# DAY 4 – Friday, 14th June 2019

- **How platform's authentication and authorization works?**
  - **Users**
  - **Roles**
  - **Creation and management of staff users**
- **Exploring Source Code docs**
- **How Build Report and distribute them to Staff Users**
- **System Architecture (Web Server, DBMS, etc.) and brief relation of the attendees about the progress of the project to replicate the platform's architecture in the different countries**

# Platform authentication and authorization

Summary



## 1. .NET membership and specific tables

The platform is based on .NET framework, and for the authentication it exploits the built-in membership provider. In particular, we defined a class named `MyMembershipProvider` which inherits from the class `MembershipProvider`.
In this new class many of the properties and method of the inherited class are overridden, and other new properties and methods are added in order to improve the user authentication management.
Thus, this class is used to manage the creation and deletion of users, the creation and deletion of related passwords (and their encryption), the creation of the session at each login to the platform.

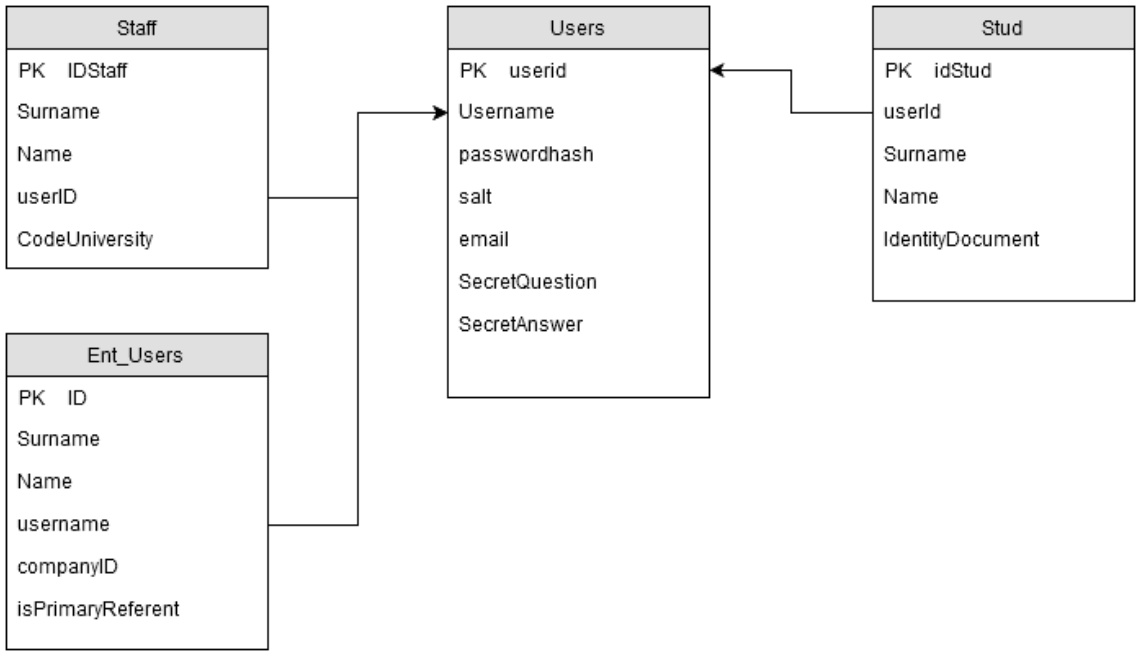
The table which manages the authentication is the table **`Users`**. This contains, in particular, these fields:

- `UserID`
- `Username`
  unique name of the user on the platform
- `PasswordHash` and `Salt`
  password stored in an encrypted mode, for authentication
- `Email`
  email address of the user, useful for the credential retrieval procedure
- `SecretQuestion` and `SecretAnswer`
  information used for the credential retrieval procedure
- `PasswordExpireDate`
  date which indicates when the password will expire
- `isLocked`
  Boolean which indicates whether the user is locked or not. In this case he won't access to the platform
- `isApproved`
  Boolean which can enable or disable a user from being authenticated
- `FailedPasswordAttemptCount` and `FailedPasswordanswerAttemptCount`
  Counters which indicate how many wrong attempt of login have been performed for a user. When the counter exceed a threshold, the user become locked and must be re-enabled.

Each login attempts to the platform is logged into the table **`tbAccessAttempts`**, whether it is successful or not.

