Tomat User Handbook

1 Introduction

This handbook is meant for users looking to configure and generate a mobile application or a mobile web-site with the Tomat solution. It supposes that you have an instance of the tomat server available and that you can build a mobile application or deploy a mobile web-site. Both of those processes are documented respectively on the Tomat github webpage and inside the Tomat user interface.

2 General principles

2.1 Architecture

Tomat is based on 3 autonomous components :

- A data server that will store, archive and serve all of the datas required for the other components, images included.
- A CMS style user interface that allows to configure the contents of the generated mobile applications (articles, news, points of interest, menus, …) and to import datas from external sources.
- A skeleton for a mobile app/web-site that allow to access the datas from the server and store it for offline use. Display different datas on ad-hoc interfaces (lists, details, maps, news, …). This skeleton can be enriched by various plugins to access other fonctionnalities of the phone (QRCode, Camera, GPS, Push, …)

2.2 Administration interface (« Back-end »)

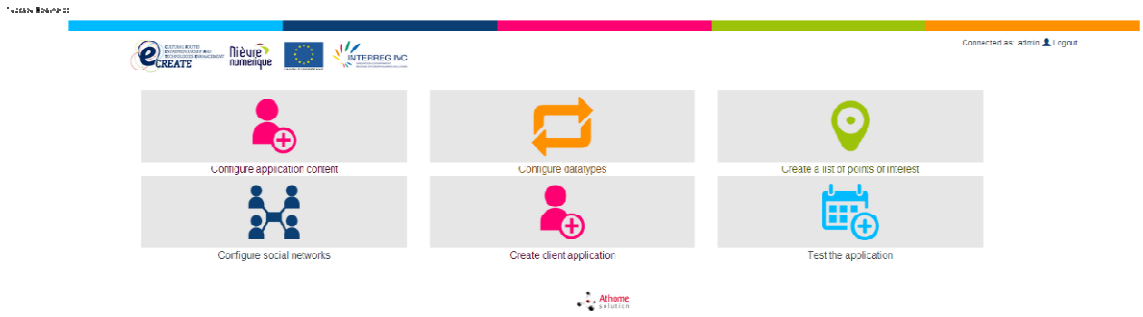


Figure 1 Home page

This will be the main component covered in this hand book. That's the part that allows to configure and then test (in web mode) your application.

2.3 Mobile application

A few basic files need to be edited to configure the application to suit your needs. Even though this is not a developer handbook, we will also cover the basic instructions to personalize and further modify the application.

If you need more detailed instructions on those subjects, it is advised to turn towards the tools used to build the application, and in particular **Ionic**, which is the main basis for the application.

3 Backend features

In this part we will cover the use of Tomat's backend various features one by one.

3.1 Application content

This part of the application allows the access to two central functionnalities of Tomat : the configuration of the application main menu, and the creation of articles that can be linked between themselves or from this menu.

