WIZ – the web portal

PROJECT

Table of Contents

INTRODUCTION 6
The framework 6
The informative property 8
ARCHITECTURE 10
THE DATABASE 13
notifications 15
notification_categories 15
settings 15
users 15
roles 16
water_qualities 16
water_quality_options 17
water_faults 17
water_fault_options 17
water_requests 18
water_request_history 18
water_request_geometries 18
water_request_geometry_zones 19
water_request_geometry_zone_properties 19
water_request_parameters 19
zones 20
zones_water_request_parameters 20
water_demand_formulas 20
water_supply 20
city_state_boundaries 21
operative_margin 21
service_area 21
service_area_operative_margin 22
plugins 22
sendmail_logs 23
ACCESS CONTROL 23
THE NOTIFICATIONS 25
PLUGINS 26
THE USERS 27
WATER RESOURCE REQUESTS 29
EPANET INTERFACE 42
PARTICIPATED QUALITY 45
THE GEOGRAPHICAL INFORMATION 46
LOG & FEEDBACK 47
INTERNATIONALISATION 48
PARAMETERS CONFIGURATION 49
GLOSSARY 54

APPENDIX A: DATABASE SCHEME 55

List of Figures

Figure 1: MVC 7

Figure 2: Yii Application 8

Figure 3: Architecture 11

Figure 4: Workflow of the application 12

Figure 5: Database 14

Figure 6: Graph with accesses control 25

Figure 7: Structure of the notifications 26

Figure 8: Login procedure 29

Figure 9: Relationship between water_requests, water_request_geometries and water_request_geometry_zones 30

Figure 10: Procedure for calculating the population equivalent and the water demand 31

Figure 11: Formulas for calculating the water demand 32

Figure 12: Parameters, conversion indexes and their relations with the intended land uses 33

Figure 13: Intended land uses hierarchically organised 34

Figure 14: Procedure for the recovery of the conversion index for the population equivalent calculation 35

Figure 15: Procedure for the recovery of the formula for the water demand calculation 36

Figure 16: Assessment of the conversion indexes 37

Figure 17: Assessment of the formulas for the water demand calculation 38

Figure 18: Detail of the water demand calculation 39

Figure 19: Comparison between water demand and operative margin 41

Figure 20: Fields of a Junction, according to the EPANET specifications....................43

Figure 21: Structure of an EPANET model file 44

Figure 22: Generation of a new EPANET file 45

Introduction

The framework

The WIZ web portal is based on a open source framework called Yii[1].

Yii is a framework used for web programming and can be used for the development of any kind of web application. It is written in PHP and is very performing thanks to sophisticated caching mechanisms, especially for applications with a lot of traffic.

Yii implements the design pattern model-view-controller (MVC), extensively used in web programming. MVC allows to separate the login from the user interface, as a result IT developers can easily modify the first one without affecting the second one.

[1] http://www.yiiframework.com/

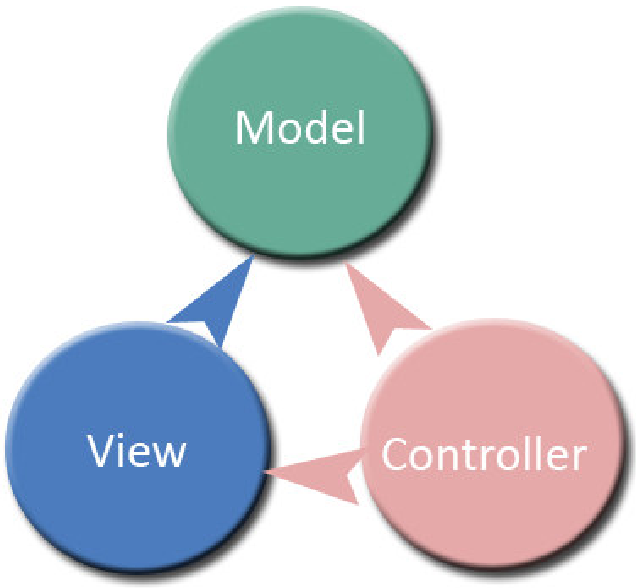


Figure 1: MVC

In MVC, the model is represented by the information (i.e. by data), the views contain user interface elements such as text or input data and the controller manages the communication between the model and the views.

The chart in Figure 2 sketches the structure of a Yii application.

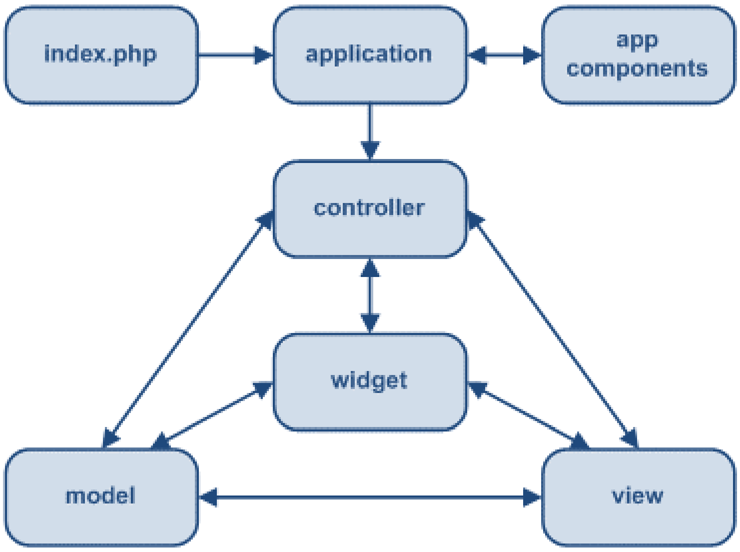


Figure 2: Yii Application

The controller is created whenever a user invokes it and executes the requested action, using data stored in the model and visualising it through an appropriate view.

The model represents a single data. It can be either a row of a table of the database or a input field of a form. Each data field is represented by a model attribute.

The view is a script in PHP that contains user interface elements such as text or input fields. Although the view can carry out also operations on data which it displays, we recommend carrying out these operations on the controller in order to have a clear separation between logic and interface.

<u>The informative property</u>

The system needs a serious of contextual data, either geographical and non-geographical, in order to work. Some of this data will physically populate the system while others will be used upon request by accessing the W*S services according to the OGC standard.

The minimum group of necessary data requested is composed of:

- Administrative limits: reports, in a shape format, the administrative limits of the city-states managed by Acque SPA.
- Service areas: reports, in a shape format, the service areas borders in which the city-states are subdivided.
- Availability Table: indicates for both, the city-states and the service areas, which is the current operative margin and possible one or more future margins.
- Consumptions table: indicated for each city-state, which is the average yearly and maximum daily water resource consumption
- DEM: Digital Elevation Model

This data is physically located in the system; its territorial coverage has to be at least equal to the land portion Acque SPA manages.

In order to make the system more appealing and to provide users more information, the following is also requested:

- Orthophotos: geo-referenced aerial photographs. These are uploaded upon request benefiting from the GEOscopio service offered by the Tuscan Region
- RTM: regional technical map. Is uploaded upon request benefiting from the GEOscopio service offered by the Tuscan Region
- Distribution network: contains georeferenced information on the water distribution network with its technical features and service characteristics
- Sources: shows the exact sources location, subdivided in:
    - Water flows
    - Lakes
    - Wells
    - Springs
- Facilities: show the exact location of the facilities, subdivided in:

    - Water purification system
    - adductions
    - accumulation
    - pumping station
- Street map: georeferenced street names, squares and places.

The Orthophotos and the RTM are external data, which are uploaded upon request benefiting from the GEOscopio offered by the Tuscan Region; all other data is physically located on the system instead.

As this is considered to be contextual data and as such not essential for the correct system operation, its coverage can also be less than the land portion Acque SPA manages; however, it is desirable that also for this data coverage reaches at least a value equal to the one of the city-states managed by Acque SPA.

Architecture

The system was designed to interact with a geographical basis of data. Data is stored in a Postgres database with the extension PostGIS.

A map server called GeoServer is used, to visualise geographical information and make it available to third parties through the OGC standard by using WFS/WMS services.

The architecture is completed by a group of open source JavaScript libraries by which the map server can be easily queried and functionalities added to the user interface. Figure 3 broadly represents a workflow with the architecture.

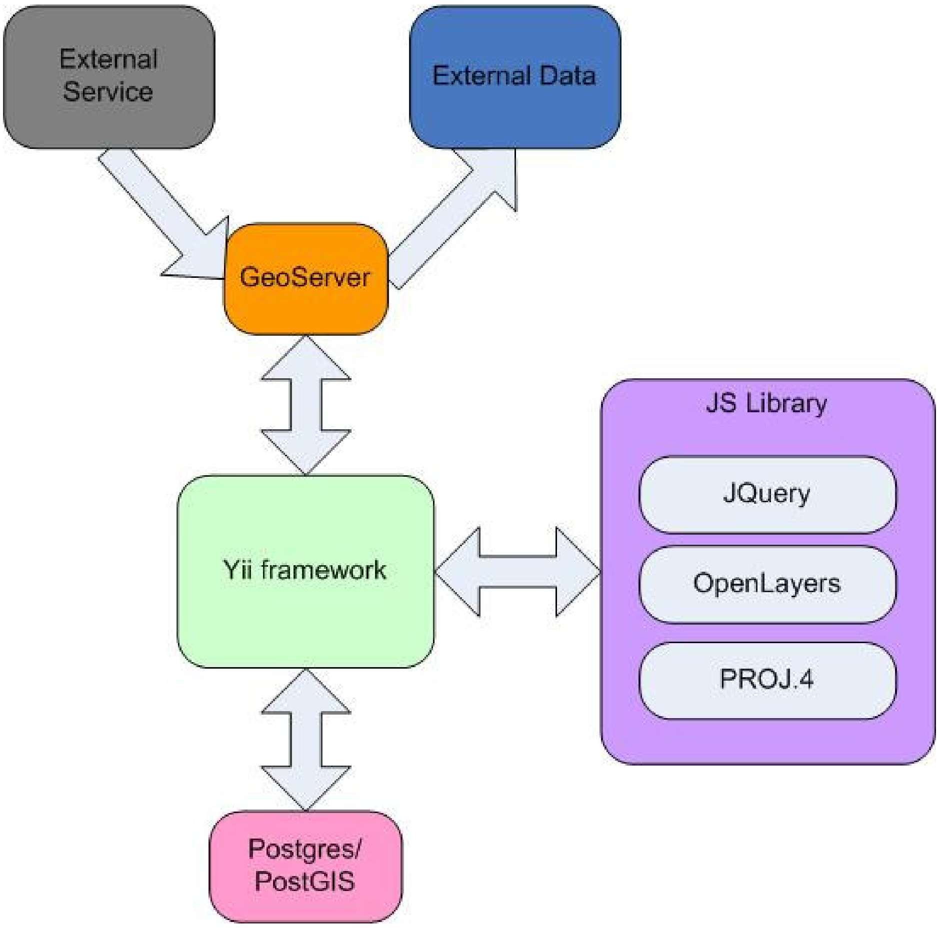


Figure 3: Architecture

Each time a request is submitted, the system starts possessing using all or part of the architecture components.

