
Georiviere

Release 1.2.2

Makina Corpus

Aug 09, 2023

USER MANUAL

1	Common features	3
1.1	Interface	3
2	Modules	5
2.1	Streams	5
2.2	Descriptions	6
2.3	Knowledges	6
2.4	Follow-ups	6
2.5	Stations	7
2.6	Studies	7
2.7	Finance and administrative files	7
2.8	Proceedings	7
2.9	Interventions	7
3	Configuration	9
3.1	Email settings	11
3.2	API GeoRiviere Portal	11
4	Valorization	13
4.1	Portal	13
4.2	Map Base Layers	13
4.3	Layers	14
4.4	Groups	14
5	Import data	17
5.1	Import data	17
5.2	Import altimetry file	18
5.3	Import stations from Hub'Eau	18
5.4	Import data references from Sandre	19
5.5	Import zoning data from file	19
6	Install instructions	21
6.1	Requirements	21
6.2	Install	21
6.3	Update	22
7	Basic settings	23
7.1	Spatial reference identifier	23
7.2	Default Structure	23
7.3	Translations	23
7.4	Spatial Extent	23

8	Advanced settings	25
8.1	Georiviere settings	25
8.2	Based on Geotrek or Mapentity settings	25
8.3	Override translations	25
9	Install local Environment	27
9.1	Installation	27
9.2	Tests	27
9.3	Dependencies	28
10	Documentation	29
11	Publication	31
11.1	CI	31
11.2	Release	31
11.3	Docker image	31
12	Authors	33
12.1	Authors & trademark	33
13	Changelog	35
13.1	CHANGELOG	35
14	Indices and tables	41

Georiviere-admin is an application to handle aquatic environment (stream, river, watersheds...) in a given territory. It is a [Django](#) application, based on [Geotrek-admin](#) and [Mapentity](#).

COMMON FEATURES

Since Georiviere is based on Geotrek-admin, it has the same features:

- display and edit contents with their geom
- export contents in ODT/DOC, PDF, GPX or shape files
- add linked files
- history

1.1 Interface

The screenshot displays the Georiviere interface with a list of stations on the left and a map on the right. Red arrows highlight key features:

- Liste 14**: Number of results.
- + Ajouter une station**: Add new object.
- Filtrer**: Filter.
- Navigation**: Map navigation controls.
- Zoomer sur une commune, un secteur ou un bassin versant**: Zoom on a commune, sector, or watershed.
- Rechercher**: Search.
- Rechercher parmi les résultats**: Search among results.
- Parcourir les pages de résultats**: Browse result pages.
- Exporter la liste**: Export list.

The list of stations includes columns for 'Code station' and 'Libellé station'. The map shows a topographic view of a river area with various locations marked.

To get more detail on how add and edit contents, have a look on [Geotrek-admin documentation](#) (in french).

MODULES

Several modules are available in Georiviere, to display and edit professional content:

- Streams
- Descriptions (usage, land, morphology, status)
- Knowledges (vegetation, work or other knowledges)
- Follow-ups for a given knowledge
- Stations (hydrometric, temperature, physico-chemical quality or other types)
- Studies
- Finance and administrative files
- Proceedings
- Interventions (on knowledges)

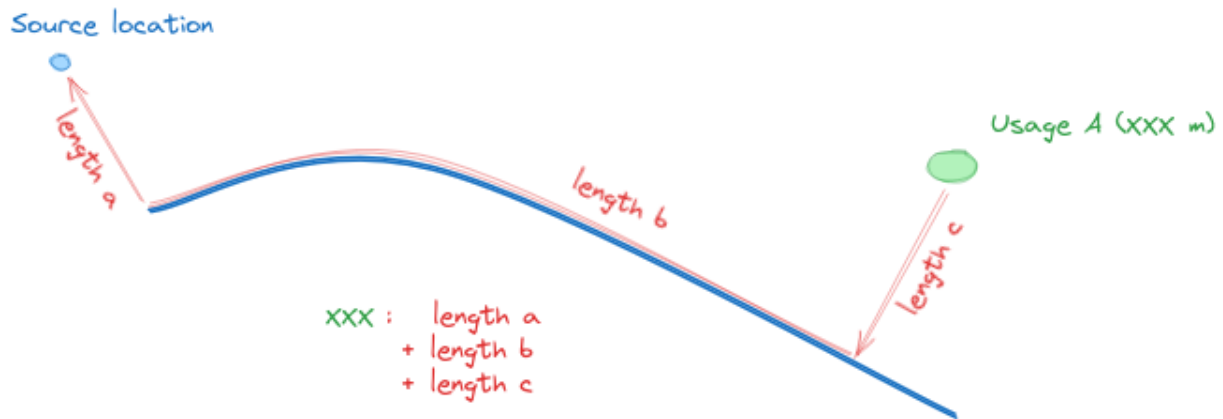
In a content detail page, nearby other contents are displayed.

