|  |  |  |
| --- | --- | --- |
| Customer | EEA | |
| Project name | SC 57032  New website for the Reporting Obligations Database (ROD) - JAVA MVC Development | |
| Document | Technical annex |  |

|  |  |
| --- | --- |
| Date and version | Signature |
| April 2018 |  |

***Table of contents***

[**Architecture for ROD3** 5](#_Toc511214671)

[Modules 6](#_Toc511214672)

[Home module 6](#_Toc511214673)

[Search reporting obligations: 9](#_Toc511214674)

[Search in ROD website 10](#_Toc511214675)

[Latest news 13](#_Toc511214676)

[Countries/territories module 14](#_Toc511214677)

[View deadlines of one country 16](#_Toc511214678)

[Filter a search of deadlines 18](#_Toc511214679)

[View obligation of a deadline 21](#_Toc511214680)

[View client of a deadline 21](#_Toc511214681)

[View contacts of a deadline 21](#_Toc511214682)

[View deliveries of a deadline 23](#_Toc511214683)

[Obligations module 27](#_Toc511214684)

[Filter search of obligations 29](#_Toc511214685)

[Obligation overview 32](#_Toc511214686)

[Obligation legislation 34](#_Toc511214687)

[Obligation deliveries 36](#_Toc511214688)

[Obligation history 38](#_Toc511214689)

[View instrument of one obligation 40](#_Toc511214690)

[***View client of one obligation*** 40](#_Toc511214691)

[Add new obligation (previously logged) 41](#_Toc511214692)

[Delete various obligations (previously logged) 46](#_Toc511214693)

[Edit obligation (previously logged) 48](#_Toc511214694)

[Delete obligation (previously logged) 51](#_Toc511214695)

[Clients module 53](#_Toc511214696)

[View client 55](#_Toc511214697)

[Add client (previously logged) 58](#_Toc511214698)

[Delete various clients (previously logged) 61](#_Toc511214699)

[Edit client (previously logged) 63](#_Toc511214700)

[Delete obligation (previously logged) 66](#_Toc511214701)

[Global History module 68](#_Toc511214702)

[See the changes of previous version 70](#_Toc511214703)

[Instruments module 75](#_Toc511214704)

[View instrument 77](#_Toc511214705)

[Add instrument (previously logged) 80](#_Toc511214706)

[Edit instrument (previously logged) 83](#_Toc511214707)

[Delete instrument (previously logged) 86](#_Toc511214708)

[Core data flows module 87](#_Toc511214709)

[EEA data flows module 90](#_Toc511214710)

[Database statistics module 92](#_Toc511214711)

[Show obligations with EEA\_CORE=1 94](#_Toc511214712)

[Show obligations with EEA\_PRIMARY=1 97](#_Toc511214713)

[Show obligations with FLAGGED=1 97](#_Toc511214714)

[Show most recent updates 98](#_Toc511214715)

[Show obligations without issues 101](#_Toc511214716)

[Advanced search 104](#_Toc511214717)

[Help module 107](#_Toc511214718)

[RSS XML/RCP data extraction help 109](#_Toc511214719)

[Disclaimer 111](#_Toc511214720)

[Harvester 113](#_Toc511214721)

[Harvest 115](#_Toc511214722)

[Data base model 115](#_Toc511214723)

[117](#_Toc511214724)

[UML diagram 117](#_Toc511214725)

[Data model Description 118](#_Toc511214726)

[HLP\_AREA 118](#_Toc511214727)

[T\_CLIENT 119](#_Toc511214728)

[T\_CLIENT\_OBLIGATION\_LNK 120](#_Toc511214729)

[T\_CLIENT\_SOURCE\_LNK 121](#_Toc511214730)

[T\_DELIVERY 121](#_Toc511214731)

[T\_ISSUE 123](#_Toc511214732)

[T\_SAPATIAL 124](#_Toc511214733)

[T\_OBLIGATION 125](#_Toc511214734)

[T\_RASSPATIAL\_LNK 132](#_Toc511214735)

[T\_RAISSUE\_LNK 132](#_Toc511214736)

[T\_ROLE 133](#_Toc511214737)

[T\_UNDO 134](#_Toc511214738)

[T\_OBLIGATION\_RELATION 135](#_Toc511214739)

[T\_SOURCE\_LNK 136](#_Toc511214740)

[T\_SOURCE\_CLASS 137](#_Toc511214741)

[T\_SOURCE 138](#_Toc511214742)

# **Architecture for ROD3**

Figure 1

# Modules

The new application, available in the development environment at EEA[[1]](#footnote-1), gathers all modules and options contemplated in current ROD[[2]](#footnote-2) application. Each module contains backend and frontend sections as described in the following paragraphs.

## Home module

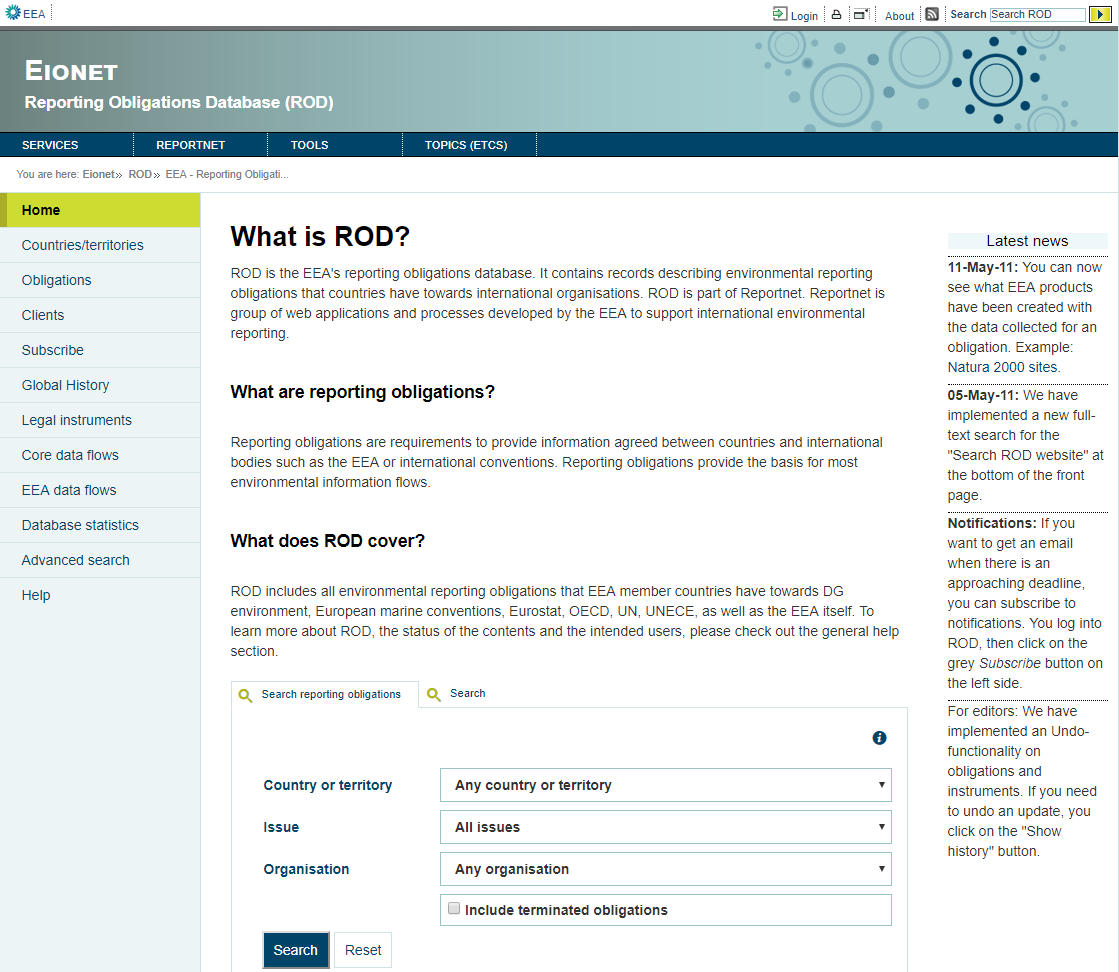


Figure 2

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService which calls the SpatialDao to use its methods.
  + Data Access Layer (Using Data Access Object - DAO):
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - HelpDao: Used to get the information of Latest news.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
    - Documentation
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.SimplePageController
  + View: index.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + actions:
    - Search reporting obligations
    - Search in ROD website

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_issue
  + t\_client
  + t\_spatial
  + hlp\_area

### Search reporting obligations:

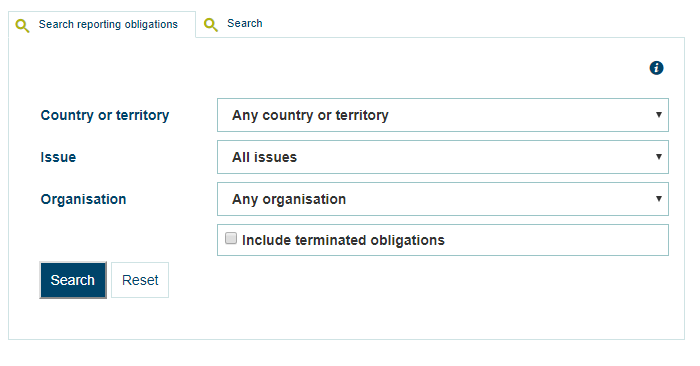


Figure 3

This first tab “Search reporting obligations” the user can search obligations by country/territory, issue and organization including also terminated obligations. Once the user clicks on the “Search” button the application goes to /obligations/search presenting the obligations (Figure 4) with the Country/Issue/Organisation selected.



Figure 4

### Search in ROD website

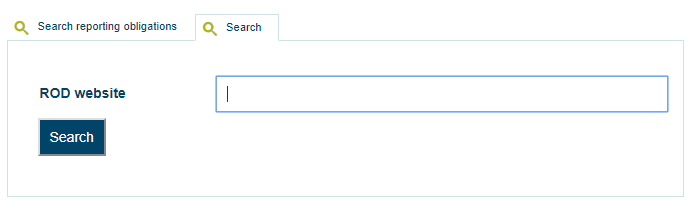


Figure 5

Selecting the second tab “Search” and clicking in the Go button the application goes to /simpleSearch?expression=”search” presenting all obligations, instruments and clients (Figure 6) with the searched word in their title or description.

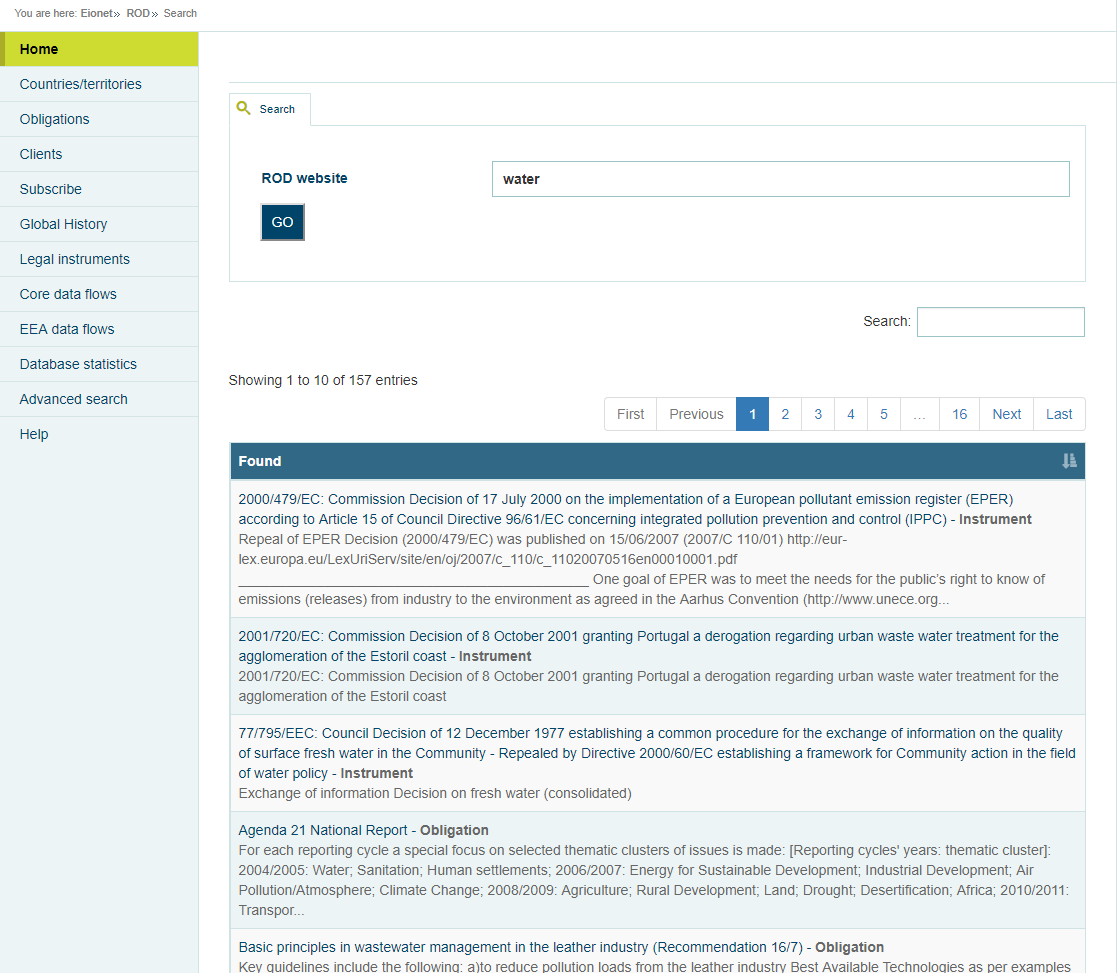


Figure 6

The search result it is done through SPARQL, detailed below:

Sparql url: http://cr.eionet.europa.eu/sparql

Sentence:

PREFIX rod: <http://rod.eionet.europa.eu/schema.rdf#>

PREFIX dct: <http://purl.org/dc/terms/>

SELECT DISTINCT ?subject ?type ?found ?name

FROM <http://rod.eionet.europa.eu/obligations/rdf>

FROM <http://rod.eionet.europa.eu/instruments/rdf>

FROM <http://rod.eionet.europa.eu/clients/rdf>

WHERE {

?subject a ?type . FILTER (?type IN (rod:Instrument, rod:Obligation, rod:Client))

?subject ?p ?found . ?found bif:contains \"'"+expression+"'\" .

OPTIONAL { { ?subject dct:title ?name } UNION { ?subject foaf:name ?name } }

}

### Latest news

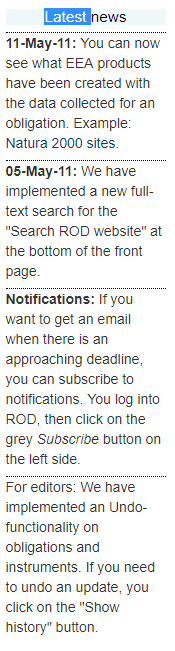
****

Figure 7

Getting the information from the HLP\_AREA table, this section presents the latest news in the index page.

## Countries/territories module

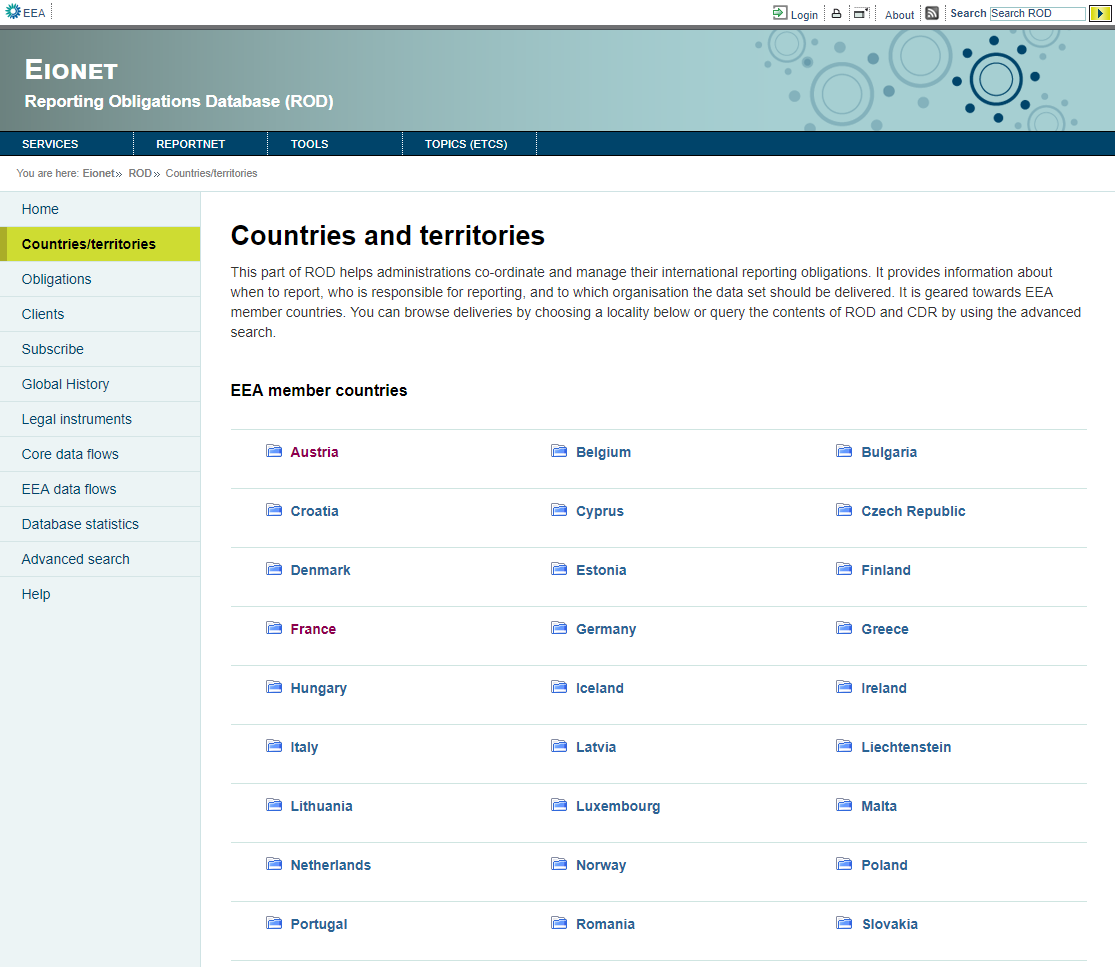


Figure 8

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService which calls the SpatialDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO):
    - SpatialDao: Used to get all countries.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.SpatialController
  + Dao: eionet.rod.dao.SpatialDao
  + Service: eionet.rod.service.SpatialService
  + Model: eionet.rod.model.Spatial
  + View: spatial.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + Actions:
    - View deadlines of one country
    - Filter a search of deadlines

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_spatial

### View deadlines of one country

When the user clicks in one country the application goes to /spatial/”spatialId”/deadlines (figure 9).

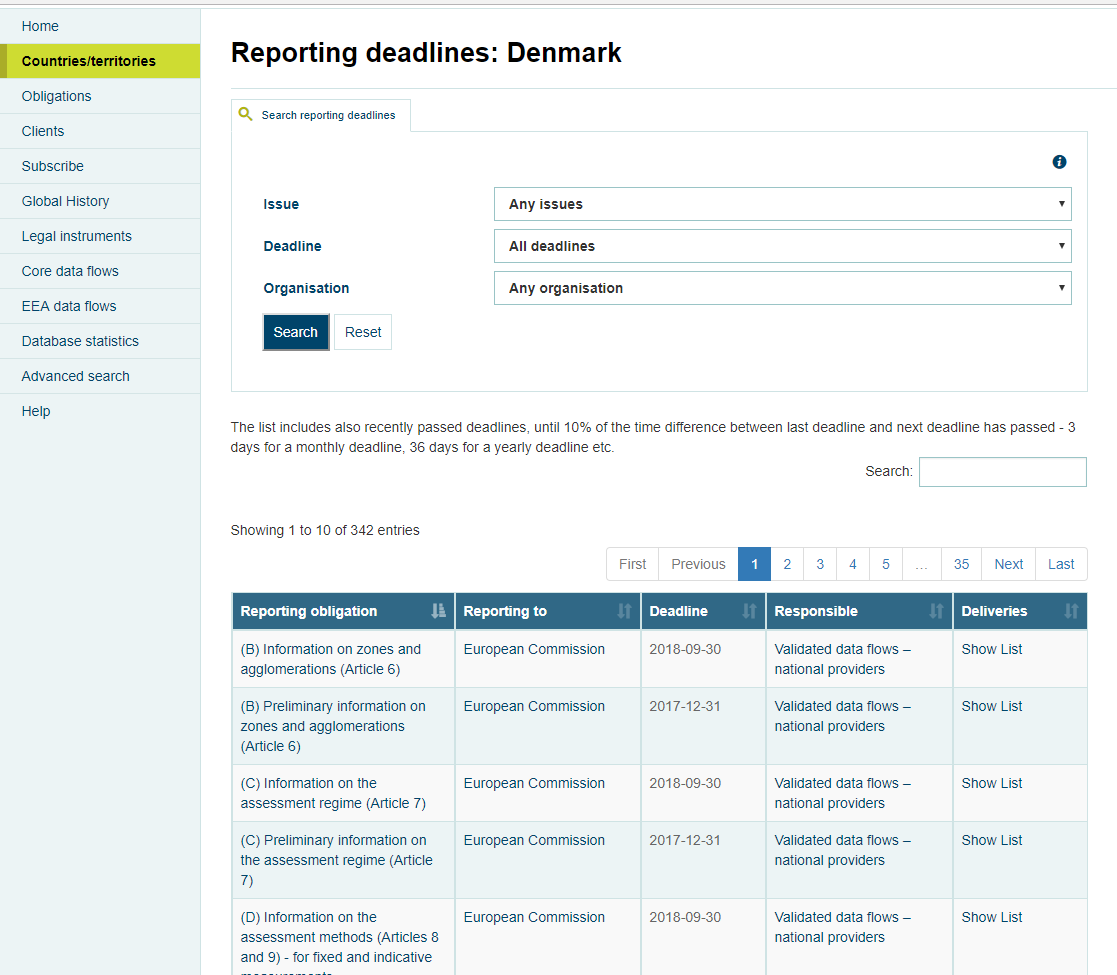


Figure 9

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService which calls the SpatialDao to use his methods.
    - ObligationService which calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO):
    - SpatialDao: Used to get the name of the country.
    - IssueDao: Used to get all issues and charge the Issue combo.
    - ClientService: Used to get all clients and charge the Organisation combo.
    - ObligationDao: Used to get the obligations list that this country reported.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Obligations
    - ClientDTO
    - Issue
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.SpatialController
  + Dao: eionet.rod.dao.SpatialDao
  + Service: eionet.rod.service.SpatialService
  + Model: eionet.rod.model.Spatial
  + View: deadlines.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_spatial
  + t\_issue
  + t\_client
  + t\_obligation
  + t\_source
  + t\_client\_obligation\_lnk
  + t\_raspatial\_lnk
  + t\_role
  + t\_raissue\_lnk

### Filter a search of deadlines

With this option the page charges the list of deadlines filtered by Issue, Deadline and Organisation selected (Figure 10).

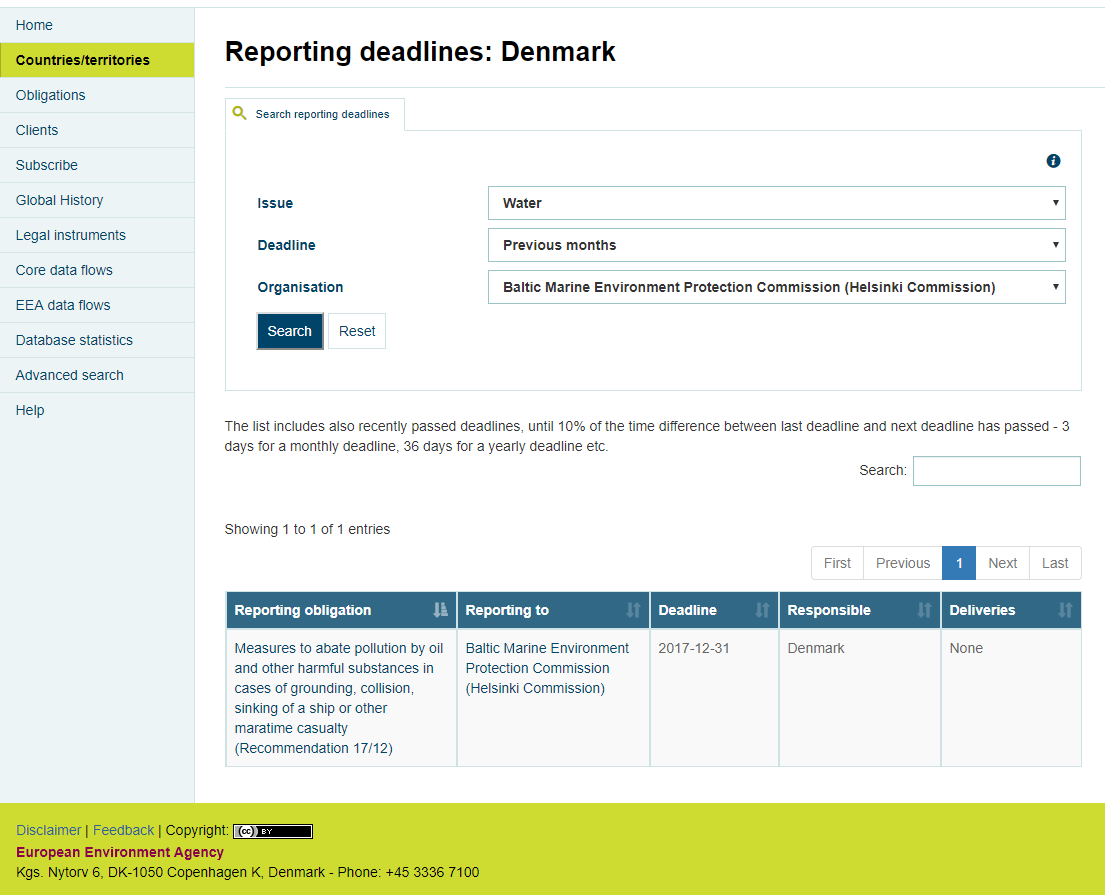


Figure 10

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService which calls the SpatialDao to use his methods.
    - ObligationService which calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO):
    - SpatialDao: Used to get the name of the country.
    - IssueDao: Used to get all issues and charge the Issue combo.
    - ClientService: Used to get all clients and charge the Organisation combo.
    - ObligationDao: Used to get the obligations list that this country reported with the issue, deadline and organisation selected in the search.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Obligations
    - ClientDTO
    - Issue
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.SpatialController
  + Dao: eionet.rod.dao.SpatialDao
  + Service: eionet.rod.service.SpatialService
  + Model: eionet.rod.model.Spatial
  + View: deadlines.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_spatial
  + t\_issue
  + t\_client
  + t\_obligation
  + t\_source
  + t\_client\_obligation\_lnk
  + t\_raspatial\_lnk
  + t\_role
  + t\_raissue\_lnk

### View obligation of a deadline

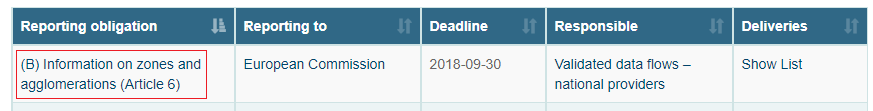
****

Figure 11

Clicking on a reporting obligation of one deadline, the application goes to /obligations/”obligationId” (Figure 21).

### View client of a deadline

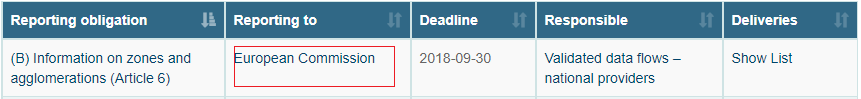
****

Figure 12

Clicking on the client of one deadline, the application goes to /clients/”clientId” (Figure 36).

### View contacts of a deadline

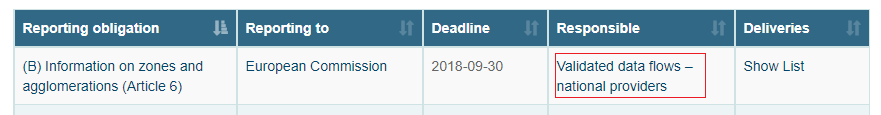


Figure 13

Clicking on the responsible of one deadline, the application goes to /contacts?roleId=”roleId” (Figure 14).

