



Advanced Design Tools for Ocean Energy Systems
Innovation, Development and Deployment

Deliverable D7.5

Database visualisation and maintenance tool

Lead Beneficiary	OCC
Delivery Date	30/03/2020
Dissemination Level	Public
Status	Released
Version	1.1
Keywords	Catalogue, Use Cases, Business Logic, API, GUI



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 785921

Disclaimer

This Deliverable reflects only the author's views and the Agency is not responsible for any use that may be made of the information contained therein

Document Information

Grant Agreement Number	785921
Project Acronym	DTOceanPlus
Work Package	WP 7
Related Task(s)	T7.3
Deliverable	D7.5
Title	Database visualisation and maintenance tool
Author(s)	Frederic Pons, Edouard Mei (OCC), Pablo Ruiz-Minguella, Vincenzo Nava (Tecnalia)
File Name	DTOceanPlus_D7.5_Data_Visualization_OCC_20200326_v1.1.docx

Revision History

Revision	Date	Description	Reviewer
0.1	19/11/2019	Structure and Initial Content included	F. Pons / E. Mei
0.3	09/12/2019	Initial version	Task 7.3 partners
0.8	13/12/2019	Full draft for QA review	Neil Luxcey (FEM)
1.0	20/12/2019	Final version for the EC	EC
1.1	30/03/2020	Update §3.2.4 and §3.4	F.Pons



EXECUTIVE SUMMARY

Deliverable D7.5 “Database visualisation and maintenance tool” of the DTOceanPlus project includes the details of the Catalogues module, and it presents the operational status of the implementation of the catalogues. This document includes the main requirements, use cases, as well as implementation details and is completed with some examples; it can serve as a technical manual of the Catalogues module.

This document describes the functionalities and the technical aspects of the code implemented to meet them. The Catalogues module provides users with a single source of reference data that can be managed and used in other modules or tools of the DTOceanPlus project to ease inputs during different phase of a project.

The Business Logic of the code, the functions of the Catalogues module, is implemented in Python 3 and the corresponding Application Programming Interface (API) in OpenAPI, both are generated using the open-source SAFRS framework. The Graphical User Interface (GUI) is generated using the open-source Flask-Admin framework and will be customized to be consistent with other modules of the DTOceanPlus platform. The GUI allows the users to interact easily with the Catalogues tools, for managing and visualising data.

The current approach for the implementation of the tool is geared towards easy configurability and adaptability to change to provided catalogues. A section of examples shows the operational capabilities of the tool, the currently provided catalogues, and remarks for any change of these catalogues’ data.



TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
LIST OF FIGURES	6
ABBREVIATIONS AND ACRONYMS	7
1. INTRODUCTION.....	8
1.1 SCOPE AND OUTLINE OF THE REPORT	8
1.2 SUMMARY OF THE DTOCEANPLUS PROJECT	8
2. USE CASES AND FUNCTIONALITIES	10
2.1 THE USE CASES	10
2.1.1 Managing the catalogues.....	10
2.1.2 Using the catalogues	11
2.2 THE FUNCTIONALITIES	11
2.2.1 Browsing the Catalogues	11
2.2.2 Managing the Catalogues.....	12
2.2.3 Using the Catalogues from another module	13
3. THE IMPLEMENTATION.....	14
3.1 MAIN PRINCIPLES	14
3.2 THE ARCHITECTURE	14
3.2.1 Database	14
3.2.2 Business Logic	15
3.2.3 API.....	15
3.2.4 GUI	15
3.3 THE TECHNOLOGIES	20
3.4 THE TESTING AND VERIFICATION	21
3.4.1 Description	21
3.4.2 Backend	21
3.4.3 Frontend.....	22
3.4.4 E2E tests (Cypress)	23
3.4.5 E2E Application integration tests (Cypress)	24
3.5 RELEASE	25
4. EXAMPLES.....	26



4.1 Generic presentation	26
4.1.1 Logistics and Marine Operations:	26
4.1.2 Station Keeping	27
4.1.3 Energy Delivery	27
4.1.4 Energy Transformation.....	28
4.1.5 Environment and Social Acceptance.....	28
4.2 Energy Delivery Example.....	28
5. CONCLUDING REMARKS AND FUTURE WORK	30
5.1 Conclusion	30
5.2 Future work	30



LIST OF FIGURES

Figure 1-1 Representation of DTOceanPlus tools	9
Figure 2-1 Catalogues Use Cases	10
Figure 3-1 GUI mockup for DTOceanPlus start page	16
Figure 3-2 GUI mockup of the Catalogues main page	16
Figure 3-3 Browse Catalogue	17
Figure 3-4 Add a value to a Catalogue	18
Figure 3-5 EDIT a value to a Catalogue	18
Figure 3-6 Select a value in a catalogue	19
Figure 3-7 Backend test report.....	22
Figure 3-8 End to End test report.....	23
Figure 3-9 Application integration test report	24
Figure 3-10 repository of the catalogue module	25
Figure 4-1 List of catalogues by Categories	26
Figure 4-2 Energy delivery catalogue	28
Figure 4-3 Collection point	29



ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
BL	Business Logic
CRUD	Create, Read, Update and Delete
CSV	Comma-Separated Values
EC	Energy Capture (module)
ED	Energy Delivery (module)
ESA	Environmental and Social Acceptance (module)
ET	Energy Transformation (module)
GUI	Graphic User Interface
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
JSON	Java Script Object Notation
LMO	Logistics and Marine Operations (module)
RAMS	Reliability, Availability, Maintainability, Survivability (module)
REST	Representational State Transfer
RDBMS	Relational Database Management System
ROV	Remotely Operated Vehicle
SAFRS	SqlAlchemy Flask-Restful Swagger
SC	Site Characterisation (module)
SG	Stage-Gate (tool)
SI	Structured Innovation (tool)
SK	Station Keeping (module)
SLC	System Lifetime Costs (module)
SPEY	System Performance and Energy Yield (module)
SQL	Standardised Query Language
URL	Uniform Resource Locator
VEC	Vessel-Equipment Combinations
YAML	Yet Another Markup Language



1. INTRODUCTION

1.1 SCOPE AND OUTLINE OF THE REPORT

All the DTOceanPlus tools will require a large amount of data to perform the computations they are designed for. Conveniently, as it was in the DTOcean platform, data will be stored in a global database, whose main functionality is therefore to store input data. The purpose of the global database is: to reduce the burden on the user in the phase of inputting information before running the tools as well as to provide a static source of stored data which could be used not only while running several simulations using the same scenario/project, but also for different scenarios/projects.

The Catalogue module serves as a centralised storage for common references of the applications. It contains a list of catalogues that will be accessible by any module. Its purpose is to store permanent data shared among the different design tools, reducing inefficiencies and ambiguities as well as reducing the burden in uploading data by the user. The modules will consult the catalogues through one or several services offered by themselves or the main application.

