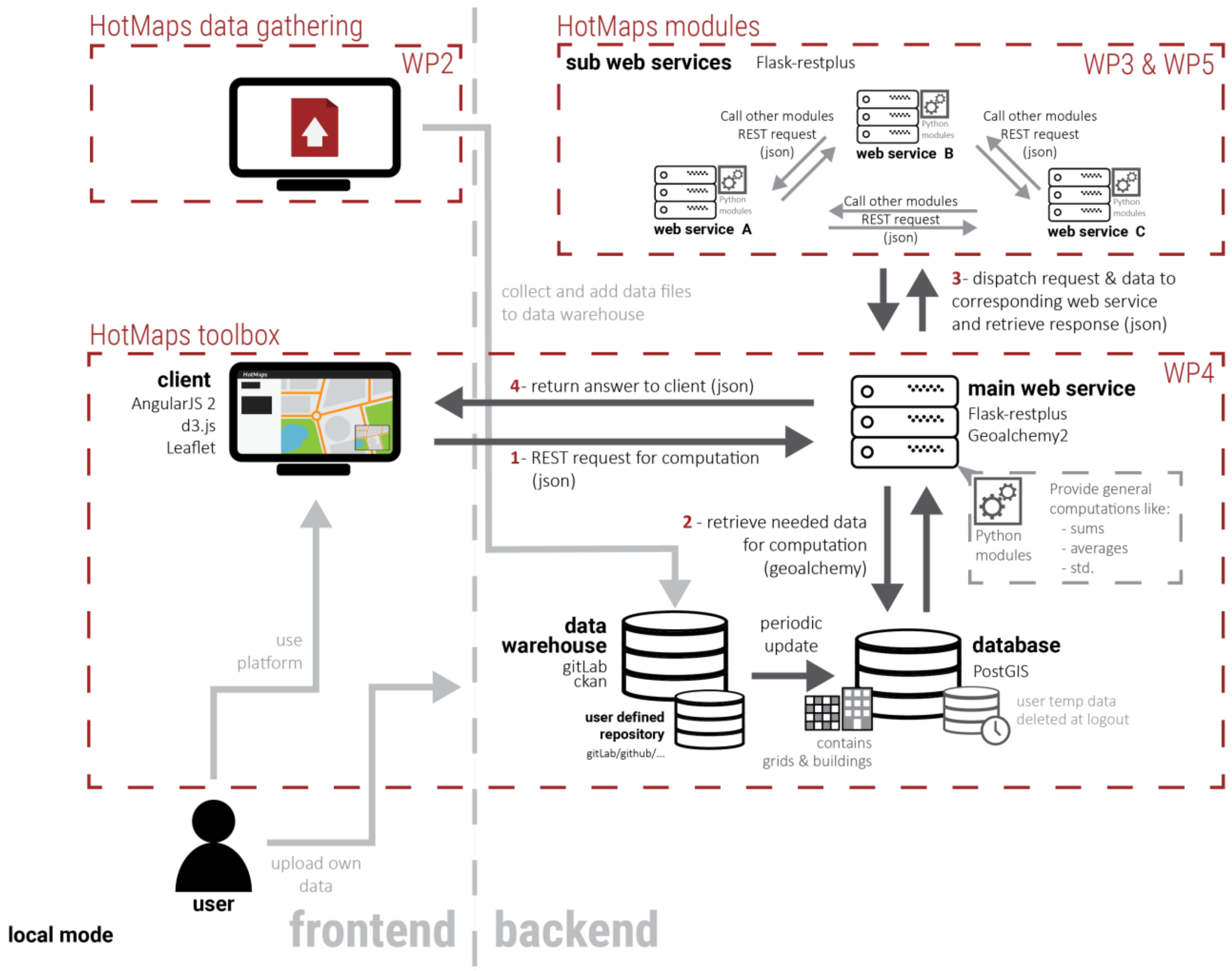
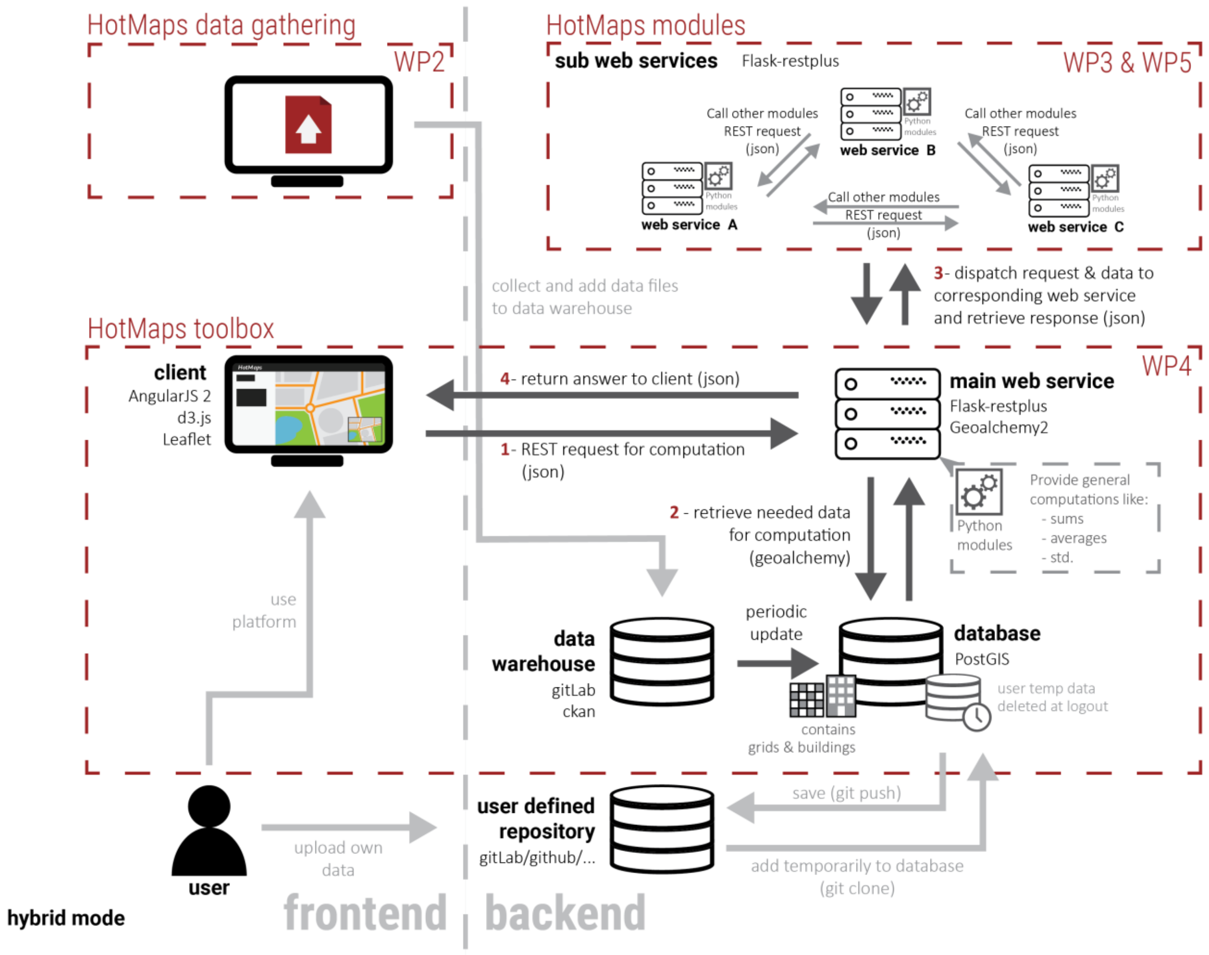


Figure 2: Local mode architecture of the Hotmaps tool

5.1.2 Hybrid configuration

This second architecture is more complex as it allows the user to upload his data directly onto the online toolbox. The user should then be able to decide which information he wants to share and which he wants to keep private. This architecture is considered as nice-to-have as it is more complicated to implement (see **Fehler! Verweisquelle konnte nicht gefunden werden.**). As shown in the figure, the user can define his/her own data repository that will be synchronized with the platform.



HotMaps data gathering
WP2
HotMaps modules
sub web services
Flask-restplus
WP3 & WP5
Call other modules REST request (json)
web service A
web service B
web service C
Python modules
Call other modules REST request (json)
Call other modules REST request (json)
collect and add data files to data warehouse
3- dispatch request & data to corresponding web service and retrieve response (json)
HotMaps toolbox
client
AngularJS 2
d3.js
Leaflet
4- return answer to client (json)
1- REST request for computation (json)
main web service
Flask-restplus
Geoalchemy2
WP4
Python modules
Provide general computations like:
- sums
- averages
- std.
2 - retrieve needed data for computation (geoalchemy)
use platform
data warehouse
gitLab
ckan
periodic update
database
PostGIS
user temp data deleted at logout
contains grids & buildings
save (git push)
user defined repository
gitLab/github/...
upload own data
add temporarily to database (git clone)
user
hybrid mode
frontend
backend

Figure 3: Hybrid mode architecture of the Hotmaps tool

5.2 Chosen technologies and why

5.2.1 Server side

Database

PostgreSQL was selected as database technology because it is the most advanced open source database on the market, and also because it supports spatial and geographic objects with the extension PostGIS. These two technologies are well documented and widely used, benefitting from a rich community around those topics.

API

The main web service and the calculation modules will follow RESTful API, mainly due to its proven performance and simplicity. For comparison purposes, RESTful API is indeed lighter than SOAP, less CPU expensive and really easier to setup.

The chosen data exchange format is JSON as it is known to be easy to read, light and native to JavaScript (particularly useful as it was decided to use JavaScript framework Angular 2 for the interface).

The different web services are developed in Python as this is a well-known coding language for scientific computations and because lots of existing libraries are available in that same field.

To build each of these services, it was decided to use the framework Flask and Flask-RESTplus. Flask is well known to be lightweight and easy to setup for REST API. Flask-RESTplus is an extension for Flask that offers even more minimal setup and that has the advantage of being compatible with Swagger (that will be used as API framework). The main web service also integrates the Python extension SQLAlchemy as Object Relational Mapper and GeoAlchemy 2 to make SQLAlchemy support PostGIS objects.

Calculation modules

The calculation modules mentioned above are the core of the toolbox. They offer computations of all kind regarding the energy data available on the platform. As each calculation module is responsible for specific tasks, they are hosted on separate web services to keep a modular use (see Chapter 4).

These services are made available across the backend of the toolbox. Thus, each module can benefit from the others by calling functions. Furthermore, each module is made available to the toolbox by the main web service that is acting as relay between the frontend and the backend.

Calculation modules are all built using Python language. They use the same technologies as the main web service and might integrate other technologies like GRASS depending on the needs.

5.2.2 Client side

AngularJS 2

AngularJS2 is used to develop the web interface. AngularJS (commonly referred to as ""Angular.js"" or ""AngularJS 2.X"") is a JavaScript-based open source front-end web application framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications.

The benefits of using AngularJS2 are:

- If an application is a heavy load, then Angular 2 keeps it fully UI responsive.
- It uses server side rendering for fast views on mobile.
- It works well with ECMAScript and other languages that compile to JavaScript.
- It uses dependency injection to maintain applications without writing too long code.
- AngularJS2 use the component based approach .

Leafleft

Leaflet is used for mapping action in the web interface. Leaflet is the leading open source JavaScript library for mobile-friendly interactive maps. It has all the mapping features most developers ever need.

Leaflet is designed with simplicity, performance. It works efficiently across all major desktop and mobile platforms, can be extended with lots of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.

The benefits of using Leaflet are:

- Light weight
- Free
- Strong community
- Evolutive via plugins

d3.js

D3.js is used to display the graphics of Hotmaps-toolbox. D3.js is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards.

The benefits of using d3.js are:

- Infinitely customizable
- Fully interactive

6 Hosting and data management

6.1 During project

6.1.1 Code

All the source code produced during the development of Hotmaps will be accessible on the open source platform GitHub. Each partner will place its code directly on GitHub as soon as the partner can. GitHub is the only way partners can exchange and integrate code for the final development of Hotmaps.