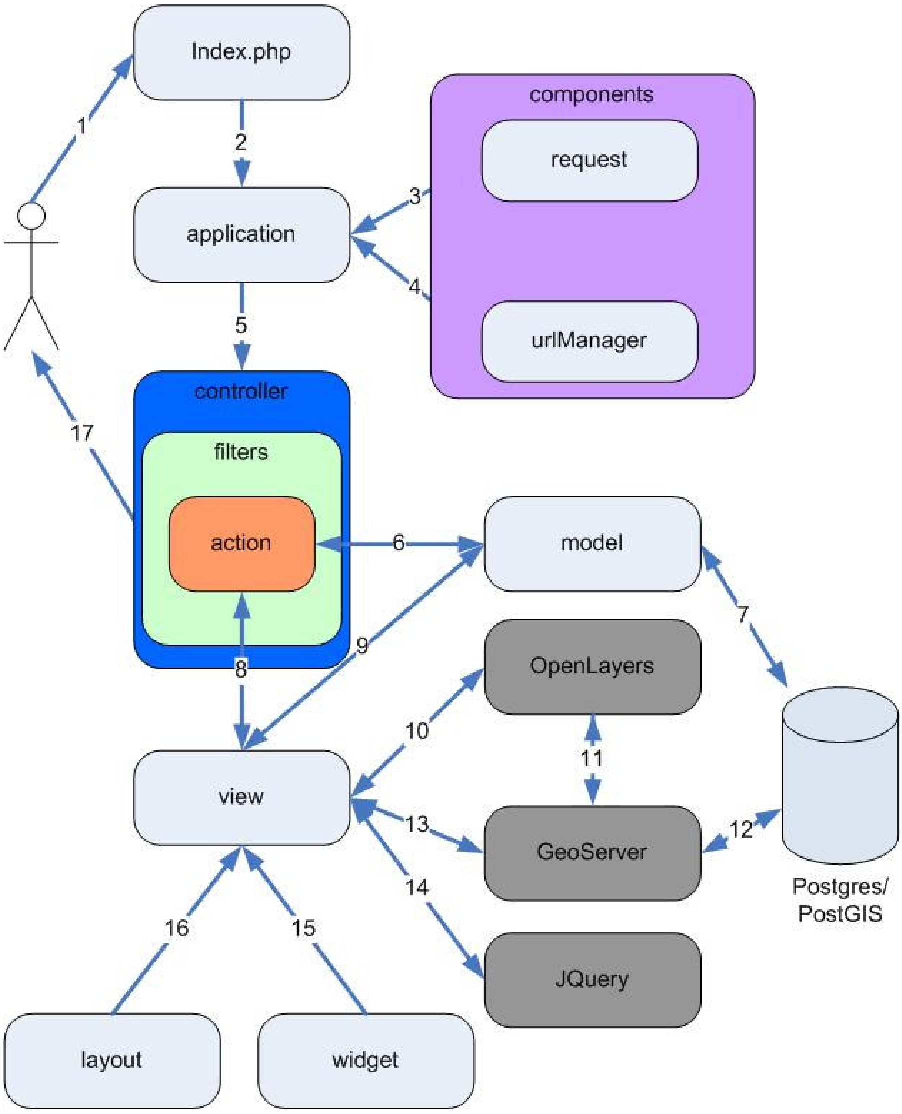


Figure 4: Workflow of the application

Relating to the workflow in Figure 4:

1. the user requests a page
2. an ‘application request’ is created
3. application receives information on the request from ‘request’
4. application determines the controller to be invoked thanks to a ‘urlmanager’

5. The controller determines the action to be used (after having invoked the appropriate filters)
6. action reads data from model
7. model recovers data from the DB
8. action renders the appropriate view
9. view reads the model data
10. view invokes OpenLayers functions
11. interfaces with GeoServer
12. GeoServer recovers data from the DB
13. view directly invokes GeoServer
14. view invokes Jquery functions
15. view executes the widget
16. view renders the result in a layout
17. the result is shown to the user

The Database

WIZ uses a database with a spatial extension to store data; the chosen DBMS is Postgres with the extension PostGIS. The database contains both geographical and non-geographical tables. Figure 5 displays the complete pattern of the database and the relations among the different tables.

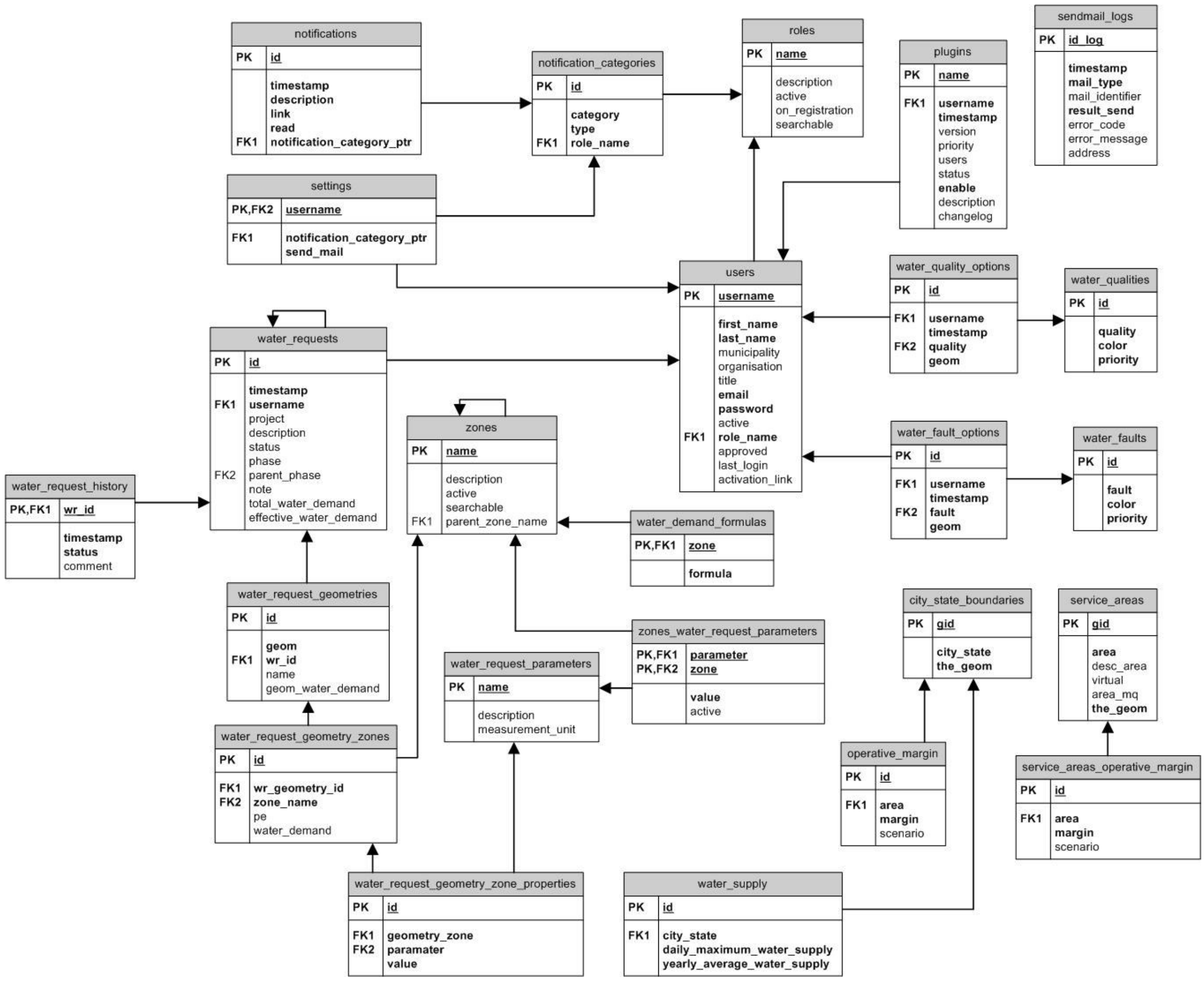


Figure 5: Database

notifications

Stores all notifications triggered by the system. Each notification is defined by:

- a unique identifier
- a timestamp
- a description
- a link that leads back to the entity that triggered the notification
- a flag that indicates whether a notification has already been read
- a category

notification_categories

Stores the possible categories for a notification. Each category is defined by:

- a unique identifier
- the entity that triggers the notification
- the type of operation that triggers the notification
- the role receiving the notification

settings

Stores the settings on the emails to be sent for the different users. Each setting is defined by:

- the user's username the setting refers to
- a notification category
- a flag that indicates whether to send an email or nor given the notification category

users

This is the table of the system users. Each user is defined by:

- a unique username
- a first name
- a last name
- a municipality
- an organisation
- a title
- an email address
- a password
- a flag that indicates whether the account is active
- a role
- a flag that indicates whether the account has been approved
- a timestamp relating to the last login
- a link to quickly activate an account

roles

Stores available users' roles. Each role is defined by:

- a unique name
- a description
- a flag that indicates whether a role is active and hence can be assigned to a user
- a flag that indicates whether the role can be chosen autonomously by a user during registration
- a flag that indicates whether it is possible to search users with that specific role

water_qualities

Stores the quality levels that a user can express. Each level is defined by:

- a unique id
- a quality description
- a colour for the visualisation
- a priority for the visualisation order

water_quality_options
Stores the quality opinion sent by users. Each opinion is defined by:

- a unique id
- the user's user name who expressed the opinion
- a timestamp
- the expressed quality
- a (geographical) reference point

water_faults
Stores the faults levels that a user can express. Each level is defined by:

- a unique id
- a description of the faults
- a colour for the visualisation
- a priority for the visualisation order

water_fault_options
Stores the faults send by the users. Each fault is defined by:

- a unique id
- the user's user name who indicated the fault
- a timestamp

- the type of fault
- a (geographical) reference point

water_requests

Stores the water resource requests sent by the users. Each request is defined by:

- a unique id
- a timestamp indicating the date of creation of the request
- the project's name
- a description
- a state
- a phase
- a link to a parent request (if it exists)
- the notes
- the total water demand
- the effective water demand

water_request_history

It keeps track of the state transactions of a water resource request. Each transaction is defined by:

- the id of the request
- a timestamp in which the transaction occurs
- the final state
- a comment

water_request_geometries

Stores the geometries that creates a water resource request. Each geometry is defined by:

- a unique id
- a group of (georeferenced) points that define it
- the id of the request it belongs to
- a name
- the associated water demand

water_request_geometry_zones
Stores the intended land uses relating to a geometry. Each intended land uses is defined by:

- a unique id
- the id of the geometry it refers to
- the intended land use
- population equivalent
- the associated water demand

water_request_geometry_zone_properties
Stores the parameter-value pair that defines a intended land use. A couple is defined by:

- a unique id
- the referential intended land use
- a parameter
- a value associated to the parameter

water_request_parameters
Contains the directory of the available parameters that can be associated to a intended land use. Each parameter is defined by:

- a name

- a description
- a unit of measurement

zones

Contains the directory of all possible intended land uses. Each intended land use is defined by:

- a name
- a description
- a flag that indicates whether the intended land use is active, and can hence be selected
- a flag that indicates whether the intended land use can be searched
- a parent intended land use (if it exists)

zones_water_request_parameters

Indicates for each intended land use which are the parameters that have to be requested. Each parameter is defined by:

- a nome
- a intended land use
- a conversion index
- a flag that indicates whether the parameter is active or not

water_demand_formulas

Stores the necessary formulas for reckoning the water demand. Each formula is defined by:

- the intended land use the formula is applied to
- the formula

water_supply

Stores the water resource consumptions relating to a city-state. Each consumption is defined by:

- a unique id
- the city-state it refers to
- the maximum daily consumption (in l/s)
- the average annual consumption (in l/s)

city_state_boundaries

Contains the list with the city-states. Each city-state is defined by:

- a unique id
- a name
- a group of georeferenced points that represent the borders

operative_margin

Indicates, for each city-state, which is the operative margin of the resource up to today and for the future. Each margin is defined by:

- a unique id
- the city-state it refers to
- the margin (in l/s)
- a description of the scenario for the margin (empty if the margin refers to the current situation)

service_area

Contains the list of the service areas. Each service area is define by:

- a unique id
- a name
- a description

- a flag indicating whether the service area is real or virtual
- the area in square meters
- a group of georeferenced points that represent the borders

service_area_operative_margin

Indicates for each service area which is the operative margin of the resource today and in the future. Each margin is defined by:

- a unique id
- the service area it refers to
- the margin (in l/s)
- a description of the scenario for the margin (empty if the margin refers to the current situation)

plugins

Stores the plugins installed on the system. Each plugin is defined by:

- a unique id
- the name of the user who installed it
- the timestamp of the installation
- the version
- a priority that establishes how it is displayed on the menu
- the users that can use it
- the state of the installation
- a flag that indicates if the plugin is active
- a description
- a changelog

sendmail_logs

Stores the log relating to emails that are sent. The log is defined by:

- a unique id
- a timestamp of what has been sent
- the category of the email sent (notification, warning, etc.)
- a reference to the entity that triggered the email sending
- the result of the sending operation (success/error)
- the possible error code
- the possible message associated to the error
- the recipient of the email (a user or all users belonging to a specific role)

Access control

Access control is quite sophisticated and is based on the *authorization* concept. The authorisation, that is the permission of a user to carry out a certain operation, can be:

- a role
- a task
- an operation

A role is composed of different tasks, each task is composed of different operations and an operation is a permission that is atomic.