This document summarises:

- The use cases and the functionalities of the Catalogues module, namely providing the user with a reference database to simplify data inputting in other DTOceanPlus tools. Moreover, the module also provides management functionalities allowing users to manage the data.
- The implementation of the module. The module is implemented using frameworks to provide a reactive implementation, which can be adapted and configured to data provided by other DTOceanPlus partners during development phase.
- Examples to describe the data format provided in the implementation.

1.2 SUMMARY OF THE DTOCEANPLUS PROJECT

The Database visualisation and maintenance tool, also referred to as Catalogue module, belongs to the platform of tools "DTOceanPlus" developed within the EU-funded project DTOceanPlus.

DTOceanPlus will accelerate the commercialisation of the Ocean Energy sector by developing and demonstrating an open source suite of design tools for the selection, development, deployment and assessment of ocean energy systems (including sub-systems, energy capture devices and arrays).

At a high level, the suite of tools developed in DTOceanPlus will include:

- ▶ **Structured Innovation Tools SI**, for concept creation, selection, and design.
- ▶ **Stage Gate Tools SG**, using metrics to measure, assess and guide technology development.
- ▶ **Deployment Tools**, supporting optimal device and array deployment:
 - Site Characterisation SC (e.g. metocean, geotechnical, and environmental conditions);
 - Energy Capture EC (at an array level);
 - Energy Transformation ET (PTO and control);
 - Energy Delivery ED (electrical and grid issues);



- Station Keeping SK (moorings and foundations);
- Logistics and Marine Operations LMO (installation, operation, maintenance, and decommissioning).
- ▶ **Assessment Tools**, to quantify key parameters:
 - System Performance and Energy Yield SPEY;
 - System Lifetime Costs SLC;
 - System Reliability, Availability, Maintainability, Survivability (RAMS);
 - Environmental and Social Acceptance ESA.

These will be supported by underlying common digital models and a global database, as shown graphically in Figure 1-1.

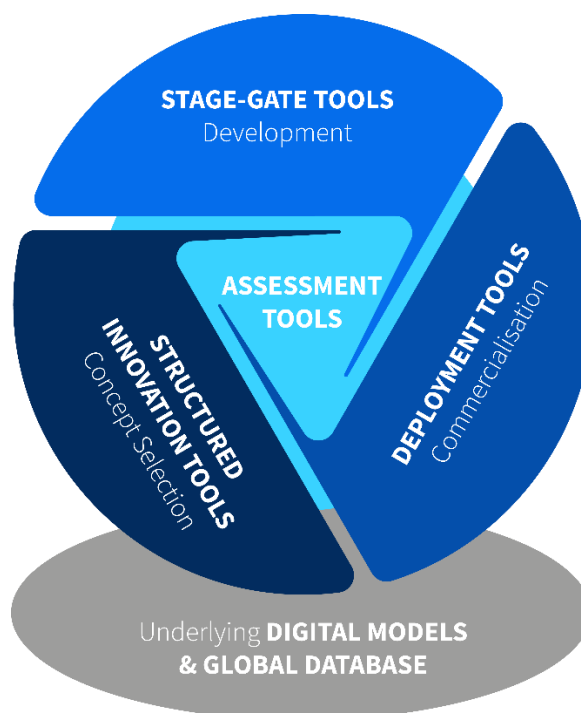


FIGURE 1-1 REPRESENTATION OF DTOCEANPLUS TOOLS

2. USE CASES AND FUNCTIONALITIES

2.1 THE USE CASES

The Catalogues module regroups a list of catalogues which are used as a source of static, reference data. It is intended to allow DTOceanPlus users, either technology developers, project developers or stakeholders to browse, manage and use the data stored in the catalogues.

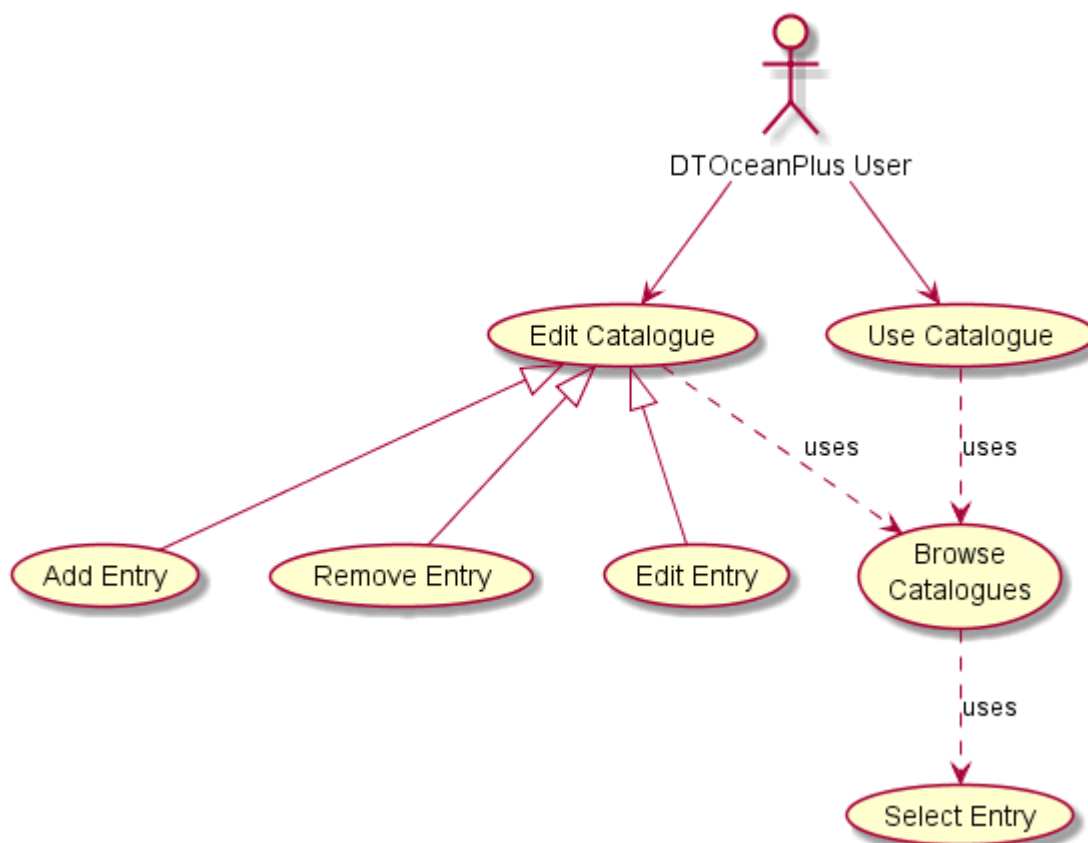


FIGURE 2-1 CATALOGUES USE CASES

Thus, the Catalogues module provides a centralised and stable point of entry for various types of data. For a user of the DTOceanPlus tools there are two main identified use cases. The first one consists of managing the catalogues themselves, and the second one of using the catalogues data from within the modules of the application.