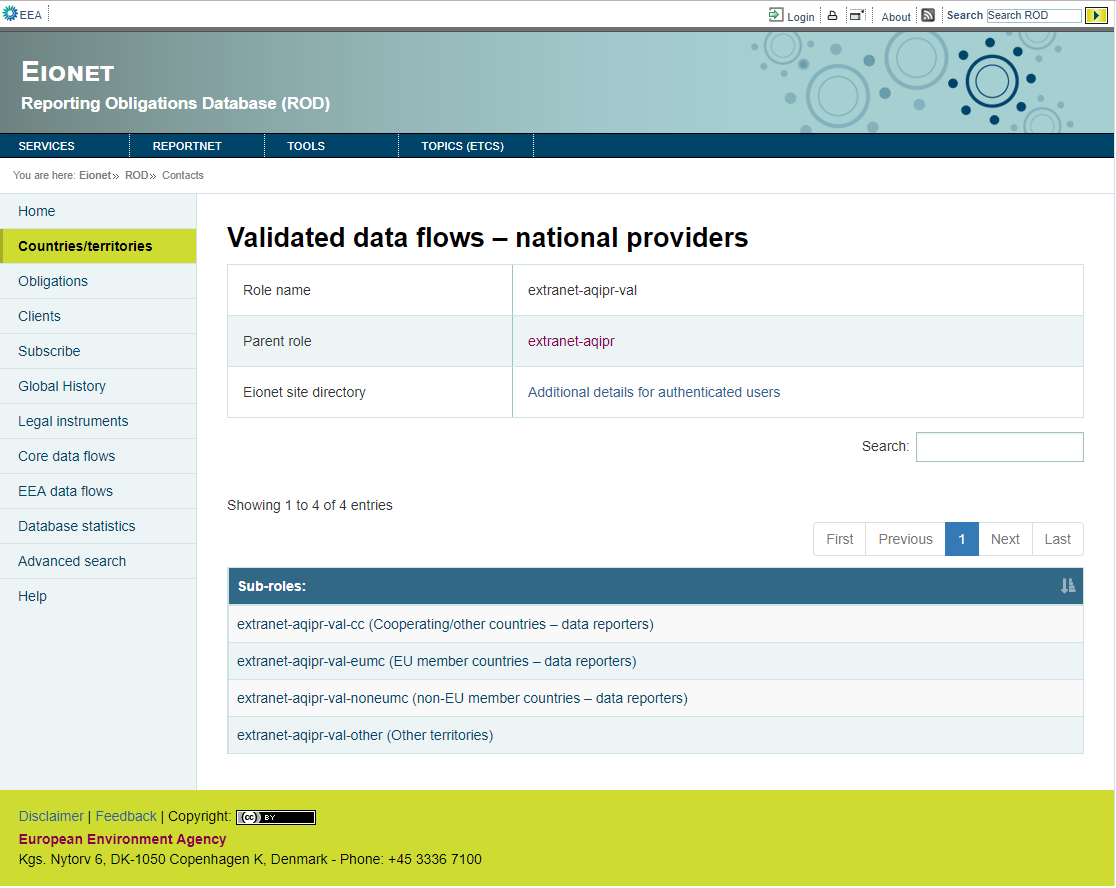


Figure 14

The contacts page shows the sub roles (Figure 14) or members (Figure 15) of a role.



Figure 15

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - DirectoryService: Class of eionet-dir library used to get the subroles or members of a role.
  + Data Access Layer (Using Data Access Object - DAO):
  + Data Model Layer (Using Data Transfer Object – DTO):
    - RoleDTO

* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ContactsController
  + View: contacts.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* This module doesn’t use tables of the database.

### View deliveries of a deadline

Clicking in “Show list” of one deadline, the application goes to countrydeliveries?actDetailsId=”obligationId”&spatialId=”spatialId” (Figure 17).

Deadlines/Deliveries part of ROD3, formerly called Country Services, is a tool that helps member countries to co-ordinate and to supervise their international reporting. It does that by gathering and presenting information from various sources (without containing information in itself), helping the user to track the status of ongoing/future reporting processes. In the current version, it gives an overview of the following:

• Which country had/has to report what?

• When was/is the deadline?

• When is the deadline for the next deadline?

• Is the reported data set in Reportnet?

• Who was responsible for delivering?

• To who was the data set delivered?

Since Deliveries needs a large amount of information from ROD, it is designed as a part of it.

The process starts by selecting ‘Deadlines’ from the left-hand pane of ROD. The window shown in Figure 8 opens

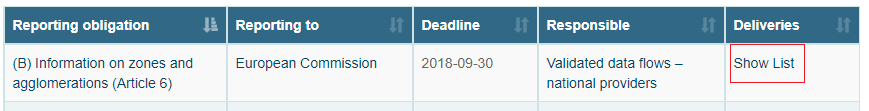


Figure 16

The countrydeliveries page shows the deliveries of the selected country and obligation.

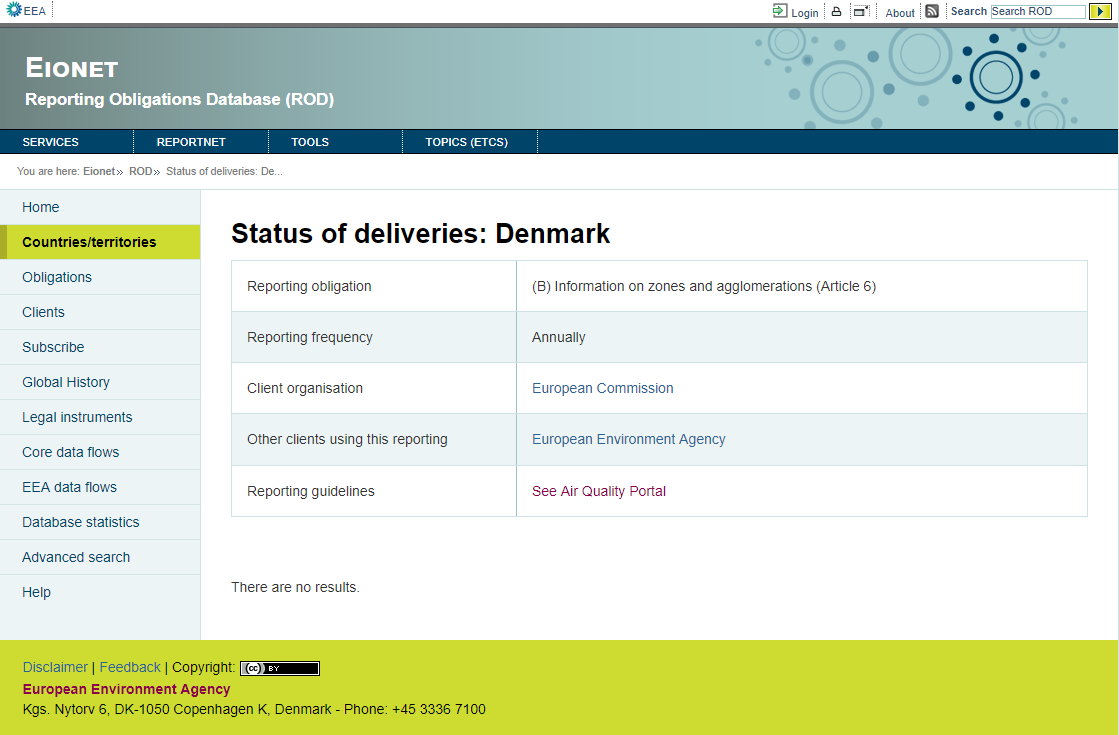


Figure 17

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: It calls the SpatialDao to use his methods.
    - ObligationService: It calls the ObligationsDao to use his methods.
    - DeliveryService: It calls the DeliveryDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO):
    - SpatialDao: Used to get the name of the country.
    - ClientService: Used to get clients of the obligation.
    - ObligationDao: Used to get information of the obligation.
    - DeliveryDao: Used to get all deliveries of this obligation and country.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Obligations
    - ClientDTO
    - Delivery
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.DeliveryController
  + Dao: eionet.rod.dao.DeliveryDao
  + Service: eionet.rod.service.DeliveryService
  + Model: eionet.rod.model.Delivery
  + View: countrydeliveries.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_delivery
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source

## Obligations module

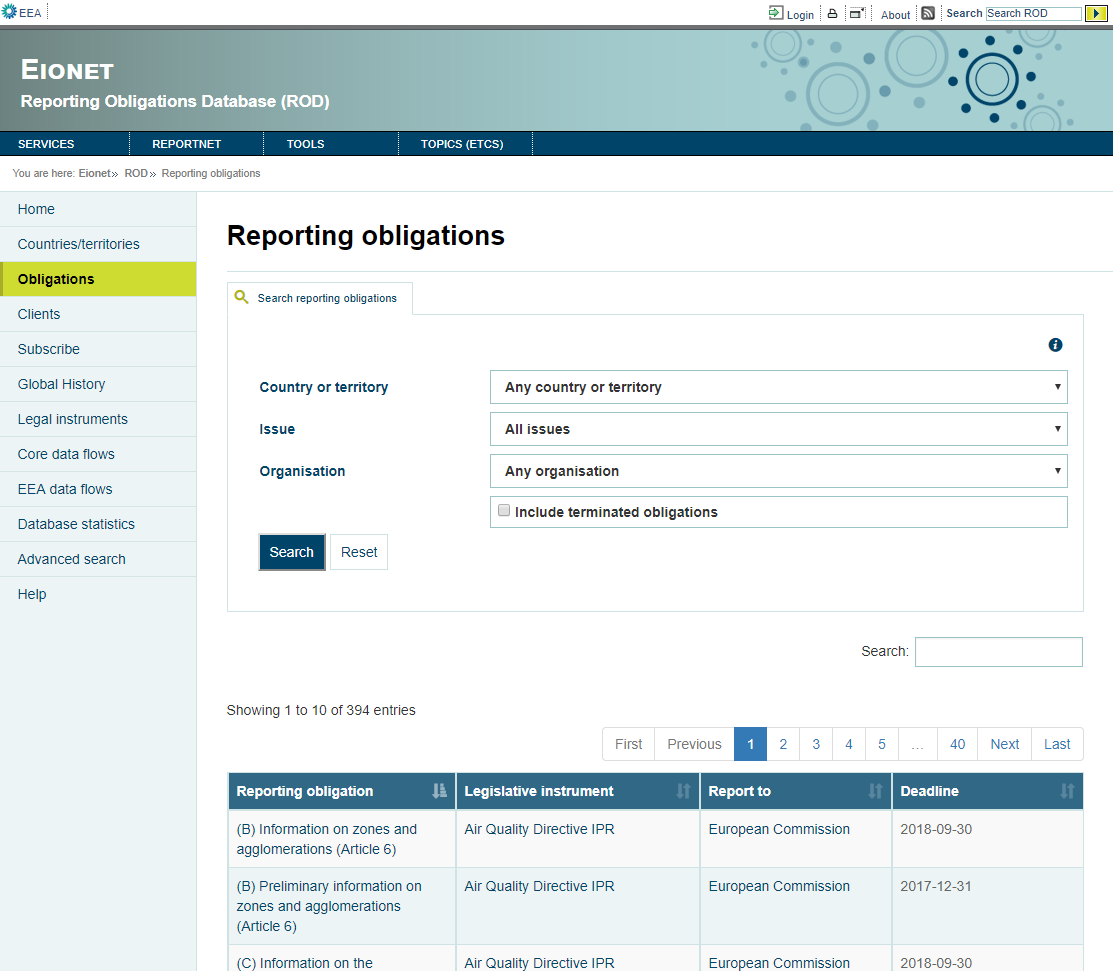


Figure 18

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: It calls the SpatialDao to use his methods.
    - ObligationService: It calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + actions:
    - Filter search of obligations
    - Obligation overview
    - View instrument of obligation
    - View client of obligation
    - Add new obligation (previously logged)
    - Delete various obligations (previously logged)
    - Edit obligation (previously logged)
    - Delete one obligation (previously logged)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

### Filter search of obligations

Clicking the button Search the application goes to /obligations/search and the page charges the obligations with the Issue, Country and Organisation selected (Figure 19).

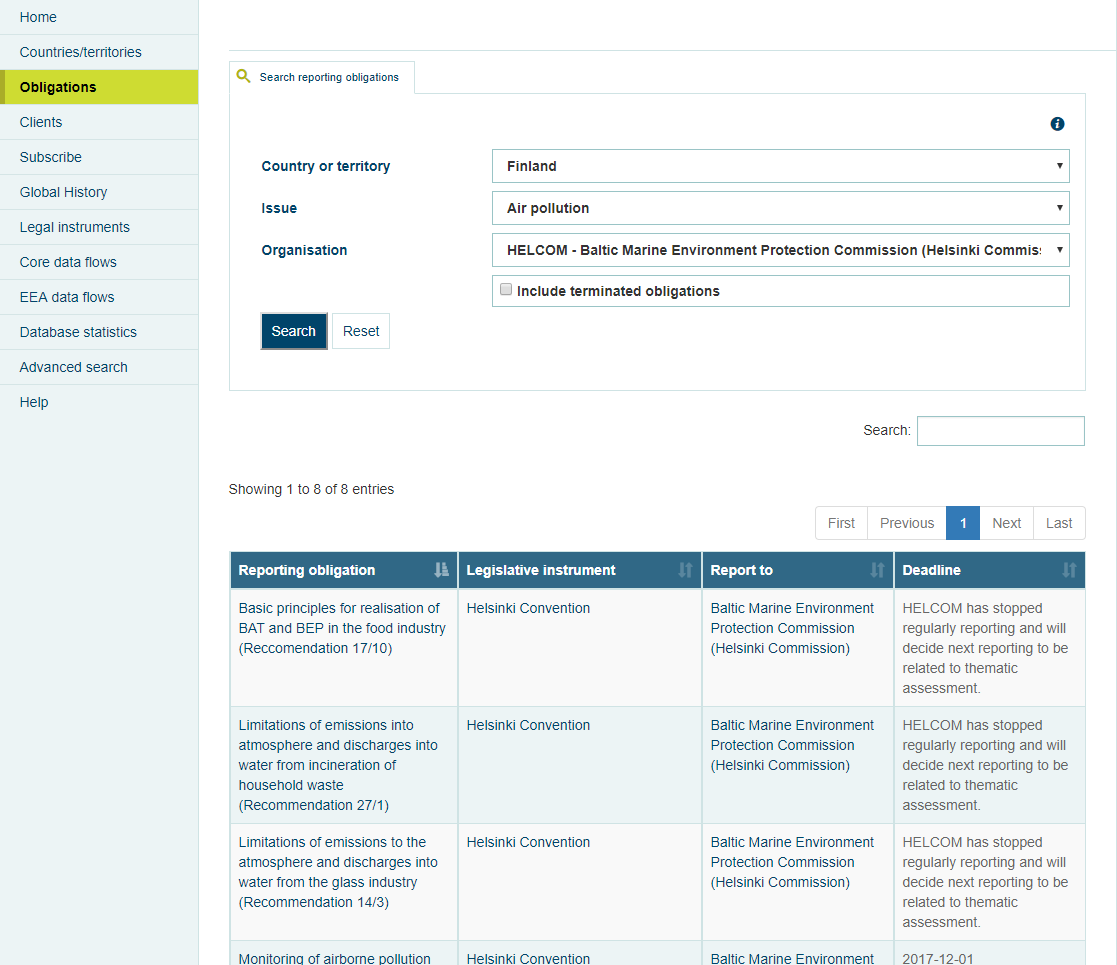


Figure 19

Backend specifications

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: Calls the SpatialDao to use his methods.
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get the searched obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

### Obligation overview



Figure 20

Clicking in one obligation goes to /obligations/”obligationId” (Figure 21).

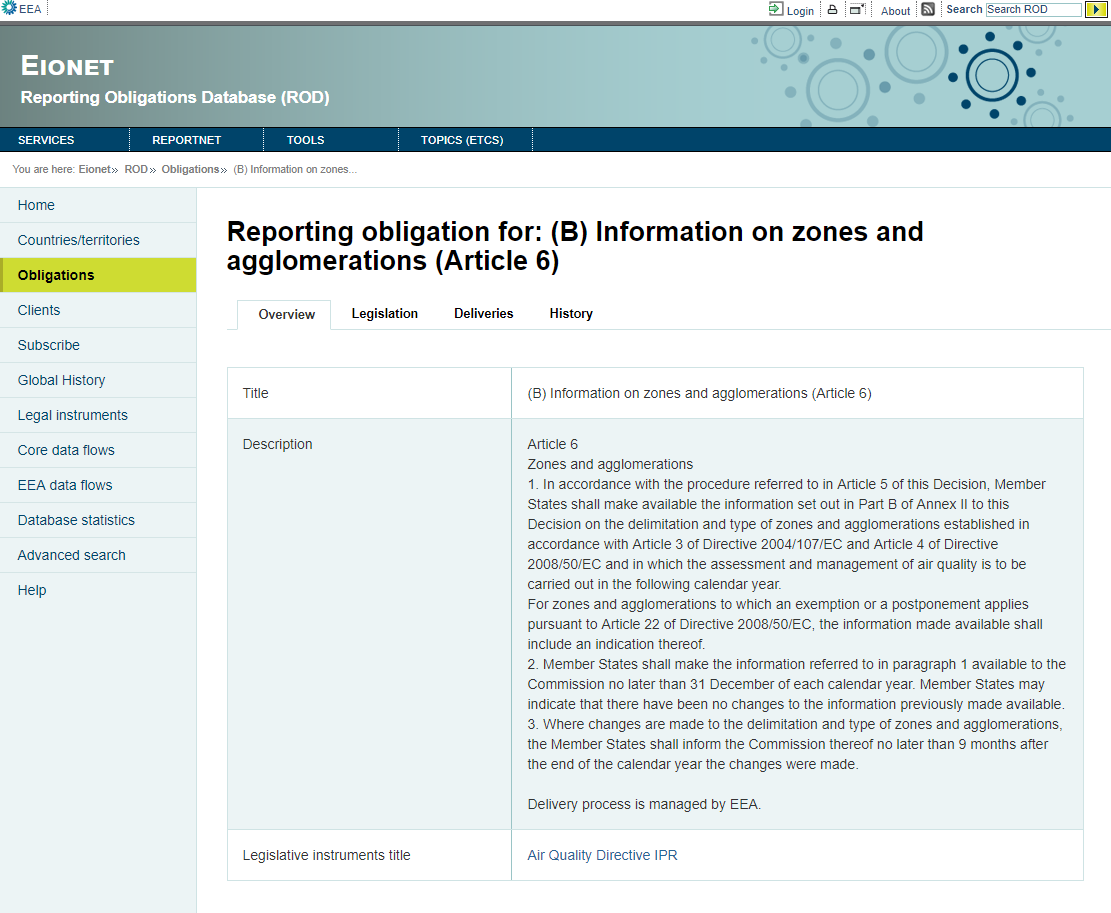


Figure 21

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - ClientService: Used to get the clients of the obligation.
    - ObligationsDao: Used to get the information of obligation and the relation with other obligation.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligation\_overview.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_source
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_obligation\_relation

### Obligation legislation

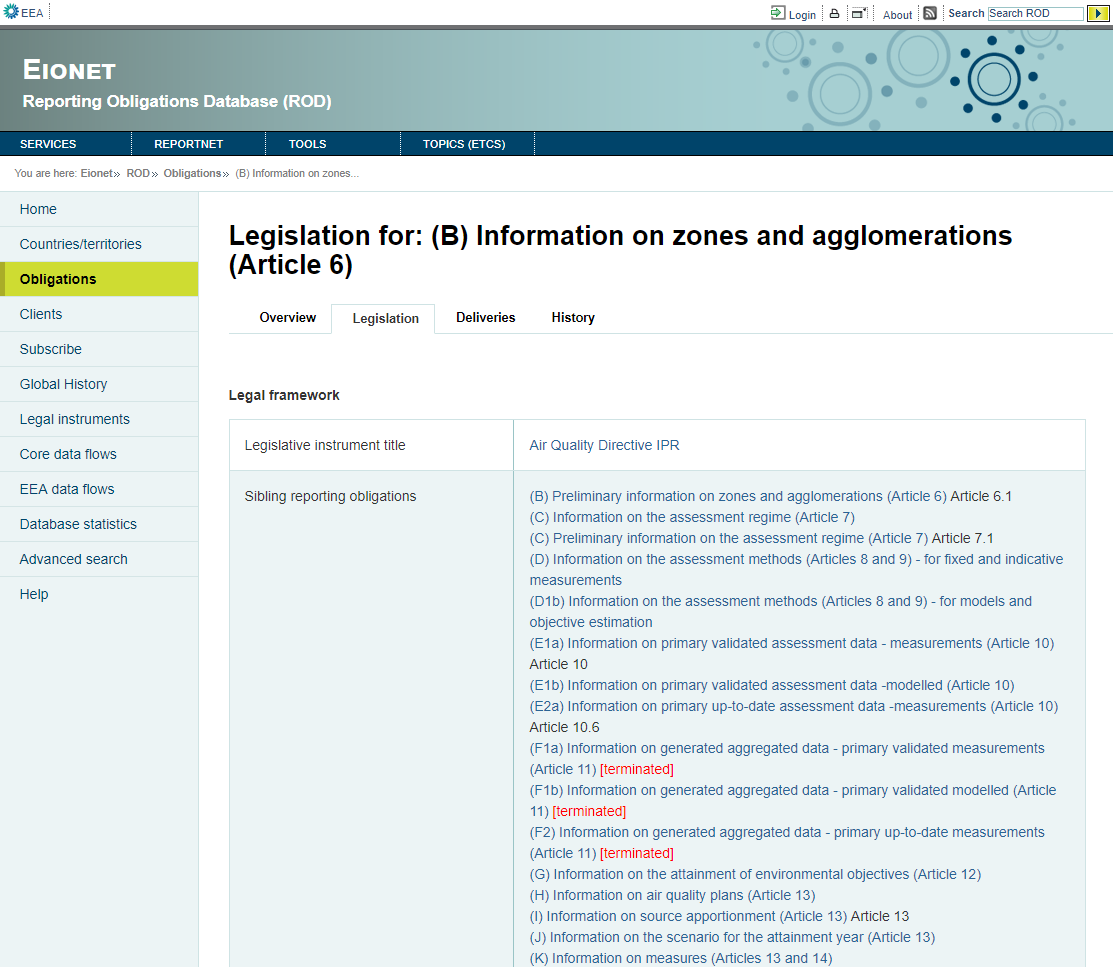


Figure 22

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
    - SpatialService: Calls the SpatialDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - SpatialDao: Used to get the countries of the obligation.
    - ObligationsDao: Used to get the information of obligation and the sibling obligations.
    - IssueDao: Used to get issues of obligation.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ObligationCountry
    - Obligations
    - Issue
    - SiblingObligation
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligation\_legislation.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_source
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_obligation\_relation
  + t\_issue
  + t\_raissue\_lnk
  + t\_spatial
  + t\_raspatial\_lnk

### Obligation deliveries



Figure 23

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
    - DeliveryService: Calls the DeliveryDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - DeliveryDao: Used to get the deliveries of the obligation.
    - ObligationsDao: Used to get the information of the obligation.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Obligations
    - Delivery
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligation\_deliveries.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_source
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_delivery
  + t\_spatial

### Obligation history

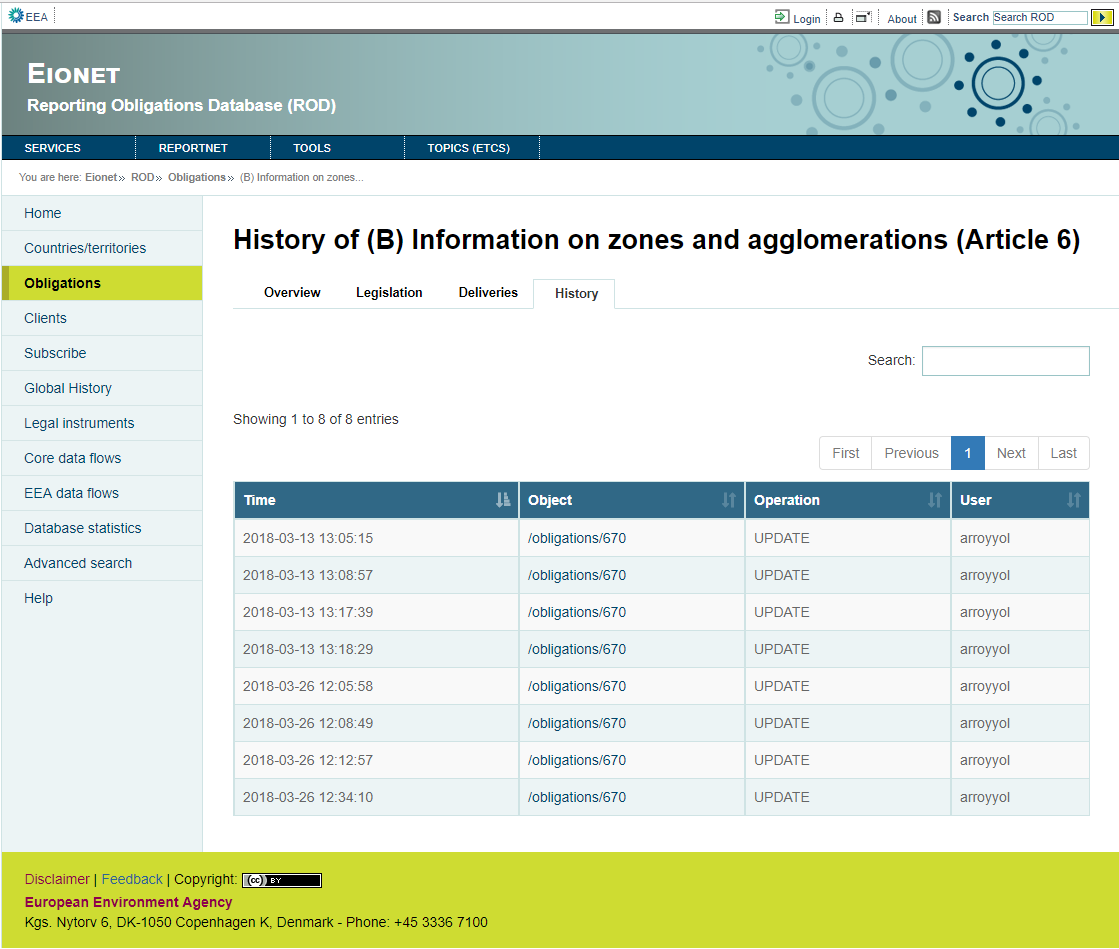


Figure 24

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - UndoService: Used to get the updates of obligation.
    - ObligationsDao: Used to get information of the obligation.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Obligations
    - UndoDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligation\_history.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_source
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_undo

### View instrument of one obligation



Figure 25

Clicking the instrument goes to /instrument/”instrumentId” (Figure 46).

### ***View client of one obligation***

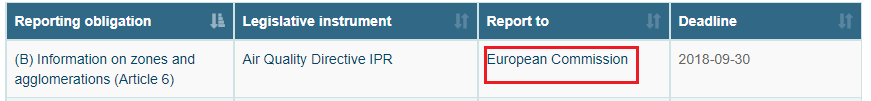


Figure 26

Clicking in the client goes to /clients/”clientId” (Figure 33).

### Add new obligation (previously logged)

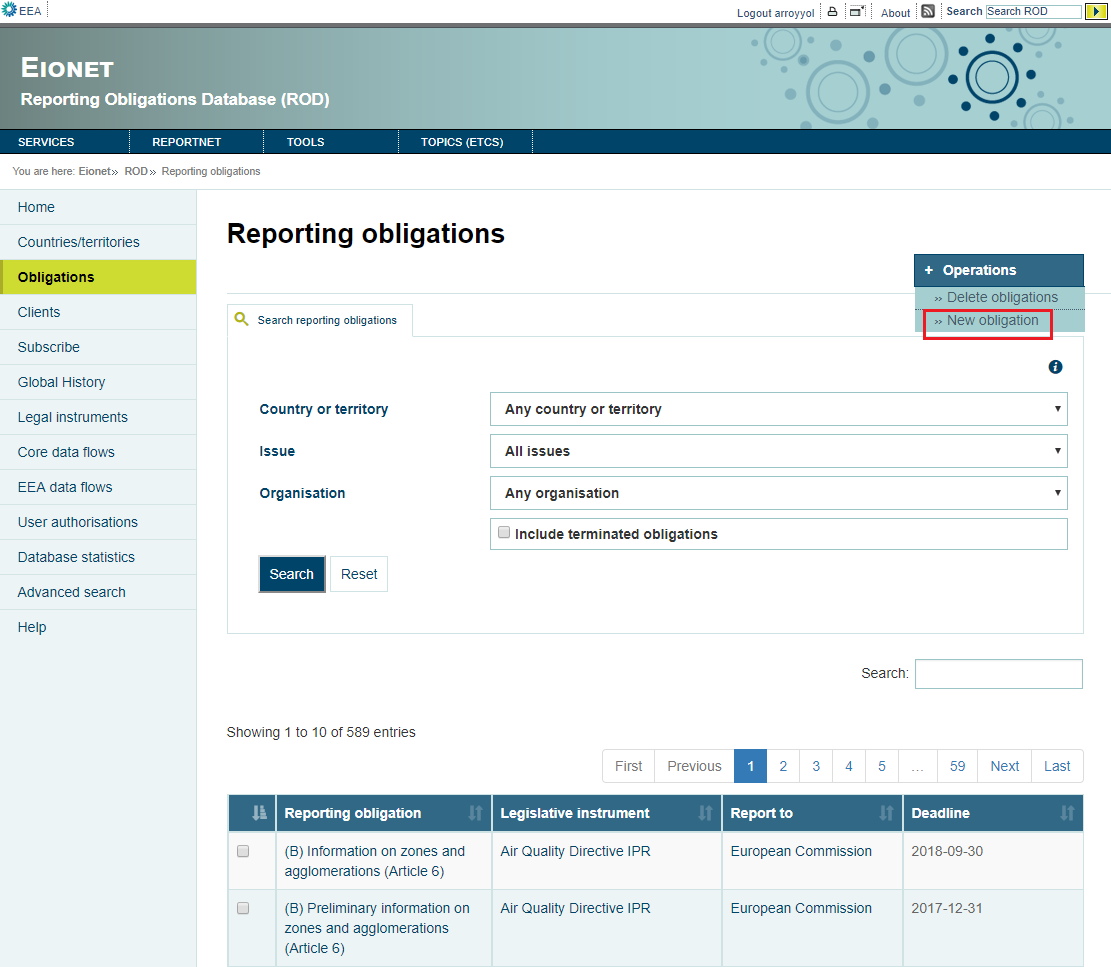


Figure 27

Going to Operations – New Obligation goes to /instruments?mode=X (Figure 28).

