



D5.1 - MANIFESTS DSS - Implementation reports

COPtool - COP Viewer

Web application – Model interface

Silvia Allen-Perkins, Garbiñe Ayensa, Pedro Montero, Samuël Orsi



ACKNOWLEDGEMENT

The work described in this report was supported by the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) of the European Union through the Grant Agreement number 101004912 - MANIFESTS – UCPM-2020-PP-AG, corresponding to the Call objective “Enhancing prevention and protection from the effects of maritime disasters” under priority 1: “Developing response capacity for marine pollution”.

DISCLAIMER

The content of this document represents the views of the author only and is his/her sole responsibility; it cannot be considered to reflect the views of the European Commission and/or the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) or any other body of the European Union. The European Commission and the DG-ECHO is not responsible for any use that may be made of the information it contains





MANIFESTS DSS

IMPLEMENTATION GUIDE

Pedro Montero, Silvia Allen-Perkins, Garbiñe Ayensa



Co-funded by the European Union Civil
Protection

ACKNOWLEDGEMENT

The work described in this report was supported by the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) of the European Union through the Grant Agreement number 101004912 - MANIFESTS – UCPM-2020-PP-AG, corresponding to the Call objective "Enhancing prevention and protection from the effects of maritime disasters" under priority 1: "Developing response capacity for marine pollution".

DISCLAIMER

The content of this document represents the views of the author only and is his/her sole responsibility; it cannot be considered to reflect the views of the European Commission and/or the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) or any other body of the European Union. The European Commission and the DG-ECHO is not responsible for any use that may be made of the information it contains.



MANIFESTS

MANaging risks and Impacts **FROM** **And**vaporating
and gaseous **Substances** **To** population **Safety**



Co-funded by the European Union Civil
Protection

Project Acronym	MANIFESTS
Project Full Title	MANaging risks and Impacts From Evaporating and gaseous Substances To population Safety
Gant Agreement Nr.	101004912
Project Website	https://www.manifests-project.eu/

Deliverable Nr.	D5.1
Status (Final/Draft/Revised)	Final
Work Package	5
Task Number	5.1
Responsible Institute	INTECMAR
Author/s	Pedro Montero, Silvia Allen-Perkins, Garbiñe Ayensa
Recommended Citation	
Dissemination Level	

Document History			
Version	Date	Modification Introduced	
		Modification Reason	Modified by
1.0	2023/02/01		

Content

1. Background.....	6
2. Objective	7
3. Structure Of The Decision Support System	7
3.1. Schema of a DSS for marine pollution contingencies	8
3.2. Use of DSS prior to any contingency	13
3.3. Use of DSS during contingency.....	15
3.4. Elements of DSS: Sources of information.....	17
3.5. DSS Elements: Users.....	20
4. System Implementation	21
4.1. COPtool.....	23
4.2. COP Viewer	23
4.3. Auxiliary database and associated scripts.....	24
4.4. Relationships between the different components	25
5. DATABASE AND INFORMATION MANAGER	25
5.1. Database Server	25
5.2. Databases	25
5.2.1. COOPTOOL. Management of users, layers and COPs	26
5.2.2. COOPTOOL. Insertion tools	31
5.2.3. COOPTOOL. General outline.....	36
5.2.4. COOPTOOL. Auxiliary databases	36
6. Resources	39
7. Knowledgments.....	40



1. Background

In the event of a maritime accident involving Harmful and Noxious Substances (HNS), the maritime authorities must make numerous decisions to organize the best response strategy, that is, the one that minimizes the risks to human health (including rescuers, crew members and coastal communities), for the marine environment, for maritime safety and for socio-economic activities and facilities. While many key decisions and considerations are prescribed in national or regional contingency plans, operational response activities will generally need continuous adjustment or revision to reflect the latest information available as the contamination event evolves. In such a rapidly changing situation, an efficient two-way information exchange between decision-making authorities and response teams on the ground can greatly facilitate both, decision-making and organizational processes.

The purpose of work package 5 of the Manifests project (Manifests decision support system) is to develop an efficient information system that helps (1) decision makers understand the situation at stake and its likely evolution in the coming hours and days; (2) identify the population, ecosystems and socio-economic assets at risk and (3) share useful information with response teams deployed at sea, in the air or on the coast.

Building on the experience gained and development carried out during the previous HNS-MS and MARINER projects, the MANIFESTS decision support system (DSS) will integrate several services, including the DSS Common Operational Picture (COPTool).

This COPTool refers to a system designed so that during a contingency, the exchange of information that occurs between the maritime authorities and the different response teams (sea, coast, air) is carried out in the most efficient way possible, ensuring that all actors involved in the crisis committee and response teams can access the same data. These can be standard reports (such as the Standard Pollution Observation Report of the Bonn Agreement), images, videos and any other georeferenced data collected by response teams, as well as satellite observations, model simulation results, exclusion areas, location of response media, requests for new response actions shared by the crisis.

The COPTool documentation is composed by:

- Installation guides.



- User Manual.
- DSS Implementation report (this document).

2. Objective

The purpose of this document is to explain the structuring of a contingency in reference to the information handled during each contingency and the users of this information.

In the first part of the document, the different sources of information that are usually present during an episode of marine pollution will be described, considering both their temporal and spatial specificities. In addition, the layers of data and reports that are generated during the accident in relation to the fight against pollution and the different participating users and their functionalities will also be explained.

In the second part, the technological solution for the insertion and distribution of information for the different actors in the contingency will be described to later specify in detail each type of information layer to be distributed.

3. Structure Of The Decision Support System

This section describes the different elements of the response decision system. A Decision Support System or DSS (Decision Support System) is a computer system that helps the decision-making process. In rather more specific terms, a DSS is "an interactive, flexible and adaptable computer-based information system specially developed to support unstructured management problem solving to improve decision making. It uses data, provides a friendly interface, and allows decision-making in the analysis of the situation" (Turban, E. 1995).¹

The proposed DSS is based on the following elements:

1. A resource center that centralizes and gives access to management protocols, guidelines, reports and other sources of knowledge, such as sensitivity maps useful for assessing risks and planning response actions. It will consist of a web tool for the management of this information

¹ Turban, E. (1995). Decision support and expert systems: management support systems. Englewood Cliffs, N.J., Prentice Hall. [ISBN 0-02-421702-6](#)

and the management of data insertion and simulations, contingencies and users. It will be called **COptool**.

2. An application accessible to all response teams, coast guard officers and maritime authorities in order to efficiently exchange information on the latest developments of the pollution event and response actions. This application will be called from now on, **COP Viewer**.
3. A series of databases and associated procedures to include those internal sources of each organization in the resource center and be managed by the COptool. This set of tools will be called **Auxiliary COP**.

Before explaining each of these elements, one must understand the whole schema of a DSS for marine pollution contingencies.

3.1. Schema of a DSS for marine pollution contingencies

When making decisions for the response to a marine accident, the availability of up-to-date information is essential. This information is usually georeferenced, that means one property of it is a location in the space. This kind of information is often called information layers, since they become a set of superimposed maps. Some of these layers have been created before the contingency, such as the coastline, bathymetry, or areas of environmental importance, and some of them are generated during the contingency, for example, the prediction of drifters, the positions of the spills, or the situation of the anti-pollution equipment in use.

On the other hand, for using this information effectively during a contingency, this information must be distributed quickly, efficiently, and safely to the different actors involved in the crisis, such as the contingency management, political advisers, or staff involved. The needed information for each of them is usually different. In addition, the degree of confidentiality of the information is not always the same and therefore there will be information that will not be available to all users.

Layers of information

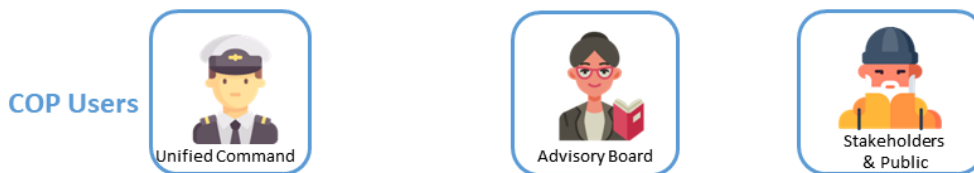


Fig. 1: Elements in a contingency: On the one hand, there are layers of information (prior to the contingency or generated during it). On the other, there are various actors involved in the contingency with different degrees of access to this information.

The challenge of this work is to create a system where a wide variety of information, mostly georeferenced, is available quickly and efficiently by the different actors involved, without overloading each of them with information and, at the same time, guaranteeing grades of confidentiality when necessary.

The DSS is able to easily create a Common Operational Picture (COP) for each contingency customized for each user. This means that, on the one hand, there will be access to the precise information of each contingency and on the other, each user depending on their level, will be able to access different

information.

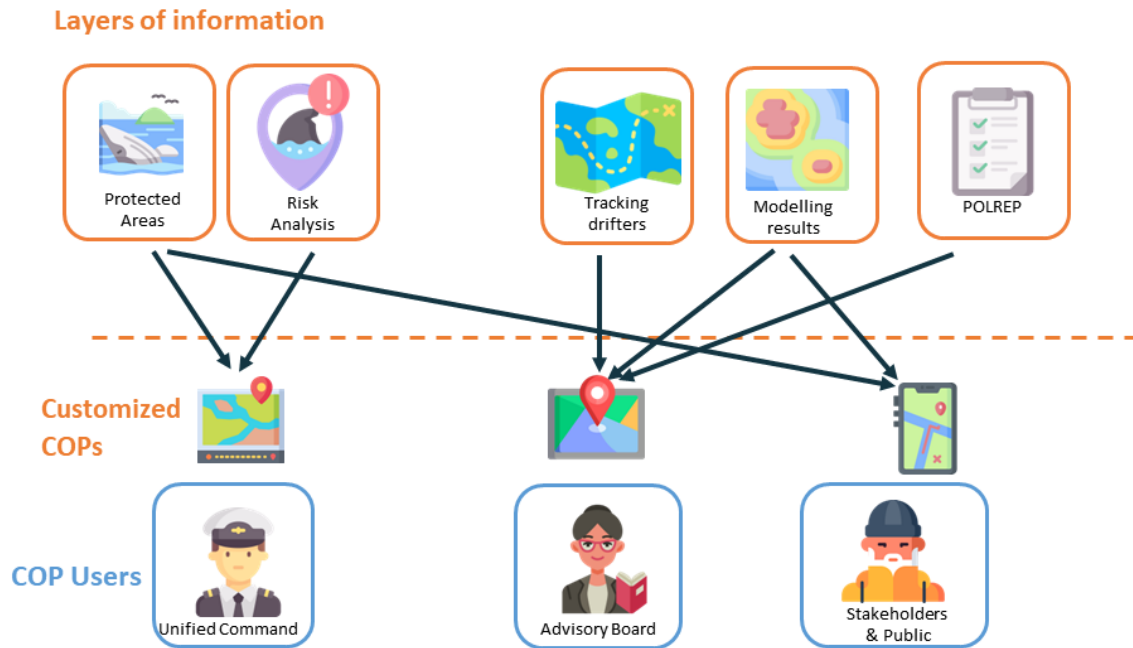


Fig. 2: Users will consult the information layers through a COP (Common Operational Picture) tailored to the contingency and their needs.

In addition, much of the information is generated manually during the contingency: POLREP reports, other reports, pictures, etc. Therefore, the DSS must not only collect information prepared before the accident but also equip itself with systems that allow it to ingest data and information generated during the contingency. Therefore, beyond the users with access to information, there will be users with the ability to insert information.

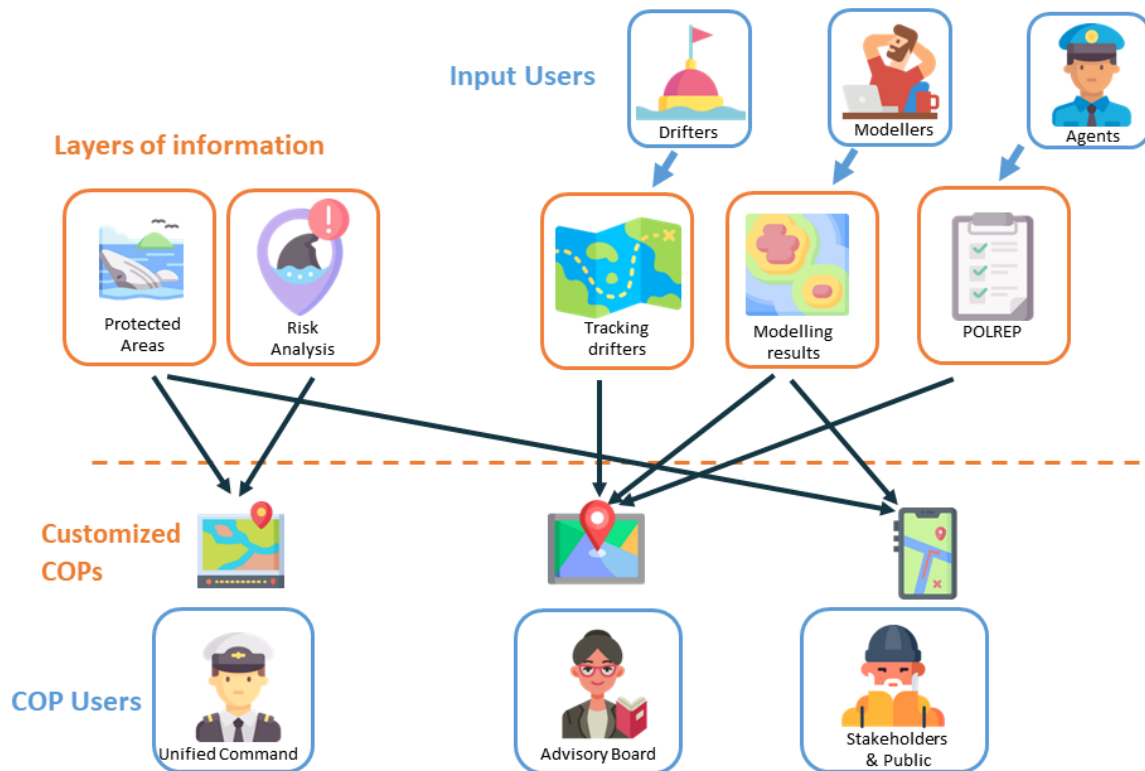


Fig. 3: In addition to the above elements, information layers and COPs users, there are other users of the system that are the information insertors, either manually generated as POLREP reports or in a more automatic way such as model results or drift buoy positions.

Above these COP users and input users, a **COPs Manager** is needed to manage the different users who can use the system during the contingency, as well as what information is accessible during it. The COPs Manager is the user who creates a COP for a contingency, associates users with that contingency, and attributes the necessary layers to it. Each user will be able to consult the layers associated with that contingency depending on the access permissions they have at the time of the contingency. Users with the lowest level of confidentiality will not be able to view and query layers that are restricted to the higher ones.