2.1 Streams

Streams is a line with a source location and flow type. Length is computed from geometry.

Distances to every objects are computed during the creation of an object and stocked in the table `distancetosource`. It's the distance of the shortest path between the object and the stream added with the length between the point of junction between the shortest path and the stream and the source location.

If the source location is not at the beginning of the stream, we also add the distance of the shortest path between source location and the stream.



2.2 Descriptions

Four description models are available:

- usage
- land
- morphology
- status

Land and morphology have both a geom relative to a stream, and are created along to a stream on its creation. They can be cut to edit more precisely their attributes.

Usage and status are both standalone geometry and can be whatever point, line or polygon.

2.3 Knowledges

Knowledges can be a point, line or polygon about a stream, of vegetation, work or other type.

Vegetation and work type knowledges have specific fields. For this, first value in `type_connaissance` = végétation and second value = ouvrage.

2.4 Follow-ups

Follow-ups can be added to a knowledge, to take regular readings related to this knowledge.

2.5 Stations

Stations are points of measures, but they can be line or polygon too. Tracked parameters can be added to a station, with their name, measure and transmission frequency, etc.

Station of hydrometric, temperature, physico-chemical quality can be imported from Hub'Eau API.

2.6 Studies

A study is just a content with authors and year.

2.7 Finance and administrative files

Estimated or actual costs, fundings, or organisations involved in a project can be filled in an administrative file.

Every content in georiviere can be linked to an administrative file with operation, and for each you can edit its estimated, material, sub-contracting or man-days costs.

2.8 Proceedings

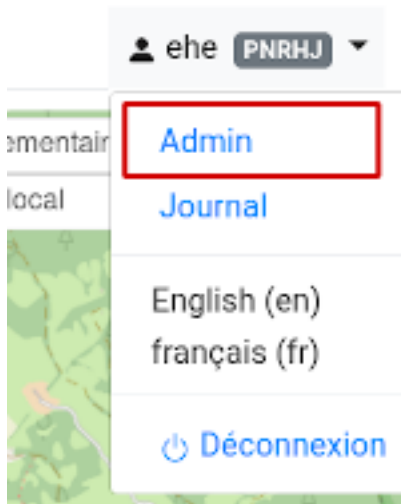
A proceeding can list all juridic events related to it.

2.9 Interventions

Intervention is a maintenance intervention related to a follow-up.

CONFIGURATION

To customize lists for each module, go to django administration page.



- **Description**

- Bank states
- Facies diversities
- Flow types
- Granulometric diversities
- Habitat types
- Habitats diversities
- Land types
- Plan layout types
- Sediment dynamics
- Status types
- Usage types
- Working space types

- **Finances and administration**

- Admin file domains

- Admin file types
 - Administrative operations
 - Job categories
 - Organisms
- **Knowledge**
 - Age class diversities
 - Follow-up types
 - Knowledge types
 - **Vegetations:**
 - * Specific diversities
 - * Vegetation states
 - * Vegetation stratas
 - * Vegetation thickness types
 - * Vegetation types
 - **Work:**
 - * Work bank effects
 - * Work fish continuity effects
 - * Work materials
 - * Work sediment effects
 - * Work states
 - * Work stream influences
 - * Work types
- Main: File types
- **Maintenance**
 - Intervention's disorders
 - Intervention's stakes
 - Intervention's statuses
 - Intervention's types
 - Interventions
- **Observations**
 - Parameter categories
 - Parameters
 - Station profiles
 - Units
- **Portal**
 - Portals