3.1.1 Articles creation

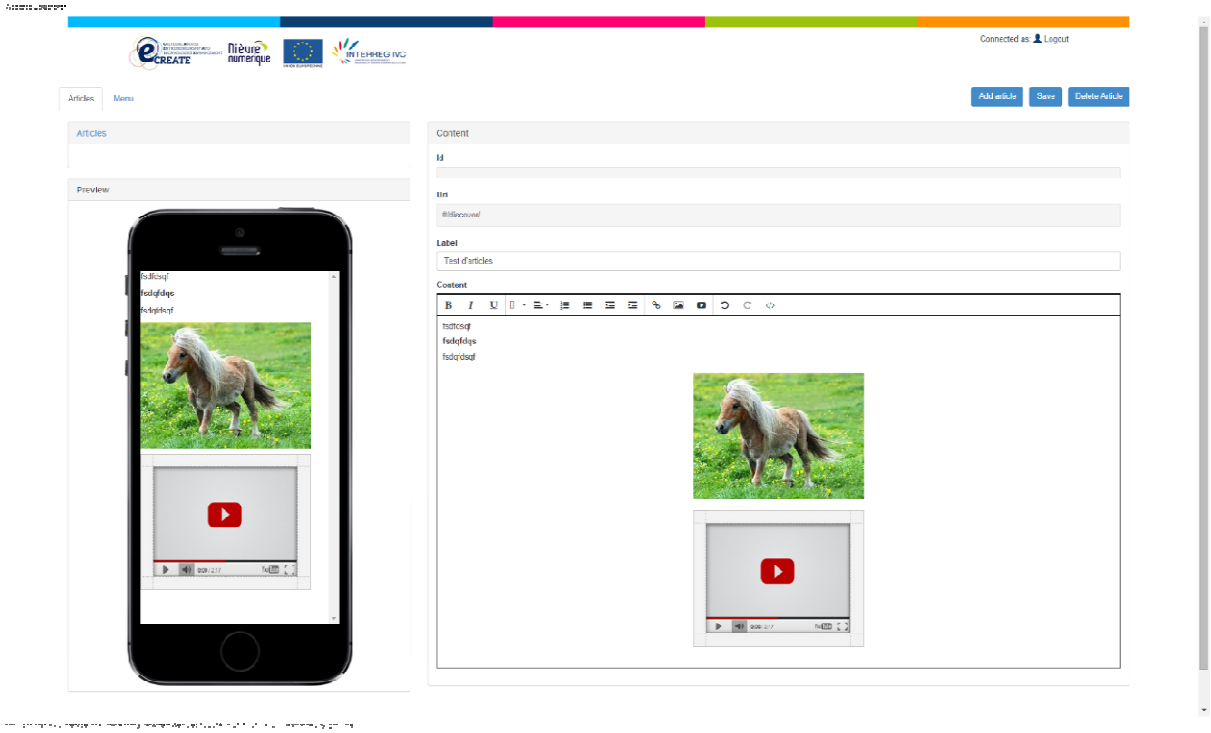


Figure 2The article edition and creation interface

This interface allow you to add articles and delete or modify previous articles. It contains a rich text editor allowing you to modify the content of the article, the list of the already created articles, and a small preview zone to get an idea of how your article will look once inside the application.

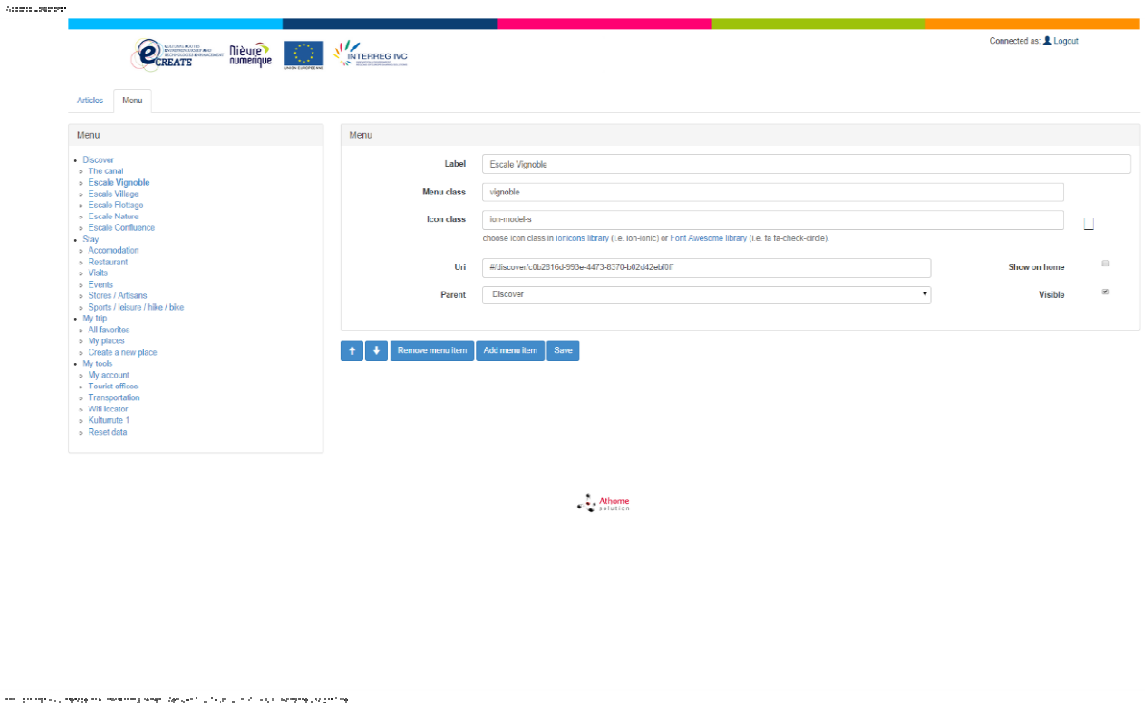
Each article, once created (depending on your browser configuration you might have to reload the page), will have a unique Id and Uri. This address will be used if you want to link the article from the menu or from another article (navigation inside the application work just like inside a web page).