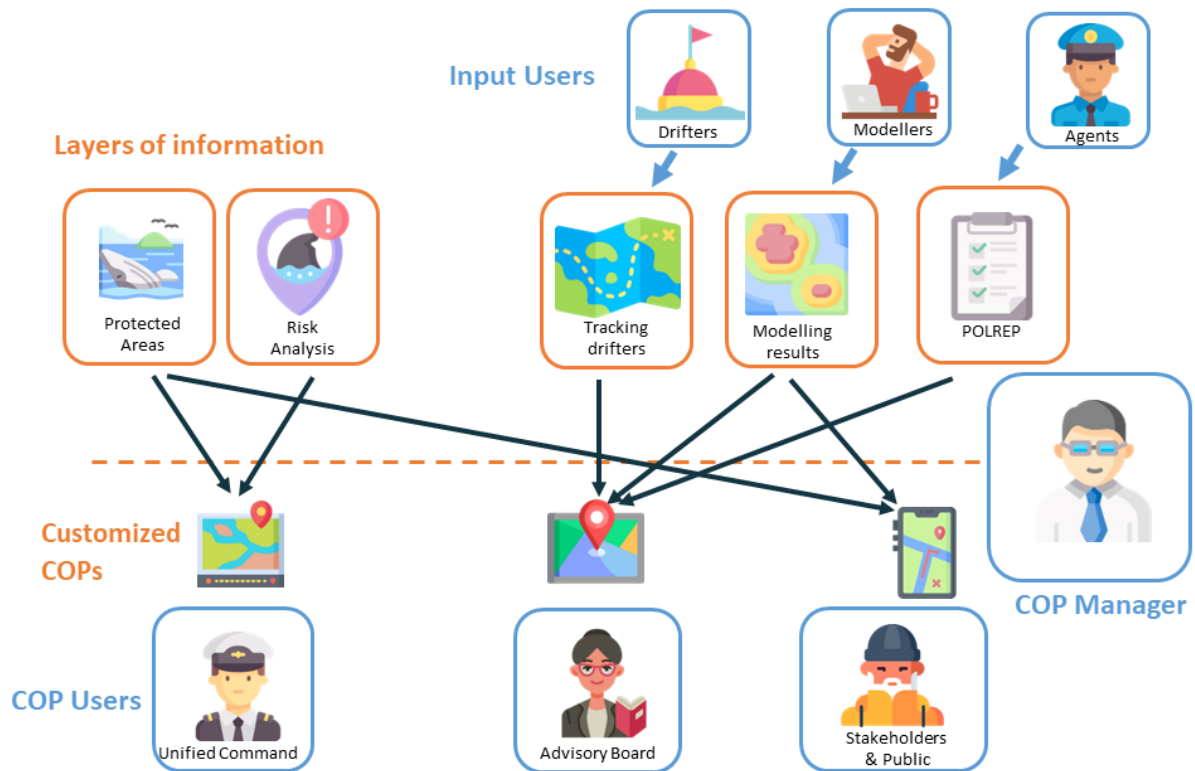


Fig. 4: There is the COPs Manager user whose main task is to create the different COPs customized for each user when the contingency occurs.

Like any computer system, there's a super user or Administrator, named COPTool Manager, who manages the tool and has two main functions: giving access to the tool to the rest of the users and establishing the addresses and metadata of the information that will be available for access in future contingencies.

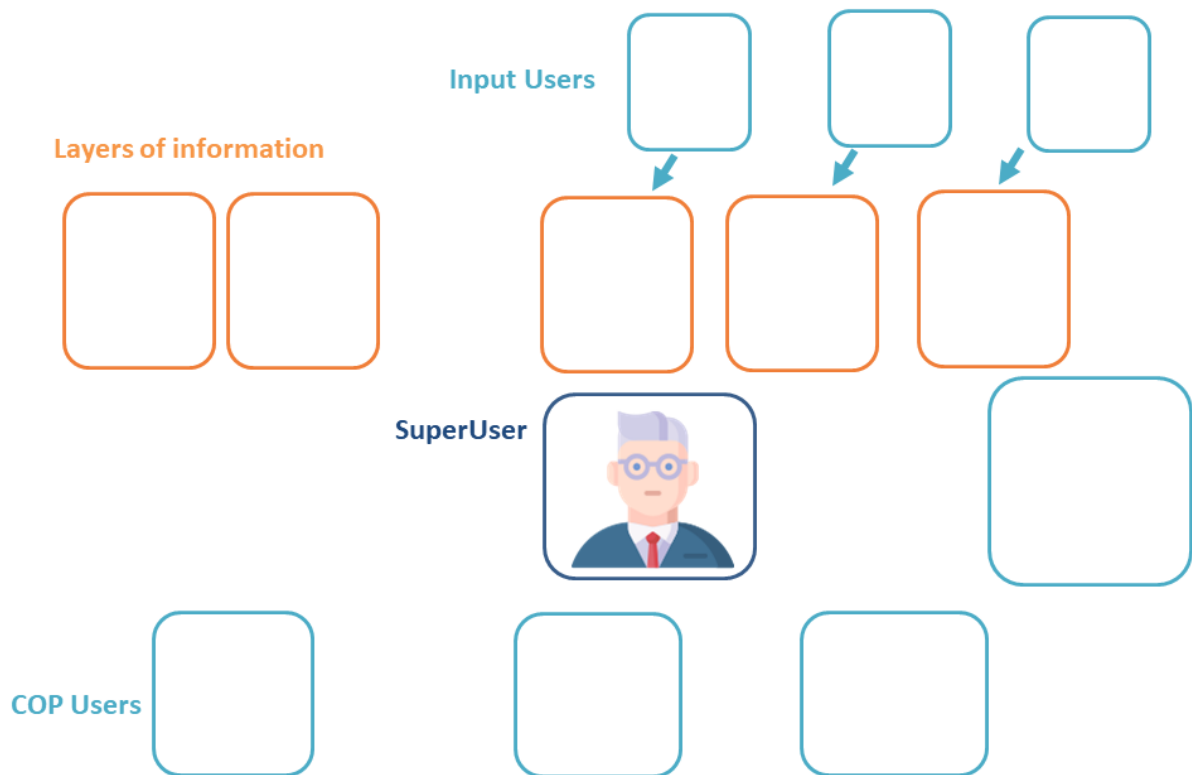


Fig. 5: In addition to these users, there is the Administrator user who inserts all the layers that will be available in the system and manages the access of the rest of the users.

It is important to distinguish between the COptool Manager and the COPs Manager for the different contingencies. The first one gives access to all users (including the COPs manager) and includes in the system the references for the different layers of information contained in the data base, while the second determines which users have access to the specific information for a contingency.

3.2. Use of DSS prior to any contingency

A normal use of DSS prior to any contingency would be as follows:

1. The COptool Manager **inserts layers** of information into the system. This means that there will be a reference to a WMS (Web Map Service) address for each layer and therefore that the information of this layer can be included in a future contingency.

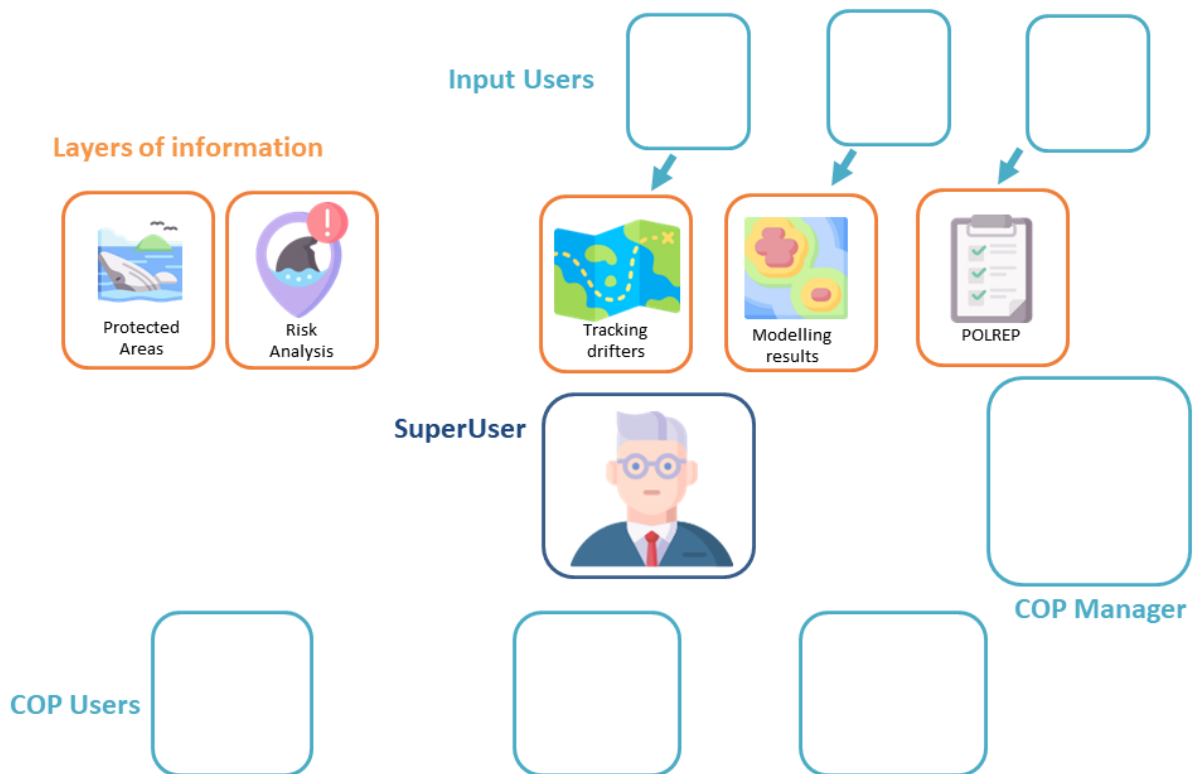


Fig. 6: The Administrator inserts the information layers into the referenced system with a WMS address.

2. The COptool Manager **enables access** to other users: they can have different functionalities. The main users are:
 - a) **COPs Manager:** Allows you to create a COP for a contingency, choose the layers of information that can be seen in that contingency and which users can see them.
 - b) **COP Viewer users:** They are users to consult the COPs of the contingencies to which they have access. Keep in mind that not all users have the same level of query, regardless of whether they have access to a contingency or not.
 - c) **Inserting users:** These are users who have the ability to insert information and reports. They can be users insertors of POLREP, SCAT, communications, maps, etc.

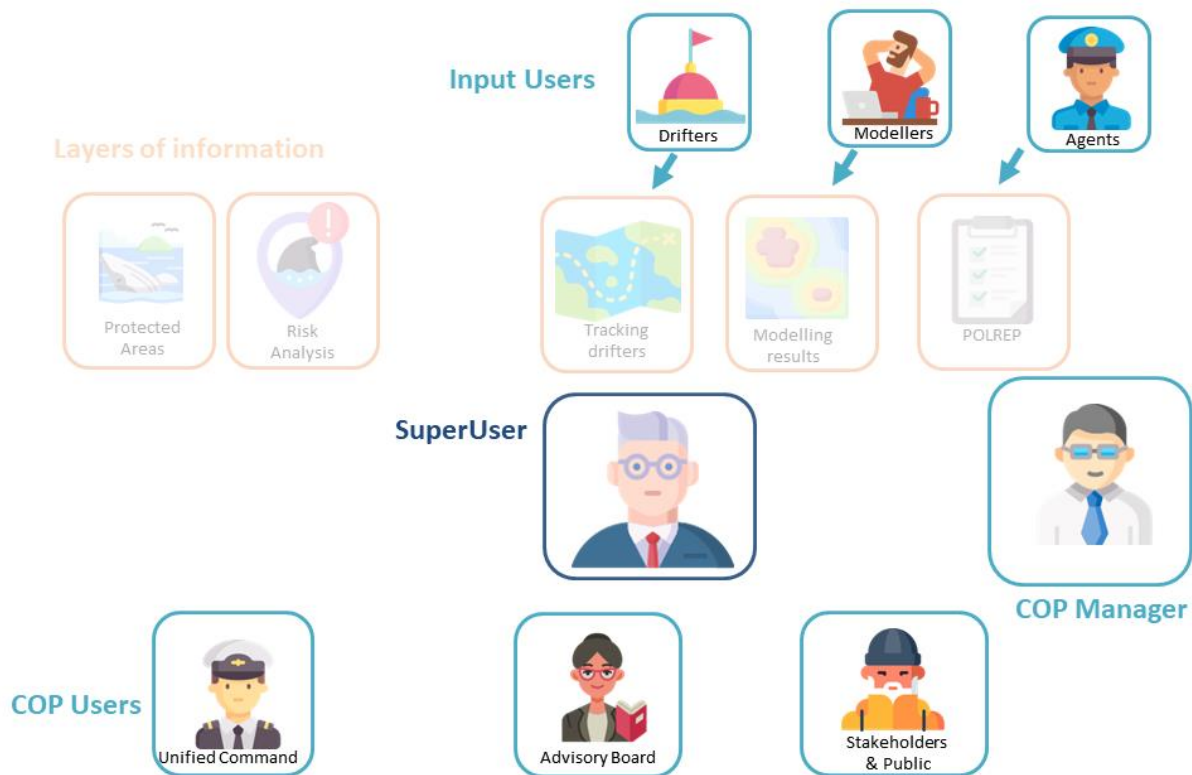


Fig. 7: The Administrator gives access to other users: COPs Manager, COPs Viewer users and Inserting Users.

3.3. Use of DSS during contingency

A normal use of DSS during a contingency would be as follows:

1. The COPs Manager **creates a new COP** for that contingency, including the different layers of information that can be consulted and the users who have access. He or she can also link reports, photos, and maps to that contingency.
2. **COP Viewer Users can access the COP** of that contingency and consult the information to which they have access.
3. **Inserting Users can insert** POLREP, SCAT, reports, pictures, or documents. If these reports are linked to the contingency, it may be consulted as information about it.
4. **Automatic or external processes** will insert information into the data base. The usual information that is included in these external processes are simulations, drift buoys, etc.