These entities are hence connected among them through a parent-child relationship; moreover, it is possible to connect roles, tasks and operation among them again through a parent-child relationship to grant major flexibility. If an entity A is a parent of another entity B, then A inherits all permissions represented by B.

What you get at the end is a hierarchical partially ordered graph rather than a tree, where you can find the roles in the upper level and the operation in the lower one, while in the middle you can find the tasks.

Eventually, in order to have more flexibility, you can associate a rule to each entity (role, task and operation). The rule is a piece of code, which is valued each time the system needs to establish whether a user has a specific permission; only if the execution brings back a positive outcome the user will have the permission represented by the entity.

Figure 6 shows the graph with the accesses of the system. For ease only the main connections have been reported. The blue boxes represent the roles; the yellow ones the tasks, the orange ones the operations.

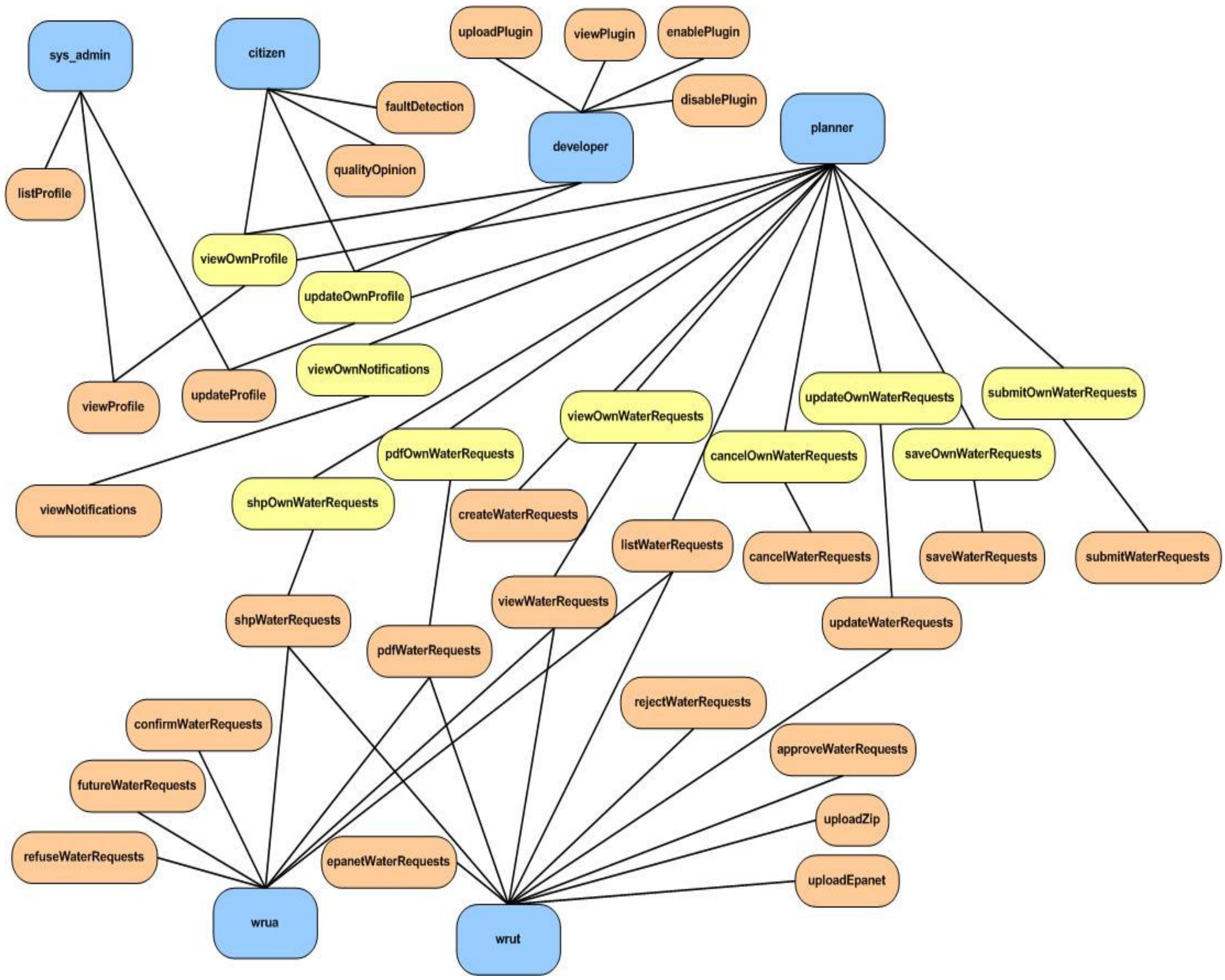


Figure 6: Graph with accesses control

The notifications

The system automatically generates notifications when certain events occur. Each notification is defined by a category that indicates:

- the entity involved in the notification (users, water resource requests,…)
- the event that generates the notification (creation, up-date,…)

Notifications can be addressed to a single user or to all users belonging to the same role.

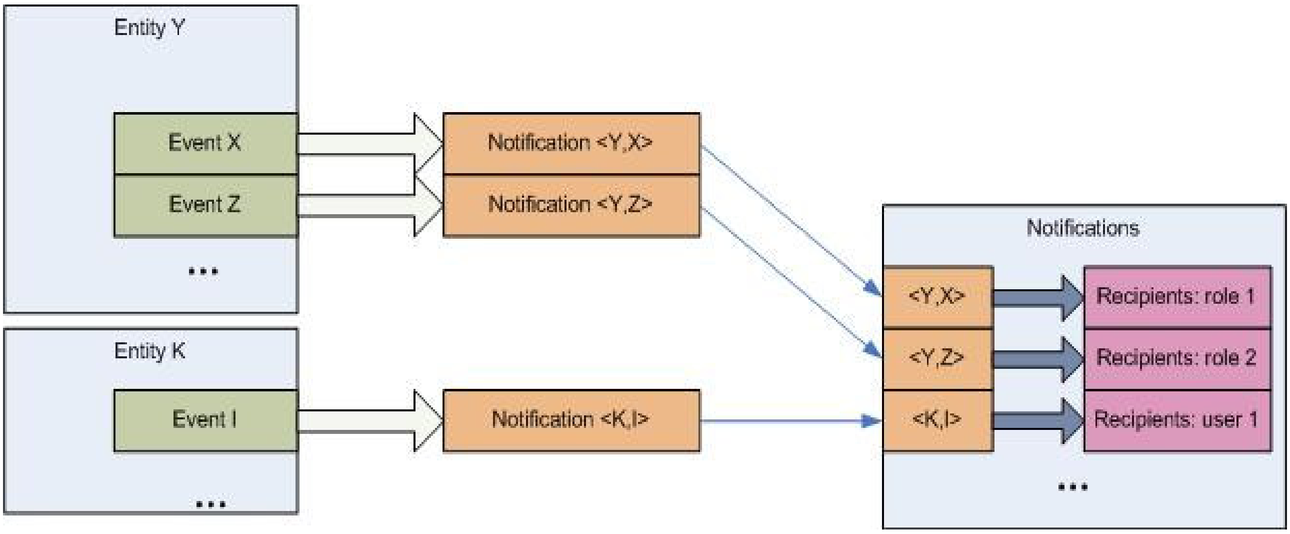


Figure 7: Structure of the notifications

Anyhow, in this second case, the notification is only one, even though it is visible to more users (that is all those having the same role). Hence, if a user marks the notification as 'read', also the others will see it as 'read'. In this case the flag, more than 'read', has to be considered 'taken in charge'.

Each user sees the notifications that he receives by accessing the corresponding section in the portal. The system forwards the content of the notification also by email to the user or the users addressed by the notification itself. However, each user can disable that emails are sent for a particular entity or for a specific operation by accessing the section relating to the settings.

Plugins

Plugins allow to enrich the functionalities of the system. They are uploaded by the IT developer user under an archive; then the IT developer user can manage the plugins through an appropriate user interface.

A plugin is simply an entity with its views, its model and its controller; it is to all intents and purposes a MVC component that easily integrates itself with the system. This structure considerably simplifies the installation procedure of new plugins; it is in fact sufficient to add to the existing views, as well as to the models and the controllers, those relating to the plugins and if necessary run a sql script (if provided together with the plugin). For a better overall plugin management it also requested to upload information on the plugin version and a description on its functionalities.

Evidently, the system expects to receive this information in a structured way; moreover the controller, the views and the models provided have necessarily to comply with simple rules; for more information on the plugins and their creation, please refer to the document 'Plugins and extensions'.

The users

The system offers the majority of functionalities only after logging in. For this reason, a user who wants to use the system, needs to registers beforehand by inserting the following data:

- First Name (mandatory)
- Last name (mandatory)
- Municipality the user lives in(mandatory)
- Organisation
- Title
- Email address (mandatory)
- Username (mandatory)
- Password (mandatory)
- Role(mandatory)

The password is stored by doing a md5 hash.

The role field allows a user to choose a role inside the system; the list of roles available during the registration phase is recovered from the *role* table thanks to the flag *on_registration*.

All registrations will have an *approved* flag set as false, to indicate that the account needs to be pre-emptively activated by the *system manager*. Therefore, the role the user autonomously chooses during registration has to be validated subsequently by the *system manager*.

The *system manager* can approve user accounts through an appropriate interface or by clicking on a link that he receives automatically per email after each registration. The system generates in fact a unique link upon a user registration that contains a string such as *username_<hash_di_caratteri_casuali>* and sends it by email to the *system manager*. Only the system manager can know this link and hence activate the user account concerned, by just clicking on the link itself.

Once the registration phase is completed, the user can authenticate himself by logging in.

Information requested while logging in are:

- Username (mandatory)
- Password (mandatory)

The login procedure will have a positive outcome whereby a user can access the system provided that:

- A user having that username exists
- The password matches the one stored in the system
- The user account has not been blocked (*active* flag set as true)

Access occurs with the user role only if the account has been approved by the *system manager*; failing this access will occur with the *citizen* role.

Figure 8: Login shows the procedure to log in.