The rest this interface shoud be self explanatory. Two features worth of note in the rich text editor :

The last button of the button bar allows you to see your article actual HTML markup, which can allow you to fine-tune your layout.

There is a feature to add a youtube video directly inside the article, you just need to recover the id of the video from the Youtube website. This is a convenient way to add videos to your websites without having to handle specific hosting, etc… issues. (You can still, of course, embed your own video inside articles. Please not however that no mobile platform supports Flash anymore, so be sure to add HTML5 videos).

3.1.2 Menu

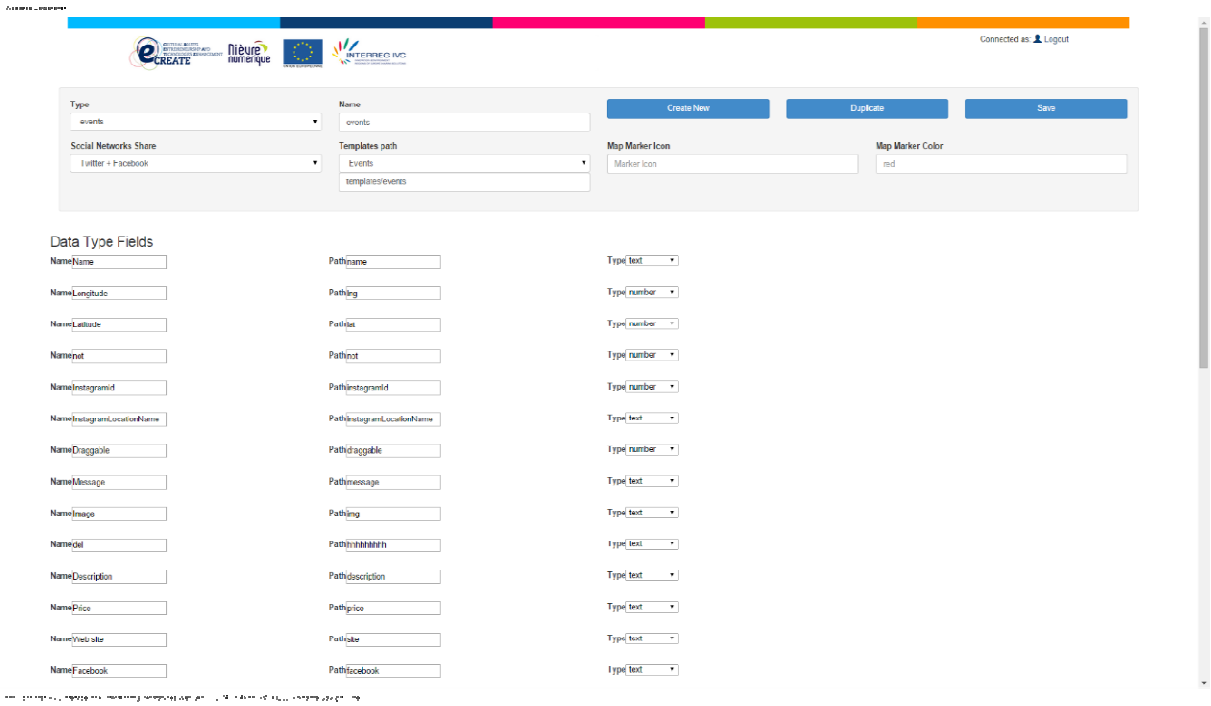


This page allows you to configure the content of the menus and sub menus of your application. There are four main main menu titles which you can of course modify to your liking, and each of them can contain as many menu items as you like.