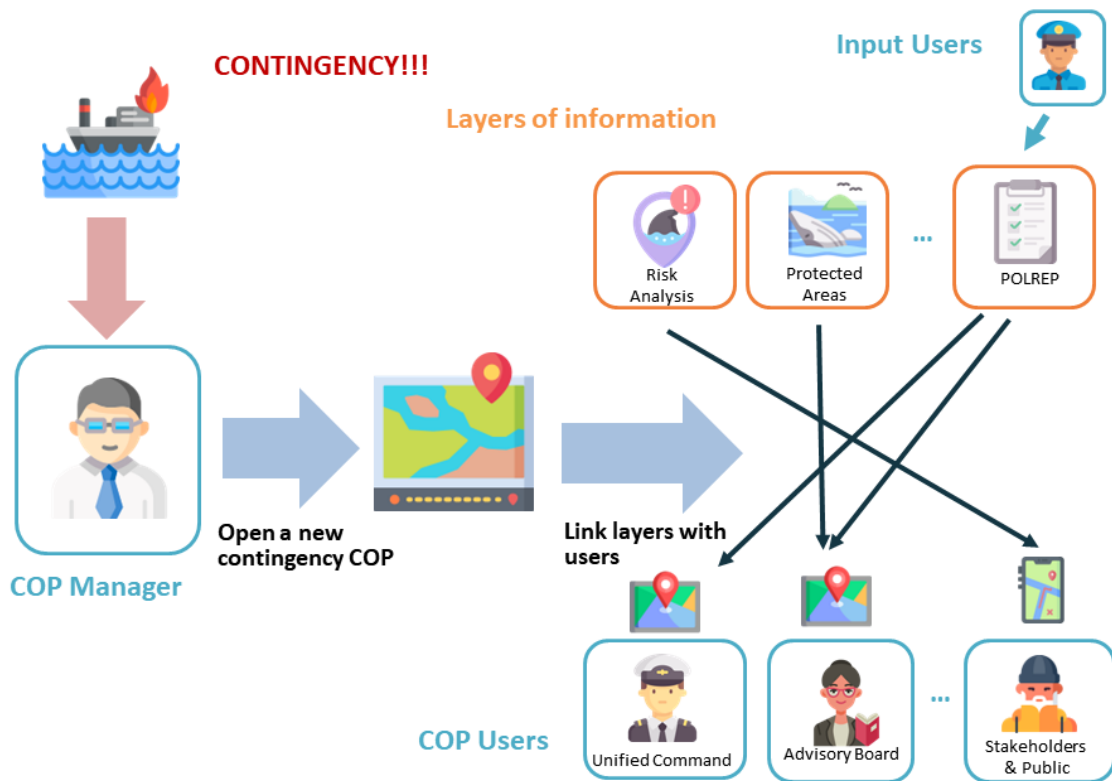
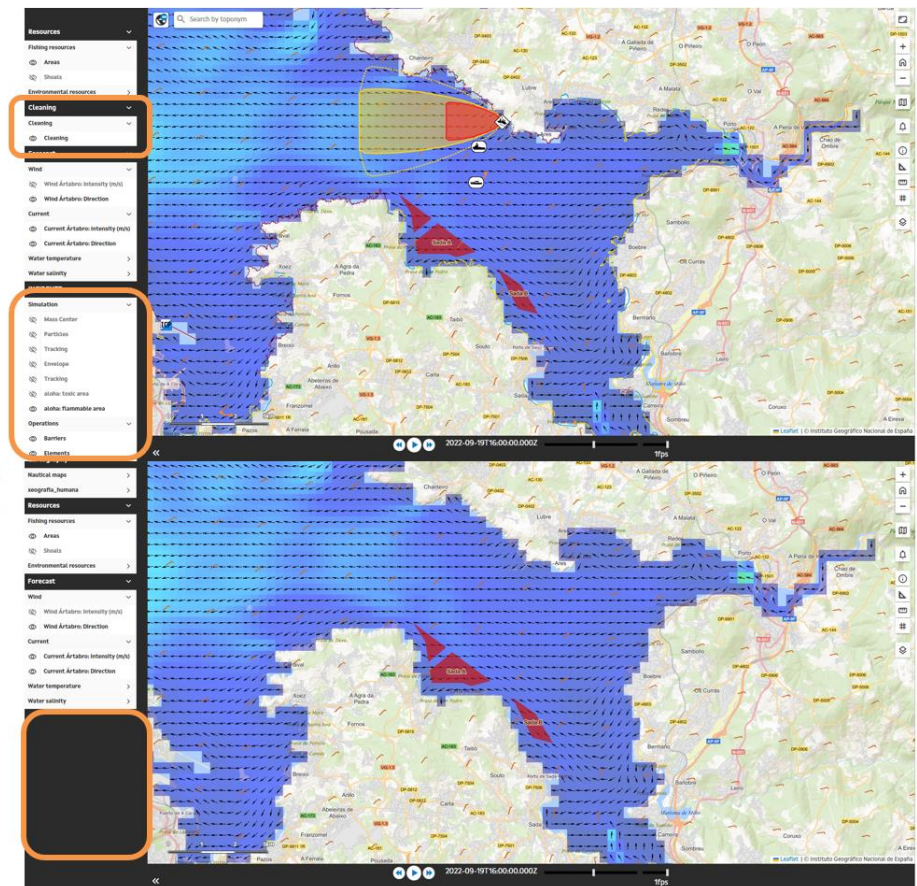


Fig. 8: During a contingency the COPs Manager opens a new contingency in the DSS, and assigns layers of information and users to that new contingency.

The result is that a COP Viewer User can access a COP of the contingency that allows him or her to visualize the layers of that contingency visualizing only the layers he has access to. The COP Manager carries out the customization of the COP to each contingency and each COP Viewer User in a relatively short time, no more than a few minutes, can visualize it.

High degree of confidentiality user



Low degree of confidentiality user

Fig. 9: COP viewer of the same contingency for two users with different degrees of confidentiality. The user with low confidentiality (bottom) is not allowed access to all layers of a user with a higher degree of confidentiality (above), as shown by the orange rectangles.

In addition, these contingencies receive different inputs from the different Inserting Users, which are automatically updated for consultation by COP Viewer Users.

3.4. Elements of DSS: Sources of information

This section will describe the different types of information layers that are commonly used when making decisions during a contingency. It should be noted that these layers depend on the specific information needs and availability of each organization. The following classification is a general guide and not a rigid scheme.

In general, the layers of information needed for decision-making during a contingency can be classified into:

- 1) **Pre-contingency information:** This includes all layers of information that exist regardless of whether there is a contingency or not. It also includes dynamic information that exists even if there isn't a contingency, like meteorological or hydrodynamic predictions.

It can be divided into:

- *Static information*: includes information on base layers, cartography, fishery and environmental resources and a coastal inventory.
- *Dynamic information*: This information varies over time and exists whether there is contingency or not. This includes information from meteorological, hydrodynamic, and wave models, satellite observations, HF radar, or automatic platforms, and layers with AIS or ship location information.

2) **Information created during the contingency**: This consists of the information of the contingency.

Here it is contemplated:

- Results of prediction models of spill drifts, HNS, toxic clouds, and exclusion areas.
- Drifter tracks and spill monitoring systems
- Location of assets and booms, tactical maps, etc.
- POLREP, SCAT, pictures, documents and reports.

Typical sources during a contingency can be outlined in the following figure:

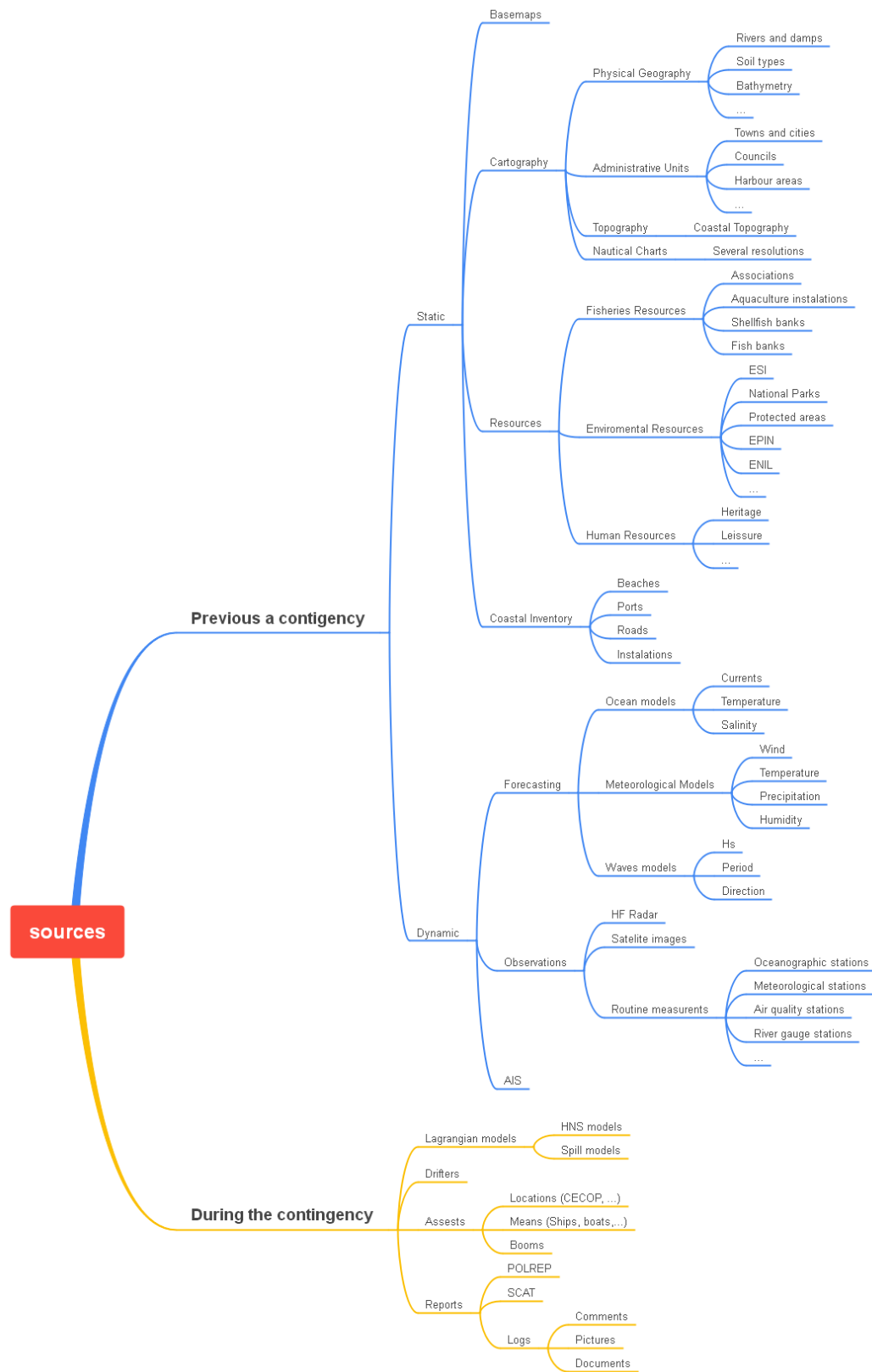


Fig. 10: Kind of information managed during a contingency: there are two large groups: the information created prior to the contingency and the other one generated during and for the contingency.

References to these sources of information are made through the Open Geospatial Consortium (OGC (<https://www.ogc.org>) service, called Web Map Service (WMS). This service basically consists of a web service that enables the access a thematic map, with certain functionalities, such as obtaining information about the items represented. Thus, each layer included in the COPtool will be associated with a WMS address that will provide the information in the form of a searchable geographical layer. It is the function of the COPtool to store these references.

Therefore, the connection between the COP viewer and the information layers is made through this type of services. This allows us to include in the viewer any type of layer that is served through WMS, whether from servers external to the system or own servers.

For the information stored in the organization itself, whether from the DSS-COPtool database or others, it is proposed to provide this WMS service through a Geoserver (<https://geoserver.org>).

A more detailed explanation of this implementation will be given later.

3.5. DSS Elements: Users

As it has been referenced before, from a conceptual point of view there are several types of users: Administrator, COPs Manager, COP Viewer Users, Inserting Users, etc.

However, when implementing the system it has been chosen to create only two users: Administrator and User, which can be assigned one or more functionalities or modules:

- **Administrator:** Super user, with access to all modules, as well as the management of users and information layers of the system.
- **User:** With access to those modules (one or several) assigned by the administrator.

The different modules that allow users to perform each of the functions are:

Module	Functions
User Management	Creation/editing of user accounts.
Layer Management	Managing the information layers that are available in the system.
Management of COP (Common Operational Picture)	Management of the information to be distributed during the contingency.

POLREP	Insertion of POLREP reports into the system.
SCAT	Insertion of SCAT reports in the system.
Reports	Insertion of information derived from different inspections (photos, videos, comments, etc.)
COP Viewer	Access to the COP viewers to consult the COP of the contingency

The user management and layer management modules are restricted to the administrator user, and only the administrator user can make changes. The rest of the users will be able to access one or several modules depending on the permissions granted by the administrator at the time of registering in the system.

For example, a COP Manager at the conceptual point, is a User with the COP Management functionality from the point of implementation. This change allows the existence of users with different functionalities, providing greater plasticity to the DSS.

Additionally, users will also have different levels of access to the information layers, with level 1 having the lowest access to information and level 3 having the highest access. These generic access levels can be modified for each user and contingency if necessary.

4. System Implementation

The system can be divided into three components:

- **COPtool:** Software consisting of a graphical interface and an associated database for user management, and its modules.
- **COP Viewer:** Software for exploring the maps of a contingency managed by the COPtool.

- **Auxiliary COP:** Database with other relevant information such as model results, drifter tracks, etc. and associated inserting scripts. This component depends largely on the availability of this information by the agency so its implementation is not always similar and necessary.

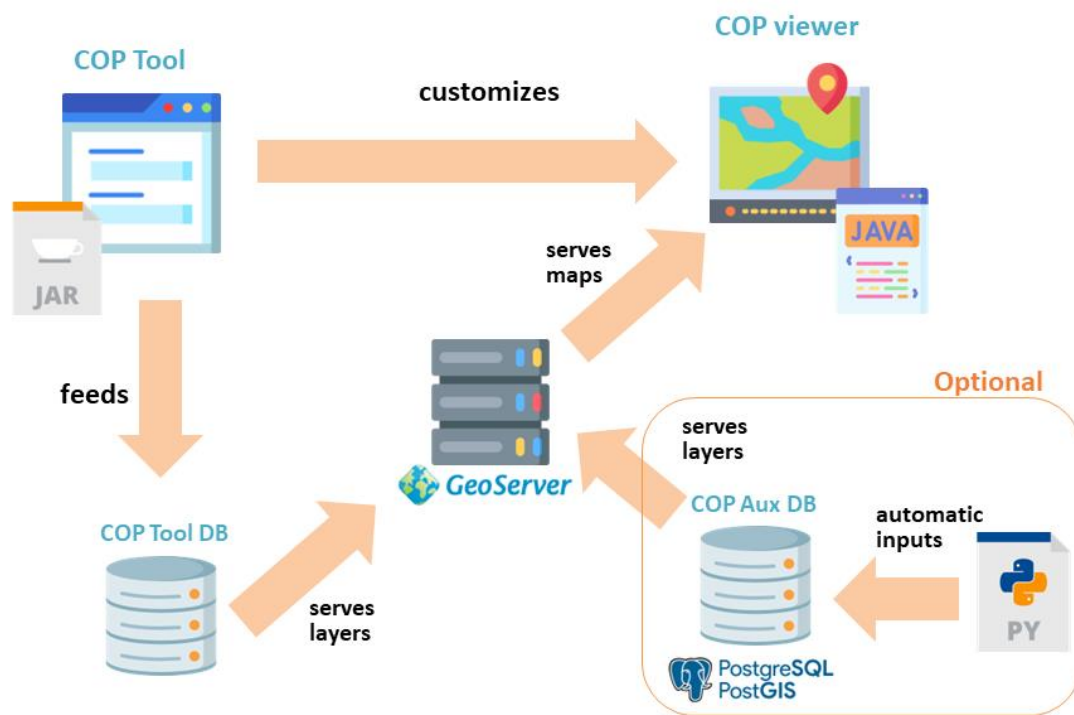


Fig. 11: Implementation of the complete system with its relationships.