6.1.2 Default data (GitLab)

Default data used in Hotmaps are hosted on GitLab. Publicly available data are stored in a public repository allowing any user of Hotmaps to access data.

6.1.3 Pilot area data

Private data from pilot areas are hosted on GitLab. However, private repository is only used so only people having access to the repository can use data stored in such repository.

6.1.4 Test server

HES-SO provides access to a test server during the project duration. The server will host the Hotmaps toolboxes, backend and frontend so all partners can test directly online features made available by the IT development team. This test server will be disconnected at the end of the project and will be replaced by a production server that will be maintained by Energy Cities.

6.2 After project

6.2.1 Code

After the project is finished, the source code produced during the development of the project will not be moved. It will stay on GitHub as it is the main git repository hosting open source projects from all round the world. This will ensure that the outcome of the project can be used as long as possible after its end and the community can modify the source code so it stays up to date with more efficient hardware and algorithms.

6.2.2 Default data (GitLab)

Same as in subsection 6.1.2

6.2.3 Pilot area data

Same as in subsection 6.1.3

6.2.4 Production server

As soon as the project is finished, the latest version of Hotmaps will accessible from a production server maintained by Energy Cities.

6.3 INSPIRE and OGC compliance

CREM and HES-SO are committed to develop a data warehouse that is INSPIRE compliant and that will support the Catalog Services for the web (CSW), a specification from the Open Geospatial Consortium for exposing geospatial catalogues over the web. For these purposes, the tool CKAN is used with extensions that ensure compliance with INSPIRE and CSW.

CKAN is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers (national and regional governments, companies and organizations) wanting.

In order to make sure that the data warehouse is INSPIRE and CSW from the OGC, the following CKAN extensions will be used: "inspire_harvester" and "inspire_theme", both designed for working with INSPIRE metadata in CKAN and enabling:

- To display INSPIRE metadata in a user-friendly form at CKAN interface.
- To export INSPIRE metadata in extended GeoDCAT-AP 1.0 RDF format.

Furthermore, CKAN offers support for the Catalogue Service for the Web (CSW) standard. This support consists of:

- Ability to harvest records from remote CSW servers (as well as individual documents available online or Web Accessible Folders of them). CKAN supports the ISO-19139 encoding.
- Basic CSW interface (for the harvested records)

7 Licence

Chosing the right licence under which the toolbox will be available and distributed is of importance, especially given that many possibilities exist amongst the open source ones.

Important before any choice to keep in mind the following:

- A defined license applies all the time.
- But the obligation (if any) to distribute the modified source code is linked to the distribution of the software.

This said, the selection approach started by comparing the two main families of licences: the BSD and GPL types, where:

- BSD types (e.g. MIT, Apache) allow anyone to modify and further distribute the software without any obligation to disclose the source code (basically the improvements).

- GPL types force to distribute the modified source code as soon as the software is distributed ß any evolution of the code which is made available through a web application for instance is not considered as a distribution of the code, except for the AGPLv3.
- BSD types are not submitted to the condition of releasing modifications under the same license as the source code.
- This condition applies for GPL types.

In parallel, a discussion within the IT Team is required to identify the intentions with regard to a use and distribution of the Hotmaps' code. Three objectives were identified:

- Keep being the reference partners for any further developments of Hotmaps and avoid that another branch of the code becomes the “reference code”.
- That a community grows as much as possible around Hotmaps
- Leave the door open for commercial opportunities with Hotmaps

Concerning the first pursued objective, it appears that no license can ensures this, as long as the code is not distributed. And having control over time on the reference code is a matter of being a main contributor. It is however requested, if possible, to force any distributed code to make reference to its source (the Hotmaps project).

Concerning the second objective, it was agreed that GPL types are rather considered as a “contaminating license”, mostly by lack of knowledge, and may lead to refrain the community to contribute to Hotmaps. Whereas, BSD licence types, more popular, may foster the creation of an active community around Hotmaps.

Concerning the third objective, both GPL and BSD license types allow for commercial activities. MIT type leaves the additional possibility to ask for royalties to anyone who markets the application.

The following decisions were made based on the above arguments and more detailed considerations of available licences (see annex 8.3):

- To adopt one of the known licences (not to create a specific licence)
- To go for a BSD type licence, and with a preference for Apache.

If Apache is the retained licencing solution, there is only one remaining point to be confirmed which relates to the first objective: that the reference to the Hotmaps project is kept. This point will be further investigated.

Annexes

GitFlow Guidelines

Templates Guidelines

Taiga Guidelines

GIT-FLOW Guidelines

For IT developers

Elaborated by CREM :

- Lesly Houndole, lesly.houndole@crem.ch
- Vincent Roch, vincent.roch@crem.ch

With the support of HES-SO :

- Daniel Hunacek, daniel.hunacek@hevs.ch
- Antoine Widmer antoine.widmer@hevs.ch

Tuesday, 31 January 2017

Funded by the Horizon 2020 programme of the European Union

Date: 31/01/2017

Table of Contents

Introduction

This document has been written to outline and define the version control process we will be using for HotMaps 2020 project. It is aimed at anyone who would like to get involved by contributing to the source code. We will be using Git to version control the codebase.
Our aim is to keep a clean but informative history of commits, allowing us to revert mistakes easily.

We will use “Bump version” in order to increment the version number to a new, unique value. Bump is a software project's VERSION which adds the CHANGES, and tags with GIT. You can download bump at:
https://gist.github.com/pete-otaqui/4188238

Glossary of terms

This section describes the basic terms we use and assume knowledge of. Full documentation of Git terms can be found at:
https://git-scm.com/doc

Term	Description
Repository	A Git repository is essentially a collection of branches, all related to the same code base.
Commit	A commit in Git can be thought of as a snapshot of the repository at any given point.
Commit message	A commit message is the message that describes the changes made in a commit, viewable with 'git log'.
Branch	A branch in Git can be thought of as a sequence of commits which can be separated from all other commits. A branch may be merged or rebased onto another branch to insert its commit history to other branches.
Master	The master branch is the parent branch of all other branches, including develop branch. In our process, it will receive only two types of commits: Hotfixes and Major updates. The master branch will always contain stable code and will be what the public will receive.
Develop	The develop branch is a branch take from the master branch. It is where all feature branches will be derived from. The code contained in develop should always be stable. This branch will be the source of the latest codebase, including nightly builds for example. Develop will receive all Hotfixes as well as master.
Feature	A feature branch is a branch used to develop a new functionality. Very large feature additions may require multiple feature branches to be created.
Hotfix	A hotfix branch is used for critical bug fixes. This type of branch is taken directly from master and, once the work is completed, merged directly to master and develop
Release	A release branch support preparation of a new production release
Tag	A tag is the term used to define a textual label that can be associated with a specific revision.
Pull requests	Pull requests let you tell others about changes you have pushed to a repository on Git. Once a pull request is opened, you can discuss and review the potential changes with collaborators and add follow-up commits before the changes are merged into the repository.

Responsabilities

CREM, HES-SO and TUW will be responsible for reviewing the code. Everyone else will have to use pull request in order to modify master, develop, and release branch.

All source code for HotMaps Project must take place in the Git repositories owned by TUW.

Access rights

There are two kinds of access rights:

- **owner** access
- **collaborator** access

The repository **owner** will have full access to the repository:

- Invite collaborators (https://help.github.com/articles/inviting-collaborators-to-a-personal-repository)
- Change the visibility of the repository
- Merge a pull request on a protected branch, even if there are no approved reviews
- Delete the repository (https://help.github.com/articles/deleting-a-repository)

Collaborator access has limited access to the repository:

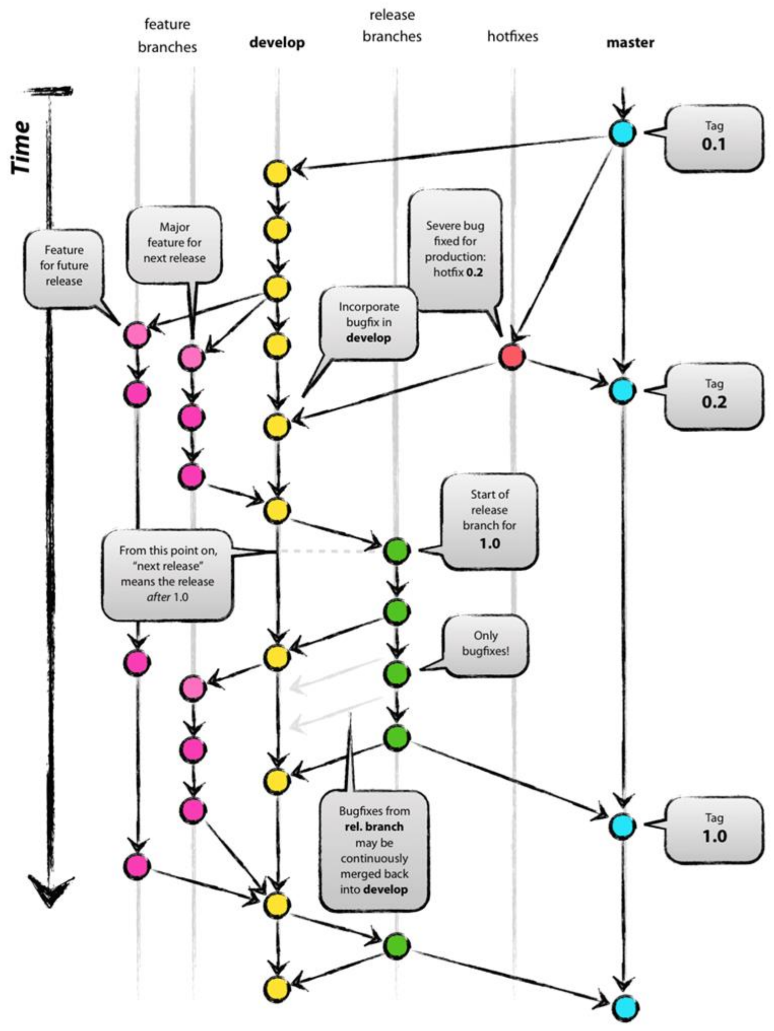
- Push to (write), pull from (read), and fork (copy) the repository
- Apply labels and milestones
- Open, close, re-open, and assign issues
- Edit and delete comments on commits, pull requests, and issues
- Merge and close pull requests
- Send pull requests from forks of the repository
- Create and edit Wikis
- Create and edit Releases
- Remove themselves as collaborators on the repository
- Submit a review on a pull request that will affect its “mergeability”

For HotMaps projects, the following collaborators will have the **owner** access:

- Daniel Hunacek, HES-SO, daniel.Hunacek@hevs.ch
- Mostafa Fallahnejad, TUW, fallahnejad@eeg.tuwien.ac.at
- Lesly Houndole, CREM, Lesly.houndole@crem.ch
- Sara Fritz TU Wien, fritz@eeg.tuwien.ac.at

Git-flow definition

Git-flow is a workflow for development with the Git version control system. It is how we structure our repository, our commits and branches. Git-flow is essentially no more than a set of procedures that every team member has to follow in order to come to a managed software development process. The diagram below (see URL below) shows an overview of this process:


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

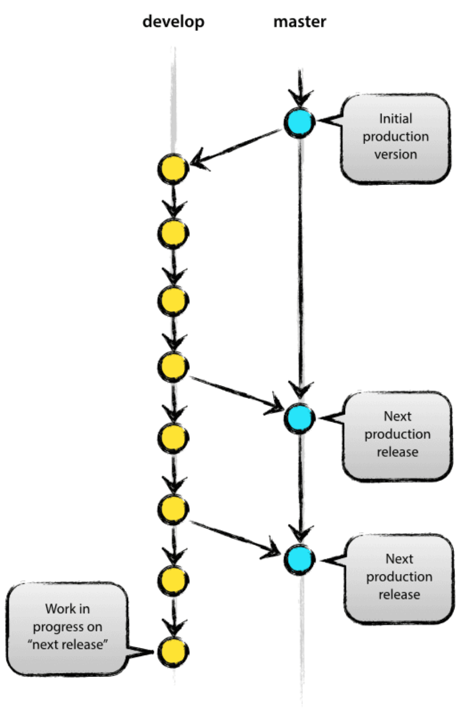
For the most part, this document is based on Vincent Driessen's article "A successful Git branching model" at http://nvie.com/posts/a-successful-git-branching-model/.