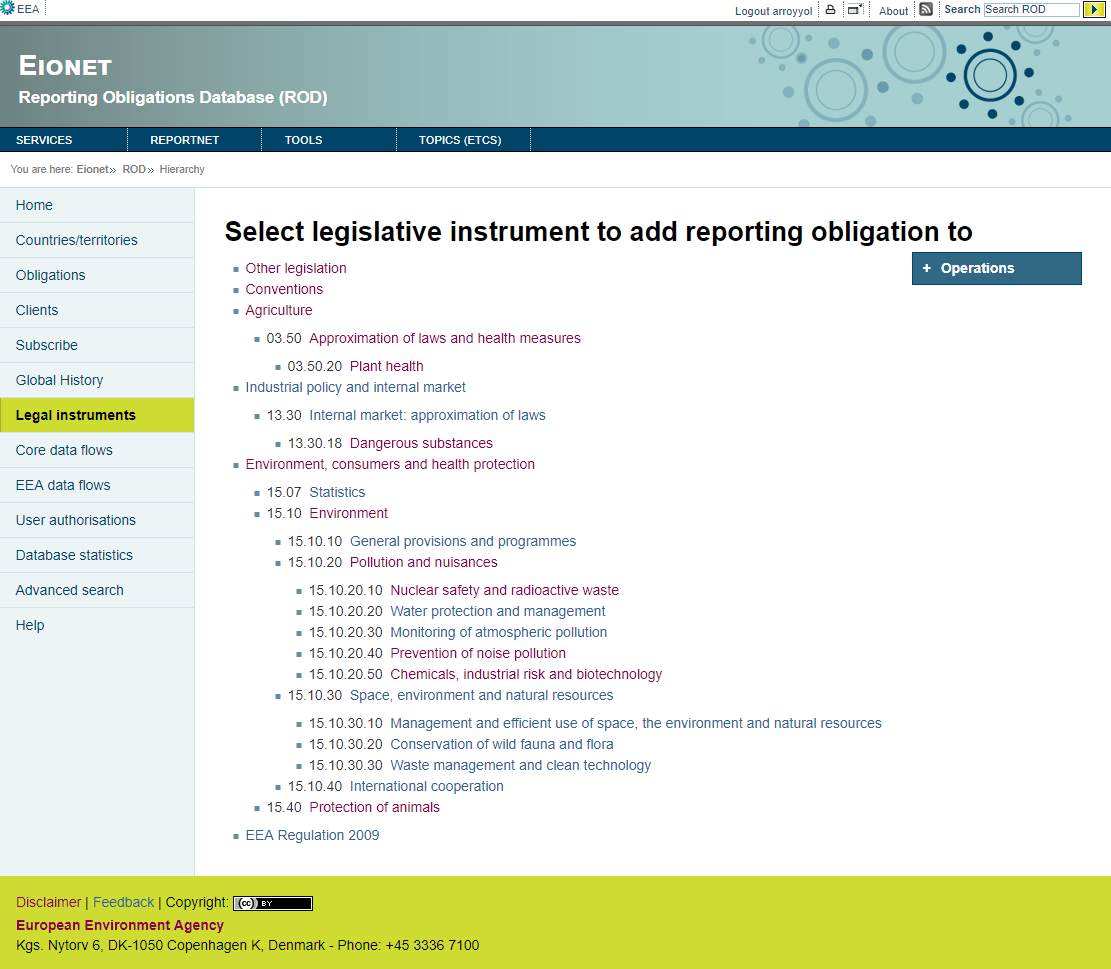


Figure 28

Selecting the instrument to add obligation goes to /obligations/add/”instrumentId”.

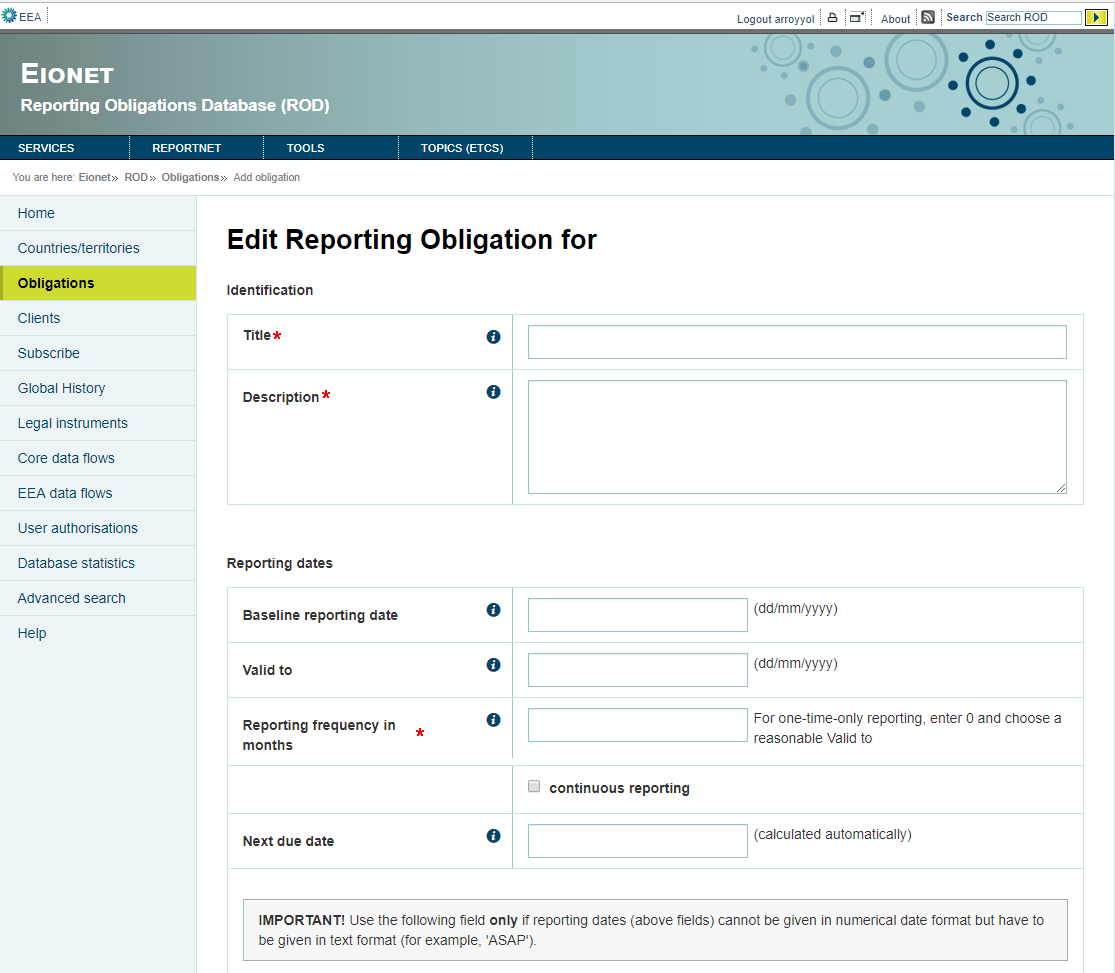


Figure 29

Filling the fields with the data of the new obligation and clicking Add button, the new obligation is saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls ObligationsDao to use his methods.
    - SpatialService: Calls SpatialDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - SourceService: Used to get hierarchy of the instrument and get hierarchy tree.
    - ObligationsDao: Used to insert a new obligation.
    - SpatialDao: Used to get all countries.
    - IssueDao: Used to get all issues.
    - ClientService: Used to get all clients.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - InstrumentListDTO
    - HierarchyInstrumentDTO
    - Obligations
    - ClientDTO
    - Issue
    - Spatial
* Folder structure inside Java project:
  + Controller:
    - eionet.rod.controller.InstrumentController
    - eionet.rod.controller.ObligationsController
  + Service:
    - eionet.rod.service.ObligationService
  + Dao:
    - eionet.rod.dao.SourceService
    - eionet.rod.dao.ObligationsDao
  + Model:
    - eionet.rod.model.HierarchyTreeInstrument
    - eionet.rod.model.InstrumentListDTO
    - eionet.rod.model.Obligations
  + View:
    - instrumentsx.html
    - eobligation.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source\_lnk
  + t\_source\_class
  + t\_source
  + t\_obligation
  + t\_client
  + t\_spatial
  + t\_issue
  + t\_client\_obligation\_lnk
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_obligation\_relation

### Delete various obligations (previously logged)

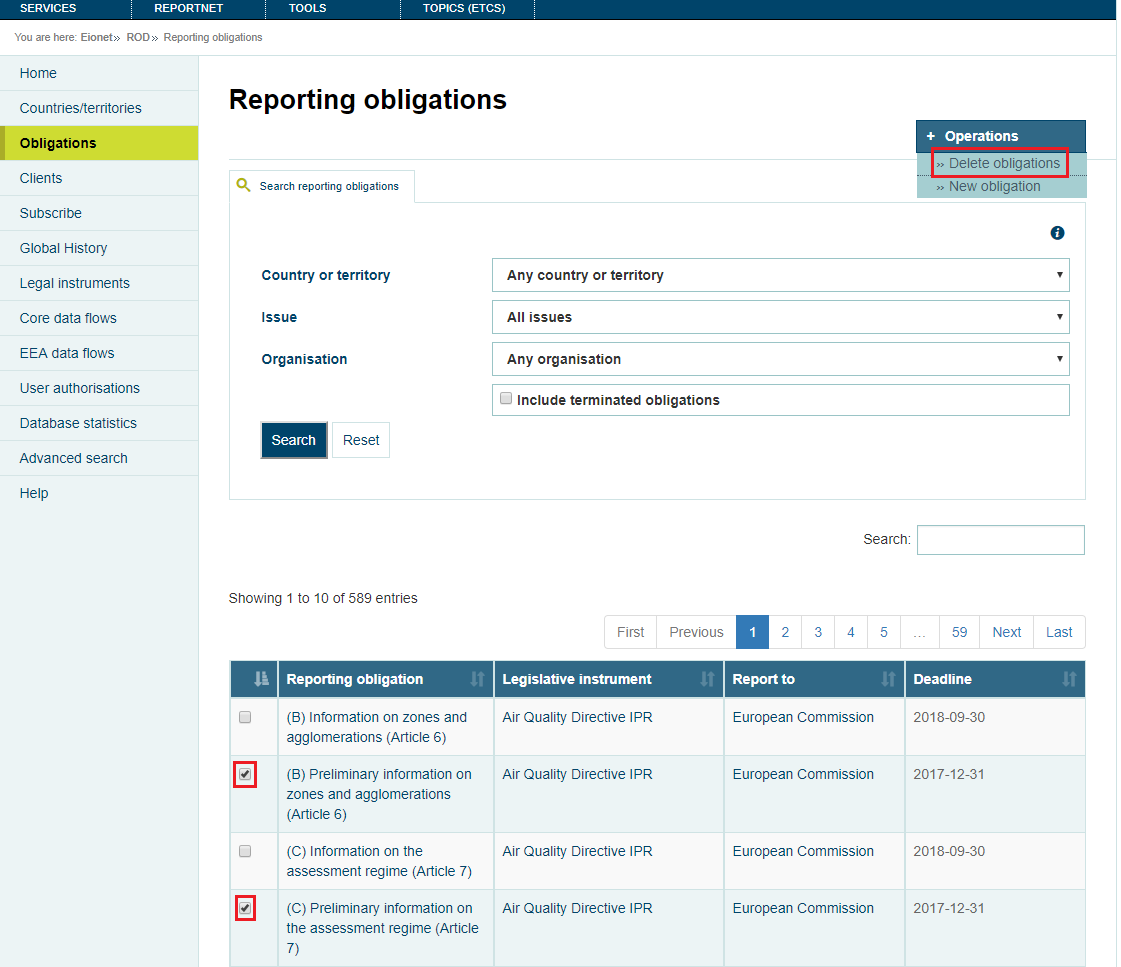
****

Figure 30

Checking various obligations and going to Operations – Delete obligations, the selected obligations will be eliminated.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - ObligationsDao: Used to delete the selected obligations.
    - UndoService: Used to insert the deleted values of the obligations into database.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_undo

### Edit obligation (previously logged)

****

Figure 31

In obligations/”obligationId” page there is Operations section which contains Edit and Delete actions. The first one goes to obligations/”obligationId”/edit (Figure 32).

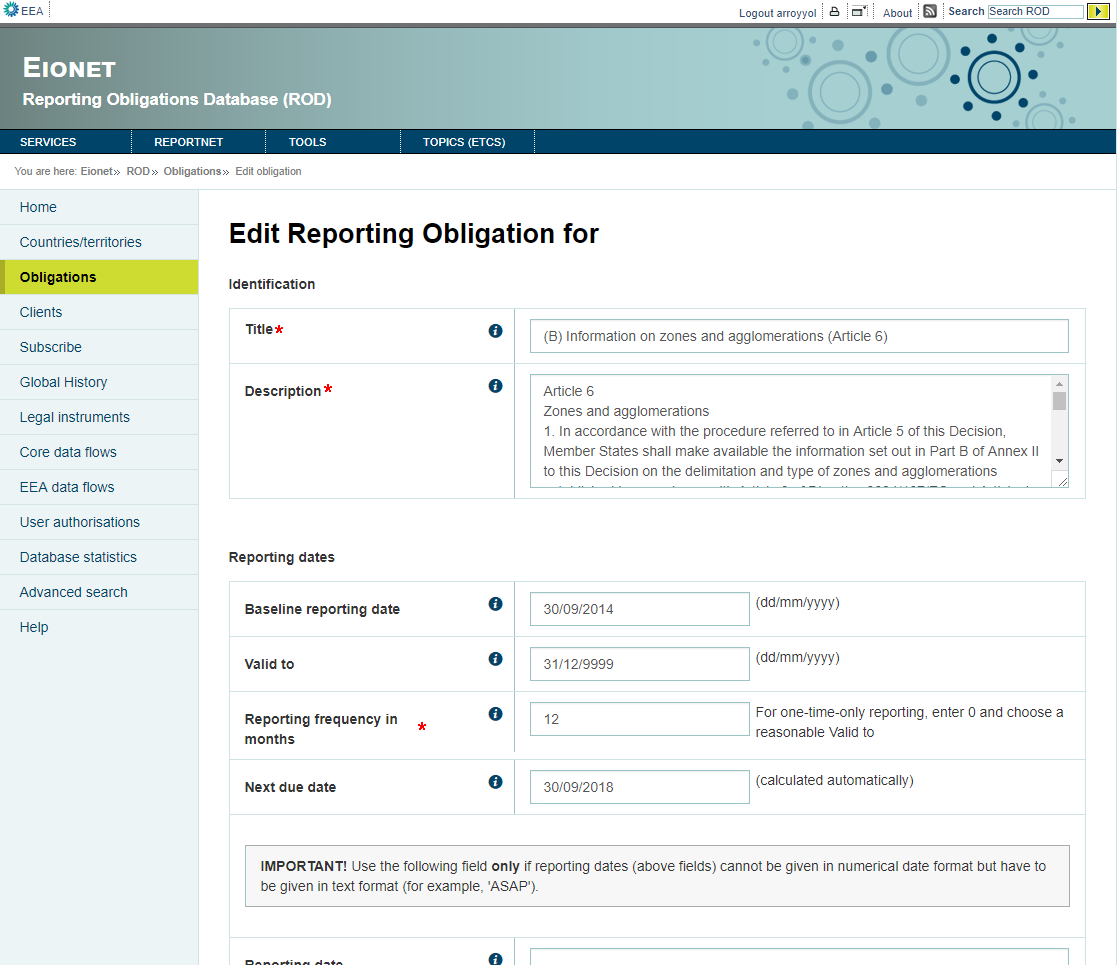


Figure 32

Updating the data of the obligation and clicking Save changes, the changes will be saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls ObligationsDao to use his methods.
    - SpatialService: Calls SpatialDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - ObligationsDao: Used to update obligation. Get the attribute values, clients, countries, issues and relation of obligation.
    - SpatialDao: Used to get all countries.
    - IssueDao: Used to get all issues.
    - ClientService: Used to get all clients.
    - UndoService: Used to insert the values before the update into the database.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Obligations
    - ClientDTO
    - Issue
    - Spatial
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Service: eionet.rod.service.ObligationService
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + View: eobligation.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_client
  + t\_spatial
  + t\_issue
  + t\_client\_obligation\_lnk
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_obligation\_relation
  + t\_undo

### Delete obligation (previously logged)

The second one ·Delete obligation” goes to obligations/”obligationId”/delete and it is used to delete the selected obligation (Figure 33).

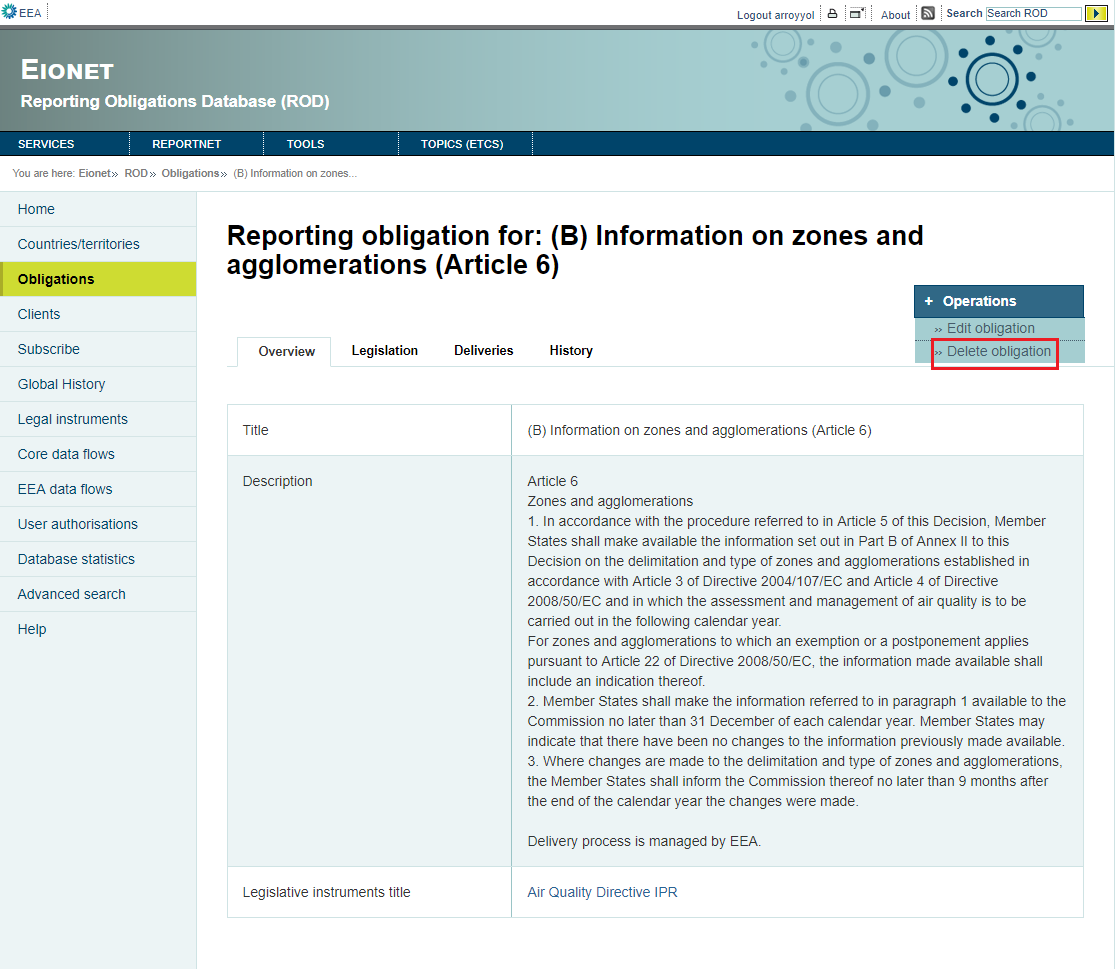
****

Figure 33

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - ObligationsDao: Used to delete the obligation.
    - UndoService: Used to insert the deleted values of the obligation into database.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_undo

## Clients module

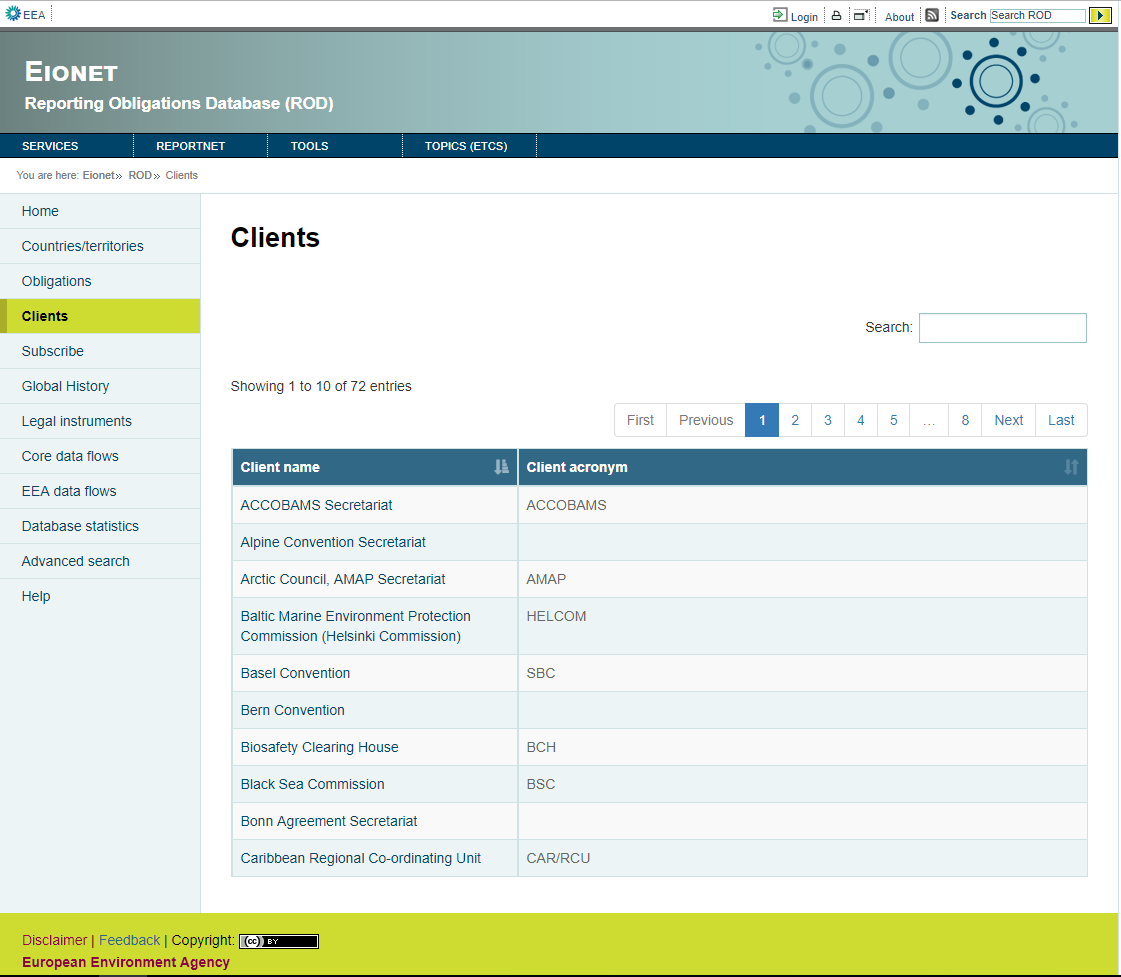


Figure 34

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to get clients.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clients.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + Actions:
    - View client
    - Add client (previously logged)
    - Delete various clients (previously logged)
    - Edit client (previously logged)
    - Delete obligation (previously logged)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client

### View client

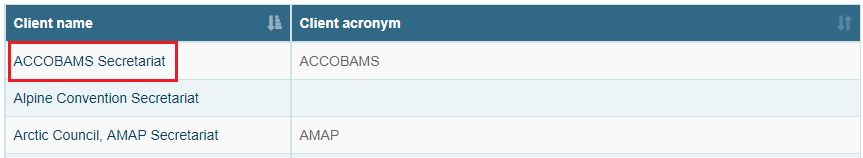
****

Figure 35

Clicking in a client the application goes to /clients/”clientId” (Figure 36) to show the details.

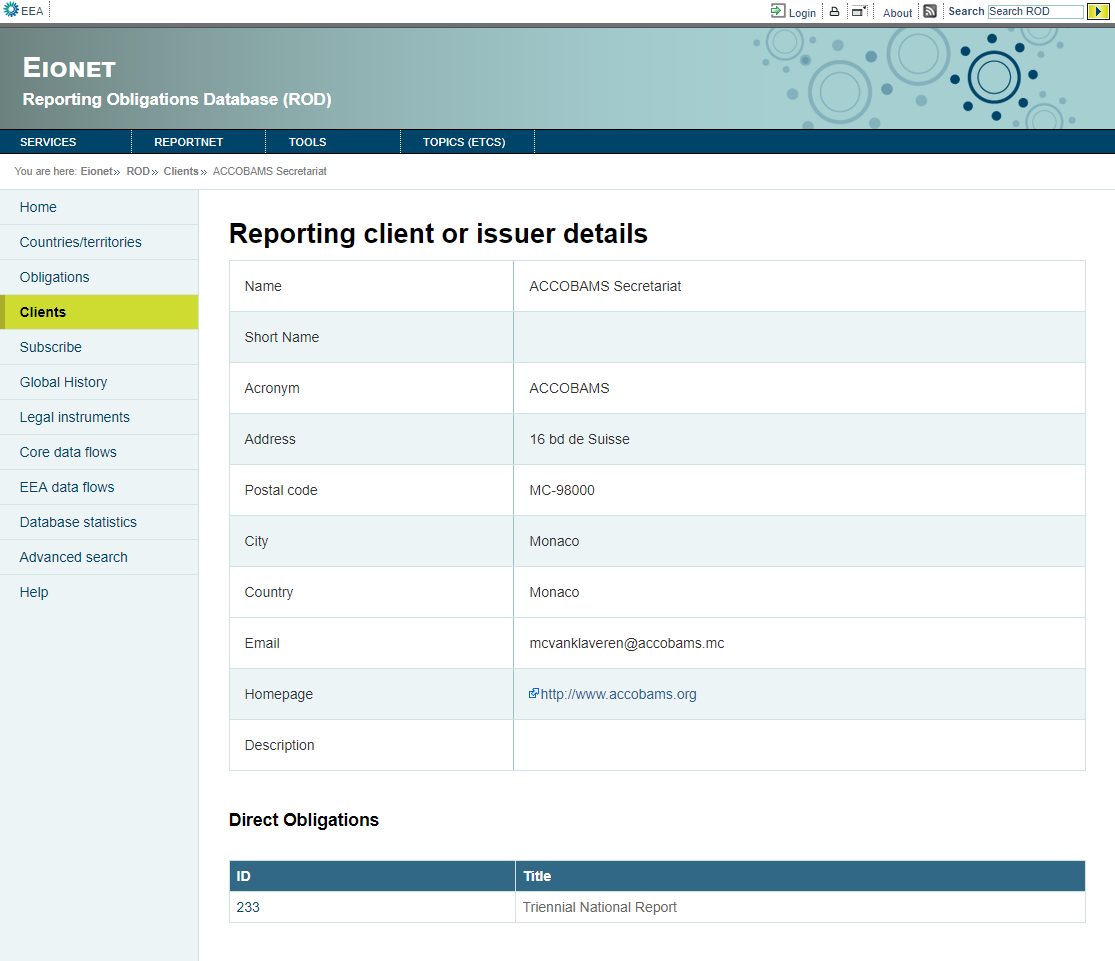


Figure 36

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to get details, obligations and instruments of client.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clientFacsheet.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client
  + t\_obligation
  + t\_client\_obligation\_lnk
  + t\_source
  + t\_client\_source\_lnk

### Add client (previously logged)

****

Figure 37

The operations menu contains clients maintenance operations (new and delete). The first one goes to /clients/add (Figure 38).