### Specific tables

Basing on what type of user registers to the platform, we recognize 3 different tables where to set their personal information. Here follows the schema of the relations of these tables with the authentication table (`Users`)




## 2. Roles and authorizations

Below the user authentication, the system adopts a role based access control, so every user (which record is stored into the Users table) has got 1-N roles, depending on its type and authorization gained.

Every role gives access to specific privileges. The roles present into the platform (from platform roles) are the following:

- `Amministratore`
  the administrator role, has got the maximum power on the whole platform areas
- `Staff`
  role assigned to all the staff users. It enables directly the access to the multilingual editing area and to the staff report area.
- `Stud`
  role assigned to all the students
- `Ent`
  role assigned to a company user.
- `Helpdesk`
  particular staff user which enables the access to the help desk staff area
- `entstaff`
  particular staff user which enables the access to the enterprise management staff area
- `admData`
  particular staff user which enables the access to the administrative data management staff area

The system always checks the consistency between user roles and available areas/functions authorized per each role.

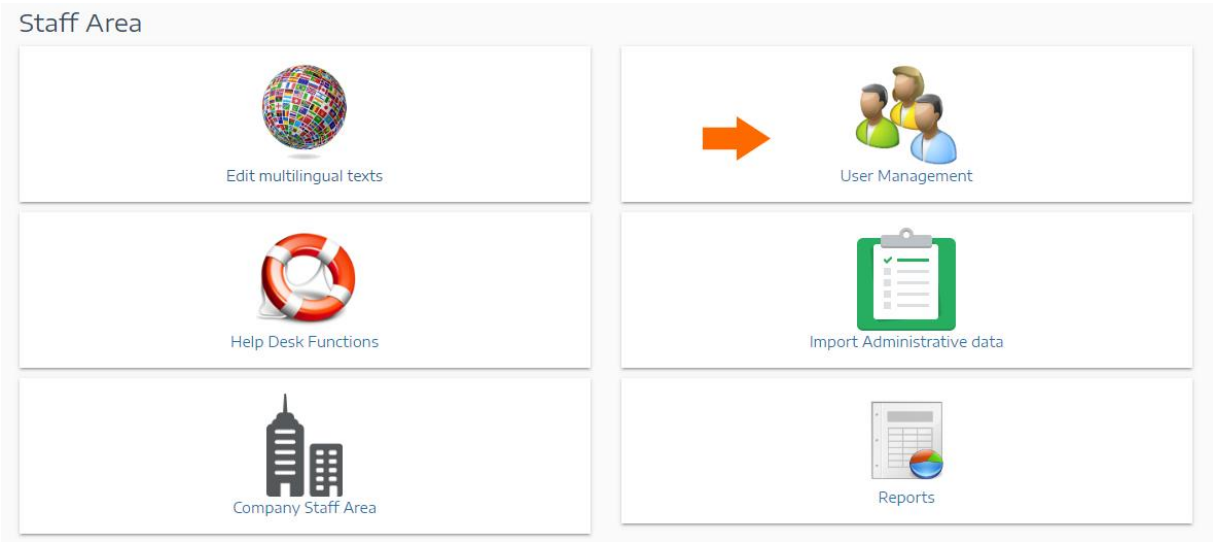
The 3 main roles per area (`stud`, `ent` and `staff`) are generally mutual exclusive. The only **exception** is the enterprise staff user which usually has both staff and ent role.
The roles `helpdesk`, `entstaff` and `admData` can be assigned to staff users in order to enable new features for him. It is possible to add more than one of these roles for every staff user.

The role is used also for the correct access to a specific site area (regulated via the `web.config authorization` section) and the correct visualization of the site navigation (via the sitemap).