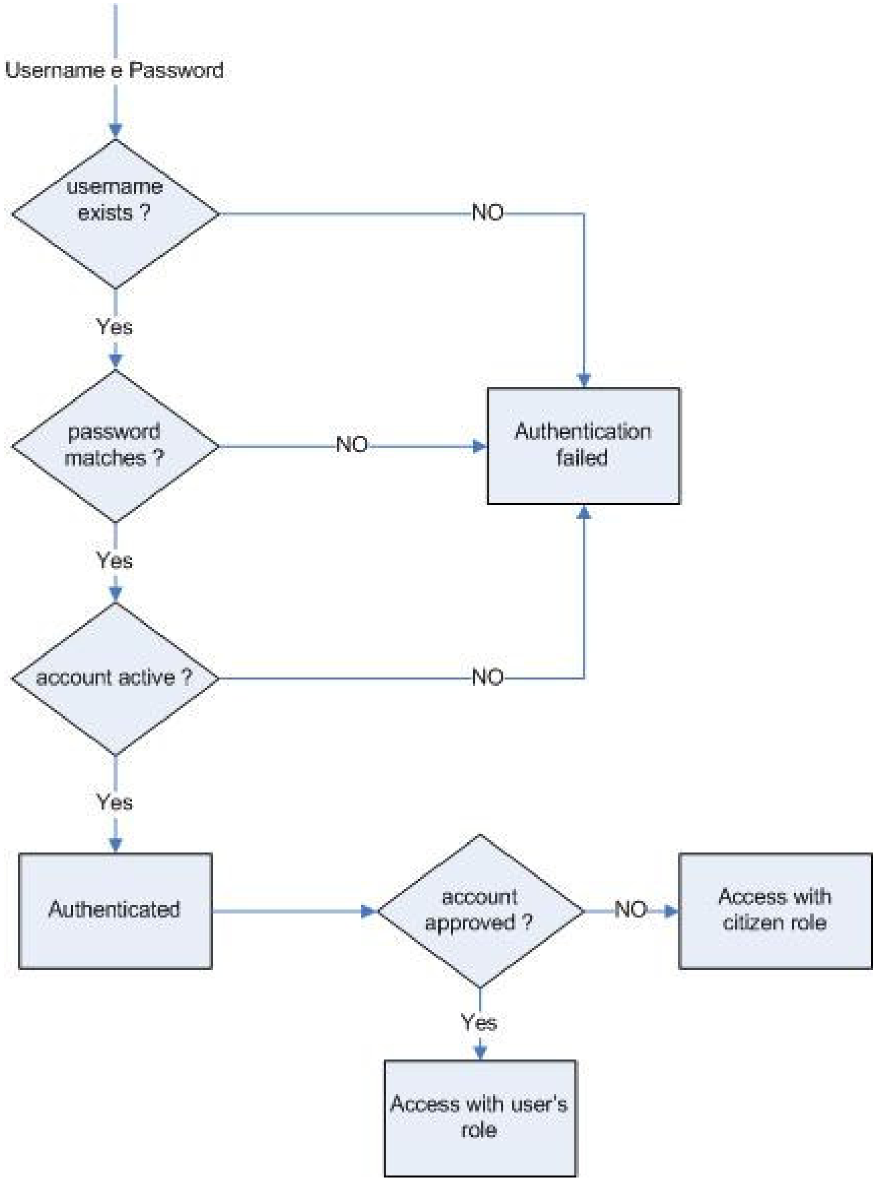


Figure 8: Login procedure

Water resource requests

Water resource requests represent the most complex entity of the system. It is thanks to this entity that a planner user can create and submit his own water resource requests to the utility. There are two types of water resource requests and each request evolves through phases: for more information please read the document 'Specifications'.

A water resource request is stored in the table *water_requests* and, as you can see from the database in Figure 5 it involved different entities. *water_request_geometries* contains all geometries that make up the request. *geom_water_demand* is among the most important characteristics of the geometry that represent the water demand associated to the geometry. The sum of the various *geom_water_demand* of the geometries that make up a request provides the field *total_water_demand*.

Each geometry is in turn composed of intended land uses. Intended land uses are stored in the table *water_request_geometry_zones* that indicates both the population equivalent and the water demand reckoned for a specific intended land use. The sum of the water demand of the different intended land uses inside the same geometry provides the field *geom_water_demand*.

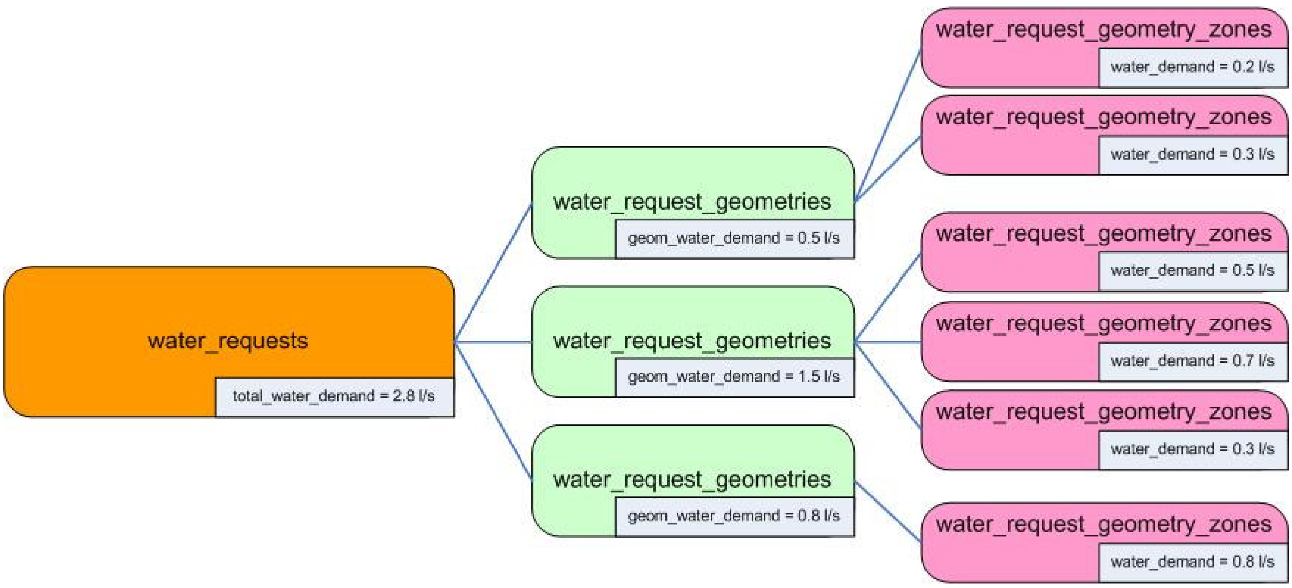


Figure 9: Relationship between water_requests, water_request_geometries and water_request_geometry_zones

The calculation of the population equivalent and subsequently the water demand is done automatically by the system and thanks to the user's support who has to reply to a few simple questions. Figure 10 shows the mechanism:

1. The user selects an intended land use among all those admissible contained in the table *zones*

2. On the basis of the intended land use, the system recovers one or more pairs parameter name - conversion index from the table *zones_water_request_parameters*

3. For at least one parameter coming from phase 2, the user has to insert a value that will be subsequently multiplied with the corresponding conversion index to obtain the population equivalent.

4. On the basis of the intended land use, the system recovers a formula from the table *water_demand_formulas* to be used for reckoning the water demand

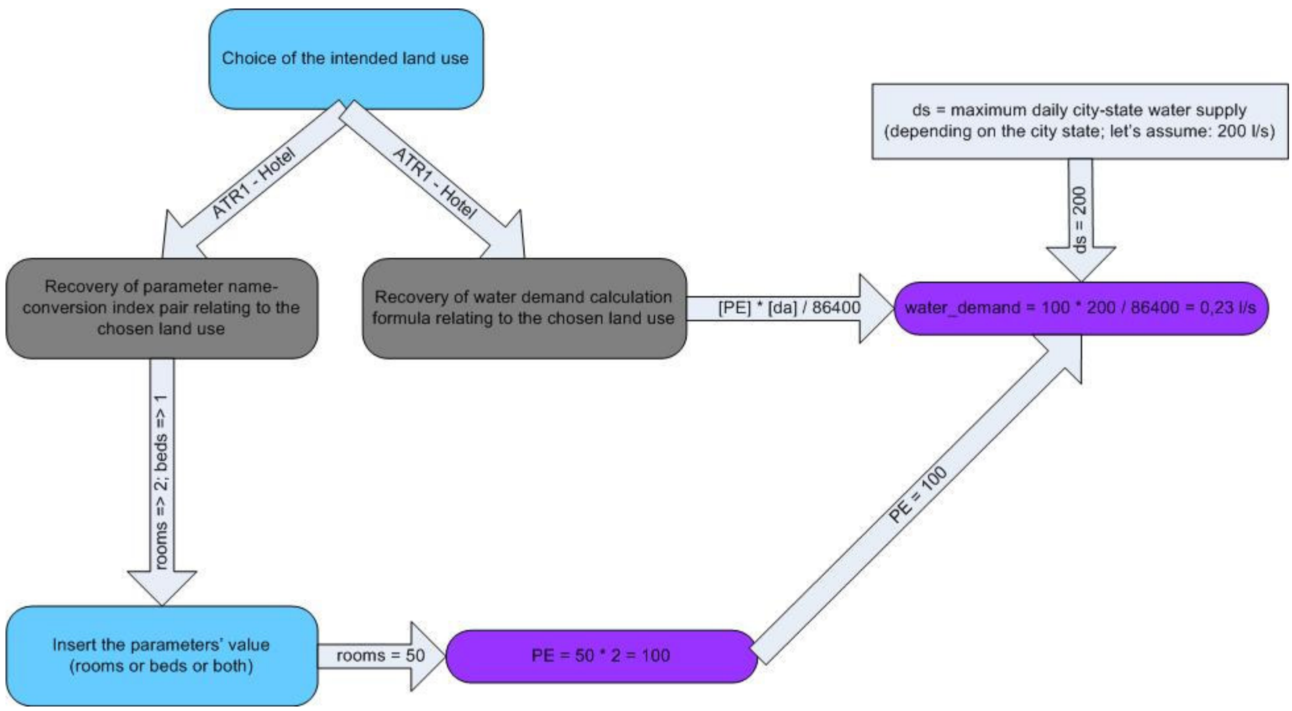


Figure 10: Procedure for calculating the population equivalent and the water demand

Formulas hence allow to calculate the water demand associated to a geometry while the conversion indexes are necessary to determine the population equivalent; both depend on the intended land use. Figure 11 shows the formulas used by the system.

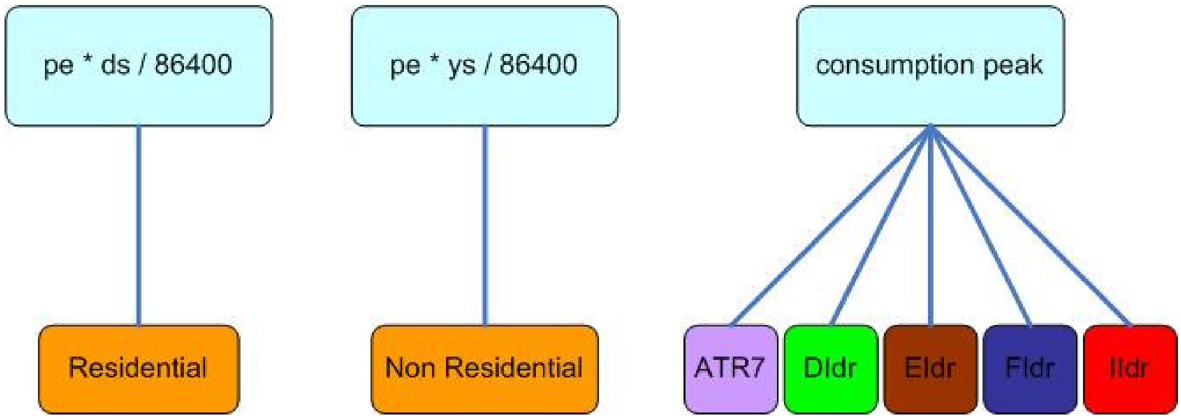


Figure 11: Formulas for calculating the water demand

The conversion indexes are also used by the system to ask focused questions to the user: a specific intended land use provided, the system knows which are the associated conversion indexes and will hence request the user to insert at least one of the parameters for which the conversion index exists (dropping the remaining ones). Figure 12 indicates the parameters recognised by the system, their association with the intended land uses and the corresponding conversion index. The same parameter can clearly be associated to more than one intended land uses (and for each association the value can change) and a intended land use needs to have at least one associated index.