Each menu item is composed of a label, a specific CSS class (only needed to do custom presentation for the menu), an icon class, which allows you to choose your menu icons amongst a wealth of predefined choices, or define your own class, the uri the menu item points to and the menu title it belongs to. Finally two check boxes allow you to display or hide a specific menu item, and to show it as a small icon on the home page or not (only a few items for each titles can be shown on the main page).

The uris can be of three kinds : links to article (of the form #/discover/<article-id>), links to POIs lists (of the form #/pois/<poi-type>) and links to specific functionality of the application (favorites, reset, …) which will correspond to custom pages created in the mobile application itself.

3.2 Datatypes



This part will allow you to define and configure your POIs datatypes. Default datatypes are created with the application, but you'll most likely want to modify them or create new ones. Note that if you create new datatypes with different fields, you must also create the corresponding page in the mobile application to present them.

A datatype will be used to define the data stored in a given type of POIs. Restaurant, say, will have position, price, and category, while Treks might have a starting position, a length, a difficulty and an average time.

Each datatype is comprised of several informations :

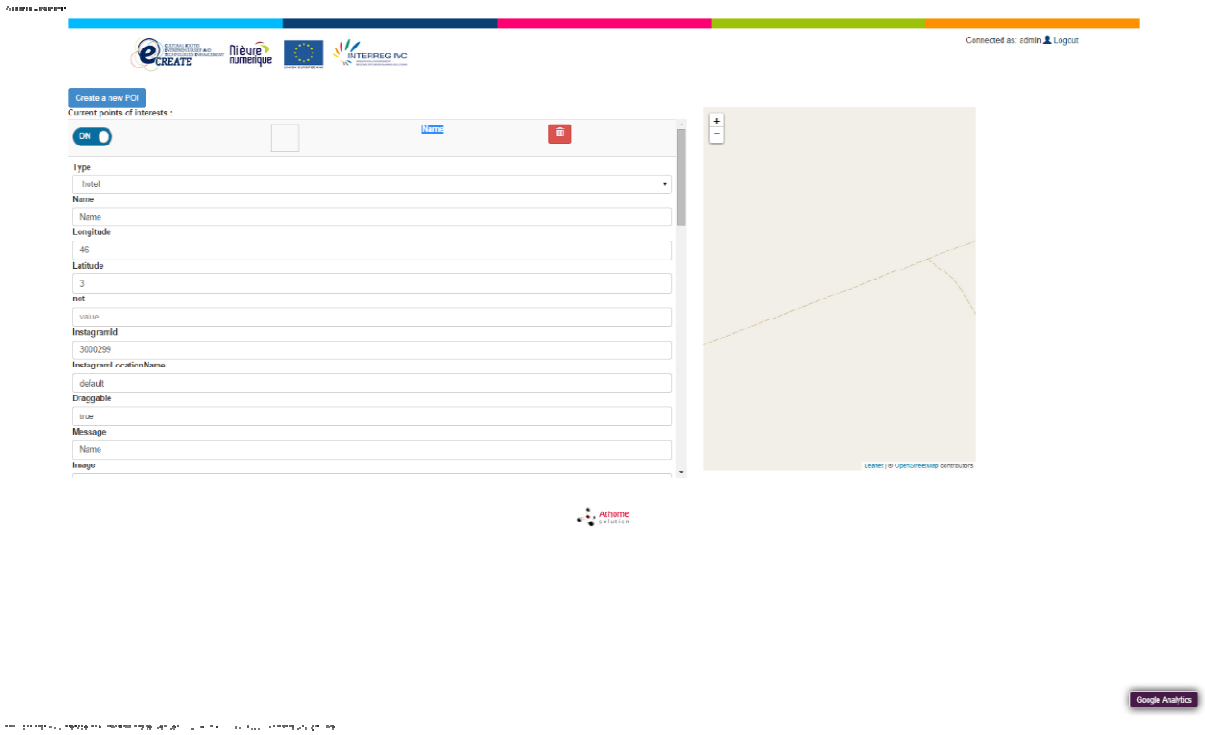
- Its name, which will be used to identify it throughout the application and to access it from the menus
- Its templates path, map marker icon and map marker color, which will be reused in the mobile application
- Fields. Each field is a data from the datatype that has to be either input by hand or imported from an external source. Each field has a name, a type, and a path. The name is used to identify the field, the path will be used to access it inside the mobile applications templates, and the type defines the type of datas to be expected (text, number, date, url, position, …).