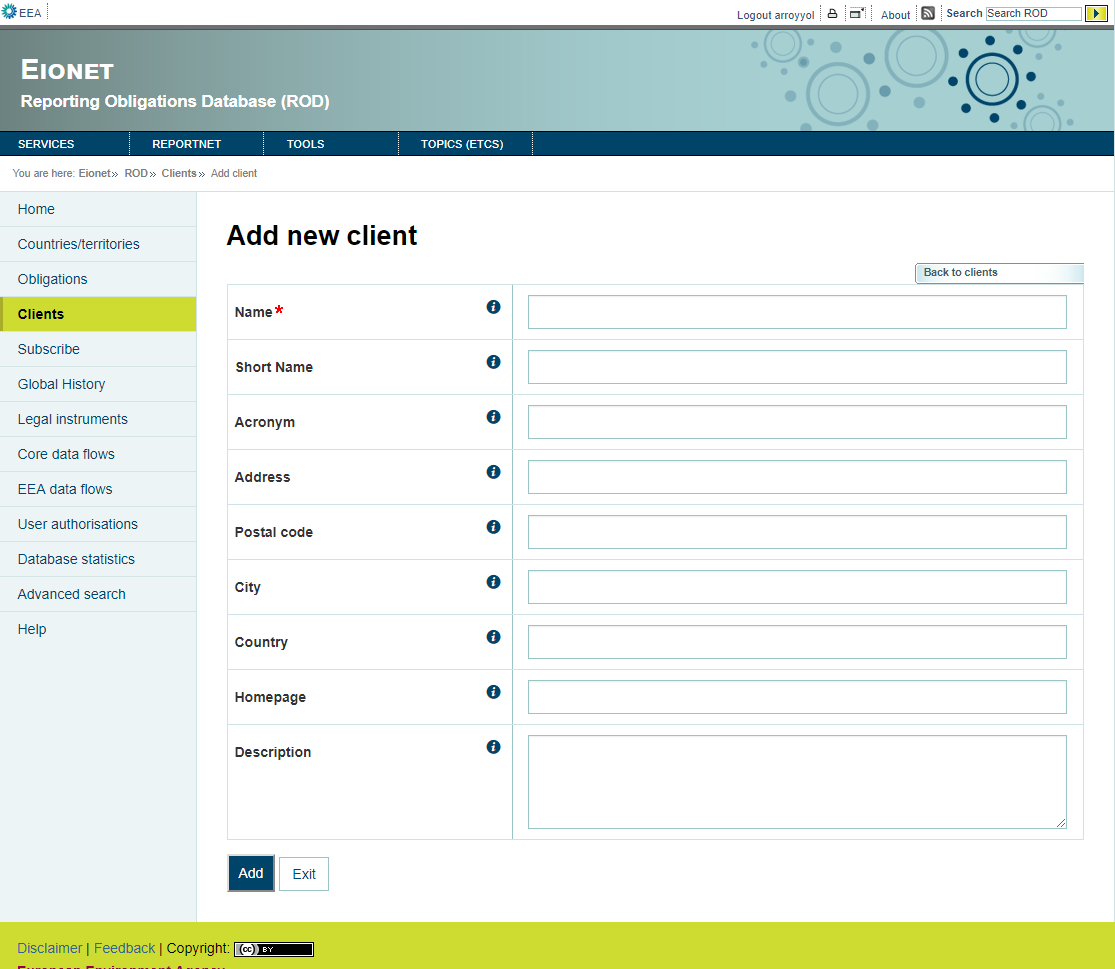


Figure 38

Filling the fields and clicking Add button, the client is saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to insert new client.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clientNewClient.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client

### Delete various clients (previously logged)

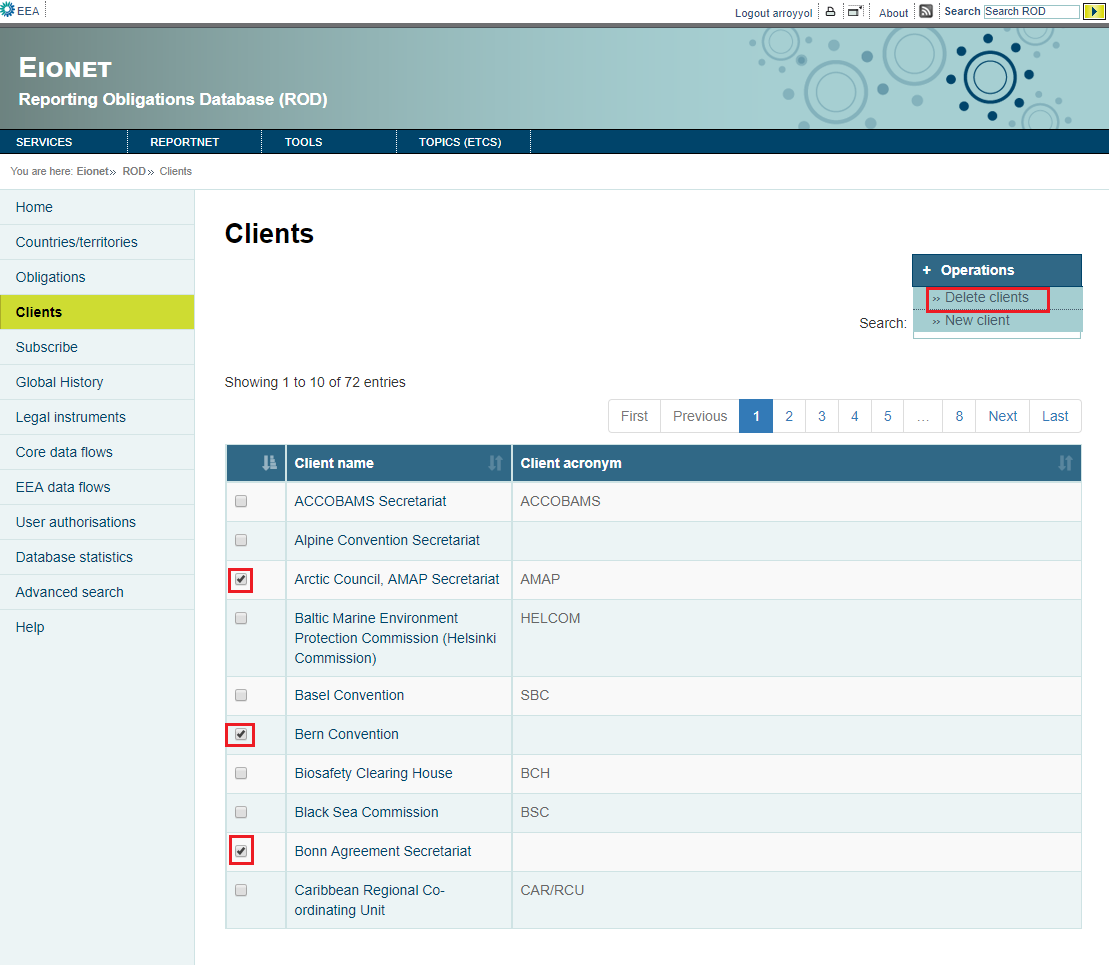
****

Figure 39

Checking various clients and going to Operations – Delete clients, the selected clients will be deleted.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to delete selected clients.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clients.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client

### Edit client (previously logged)

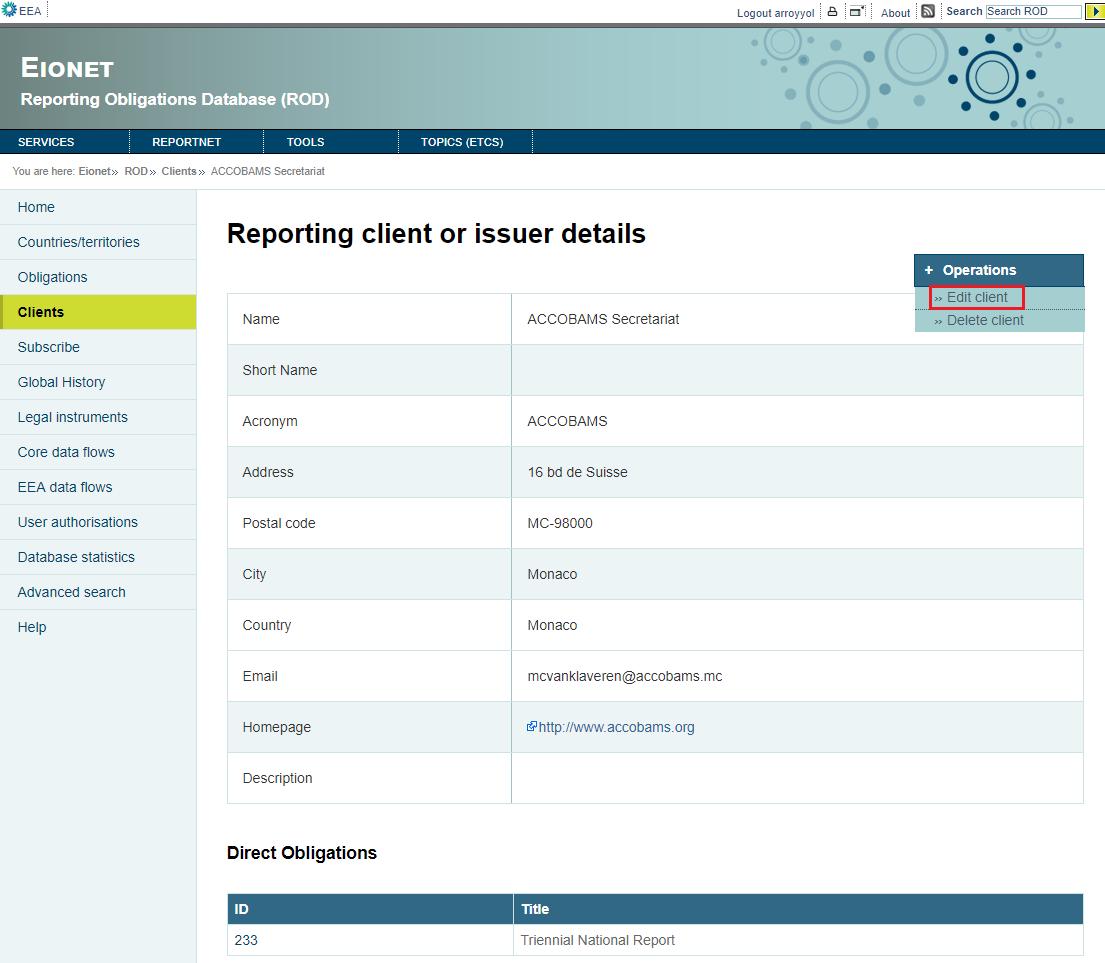
****

Figure 40

In clients/{clientId} page going to Operations – Edit client goes to clients/{clientId}/edit (Figure 41).

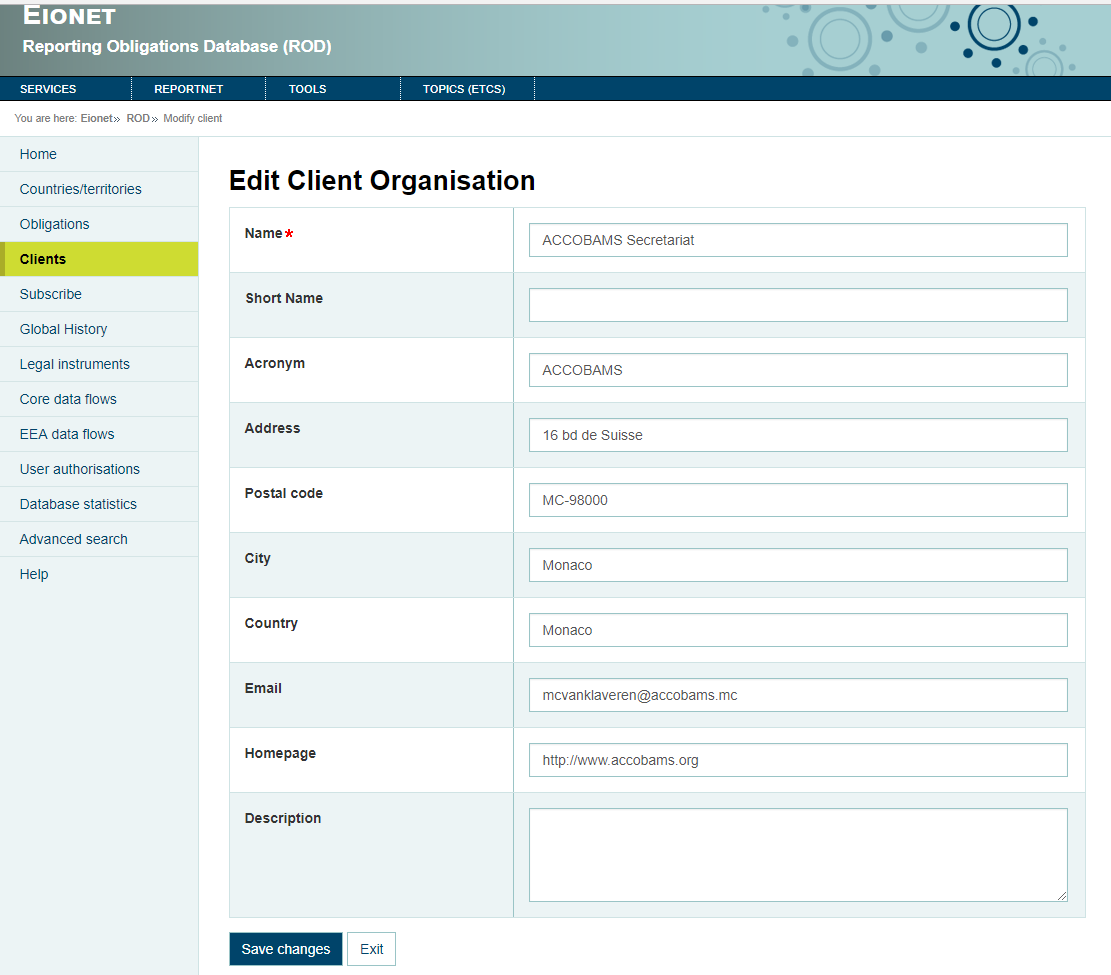


Figure 41

Once the user has updated the data and clicks “Save changes” button, the changes will be saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to update client.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clientEditForm.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client

### Delete obligation (previously logged)

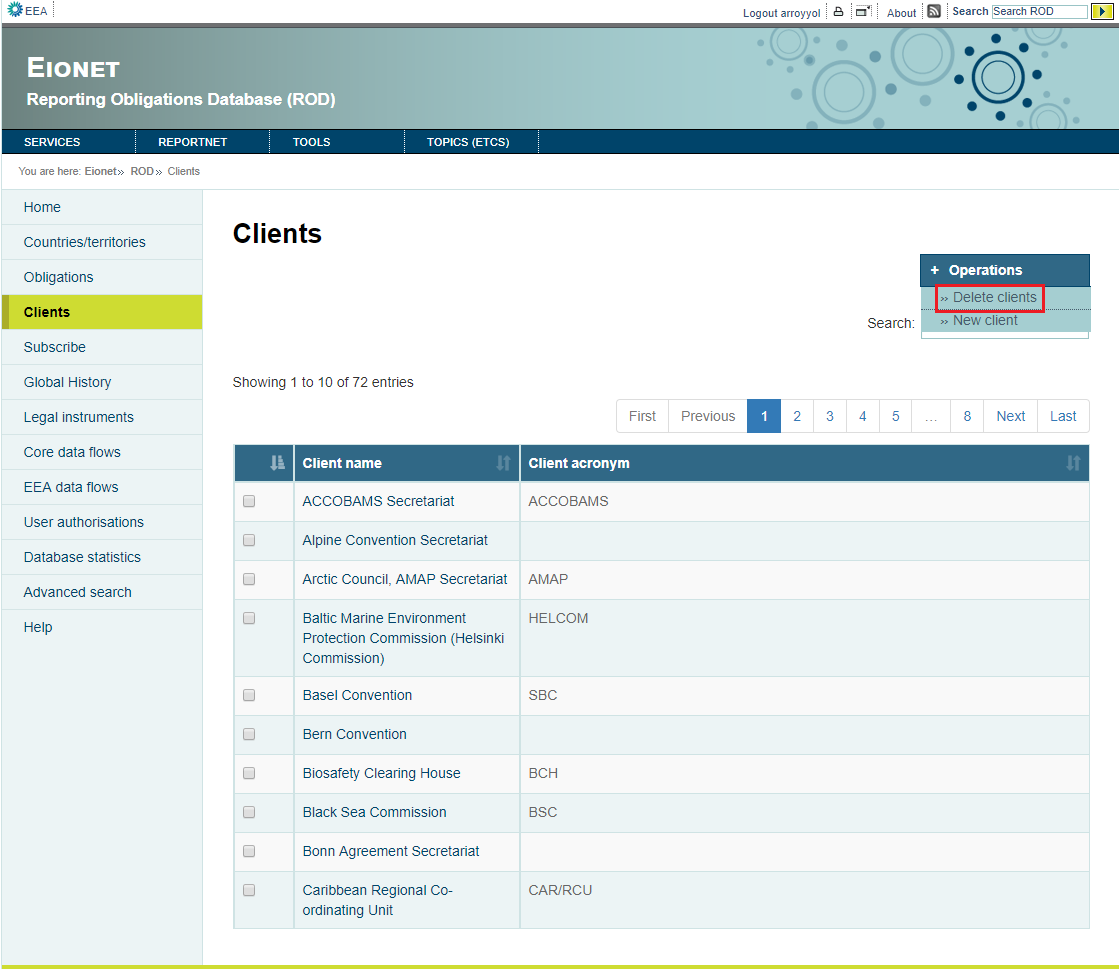
****

Figure 42

In clients/{clientId} page going to Operations – Delete client, the client will be deleted.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - ClientService: Used to delete client.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - ClientDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ClientsController
  + Dao: eionet.rod.dao.ClientService
  + Model: eionet.rod.model.ClientDTO
  + View: clientFacsheet.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_client

## Global History module

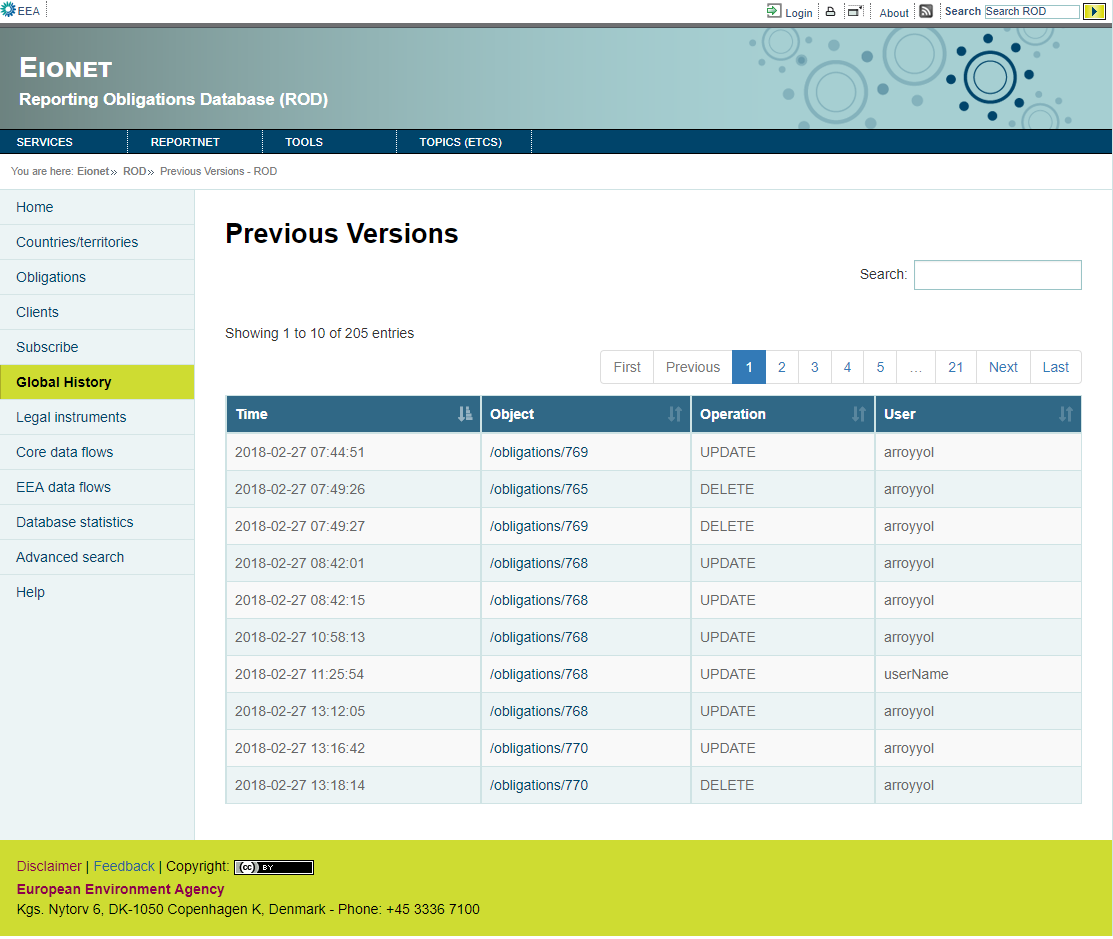


Figure 43

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - UndoService: Used to get previous actions of the instruments and obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - UndoDTO.
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.UndoController
  + Dao: eionet.rod.dao.UndoService
  + Model: eionet.rod.model.UndoDTO
  + View: versions.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + Actions:
    - See the changes of previous version

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_undo

### See the changes of previous version

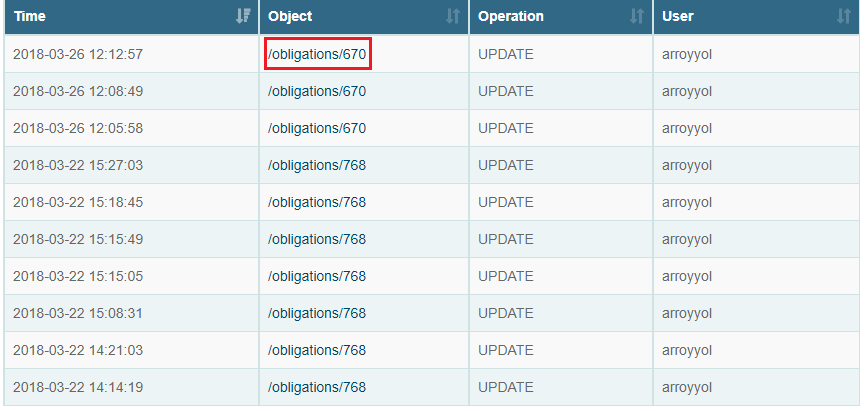
****

Figure 44

Clicking the Object of a previous version the module goes to /undoinfo?ts=”timeOfUpdateOrDelete”&tab=”table”&op=”operation”&id=”id”&user=”user” (Figures 42 and 43).

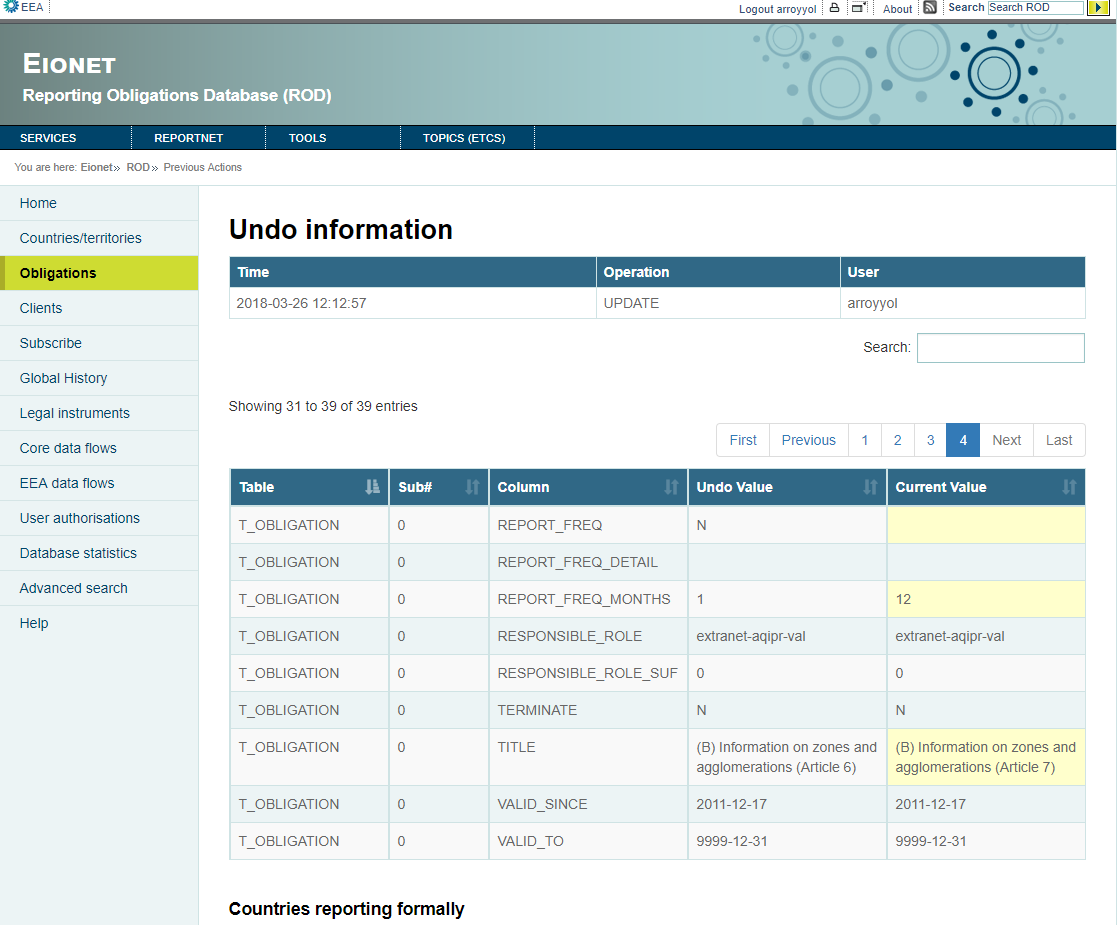


Figure 45



Figure 46

This page shows the values of the current version and the values of the previous version (Undo value) related to the instrument/obligation.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: It calls the ObligationsDao to use his methods.
    - SpatialService: It calls he SpatialDao to use his methods
  + Data Access Layer (Using Data Access Object - DAO):
    - UndoService: Used to get previous values of the instrument or obligation.
    - ObligationsDao: Used to get the actual values of the obligation.
    - SpatialDao: Used to get the countries that report the obligation.
    - IssueDao: Used to get the environmental issues of the obligation.
    - ClientService: Used to get the clients that use the report of obligation.
    - SourceService: Used to get the actual values of the instrument.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - UndoDTO
    - Obligations
    - Spatial
    - Issue
    - ClientDTO
    - InstrumentFacsheetDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.UndoInfoController
  + Dao: eionet.rod.dao.UndoService
  + Model: eionet.rod.model.UndoDTO
  + View: undoinfo.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_undo
  + t\_obligation
  + t\_source
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_issue
  + t\_raissue\_lnk
  + t\_obligation\_relation
  + t\_spatial
  + t\_raspatial\_lnk

## Instruments module

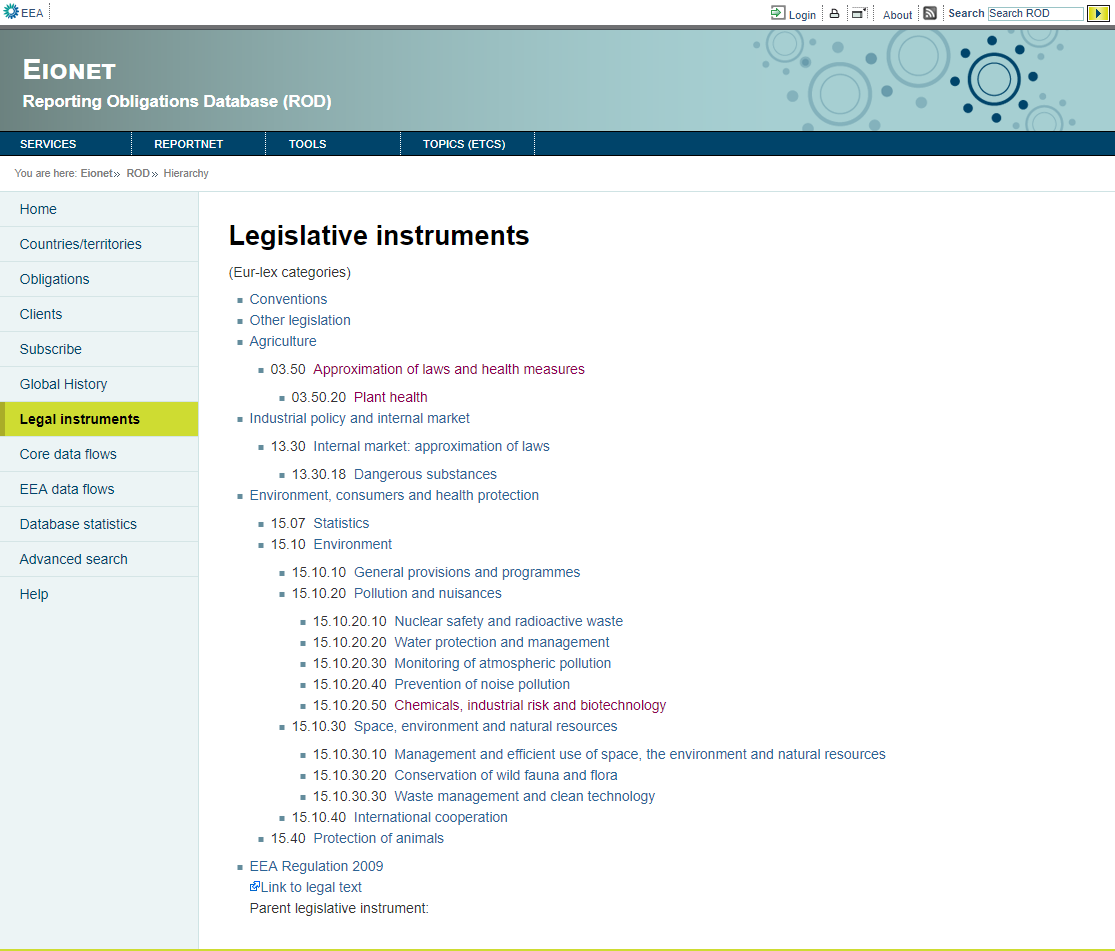


Figure 47

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Data Access Layer (Using Data Access Object - DAO):
    - SourceService: Used to get instruments and his hierarchy.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - HierarchyInstrumentDTO
    - InstrumentListDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.InstrumentsController
  + Dao: eionet.rod.dao.SourceService
  + Model:
    - eionet.rod.model.HierarchyInstrumentDTO
    - eionet.rod.model.InstrumentListDTO
  + View: instruments.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + Actions:
    - View instrument
    - Add instrument (previously logged)
    - Edit instrument (previously logged)
    - Delete instrument (previously logged)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source\_class
  + t\_source\_lnk
  + t\_source

### View instrument

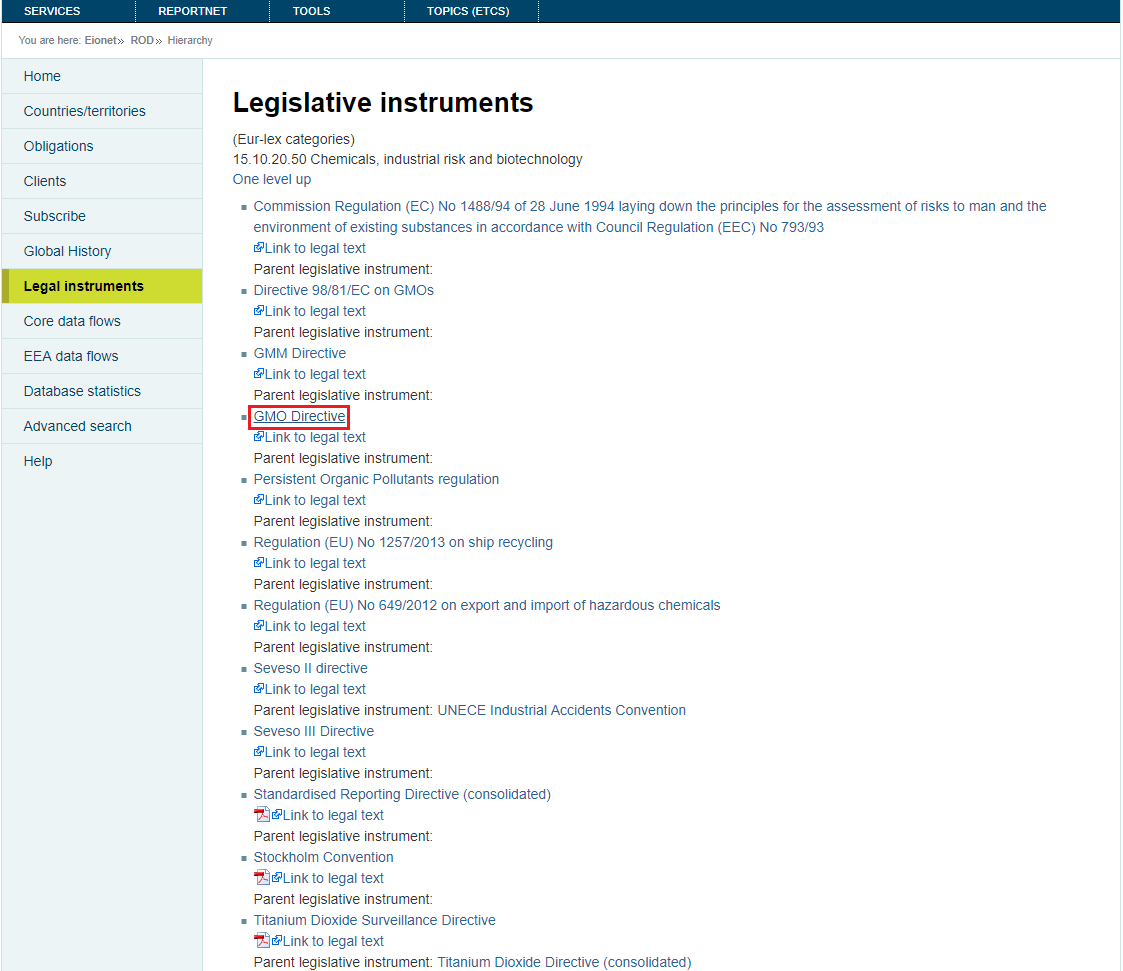
****

Figure 48

Clicking on any instrument the application goes to /instrument/”instrumentId” (Figure 49).