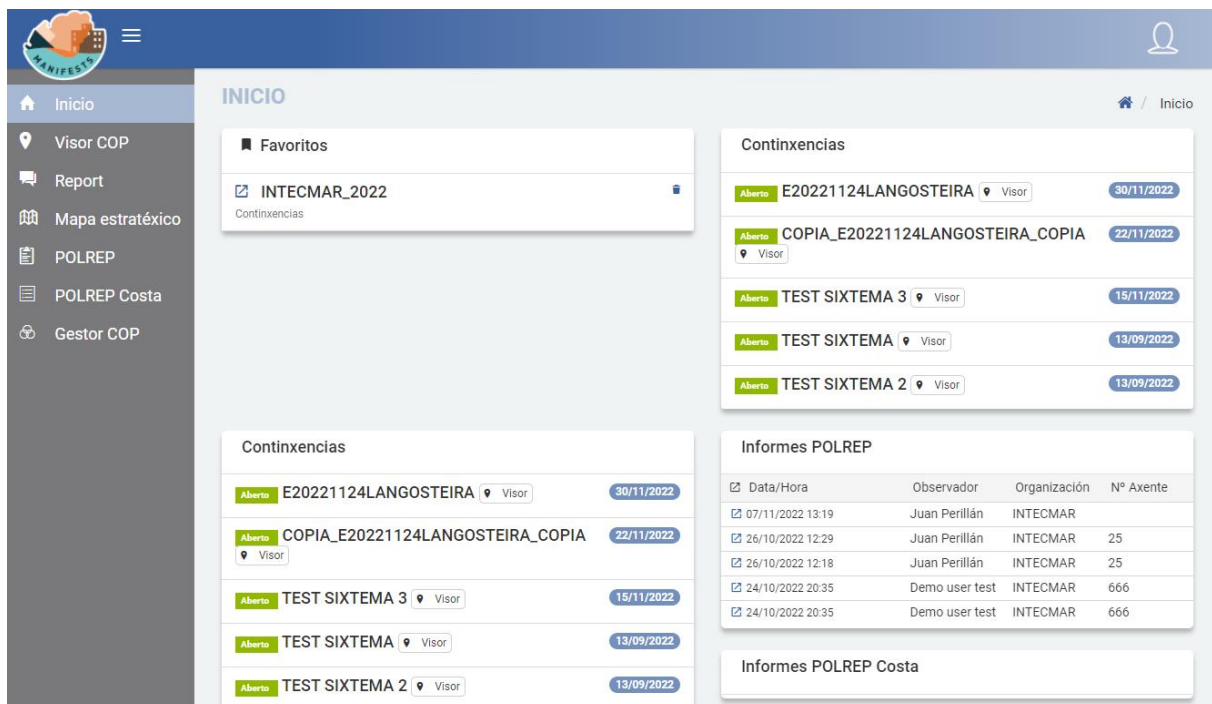
COptool relations with COP Viewer are:

- The COptool customizes the information available for consultation through the COP Viewer and it depends on the contingency and the validated user. The COP Viewer receives a configuration file with certain specifications each time it is accessed through COptool. The specifications are addresses of WMS map services that will be displayed in the COP Viewer.
- WMS map services can be external to the organization or internal. In this case, they can come from the COptool database or from the Auxiliary COP database and both need a WMS map server implemented. We recommend Geoserver software as WMS server.

4.1. COptool

The purpose of COptool is the relationship between the COPTOOL database and the users, whether they are administrators and/or users, this relationship is carried out through an interface. Once they have been validated, the users can perform their previously registered functions, be it administration, consultation, or information insertion. This interface has various modules, which are visible depending on the level of user access to them. It also manages the different COPs for each contingency, the information inserted in each contingency and the query to the COP Viewer once the user has been validated.

The users can access this interface through a web browser, and after validation they are able to perform the tasks they have access to.



The screenshot shows the COptool web interface. On the left is a navigation menu with options: Inicio, Visor COP, Report, Mapa estratégico, POLREP, POLREP Costa, and Gestor COP. The main content area is titled 'INICIO' and contains several sections:

- Favoritos:** A card for 'INTECMAR_2022' with a 'Contingencias' label.
- Contingencias:** A list of active contingencies:
 - E20221124LANGOSTEIRA (30/11/2022)
 - COPIA_E20221124LANGOSTEIRA_COPIA (22/11/2022)
 - TEST SIXTEMA 3 (15/11/2022)
 - TEST SIXTEMA (13/09/2022)
 - TEST SIXTEMA 2 (13/09/2022)
- Contingencias (Detailed View):** A similar list of contingencies.
- Informes POLREP:** A table with columns: Data/Hora, Observador, Organización, and N° Axente.

Data/Hora	Observador	Organización	N° Axente
07/11/2022 13:19	Juan Perillán	INTECMAR	
26/10/2022 12:29	Juan Perillán	INTECMAR	25
26/10/2022 12:18	Juan Perillán	INTECMAR	25
24/10/2022 20:35	Demo user test	INTECMAR	666
24/10/2022 20:35	Demo user test	INTECMAR	666
- Informes POLREP Costa:** A section for regional reports.

Fig. 12: COptool initial screenshot after user validation.

From the implementation point of view, the COptool is based on microservices programmed in Java (WEB and API). Along with these microservices, the COptool is associated with a PostGIS database that we will call COPTOOL-DB and that will be described in chapter 5.

4.2. COP Viewer

If the user wants to view one of the COPs they have access to, the COptool redirects them to another web page with the COP of the contingency and the information that specific user is allowed to view.

To do this, the COPTool creates an XML configuration file that tells the COP Viewer the specifications of that COP: visible layers, access to reports, permissions, etc.

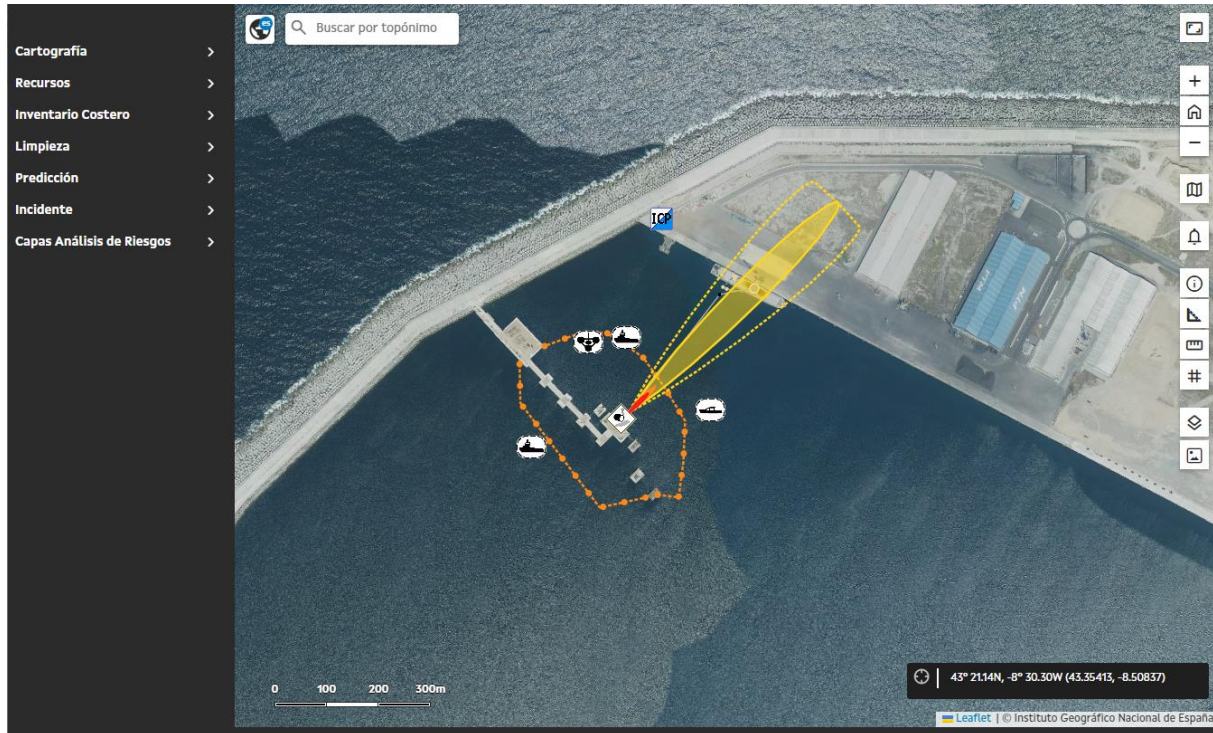


Fig. 13: COP Viewer showing different information of a contingency: dispersion of a simulated cloud, boom locations, means positions, etc.

The code of the COP Viewer has been coded in JavaScript and PHP.

4.3. Auxiliary database and associated scripts

In addition to this database that manages the COPTool, there can be information that, being included in the COP, is sufficiently independent of the tool and at the same time sufficiently dependent on the specific agency to belong to a different database. We will call this database COPTOOL_AUX, which will be described later.

Data to be included in this auxiliary database may be the positions of the drifters, the results of a Lagrangian model or the exclusion zones of an air pollution model such as ALOHA.

Associated with this database, different scripts have been created for the insertion of the information, which have been written using Python. However, as indicated at the beginning, this database and its scripts are strongly dependent on the agency, so their implementation.

4.4. Relationships between the different components

The relationship between database information and the COP Viewer is established through web services, mainly the Web Map Service (WMS) of the Open Geospatial Consortium (OGC).

Thus, when a user accesses a contingency COP through the COptool, it creates an COP configuration XML file, which is interpreted by the COP Viewer. Among other things, this configuration file defines the different layers of information that this user can consult. These information layers are maps served by web services, mainly WMS.

Therefore, the XML file mainly relates a user and a contingency with the different WMS layers. As mentioned before, there is an Administrator who has the ability to include layers and classify them into groups and panels. Before a layer can be used by any specific COP, this layer must be included in the system, storing these references to its WMS address.

This address can be internal, that is, it will use the WMS service established in the agency itself, or external, that means, it belongs to WMS servers that are outside the databases managed by the COptool and its ecosystem. For the service, a GeoServer server is used that provides the different layers of both the COPTOOL database and the COPTOOL_AUX.

5. DATABASE AND INFORMATION MANAGER

This part will describe the implementation of the databases and what information it contains.

5.1. Database Server

The DSS uses as a database engine the PostgreSQL (<https://www.postgresql.org/>) software in its version 9.5.1 to which the PostGIS extension (<https://postgis.net/>) has been added to create databases with the capacity to store geographic objects.

5.2. Databases

COPTool uses a main database called COPTOOL. The two main functions of this database are:

1. Manage users, layers, and COPS
2. Store those data that are entered manually, that is:
 1. POLREP Report
 2. SCAT

3. Reports, documents and photos of a contingency.

In addition to this COPTOOL database, there will be the necessary databases to store all the necessary internal information, such as model outputs, derivative buoys, etc. In the example case, this database is called COPTOOL_AUX and each entity is stored in a schema.

5.2.1. COPTOOL. Management of users, layers and COPs

The COPTOOL database allows you to manage the users of the system, defining the responsibilities/functions of each of them. These functions will be translated as access to modules on the platform.

The following diagram shows a simplified version of the tables related to the User table, the logic of which has been moved to the database.

ER diagram for users

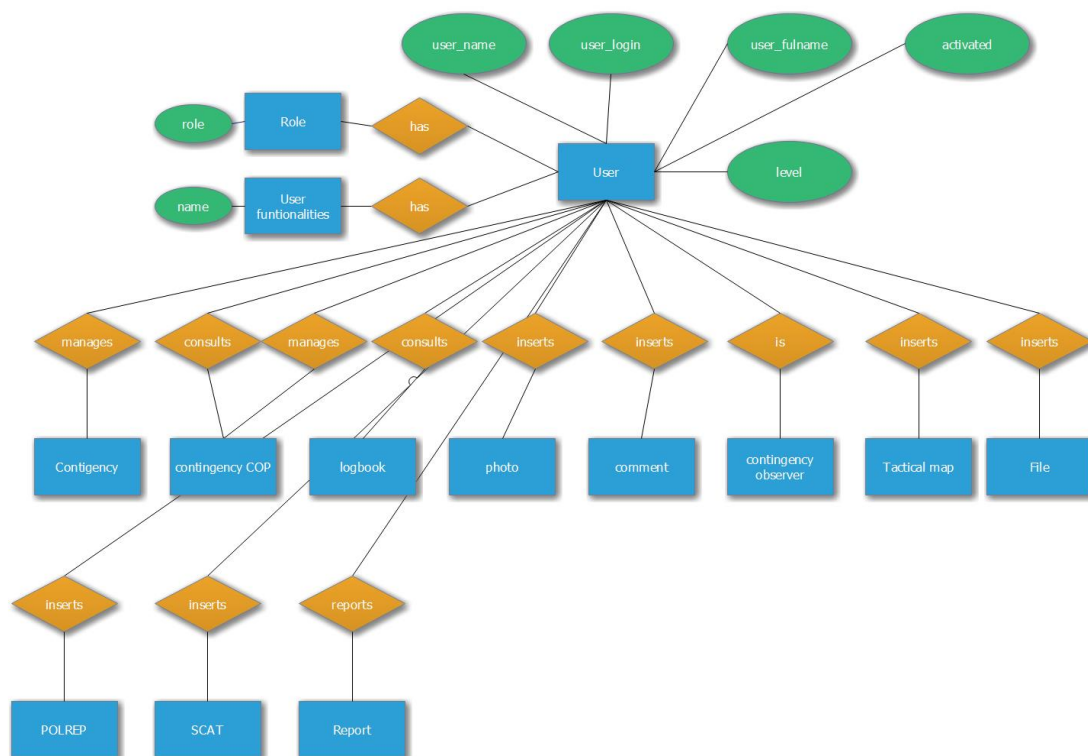


Fig. 14: Entity-Relationship diagram for Users.