2.1.1 Managing the catalogues

Managing a catalogue consists in the CRUD operations: create, read, update, and delete. It is possible for users to edit the existing data stored in catalogues, add entries and remove entries. This can be

done programmatically by other modules of the DTOceanPlus tools, or its users via a user interface, making the process easier and more flexible for any users of the DTOceanPlus platform.

Catalogues will be populated by importing existing data from existing files. Different file formats are to be supported (CSV, JSON, Excel).

2.1.2 Using the catalogues

The catalogues allow to display and expose data relevant to DTOceanPlus tools. Data are to be exposed via API or displayed via GUI and can be exported in the supported import format (CSV, JSON, and Excel).

A user can interact with the Catalogue module either directly or from within another module in the frame of a DTOceanPlus project.

While interacting directly with the Catalogues, the main usage consists of browsing existing data, and managing existing data (editing) or the catalogue (by adding or removing data). The module is user-friendly and provides functionalities to search, filter and show/hide data to improve finding and reading of the relevant data. It is also possible to extract and export this data.

Users can also be invited to select catalogues entries (either one, all, or some using filtering features) from within DTOceanPlus tools to be consumed as inputs for the different modules. The process should be as follows:

- Link to the catalogue
- Open the selected catalogue and display the values
- User selects a value and validate
- Selected value is displayed in the module

2.2 THE FUNCTIONALITIES

Next there is a list of functionalities required by identified use cases. These functionalities are accessible by API and GUI.

2.2.1 Browsing the Catalogues

Users can browse catalogues to visualize existing stored data, which can be managed if necessary.

By default, all listed entries of a catalogue are displayed in full, but users can further filter entries by attribute value to visualize only relevant ones, or search for a specific entry.

2.2.1.1 List and display all values of a catalogue

When browsing a catalogue, all entries are listed and displayed. Pagination in the GUI will be provided to ease navigation, as well as possibly hiding columns.



2.2.1.2 Filters

Filters for each column will be provided to order catalogue entries by relevant value. These are basic filters, alphabetic order, and numerical one.

2.2.1.3 Search

Users can search for an entry by providing a value to be searched in the current catalogue (and specifying a column).

2.2.2 Managing the Catalogues

Users can manage the Catalogues by providing new data or modifying existing ones. This can be done via the Catalogues' GUI or through scripts.

2.2.2.1 Import data

The Catalogues module provides importing functionality via dedicated scripts. It can import data from different file formats (CSV, JSON, and Excel are likely examples) and validate data against the defined data model.

This will be the preferred way to load large amount of data into the Catalogues' database. However, the Catalogues will provide a functionality to export the data to the different file formats.

2.2.2.2 Add a value in a catalogue

Users of Catalogues may have to add a new entry to the existing data. This can be done via the API or the GUI and will be a punctual action.

In the GUI, a form will prompt the user to enter all required attributes. The form will also check for the format of the data entered to check the validity of the new entered entry.

2.2.2.3 Edit a value in a catalogue

Users of Catalogues may have to edit an existing entry of a catalogue. This can be done via the API or the GUI and should be a punctual action.

In the GUI, a form will prompt the user to enter new values. Fields for the attribute will display existing data, and if left unchanged will retain existing value. The form will also check for the format of the data entered to check the validity of the new entered entry.

2.2.2.4 Remove a value in a catalogue

Users of catalogues may need to remove an existing entry from a catalogue. This can be done via the API or the GUI and will be a punctual action.



Since it can be difficult to verify if the value the user wants to delete has been used or not, it will be only possible to “disable” a value. When a value is disabled it cannot be selected anymore but projects that use it remain valid.

2.2.3 Using the Catalogues from another module

Users can be prompted to use the Catalogues from within another module, to provide inputs for computation. This can be done by adding a custom value (see §2.2.2.2), selecting an existing value or exporting an existing value.

2.2.3.1 Select a value in catalogue

Catalogues can will be used by other modules as reference data. For example, a module will ask the user to select one or several value in a specific Catalogue, like a Port or a Type of Soil.

For this, the module will provide a link to the corresponding Catalogue, and when the user will click on this link, the Catalogue module will present the corresponding data, with filtering functionalities (see §2.2.1.2 and §2.2.1.3) to simplify the selection process. Check boxes will allow the selection multiple input entries.

2.2.3.2 Export data

While not limited to usage from within a module, it can be necessary to provide data in a specified format (such as JSON or CSV) to another module via API. This would be the case for modules not requiring the users to make the choice directly but rather when the module is automating the selection process after a preliminary optimization computation.



3. THE IMPLEMENTATION

3.1 MAIN PRINCIPLES

The implementation strategy is to provide an easily configurable, scalable and maintainable solution during the operational phase of the related DTOceanPlus tools, as currently the general description of catalogues and the module's needs are provided, but the final form of the data might suffer changes.

Thus, during the integration, verification and validation of the tools it will be easy to accommodate (last minute) changes (such as adding/removing columns, changing types or constraints, adding/removing catalogues).

This is done by using frameworks automating the generation of the database models and tables, importing of data, automating the generation of the Business Layer, API and GUI.

3.2 THE ARCHITECTURE

As other DTOceanPlus tools, the Catalogues module is implemented in two layers a Business and Service layer, which here revolve around providing visualization of data from a database.

The Business Layer covers the Business Logic mainly the CRUD operations, ability to create, read, update and delete entries from a database, for the management of data in a catalogue.

The Service Layer covers the Service Logic, exposing the business logic via API (Application Programming Interface) or GUI (Graphic User Interface). The API allows other modules to interact directly with the Catalogues module, while the GUI allows the users to do so.

The code is implemented using Python and the web framework Flask, as the other third-party libraries are detailed in §3.3.

3.2.1 Database

The Catalogues module provides storage for a set of reference data.

The persistence of data can be ensured by using a Relational Database Management System (RDBMS); as the catalogue can be classified as a "simple" or "relational" table. Most of catalogues are simple tables, a simple table consists of a set of independent columns, whereas relational tables are those establishing a 1-to-1 or 1-to-many relationship.

In order to guarantee high quality levels of the data stored, the database applies data validation rules on new and updated data records. The levels at which such validation may operate (not an exhaustive list) are Data Type, Field Size, Nullability, and Uniqueness.

The database can be deployed locally, or on a server.



3.2.2 Business Logic

The business logic layer is developed using Python and associated libraries. It provides CRUD (create, read, update and delete) principles for its data and database management.

The skeleton of the Business Logic methods is created by a script that interprets the structure of each table, and the SAFRS framework generates the corresponding SQLAlchemy models, and CRUD operations.

Note regarding §2.2.2.1: during the integration phase (to improve reactivity to change of catalogue format until they are finalized) the import function will automate the model definition and Business Logic and its corresponding OpenApi.

