# User Management

## Responsibilities

The UserManagement component is responsible to maintain a list of rights/permissions for the users.

The access rights define what a user can do or cannot in the application. Almost every page, service, data, or functionality must be protected by access rights.

Rights can be assigned directly to a user, or through roles. A role is defined by a set of access rights, and then roles can be assigned to users. A user can have several roles.

The total rights a user has is the union of (1) the rights directly attached to this user (2) the union of the rights of all the roles of this user. When the same right is present several times, the less restricted is kept.

One particular role will be defined by default: Local administrator. A user having this role has all the rights and can do everything on the local database. This role should not be assigned to a normal user, but to a special dedicated user which will be used only when needed. This to avoid doing mistakes: if someone has all the rights each time he/she connects to the application, this may lead to mistakes.

## Dependencies

Components_UserManagement_Dependencies.wmf

UserManagement depends on Authentication component in order to validate username and password when a user login.

## Functionalities

TODO: use case



The component will keep in the session information about the user currently authenticated:

* Its domain
* Its username
* The token returned by the authentication system when the user has been authenticated
* The result of the computation of all its rights, by domain (to reduce DataBase access, as any page will needs security checks, we will store the computation as soon as we need it)

The component will provide functionalities to other components:

* Login and logout
* Check if the user has a specific right, in the current domain

And 2 events the other components may subscribe:

* Logged\_in: raised when a user logged in, so other components may populate information about the logged user (like its first name and last name…)
* Logged\_out: raised when a user logged out.

UserManagement provides also classes for the other components to specify access rights:



Access rights are organized per category, in order to be able to display them in an organized way to the user. A category is simply a list of rights, with a localized name.

Each *Access Right* is named (unique name used to identify it), and has a translation so it can be displayed to the user.

We want to be as precise as possible in the rights assignments (meaning it is better to define several rights instead of one very general). For example: “can see the list of users”, “can see the rights of a users”, “can see the roles of a user”, “can edit the rights of a user”, “can edit the rights of a role”, “can assign roles to users”……

But we want to keep the security checks simple, to avoid mistakes, and we want the rights to be consistent. For this a right may imply other rights. For example: “can edit the rights of a user” implies “can see the rights of a user”, else it would be non-sense. That means that anyone having the right “can edit the rights of a user” gets automatically the right “can see the rights of a user”.

More generally, the implications specified are represented by a map: for a given value of the current right, it implies a list of rights together with the value of those rights.

## Data



A user is defined by its domain and username.

A user can have rights attached directly through the table *UserRights*.

A user can have roles attached through the table *UserRole*.

Each role is defined by a name (in table *Role*), and has rights attached through the table *RoleRights*.

Each right is defined by a name, and a value which may have any type (it will be the responsibility of the component managing this right to handle the value accordingly).

## Access rights

The UserManagement component itself defines rights:

#### Read rights

* consult\_user\_list (Boolean): if true the user can see the list of users
* consult\_user\_roles (Boolean): if true the user can see the roles attached to the users
* consult\_user\_rights (Boolean): if true the user can see the rights another user has
  + Implies consult\_user\_roles = true

#### Write rights

* manage\_roles (Boolean): if true, the user can create, remove and edit roles
* assign\_roles (Boolean): if true, the user can assign or unassign roles to users
  + implies consult\_user\_list = true
  + implies consult\_user\_roles = true
* edit\_user\_rights (Boolean): if true, the user can assign or unassign rights directly to a user (without using a role)
  + implies consult\_user\_list = true
  + implies consult\_user\_rights = true

## Screens

TODO