Initially, you might want to start off with a simple datatype to familiarize yourself with the application, and then elaborate on that. Please not that if you modify the default datatypes, you might have to also modifiy the default templates inside the mobile application.

3.3 Create a list of points of interest

This will allow you to create your list of geographical points of interest to be displayed for your users. You can either input them manually, or connect to an external data source.

3.3.1 Manual creation

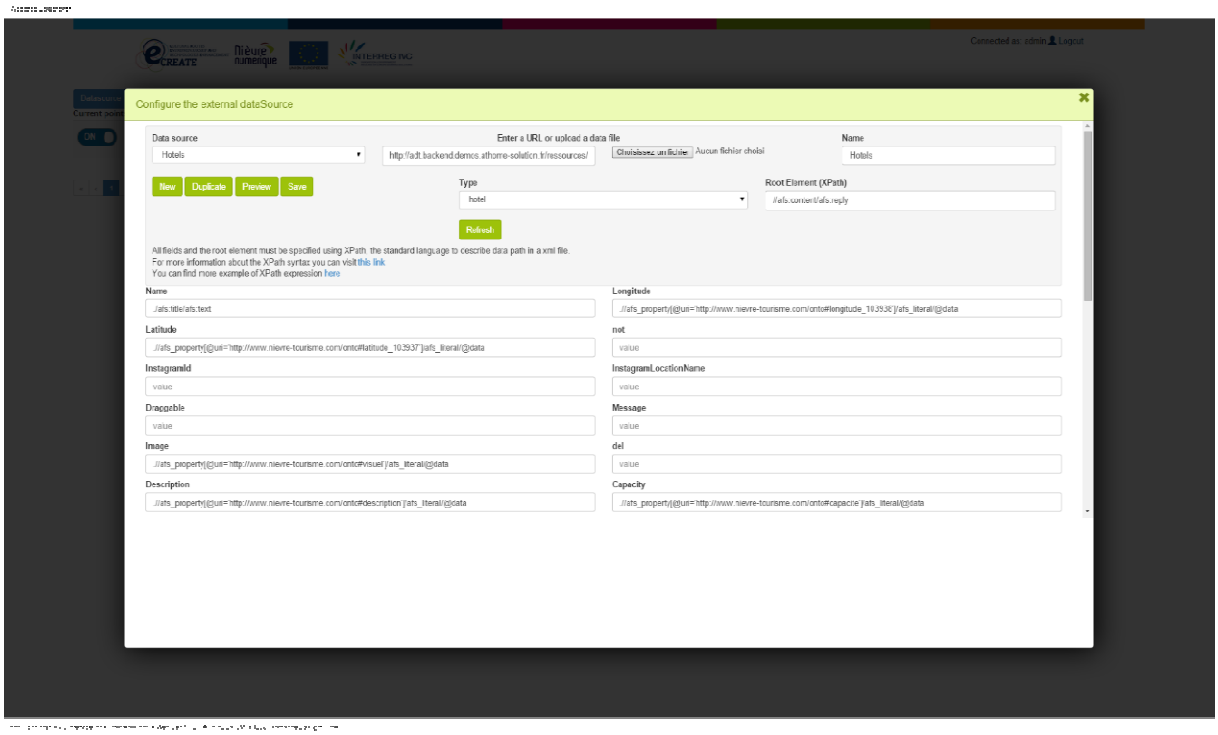


This interface will allow you to manually create POIs or edit the ones that have already been created. To edit a given POI you simply need to click on it. After that you can chose its type, which will of course change wich fields need to be filled, and fill the different data field for the POI.

The interface is tied to a map to allow you to set the POI position more easily.

Finally, each POI has a on/off button that allows to chose whether to display it in the application or to store it for future use.

3.3.2 Automated creation



This interface is the one which is used to import datas from external data sources. A different datasource much be created for each file or URL, even if they contain datas of the same type. Conversely, if the same file contains several types of poi, it must be imported several times.

The main component of a data source is either a URL containing the XML datas or a file. Each datasource must also have a name for future reference, and define the type of POI it will import into. The “refresh” button is useful if you want to make sure you are using the latest version of the type.