Author: Ing. Salvo Di Mare
Phone: +39050931630
E-mail: s.dimare@cpr.it

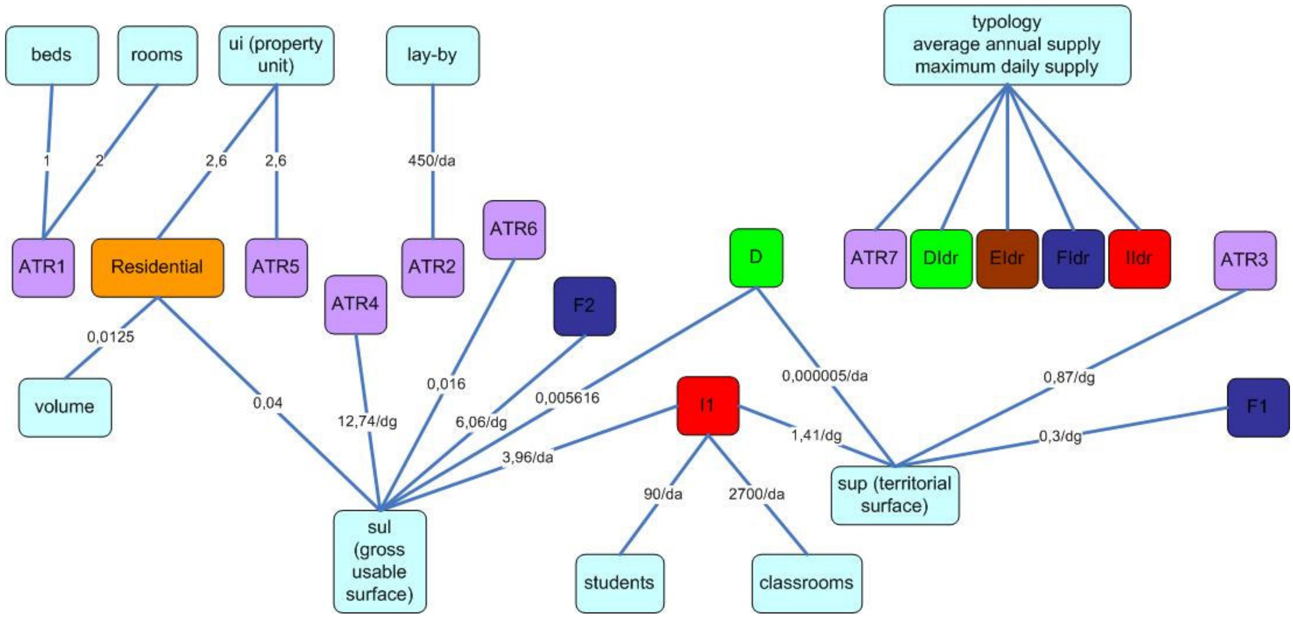


Figure 12: Parameters, conversion indexes and their relations with the intended land uses

With reference to Figure 12, and assuming that there is an intended use for a 'residential' area:

- The system will request the user to specify at least one parameter among:
    - ui (property unit)
    - sul (gross usable surface)
    - volume

  because only these parameters are associated with a conversion index relating to the 'residential' area
- assuming that the user inserts a sul value of 1000m^2 , the system will calculate 40 population equivalent (1000 * 0.04)

Author: Ing. Salvo Di Mare
Phone: +39050931630
E-mail: s.dimare@cpr.it

To streamline the entire process, to simplify the management of the system and minimize the requested data, the intended land uses, stored in the table *zones*, have been organised in a hierarchical way; Figure 13 shows the scheme[2].

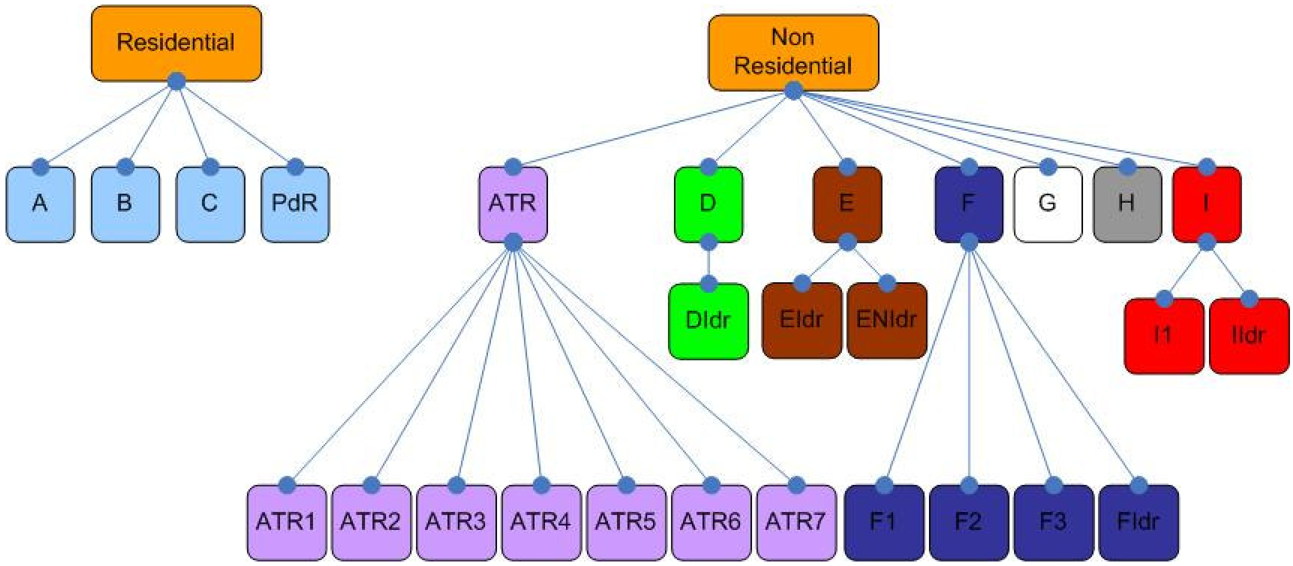


Figure 13: Intended land uses hierarchically organised

This hierarchical structure is largely exploited in steps 2 and 4. In fact, when the system searches the pair parameter name - conversion index (step 2 in the list of page 30) and a correspondence is not found with the chosen parent intended land use, it tries with the parent intended land use and so on retracing the tree until the top. Figure 14 illustrates the procedure.

[2] A (Historicized Areas); B (Areas for residential completion); C (Areas of residential expansion); PdR (Residential Recovery Plan); ATR (Areas for tourism equipped with facilities); ATR1 (Hotels); ATR2 (Camping); ATR3 (Rural farm tourism); ATR4 (Restaurants); ATR5 (Libraries); ATR6 (Religious institutes); ATR7 (other area with a water demand); D (productive / commercial completion and/or expansion zones); Dldr (productive users with a high water demand); E (Agricultural areas); Eldr (Areas with a water demand); ENldr (Areas WITHOUT a water demand); F (Infrastructures and facilities with a public interest); F1 (Sports centres); F2 (Sports hall); F3 (Swimming pool); Fldr (other area with a water demand); G (Green areas); H (Areas for the protection of the environment, the landscape, the villages, the nature); I (Areas for education); I1 (Schools); Ildr (other area with a water demand)

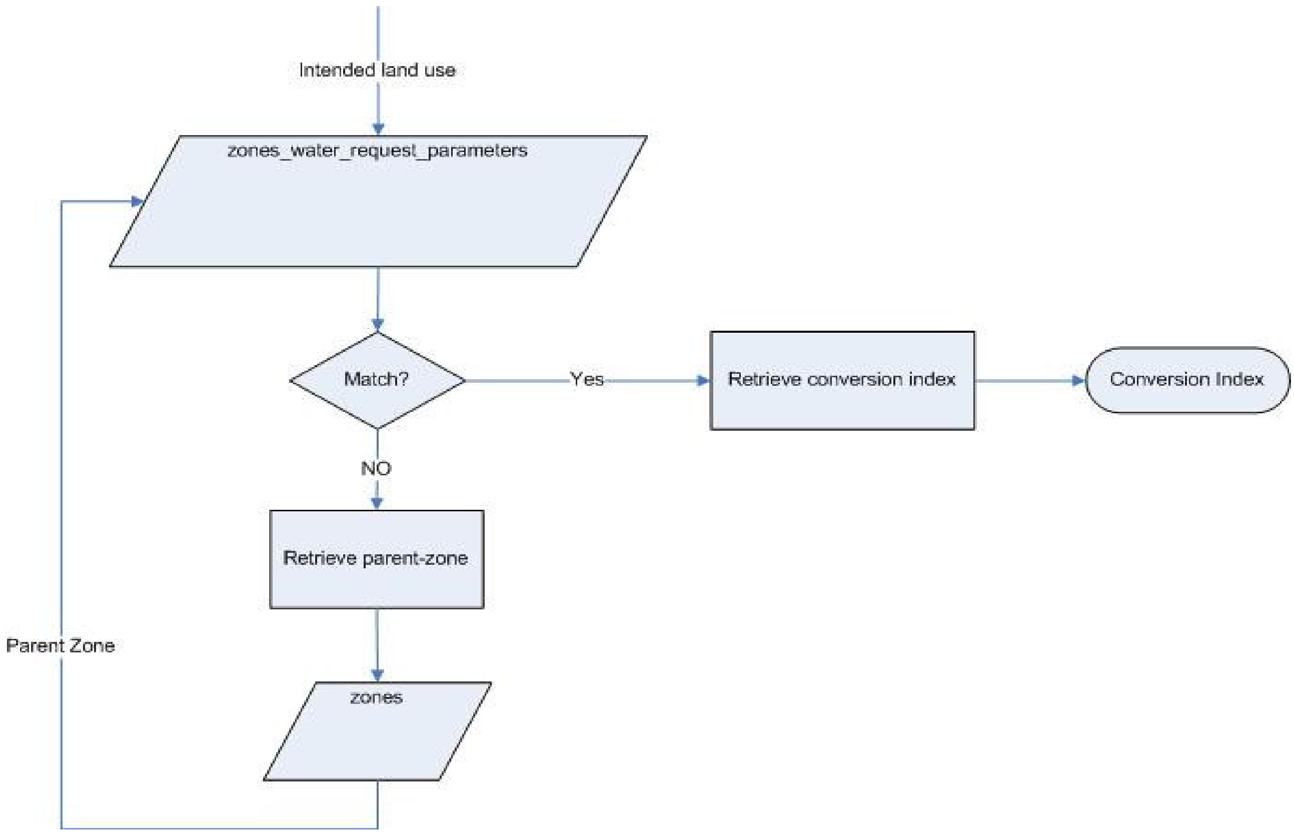


Figure 14: Procedure for the recovery of the conversion index for the population equivalent calculation

Assuming you have chosen the land use 'A - historicized Areas' the system:

1. Searches the table *zones_water_request_parameters* to see whether there is at least one parameter that is associated to the 'A' zone
2. If the system does not find any, it recovers the parent land use of 'A', that is the 'Residential' zone
3. Go back to the table *zones_water_request_parameters* and check whether there is at least one parameter corresponding to the 'Residential' zone
4. If the system does not find any, go back to step 2; otherwise the procedure ends providing one or more pairs parameter name - conversion index

Similar arguments are true for the formulas (step 4 in the list of page 31): first a formula relating to the chosen land use is searched and, if it is not found, a parent formula is searched and so on. Figure 15 shows the latter procedure.