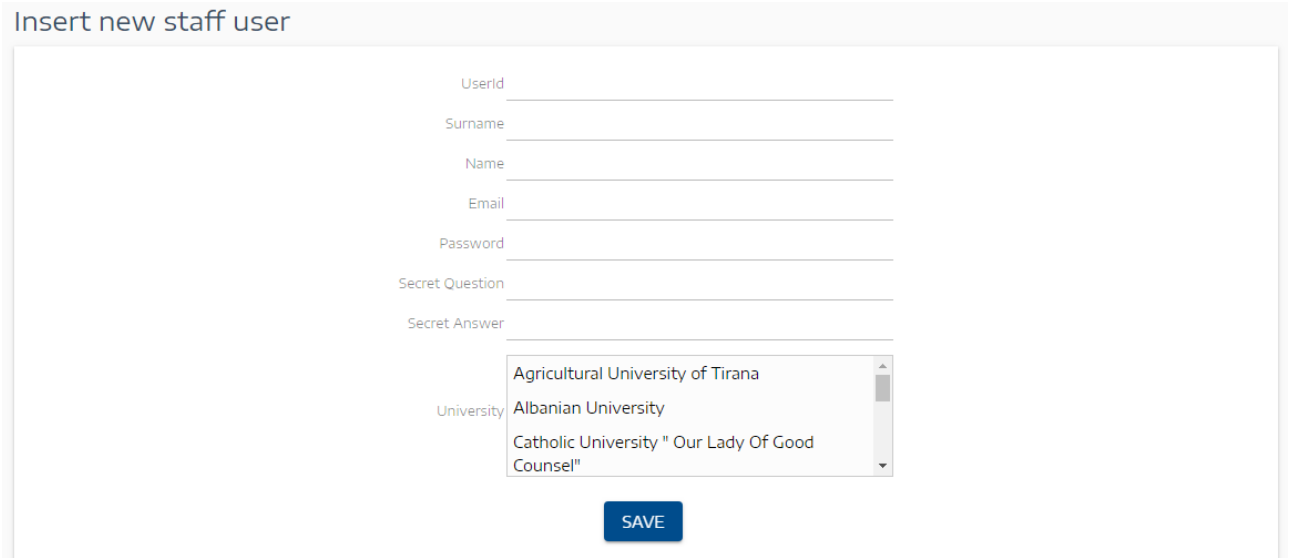
## 3. Creation and management of staff users

The administrator user is the only one which can access to the user management area, where he can create new users or assign different roles to them.

By accessing to the user management area it is possible to search a user and to edit its information.



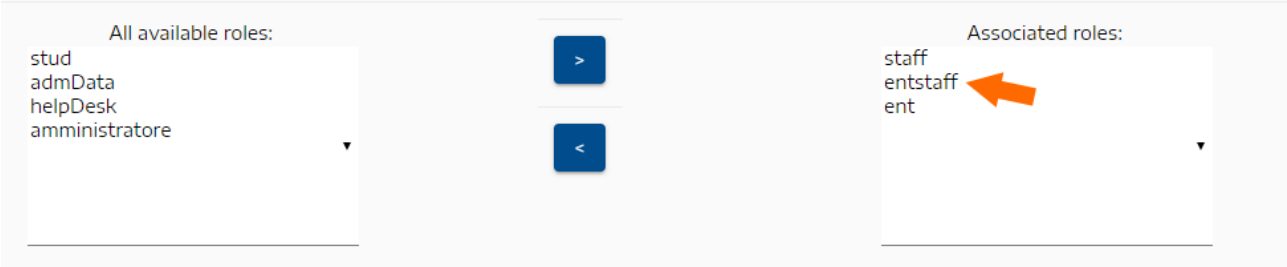

An administrator can create a new staff user, and needs to define the field requested in the form. In particular, if it is important to assign the correct university of the new user created.



After the creation, it is possible to add further roles to the staff user within the same page. A particular itinerary to follow is however requested for the creation of an enterprise staff user.