The develop branch will always aim to be stable, and bugs will have a branch created to fix them. We will enforce this by running tests for pre- and post- merges of the develop branch, and code reviews for all merge requests.

How do we use Git-flow?

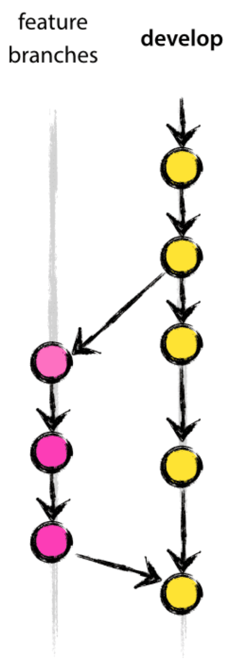
Branches

Master and Develop branches


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

There will be two long running branches, master and develop. These will be protected branches, meaning only a few persons will have the right to push changes directly to these branches. All additions to these branches must be done via a merge request (pull request). This allows code review of everything that enters the framework and ensure both master and develop branches stay stable.

Feature branches



When developing new functionalities or features, or simply refactoring/optimizing core code, you should create a feature branch. Feature branches are branched from develop and named after their ticket number or a brief description of the work being done.

Rules for feature branch:

- Must branch off from develop
- Must merge back into develop

Naming convention:

Feature branches should always be prefixed with “feature-”, for example “feature-12345” or “feature-something”.

Commands :

Switched to a new branch "myfeature":

```
$ git checkout -b myfeature develop
```

After adding new feature “myfeature” branch

Switch to branch *develop*:

```
$ git checkout develop
```

Merge “myfeature” branch to develop

```
$ git merge --no-ff myfeature
```

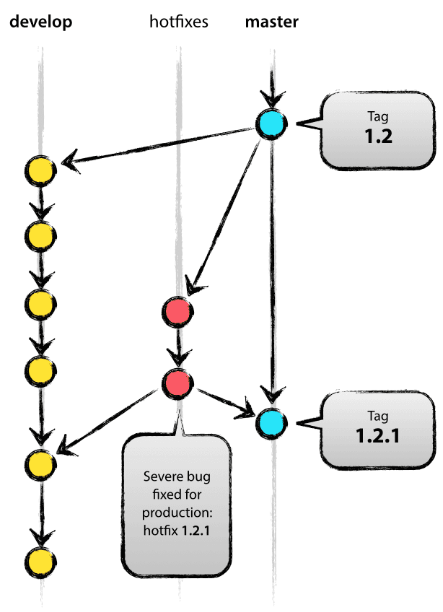
Delete branch “myfeature”

```
$ git branch -d myfeature
```

Push to develop branch

```
$ git push origin develop
```

Hotfix branches



When a serious bug is found or a fix is needed immediately, you should create a hotfix branch. Hotfix branches are spun directly from master, tested as soon as possible and merged into the master branch using a -squash merge. This gives us a commit checkpoint of the fix being implemented. Hotfix branches should also be merged directly into develop.

Rules for Hotfix branch:

- Must branch off from master
- Must merge back into develop and master

Naming convention:

Hotfix branches should always be prefixed with “hotfix-”, for example “hotfix-*”

Command:

Switched to a new branch "hotfix-1.2.1"

```
$ git checkout -b hotfix-1.2.1 master
```

Files modified successfully, version bumped to 1.2.1.

```
$ ./bumpversion.sh 1.2.1
```

[hotfix-1.2.1 41e61bb] Bumped version number to 1.2.1

```
$ git commit -a -m "Bumped version number to 1.2.1"
1 files changed, 1 insertions(+), 1 deletions(-)
```

It is really important to bump the version number after branching off!

Then you should commit the fix

```
$ git commit -m "Fixed severe production problem"
[hotfix-1.2.1 abbe5d6] Fixed severe production problem
5 files changed, 32 insertions(+), 17 deletions
```

At the end we merge back into master and develop.
Switched to branch master

```
$ git checkout master
```

Merge made by recursive. (Summary of changes).

```
$ git merge --no-ff hotfix-1.2.1
```

You should not forget to tag this version of code.

```
$ git tag -a 1.2.1
```

After the code is pushed on master you should push it again on develop.
Switched to branch develop.

```
$ git checkout develop
```

Merge made by recursive. (Summary of changes)

```
$ git merge --no-ff hotfix-1.2.1
```

There is an exception to the rules we write above. If there is a release branch, you must merge on release branch instead of develop.

Rules for Hotfix branch when release branch currently exists:

- Must branch off from master
- Must merge back into release and master and develop

Delete branch hotfix-1.2.1 (was abbe5d6).

```
$ git branch -d hotfix-1.2.1
```

Release branches

Release branch is the branch that will be used for the new production release. Release branches are created from develop branch

Rules for release branches:

- Must branch off from develop
- Must merge back into develop and master

Naming convention:

Release branches should always be prefixed with “release-*/”, for example “release-1.2”

Command:

Switch to a new branch "release-1.2"

```
$ git checkout -b release-1.2 develop
```

Files modified successfully, version bumped to 1.2.

```
$ ./bump-version.sh 1.2
```

After you should make a commit

```
$ git commit -a -m "Bumped version number to 1.2"
```

When the release branch is online, you should switch to release and merge it to master, then tag the master branch. Afterwards you should merge back to develop as follow.

Switched to branch master

```
$ git checkout master
```

Merge made by recursive. (Summary of changes)

```
$ git merge --no-ff release-1.2
```

Tag this version of code

```
$ git tag -a 1.2
```

Switched to branch develop

```
$ git checkout develop
```

Merge made by recursive. (Summary of changes)

```
$ git merge --no-ff release-1.2
```

References

- https://gist.github.com/pete-otaqui/4188238
- http://nvie.com/posts/a-successful-git-branching-model/
- https://git-scm.com/doc
- https://git-scm.com/book/en/v2/Git-Tools-Submodules

www.hotmaps-project.eu

Funded by the Horizon 2020 programme of the European Union

Templates guidelines for user stories description

Prepared by Vincent Roch, CREM

Reviewed by Sara Fritz, TUW

2017-05-02

Funded by the Horizon 2020 programme of the European Union

Table of contents

1 Introduction

In order to describe user stories, templates have been created. They serve the following main purposes:

1. Align every partner on:
    a. what the end user wants to be able to do, get and see in the Hotmaps toolbox
    b. the effective capacity to satisfy these needs in terms of:
        i. Data availability
        ii. Calculation Modules
        iii. IT development
    c. a standardized way to describe it
    d. the persons who are responsible for coherence

2. Generate, at the end, requirement specifications for IT developers

Templates have been created through the usage of online shared Google Sheets files and Scripts (more information about files organisation are available in 3.2).

2 Description of the templates

Four types of templates have been created:

1. **US** template (**User Stories**):
   Dedicated to the end users of the Hotmaps tool to describe their needs.

2. **CM** template (**Calculation Module**)
   Dedicated to energy engineers/specialist to describe the inputs/outputs of their Calculation Modules.

3. **AD** template (**Available Data**)
   Dedicated to energy engineers/specialist to describe Available Data, their source and their access.

4. **F** template (**Feature**)
   Dedicated to energy engineers/specialist to describe and coordinate how the end user needs (Hotmaps toolbox functionalities) will be implemented from the Calculation Modules and the Available Data.

***Remarks*:** General scheme of the templates organisation is available on page 11.

2.1 US template

The US template is dedicated to end users to describe what are their needs in terms of tool's functionalities:

- There is one file including all the US templates (tab)
- There is one template (tab) per user story
- Each user story is divided into Features which are unitary functionalities
- For each Feature constituting the user story, the following information are requested:
  - Unique user story ID (US.*UserStoryID.UniqueNumber*)
  - Priority according to the pilot areas
  - Corresponding Feature ID (defined in F template)
- From the Feature ID, the description (available in F template, see 2.4.1) of the corresponding Feature is automatically completed (autocompletion will be referred as "auto" in the rest of this document). It encompasses:
  - Name of the Feature
  - User action: how the end user wants to interact with the tool
  - Indicators/values: what he/she wants to get in terms of indicators/values
  - Visualization: in what form he/she wants to see them
  - Mockups
- A summary table is available under the "US_overview" tab which lists all the user stories (US tabs)

2.2 CM template

CM templates are dedicated to energy engineers/specialist to describe the inputs/outputs (I/O) of their Calculation Modules used in Features (see 2.4.4):

- There is one file per partner
- There is one template (tab) per Calculation Module coming from this partner with the following information:
    - Calculation Module name
    - Calculation Module ID (CM.*PartnerName.UniqueCMnumber*)
    - Date of last modification (auto)
    - Responsible person, Institute, e-mail address
    - Programming languages and libraries
    - Calculation time estimation
- Each I/O is described with the following information:
    - Unique I/O ID (In/Out.CM.*PartnerName.UniqueCMnumber.UniqueNumber*)
    - Last modification date (auto)
    - Data name
    - Data attributes
    - Attributes description
    - Format

2.3 AD template

The AD template is dedicated to energy engineers/specialists to describe the Available Data they have at their disposal as well as their source and access. This information is then used in Features (see 2.4.4):

1. There is one file per partner with only one template (tab) with the following information:
    - Available Data ID (AD.*PartnerName*)
    - Date of last modification (auto)
    - Responsible person, Institute, e-mail address

2. For each Available Data, the following information is requested:
    - Unique ID (AD.*PartnerName.UniqueNumber*)
    - Last modification date (auto)
    - Data name
    - Data attributes
    - Attributes description
    - Data size
    - Source of data
    - Data source for Hotmaps tool (dropdown list):
        - External DB
        - Default EU28 (Data Warehouse)
        - Internal DB (Not published)
        - User Input ONLY

        - Person in charge
        - Data availability (dropdown list):
            - yes
            - no
            - not yet

2.4 F template

F template is dedicated to energy engineers/specialist to describe and coordinate how the end user needs (Hotmaps tool's functionalities) will be implemented from CM and AD. There is one file including all the F templates (tab) described here below:

2.4.1 *FeaturesList* tab

The *FeaturesList* tab is dedicated to the "FeaturesList guardian" which is the responsible person of the consortium of all the Features:

- This tab contains the list of all the Features used in US template
- For each Feature, all the following information is requested:
    - Feature ID (F.*UniqueNumber*)
    - Date of last modification (auto)
    - Name of the Feature
    - User action: how the end user wants to interact with the tool
    - Indicators/values: what he/she wants to get in terms of indicators/values
    - Visualization: in what form he/she wants to see them
    - Responsible person
    - Required other needed Features
    - Mockup
    - Development priority
    - State (dropdown list):
        - ready for description
        - description in progress
        - waiting for approval
        - ready for development
        - development in progress
        - ready for testing
        - test in progress
        - Complete

Remarque:

- Only the "FeaturesList guardian" is allowed to modify this tab. Any new Feature proposition by other partners must be submitted in the *NewFeatureProposition* tab (see 2.4.2)
- A mockup refers to corresponding pictures available in "Mockups" folder (see 3.2.1)

2.4.2 *NewFeatureProposition* tab

The *NewFeatureProposition* tab is used by any partner to suggest to the "FeaturesList guardian" a new Feature that does not exist yet. Requested information is the following:

- Name of the person proposing a new Feature
- Date of last modification (auto)
- Name of the Feature
- User action: how the end user wants to interact with the tool
- Indicators/values: what he/she wants to get in terms of indicators/values
- Visualization: in what form he/she wants to see them
- Required other needed Features
- Mockup

Remarque:

Only the "FeaturesList guardian" is allowed to accept a new Feature coming from *NewFeatureProposition* tab. Once accepted, the new Feature will be automatically integrated in the *FeatureList* tab.