This Entity-Relationship diagram is transposed to the database in the following schema:

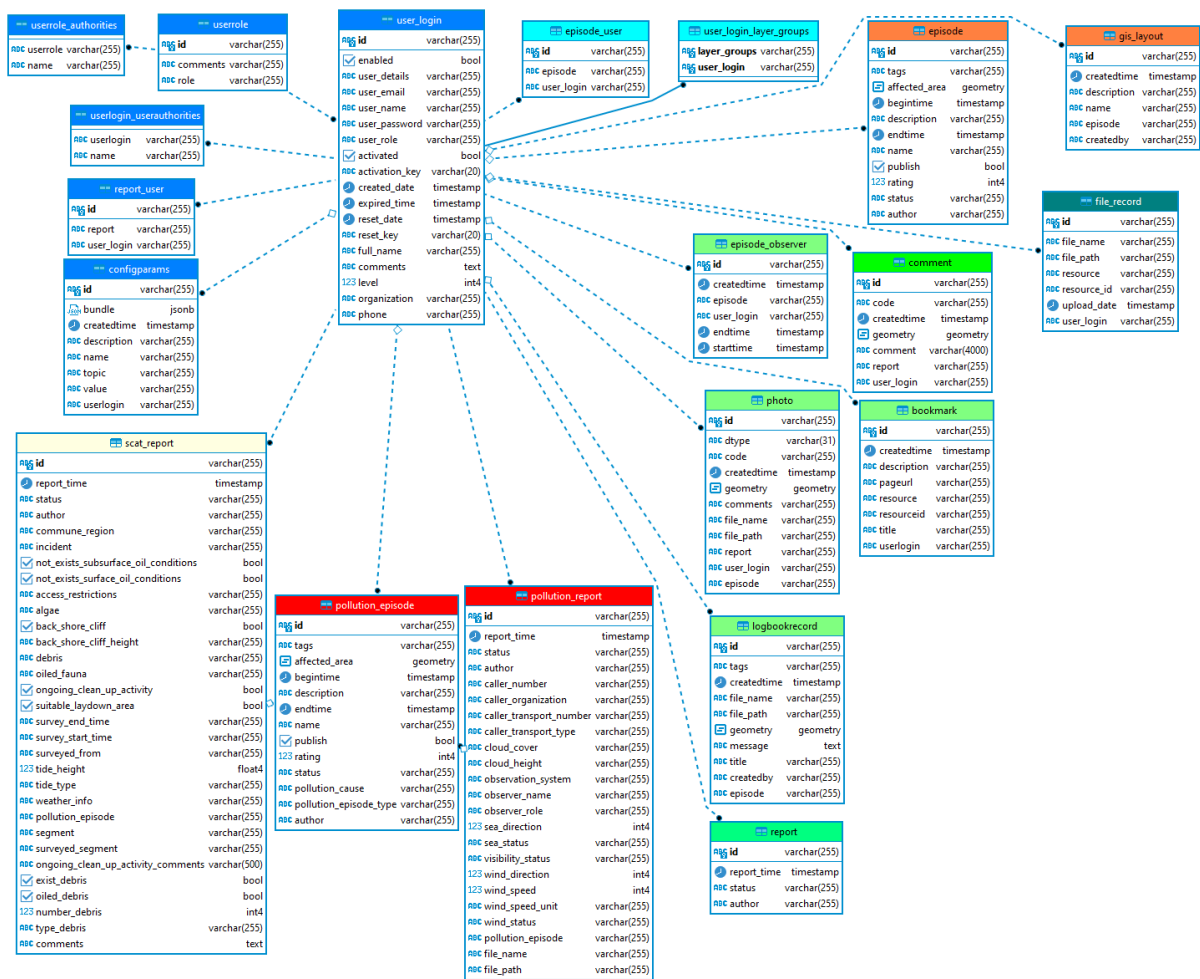


Fig. 15: Schema of User relationships in the database.

In addition to the users, the COPTool must manage the layers of information that will be visible by the different users.

When defining that the system contemplates a layer, what is done is to collect which WMS service or another must be consulted so that this layer is displayed. Therefore, a layer is defined by the name of the layer, the WMS service (or other) and a series of parameters as projection, and the coordinates of the bounding box.

To do this, keep in mind that each layer belongs to a layer tree that is structured in Panel, Group, Layer. In addition, each user can access certain episodes of which only the layers allowed for that user will be displayed. In addition, layers can be filtered by some field.

The following figure shows a simplified Relationship Entity schema.

ER diagram for layers

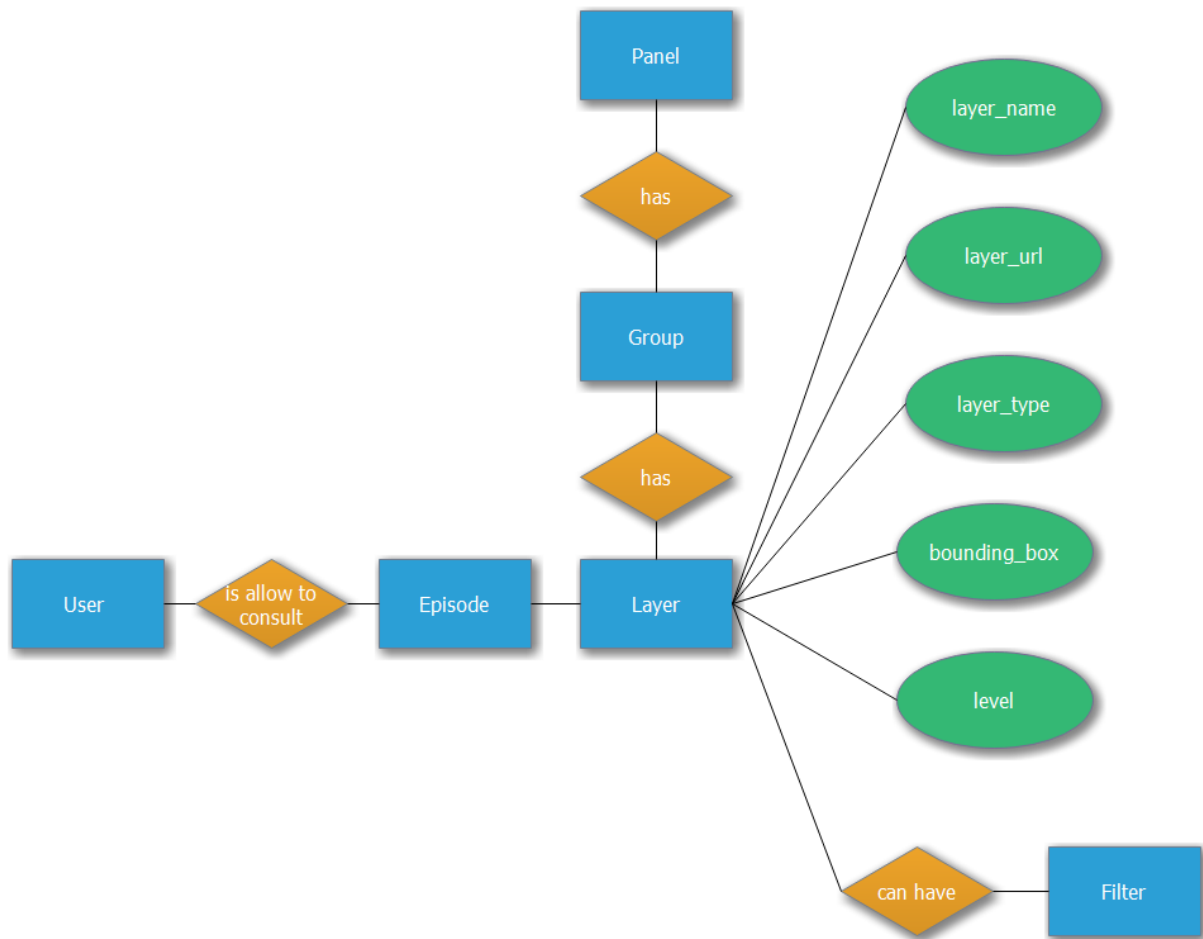


Fig. 16: Entity-Relationship Diagram for Layers16

This schema has been moved to the database as follows:

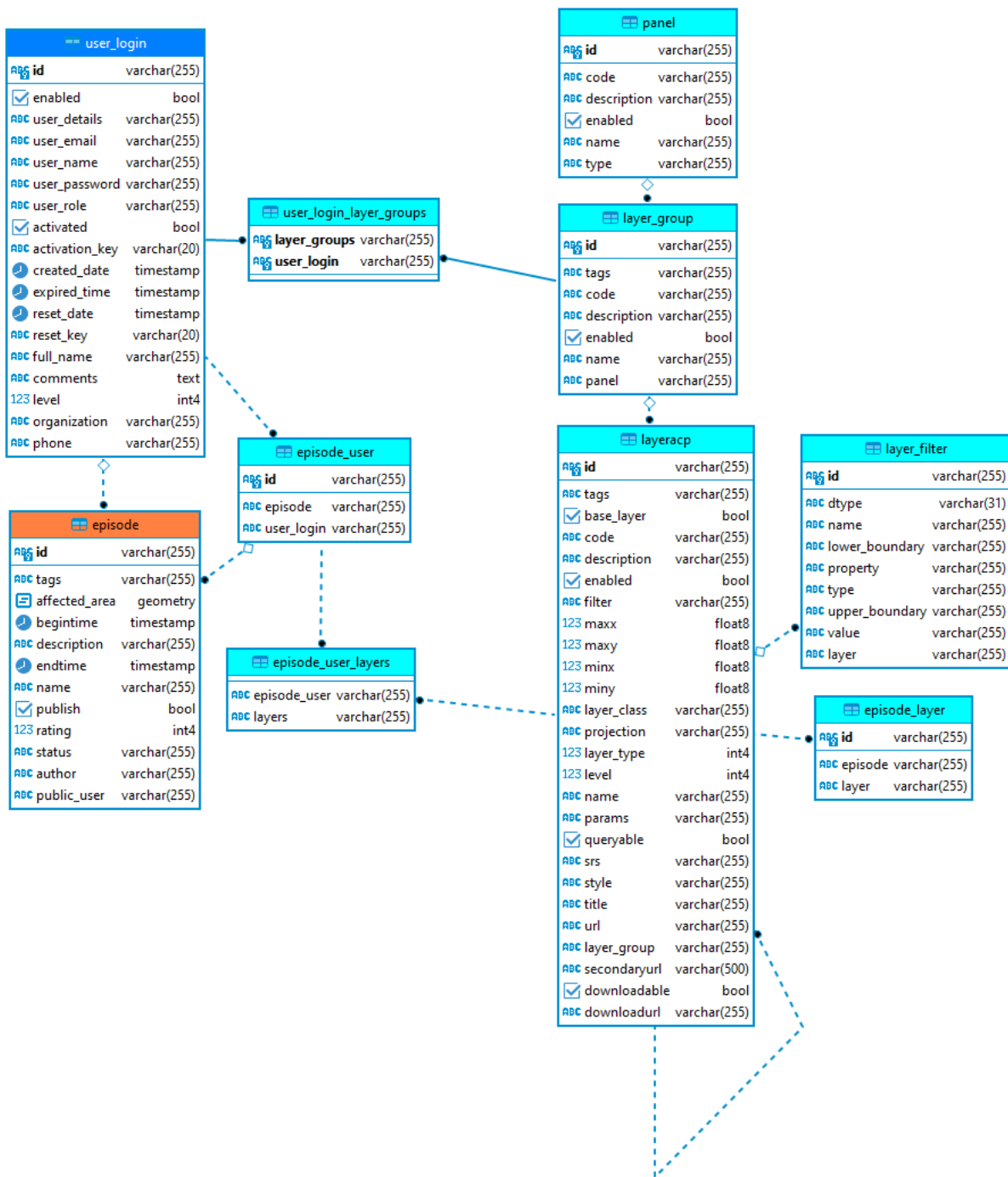


Fig. 17: Schema of the Layers relationships in the database.17

Finally, it is necessary to manage the COP with its contingencies or episodes. The term episode is used interchangeably to contingency and is maintained for compatibility with past solutions.

The following entity relationship scheme shows the different properties and relationships of a contingency COP. The contingency is created by a contingency manager, has an area and a beginning and end (which in this case refer to the associated COP). In addition, each contingency will have

associated COP users, as well as others. A contingency may have associated POLREP, SCATs or other types of reports.

ER diagram for episodes

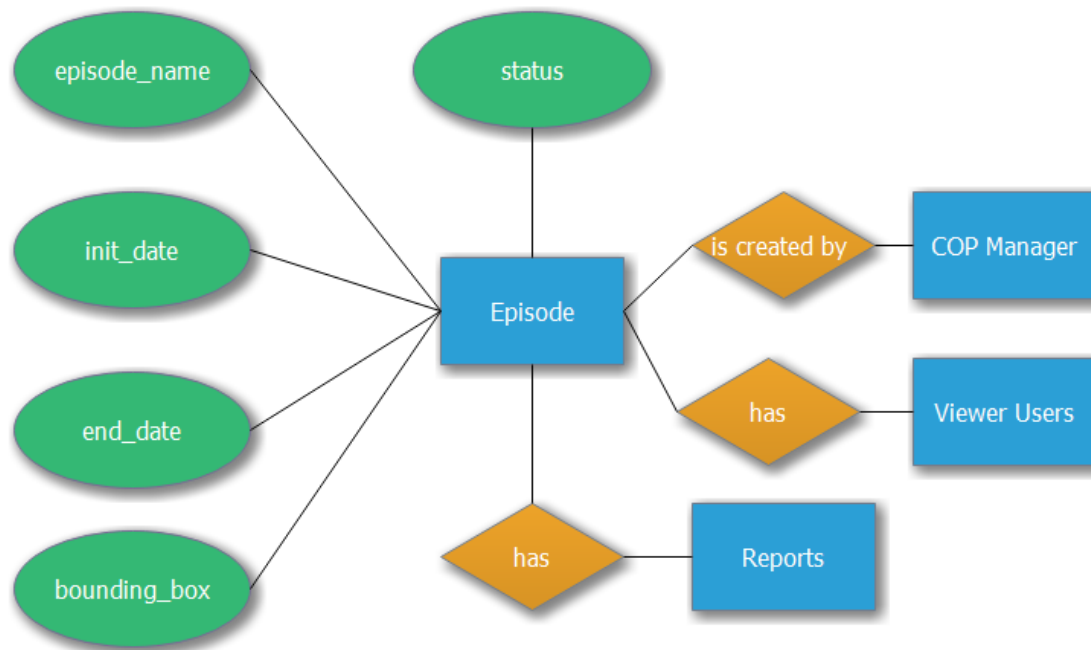


Fig. 18: Entity-Relationship Diagram for Episodes.18

This schema is represented in tables within the database as follows:

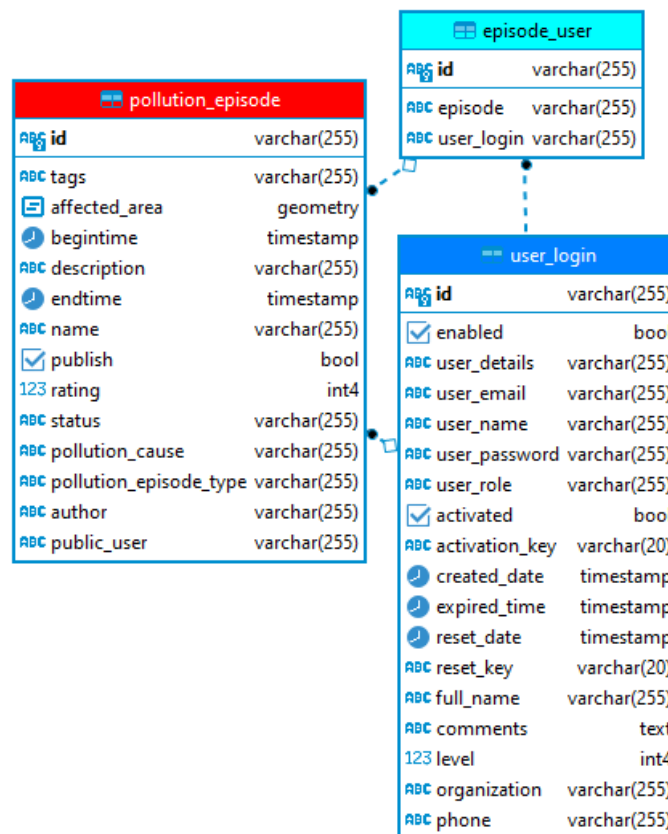


Fig. 19: Outline of Episodes relationships in the database.