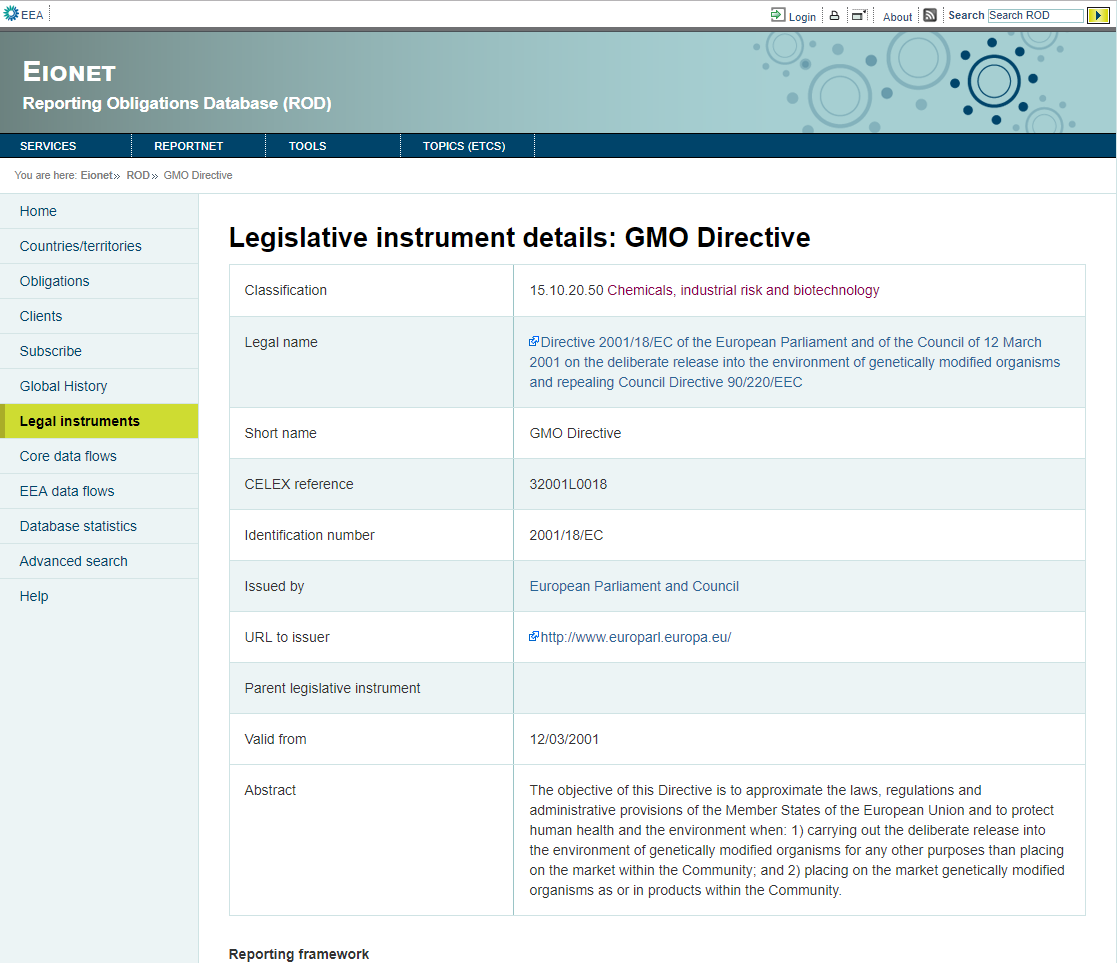


Figure 49

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Data Access Layer (Using Data Access Object - DAO):
    - SourceService: Used to get information of the instrument.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - InstrumentFacsheetDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.InstrumentsController
  + Dao: eionet.rod.dao.SourceService
  + Model: eionet.rod.model.InstrumentFacsheetDTO
  + View: instrumentFacsheet.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source
  + t\_source\_lnk
  + t\_source\_class

### Add instrument (previously logged)

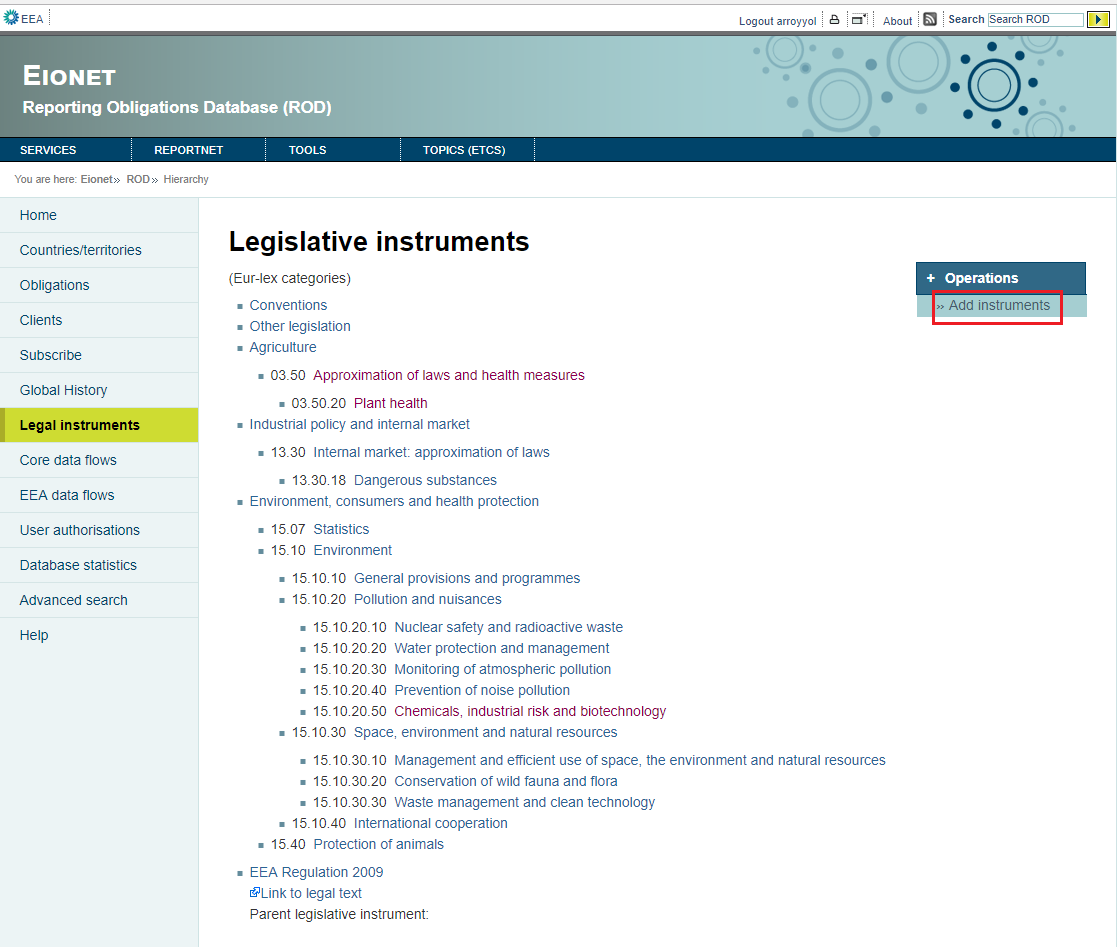
****

Figure 50

Add instrument appears in the Operations menu which executes /instruments/add (Figure 51).

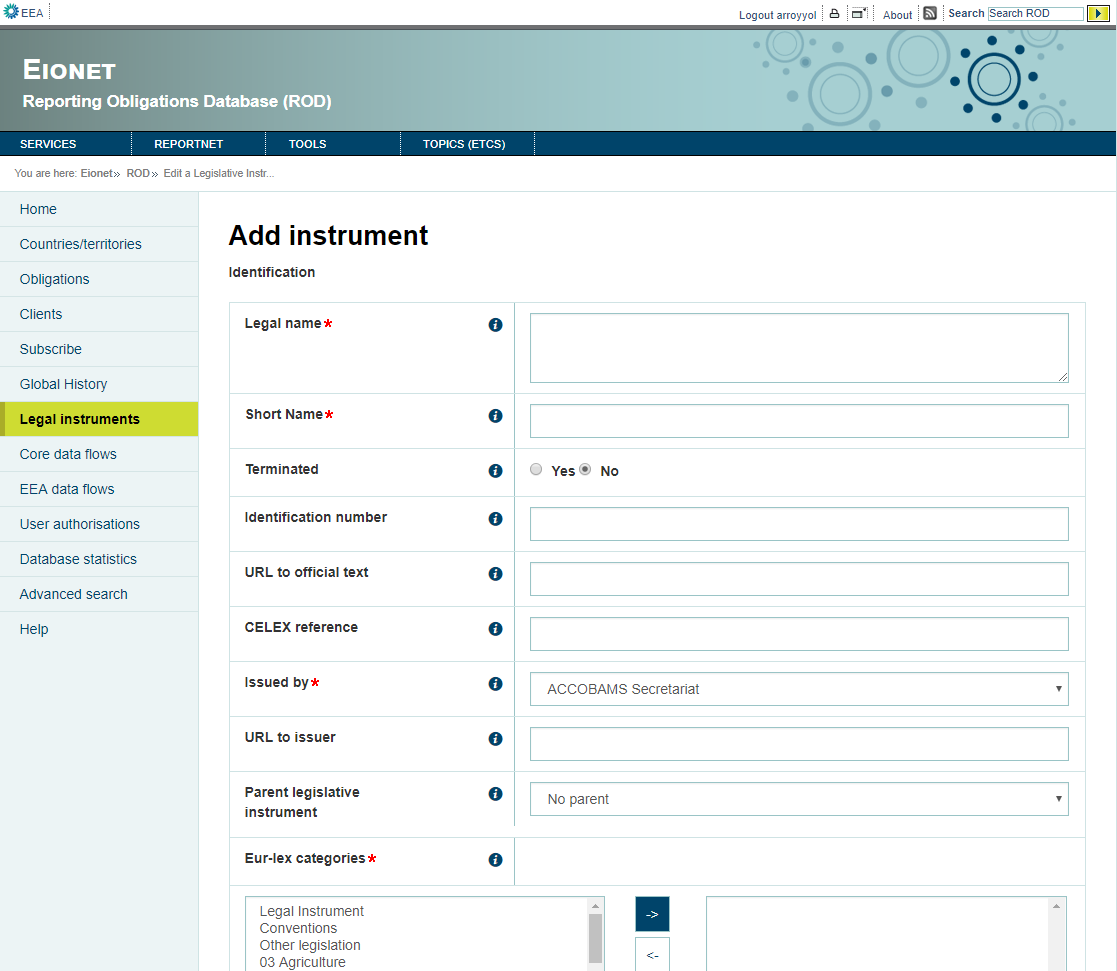


Figure 51

Filling the fields with the data of the new instrument and clicking Add button, the new instrument is saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - SourceService: Used to add new instrument, get all instruments to charge parent combo and get all classifications.
    - ClientService: Used to charge the clients combo.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - InstrumentFacsheetDTO
    - ClientDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.InstrumentsController
  + Dao: eionet.rod.dao.SourceService
  + Model: eionet.rod.model.InstrumentFacsheetDTO
  + View: instrumentEditForm.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source
  + t\_client
  + t\_source\_class
  + t\_client\_source\_lnk
  + t\_source\_lnk

### Edit instrument (previously logged)

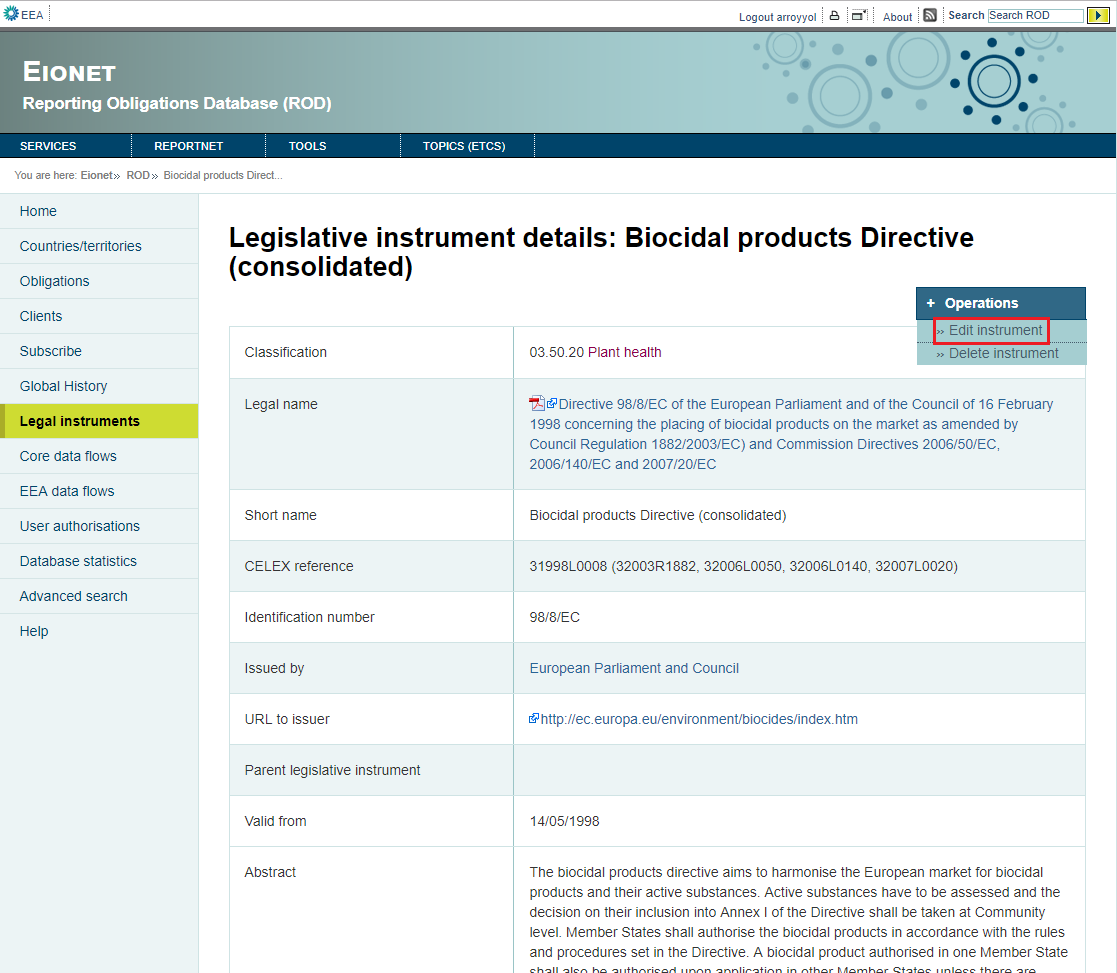
****

Figure 52

In instruments/”instrumentId” page Operations menu contains maintenance options. Edit obligation goes to instruments/edit?sourceId=”instrumentId” (Figure 50).



Updating the fields and clicking Save changes button, the changes will be saved in the database.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - SourceService: Used to get instrument information, edit instrument, get all instruments to charge parent combo and get all classifications.
    - ClientService: Used to charge the clients combo.
    - UndoService: Used to insert the values before the update into the database.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - InstrumentFacsheetDTO
    - ClientDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.InstrumentsController
  + Dao: eionet.rod.dao.SourceService
  + Model: eionet.rod.model.InstrumentFacsheetDTO
  + View: instrumentEditForm.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source
  + t\_client
  + t\_source\_class
  + t\_client\_source\_lnk
  + t\_source\_lnk
  + t\_undo

### Delete instrument (previously logged)

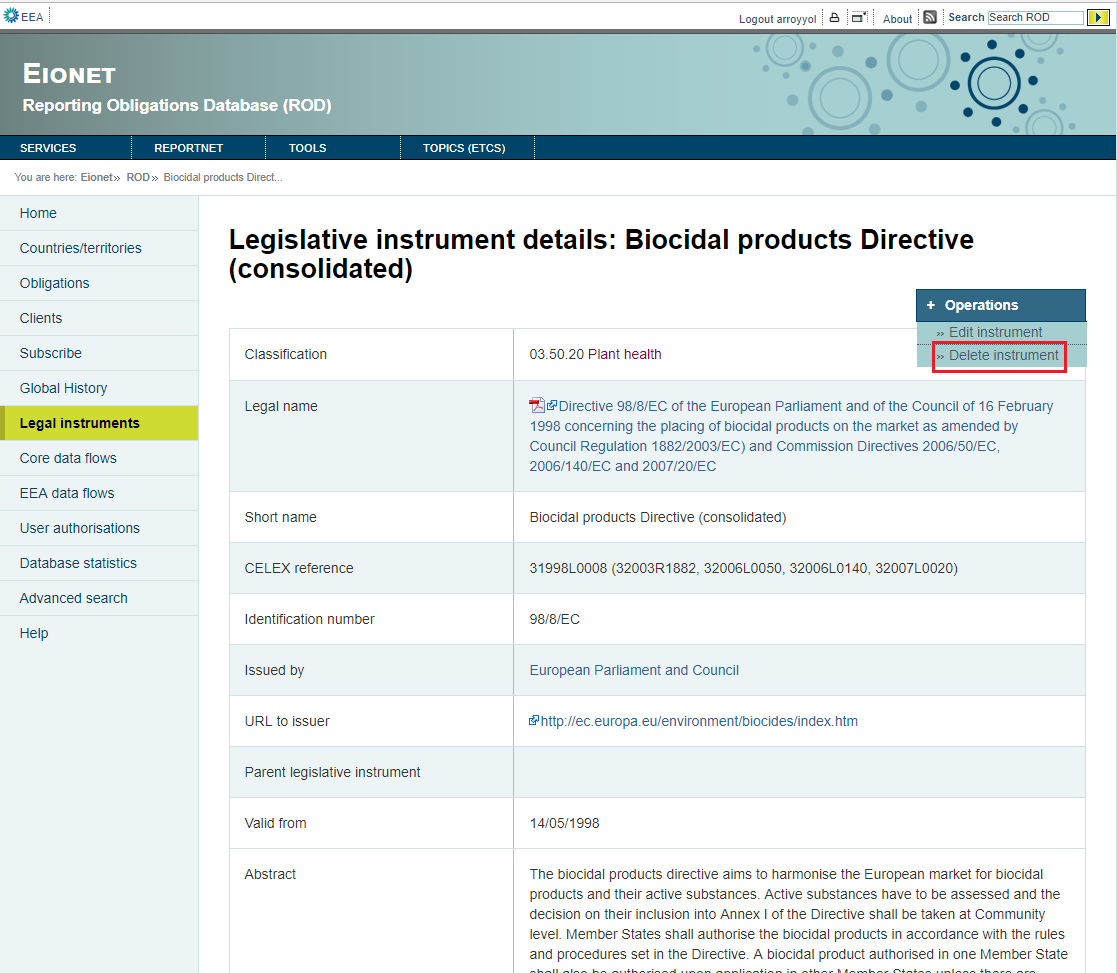
****

Figure 54

Operations menu contains a second option– Delete instrument. With this option the instrument will be deleted.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO):
    - SourceService: Used to delete instrument.
    - UndoService: Used to insert the obligation deleted values into the database.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - InstrumentFacsheetDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.InstrumentsController
  + Dao: eionet.rod.dao.SourceService
  + Model: eionet.rod.model.InstrumentFacsheetDTO
  + View: instrumentFacsheet.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_source
  + t\_source\_lnk
  + t\_client\_source\_lnk
  + t\_undo

## Core dataflow module

The application goes to /obligations?anmode=P and shows the obligations used for the Eionet core data flows.

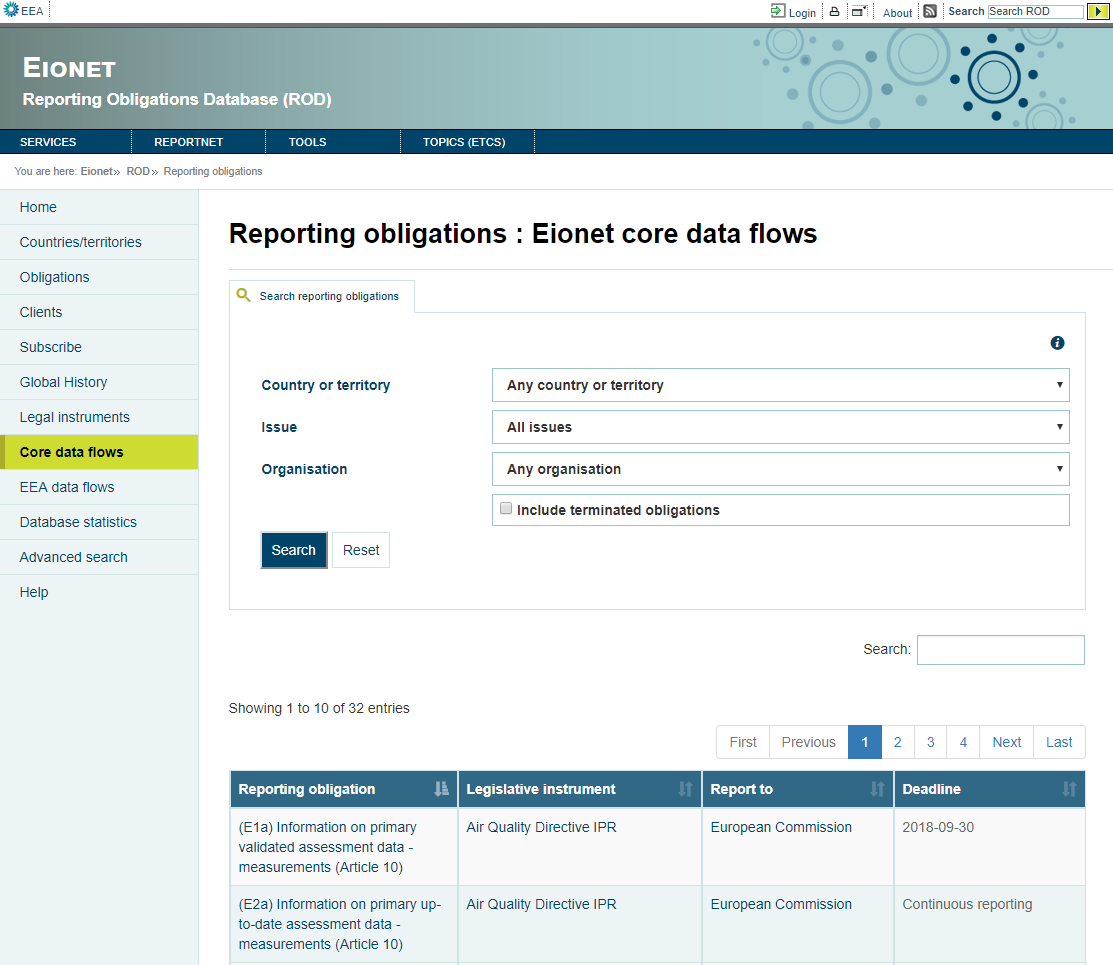


Figure 55

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: Calls the SpatialDao to use his methods.
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations or the searched obligations with EEA\_PRIMARY=1.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

## EEA dataflow module

The application goes to /obligations?anmode=F and shows the obligations in which the delivery process is managed by EEA.