2.4.3 *AD-request* tab

The *AD-request* tab is used by any partner to request new data from another partner. If the recipient accepts to provide this data, the data descriptions should be manually included in the corresponding AD template file. Requested information is the following:

- Name of the requester
- Last modification date (auto)
- Data name
- Data attributes
- Attributes description
- Data source for Hotmaps tool (dropdown list):
  - External DB,
  - Default EU28 (Data Warehouse)
  - Internal DB (Not published)
  - User Input ONLY

- E-mail of the recipient
- Status (dropdown list):
  - Waiting for response
  - Accepted
  - Rejected

Remarque:

The requester can automatically send an e-mail to the recipient using a dedicated "press button" included in the tab.

2.4.4 Features description tabs

Feature description template (tab) is used to describe how the Calculation Modules and the Available Data are connected in order to implement the specific Feature described in *FeaturesList* tab.

5 fields has to be completed for each Feature description template (tab):

- **General information:**
    - Feature name (auto)
    - Feature ID (F.*UniqueNumber*)
    - Geographical level of Feature (dropdown list):
        - Local
        - Regional
        - National
        - Does not apply
    - Date of last modification (auto)
    - Responsible person, Institute (auto), e-mail address
    - Check for modification in FeatureList (auto):
        - ok
        - be careful!
    - Feature consistency check (dropdown list):
        - ?
        - ok
        - not ok
    - Approved by
    - State (auto, see 2.4.1)
- **All inputs** needed for this Feature:
    - Only corresponding ID coming from AD files or CM files (Output) has to be filled
    - Corresponding information will be automatically imported
- **All Calculation Modules** needed for this Feature:
    - Only corresponding ID coming from CM files has to be filled
    - Corresponding information will be automatically imported
- **All outputs needed** for this Feature:
    - Only corresponding ID coming from AD files or CM files (Output) has to be filled
    - Corresponding information will be automatically imported
- **A flow chart** connecting all the Feature inputs, outputs and CM (referring to their respective ID):
    - To modify the flowchart, just double click on it

- To see the modifications, the page has to be reloaded

Remarque:

- In the Feature inputs, an "end user input possibility" box has to be filled depending on whether the end user can interact with the Feature. Three options are possible:
    - "Required":
      This means that the Feature input can only come from the end user. This input has to be described in an AD file with "Data source for Hotmaps tool" box set to "User Input ONLY"
    - "Not possible":
      This means that the input can only come from:
        - Output of a CM
        - AD file (excluding AD with "User Input ONLY" tag)
    - "Optional":
      This means that the input can come either from the end user or default data that are available in AD files (excluding AD with "User Input ONLY" tag)
- Feature input cannot come from a CM input (In.xxx) since this only specifies the input format of a CM and does not say anything about the data availability of this input. Therefore, Feature inputs can only come from AD files or CM outputs.
- In Inputs/Outputs/CM field, a "check for modification" in corresponding AD or CM files automatically attests whether a conflict could arise following modifications in these source files. If the date of the last modification in the AD or CM files descriptions comes after the date of their usage in the Feature description, then a "be careful!" arises as the coherence in the Feature description might not be valid anymore.
- The "Check for modification in FeatureList" box is also used to check if source data in "FeaturesList" tab have been modified after last updates of corresponding "F.x" tab. If it is the case, a warning appears and the "F.x" responsible person will have to check for coherence and update the date (by updating one of the row)
- The button "Update now" has to be pressed in order to get the most updated state of the "check for modification" boxes.
- In the outputs field, a "Consistency check from Inputs to Output" box is available and has to be filled (ok, no ok) in order to attest whether each Feature outputs can effectively be obtained regarding the flowchart connections. This check box should not be completed by the person who completed the Feature description.
- The same check box is available for the entire Feature description once all outputs have been approved. Only the "FeaturesList guardian" is allowed to approve a Feature.

- To add a new row in Feature inputs/CM/outputs fields, the row/cell where a row wants to be added has to be selected and the custom function “AddRow” in menu bar has to be used, then “Add new row below selected cell”. Doing so, all the formulas will be automatically pasted

2.5 General templates organisation

General organisation of the 4 templates are depicted in Figure 1. For a better understanding of this scheme, templates description presented in chapter 2 can be useful for parallel reading.

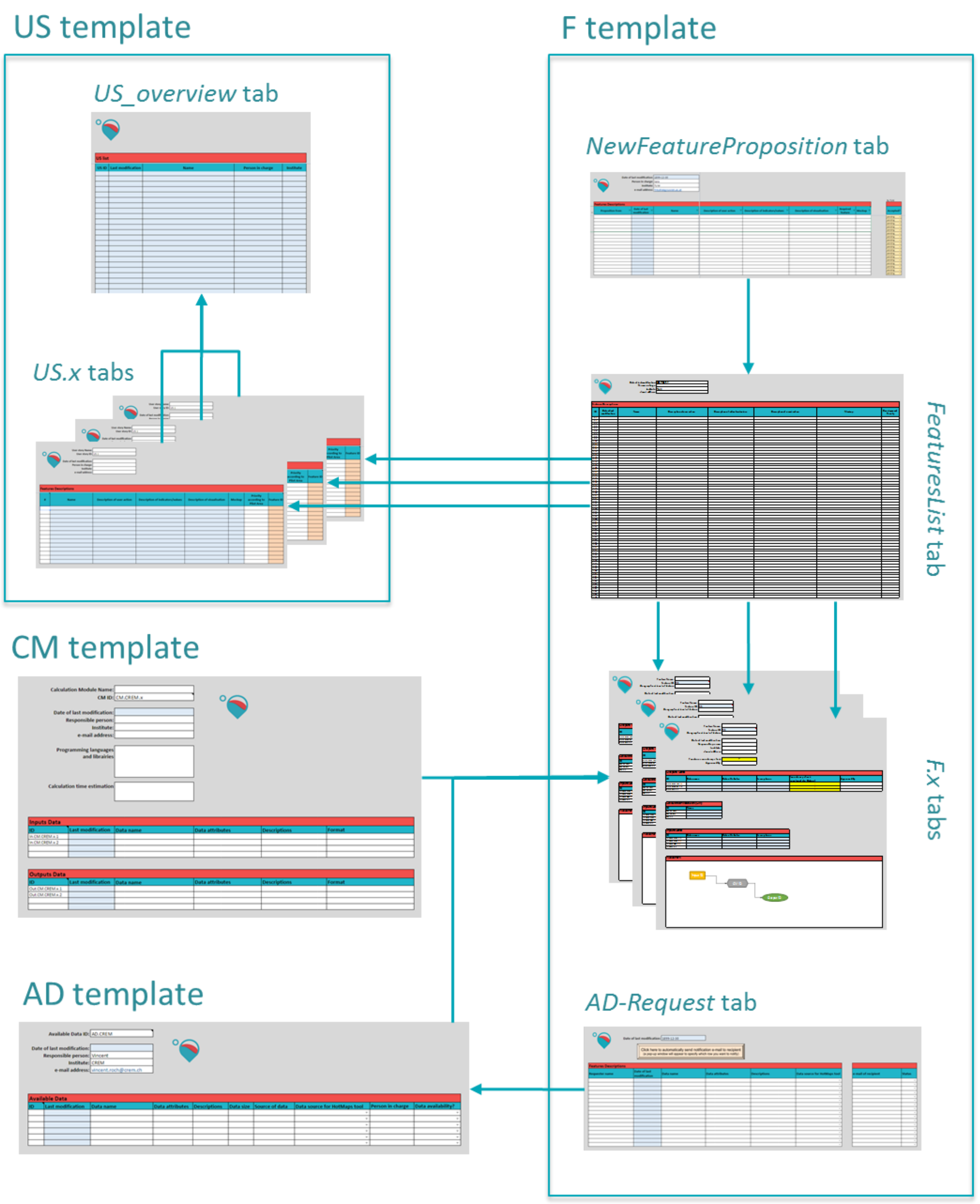


Figure 1: Templates organisation scheme

3 Additional information

3.1 Working with the files/templates

- Cells in light blue colour are updated automatically and must not be manually edited
- Sometimes, automatic updates can take several seconds to appear
- IDs and files nomenclature is very important as scripts are using them to connect all files/tabs together. If things are not working properly, it is probably because of wrong ID/files name format
- Any new tab must be created only by duplicating existing ones (NO range/cells copy-paste) → "right click" on tab and "duplicate"
- Additional comments are available in certain cells (especially headers)
- When someone wants to work on the files, he/she must log in to his Google Account so we can see who is working on what and also use chat functionalities offered by Google Sheets when working in parallel

3.2 Files organisation

3.2.1 Google Drive

A folder is shared with every Hotmaps partner through Google Drive. The structure of this folder is depicted in Figure 2.

Remarks:

- If a .gsheet file has to be added, especially in "CalculationModules" and "AvailableData" folder, a special request has to be send to CREM since some other files/Google Script will have to be modified in consequence
- List_URLs.gsheet contains the URLs of all the other .gsheet files that are need for Google Scripts. No modification of this file is allowed without the consent of CREM.

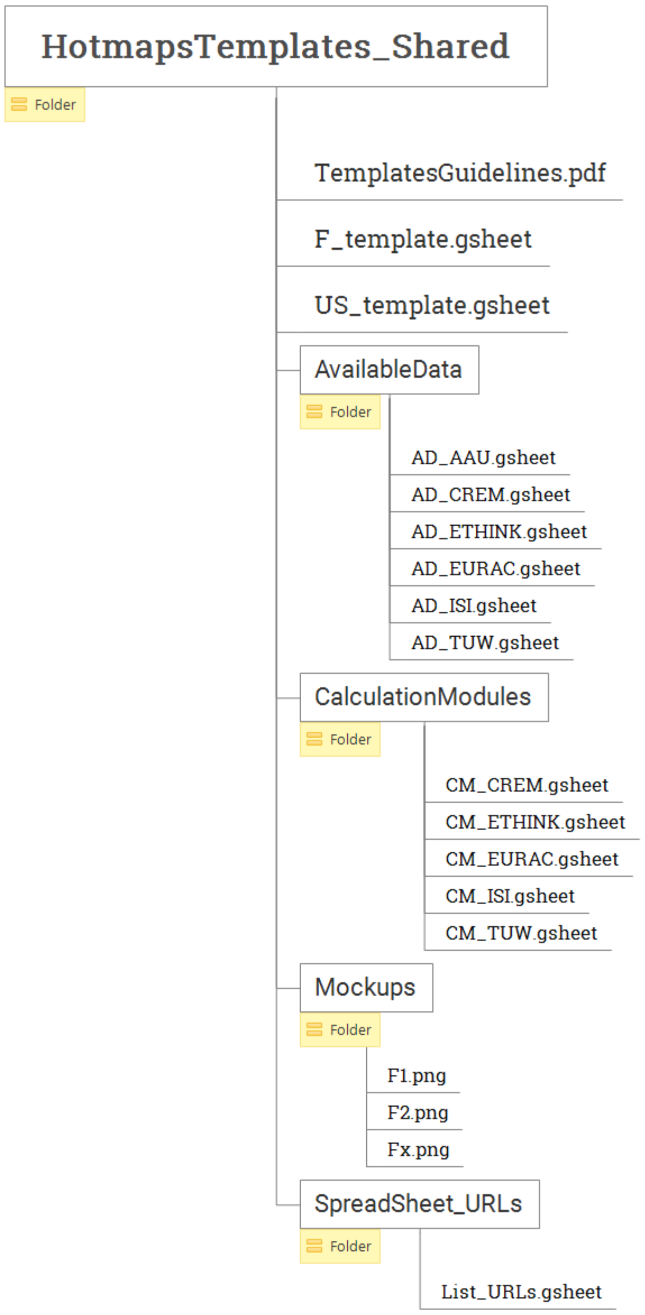


Figure 2: Shared folder hierarchy

3.2.2 Google Scripts

Google Scripts are used in the different .gsheet files for specific functionalities. No modification of these script files is allowed without the consent of CREM.

www.hotmaps-project.eu

Funded by the Horizon 2020 programme
of the European Union

Taiga

Guidelines & Tutorial

Elaborated by CREM :