- Map base layers
- Group layers
- Layers
- Proceeding: Event types
- Studies Study types
- **Watershed**
 - Watershed types
 - Watersheds
- **Zoning**
 - Cities
 - Districts
 - Restricted area types
 - Restricted areas

3.1 Email settings

Georiviere-admin will send emails:

- to administrators when internal errors occur
- to managers when a contribution is created
- to contributors when a contribution is created

Email configuration takes place in `var/conf/custom.py`, where you control recipients emails (ADMINS, MANAGERS) and email server configuration.

You can test your configuration with the following command. A fake email will be sent to the managers:

```
docker-compose run --rm web ./manage.py sendtestemail --managers
```

If you don't want to send an email to contributors when they create a contribution on portal website, change this setting in `var/conf/custom.py`:

```
SEND_REPORT_ACK = False
```

3.2 API GeoRiviere Portal

To enable the schema of your api you need to modify the settings :

```
API_SCHEMA = True
```

It will allow to get the schema with the xml format :

<http://domain.com/api/portal/schema/>

For accessing the api as a swagger, you need to modify the settings :

```
API_SWAGGER = True
```

Then, you can access the swagger of portals (<https://swagger.io/>):

<http://domain.com/api/portal/schema/swagger/>

Last settings allow you to show the api as redoc (<https://redocly.com/redoc/>)

```
API_REDOC = True
```

you can access this version of the schema with :

<http://domain.com/api/portal/schema/redoc/>

VALORIZATION

To generate a good structure of your portal Georiviere, firstly you need to configurate in admin portals, map base layers, layers and groups.

4.1 Portal

Fields :

- Name
- Web site
- Title
- Description
- Main color
- Min zoom on the map of your portal
- Max zoom on the map of your portal
- Spatial extent of your portal

4.2 Map Base Layers

Map base layers are linked with one unique portal

- Label
- Url
- Min zoom of this map base layer
- Max zoom of this map base layer
- Attribution
- Portal
- Order

4.3 Layers

Layers are generated for each type of element you could add on your portal web site :

- Contributions
- Sensitivity
- Districts
- Cities
- POIs (each category has it layer)
- Streams
- Watersheds

Firstly, you need to create your portal before accessing layers. Then, you will be able to add them in a group of layer, change its labels. ...

If you create a new category of poi, a new layer is generated.

The fields of layers are :

- Label
- Group
- Active by default
- Style
- Order
- Hidden

Style can be changed following the documentation of leaflet, for example: `{“fillColor”: “#d4b485”}`

(<https://leafletjs.com/reference.html#path-stroke>)

When a layer is hidden, the layer is not used in the portal.

The label is used in the portal and will be shown it.

4.4 Groups

You can create as many group as you need and can add layers in the group.

When a layer is not assigned to a group, the layer is grouped in a default group.

The fields of groups are :

- Label
- Order

It's better to create everything in this order :

- Create your portal
- Create your base layers
- Create the groups

- Modify the generated layers

IMPORT DATA

To import data, you have to run these commands from the server where Georiviere-admin is hosted.

5.1 Import data

Put your data file named CoursEau_FXX.shp in var/ folder, and run command

Custom your import file named import_bdtopage.py et put this file in georiviere/

```
#!/usr/bin/env python
import os
import django
import logging
from django.contrib.gis.gdal import DataSource

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'georiviere.settings')
django.setup()
from geotrek.authent.models import Structure
from django.contrib.gis.geos.collections import MultiLineString
from georiviere.river.models import Stream

# get first structure, adapt with correct structure
structure = Structure.objects.first()

ds = DataSource('var/CoursEau_FXX.shp')

for feat in ds[0][4:]:
    try:
        # get geos object
        geom = feat.geom.geos
        # force 2D
        geom.dim = 2
        name = feat.get('name') or 'No name'
        flow = feat.get('flow')
        if flow == 1:
            flow = 1
        elif flow == 2:
            flow = 2
        else:
            flow = 0
```