5.2.2. COOPTOOL. Insertion tools

The COPTool not only allows you to manage COPs, contingencies and users, but also has the ability to allow the management of standard reports and associated information that is prepared during the contingency. These standard reports are the POLREP (Pollution Report) and the POLREP Costa (or SCAT). These reports are widely used by the international community and are based on the recommendations of the International Maritime Organization.²

Other associated information is the reports, or information during the contingency that may be reported by an agent or observer and that may consist of text, files and photographs or information about the disposition of the media and barriers.

²<https://www.rempec.org/en/our-work/pollution-preparedness-and-response/emergency-response/emergency-response/polrep>

<https://www.rempec.org/en/our-work/pollution-preparedness-and-response/response/tools/shoreline-assessment>

It has been decided that all this information (whether the reports, POLREPs or POLREP-Costa) may or may not be associated with an episode. The fact that they can be generated without having been associated with an episode allows greater flexibility when creating them, which is an advantage because much of the information could be recorded even before the creation of the COP the contingency itself.

For the insertion of any report of these, it is necessary on the one hand to be a COPTool user and on the other hand that this user has enabled the functionality of inserting each type of report.

On the other hand, the COPs manager can associate the reports that have been made to a specific contingency.

The following figures show the E-R diagrams for the POLREP and POLREP-Costa report. A new type of data ingestion will use a similar diagram.

ER diagram for POLREP

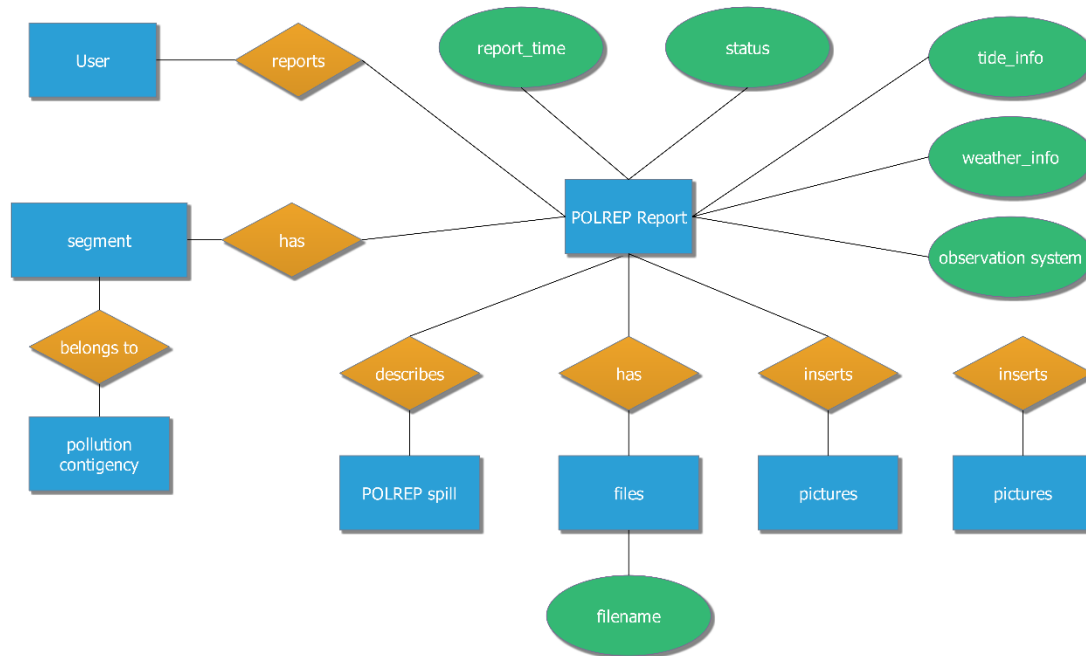


Fig. 20: Entity-Relationship Diagram for POLREP.20

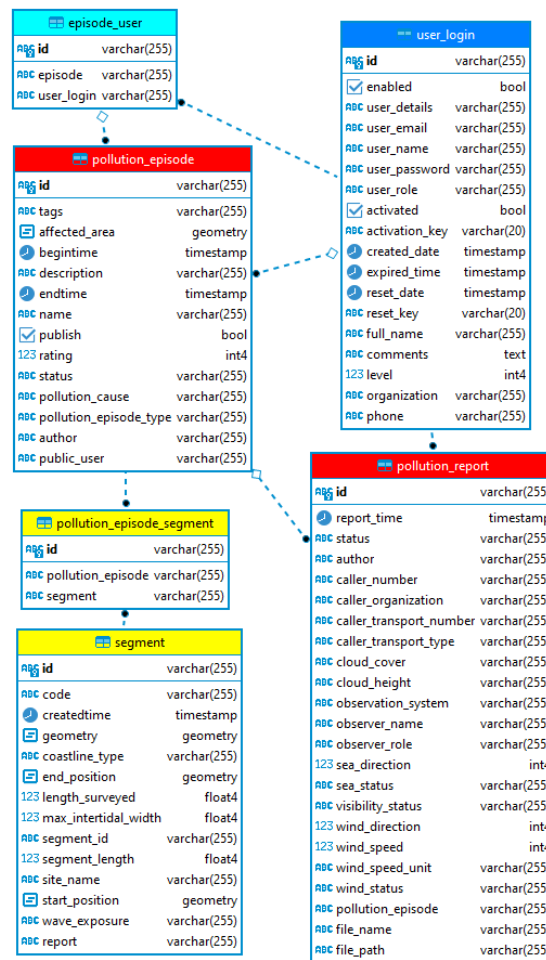


Fig. 21: Schema of POLREP relationships in the database.21

ER diagram for SCAT

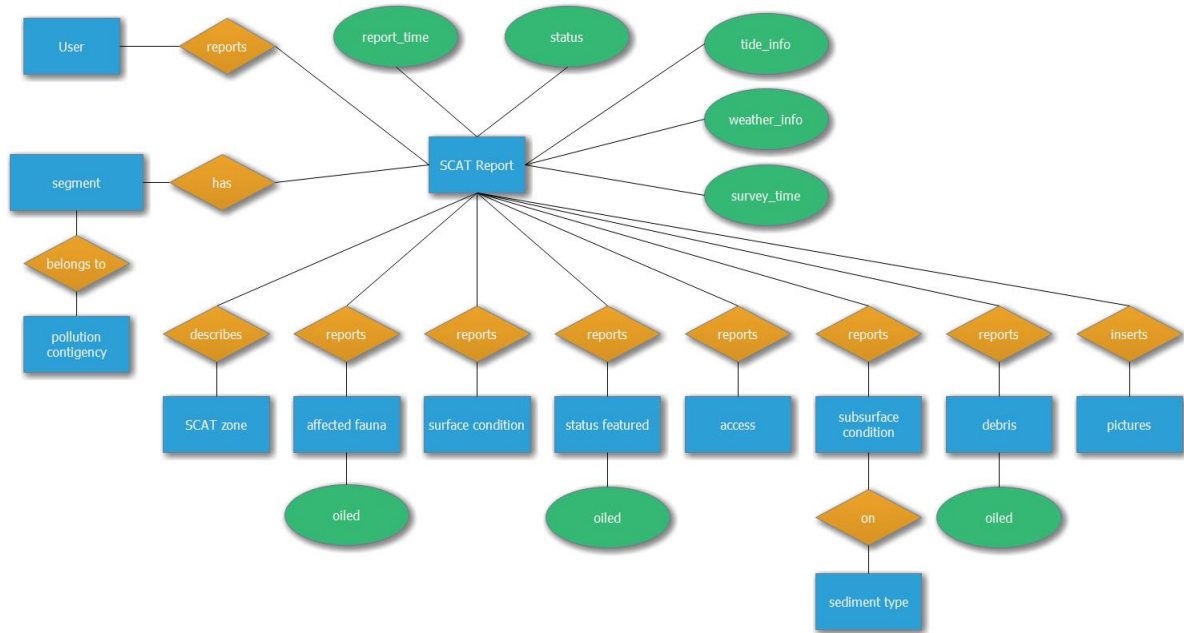


Fig. 22: Entity-Relationship Diagram for SCAT.22

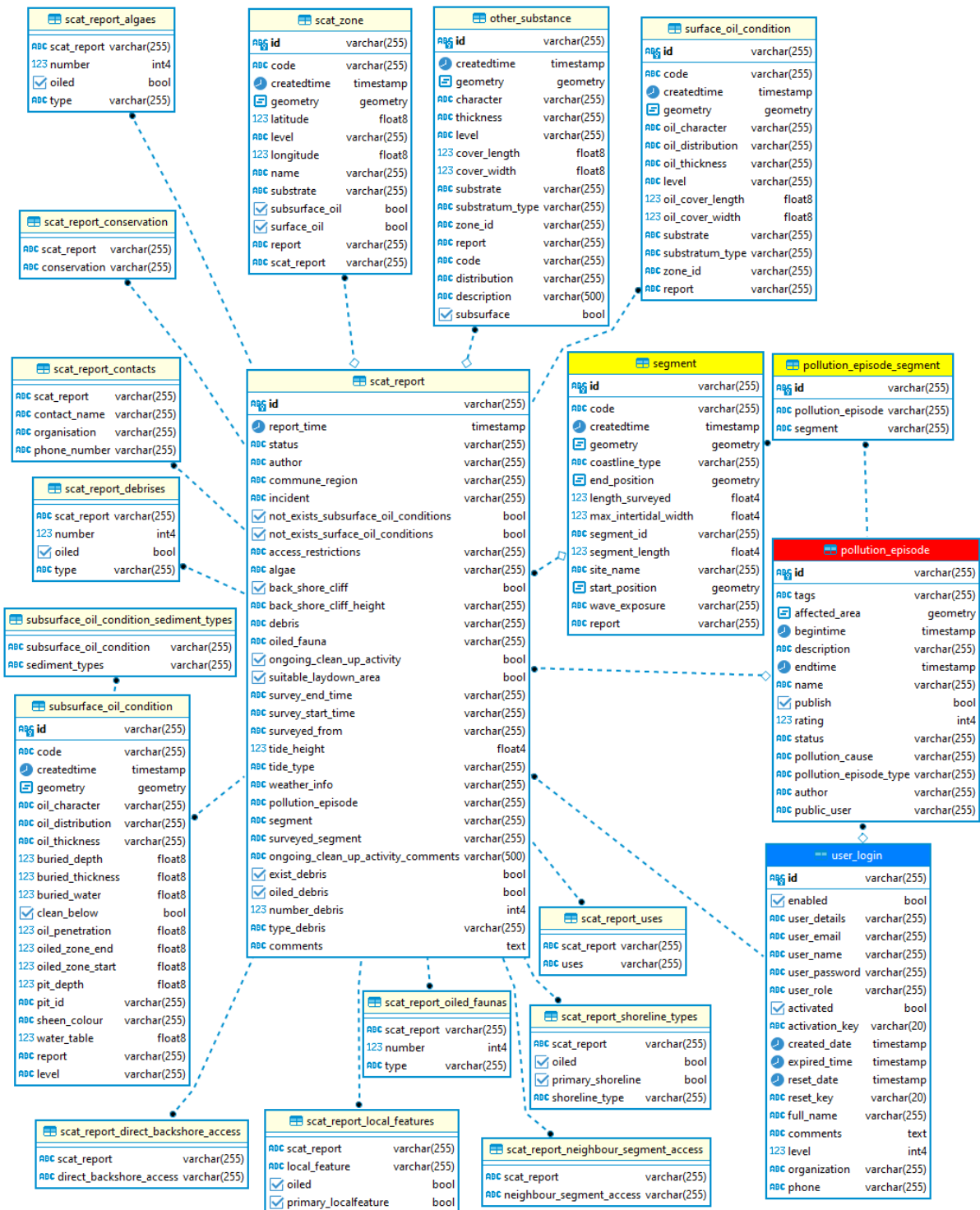


Fig. 23: Schema of SCAT relationships in the database.23

5.2.3. COOPTOOL. General outline

Once all the partial schemes are put together, the general scheme of the COptool is formed. The following figure shows the general outline of the entire COptool. each color in the table corresponds to each of the parts of the general database

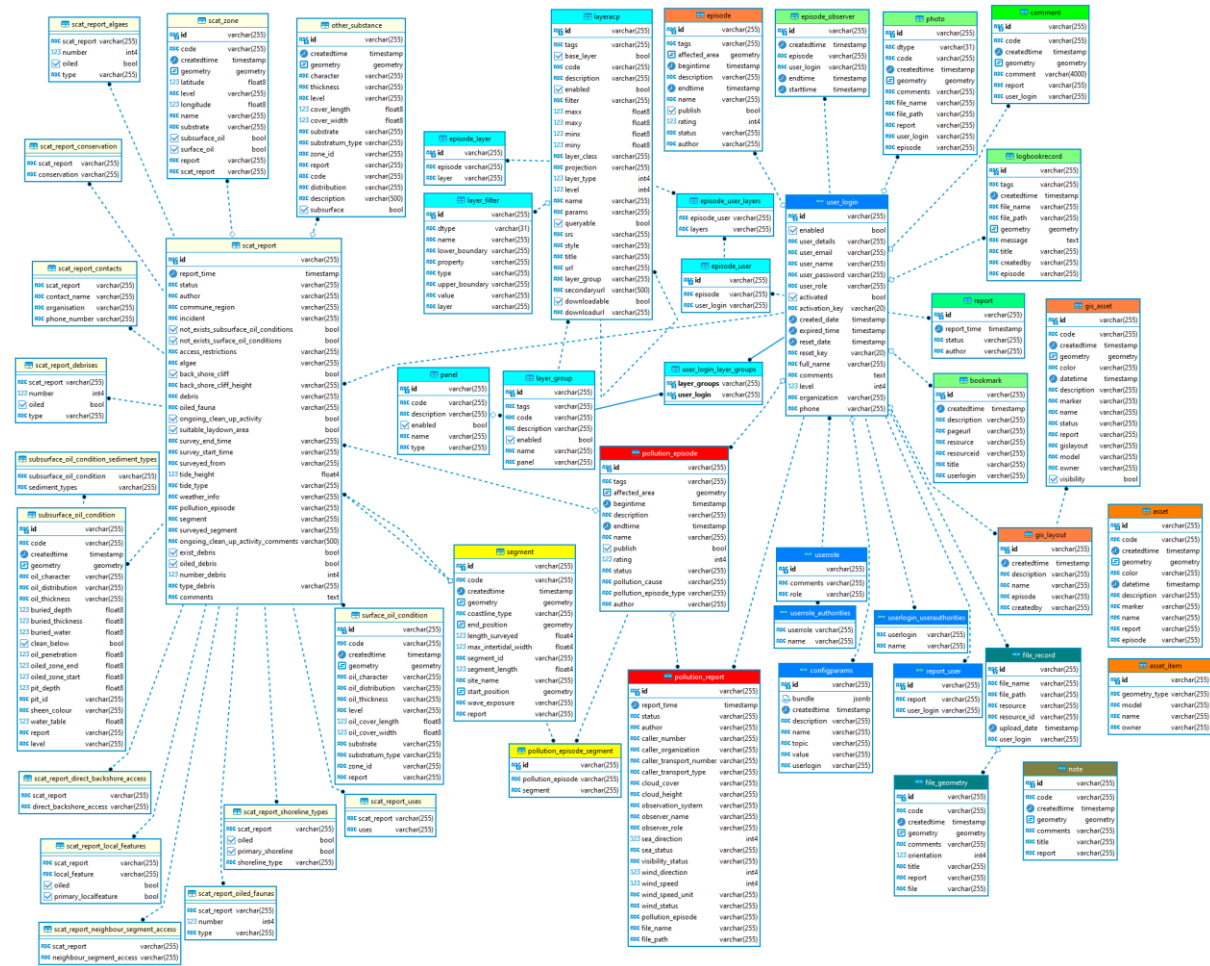


Fig. 24: Complete schema of the database relationships.24

For the access and representation of several tables, different views have been created, which give access to the data in the form of a table.