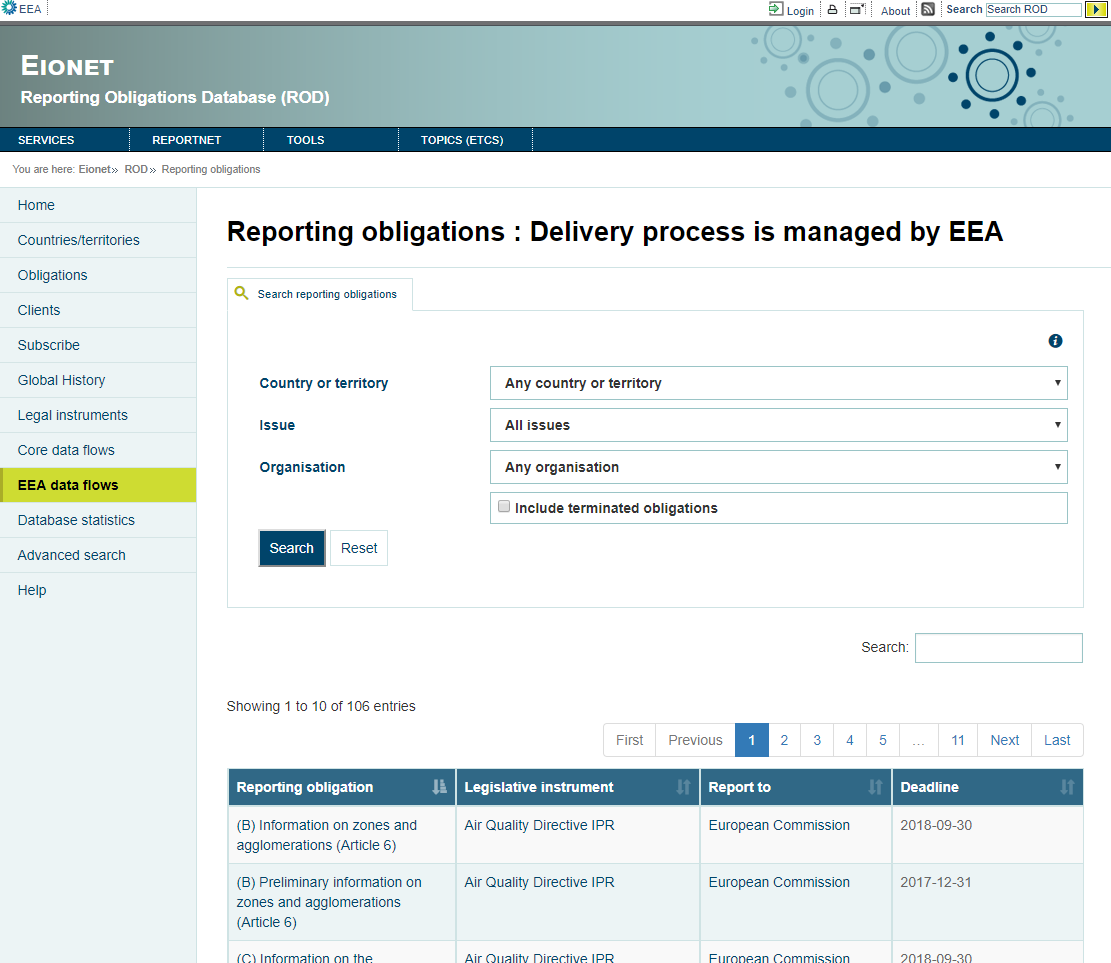


Figure 56

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: Calls the SpatialDao to use his methods.
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations or the searched obligations with FLAGGED=1.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

## Database statistics module

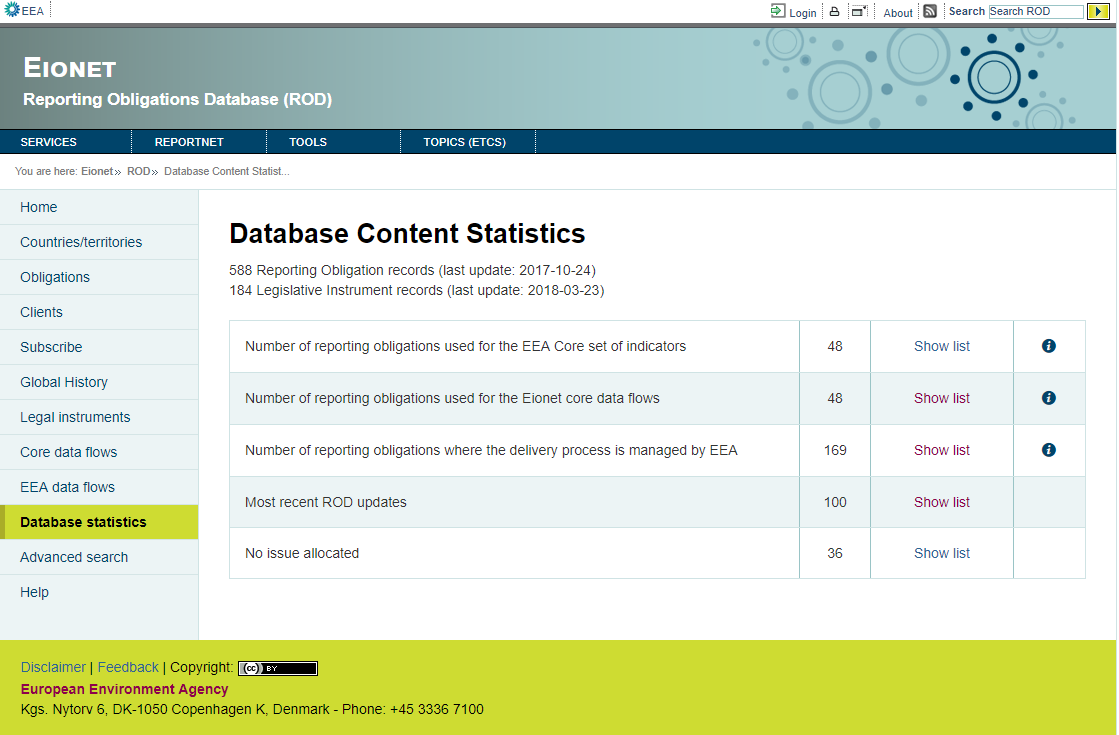


Figure 57

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - AnalysisService: Used to get statistics on instruments and obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - AnalysisDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.AnalysisController
  + Dao: eionet.rod.dao.AnalysisService
  + Model: eionet.rod.model.AnalysisDTO
  + View: analysis.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + actions:
    - Show obligations with EEA\_CORE=1
    - Show obligations with EEA\_PRIMARY=1
    - Show obligations with FLAGGED=1
    - Show most recent updates
    - Show obligations without issues

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_source

### Show obligations with EEA\_CORE=1

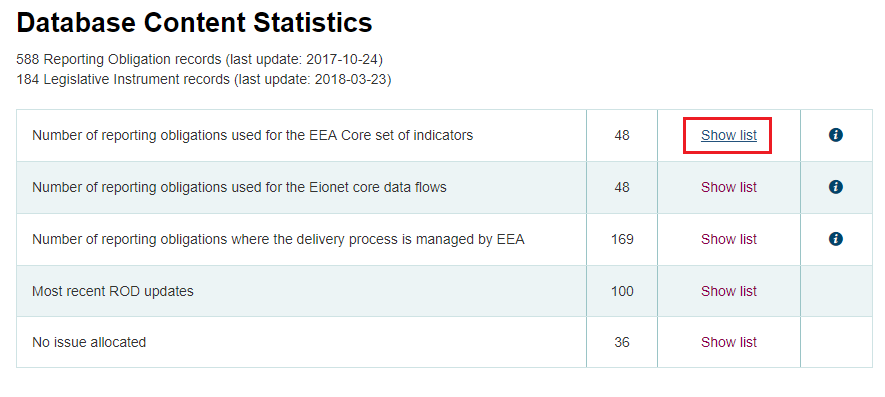


Figure 58

The application goes to /obligations?anmode=C and shows the obligations used for the EEA Core set of indicator (Figure 58).

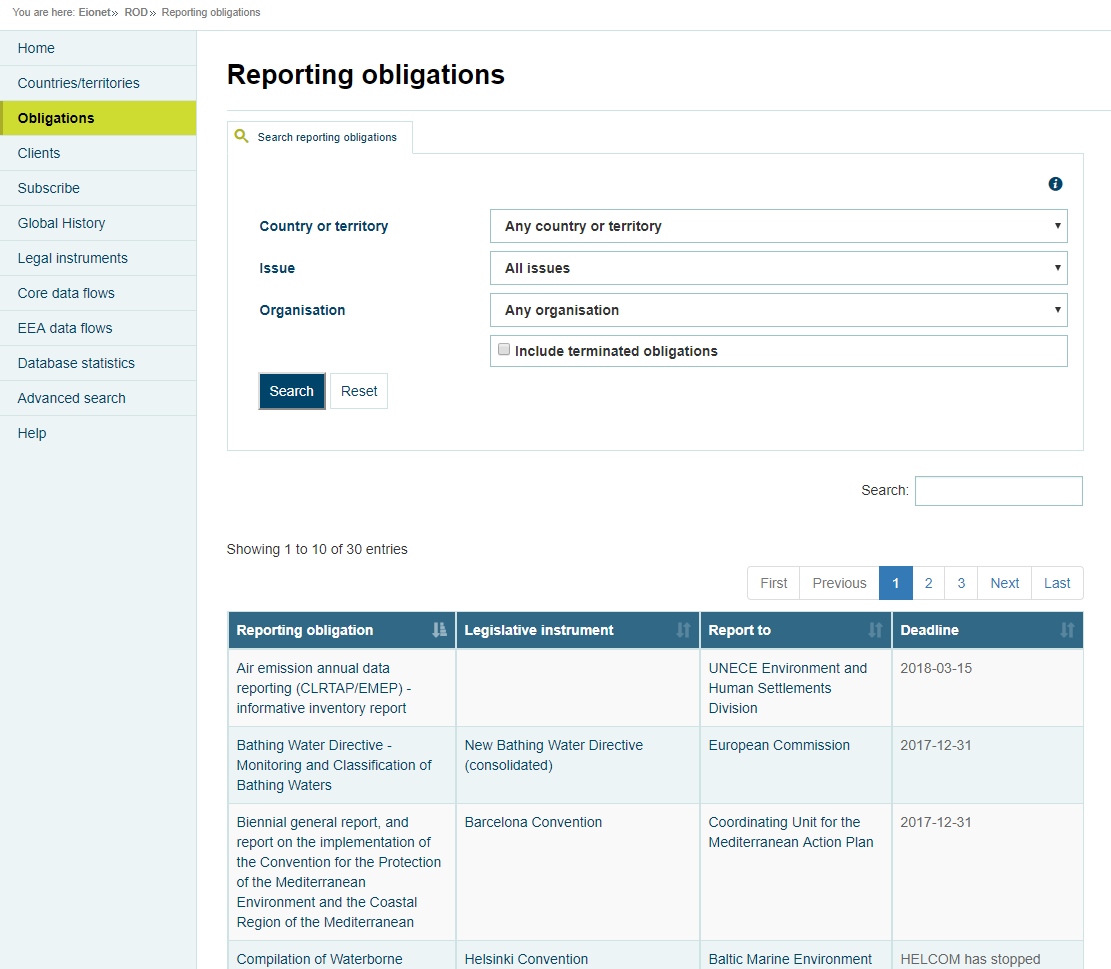


Figure 59

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: Calls the SpatialDao to use his methods.
    - ObligationService: Calls the ObligationsDao to use his methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations or the searched obligations with EEA\_CORE=1.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

### Show obligations with EEA\_PRIMARY=1

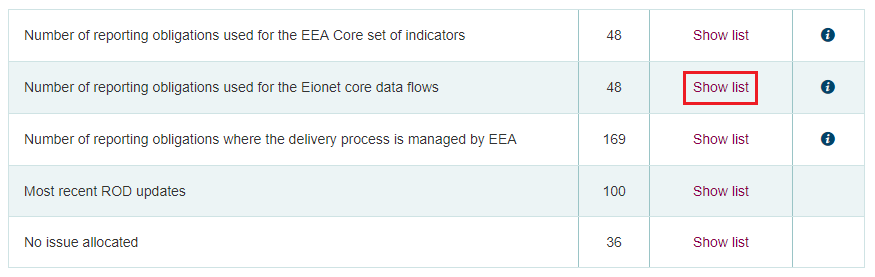


Figure 60

The application goes to /obligations?anmode=P and shows the obligations used for the Eionet core data flows (Figure 55).

### Show obligations with FLAGGED=1

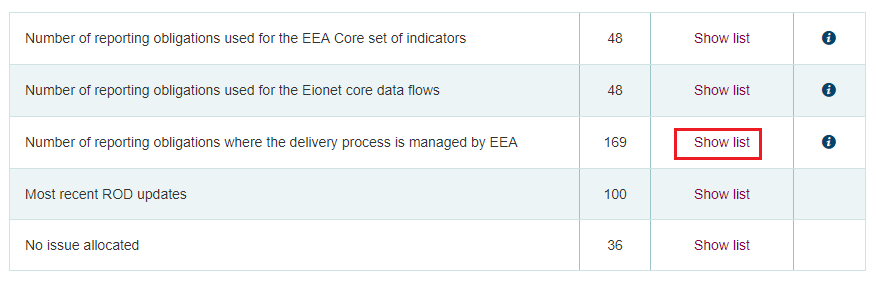
****

Figure 61

The application goes to /obligations?anmode=F and shows the obligations where the delivery process is managed by EEA (Figure 56).

### Show most recent updates

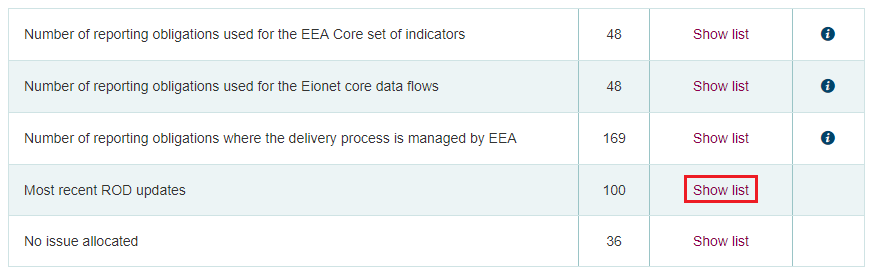
****

Figure 62

The application goes to /updatehistory showing the last 100 updates of obligations and instruments (Figure 62), in this example /obligations/670.

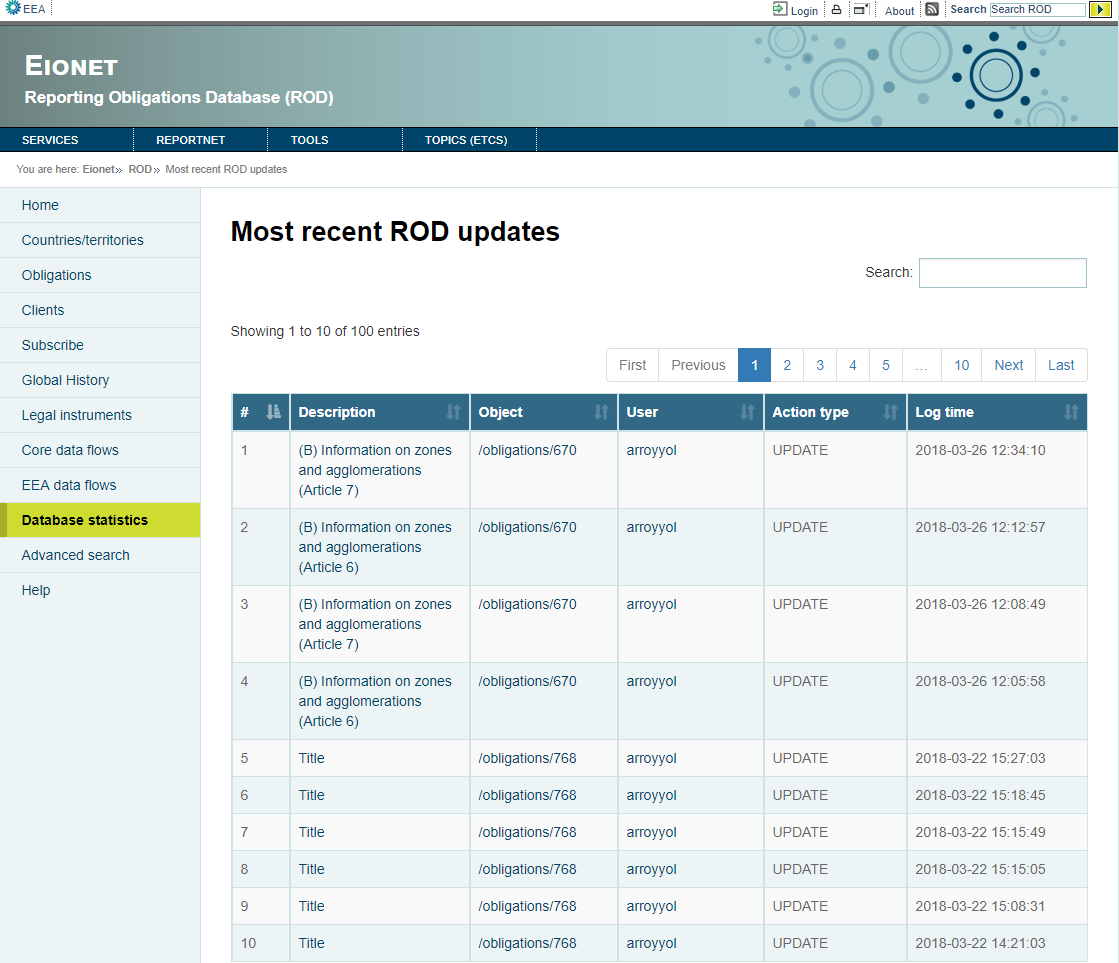


Figure 63

Clicking on the Object link the application shows only the updates of this object (Figure 63).

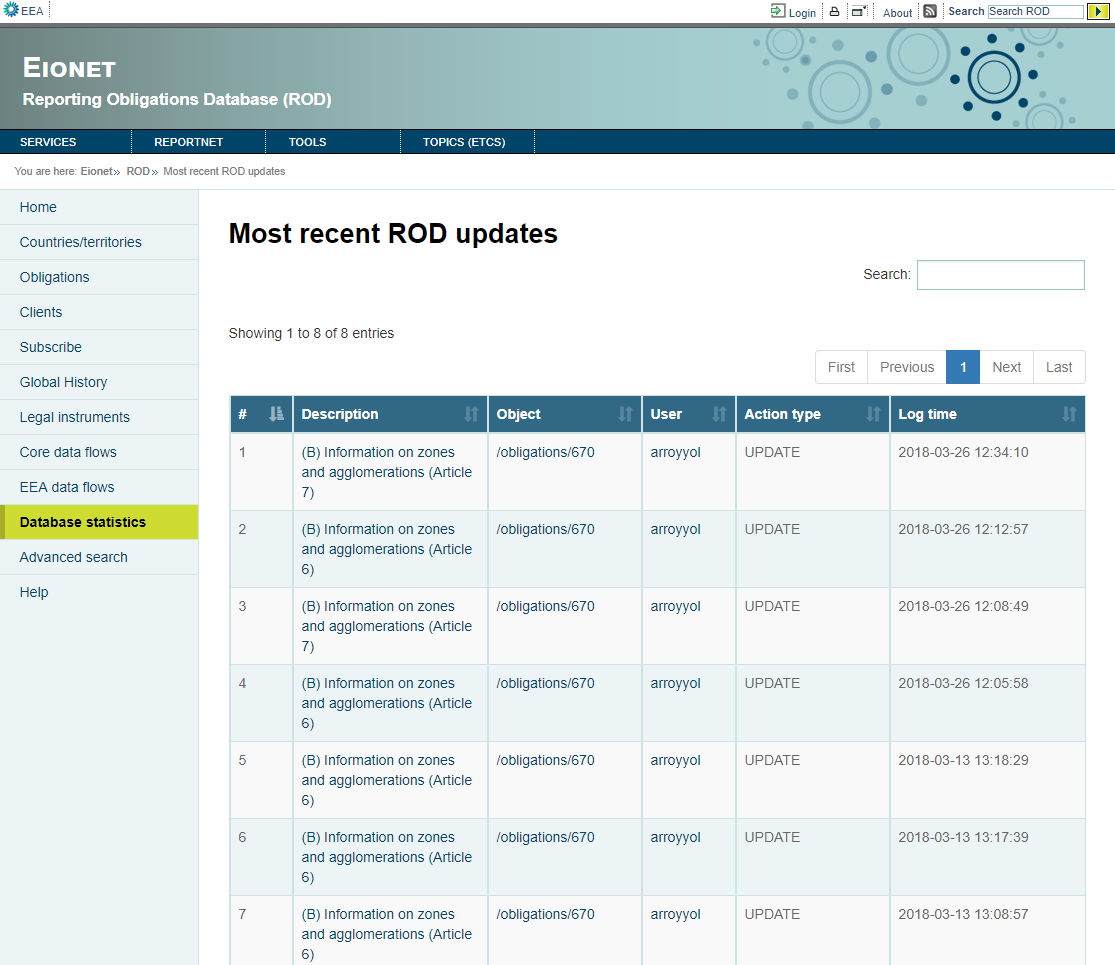


Figure 64

Clicking on the User link the application shows only the updates that user has made (Figure 64)

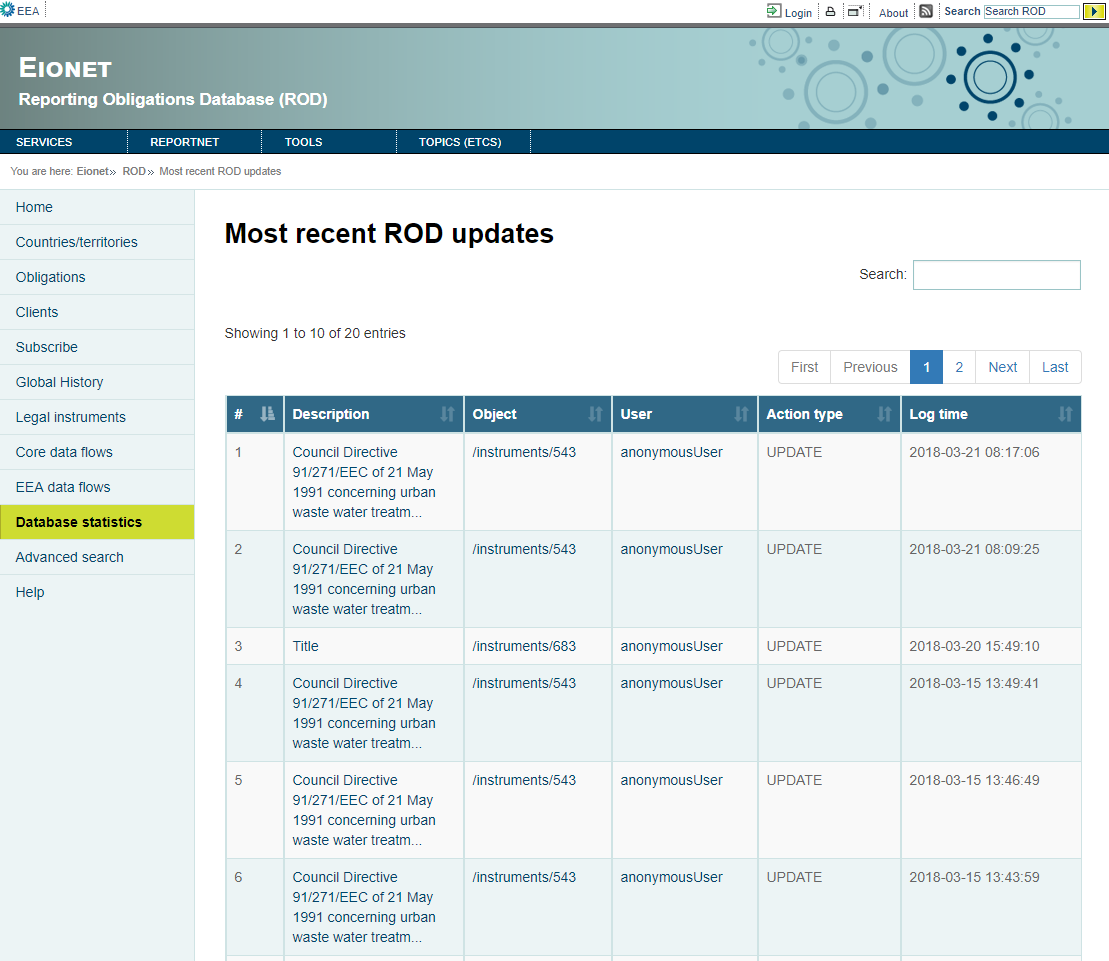


Figure 65

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - UndoService: Used to get the last 100 updates of instruments and obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - UndoDTO
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.UpdateHistoryController
  + Dao: eionet.rod.dao.UndoService
  + Model: eionet.rod.model.UndoDTO
  + View: updatehistory.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_undo

### Show obligations without issues

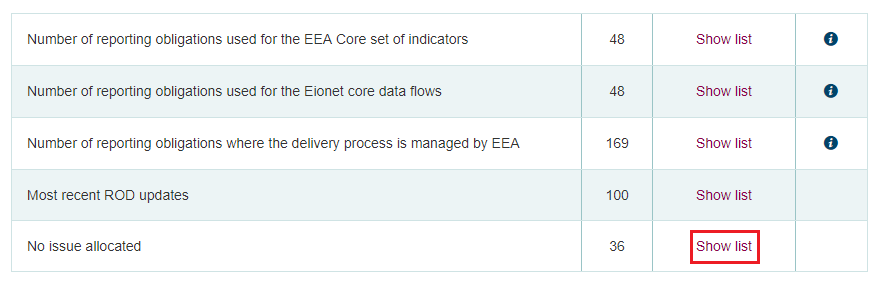


Figure 66

The application goes to /obligations?anmode=NI and shows the obligations without issues (Figure 67).



Figure 67

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: It calls the SpatialDao to use its methods.
    - ObligationService: It calls the ObligationsDao to use its methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations or the searched obligations without issues.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.ObligationsController
  + Dao: eionet.rod.dao.ObligationsDao
  + Model: eionet.rod.model.Obligations
  + Service: eionet.rod.service.ObligationService
  + View: obligations.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_role
  + t\_client\_obligation\_lnk
  + t\_client
  + t\_source
  + t\_raspatial\_lnk
  + t\_raissue\_lnk
  + t\_issue

## Advanced search

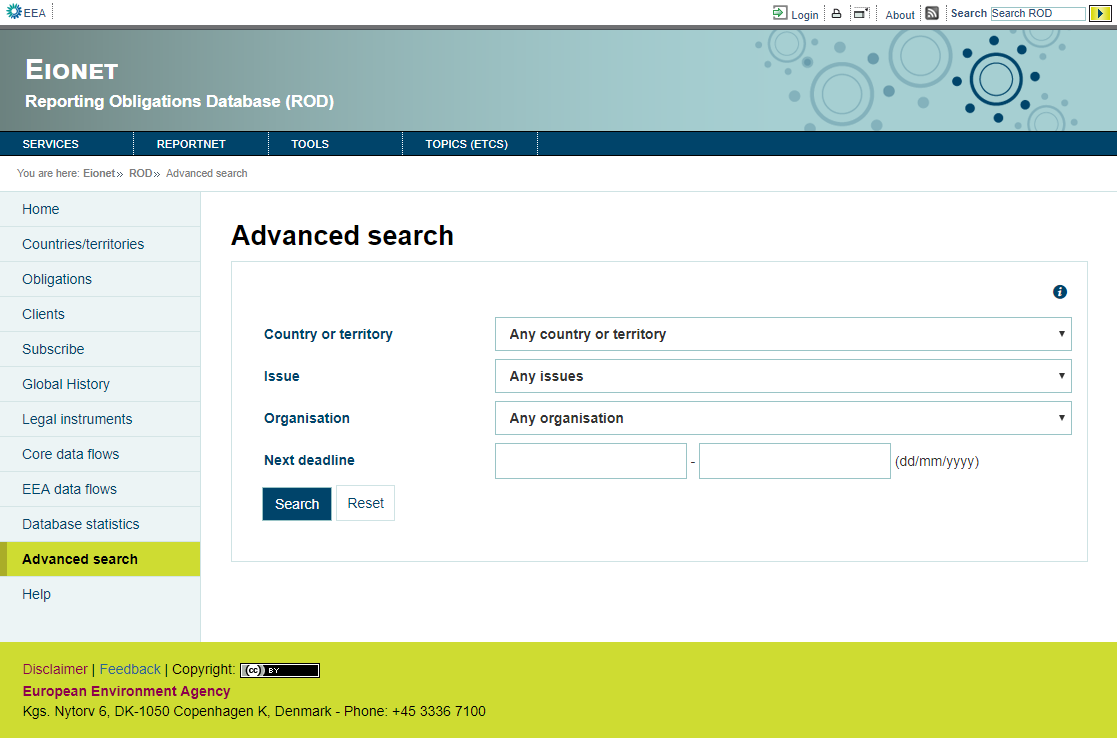


Figure 68

This form allows the list of deadlines to be refined to only show deadlines for a certain period or for an issue or organisation (Figure 68).  
  
That text deadlines will appear in all lists generated, as the application does not know when these are calendared for.