(continues on next page)

(continued from previous page)

```

        stream = Stream.objects.create(structure=structure,
                                        geom=feat.geom.
↳ geos,                                name=name,
                                        flow=flow)

    except Exception as exc:
        logging.warn(exc, feat.geom.geos.ewkt)

```

And run command : `docker compose run --rm web ./import_bdtotopage.py`

5.2 Import altimetry file

Put your altimetry file in `var/` folder, and run command

```
docker-compose run --rm web ./manage.py loadDEM <dem_path>
```

where `<dem_path>` is `/opt/georiviere-admin/var/my_dem_file.tiff`

5.3 Import stations from Hub'Eau

Stations can be imported from french Hub'Eau APIs :

- Temperature stations with `import_temperature_stations`
- Hydrometry with `import_hydrometric_stations`
- Physico-chemical quality with `import_pcquality_stations`
- Hydrobiology stations with `import_hydrobiologie_stations`

Optional arguments:

```

--department DEPARTMENT [DEPARTMENT ...]
                        Department code
-p, --with-parameters  Get also parameter tracked by the station
--size SIZE            Results per page

```

Example:

```
docker-compose run --rm web ./manage.py import_pcquality_stations --department 39,25
```

5.4 Import data references from Sandre

Some data references can be imported from Sandre, for now only units are imported.

Usage:

```
docker-compose run --rm web ./manage.py import_reference_data
```

5.5 Import zoning data from file

Put your files into var/ folder as for altimetry profile import.

5.5.1 Load cities

Load Cities from a file within the spatial extent : `loadcities <file_path>`

Optional arguments::

```
--code-attribute CODE, -c CODE      Name of the code's attribute inside the file
--name-attribute NAME, -n NAME      Name of the name's attribute inside the file
--encoding ENCODING, -e ENCODING    File encoding, default utf-8
--srid SRID, -s SRID               File's SRID
--intersect, -i                   Check features intersect spatial extent and not only within
```

Example:

```
docker compose run --rm web ./manage.py loadcities /opt/georiviere-admin/var/commune.shp
↪ --name-attribute nom --code-attribute insee_com
```

5.5.2 Load districts

Load Districts from a file within the spatial extent `loaddistricts <file_path>`

Optional arguments::

```
-h, --help                show this help message and exit
--name-attribute NAME, -n NAME      Name of the name's attribute inside the file
--encoding ENCODING, -e ENCODING    File encoding, default utf-8
--srid SRID, -s SRID           File's SRID
--intersect, -i              Check features intersect spatial extent and not only within
```

Example:

```
docker compose run --rm web ./manage.py loaddistricts /opt/georiviere-admin/var/epci.shp
↪ --name-attribute nom --code-attribute code_siren
```

5.5.3 Load Restricted Area

Load Restricted Area from a file within the spatial extent loadrestrictedareas <file_path> and specify the name of the Area type

Optional arguments::

```
-h, --help            show this help message and exit
--name-attribute NAME, -n NAME
                        Name of the name's attribute inside the file
--encoding ENCODING, -e ENCODING
                        File encoding, default utf-8
--srid SRID, -s SRID  File's SRID
--intersect, -i       Check features intersect spatial extent and not only within
```

Example:

```
docker compose run --rm web ./manage.py loadrestrictedareas /opt/georiviere-admin/var/
↪ pnrhj.shp PNR --name-attribute nom
```

5.5.4 Import watershed

To import, use QGIS and edit watershed_watershed layer and specify “name” and “watershed_type_id” in attributes

INSTALL INSTRUCTIONS

6.1 Requirements

- **You need docker installed. Docker-compose is recommended in the configuration below.**
See [Docker](#) and [Docker compose](#) install documentations.
- **Optional** : if you want to use external database, prepare a postgresql 10+ postgres2.5+ database with postgres and postgres_raster enabled, and a dedicated user.