- Lesly Houndole, lesly.houndole@crem.ch
- Vincent Roch, vincent.roch@crem.ch

Wednesday, 01 February 2017

Funded by the Horizon 2020 programme of the European Union

Date: 01/02/2017

Table of Contents

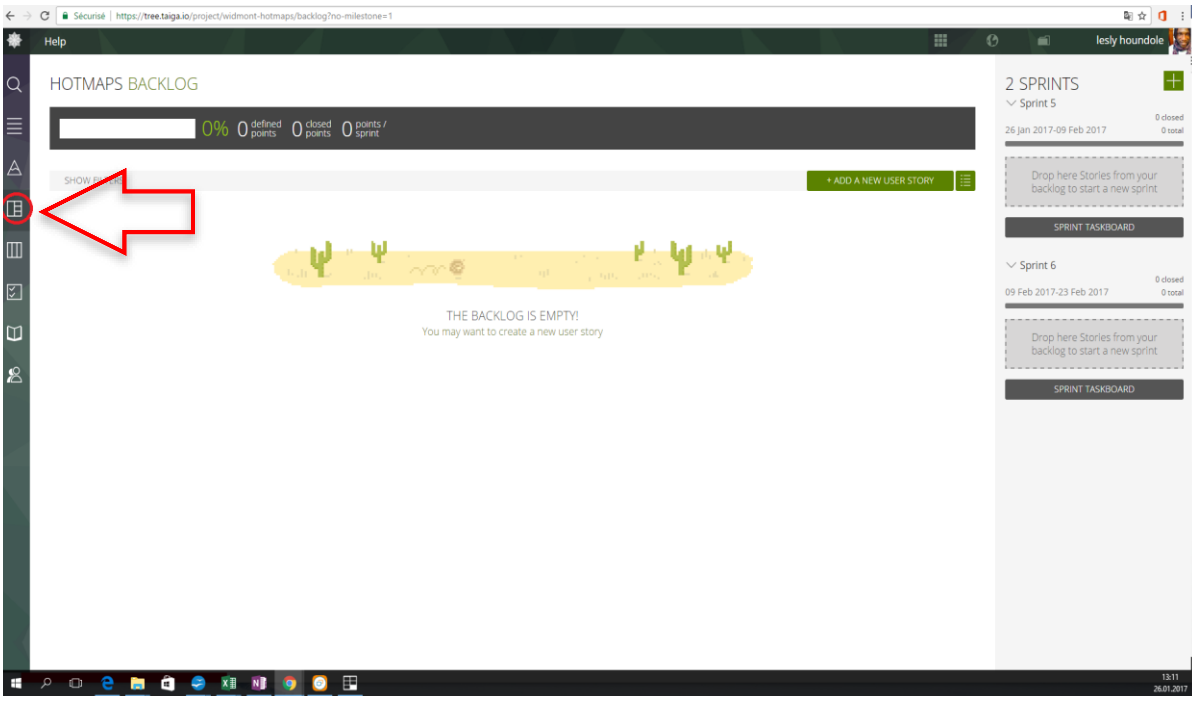
Taiga guidelines

- Each **Sprint** is created in Taiga 3 days before each SCRUM meeting at 11am
- Then, each developer will have 1 day to prepare Taiga (2 days before the SCRUM Meeting, 11am,) by filling in his/her **Tasks** for the newly created Sprint
- During the SCRUM meeting, we will then discuss about the prioritization of the tasks for the next Sprint

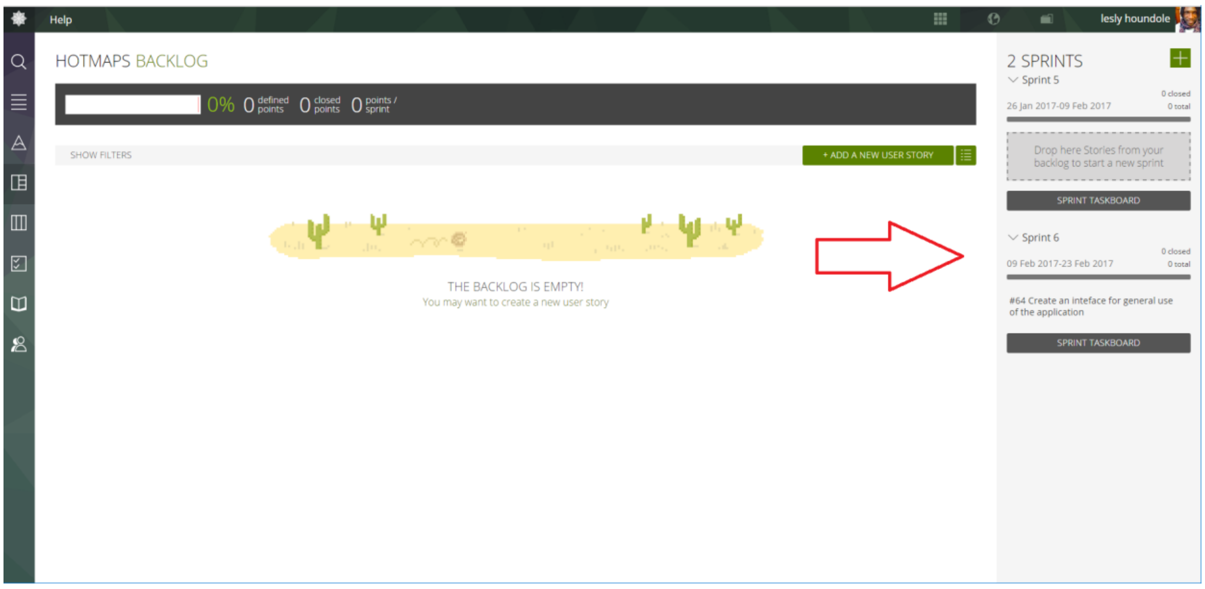
Taiga Tutorial

Task creation

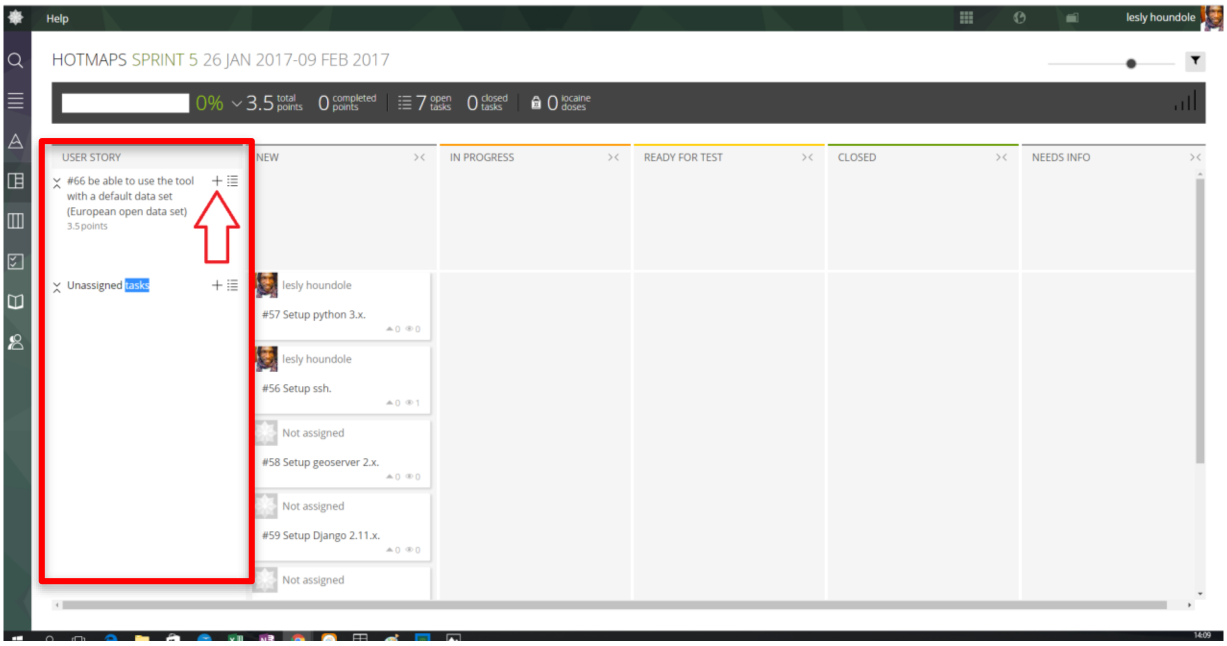
1. Click the HotMaps backlog button



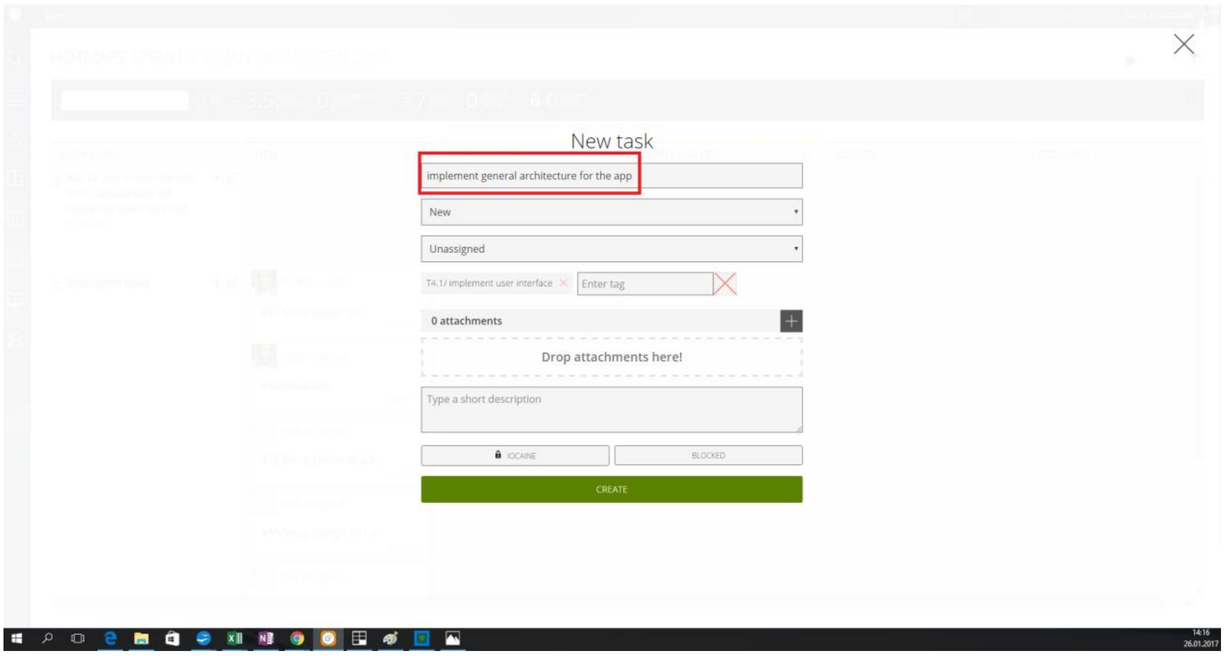
2. In this view, you can find sprints in progress and select the sprint for your tasks



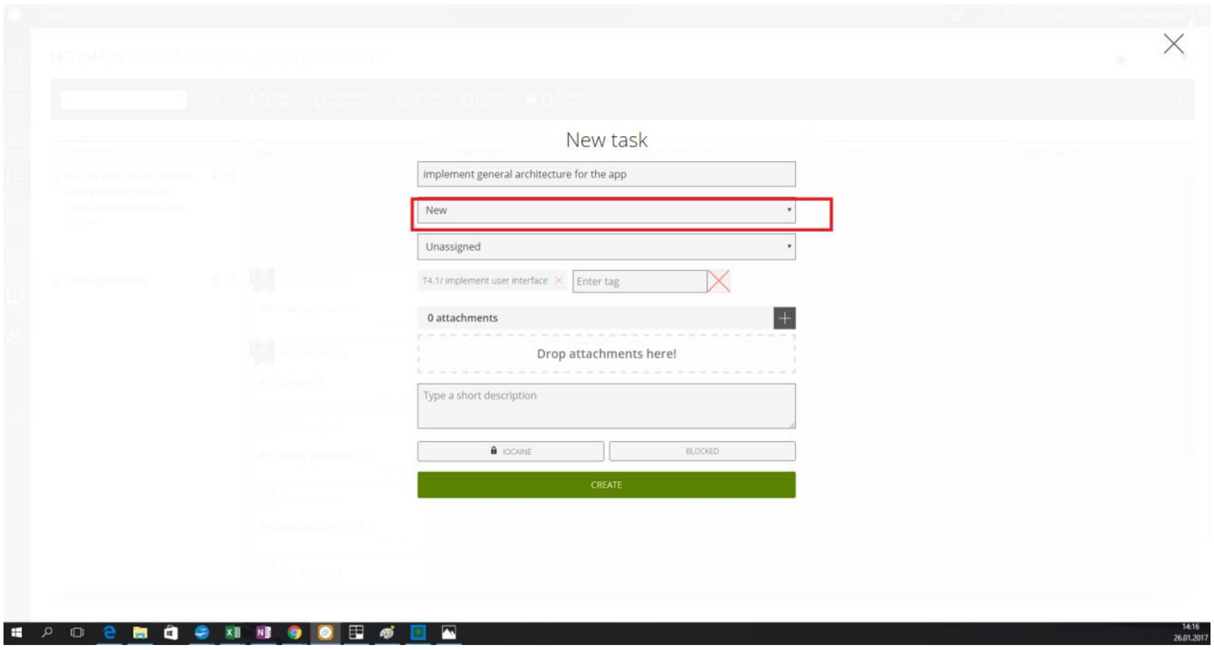
3. A task can pertain to a user story or not. Click the “+” sign to create a new task under the corresponding user story.
   - If the task does not pertain to a specific user story, create your task under “Unassigned tasks”
   - If the user story does not exist yet, see next chapter “user story creation”



4. add the task name



5. Add the status


New task

implement general architecture for the app

New

Unassigned

T4.1/ implement user interface

Enter tag

0 attachments

Drop attachments here!