The rest of the document are composed of XPath expression. Explaining the workings of XPath in detail would be beyond the scope of this document (up-to-date links to XPath documentation can be found inside the interface), but here are the basic principles :

- The root element must contain the path to the XML content that correspond to one POI. For instance, if all your hotels are inside <hotel></hotel> elements, the root element could be //hotel
- After that each field must contain the path to its value, regardless of whether it is an element or an attribute. For instance, if each hotel element contains a <position lat=”45” lng=”3.3”/> element, the path for lat would be //position/@lat and the one for lng //position/@lng. But if the image url is contained inside an <illustration> element (like <illustration>http://test.img</illustration> the path would be //illustration.

Once the various fields are filled you can hit the “preview” button to display the first few items of your files, to check that everything is configured properly. Once that is done you can hit the save button to import/re-import your data inside your server.

3.4 Social network

This should be pretty self-explanatory. This will be needed only if you need advanced social network features or you want to link your application to a social network app. For basic social network sharing, everything is done inside the client application.

3.5 Create client application

This allows you to download the sources for the client application (which can also be found on github), and to access the instruction to install and build it. It also contains links to all of the relevant developer documentation for the components we used.

3.6 Test the application

If you are building a mobile web app this can point to a link allowing you to test your mobile application in real time.

4 Mobile development

The basic installation and deployment of the mobile application is already described inside Tomat. This part of the documentation will focus on describing the mobile application architecture, and pointing to which files and parts should be edited to customize the application.

Please note that some knowledge in HTML, CSS, Javascript or mobile app development might be required depending on the tasks you want to achieve. The description of those skills is beyond the scope of this documentation.

4.1 Architecture

Before you dive into the mobile app personalization, we strongly advise you to spend a few minutes skimming through ionic's documentation. This will give you a pretty good idea of how the framework used to build Tomat's work, and of the depth of its capabilities.

For most of your personalization, the only folder you will be interested in will be the app folder, which contains the html, css and javascript of the application. The various folder names should be self-evident, but for documentation purposes : the assets folder contains less files (that generates css). The fonts folder contains the font files used in the application. The I18n folder contains the internationalization file for your application. The images folder contains static images for the

application. The scripts folder contains all the javascript files that define the app. In particular, it contains two very interesting files config.js and routes.js, which we'll come back to later.

The styles folder contains the compiled css. You should not edit that unless you have disabled less compilation of for testing purposes (changes will be overridden by less compilation).

Finally the templates folder contains all of the html for the application.

4.2 Configuration

The configuration of the application mainly occurs in the config.js file. This file will contain the lists of locales the application can display, the url of the tomat server the application accesses, the message template for the social sharing of POIs or favorites. The html file to be used to display different kinds of POIs.

Besides the url of the tomat server, which should be edited to point to your test server at first, and then your production server, the rest should probably be left as is at the beginning. After you've added a new type of POIs, you'll want to add it to the config_poi list with the address of its templates, … The other configuration option should only be used in development scenarii

4.3 Personalization

The personalization of the application will mainly use two files :

- The files in the assets/less folder that will be used to modify the CSS of the application. We strongly advise you to test your application in web mode when you are editing those files, to be able to preview your modifications in real time
- The html in the templates folder.the main navigation pages for the application are at the root, and the templates for each kind of POIs are in a subfolder. These files make heavy use of Ionic compoments, so you might have to experiment a little bit to get the look&feel you want. As for CSS, testing in web-mode is strongly recommended.

4.4 Development (for advanced use only)

The native code for the different plugins can be found (and edited) in the plugins directory. This actually allow you to access the phone APIs at native level depending on the functionality you want to add or edit. More information on plugin maintenance and development can be found in the Phonegap/Cordova documentation.