You can use external database by commenting postgres container and volume references in docker-compose.yml, and set variables :

- POSTGRES_HOST
- PGPORT
- POSTGRES_USER
- POSTGRES_PASSWORD
- POSTGRES_DB

Add local IPs in *pg_hba.conf* to allow connection from docker containers to your database.

- You can use external nginx proxy. Edit provided nginx conf file and comment nginx references in docker-compose.yml. Fix web:8000 to 127.0.0.1:8000 in nginx.conf.

6.2 Install

- Download [zip package](#)
- Unzip it where you want

```
unzip install.zip  
cd georiviere
```

- Prepare environment variables

```
mv .env.dist .env
```

-> **Set all required values**

- Pull images

```
docker-compose pull
```

(Use ``docker compose`` **if** docker-compose version is `>=2.4.x`)

- Init default var folder

```
docker-compose run --rm web bash -c "exit"
```

- Set at least these variables in `var/conf/custom.py`:

- SRID
- DEFAULT_STRUCTURE_NAME
- SPATIAL_EXTENT

As geotrek overlay, these settings should be set BEFORE database initialization. See [Basic settings](#) for details

- Init database and project config

```
docker-compose run --rm web update.sh
```

- Create your super user

```
docker-compose run --rm web ./manage.py createsuperuser
```

- Load initial data

```
docker-compose run --rm web ./manage.py loaddata georiviere/**/fixtures/basic.json
```

- Launch stack

```
docker-compose up -d
```

6.3 Update

- Read [release notes](#) about bugfix, news and breaking changes.
- Backup your data (database and var folder)
- Pull latest image

```
docker-compose pull
```

- Run post update script

```
docker-compose run --rm web update.sh
```

- Relaunch you docker-compose stack

```
docker-compose down  
docker-compose up -d
```

BASIC SETTINGS

Settings can be overridden in `var/conf/custom.py` file.

Basic settings should be defined on installation. See *Geotrek-admin documentation* <<https://geotrek.readthedocs.io/en/master/advanced-configuration.html#basic-settings>> for details.

7.1 Spatial reference identifier

```
SRID = 2154
```

Spatial reference identifier of your database. Default 2154 is RGF93 / Lambert-93 - France

7.2 Default Structure

```
DEFAULT_STRUCTURE_NAME = "GEOTEAM"
```

Name for your default structure.

7.3 Translations

```
MODELTRANSLATION_LANGUAGES = ('en', 'fr', 'it', 'es')
```

Languages of your project. It will be used to generate fields for translations. (ex: `description_fr`, `description_en`)

7.4 Spatial Extent

```
SPATIAL_EXTENT = (105000, 6150000, 1100000, 7150000)
```

Boundingbox of your project : x minimum , y minimum , x max, y max

ADVANCED SETTINGS

More settings can be overridden in `var/conf/custom.py` file.

After any change in `custom.py`, run:

```
docker compose restart
```

8.1 Georiviere settings

Base intersection margin

```
BASE_INTERSECTION_MARGIN = 2000
```

8.2 Based on Geotrek or Mapentity settings

Some settings come from Geotrek-admin or Mapentity, on which Georiviere is based:

- [Email settings](#)
- [Change or add WMTS tiles layers](#)
- [Map layers colors and style](#)

See [Geotrek-admin documentation](#) for further information.

8.3 Override translations

You can override default translation files available in each module

Don't edit these default files, use them to find which words you want to override.

Create the custom translations destination folder:

Create a `django.po` file in `var/conf/extra_locale` directory. You can do one folder and one `django.po` file for each language (example `var/conf/extra_locale/fr/LC_MESSAGES/django.po` for French translation overriding)

Override the translations that you want in these files.

Example of content for the French translation overriding:

```
# MY FRENCH CUSTOM TRANSLATION
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR <EMAIL@ADDRESS>, YEAR.
#
msgid ""
msgstr ""
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2018-11-15 15:32+0200\n"
"PO-Revision-Date: 2018-11-15 15:33+0100\n"
"Last-Translator: \n"
"Language-Team: LANGUAGE <LL@li.org>\n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=UTF-8\n"
"Content-Transfer-Encoding: 8bit\n"
"Project-Id-Verésion: PACKAGE VERSION\n"
"Plural-Forms: nplurals=2; plural=(n > 1);\n"
"Project-Id-Version: \n"
"X-Generator: Poedit 1.5.4\n"

msgid "City"
msgstr "Région"

msgid "District"
msgstr "Pays"
```

Apply changes (French translation in this example) :

```
sudo docker-compose run --rm web update.sh
```

INSTALL LOCAL ENVIRONMENT

9.1 Installation

- Configuration

To get local environment working, we recommend to use a custom domain, as 'georiviere.local'. Define it in your /etc/hosts.