5.2.4. COOPTOOL. Auxiliary databases

As mentioned, this database depends on the needs of each agency and can be implemented in many different ways, although it must be remembered that, for this information to be displayed in the COP Viewer, queries must be able to be served through a WMS service. In our case, most of the maps served through WMS correspond to views associated with the database.

In this case, the associated database COP_AUX also used as the database engine to PostGIS. Each type of information will be associated with a schema in the database. For each layer that you want to show, a query will be built in such a way that the information appears in the form of a table with a geometry field and thus be able to serve it through the WMS service.

As an example, the following lines describe the scheme for aerial dispersion models of pollutants. In this case, the Entity-Relationship schema is as shown in the figure:

ER diagram for HNS Model

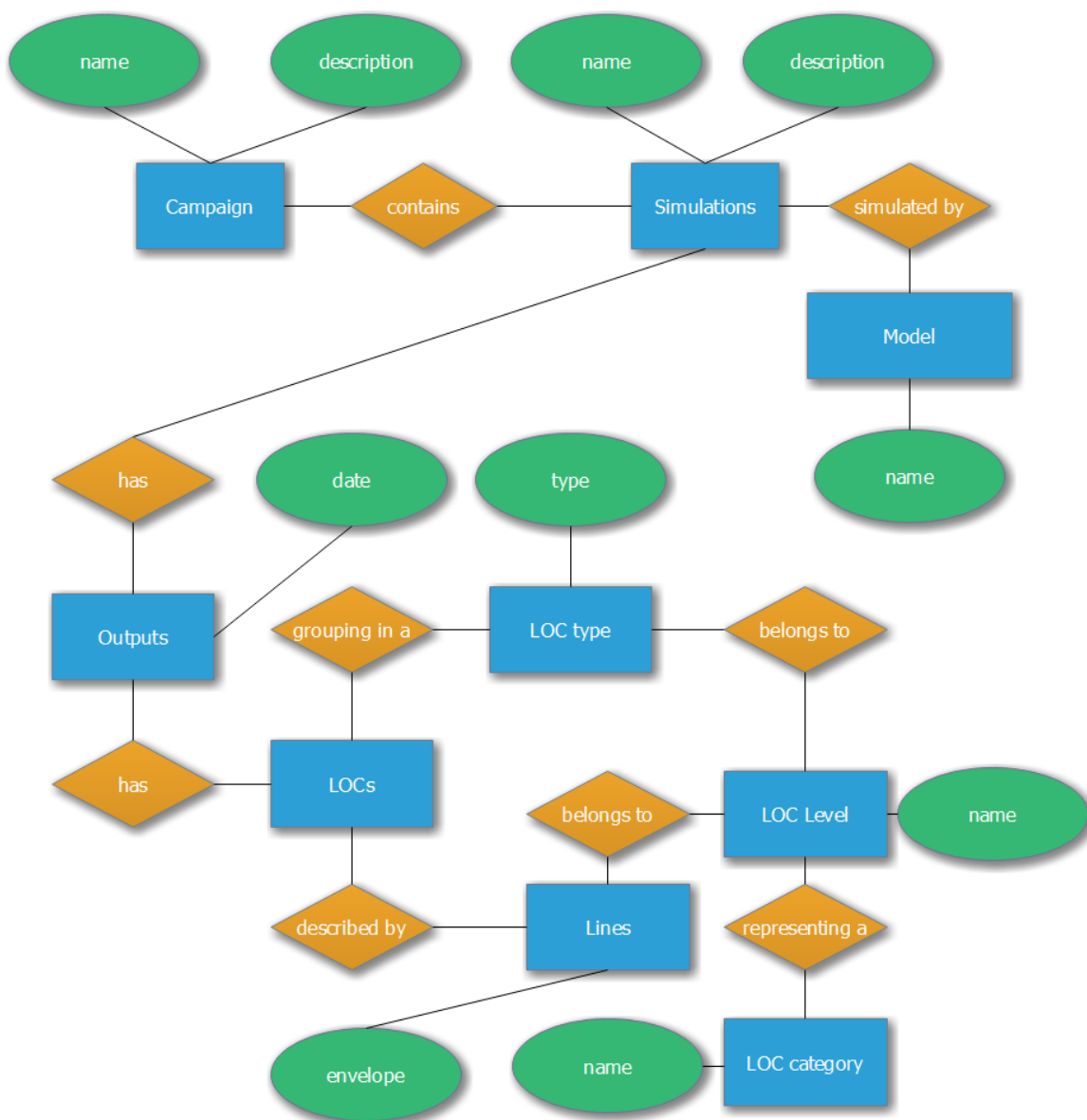


Fig. 25: Entity-Relationship Diagram for HNS Model.

This schema is implemented in the Auxiliary Database as shown in the figure.

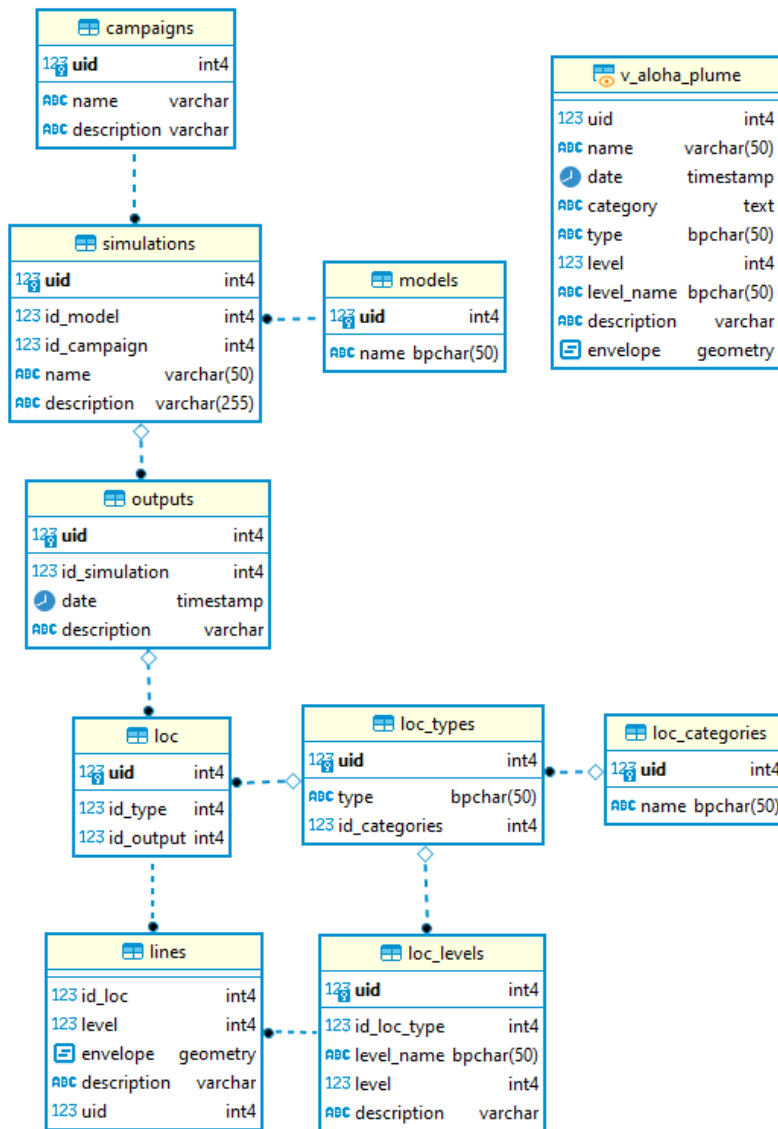


Fig. 26: Schematic of HNS Model relationships in the auxiliary database AUX_COP.26

As mentioned before, for the database information to become a WMS layer served by Geoserver, it must be a table or view that has a spatial component. That is why, accompanying the previous schemes, there are views of the data which are necessary to establish the wms layers that you want to serve.

The following table shows the required view *v_aloha_plume* to serve a LOC layer of a pollutant dispersion model from this database.


```
SELECT lines.id,
       simulations.name,
       outputs.initial_date,
       btrim(loc_categories.name::text) AS category,
       loc_types.type,
       lines.id_loc_level,
       loc_levels.level,
       loc_levels.level_name,
       lines.description,
       lines.envelope
FROM hns_models.lines
     JOIN hns_models.loc ON lines.id_loc = loc.id
     JOIN hns_models.outputs ON loc.id_output = outputs.id
     JOIN hns_models.simulations ON outputs.id_simulation = simulations.id
     JOIN hns_models.loc_types ON loc.id_type = loc_types.uid
     JOIN hns_models.loc_categories ON loc_types.id_categories =
loc_categories.uid
     JOIN hns_models.loc_levels ON loc_levels.uid = lines.id_loc_level
ORDER BY loc_levels.level DESC;
```

The ingestion of data in this system is inherent to each organization and its developments. In the case of the Camgal Plan, most of the insertions are done through ingestor software, developed in Python.

A series of scripts for inserting the information are listed below.

- Aloha2COP.py
- Drifter2COP.py

This software is included in the GitHub of the project, but as mentioned, most of the processes are specific to each organization and therefore, it is necessary to adapt these scripts to each instrument and / or institution.

6. Resources

The exact specifications of these applications were defined in conjunction with the advisory board, taking into account the resources and skills available. Its development is based on harmonized and/or standard formats and its source code is available in <https://github.com/MANIFETS-DSS>. It should be noted that, although the tools are as generic as possible, it is likely that some configuration and adaptation will need to be tailored to the purpose and specificities of the individual maritime authorities using the MANIFESTS decision support system.

The power of the MANIFESTS decision support system is demonstrated in the real example of its use by the Camgal Plan (<https://www.planacamgal.gal>) as well as in the transfer of this development to other members of the MANIFESTS project.



Due to its open source approach, this DSS can be replicated and adapted in any region and country.

7. Acknowledgments

Authors would like to express their sincere gratitude to the Galician Coast Guards for their invaluable help and advice in the development of the tool, as well as for their participation in field tests. Their dedication and expertise have been instrumental in ensuring that the tool meets the highest standards of accuracy and reliability. Authors are truly grateful for their support and partnership in this project.

Icons made by Freepik from www.flaticon.com.





D5.1 - MANIFESTS DSS - Implementation report

Web application – Model interface

Samuël Orsi

ACKNOWLEDGEMENT

The work described in this report was supported by the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) of the European Union through the Grant Agreement number 101004912 - MANIFESTS – UCPM-2020-PP-AG, corresponding to the Call objective “Enhancing prevention and protection from the effects of maritime disasters” under priority 1: “Developing response capacity for marine pollution”.

DISCLAIMER

The content of this document represents the views of the author only and is his/her sole responsibility; it cannot be considered to reflect the views of the European Commission and/or the Directorate-General for European Civil Protection and Humanitarian Aid Operations (DG-ECHO) or any other body of the European Union. The European Commission and the DG-ECHO is not responsible for any use that may be made of the information it contains.





Project Acronym	MANIFESTS
Project Full Title	MANaging risks and Impacts From Evaporating and gaseous Substances To population Safety
Gant Agreement Nr.	101004912
Project Website	https://www.manifests-project.eu/

Deliverable Nr.	D5.1
Status (Final/Draft/Revised)	Final
Work Package	WP5
Task Number	5.3
Responsible Institute	RBINS
Author/s	Samuël Orsi
Recommended Citation	Samuël Orsi (2023) D5.1 - MANIFESTS DSS - Implementation report Deliverable of MANIFESTS project
Dissemination Level	MANIFESTS Consortium, DG ECHO

Document History			
Version	Date	Modification Introduced	
		Modification Reason	Modified by
1.0.0	28/04/2023	First version	Samuël Orsi



1. Introduction.....	6
2. Objectives.....	7
3. Technical choices.....	7
3.1. Language: Python 3.....	7
3.2. Framework: Django LTS	8
3.3. Communication: OpenAPI 3.....	8
3.4. Handling of georeferenced data	8
3.4.1. Maps.....	8
3.4.2. Georeferenced objects.....	8
3.5. Legend for map data	9
4. Libraries used	9
5. Project architecture	10
6. App architecture.....	14
7. Added capabilities	17
8. Communications	18
8.1. To the computing service.....	18
8.2. From the computing service	19
9. Useful documentation.....	20





1. Introduction

When a pollution at sea occurs, comes the need to know where, when and how it will impact economic zones or protected areas. Mathematical modelling becomes the indispensable tool to organise any response strategy.

There are many mathematical models depending on the pollution scenario, each producing its own results. It is not easy to know how to request a simulation and who to contact.