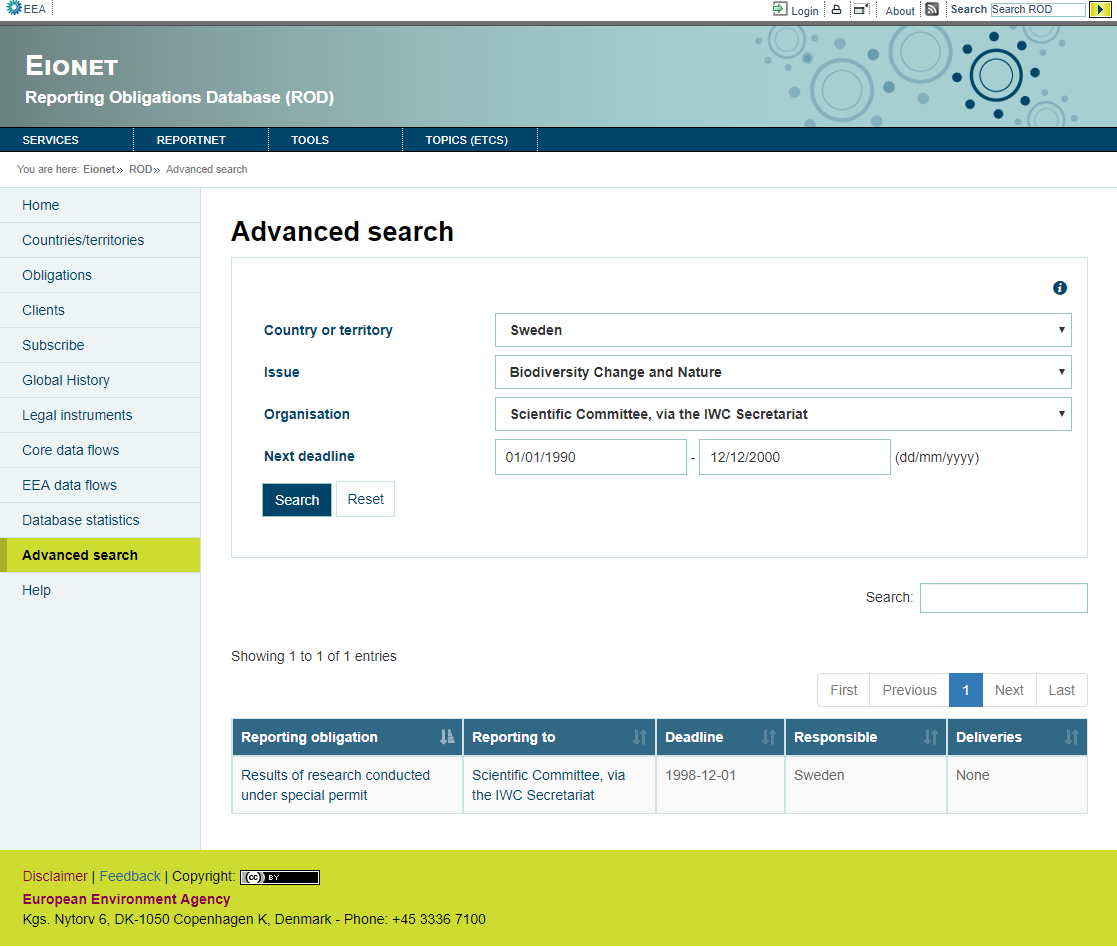
****

Figure 69

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - SpatialService: It calls the SpatialDao to use its methods.
    - ObligationService: It calls the ObligationsDao to use its methods.
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - IssueDao: Used to charge the Issue combo.
    - ClientService: Used to charge the Organisation combo.
    - SpatialDao: Used to charge the Country or territory combo.
    - ObligationsDao: Used to get all obligations or the searched obligations.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Spatial
    - Issue
    - ClientDTO
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.SearchController
  + View: search.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_obligation
  + t\_spatial
  + t\_issue
  + t\_client
  + t\_source
  + t\_client\_obligation\_lnk
  + t\_raspatial\_lnk
  + t\_role
  + t\_raissue\_lnk

## Help module

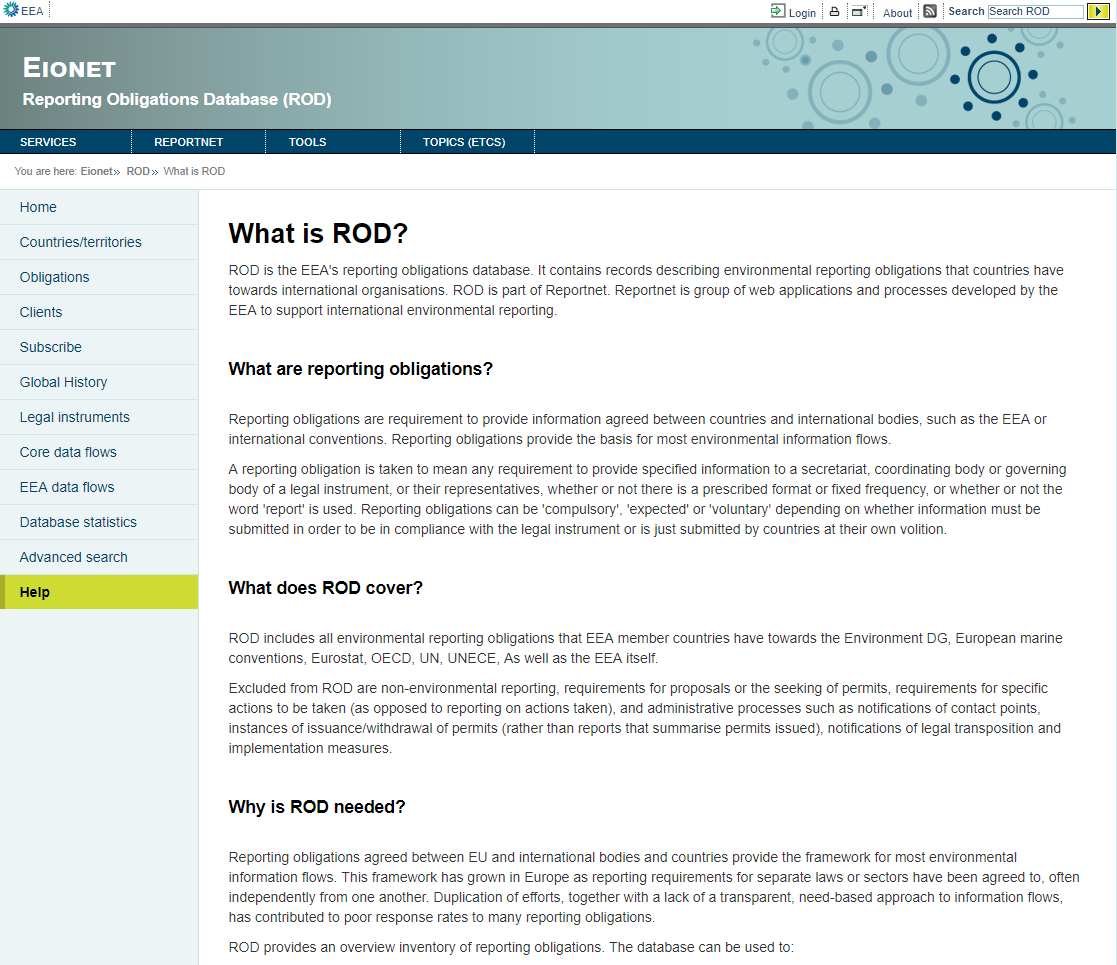


Figure 70

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.HelpDocController
  + View: help.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + actions:
    - RSS XML/RCP data extraction help
    - Disclaimer

DATABASE SPECIFICATIONS

* Doesn’t use tables of the database.

### RSS XML/RCP data extraction help

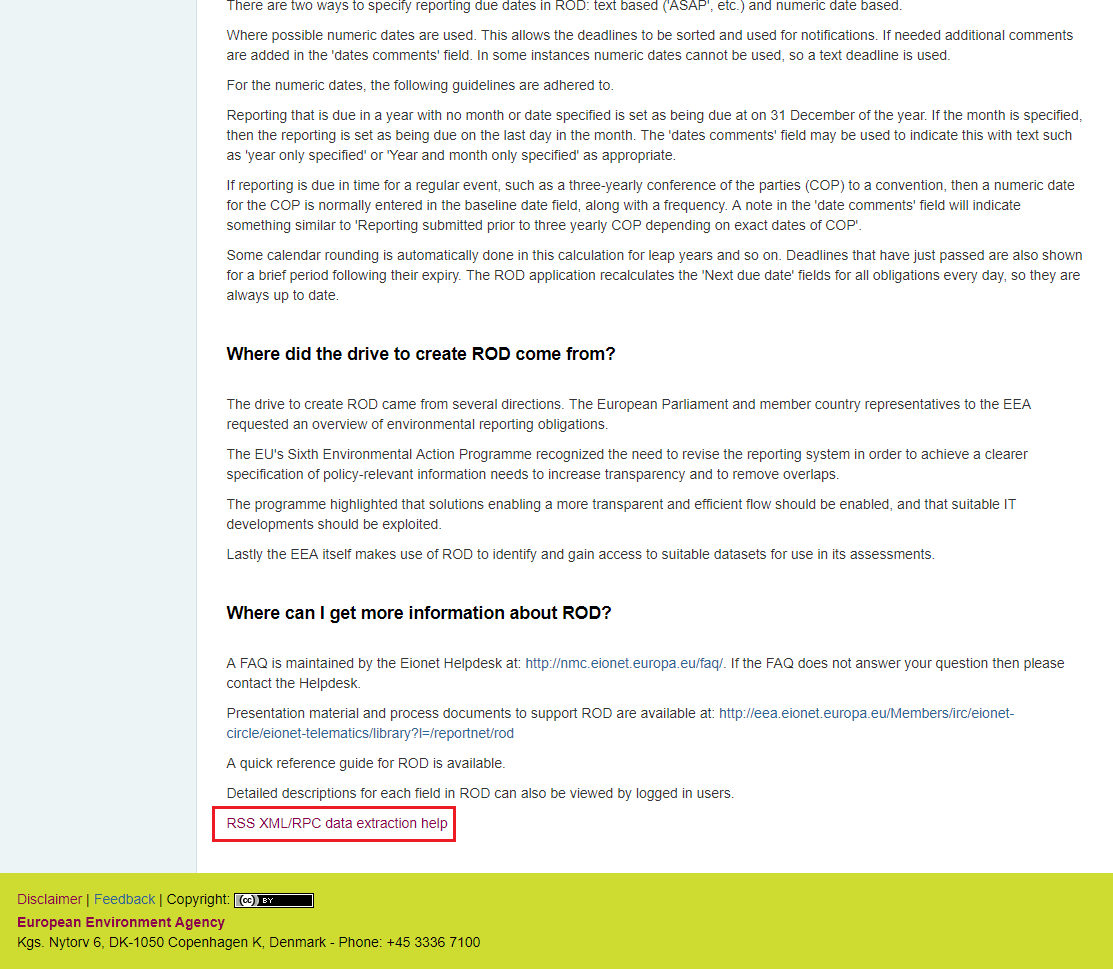
****

Figure 71

The application goes to /RSSHelp (Figure 72).

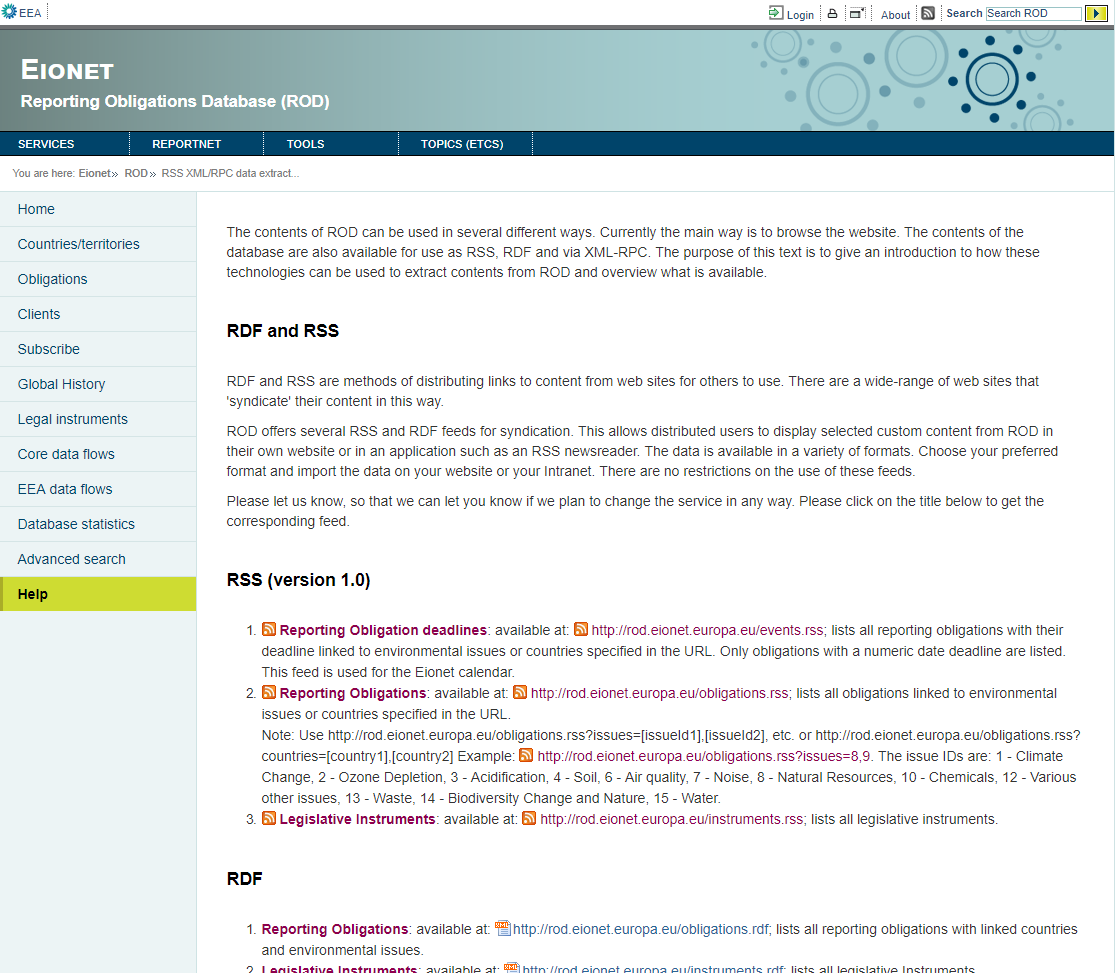


Figure 72

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.RSSHelpController
  + View: rssHelp.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* Doesn’t use tables of the database.

### Disclaimer

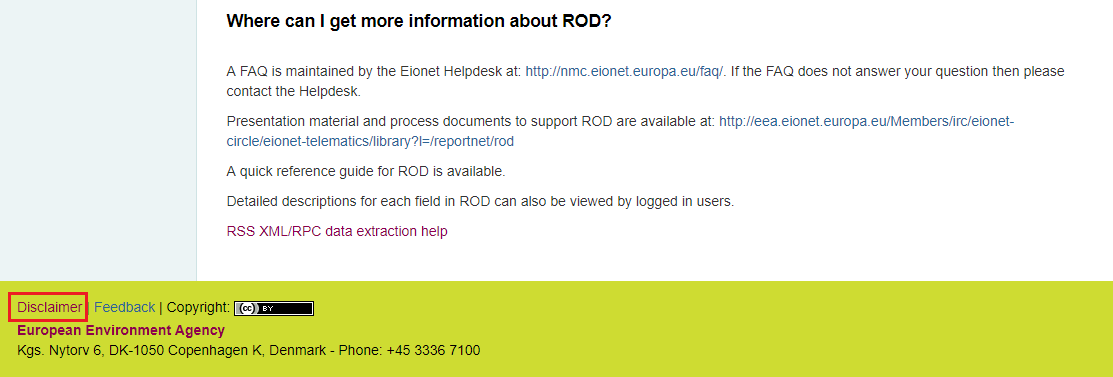


Figure 73

The application goes to /disclaimer (Figure 74).

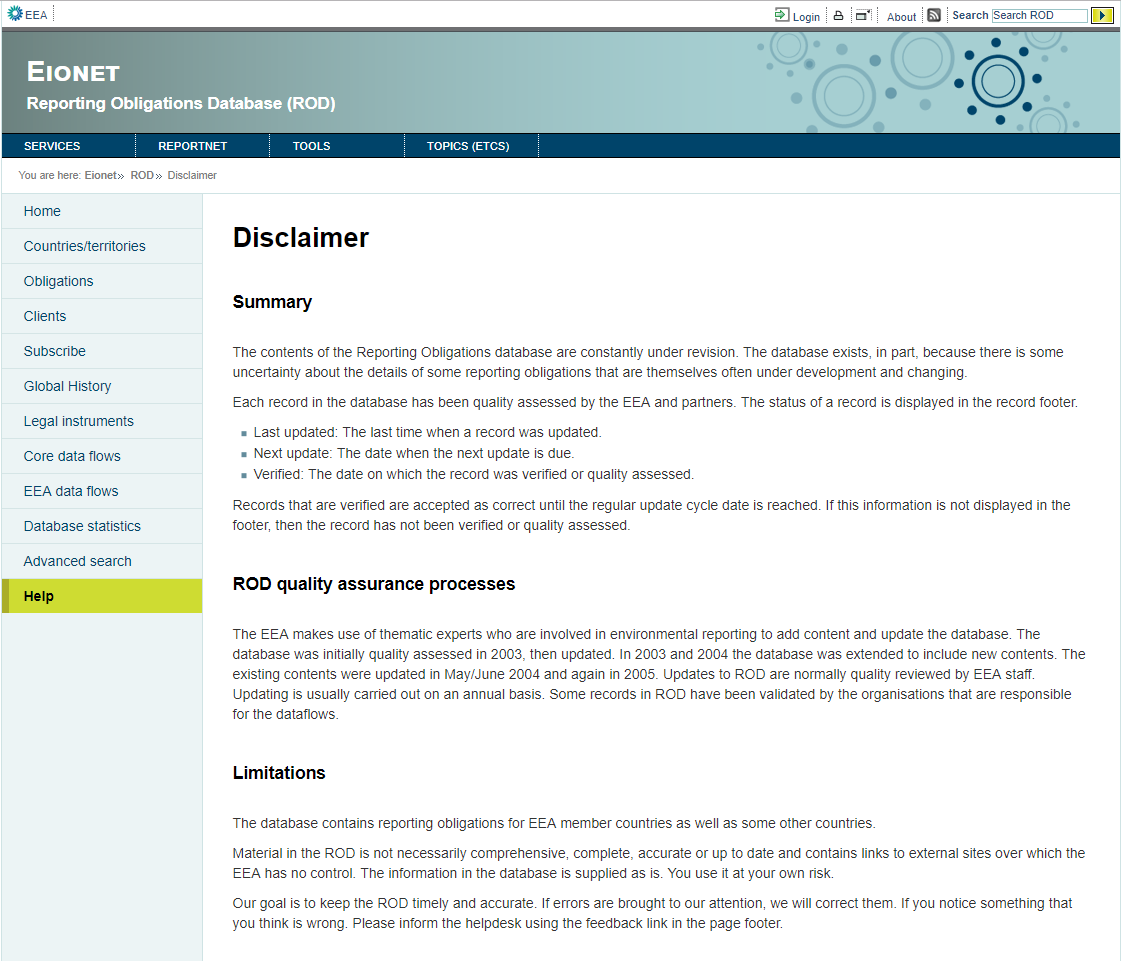


Figure 74

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
  + Data Model Layer (Using Data Transfer Object – DTO):
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.DisclaimerController
  + View: disclaimer.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* Doesn’t use tables of the database.

## Harvester

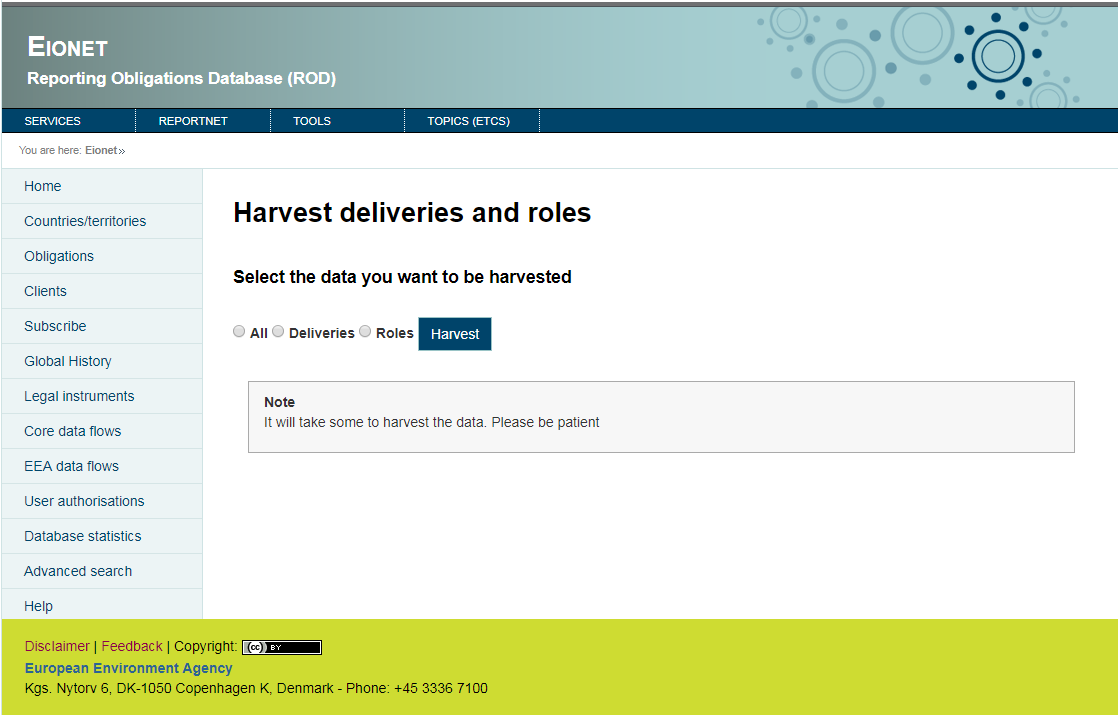


Figure 75

The data on deliveries and/or roles are automatically harvested.

BACKEND SPECIFICATIONS

* Framework Spring 4.1.6
* Inside the backend we distinguish the following layers/levels:
  + Services:
    - ObligationService: It calls the ObligationsDao to use his methods
    - RoleService: It calls the RoleDao to use his methods
    - DeliveryService: It calls the DeliveryDao to use his methods
    - UndoService: It calls the UndoDao to use his methods (comment in source)
    - FileServiceIF: It calls the eionet.rod.service.modules to use his methods
  + Data Access Layer (Using Data Access Object - DAO): References the following DAOs.
    - ObligationsDao: Used to search the responsible roles an coordinator roles.
    - RoleDao: Used to remove leftovers from previous harvest and save new roles.
    - DeliveryService: Used to back up currently existing deliveries and save the new deliveries.
  + Data Model Layer (Using Data Transfer Object – DTO):
    - Delivery
    - Roles
    - Obligations
* Folder structure inside Java project:
  + Controller: eionet.rod.controller.HarvesterController
  + Constants: eionet.rod. extractor.ExtractorConstants
  + Service: eionet.rod.extractor.extractorService
  + Other Services uses (eionet.rod.services):
    - ObligationService
    - RoleService
    - DeliveryService
    - UndoService
    - FileServiceIF
  + View: harvester.html

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)
* Inside the frontend we distinguish the following properties:
  + actions:
    - Harvest

DATABASE SPECIFICATIONS

* The following tables are used in this module:
  + t\_delivery
  + t\_role

### Harvest

Clicking the button Harvest the application goes to /harvester.

FRONTEND SPECIFICATIONS

* Thymeleaf HTML5, CSS3 and JavaScript (jQuery and jQuery-UI)

DATABASE SPECIFICATIONS

* Doesn’t use tables of the database.

# Data base model

CONCEPTUAL DATABASE MODEL

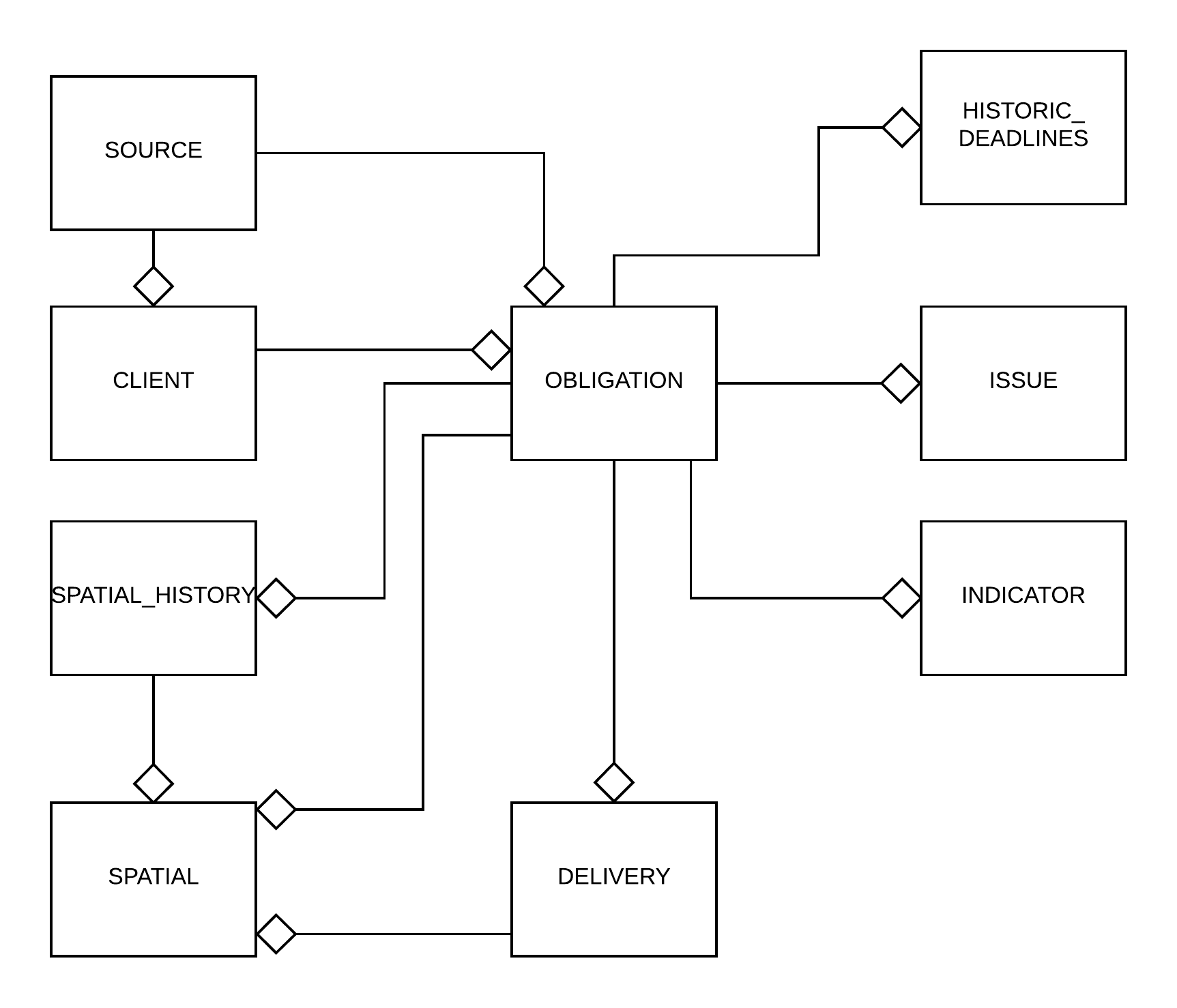


Figure 76

## UML diagram

Figure 77

## Data model Description

### HLP\_AREA

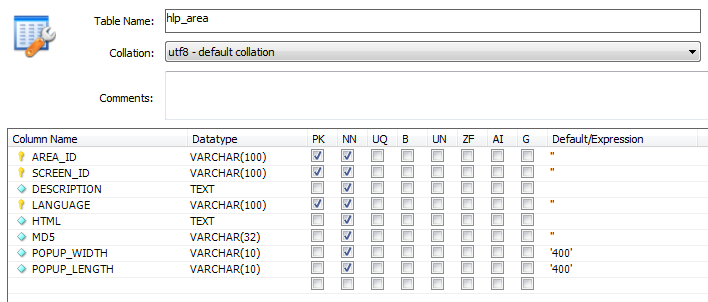


Figure 78

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| AREA\_ID | Varchar(100) | PK  Help identification | Yes |
| SCREEN\_ID | Varchar(100) | PK | Yes |
| DESCRIPTION | Text | Help description | Yes |
| LANGUAGE | Varchar(100) | PK  Help language | Yes |
| HTML | Text | Contents the text of HTML  Test in html of the help | Yes |
| MD5 | Varchar(32) |  | Yes |
| POPUP\_WIDTH | Varchar(10) | Contents the width of the help pop-up  Width of the help pop up  (not used) | Yes |
| POPUP\_LENGTH | Varchar(10) | Contents the length of the help pop-up  Height of the help pop up  (not used) | Yes |

### T\_CLIENT

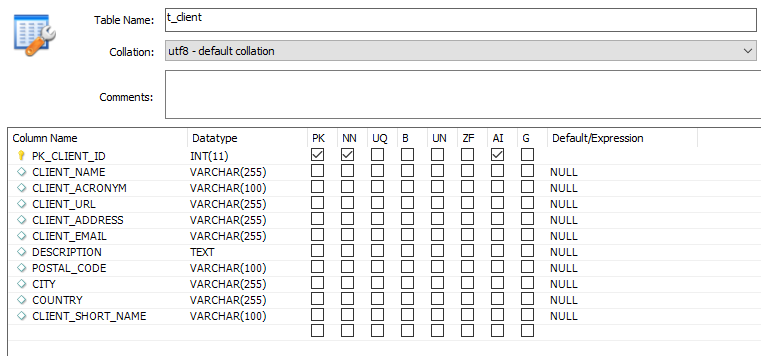


Figure 79

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| PK\_CLIENT\_ID | Int(11) | PK  Client Identification | Yes |
| CLIENT\_NAME | Varchar(255) | Name of client | No |
| CLIENT\_ACRONYM | Varchar(100) | Acronym of client | No |
| CLIENT\_URL | Varchar(255) | Url of client | No |
| CLIENT\_ADRESS | Varchar(255) | Adress of client | No |
| CLIENT\_EMAIL | Varchar(255) | Email of client | No |
| DESCRIPTION | Text | Description of client | No |
| POSTAL\_CODE | Varchar(100) | Postal code of the city where the client is | No |
| CITY | Varchar(255) | City where the client is | No |
| COUNTRY | Varchar(255) | Country where the client is | No |
| CLIENT\_SHORT\_NAME | Varchar(100) | Short name of client | No |

### T\_CLIENT\_OBLIGATION\_LNK

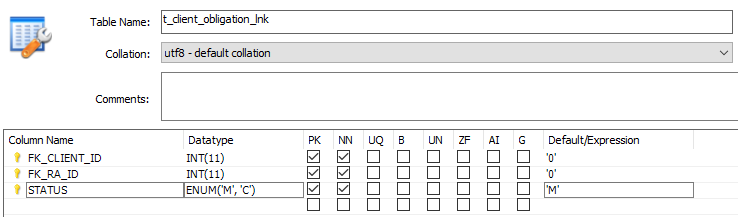


Figure 80

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_CLIENT\_ID | Int(11) | PK  Client Identification | Yes |
| FK\_RA\_ID | Int(11) | PK  Obligation Identification | Yes |
| STATUS | Enum('M', 'C') | PK  M - Report to client. This is the organisation to which the reported data set should be sent.  C - Other clients using this reporting. This field identifies organisations that make use of the information collected under this reporting obligation but who are not the official client organisation. Usually these organisations are actively involved in the reporting obligation. | Yes |

### T\_CLIENT\_SOURCE\_LNK

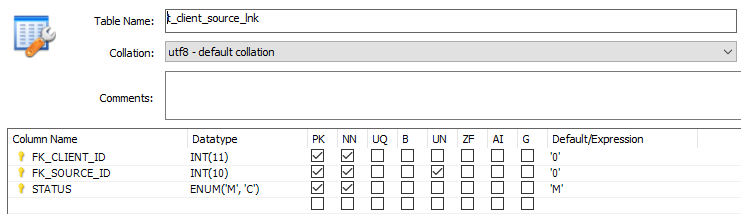


Figure 81

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_CLIENT\_ID | Int(11) | PK  Client Identification | Yes |
| FK\_SOURCE\_ID | Int(10) | PK  Instrument Identification | Yes |
| STATUS | Enum('M', 'C') | PK  M - Report to client. This is the organisation to which the reported data set should be sent.  C - Other clients using this reporting. This field identifies organisations that make use of the information collected under this reporting obligation but who are not the official client organisation. Usually these organisations are actively involved in the reporting obligation. | Yes |

### T\_DELIVERY

*Description*

Info on deliveries (reported data sets) sent by countries. The data on deliveries is automatically harvested by ROD3 on regular intervals from CIRCA Directory (role info; see *Role*) and Content Registry (delivery info). CR harvesting works as follows:

1. For any reporting obligation, Content Registry is queried with this obligation’s RA ID and all the country codes for countries that are required to report on this obligation.
2. The received information is stored in Delivery. For efficiency reasons, comma-separated list is also built for any given reporting obligation of country codes of countries that have deliveries for this obligation and stored in ReportingObligation’s ‘Deliveries from countries’ field (field 17).