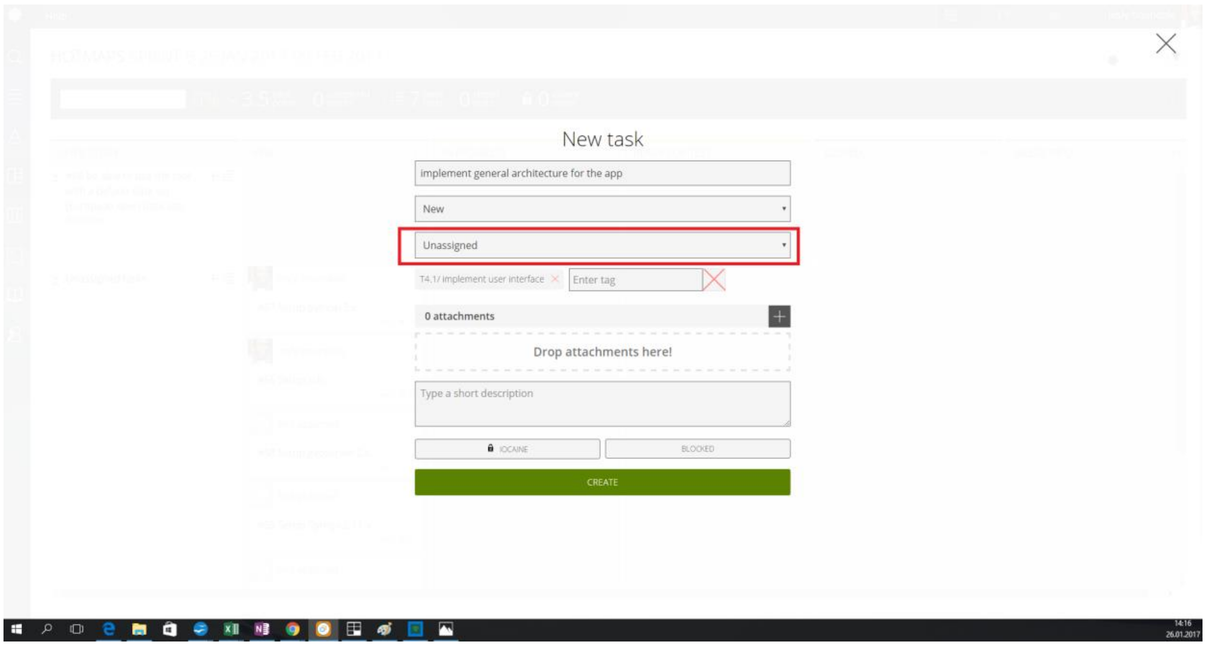
Type a short description

IOCAINE

BLOCKED

CREATE

6. Assign it to someone


New task

implement general architecture for the app

New

Unassigned

T4.1/ implement user interface

Enter tag

0 attachments

Drop attachments here!

Type a short description

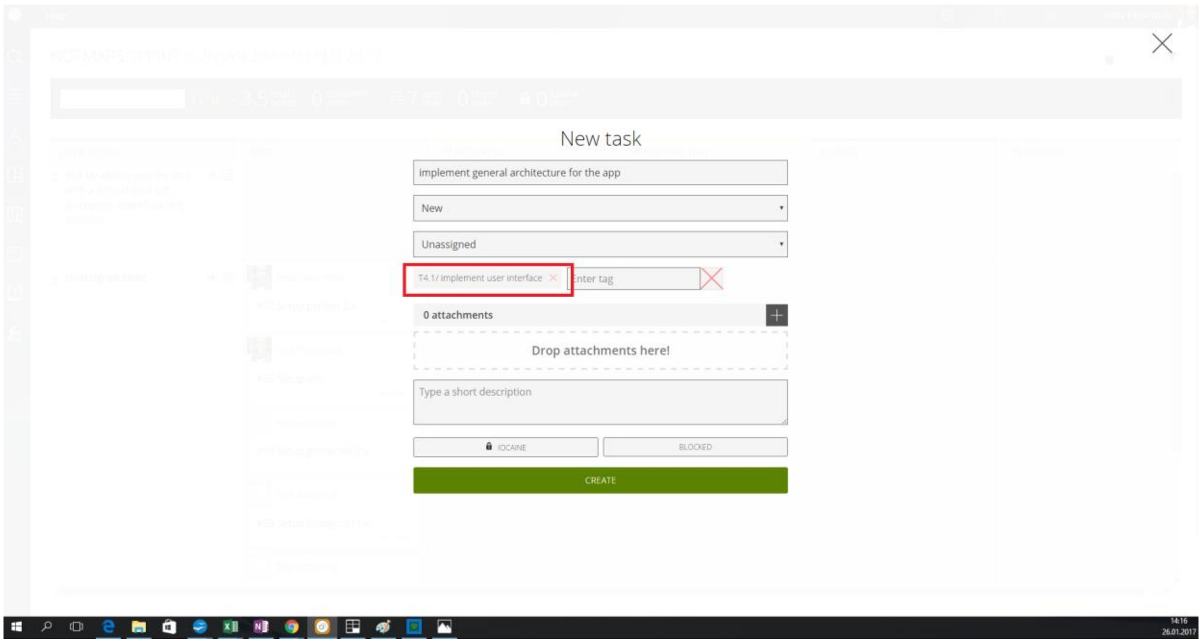
IOCAINE

BLOCKED

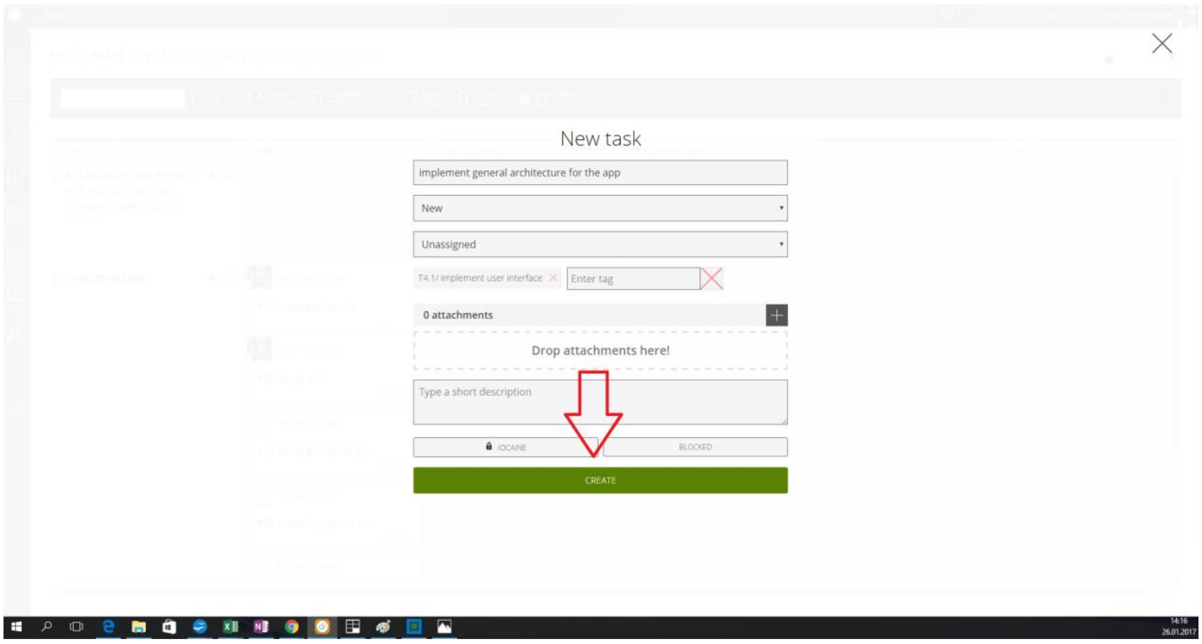
CREATE

7. Add a tag to **every** Taiga task:
   - the first part will be the Hotmaps task number (for example *T4.1*)
   - followed by the action you want to do.
     (T4.1/Setup the environment of hotmaps platform OR T4.1/Setup the user interface)

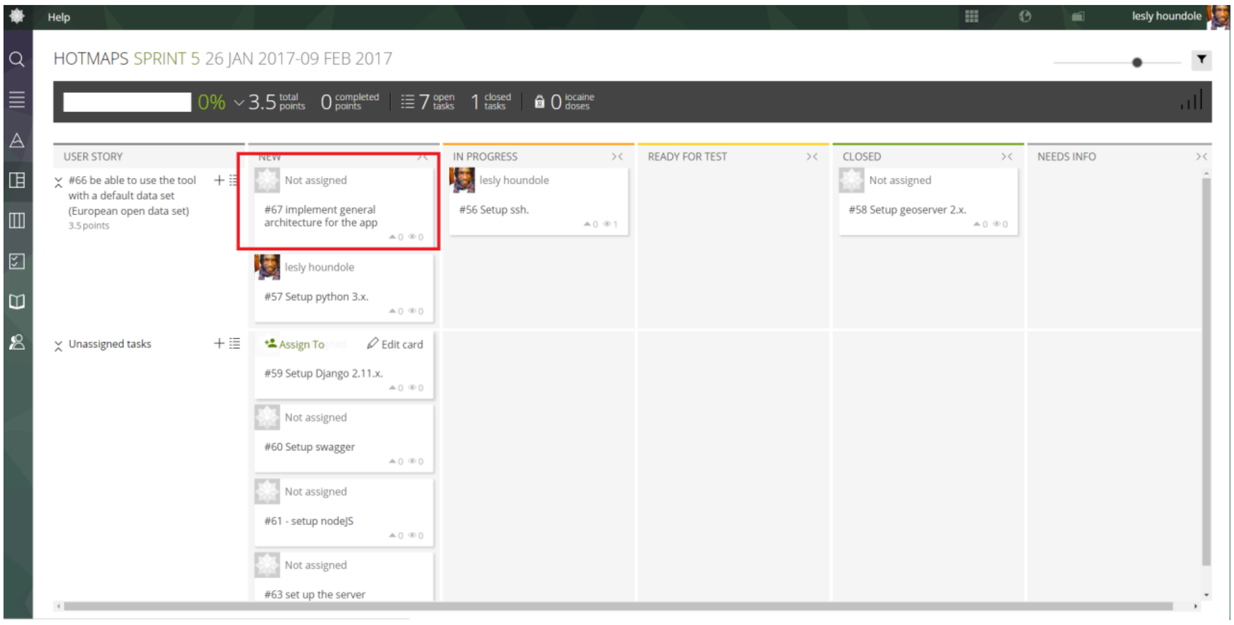
A **tag** corresponds to a **group** of Taiga tasks



8. click the button below to create this new task



9. the task newly created has been added to the sprint board



Help lesly houndole

HOTMAPS SPRINT 5 26 JAN 2017-09 FEB 2017

0% 3.5 total points 0 completed points 7 open tasks 1 closed tasks 0 iocaine doses

USER STORY | NEW | IN PROGRESS | READY FOR TEST | CLOSED | NEEDS INFO

#66 be able to use the tool with a default data set (European open data set) 3.5 points

NEW: Not assigned #67 implement general architecture for the app; lesly houndole #57 Setup python 3.x.