3.2.3 API

The development principles for the API of the Catalogues module follow those of the API of the DTOceanPlus software as a whole: the API will follow a Representational State Transfer (REST) approach and using HTTP as the transport protocol.

The OpenAPI language is adopted to document the available API functionalities. The OpenAPI file (in JSON or YAML format) indicates all the paths, services and schemas provided to other modules.

The API and its OpenAPI documentation are automatically generated from the data models of the BL using the SAFRS framework.

3.2.4 GUI

The Flask frameworks provides routing (pairing HTML to URL) to display information in different paths. The generation of the GUI is automated on the basis of existing data models described in the Business Layer using the flask-admin library.

The following mock-ups describes the GUI to catalogues.

A link on the Start page will give access to the user the list of Catalogues.



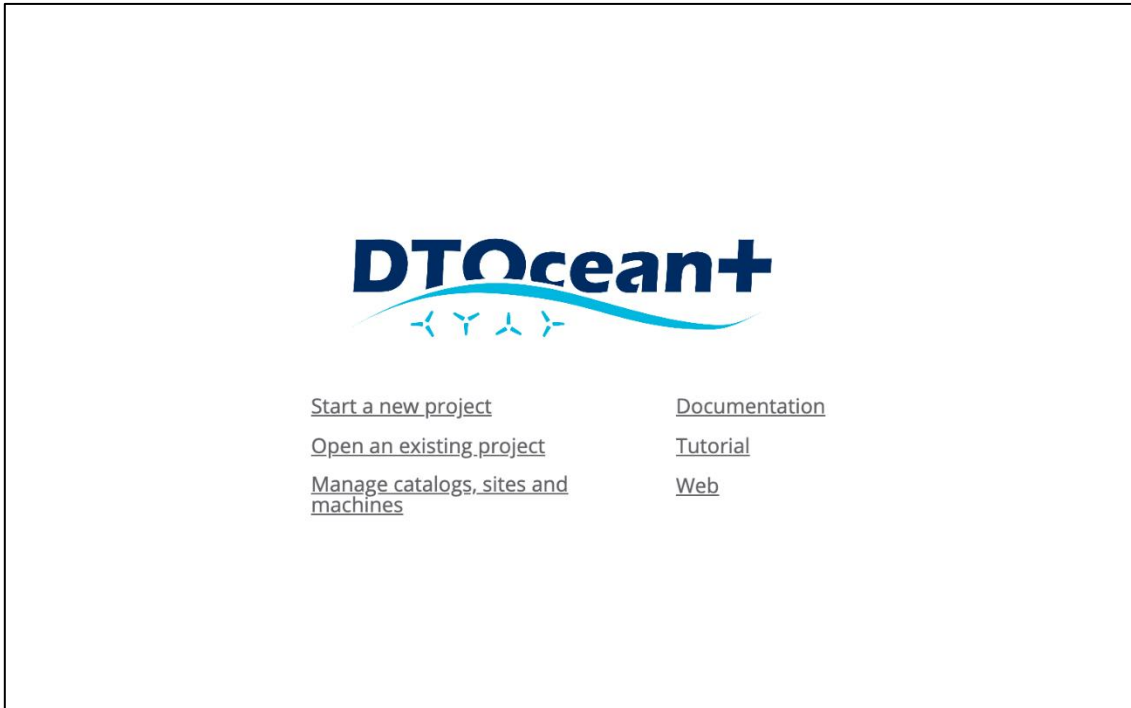


FIGURE 3-1 GUI MOCKUP FOR DTOCEANPLUS START PAGE

3.2.4.1 Manage catalogues

The Catalogues main page displays all catalogues by Modules or Categories (see §4 for examples).

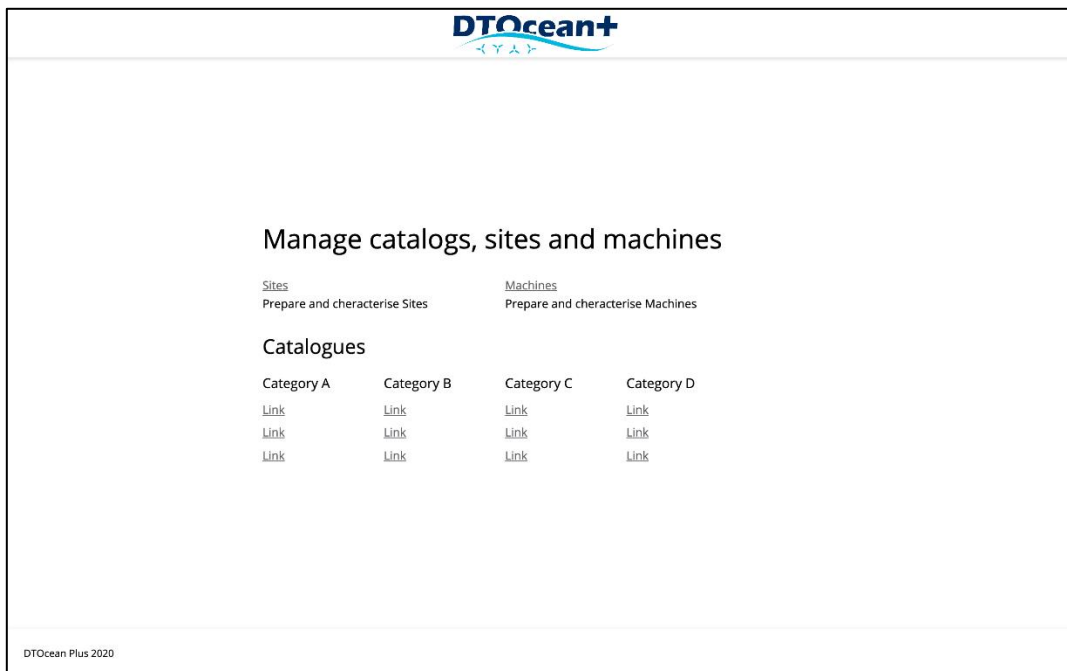


FIGURE 3-2 GUI MOCKUP OF THE CATALOGUES MAIN PAGE

By Clicking on a Catalogue, the user will be able to browse a specific catalogue.

All entries will be displayed. For tables with a large number of columns, there will be offered the possibility to show/hide them.

Filter features will help the user to browse the list based on value of an attribute.

From the same page, the user will be able to add, edit or delete an entry.