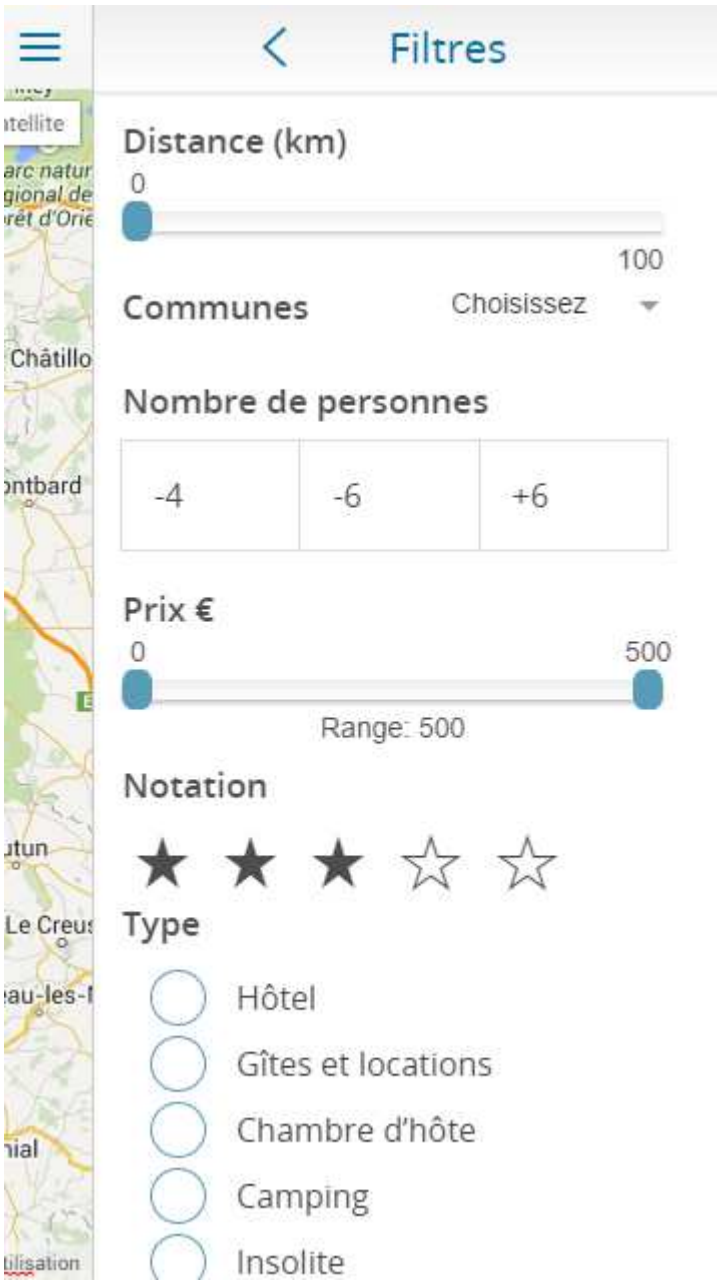
5 Mobile features

In this part, we'll take a quick look at Tomat existing mobile features and how to use them. There again, some parts might require some understanding of web or mobile development.

5.1 Geolocation and maps

This is a base feature of Tomat, and each kind of POI will automatically be geolocated unless prompted otherwise. The relevant templates can be found in the poi folder of the web application, even though the default should work form most usages.

5.2 Filters



Tomat includes a preset advanced filter system that can be used to filter through your POIs. If you want to modify or customize those filters, the relevant files can be found inside the templates/<poi-name>/filter.html file, and inside the list.js javascript file.

5.3 Social networks and sharing

Social network and sharing are enabled by default inside the mobile application. If you want to disable it you simply have to remove the share buttons from the relevant templates.

The share message and contents can be edited directly in the config.js file.

5.4 User datas and offline mode

Amongst Tomat native feature, each user can create an account, which will allow him to manage his favorites or user generated data, and which can also be used to get a list of the personal datas of the users of your app.

Another native feature is the local storage of all the application data. This allows your user to be able to use your application even in a scenario where their mobile internet access is lacking. Offline datas

can be updated at the click of a button at any time (when an internet connection is available of course)

5.5 Push

To enable push, you'll need to provide Tomat with the ids of your push accounts for the various providers. You can also use the push identifiers for the users stored in Tomat to push other datas to your user base.

Please note however that push is still considered a pretty intrusive feature, that should not be overused.

5.6 QR Codes

Through the use of a simple plugin http://phonegap.com/blog/build/barcodescanner-plugin/ a QR code scanner can be used anywhere in the application. The returned data can then be used for instance to generate a custom navigation page.