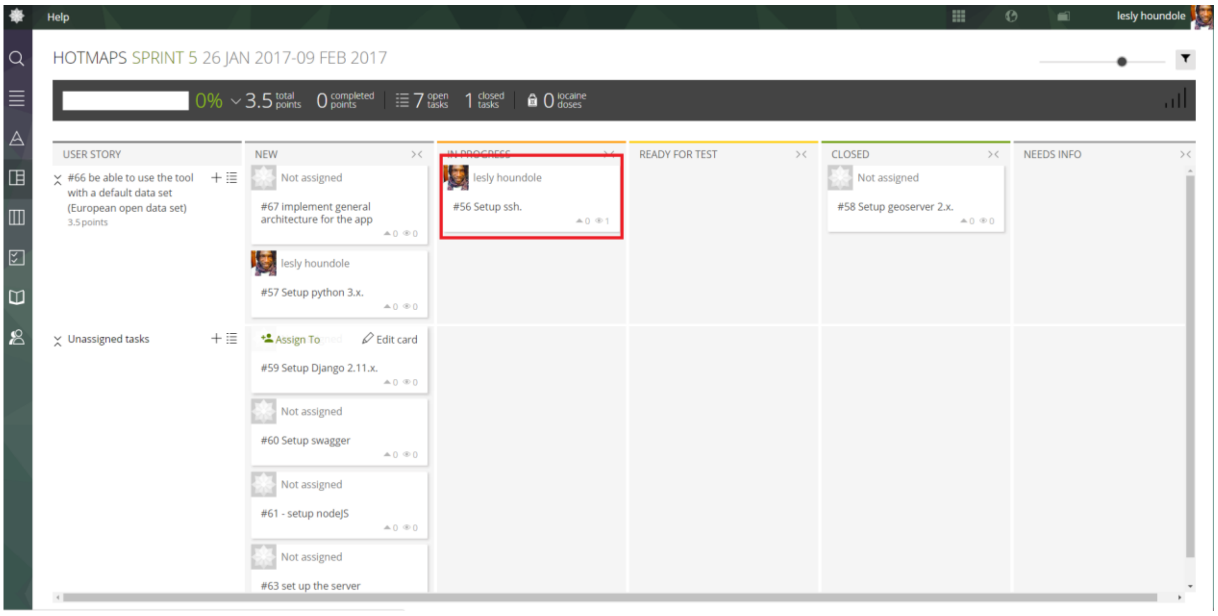
IN PROGRESS: lesly houndole #56 Setup ssh.

CLOSED: Not assigned #58 Setup geoserver 2.x.

Unassigned tasks

NEW: Assign To Edit card #59 Setup Django 2.11.x.; Not assigned #60 Setup swagger; Not assigned #61 - setup nodeJS; Not assigned #63 set up the server

10. You can change the task status using drag and drop



Help lesly houndole

HOTMAPS SPRINT 5 26 JAN 2017-09 FEB 2017

0% 3.5 total points 0 completed points 7 open tasks 1 closed tasks 0 iocaine doses

USER STORY | NEW | IN PROGRESS | READY FOR TEST | CLOSED | NEEDS INFO

#66 be able to use the tool with a default data set (European open data set) 3.5 points

NEW: Not assigned #67 implement general architecture for the app; lesly houndole #57 Setup python 3.x.

IN PROGRESS: lesly houndole #56 Setup ssh.

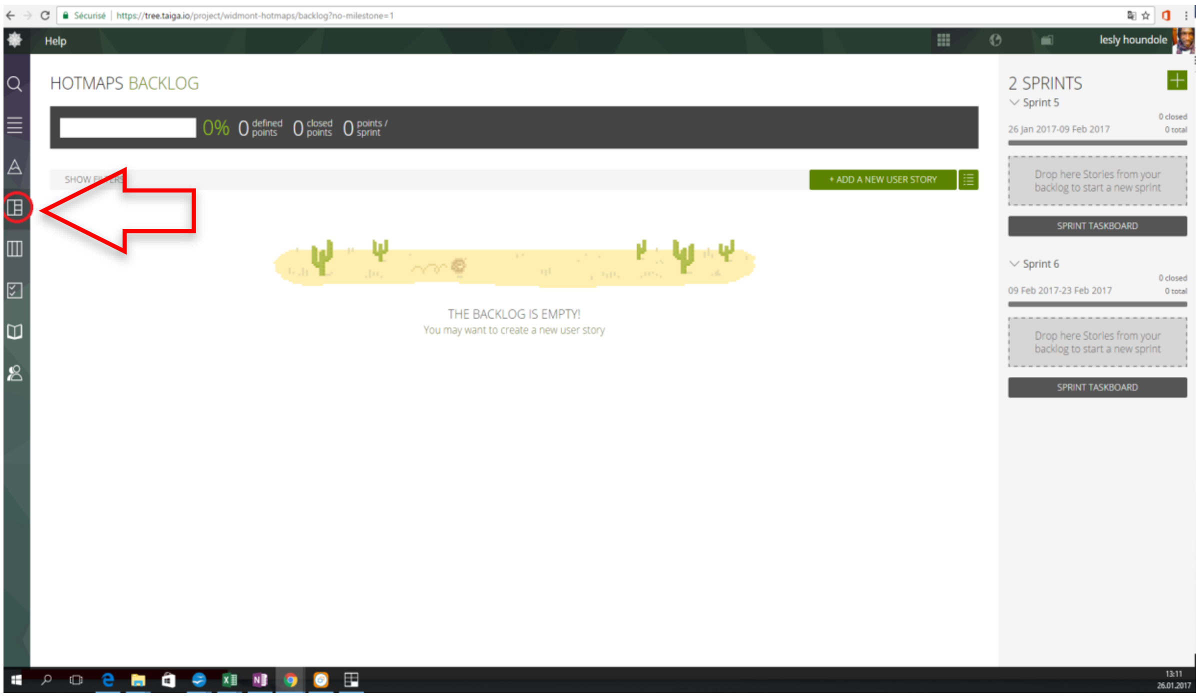
CLOSED: Not assigned #58 Setup geoserver 2.x.

Unassigned tasks

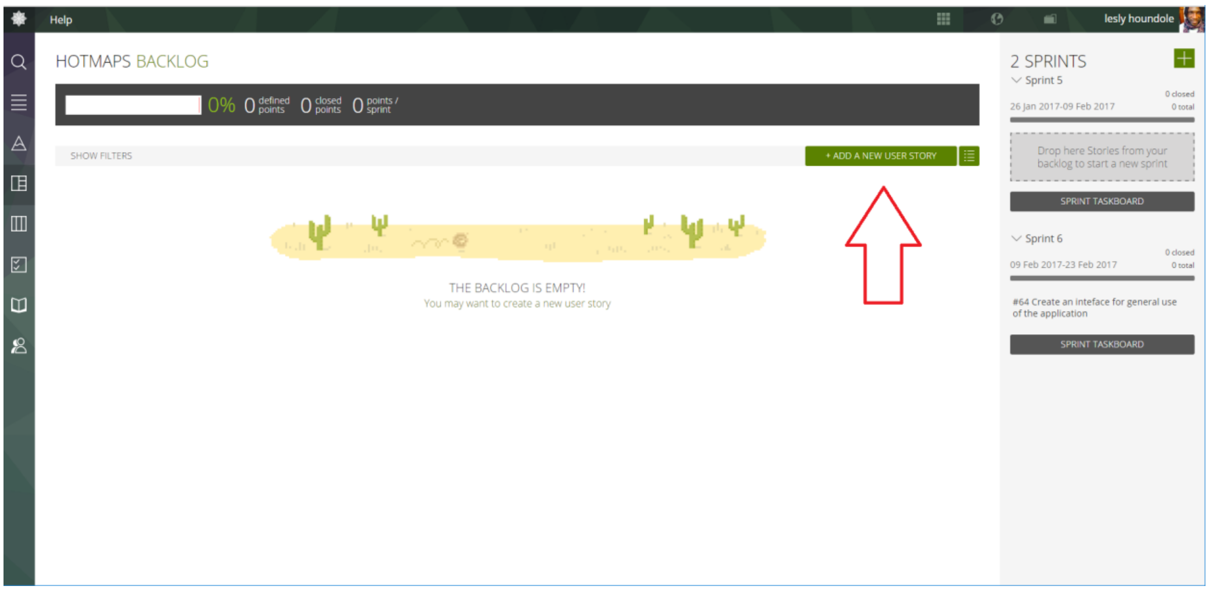
NEW: Assign To Edit card #59 Setup Django 2.11.x.; Not assigned #60 Setup swagger; Not assigned #61 - setup nodeJS; Not assigned #63 set up the server

User story creation

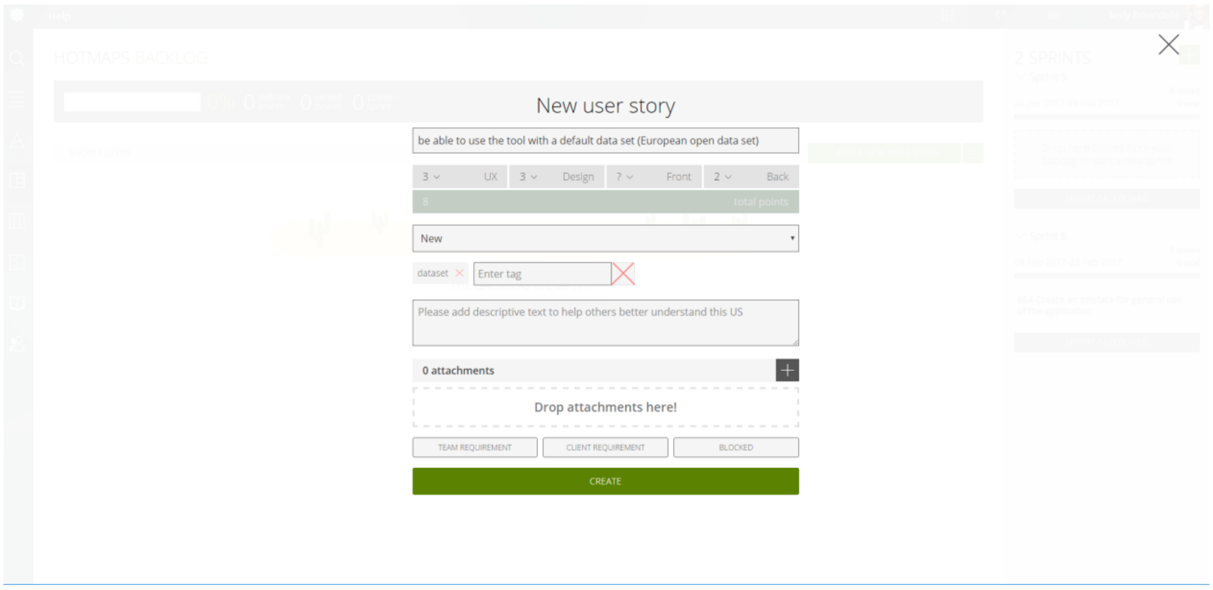
1. Click the HotMaps backlog button



2. Click this button to create a user story


Help
lesly houndole
HOTMAPS BACKLOG
0% 0 defined points 0 closed points 0 points / sprint
SHOW FILTERS
+ ADD A NEW USER STORY
THE BACKLOG IS EMPTY!
You may want to create a new user story
2 SPRINTS
Sprint 5
26 Jan 2017-09 Feb 2017
0 closed
0 total
Drop here Stories from your backlog to start a new sprint
SPRINT TASKBOARD
Sprint 6
09 Feb 2017-23 Feb 2017
#64 Create an inteface for general use of the application
SPRINT TASKBOARD