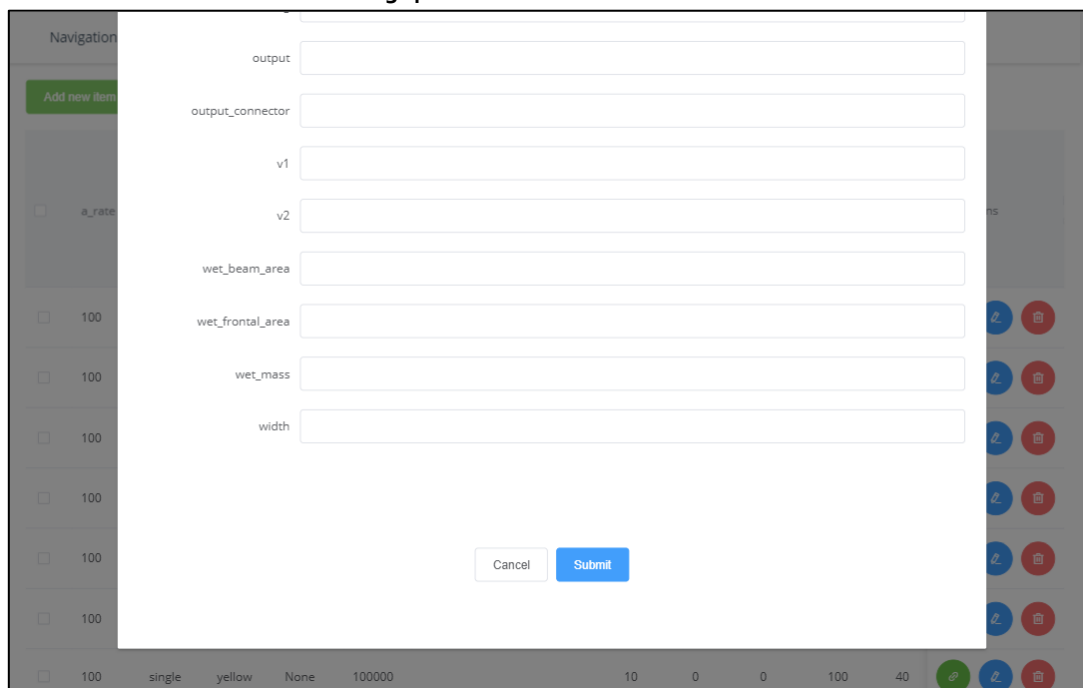
Navigation		Catalog												Operations		
<input checked="" type="checkbox"/>	a_rate	busbar	colour	cooling	cost	critical_f ailure_max	critical_f ailure_mean	critical_f ailure_min	depth	dry_bea m_area	dry_fron tal_area	dry_mas s	fibres			
<input checked="" type="checkbox"/>	100	single	yellow	None	150000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	150000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>	100	single	yellow	None	100000				10	0	0	100	40			

FIGURE 3-3 BROWSE CATALOGUE

Navigation		Catalog												Operations			
<input type="checkbox"/>		a_rate	busbar	colour	cooling	cost	critical_f ailure_max	critical_f ailure_mean	critical_f ailure_min	depth	dry_bea m_area	dry_fron tal_area	dry_mas s	fibres			
<input type="checkbox"/>		100	single	yellow	None	150000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	150000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	150000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	100000				10	0	0	100	40			
<input type="checkbox"/>		100	single	yellow	None	100000				10	0	0	100	40			



FIGURE 3-4 ADD A VALUE TO A CATALOGUE



The screenshot shows a web application interface for adding a value to a catalogue. A central white modal window is overlaid on a grey background. The modal contains several text input fields, each with a label to its left: 'output', 'output_connector', 'v1', 'v2', 'wet_beam_area', 'wet_frontal_area', 'wet_mass', and 'width'. At the bottom of the modal are two buttons: 'Cancel' and 'Submit'. The background interface is partially visible, showing a 'Navigation' menu on the left with an 'Add new item' button, a list of items with checkboxes and values (e.g., 'a_rate', '100'), and a bottom status bar with various parameters like 'single', 'yellow', 'None', '100000', '10', '0', '0', '100', '40'.

FIGURE 3-5 EDIT A VALUE TO A CATALOGUE

3.2.4.2 Use catalogues from a module

When a module needs the user to select a value in a catalogue, it will provide the user a way to select a value in the catalogue. For example, by providing a link to the corresponding catalogue. When the user clicks this link, he can browse the catalogue and select a value. Once the user selects a value, then the application goes back to the module's page and shows the value of the catalogue selected by the user.

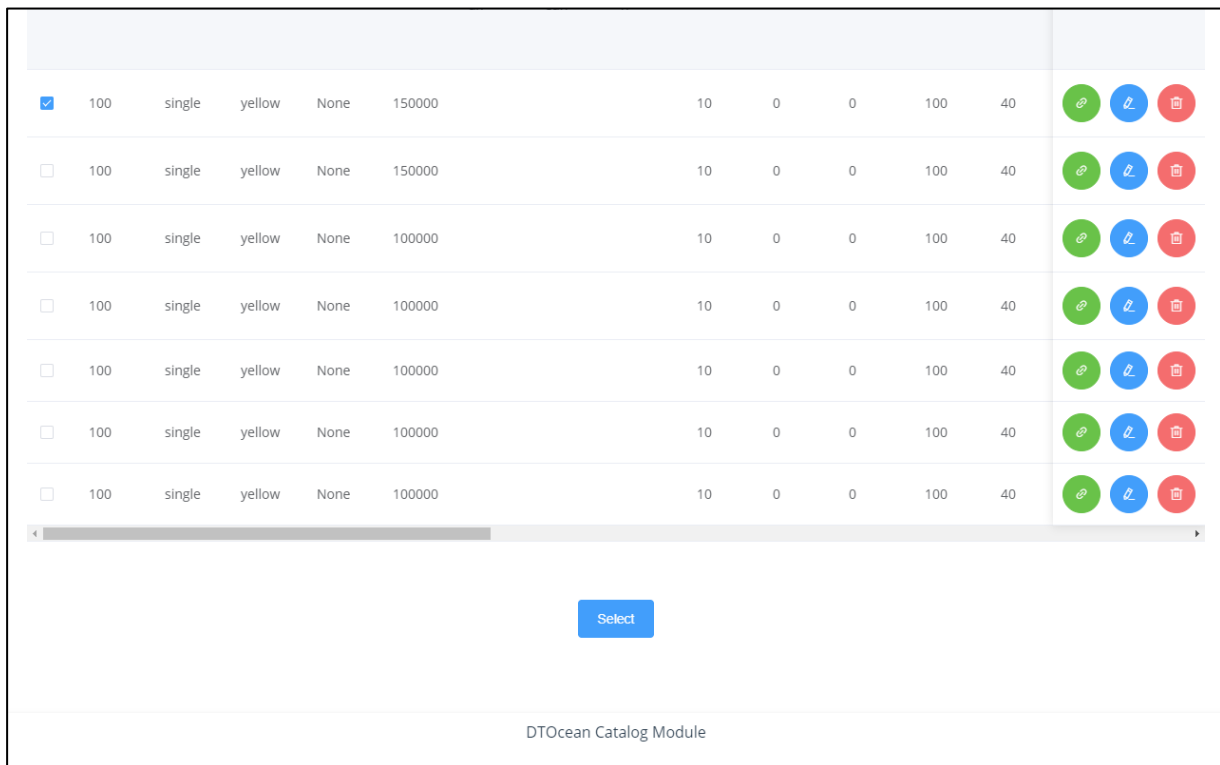


FIGURE 3-6 SELECT A VALUE IN A CATALOGUE

3.3 THE TECHNOLOGIES

The main language used is Python, with the stack of technologies proposed using the following frameworks and libraries: Flask + SQLAlchemy.