Copy the env dist file

```
cp .env.dist .env
```

Set required values, for postgres database access

- Init database:

```
docker-compose run --rm web ./manage.py migrate
```

- Create user:

```
docker-compose run --rm web ./manage.py createsuperuser
```

- Run:

```
docker-compose up
```

9.2 Tests

- Launch tests :

```
docker-compose run --rm web ./manage.py test
```

- With coverage :

```
docker-compose run --rm web coverage run ./manage.py test
docker-compose run --rm web coverage report -m
```

9.3 Dependencies

- Manage all project dependencies with pip-tools
- Use included pip-tools to generate requirements (python version should match georiviere version)

Global dependencies:

Set global dependency in requirements.in

```
docker-compose run --rm web pip-compile
```

pip-tools does not upgrade any package by default. Package is upgrade only if new dependency require another version that already fixed in requirements.txt file.

To upgrade a package, run:

```
docker-compose run --rm web pip-compile --upgrade-package django==3.1.*
```

Development packages are separated in dev-requirements.in. dev-requirements.txt depends on requirements.txt.

```
docker-compose run --rm web pip-compile dev-requirements.in
```

Warning: Geotrek is used as main library of this project. Sub-dependencies are not yet managed in geotrek setup.py. When you update geotrek, you should update requirements according geotrek dependencies versions.

DOCUMENTATION

We use sphinx doc and sphinx-rtd-theme.

Requirements are included.

To compile and test documentation on local environment, run :

```
docker-compose run --workdir /opt/georiviere-admin/docs --rm web make html
```


PUBLICATION

11.1 CI

- Each edition runs a CI build.
- All Georiviere-admin maintainers can review or merge Pull Requests.
- First time contributor not in maintainer team can request to be added. Maintainers can accept its first pull request to allow CI build.

11.2 Release

To release a new version :

- set version to georiviere/VERSION file.
- set changelog infos in docs/changelog.rst
- push or merge to master
- Go to <https://github.com/Georiviere/Georiviere-admin/releases>.
- Click on “Draft a new release”
- set new tag according older ones.
- Copy / paste changelog for version in release notes.
- In the end, CI publish publish new docker image to github packages.

11.3 Docker image

- Docker image is published after each release in Georiviere github repository: `ghcr.io/georiviere/georiviere-admin:latest`

AUTHORS

12.1 Authors & trademark



- **Georiviere**, community software available under Open Source licence.
- Trademark is registered to **Institut National de la Propriété Intellectuelle** as a word mark under n°4691040.
- Brand policy

12.1.1 Makina Corpus

- Emmanuelle Helly
- Timothée de Montety
- Jean-Etienne Castagnede

12.1.2 Credits

Icons

- Stream : Creative Commons, Adrien Coquet, FR
- Work : construction sign work from [SVG Repo](<https://www.svgrepo.com/svg/307735/construction-sign-work>) - CC0
- Observation : observation from [SVG Repo](<https://www.svgrepo.com/svg/293710/observation>) - CC0
- Knowledge : idea from [SVG Repo](<https://www.svgrepo.com/svg/293713/idea>) - CC0
- Finance & administration : report from [SVG Repo](<https://www.svgrepo.com/svg/58321/report>) - CC0
- Studies : research from [SVG Repo](<https://www.svgrepo.com/svg/109479/research>)

- Description : Makina Corpus, derivated from [SVG Repo](<https://www.svgrepo.com/svg/258092/route-start>) - CC0
- Scissors from [SVG Repo](<https://www.svgrepo.com/svg/82088/opened-scissors>)