3. Create a new user story



New user story

be able to use the tool with a default data set (European open data set)

3 UX	3 Design	? Front	2 Back

8 total points

New

dataset

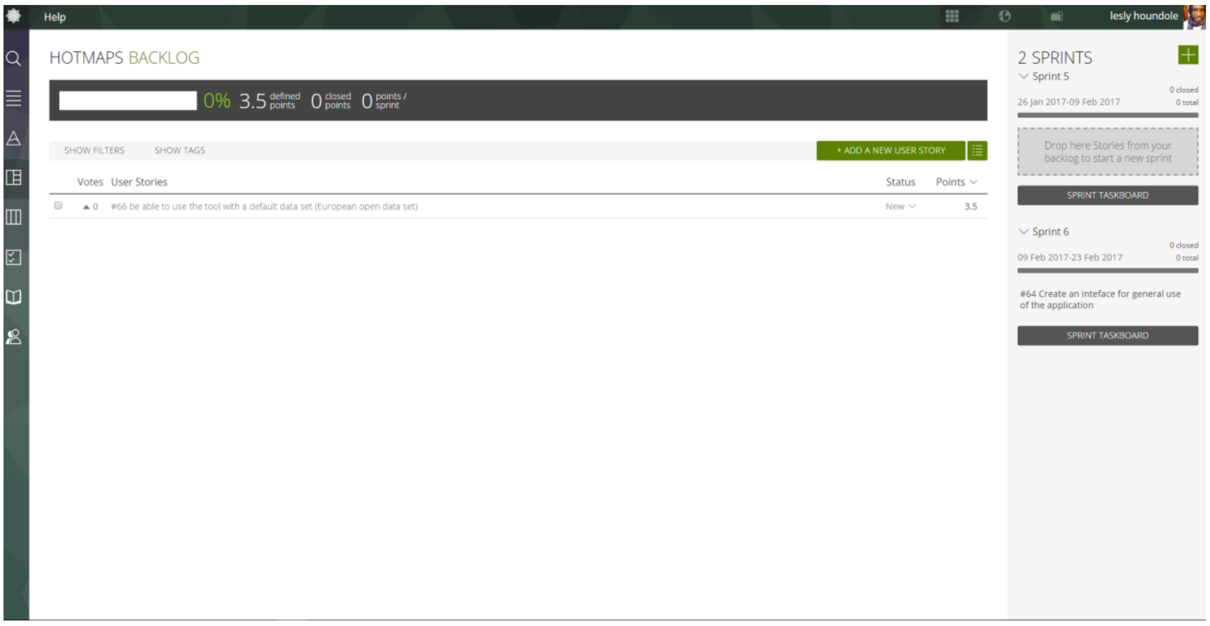
Enter tag

Please add descriptive text to help others better understand this US

0 attachments

Drop attachments here!

4. The user story has been added to the board



HOTMAPS BACKLOG

0% 3.5 defined points 0 closed points 0 points / sprint

Votes	User Stories	Status	Points
▲ 0	#66 be able to use the tool with a default data set (European open data set)	New	3.5

2 SPRINTS

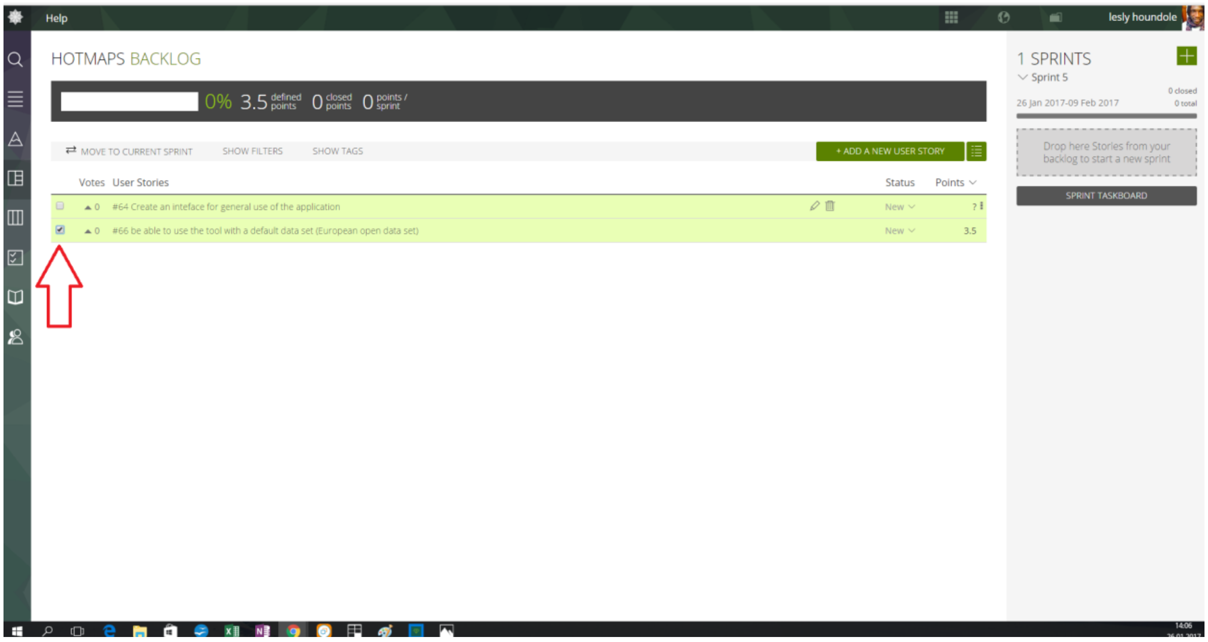
Sprint 5
26 Jan 2017-09 Feb 2017 0 closed 0 total

Drop here Stories from your backlog to start a new sprint

Sprint 6
09 Feb 2017-23 Feb 2017 0 closed 0 total

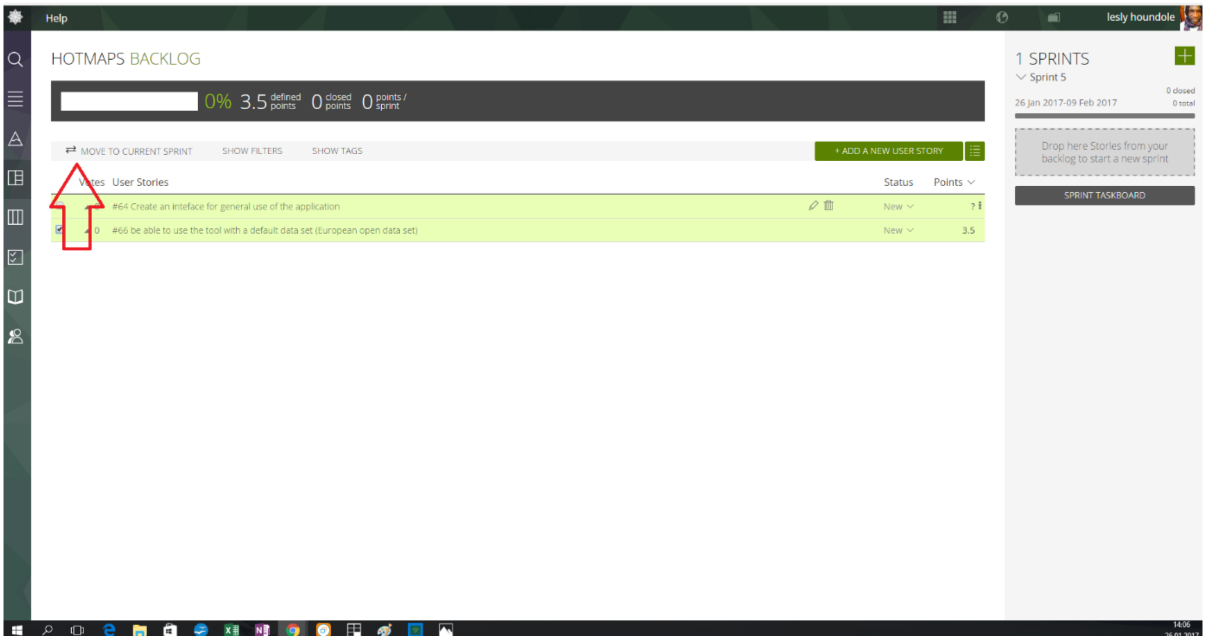
#64 Create an inteface for general use of the application

5. Select the user story you want to add to the current sprint



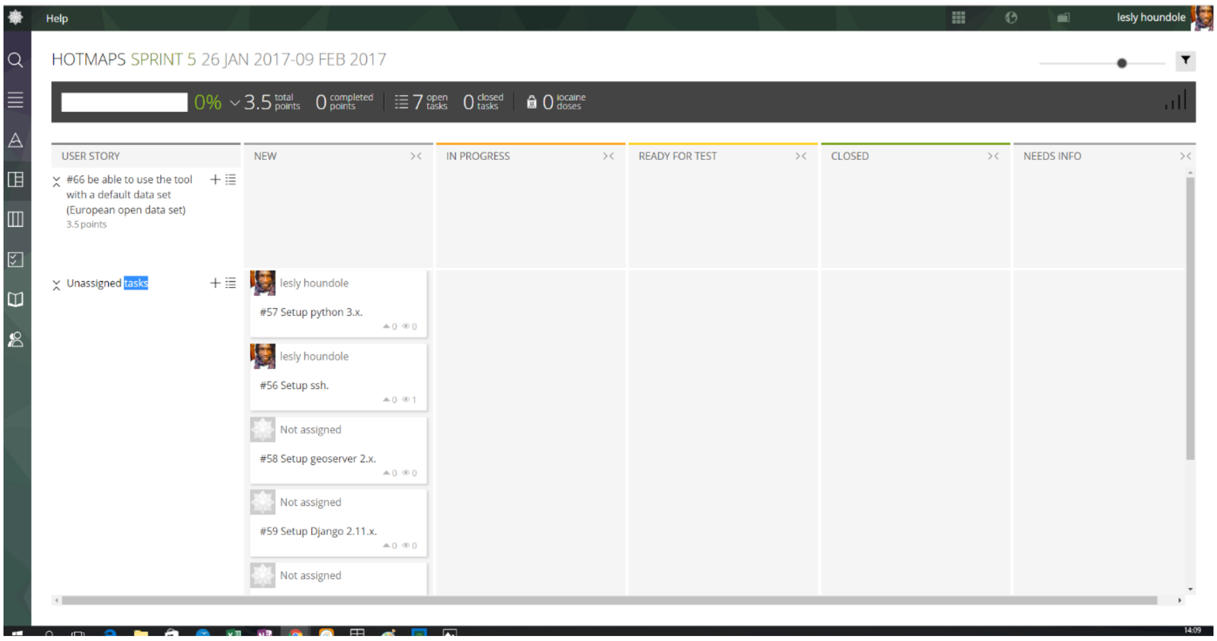
Votes	User Stories	Status	Points
▲ 0	#64 Create an inteface for general use of the application	New	?
▲ 0	#66 be able to use the tool with a default data set (European open data set)	New	3.5

6. Click this button to add the selected user story to the current sprint



Votes	User Stories	Status	Points
	#64 Create an inteface for general use of the application	New	?
0	#66 be able to use the tool with a default data set (European open data set)	New	3.5

7. In the sprint board the user story has been added and you can add tasks in it



HOTMAPS SPRINT 5 26 JAN 2017-09 FEB 2017

0% 3.5 total points 0 completed points 7 open tasks 0 closed tasks 0 iocaine doses

USER STORY	NEW	IN PROGRESS	READY FOR TEST	CLOSED	NEEDS INFO
#66 be able to use the tool with a default data set (European open data set) 3.5 points					
Unassigned tasks	lesly houndole #57 Setup python 3.x. ▲0 👁0 lesly houndole #56 Setup ssh. ▲0 👁1 Not assigned #58 Setup geoserver 2.x. ▲0 👁0 Not assigned #59 Setup Django 2.11.x. ▲0 👁0 Not assigned				

www.hotmaps-project.eu

Funded by the Horizon 2020 programme of the European Union