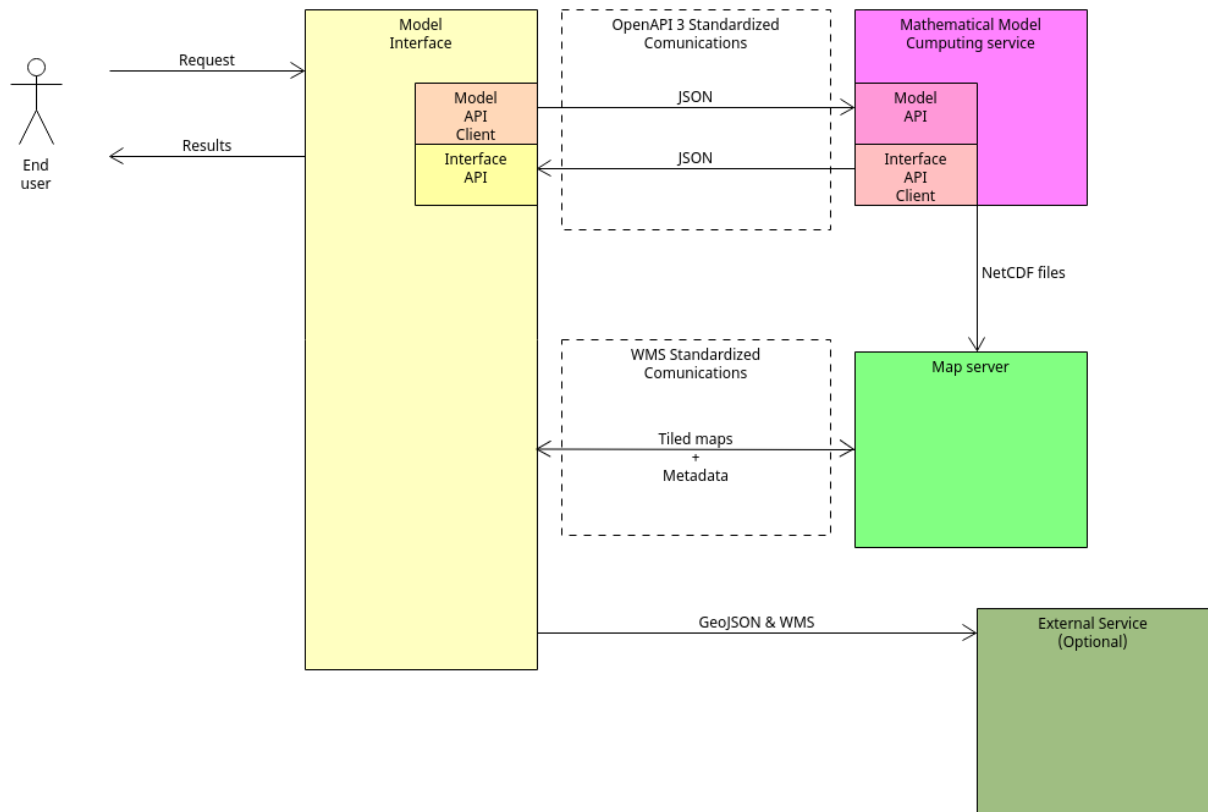


2. Objectives

The model interface web application aims at being a unified gateway between end users and mathematical models producing simulations.

As such, it:

1. Takes information from users
2. Transmits them to the calculation service
3. Retrieves results
4. Allow users to view or download results for further analysis



3. Technical choices

3.1. Language: Python 3

Python is well known in scientific computing with many scientific libraries. It has a large supporting community, ensuring its evolution and security patches.

3.2. Framework: Django LTS

It is necessary to use a framework to ease development process, debugging and application maintenance.

Django is an opensource python framework. It helps giving a base structure for the development such as:

- Database mappers (ORM – Object Relational Mappers)
- Self-webserver
- Frontend and backend development
- Comes with essential tools to handle common use cases (login, logout, user management, ...)

3.3. Communication: OpenAPI 3

The OpenAPI specification define a standard in communication interfaces which allow both humans and machines to understand how to interact with a service. There is no need to browse the source code or read any side documentation. An OpenAPI schema (language agnostic JSON document) serve this purpose and can even be used with code generators to build a client in any programming language.

On the model interface web application, this is handled by “Django REST framework”, a powerful and flexible toolkit for building Web APIs.

Outside model interface web application (on the computing server), this is handled by FastAPI, a lightweight web framework for building APIs with Python 3.

3.4. Handling of georeferenced data

3.4.1. Maps

Maps are handled by a Web Map Service (WMS). WMSs use an efficient standard protocol readily usable by any interactive map library (OpenLayers, Leaflet, ...)

3.4.2. Georeferenced objects

WKB (Well-known binary) is used to store point, trajectory and polygons coordinates in database. This format is easily convertible to GeoJSON.

3.5. Legend for map data

All WMS (Web Map Service) generate different format of legend. Some are horizontal, others are vertical, wide or large, ... We wanted the model interface web application to generate its own customisable legend regardless of the WMS used behind to take control of the layout.

This product includes color specifications and designs developed by Cynthia Brewer (<http://colorbrewer.org/>).

The matplotlib python library is used to accomplish this.

4. Libraries used

- Django 3.2 LTS
- Django REST Framework
- Django background tasks
- Django REST Framework - GIS
- Django REST Framework - SimpleJWT
- Django reCaptcha
- Django proxy
- Matplotlib
- Mod_WSGI
- Python decouple
- PyOpenSSL
- OpenAPI generator
- Poetry
- OpenLayers 6
- PostgreSQL with PostGIS 3 extensions

5. Project architecture

- ▼ Project
 - > app_1
 - > app_2
 - > app_3
 - > app_4
 - > app_5
 - > archives
 - > media
 - > Packages
 - ▼ project
 - ▼ settings
 - base.py
 - dev.py
 - production.py
 - > static
 - __init__.py
 - asgi.py
 - urls.py
 - wsgi.py
 - > project_dispatcher
 - > raster
 - > static
 - > templates
 - .dockerignore
 - .env
 - .gitignore
 - docker-compose.yml
 - Dockerfile
 - JWT_RSA_PRIVATE_KEY.pem
 - JWT_RSA_PUBLIC_KEY.pem
 - LICENSE
 - LICENSE.md
 - manage.py
 - README.md
 - requirements.txt

- “App_1” to “App_2” folders

The app name may change as needed.

Each app created inside the project represent a case, scenario, a different computing service to join, ...

They contain:

- Forms with required parameters to be send to a specific computing service
- Database to store result’s information
- Viewers allowing end users to exploit simulation results

- “Archives” folder

Folder containing ZIP archives of computed results.

This folder is not directly accessible via HTTP but only via Django views (within related app) ensuring access restriction.

- “Media” folder

Folder containing media files such as pictures, videos, ...

This folder is directly accessible via HTTP without restrictions.

It is managed by Django, do not put any file manually in it.

- “Packages” folder

Folder containing local dependencies (e.g. Communication module used to initiate a communication with the mathematical model or an external service)

- “Project” folder

The project name may change as needed.

Folder containing configuration files and common “static” files for all the project’s apps.

- “Project_dispatcher” folder

The app name may change as needed.

The dispatcher app serves as an entry point for the end user.

It manages:

- Password management form
- Signup form

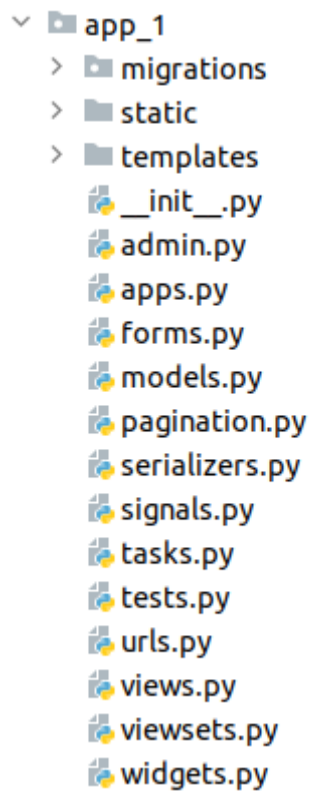
- Redirection to the other apps (App_1, ...) the end user has access to
- “Raster” folder
Folder containing results in form of raster pictures (plots, ...).
This folder is not directly accessible via HTTP but only via Django views (within related app) ensuring access restriction.
- “Static” folder
Folder managed by Django, do not put any file manually in it.
- “Templates” folder
Folder containing global HTML templates for web page rendition (error pages, base template page, ...)
- “.dockerignore” file
List of files and folder to ignore when generating a Docker image.
<https://docs.docker.com/engine/reference/builder/#dockerignore-file>
- “.env” file
File containing sensitive information not to be included in source code. (e.g. database password, ...)
- “.gitignore” file
List of files and folders to ignore when committing or pushing change to a git repository.
<https://git-scm.com/docs/gitignore>
- “Docker-compose.yml” file
Configuration file for Docker compose.
- “Dockerfile” file
Configuration file for the Docker container.



- “JWT_RSA_PRIVATE_KEY.pem” & “JWT_RSA_PUBLIC_KEY.pem” files
Asymmetric RSA key pair for JSON web token (JWT) security
- “LICENSE” & “LICENSE.md” files
Files containing the EUPL-1.2 license.
<https://joinup.ec.europa.eu/collection/eupl/eupl-text-eupl-12>
- “Manage.py” file
Django command line utility for administrative tasks.
<https://docs.djangoproject.com/fr/3.2/ref/django-admin/>
- “README.md” file
Readme file of the project
- “Requirement.txt” file
List of python dependencies required to run the project.



6. App architecture



- Migrations folder
Folder managed by Django command line utility, do not put or alter any file in it.
It contains database structure evolution.
- Static folder
Folder containing app related static files. You can manually add files here (JavaScript libraries, theme, logos, ...)
- Templates folder
Folder containing app related HTML templates used to generate web pages.
- Admin.py
Configuration file for the administration interface.



- **Apps.py**
Base configuration file of the app.
- **Forms.py**
File for Django “ModelForm” creation.
The simulation request forms are handled here.
- **Models.py**
File for managing database structures.
Each class declared here, inheriting from “django.db.models” will generate a new table in database.
These classes also allow usage of Django’s ORM to manipulate database records like python objects.
- **Pagination.py**
File where pagination schemes can be added to accommodate JavaScript libraries or external APIs requiring specific type of pagination not provided by default by Django REST Framework.
- **Serializers.py**
Serializers translates Django models into other formats (JSON, GeoJSON, python objects, ...)
A serializer class handle conversion both ways.
- **Signals.py**
A signal is a task executed when a model event occurs (e.g. insertion of data into a database table, ...)
- **Tasks.py**
File containing tasks declaration for Django background task package.
- **Tests.py**
File handling unit tests.





- **Urls.py**
File for app's URLs management.
This file needs to be imported into the project's urls.py file.
- **Views.py**
Views are functions taking a web request (GET, POST, ...) and returning a web response (HTML, JSON, ...).
Checks of permissions are handled here.

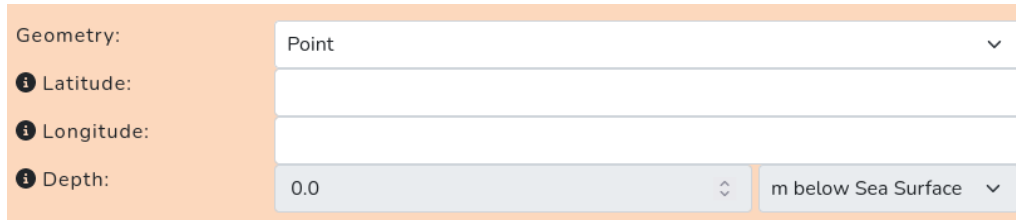
In this file are also "viewsets" declaration.
A viewset is a Django REST Framework class. It is a condensed way of declaring multiple views.
- **Viewsets.py**
File containing custom viewsets superclass.
It is useful for restricting available operations upon a specific API end point.
- **Widgets.py**
Widgets are form components, a Django's representation of a HTML input element.



7. Added capabilities

New capabilities have been developed in the scope of manifest project.

- The form widget “MANIFESTSSplitLocationWidget”:



The screenshot shows a form with four fields: 'Geometry' (a dropdown menu with 'Point' selected), 'Latitude' (an empty text input), 'Longitude' (an empty text input), and 'Depth' (a numeric input with '0.0' and a unit dropdown menu with 'm below Sea Surface' selected). Each field has an information icon to its left.

This widget allows a convenient management of “GeometryField” database objects.

- 2D and 3D geometries are supported
 - “Point”, “LineString” and “Polygon” geometries are supported
 - Detection of inverted latitude and longitude
 - Coordinate formats:
 - Decimal degree: 51.5°, 51.5° N, ...
 - Degree, minute decimal: -50° 30′, 50° 30′ S, ...
 - Degree, minute, second decimal: -50° 30′ 30″, 50° 30′ 30″ S, ...
- On the fly WMS legend generation:

Via dedicated view, each WMS requests to the map server are intercepted. If the user is not authenticated or try to access restricted data, its connection is refused.

In the case of a legit “GetLegend” request, an internal utility is triggered to generate the legend on the fly, ensuring that the displayed layout is always under control. Customisation is possible using Matplotlib’s colour bars plotting capabilities:

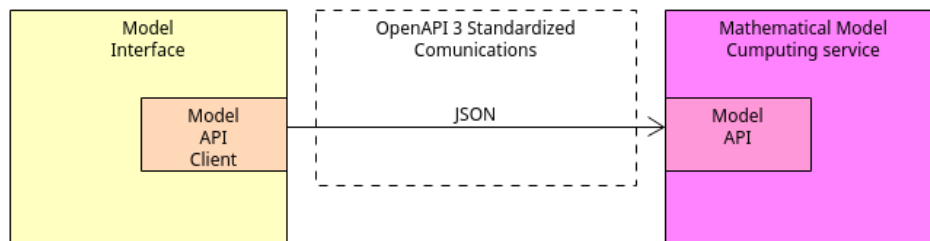
https://matplotlib.org/stable/tutorials/colors/colorbar_only.html

All other legit requests are passed to the map server.

8. Communications

8.1. To the computing service

On the computing service side, independently from the model interface web app, an OpenAPI 3 compliant API has been implemented using FastAPI framework. The “model API” is waiting data input to start a simulation.

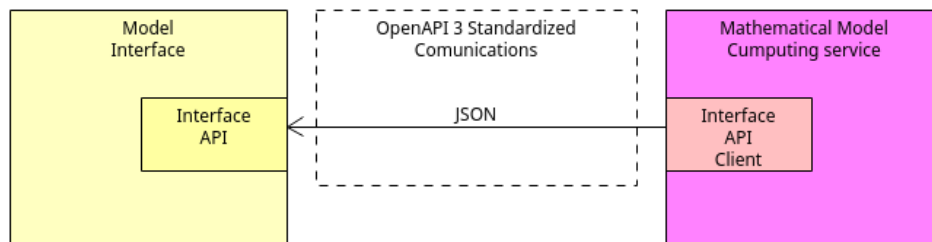


To connect the model interface to the computing service we must:

1. Get the API schema (openapi.json) from the model API.
2. Use a code generator for python to generate the model API client
3. Create a python wheel package
4. Install the model API client on the model interface project
5. Begin development of a dedicated app to this mathematical model

8.2. From the computing service

On the model interface web application, independently from the computing service, an OpenAPI 3 compliant API has also been implemented using Django REST framework. The “Interface API” is waiting data produced the simulation process.



To connect the computing service to the model interface we follow the same procedure:

1. Get the API schema (openapi.json) from the Interface API.
2. Use a code generator for to generate the Interface API client
3. Create a python wheel package
4. Install the Interface API client on the model interface
5. Sends results to the dedicated app

9. Useful documentation

- Django LTS Framework
<https://docs.djangoproject.com/en/3.2/>
- Django REST Framework
<https://www.django-rest-framework.org/>
- FastAPI Framework
<https://fastapi.tiangolo.com/>
- OpenAPI specification
<https://spec.openapis.org/oas/latest.html>
- GeoServer WMS reference
<https://docs.geoserver.org/2.22.x/en/user/services/wms/reference.html>
- Matplotlib
<https://matplotlib.org/stable/api/index>
- Docker
<https://docs.docker.com/>
- Git
<https://git-scm.com/docs>