The backend of the module manages the database using SQLAlchemy for data modelling and CRUD operations.

The API is documented via OpenAPI, and the GUI uses the web framework Flask to generate URL routes to display the data.

In addition, the open source framework SAFRS is used to automate the generation of the BL and API. Similarly, to automate the GUI, the framework Flask-admin can be used. In more detail:

- Database
 - SQLite if local
 - <https://www.sqlite.org/index.html>
 - PostgreSQL if on server
 - <https://www.postgresql.org/>
- General:
 - Python 3
 - <https://www.python.org/>
 - Flask
 - <https://www.palletsprojects.com/p/flask/>
- Business Layer
 - SQLAlchemy
 - <https://www.sqlalchemy.org/>
 - Marshmallow
 - <https://marshmallow.readthedocs.io/en/stable/>
 - SAFRS
 - <https://github.com/thomaxxl/safrs>
- API
 - OpenApi
 - <https://swagger.io/specification/>
- GUI
 - WTForms
 - Jinja2 (via Flask)
 - HTML
 - Flask-Admin
 - <https://flask-admin.readthedocs.io/en/latest/>
- Testing
 - Pytest
 - <https://docs.pytest.org/en/latest/>



3.4 THE TESTING AND VERIFICATION

3.4.1 Description

Testing is done using Pytest, a testing framework for unit testing and more.

Import and export functionalities are tested to ensure the data is imported into and exported from the database without loss or corruption.

Similarly, CRUD operations are tested. There is a need to ensure that data created or updated are properly inserted in database with data corresponding to users' inputs; that data recovered for read correspond to the requested one; and finally, that deleted data is properly removed as well as cascading related data.

As most catalogues are simple tables and follow the same templates, these tests are made on a template example, while validation is to check that all catalogues implement the template correctly.

In addition, and if a module requires it, specific tests for specific needs of catalogues will be done.

As described in the previous paragraphs, a generic approach for wrapping up XLS files into Catalogues was chosen. This approach includes a generic back-end and front-end services. Each XLS Catalogue could be considered as a separate module/deliverable. So, it could be advanced, distributed and populated on its own. Catalogues will be coupled with other Tools via REST API.

In line with this approach the Catalogues from the following module owners have been processed:

- Energy Delivery (ED) - mock_db_v2.xlsx
- Logistics & Marine Operations (LMO) - catalogues_Imo.xlsx
- Energy Transformation (ET) - ET_Catalogue_202002.xlsx

3.4.2 Backend

3.4.2.1 Unit tests

The generic Catalogue implementation is based on using external frameworks. Therefore its implementation robustness can be implicitly derived by the frameworks it is based upon, namely - flask, sqlalchemy and safrs.

	flask	sqlalchemy	safrs
Downloads per last month (how popular)	13,377,655	33,837	775
Version (how mature)	1.1.1	2.1.0	2.5.5
Last release data (alive?)	08/07/2019	17/07/2019	14/11/2019
Reference/URL	github	github	github



Its quality is guaranteed by the corresponding testing repository - [safrs-example](#).

The Backend quality is provided by unit tests for generating OpenAPI, Pytest of sqlacodegen code and Dredd tests of Catalog OpenAPI endpoints used by other project modules.

3.4.2.2 E2E tests (DREDD)

Energy Delivery: 120 tests

Logistics & Marine Operations: 150 tests

See report on [GitLab](#).

3.4.3 Frontend

3.4.3.1 Unit tests

The following scripts contain the unit tests:

- CatalogBuild.test.js
- CatalogSingleBuild.test.js
- TestComponent.test.js

```
54 PASS __tests__/CatalogBuild.test.js
194 PASS __tests__/CatalogSingleBuild.test.js
195 PASS __tests__/TestComponent.test.js
196 ===== Coverage summary =====
197 Statements : 100% ( 50/50 )
198 Branches   : 100% ( 0/0 )
199 Functions  : 96.15% ( 25/26 )
200 Lines     : 100% ( 50/50 )
201 =====
202 Test Suites: 3 passed, 3 total
203 Tests:      12 passed, 12 total
204 Snapshots:  0 total
205 Time:       3.79s
206 Ran all test suites.
207 Line coverage: 100%
208 $ npm run lint
209 > cat@0.1.0 lint /builds/opencascade/dtocean/sandbox/dtop_catalog_from_to_excel/src/frontend/app/app/cat
210 > vue-cli-service lint
211 DONE No lint errors found!
216 Job succeeded
```

FIGURE 3-7 BACKEND TEST REPORT



3.4.4 E2E tests (Cypress)

End to End tests are run with the script test.spec.js.

The report is available on [GitLab](#).

```
2475 Running: test.spec.js (1 of 1)
2476 integration
2477   ✓ View single catalog (4700ms)
2478   ✓ Edit single catalog (3191ms)
2479   ✓ Delete single catalog (2322ms)
2480   ✓ Checkbox Button (2047ms)
2481 4 passing (13s)
2482 (Results)
2483 ┌───────────────────────────────────────────────────────────────────────────────────┐
2484 │ Tests:      4 │
2485 │ Passing:    4 │
2486 │ Failing:    0 │
2487 │ Pending:    0 │
2488 │ Skipped:    0 │
2489 │ Screenshots: 0 │
2490 │ Video:      true │
2491 │ Duration:   12 seconds │
2492 │ Spec Ran:   test.spec.js │
2493 └───────────────────────────────────────────────────────────────────────────────────┘
2494 (Video)
2495 - Started processing: Compressing to 32 CRF
2496 - Finished processing: /app/cypress/videos/test.spec.js.mp4 (5 seconds)
2497 tput: No value for $TERM and no -T specified
2498 =====
2499 (Run Finished)
2500 Spec                                Tests  Passing  Failing  Pending  Skipped
2501 ┌───────────────────────────────────────────────────────────────────────────────────┐
2502 │ ✓ test.spec.js                                00:12    4      4      -      -      - │
2503 └───────────────────────────────────────────────────────────────────────────────────┘
2504 ✓ All specs passed!                                00:12    4      4      -      -      -
2506 Running after script
2508 Saving cache
2510 Uploading artifacts for successful job
2512 Job succeeded
```

FIGURE 3-8 END TO END TEST REPORT



3.4.5 E2E Application integration tests (Cypress)

For the global integration in the main application, the following test is executed.