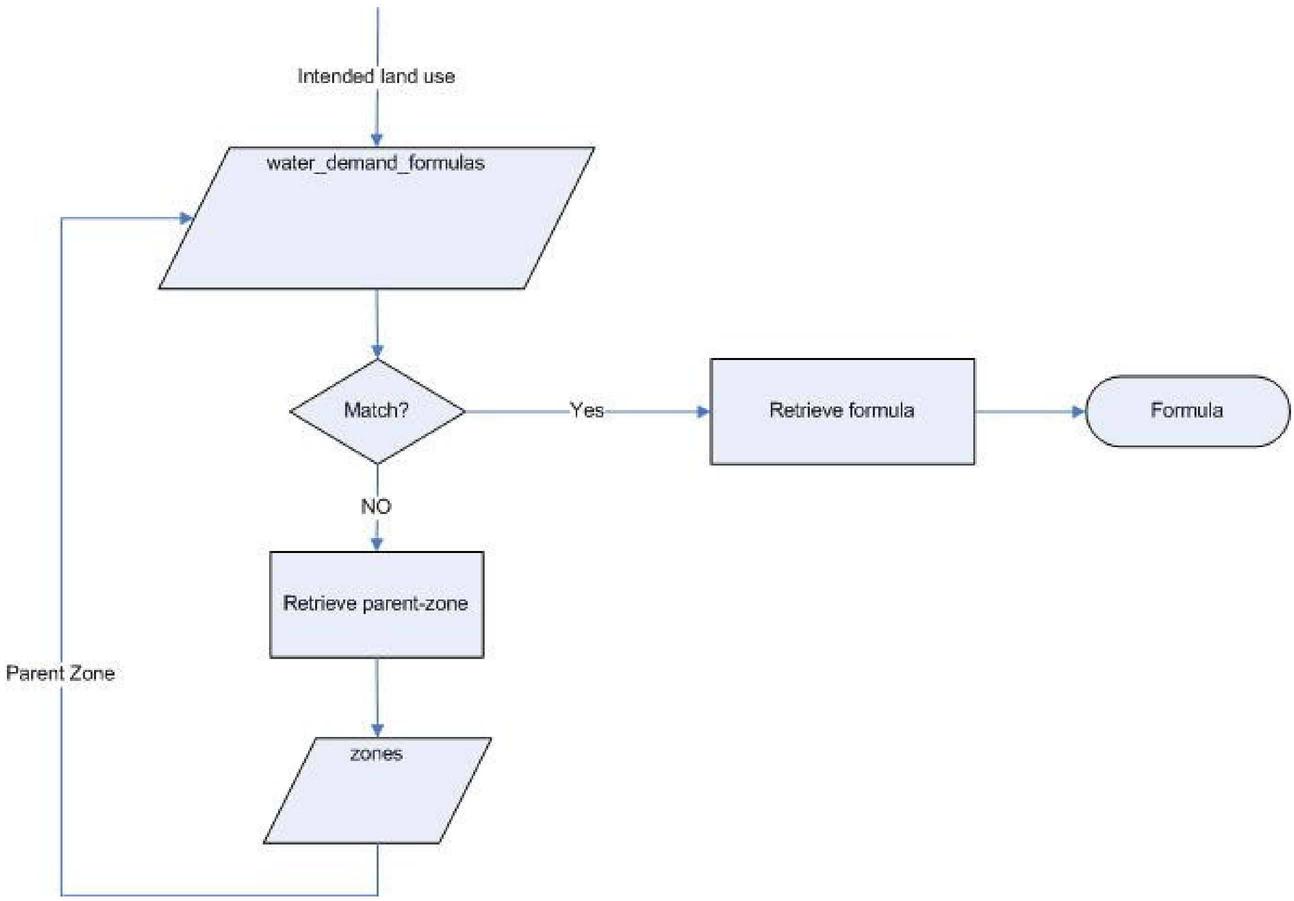


Figure 15: Procedure for the recovery of the formula for the water demand calculation

Assuming that you have chosen as land use 'ATR1 – Hotels' zone the system:

1. Searches the table *zones_water_request_parameters* to see whether there is at least one formula that is associated to the 'ATR1 – Hotels' zone
2. If the system does not find any, it recovers the parent land use of 'ATR1 – Hotels', that is the 'ATR – Areas for tourism equipped with facilities' zone
3. Go back to the table *zones_water_request_parameters* and check whether there is at least one formula corresponding to the ATR – Areas for tourism equipped with facilities' zone

4. If the system does not find any, go back to step 2; otherwise the procedure ends providing the requested formula

The conversion indexes and formulas are not simple numbers but in general are strings that contain numbers and mathematical operators as well as variables. The system performs hence an intelligent parsing assessing these strings and returning a number. As far as the conversion indexes are concerned, their structure is rather simple as these are mostly numbers; in some cases mathematical operators and/or only two variables are foreseen:

- da: yearly average municipal supply
- dg: maximum daily municipal supply

These two variables clearly depend on the city-state interested in the request and are recovered from the table *water_supply*. Figure 16 shows the procedure.

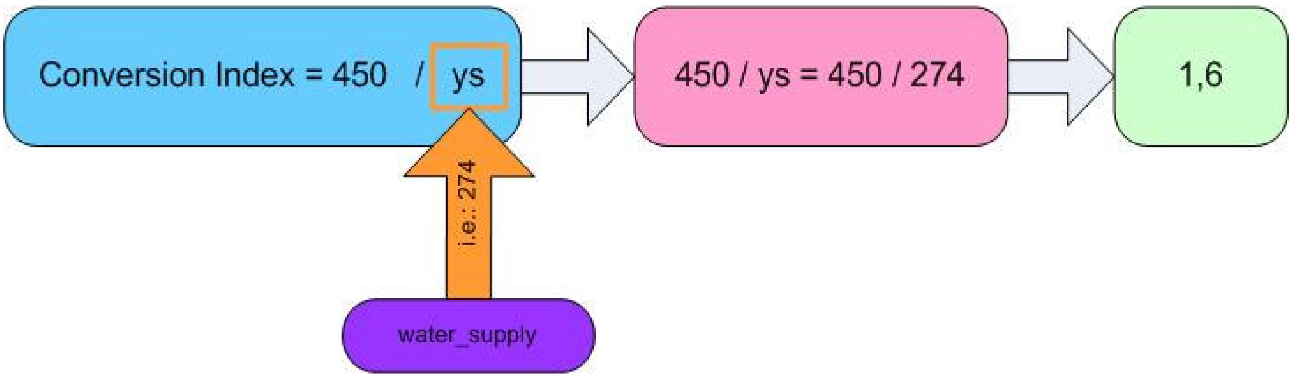


Figure 16: Assessment of the conversion indexes

As far as the formulas are concerned, the structure becomes more complicated since, in addition to what has been said for the conversion indexes, one or more parameters can be used, which will be replaced, when it comes to the formula assessment, with the value specified by the user during insertion. To clarify this concept, Figure 17 displays the functioning.

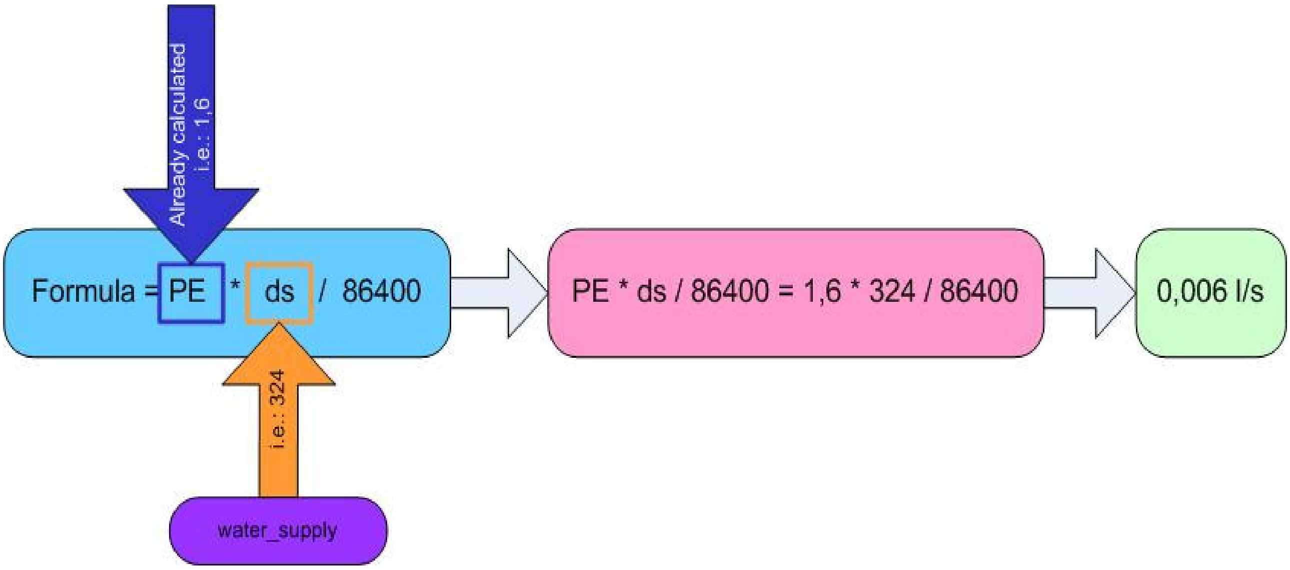


Figure 17: Assessment of the formulas for the water demand calculation

By detailing the scheme in Figure 10, Figure 17 shows and summarises the entire process and the tables involved in the water demand calculation.

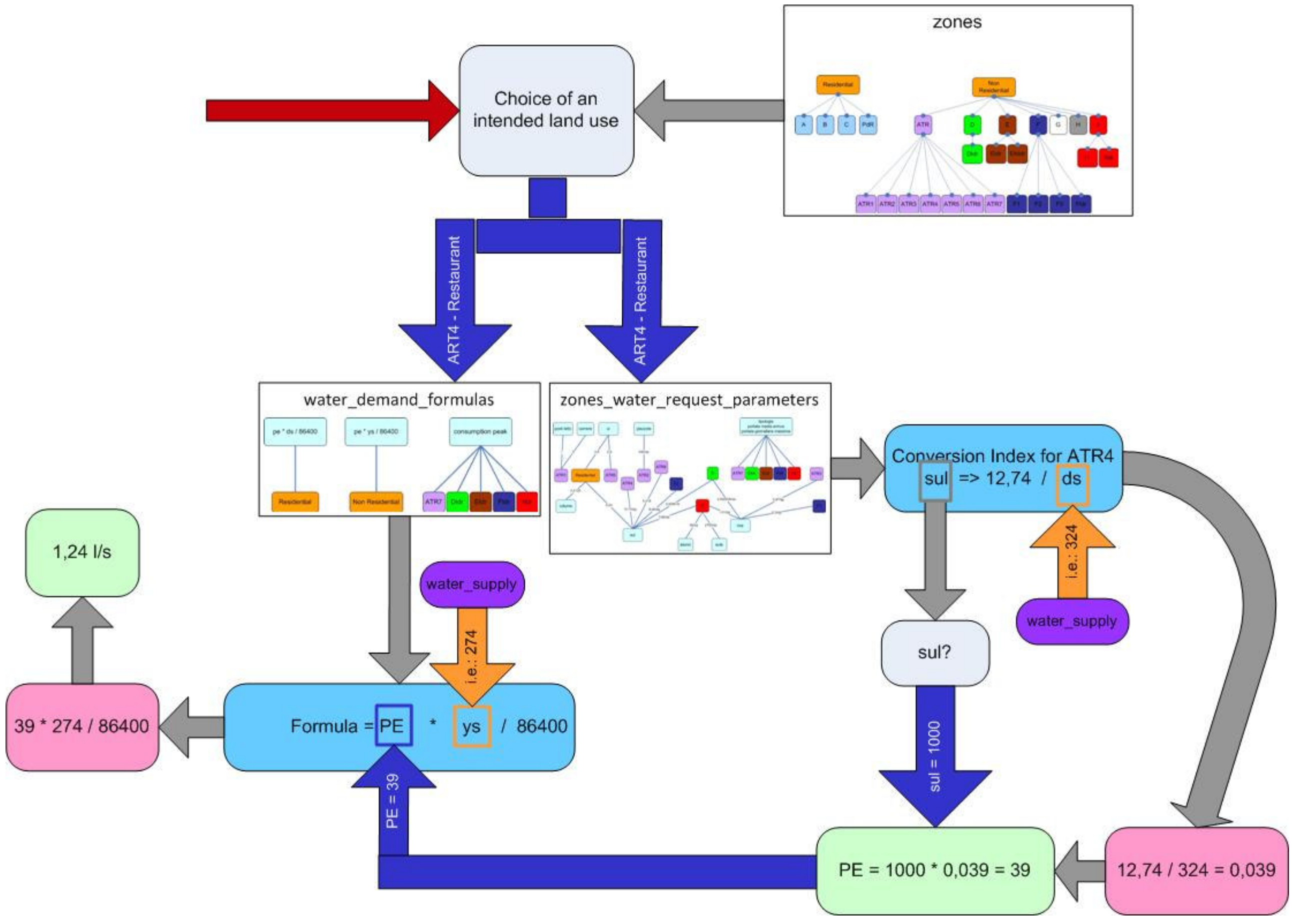


Figure 18: Detail of the water demand calculation

Once the water demand has been obtained, the system can also instantly do a control comparing the water resource request with the operative margin of the city-state stored in the table *operative_margin*. Moreover, in the same table are also stored the future forecast scenarios (if existing) on the resource trend.