For this to work, metadata for a delivery needs to contain RO ID and country code.

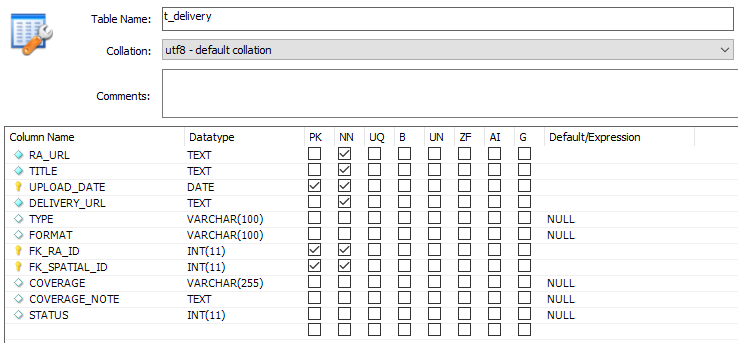


Figure 82

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| RA\_URL | TEXT | Obligation URl |  |
| TITLE | TEXT | Obligation Title |  |
| UPLOAD\_DATE | DATE | PK  Obligation date uploaded | Yes |
| DELIVERY\_URL | TEXT |  |  |
| TYPE | VARCAHR(100) |  |  |
| FORMAT | VARCAHR(100) |  |  |
| FK\_RA\_ID | INT(11) | PK  Obligation Identification | Yes |
| FK\_SPATIAL\_ID | INT(11) | PK  Country Identification | Yes |
| COVERAGE | Varchar(255) |  |  |
| COVERAGE\_NOTE | TEXT |  |  |
| STATUS | INT(11) |  |  |

### T\_ISSUE

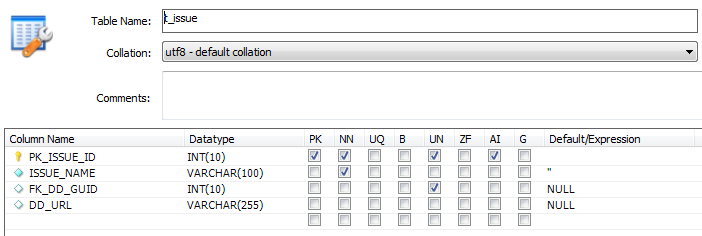


Figure 83

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_ISSUE\_ID | Int(11) | PK | Yes |
| ISSUE\_NAME | Varchar(100) | Name of the issue | Yes |
| FK\_DD\_GUID | INT(10) |  |  |
| DD\_URL | Varchar(255) |  |  |

### T\_SAPATIAL

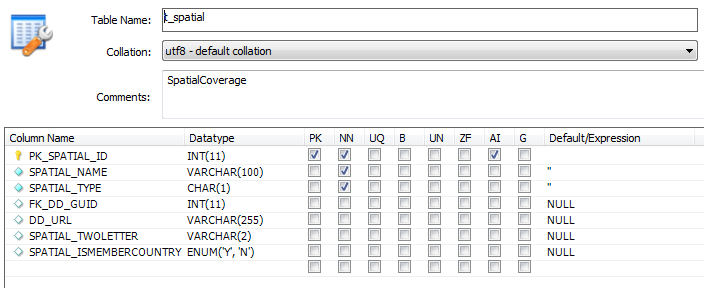


Figure 84

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| PK\_SPATIAL\_ID | Int(11) | PK  Country Identification | Yes |
| SPATIAL\_NAME | Varchar(100) | Country name |  |
| SPATIAL\_TYPE | CHAR(1) | Country type  Only used type = C |  |
| FK\_DD\_GUID | INT(10) |  |  |
| DD\_URL | VARCHAR(255) |  |  |
| SPATIAL\_TWOLETTER | VARCHAR(2) |  |  |
| SPATIAL\_ISMEMBERCOUNTRY | ENUM(Y,N) | Y - EEA member countries  N - Other countries and territories |  |

### T\_OBLIGATION

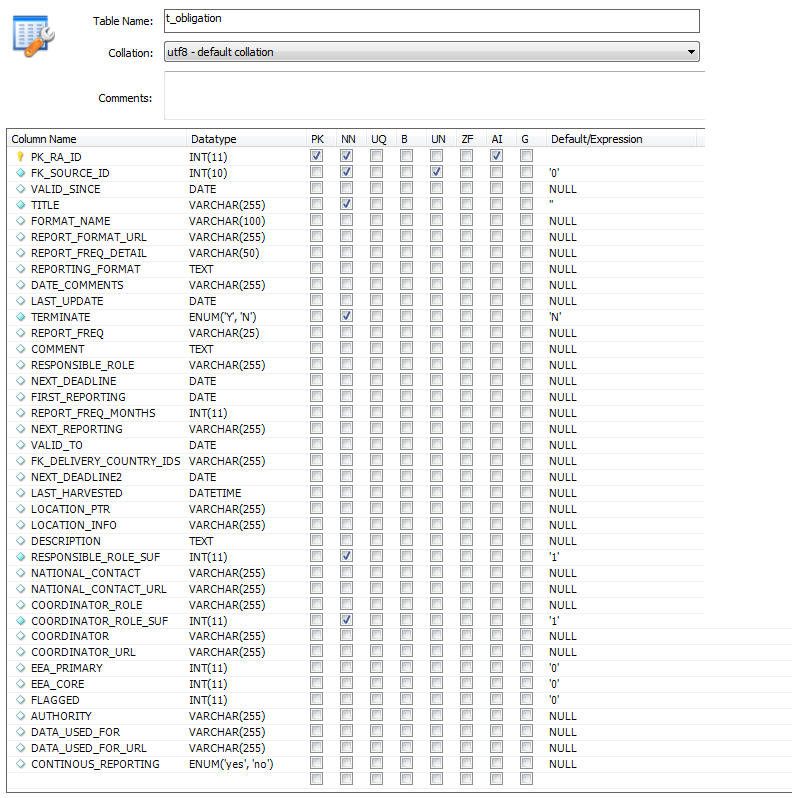


Figure 85

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| PK\_RA\_ID | Int(11) | PK  Obligation Identification | Yes |
| PK\_SOURCE\_ID | INT(11) | Instrument Identification | Yes |
| VALID\_SINCE | DATE | Format valid since - The date on which the reporting guidelines became valid. |  |
| TITLE | VARCHAR(255) | Title  This is a short identifier for the reporting obligation. | Yes |
| FORMAT\_NAME | VARCHAR(100) | Name of reporting guidelines  The name of the reporting guidelines that will appear as the hyperlink text for the link to the guidance document. |  |
| REPORT\_FORMAT\_URL | VARCHAR(255) | URL to reporting guidelines  The URL linking to the guidelines homepage or document. (This is sometimes password protected). The url should not return an error page.  URLs in ROD should start with the full address. (Example: http://www) |  |
| REPORT\_FREQ\_DETAIL | VARCHAR(50) |  |  |
| REPORTING\_FORMAT | TEXT | Reporting guidelines Extra info  This field displays extra information about the reporting guidelines. (older versions, format of submissions, extra URLs (not dynamic) etc). |  |
| DATE\_COMMENTS | VARCHAR(255) | Date comments  If the reporting dates can be entered as numerical dates but also require some explanatory text, then enter these additional notes here. |  |
| LAST\_UPDATE | DATE | Current timestamp |  |
| TERMINATE | ENUM(Y,N) |  | Yes |
| REPORT\_FREQ | VARCHAR(25) | Reporting frequency  See "Baseline reporting date" text above.  Enter a value as a whole number for the number of months between reporting due dates. For annual reporting this would be 12, for three yearly reporting, 36.  If reporting frequency in months is set to zero, the application assumes non-repeating date and uses "Baseline reporting date" as the "Next due date". |  |
| COMMENT | TEXT |  |  |
| RESPONSIBLE\_ROLE | VARCHAR(255) | National reporting contacts  This field should be filled with the details of the technical contact for the reporting for this obligation. This is the person who will prepare and upload the reported information. The contact details can be entered into ROD in different ways.  Ideally, it should be a CIRCA role making use of country suffixes. The CIRCA role prefix (without the country-specific part) should be entered (for example eionet-nrc-acc). Generally, the "national contacts" role should be the NRC for Eionet dataflows and the "national coordinator" should be the PCP Eionet role.  On saving a record in ROD, the role is checked against the CIRCA directory. Non-existent roles will result in an error message. A list of all roles in the directory can be found at: [http://eea.eionet.eu.int:8980/Members/irc/eionet-c...](http://eea.eionet.eu.int:8980/Members/irc/eionet-circle/Home/central_dir_admin?fn=roles&rd=0&af=0&ud=0&od=0&act=srch&v=*)  Alternatively a central CIRCA role may be the contact point for the list of coordinators for all countries. The third option is that the name of the list with a URL pointing to where the list of contacts for this reporting obligation is available on the Internet can be used. These options and the 3 fields: role, name and suffix should cater for most situations. |  |
| NEXT\_DEADLINE | DATE | Next due date  Values entered in the "Baseline reporting date" field and the "Reporting frequency in months" field. Are used to calculate the "Next due date" according to the following rules (assuming "today" is 05-12-2002 in our example):  If the day of the starting date is < 28, a number of months specified in reporting frequency is added to the starting date repeatedly, until "Next due date" is in the future. For example, with 15-09-2002 start date and reporting frequency of 3 months, next reporting will be 15-12-2002.  If the day of the starting date is >= 28, a number of months specified in reporting frequency is added to the starting date, until "Next due date" is in the future1, and the day set to the end of the month. For example, with 30-11-2002 start date and reporting frequency of 3 months, next reporting will be 28-02-2003.  If reporting frequency in months is set to zero, the application assumes non-repeating reporting date (one-off) and uses "First reporting" date as the "Next due date". The "Next due date" will never be larger than "Valid to" date (no need to report if the obligation is no longer valid).   The application recalculates the "Next due date" fields for all obligations every day, so they are always up to date. |  |
| FIRST\_REPORTING | DATE | Baseline reporting date  The baseline reporting date should contain the first reporting date (including any subsequent amendments to the date). |  |
| REPORT\_FREQ\_MONTHS | INT(11) | Reporting frequency  A number of months between reporting due dates. For annual reporting this would be 12, for three yearly reporting, 36.  If reporting frequency in months is set to zero, the application assumes non-repeating date and uses "Baseline reporting date" as the "Next due date". |  |
| NEXT\_REPORTING | VARCHAR(255) | Text dates  Text based ("ASAP", etc.) and numeric date based. Only one method can be used at once. This field is for the text-based format. |  |
| VALID\_TO | DATE | "Valid to" is a date field that specifies how long the reporting obligation is going to be valid. After a value for the "Baseline reporting date" field has been entered the application suggests a default value of "31/12/9999" for the valid to date.  If the reporting obligation is for a limited period, a one-off reporting or will be repealed then the end validity date should be entered.  When a reporting obligation is no longer active, the user interface displays "terminated". The "valid to" date should be set as one year after the reporting date for one-off reporting. |  |
| FK\_DELIVERY\_COUNTRY\_IDS | VARCHAR(255) | Country delivery Id separated by a semicolon. |  |
| NEXT\_DEADLINE2 | DATE |  |  |
| LAST\_HARVESTED | DATETIME |  |  |
| LOCATION\_PTR | VARCHAR(255) | URL to principle repository  This field is for the url to the repository where countries make deliveries in response to the reporting obligations. This may be a CDR url, a CIRCA url, an ftp address, an url to a web form, to a database, a downloadable application or some other Internet service.  URLs in ROD should start with the full address. Example: [http://www](http://www/) |  |
| LOCATION\_INFO | VARCHAR(255) | Name of repository  The name of the repository where countries make deliveries (for electronic reporting) should be entered here. |  |
| DESCRIPTION | TEXT | Short description of the reporting obligation to expand on the title. Guide length: three lines or less. |  |
| RESPONSIBLE\_ROLE\_SUF | INT(11) | If checked do not append a country suffix | Yes |
| NATIONAL\_CONTACT | VARCHAR(255) | Name of the national reporting contacts. |  |
| NATIONAL\_CONTACT\_URL | VARCHAR(255) | Url of the national reporting contacts |  |
| COORDINATOR\_ROLE | VARCHAR(255) | National reporting coordinators  This field should be filled with the details of the coordinator for the reporting for this obligation. This information can be entered into ROD in different ways.   Ideally, it should be a CIRCA role making use of country suffixes. The CIRCA role prefix (without the country-specific part) should be entered (for example eionet-pcp-acc). Generally, the "national contacts" role should be the NRC for Eionet dataflows and the "national coordinator" should be the PCP Eionet role.  On saving a record in ROD, the role is checked against the CIRCA directory. Non-existent roles will result in an error message. A list of all roles in the directory can be found at: [http://eea.eionet.eu.int:8980/Members/irc/eionet-c...](http://eea.eionet.eu.int:8980/Members/irc/eionet-circle/Home/central_dir_admin?fn=roles&rd=0&af=0&ud=0&od=0&act=srch&v=*)  Alternatively a central CIRCA role may be the contact point for the list of coordinators for all countries. The third option is that the name of the list with a URL pointing to where the list of contacts for this reporting obligation is available on the Internet can be used. These options and the 3 fields, Role, name and suffix should cater for most situations |  |
| COORDINATOR\_ROLE\_SUF | INT(11) | If checked do not append a country suffix | Yes |
| COORDINATOR | VARCHAR(255) | Name of the national reporting contacts |  |
| COORDINATOR\_URL | VARCHAR(255) | Url of the national reporting contacts |  |
| EEA\_PRIMARY | INT(11) | Eionet core data flow (priority dataflow flag).  This tick-box should be selected if the reporting obligation is a priority dataflow for the EEA. |  |
| EEA\_CORE | INT(11) | Used for EEA Core set of indicators (core set of indictors flag)  This tick-box should be selected if the reporting obligation is used to create an indicator that is used in the EEA core set of indicators. |  |
| FLAGGED | INT(11) | Delivery process is managed by EEA.  This tick-box should be selected if reporting under this obligation is an EEA managed data flow (EDF). |  |
| AUTHORITY | VARCHAR(255) | Authority giving rise to the obligation  This field should contain the "authority" for the obligation? E.g. what Article, Decision, Recommendation, etc. gives rise to the Obligation. This is separate and distinct from any "reporting guidelines" or "formats" and is not an organisation. (Example: Article 1, paragraph ii).  This field may not be appropriate for all obligations. If this information is in the title of the reporting obligation, it should be removed from the title and placed in this field. |  |
| DATA\_USED\_FOR | VARCHAR(255) |  |  |
| DATA\_USED\_FOR\_URL | VARCHAR(255) |  |  |
| CONTINOUS\_REPORTING | ENUM(yes,no) |  |  |

### T\_RASSPATIAL\_LNK

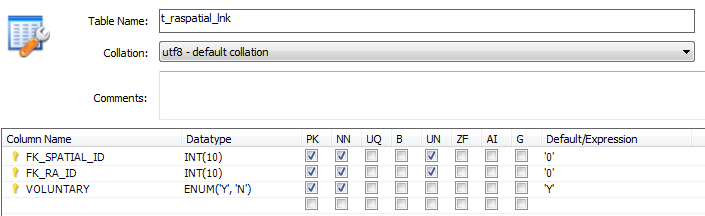


Figure 86

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_SPATILA\_ID | Int(10) | PK  Country Identification | Yes |
| FK\_RA\_ID | INT(10) | Obligation Identification | Yes |
| VOLUNTARY | ENUM(Y,N) |  | Yes |

### T\_RAISSUE\_LNK

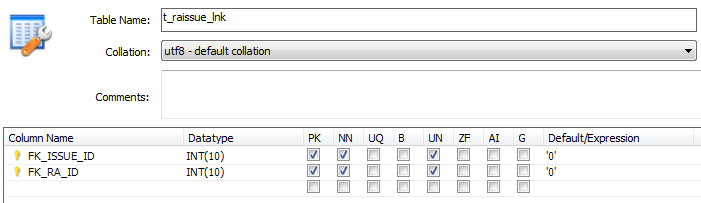


Figure 87

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_IISUE\_ID | Int(10) | PK  Issue Identification | Yes |
| FK\_RA\_ID | INT(10) | Obligation Identification | Yes |

### T\_ROLE

*Description*

Role info from CIRCA Directory. The data is automatically harvested by ROD on regular intervals. The process works as follows:

1. First, for any given obligation, countries that are required to report on this obligation are selected and their country codes fetched from SpatialCoverage.
2. (Optional: Full role id’s are then built from given obligation’s role prefix and country codes.)
3. CIRCA Directory is called with role id’s and the received info stored in Role.

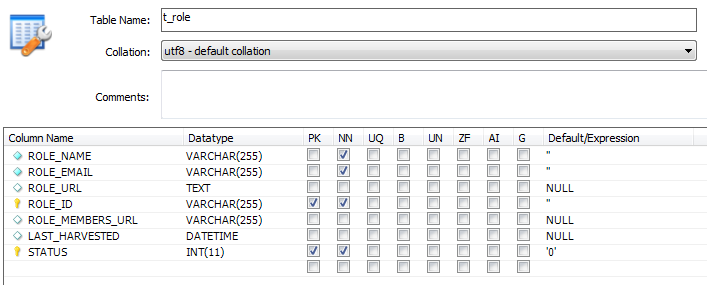


Figure 88

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| ROLE\_NAME | VARCHAR(255) | Name of the role | Yes |
| ROLE\_EMAIL | VARCHAR(255) | Email of the role | Yes |
| ROLE\_URL | TEXT | URL of the role |  |
| ROLE\_ID | VARCHAR(255) | PK  Identification Role | Yes |
| ROLE\_MEMBERS\_URL | VARCHAR(255) |  | No |
| LAST\_HARVESTED | DATETIME |  | No |
| STATUS | INT(11) | PK | Yes |

### T\_UNDO

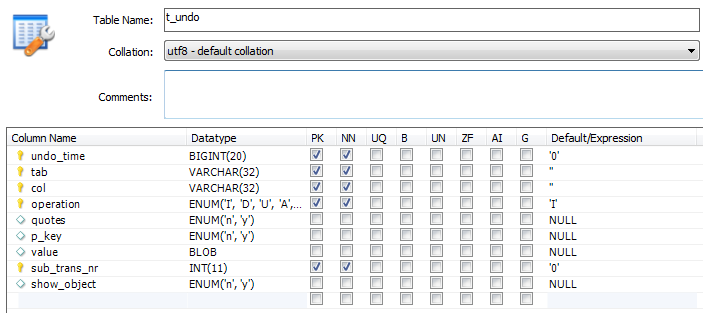


Figure 89

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| UNDO\_TIME | BIGINT(20) | PK | Yes |
| TAB | VARCHAR(32) | PK  Table alter | Yes |
| COL | VARCHAR(32) | PK  Field alter | Yes |
| OPERATION | ENUM('I', 'D', 'U', 'A', 'L', 'K', 'O', 'UN', 'UD', 'UDD', 'T', 'ACL') | I – Insert  D – Delete  U - Update | Yes |
| QUOTES | ENUM(y,n) |  | No |
| P\_KEY | ENUM(y,n) |  | No |
| VALUE | BLOB | Value alter | Yes |
| SUB\_TRANS\_NR | INT(11) |  | Yes |
| SHOW\_OBJECT | ENUM(y,n) |  | No |

### T\_OBLIGATION\_RELATION

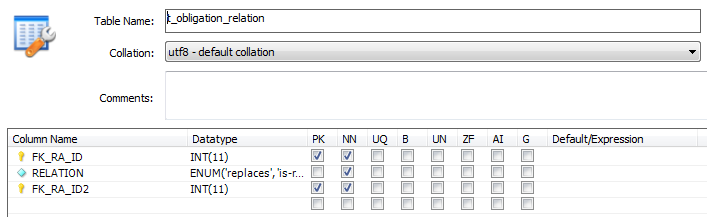


Figure 90

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_RA\_ID | INT(11) | PK  Obligation Identification | Yes |
| RELATION | ENUM('replaces', 'is-replaced-by', 'see-also', 'same-as') | Replaces  Is Replaced by  Same as | Yes |
| FK\_RA\_ID2 | INT(11) | PK  Identification of the obligation relationed | Yes |

### T\_SOURCE\_LNK

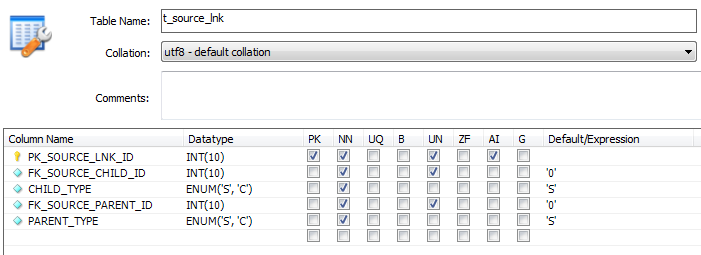


Figure 91

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_SOURCE\_LNK\_ID | INT(10) | PK  Instrument Identification | Yes |
| FK\_SOURCE\_CHILD\_ID | INT(10) | Instrument Identification | Yes |
| CHLID\_TYPE | ENUM(S,C) |  | Yes |
| FK\_SOURCE\_PARENT\_ID | INT(10) | Instrument Identification | Yes |
| PARENT\_TYPE | ENUM(C,S) |  | Yes |

### T\_SOURCE\_CLASS

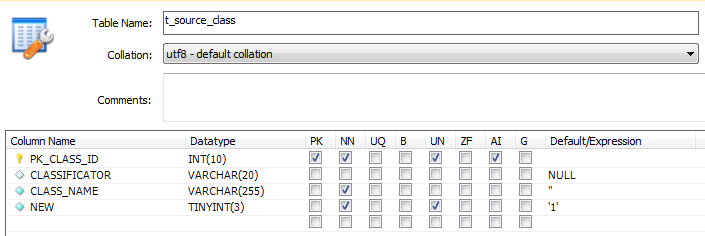


Figure 92

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| FK\_CLASS\_ID | INT(10) | PK  Eur-lex Categories identification | Yes |
| CLASSIFICATOR | VARCHAR(20) |  | No |
| CLASS\_NAME | VARCHAR(255) | Name of the Eur-lex Categories | Yes |
| NEW | INYINT(3) | 1 – New  0 – not new | Yes |

### T\_SOURCE

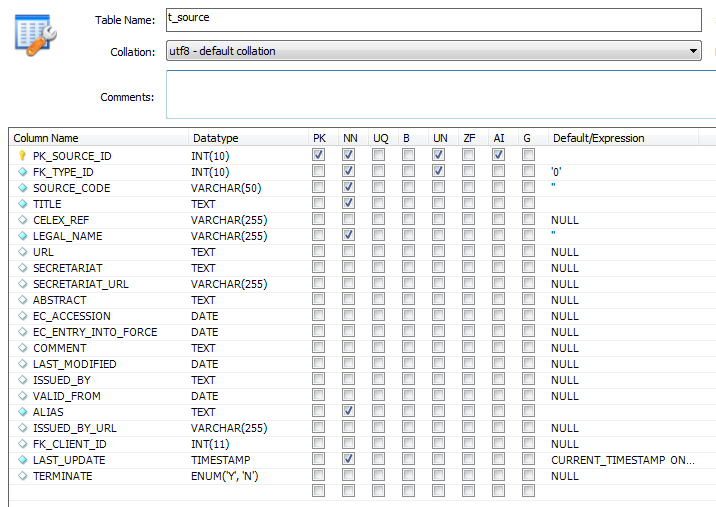


Figure 93

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Description | Required |
| PK\_SOURCE\_ID | INT(10) | PK  Instrument identification | Yes |
| FK\_TYPE\_ID | INT(10) | Type of instrument | Yes |
| SOURCE\_CODE | VARCHAR(50) | Identification number  The ID number for the legal instrument. For EU legislation, this is normally also in the title. Ex. 92/72/EEC.  Some instruments do not have an ID number. | Yes |
| TITLE | TEXT | Instrument title | Yes |
| CELEX\_REF | VARCHAR(255) | Celex reference  Every document in the Celex database has a unique identification number known as the Celex document number. This can be found in the legal text, in the Eurlex database, or in Celex and should be entered in this field for all documents legislative instrument records that are in Celex.  This field should be left blank for legislative instruments not in Celex.  The Celex number is an alphanumeric sequence consisting of four principal elements. These are: \* the sector (one character) [S]; \* the year (four digits) [YYYY]; \* the type of document code (one or two-character code) [T(T)]; \* a sequential number representing the original reference number of the act where possible (natural number) [NNN(N)]. Example: 41993Y0517(01) | No |
| LEGAL\_NAME | VARCHAR(255) | Legal Name  The full legal name of the legislative instrument. | Yes |
| URL | TEXT | URL to the official source  A web link to the Eur-lex entry or the text of conventions and other international agreements should be entered here. Where possible this should be to a stable non-password protected URL that links to pdf files displaying all tables and annexes.  URLs in ROD should start with the full address. Example: [http://www](http://www/) | No |
| SECRETARIAT | TEXT | Convention secretariat  Name of the convention secretariat. | No |
| SECRETARIAT\_URL | VARCHAR(255) | URL to Secretariat homepage  URL of the convention secretariat homepage. URLs in ROD should start with the full address. Example: [http://www](http://www/) | No |
| ABSTRACT | TEXT | A short abstract of the legislation should be put here. The focus should be on defining what the legislation covers from a reporting obligation perspective. Where possible the text should be taken directly from an authoritative summary such as from the objective in EU legislative texts. | No |
| EC\_ACCESSION | DATE | For conventions where the EC or EU is a party to the convention, the date of EC accession to a convention. Leave blank if not applicable. | No |
| EC\_ENTRY\_INTO\_FORCE | DATE | For conventions ratified by the EC / EU, the date on which convention entered into force for the EC / EU should be entered here. The date directives enter into force is 20 days after the date of publication in the Official Journal if nothing else is mentioned. | No |
| COMMENT | TEXT | Put any additional comments or remarks here. Comments are often anticipated revision dates, notes about the relationship between this instrument and other legislation, the history of significant changes, etc. | No |
| LAST\_MODIFIED | DATE |  | No |
| ISSUED\_BY | TEXT | This field should be filled from the list of organisations in the drop down list. If the issuing organisation for the legislation is not listed, create a new organisation using the "new" button.  Please think carefully whether you need to create a new organisation before doing so. Please enter all details for the new organisation in the window that opens. Note: The "issued by" organisation has a separate field from the "report to" or client organisation in ROD.  The "issued by" field is for the organisation responsible for the legislation, whereas the "report to" field is for the organisation responsible for receiving the flow of information under the reporting obligation. Often these are the same organisation. | No |
| VALID\_FROM | DATE | The date on which the legislation enters into force: the date on which the act became operative. The date directives enter into force is 20 days after the date of publication in the Official Journal if nothing else is mentioned. This date is for reference and analysis only, and is not used in any calculations for reporting obligation deadlines. | No |
| ALIAS | TEXT | The short name by which the legislative instrument is known. Example: Birds directive. | Yes |
| ISSUED\_BY\_URL | VARCHAR(255) | The URL to issuer's web page for this specific legal instrument should be entered here.  The URL entered in this field should be different from the organisational url entered in the issuer?s details. The URL in this field should point to an introductory information page about the legislative instrument, if one exists.  URLs in ROD should start with the full address. Example: [http://www](http://www/) | No |
| FK\_CLIENT\_ID | INT(11) | Client identification | No |
| LAST\_UPDATE | TIMESTAMP | Current timestamp | Yes |
| TERMINATE | ENUM('Y', 'N') |  | No |

1. ROD3 url: <http://rod3.devel1dub.eionet.europa.eu/> [↑](#footnote-ref-1)
2. ROD url: <http://rod.eionet.europa.eu/> [↑](#footnote-ref-2)