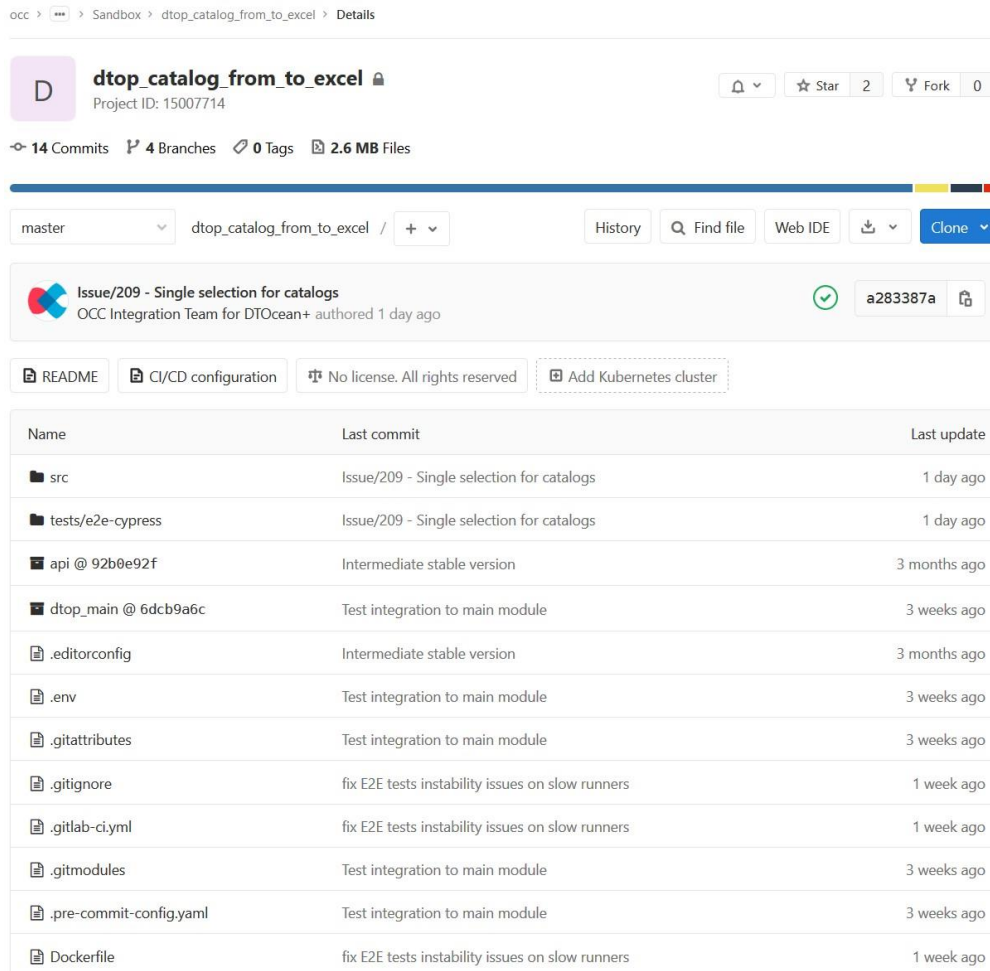
```
3685 Running: test.spec.js (1 of 1)
3686 integration
3687   ✓ View single catalog (5199ms)
3688 1 passing (6s)
3689 (Results)
3690
3691 | Tests:      1
3692 | Passing:    1
3693 | Failing:    0
3694 | Pending:    0
3695 | Skipped:    0
3696 | Screenshots: 0
3697 | Video:      true
3698 | Duration:   6 seconds
3699 | Spec Ran:   test.spec.js
3700
3701 (Video)
3702 - Started processing: Compressing to 32 CRF
3703 - Finished processing: /app/cypress/videos/test.spec.js.mp4 (1 second)
3704 =====
3705 (Run Finished)
3706 tput: No value for $TERM and no -T specified
3707 Spec                                Tests  Passing  Failing  Pending  Skipped
3708 | ✓ test.spec.js                    00:06    1      1      -      -      - |
3709
3710 | ✓ All specs passed!                00:06    1      1      -      -      - |
3711
3716 Job succeeded
```

FIGURE 3-9 APPLICATION INTEGRATION TEST REPORT



3.5 RELEASE

The Catalogue module has been developed with all the main features. The catalogues themselves are operational but not fully populated. The content of each catalogues will be provided by the module developers by the end of the verification phase (beta version). This is not an issue as the chosen solution allows to easily adapt such changes (see 3.1).



The screenshot shows the GitHub repository page for 'dtop_catalog_from_to_excel'. The repository is owned by 'D' and has a Project ID of 15007714. It has 14 commits, 4 branches, 0 tags, and 2.6 MB of files. A recent issue, 'Issue/209 - Single selection for catalogs', is highlighted, authored by the OCC Integration Team for DTOcean+ 1 day ago. The repository includes a README, CI/CD configuration, and a license. A table lists the repository's files and their last commit details.

Name	Last commit	Last update
src	Issue/209 - Single selection for catalogs	1 day ago
tests/e2e-cypress	Issue/209 - Single selection for catalogs	1 day ago
api @ 92b0e92f	Intermediate stable version	3 months ago
dtop_main @ 6dcb9a6c	Test integration to main module	3 weeks ago
.editorconfig	Intermediate stable version	3 months ago
.env	Test integration to main module	3 weeks ago
.gitattributes	Test integration to main module	3 weeks ago
.gitignore	fix E2E tests instability issues on slow runners	1 week ago
.gitlab-ci.yml	fix E2E tests instability issues on slow runners	1 week ago
.gitmodules	Test integration to main module	3 weeks ago
.pre-commit-config.yaml	Test integration to main module	3 weeks ago
Dockerfile	fix E2E tests instability issues on slow runners	1 week ago

FIGURE 3-10 REPOSITORY OF THE CATALOGUE MODULE



4. EXAMPLES

The following catalogues have been identified and implemented in DTOceanPlus. Catalogues are grouped by category, providing one catalogue for each identified table.

These examples are still subject to change during the integration phase, either to better represents the needs of each of the modules, or to correct format issues due to database technology choices.

It can also be possible to add new catalogues or to merge some of them.