To less detail the response, each city-state has been subdivided into sub-areas named <u>service areas</u>. Each service area, apart from a name and geographical borders, is also defined by an operative margin, stored in the table *service_area_operative_margin* that contains also the future forecast scenarios (if existing).

Each time a user draws a geometry inside a request, the system calculates which is the city-state involved and in which service area it lies; the two information are used, besides for the calculation as we have seen above, also to show to the user the current and future margin of the resource according to the following steps:

- the system searches whether there is a margin associated to the interested service area
- if it exists it is displayed; otherwise the margin of the city-state involved is used

According to the procedure described so far, the decision-making process that leads the system automatically to determine the water demand associated to a request is rather complicated and involves several tables of the database thanks to which the system cross-checks geographical and non- geographical data. The main tables involved in this process are:

- municipal borders (confini_comunali)
- service areas (service_area)
- *operative*_margin
- service_area_operative_margin
- water_supply
- zones
- zones_water_request_parameters

The flow diagram in Figure 19 sketches the described procedure.

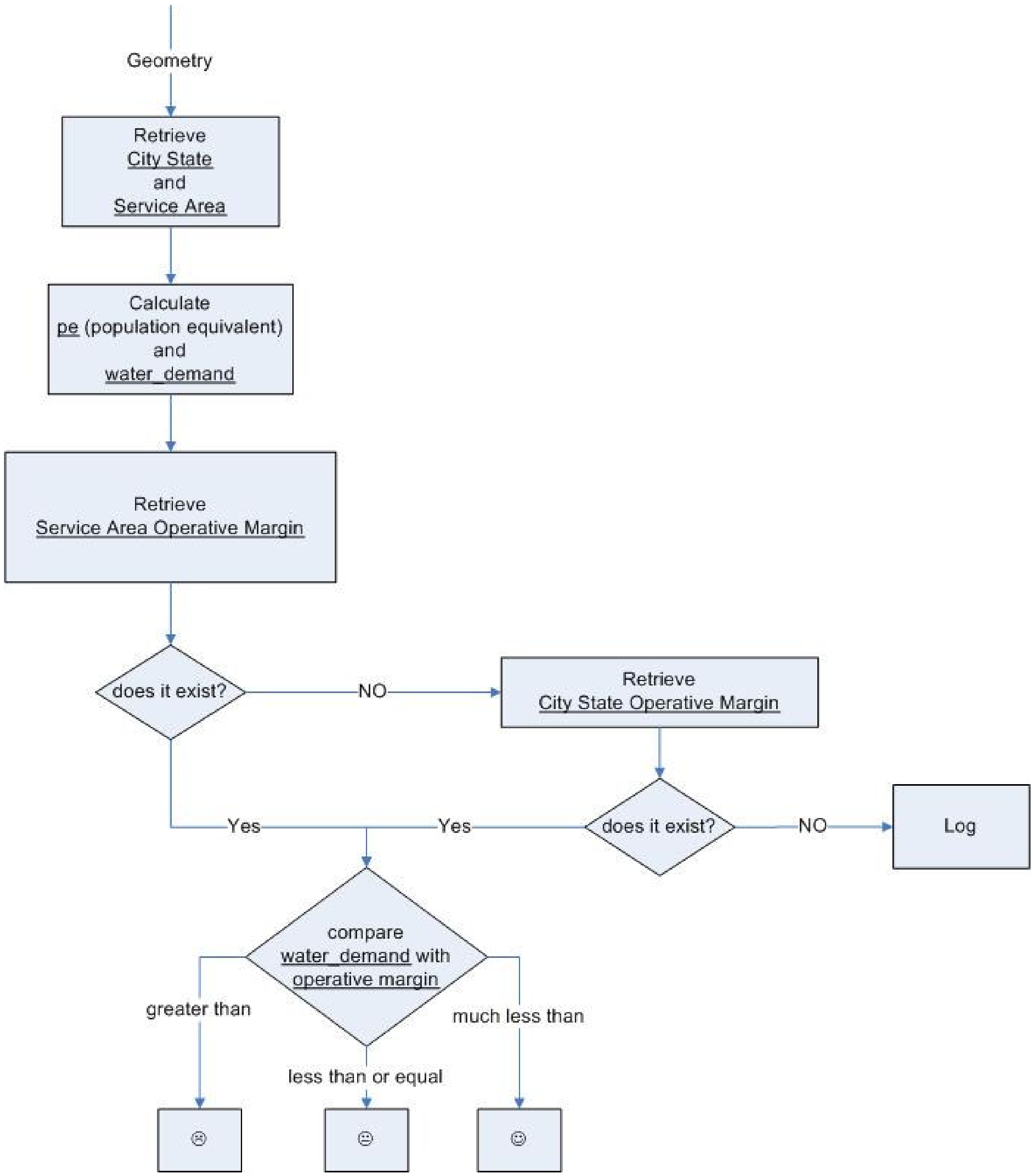


Figure 19: Comparison between water demand and operative margin

All values present in the tables involved for reckoning the population equivalent and the water demand are already pre-emptively set but can obviously be modified by the *water resource utility*. Apart from modifying data singularly, the *water resource utility* can upload this information directly from a *.csv* file.

EPANET Interface

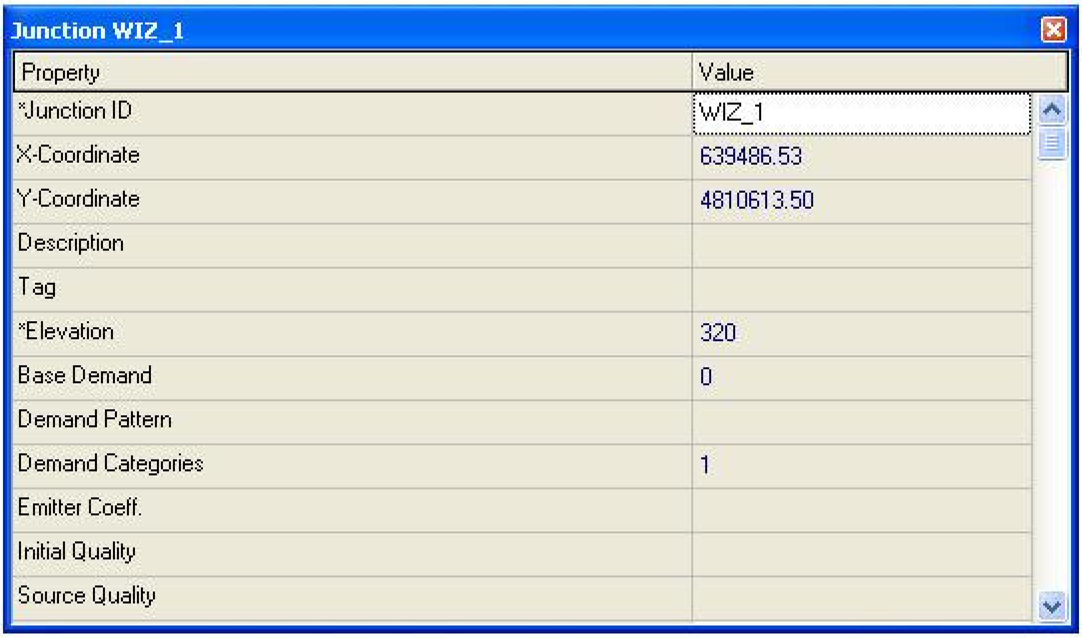
Provided there is a water resource request, the utility needs to be able to make simulations on the distribution network to decide whether the request can be granted or not. These simulations are carried out by an external simulation software, EPANET. The system has to permit hence an interface with EPANET.

From the point of view of the utility, the necessary data to be able to execute the simulations is the new file of the distribution network, updated with the nodes (in jargon EPANET, the *junctions*) relating to the water resource requested examined. A node is nothing more than a georeferenced point that corresponds to the centroid of the geometry that makes up the water resource request. Since a request can contain more geometries, one or more nodes relating to the same request will generally be generated.

Inside EPANET, a node is defined by a serious of information, such as:

- junction id
- x-coordinate
- y-coordinate
- description
- tag
- elevation
- base demand
- demand pattern
- demand categories

- emitter coeff.
- initial quality
- source quality


Junction WIZ_1

Property	Value
*Junction ID	WIZ_1
X-Coordinate	639486.53
Y-Coordinate	4810613.50
Description	
Tag	
*Elevation	320
Base Demand	0
Demand Pattern	
Demand Categories	1
Emitter Coeff.	
Initial Quality	
Source Quality	

Figure 20: Fields of a Junction, according to the EPANET specifications

Some of these parameters are generated automatically by the system while others need to be inserted by the user (and these are optional). In particular, for each node, the system generates:

- unique identifier, in the form 'WIZ_<request id>_<geometry id>' (field *junction_id*)
- intended land use of the geometry (field *description*)
- coordinates x,y of the centroids of the geometry (field *x-coordinate* and *y-coordinate*)
- elevation of the centroids of the geometry (field *elevation*)
- water demand associated to the geometry (field *base demand*)

The nodes generated by the system have to be subsequently inserted inside the EPANET file of the distribution network. To carry out this operation, the user needs to upload the EPANET files of the distribution network in such a way that the system can update it.

According to the EPANET specifications, the model file of the network is a text file, subdivided into sections. The system reads the file, locates the sections and inserts the necessary information only for the sections concerned with the *junctions*. Figure 21 illustrates an EPANET file example, which, for reasons of simplicity, reports only one *junction* with id 'WIZ_13_10'.

```
[TITLE]
Something

[JUNCTIONS]
;ID                 Elev            Demand          Pattern
 WIZ_13_10          94.0            2.7263888888889                     ;PdR - Piano di Recupero Residenziale

[TAGS]
 NODE   WIZ_13_10           A tag

[DEMANDS]
;Junction           Demand          Pattern             Category
  WIZ_13_10         0.04597000000   A pattern       ;

[EMITTERS]
;Junction           Coefficient

[QUALITY]
;Node               InitQual
 WIZ_13_10          20

[SOURCES]
;Node               Type            Quality         Pattern

[COORDINATES]
;Node               X-Coord             Y-Coord
 WIZ_13_10          641322.08           4817571.07

```

Figure 21: Structure of an EPANET model file

Since the system receives a file containing the coordinates of the nodes already present in the EPANET model, the user needs to indicate as well the system of reference used for those coordinates, in such a way that, if necessary, the system Figure 22 shows the generation process of a new EPANET file in its entirety.