CHANGELOG

13.1 CHANGELOG

13.1.1 1.2.2 (2023-08-09)

Enhancement

- Remove api color for watershed

13.1.2 1.2.1 (2023-08-08)

Documentation

- Add documentation api swagger / doc

Enhancement

- Add contributions linked on details of knowledge / interventions / followups
- Filter api portal elements without accents and uppercase
- Add detail sentivities portal

13.1.3 1.2.0 (2023-08-04)

Documentation

- Add documentation portals
- Add documentation distance to source

Enhancement

- Add informations when hub'eau does not send a json
- Add migration generation distance to source
- Add contributions validated and publication date
- Add contributions type / category filters
- Add contributions manager
- Add contribution status
- Send mail to managers when contribution is created
- Send mail to contributor when contribution is created

- Add linked objects on contributions
- Add portal SEO informations
- Add min zoom, max zoom extent portal
- Add public portals on watershed types allowing to publish them

13.1.4 1.1.0 (2023-06-13)

Enhancement

- Add public portals on stream allowing to publish them
- Add PDFs administration of rivers
- Add flatpages module
- Add valorization POIs
- Add sensitivity module

Bug fixes

- Fix all point's marker was showing point to distance
- Fix form intervention, targets was not save

13.1.5 1.0.4 (2023-04-05)

Enhancement

- Add field classification water policy on rivers (#117)
- Add possibility to show geometries overprinted on topologies (#105)
- Add possibility to create attachment with external link
- Add command import hydrobiologie stations hubeau
- Upgrade api hubeau PC quality
- Add control type on Land module
- Add phases on Administrative Files
- Allow to create operations directly from creation of studies / follow ups / interventions / stations

Bug fixes

- Fix update attachments save buttons

13.1.6 1.0.3 (2022-12-15)

Enhancement

- Change secondary flow and habitat to multiselect field in description module
- Add fields to work : upstream and downstream bed impact, water impact
- Change vegetation strata field into a multiselect field

Bug fixes

- Display flow and source in stream detail
- Add flow filter in stream list
- Fix standalone intervention creation bug (#93)

Documentation

- Update doc installation
- Fix install doc with PostgreSQL not in Docker

13.1.7 1.0.2 (2022-05-22)

Bug fixes

- Fix ``./manage.py loadem ...`` command by including postgis libraries

13.1.8 1.0.1 (2022-03-30)

Enhancement

- Add data source and flow to stream

13.1.9 1.0.0 (2022-03-10)

Enhancement

- Add chosen multiselect on usage types
- Remove unused fields from Station form
- Get more data from Hubeau (start and end measure dates, measure type)
- Change base buffer width
- Change module picto colors
- Improve map color settings
- Display layers for all modules

Bug fixes

- Display missing unit
- Fix pip-tools / pip incompatibility

13.1.10 0.9.9 (2022-01-25)

Enhancement

- External link to station opened in new window
- Add unit on distance fields
- Remove secondary information from station detail
- Add chosen on some multiselect fields

Bug fixes

- Remove unwanted padding on lists
- Fix filter in service for stations
- Remove useless restricted area filter, replaced by zoning filter

Dependencies

- Update to django-mapentity 7.0.6 and Geotrek 2.75.0

13.1.11 0.9.8 (2022-01-20)

Features

- Display distance from object to stream source

Enhancement

- Improve morpho display

Bug fixes

- Fix translations

13.1.12 0.9.7 (2021-12-23)

Enhancement

- Change module order
- Add help text for multiselect

Bug fixes

- Fix logo header for PDF
- Fix man-days and costs display
- Fix translations

Dependencies

- Update to django-mapentity 7.0.5 and Geotrek 2.74.1

13.1.13 0.9.6 (2021-12-09)

- Use mapentity standalone release
- Improve documentation
- Add source location on a stream
- Make cut topology simpler
- Add help message on how edit man-days cost
- Fix filters on intervention and follow-ups

13.1.14 0.9.5 (2021-11-08)

- Improve documentation
- Improve README, maintainers and brand mark policy

13.1.15 0.9.4 (2021-11-05)

- First code publication

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`