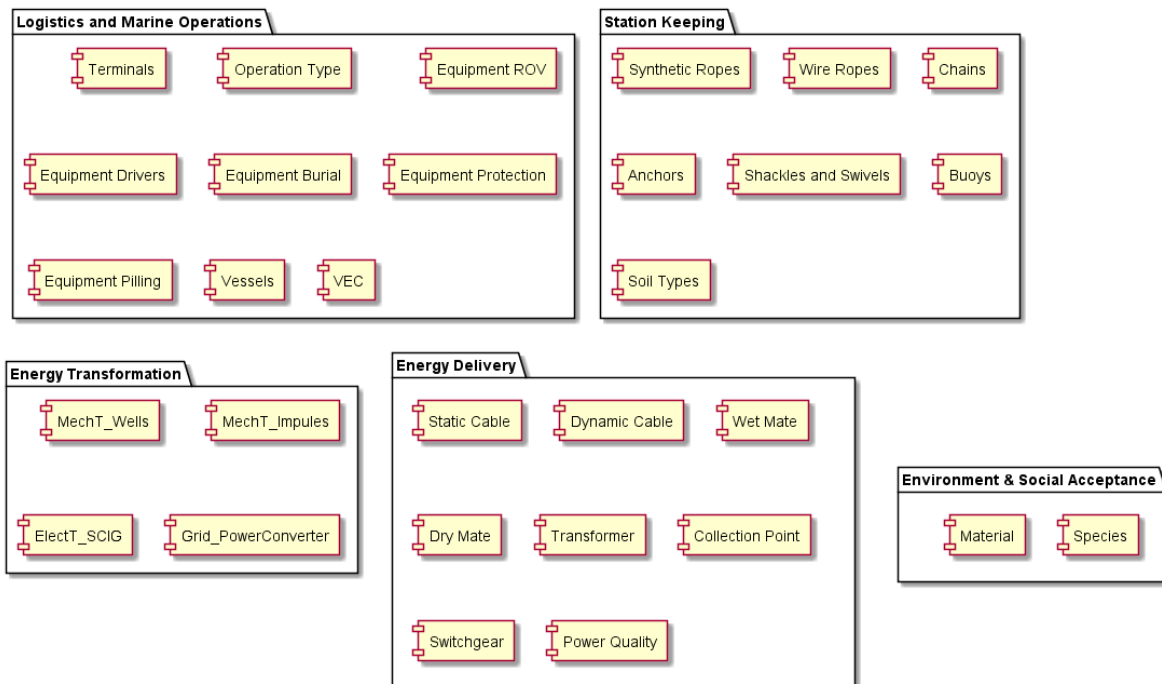


FIGURE 4-1 LIST OF CATALOGUES BY CATEGORIES

4.1 Generic presentation

4.1.1 Logistics and Marine Operations:

- Activities
- Operation Type
- Equipment (Pilling, Protection, Burial, Drivers, ROV)
- Terminals
- Vessels
- Vessel-Equipment Combinations (VECs)

The data will be stored in simple tables, with some need for one-to-one and one-to-many relations.

The Activities catalogue lists marine activities. An activity consists of an operation (from the Operations catalogue) as well as other related attributes such as duration and location.

The Operations catalogue lists marine operations. An operation requires relations to possibly multiple existing VECs.

The other catalogues can be treated as simple tables.

4.1.2 Station Keeping

- Soil type
- Buoys
- Shackles and Swivels
- Anchors
- Chains
- Wire ropes
- Synthetic ropes

The catalogues can be treated as simple tables.

As several catalogues have almost the exact same columns, there is a possibility to merge them all (except for Soil type) into one and/or separating Shackles from Swivels.

4.1.3 Energy Delivery

- Power quality
- Switchgear
- Collection point
- Transformer
- Dry mate
- Wet mate
- Dynamic cable
- Static cable

The catalogues can be treated as simple tables.

There is a possibility to merge some catalogues (Dry and Wet mate, and Dynamic and Static cable). The final tables are not defined yet, as further data could be added and merged into a Connector table.

The Energy Delivery module does not require the user to select directly input from the catalogues, instead the optimal input will be provided by the module. Catalogues are required mainly for management, browsing and export (to CSV primarily) purposes.



4.1.4 Energy Transformation

- Turbines
- Power generators and convertors

The catalogues can be treated as simple tables.

The Energy Transformation module does not necessarily require the user to select directly entries from the catalogues. Depending on the usage of the Energy Transformation module the user will be either provided the optimal input or required to enter a custom entry. Catalogues are required mainly for management, browsing and export purposes.

4.1.5 Environment and Social Acceptance

- Endangered species

The catalogue can be treated as a simple table.

The Environment and Social Acceptance module requires JSON visualisation and export of the data.

4.2 Energy Delivery Example

Using the frameworks SAFRS and Flask-Admin, here is an example based on the data provided as example from the Energy Delivery module.

The data is imported from an excel file via a script which will load the data into a PostgreSQL database, from the data read, the SQLAlchemy model is constructed and the business logic and corresponding API is generated with the SAFRS framework. The web interface GUI is generated using the Flask-Admin framework.

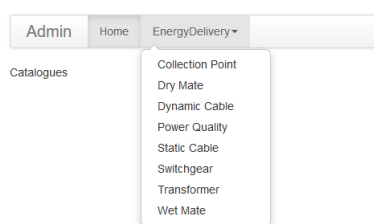


FIGURE 4-2 ENERGY DELIVERY CATALOGUE

Admin Home EnergyDelivery ▾

List (9) Create Export With selected ▾

	Input	Output	Input Connector	Output Connector	V1	V2	A Rate	Dry Mass	Wet Mass	Height	Width	Depth	Colour	Outer Coating	Foundation	Busbar	Cost	Coolir
<input type="checkbox"/>	4	1	wet-mate	dry-mate	6600	11000	100	100	100	10	10	10	yellow	metal	gravity	single	150000	None
<input type="checkbox"/>	4	1	wet-mate	dry-mate	11000	33000	100	100	100	10	10	10	yellow	metal	gravity	single	150000	None
<input type="checkbox"/>	4	1	wet-mate	dry-mate	6600	33000	100	100	100	10	10	10	yellow	metal	gravity	single	150000	None
<input type="checkbox"/>	4	1	wet-mate	dry-mate	6600	6600	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None
<input type="checkbox"/>	4	1	wet-mate	dry-mate	11000	11000	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None
<input type="checkbox"/>	4	1	wet-mate	dry-mate	33000	33000	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None
<input type="checkbox"/>	4	1	dry-mate	dry-mate	690	690	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None
<input type="checkbox"/>	4	1	dry-mate	dry-mate	690	11000	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None
<input type="checkbox"/>	4	1	dry-mate	dry-mate	690	33000	100	100	100	10	10	10	yellow	metal	gravity	single	100000	None

FIGURE 4-3 COLLECTION POINT



5. CONCLUDING REMARKS AND FUTURE WORK

5.1 Conclusion

This deliverable captures the main functional and technical aspects of the Catalogues module. While the module can be run in a standalone mode its added value comes from being connected to other modules to provide input selection. As standalone the module provides complete management functionalities of the data in a user-friendly interface. The goal is to provide a centralised and stable point of entry for various types of data.

5.2 Future work

The current implementation allows to showcase the main operational functionalities and provides an easily configurable solution which can rapidly adapt to catalogue format change during the integration phase. The module is equipped to quickly integrate new catalogues that might be provided (most of them are to be finalized into their final form), in a modular and scalable way.





CONTACT DETAILS

Mr. Pablo Ruiz-Minguela
Project Coordinator, TECNALIA
www.dtoceanplus.eu



Naval Energies terminated its participation on 31st August 2018 and
EDF terminated its participation on 31st January 2019.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 785921