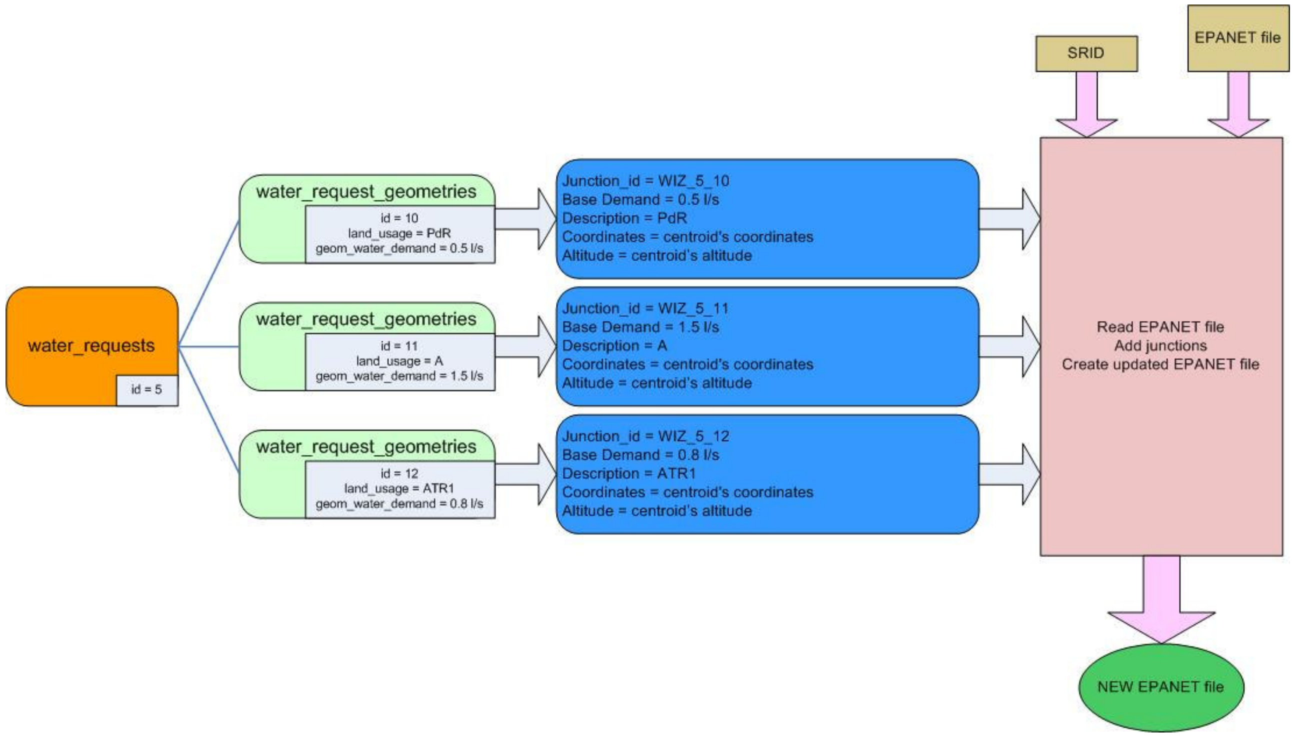


Figure 22: Generation of a new EPANET file

Participated quality

The system grants access to some functionalities, typically to the citizen, where a citizen can express his judgment on the (perceived) water quality. The judgment is composed of a textual part, that is the expressed quality, and a geographical part, that is a spot on the map the judgment refers to.

Similarly to quality, also faults can be reported through specifically foreseen functionalities.

The system manager can decide which are the possible quality or fault levels. During the installation phase the system foresees 5 quality levels:

- excellent
- decent
- good

- low
- insufficient

and 2 fault levels:

- water loss
- generic fault

By choosing a quality or fault level and a spot on the map, the system stores the pair level-georeferenced spot and makes it available to other users.

The geographical information

The system stores geographical information on a Postgres database with extension PostGIS. The map server used is GeoServer.

The geographical information that are used can be subdivided into two macro-categories:

- background data
- overlay data

Background data relate to the representation of the territory; those foreseen are:

- OpenStreetMap: physical map of the territory
- Othophoto 10K: is uploaded thanks to the GEOscopio service offered by the Tuscan Region
- CTR 10K and 2K: are uploaded thanks to the GEOscopio service offered by the Tuscan Region

Overlay data, on the other hand, represent information that can been overlapped with the background data (which are instead mutually exclusive). Overlay data that can be activated by the user are:

- Municipal borders: reports, in a shape format, the administrative borders of the city-states managed by Acque SPA.
- Service Areas: report, in a shape format, the subdivision of the city-states managed by Acque SPA in sub-zones, called Service Areas
- Geometries: represent the geometries that make up a given water resource request
- Distribution network: contains georeferenced information on the water distribution network, with its technical characteristics and services
- Sources: shows the exact location of the sources, subdivided in:
    - waterways
    - lakes
    - wells
    - springs
- Facilities: shows the exact location of the facilities, subdivided in:
    - water purification systems
    - adductions
    - accumulations
    - pumping station
- Street map: street names, squares and place, georeferenced.

Functionalities made available by the OpenLayers library are used to query GeoServer and visualise the information.

Log & Feedback

Besides the notifications, described in chapter "The notifications", the system sends emails to the administrator when there is a malfunctioning. In general, the system succeeds in locating automatically the 'dysfunctional' situations, independently from the fact that these generate a visible error to the user or not; these dysfunctional situations are stored in a log file that reports besides the date and time, a detailed description on the detected malfunctioning. In those cases where the error is critical, i.e. the system is blocked as a consequence, the error description is also sent by email to the system administrator; in this case the administrator can know which are the critical situations and step in promptly.

There are cases where the system is not able to automatically take notice of a fault; and this is true especially for the user interfaces, 'susceptible' to the different browsers and the different versions. For this reason, a user-friendly but powerful feed-back mechanism has been developed to determine the type of fault. It is possible to capture the screenshot through a link located at the bottom right of the screen, which is always visible by any kind of user, adding if needed notes or hiding sensitive data; the system will send the image of the screenshot to the administrator, adding also information on the operative system of the user who generated the feedback as well as the type and browser version used.

Internationalisation

The web portal includes functionalities for the internationalisation. Available languages are English and Italian; it is possible to add any kind of language providing beforehand the files of the translation.

The language can not be chosen by the end-user but is set during the system installation phase. The administrator user can modify the language any time; this modification will affect all users who will see the web portal in the updated language.

The translation files reside in the folder *messages/<language_code>/*; each file establishes a correspondence between key-translation where the key is used inside the code to display the translation.

Hence, it is sufficient to replicate the files present inside the folder *messages/<language_code>/* to create a new translation and do the translation only for the part 'translation', leaving intact the part 'key'. These files will be stored inside the folder *messages/<new_language_code>/*; to enable the translation, the new language needs to be set among the configuration parameters. For more information on the parameters, please refer to the chapter

Parameters configuration

All configuration parameters of the web portal reside in the file *config/params.php*. These parameters allow to specify the behaviour of some system components and to easily modify it as necessary. The system foresees also a simple user interface through which the system administrator can modify the value of these parameters or if needed add others.

The following list enumerates all parameters used and briefly describes them:

- *language*: the web portal's language. Default value: 'it'
- *adminEmail*: email address of the administrator to whom to send emails on the general system functioning. Default value: 'acque@cpr.it'
- *debugEmail*: email address of the administrator to whom to send communications on errors that arise during the system usage. Default value: 'acque@cpr.it'
- *block_debugEmail*: indicates whether to block debug email sending. Default value: 'false'
- *block_email*: indicates whether to block email sending generated by the system. Default value: 'false'
- *geoserver*: composed parameter. Contains information relating to geoserver:
    - *version*: version of installed geoserver. Default value: '1.1.0'
    - *protocol*: communication protocol with geoserver. Default value: 'http://'
    - *ip*: ip address of geoserver. Default value: '127.0.0.1'

  - *port*: geoserver port. Default value: '8080'
  - *path*: path in which to send requests for geoserver. Default value: '/geoserver'
  - *wms*: path to access wms services of geoserver. Default value: '/wms'
  - *wfs*: path to access wfs services of geoserver. Default value: '/wfs'
  - *workspace*: workspace inside which data are stored. Default value: 'acque'
  - *layer_geom*: layer name that stores geometries. Default value: 'wr_geom'
  - *default_srs*: spatial reference system. Default value: 'EPSG:4326'
  - *citystate_layer_srid*: spatial reference system used for municipal borders. Default value: '3003'
  - *water_request_geometries_srid*: spatial reference system used for geometries of a water request. Default value: '4326'
  - *pdf_geoms*: layer name of geometries used for the pdf generation. Default value: 'pdf_geoms'
  - *layer_dem*: DEM layer name. Default value: 'DEM'
  - *service_areas_layer_srid*: spatial reference system used for used for service areas. Default value: '3003'
- *water_demand_unit*: unit of measurement of water demand. Default value: 'l/s'
- *decimals*: numbers of digits after the comma. Default value: '2'
- *pe_precision*: numbers of digits after the comma for the population equivalent. Default value: '0'
- *we_decimals*: numbers of digits after the comma for the water demand. Default value: '2'

- *pe_abbreviation*: acronym indicating the population equivalent. Default value: 'PE'
- *formula*: composed parameter. Contains information relating to the water demand calculation formulas:
  - *pe*: acronym used to indicate the population equivalent. Default value: 'ae'
  - *da*: acronym used to indicate annual average supply. Default value: 'da
  - *dg*: acronym used to indicate daily maximum supply. Default value: 'dg
- *dateTimeFormat*: indicates how to format the fields datetime to make the date and time readable to the user. Default value: 'j F Y, H:i'
- *dateTimeFormat*: indicates how to format the fields datetime in order to be correctly stored on the database. Default value: 'Y-m-d H:i:sO'
- *dateFormat*: indicates how to format the fields date to make the date readable to the user. Default value: 'j F Y'
- *EPANET*: composed parameter. Contains information relating to the interface for EPANET:
  - *section_marker_start*: character indicating the beginning of the section tag, inside a file generated by EPANET. Default value: '['
  - *section_marker_end*: character indicating the end of the section tag, inside a file generated by EPANET. Default value: ']'
  - *section_regular_expression*: regular expression used to identify the name of a section. Default value: '/^\\[[a-zA-Z0-9]{1,}\\}/'
  - *comment_marker*: character indicating the beginning of a comment, inside a file generated by EPANET. Default value: ';'
  - *section_junctions*: name of the section junctions, inside a file generated by EPANET. Default value: 'JUNCTIONS'

- *section_coordinates*: name of the section coordinates, inside a file generated by EPANET. Default value: 'COORDINATES'
- *section_tags*: name of the section tags, inside a file generated by EPANET. Default value: 'TAGS'
- *section_emitters*: name of the section emitters, inside a file generated by EPANET. Default value: 'EMITTERS'
- *section_quality*: name of the section quality, inside a file generated by EPANET. Default value: 'QUALITY'
- *section_sources*: name of the section sources, inside a file generated by EPANET. Default value: 'SOURCES'
- *tag_node_junctions*: value of the field node inside tag, inside a file generated by EPANET. Default value: 'NODE'
- *source_type_junctions*: value of the field type inside source, inside a file generated by EPANET. Default value: 'CONCEN'
- *upload_dir*: indicates a directory into which store the EPANET files that are uploaded by the user. Default value: 'epanet_uploads'
- *upload_max_size*: indicates the maximum dimension (in MegaByte) of a EPANET file. Default value: '10'
- *download_dir*: indicates a directory into which store the EPANET files generated by the system that will have to be downloaded by the user. Default value: 'epanet_download'
- *download_filename_suffix*: represents the suffix that is inserted in the file name before it is downloaded by the user. Default value: '-WIZ-GENERATED'
- *junction_prefix*: represents the prefix that is placed before the name of the junctions inserted by the system inside an EPANET file. Default value: 'WIZ_'

- *src_download_folder*: indicates the directory that contains the archive of the source code of the web portal. Default value: '/download/'
- *src_zip_file*: indicates the name of the archive file in a zip format of the source code of the web portal. Default value: 'wiz_latest.zip'
- *src_tar_file*: indicates the name of the archive file in a tar format of the source code of the web portal. Default value: 'wiz_latest.tar'
- *shp_upload_folder*: indicates the directory into which the shapes are stored that are uploaded by the users. Default value: '/../uploads/shp/'
- *transition*: composed parameter. Contains information relating to the transition of state of a water request:
  - *upload_dir*: indicates the directory in which to store the archive that has to be attached to a water request after the technical approval. Default value: 'uploads/wrut'
  - *upload_max_size*: indicates the maximum dimension (in MegaByte) of the archive. Default value: '10'
- *plugins*: composed parameter. Contains information relating to the plugins:
  - *upload_dir*: indicates the directory in which to store the plugin that are uploaded. Default value: 'plugins'
  - *upload_max_size*: indicates the maximum dimension (in MegaByte) of the plugin. Default value: '20'

Glossary

PE = AE = (Abitanti equivalenti) Population equivalent

ds = dg = maximum daily municipal supply (dotazione comunale giornaliera massima)

ys = da = (dotazione comunale annuale media) average annual municipal supply

sul = (superficie utile lorda) gross usable surface

ui = (unità immobiliari) property unit

sup = (superficie territoriale) territorial surface

UTOE = (Unità Territoriali Organiche Elementari) ‘primary harmonic territorial units’, i.e. smallest territorial portions that can systemically tackle urban and territorial issues

Appendix A: Database Scheme