## How to create a new enterprise staff user

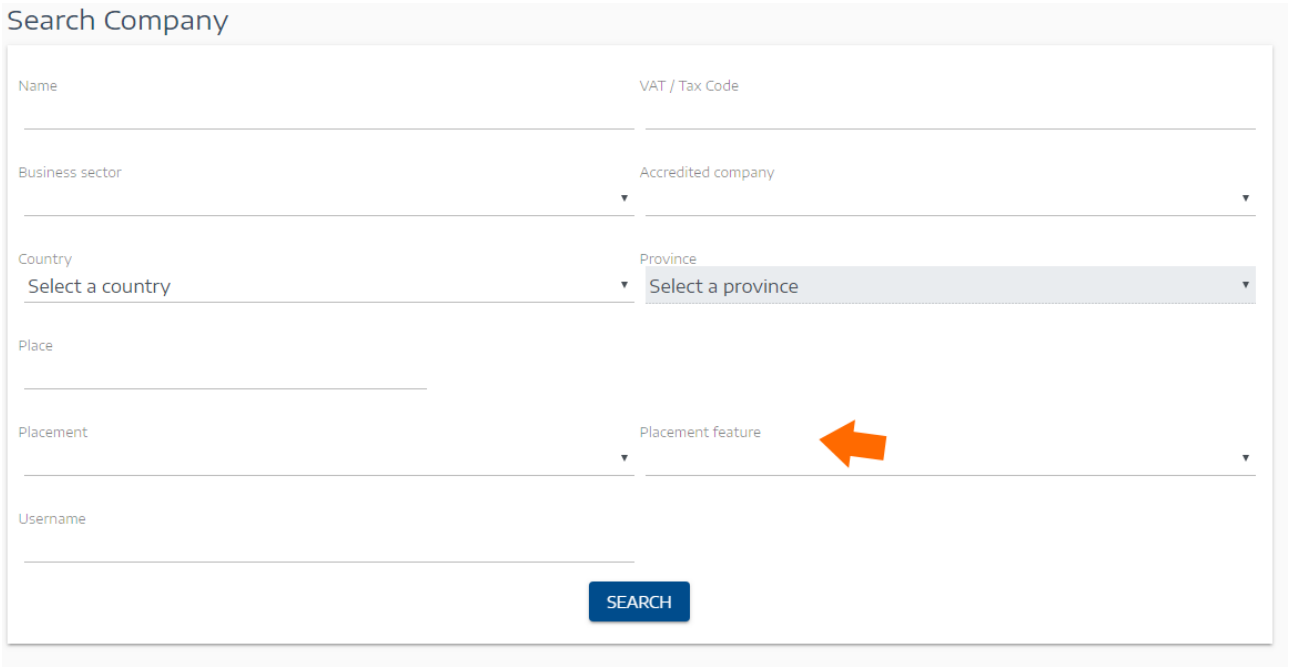
In order to create a new enterprise staff user it is necessary to follow a more complex itinerary than simply assigning them the correct role.
The first thing to do is to select the user (or also create a new one) and assign to him the role `entstaff`.



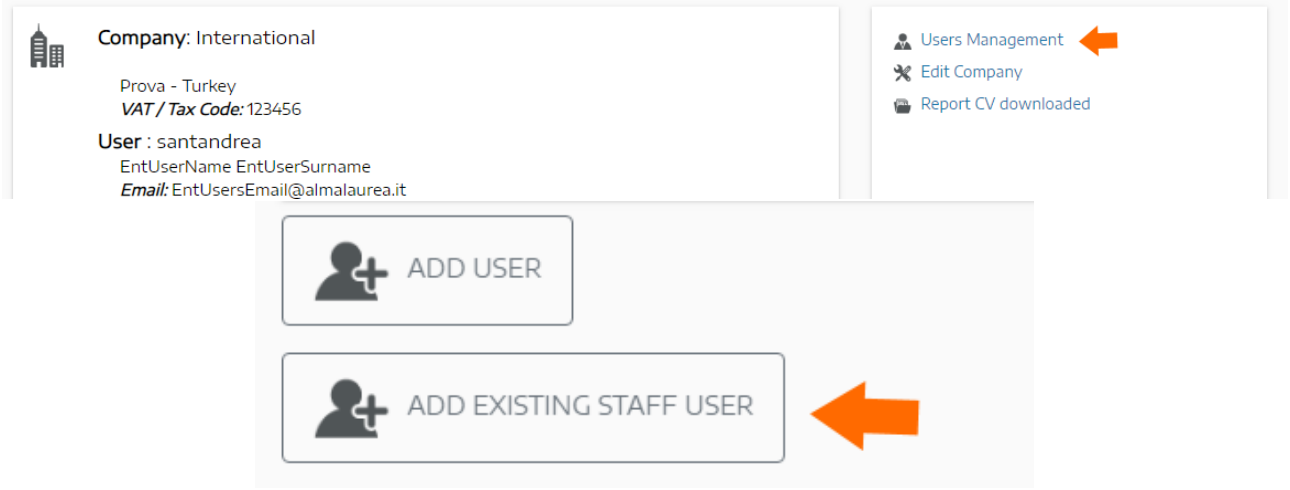

Usually, also the `ent` role is given to the user, so it will be able to simulate properly the behavior of a company user, as with the role he gains the access to the company area.

At the moment, the assignment of these roles does not allow the user to view none of the two areas of which he should have access to.
In fact, to complete the correct activation of the `entstaff` user, it is necessary to assign the user to a company which has the particular field named field `PlacementFeature` set to 1. In particular, this enterprise is the default one (named “International”).
This default company in reality is not a registered company like the others on the platform, but this is a special company deliberately created for the staff usage.

The administrator then need to access the companies’ management area search engine and search the company having the `PlacementFeature` field set to `True`.




The result company is named “International” and has ID=1. Then, access to its user management and add a new existing staff user. This action is available only to the administrator.

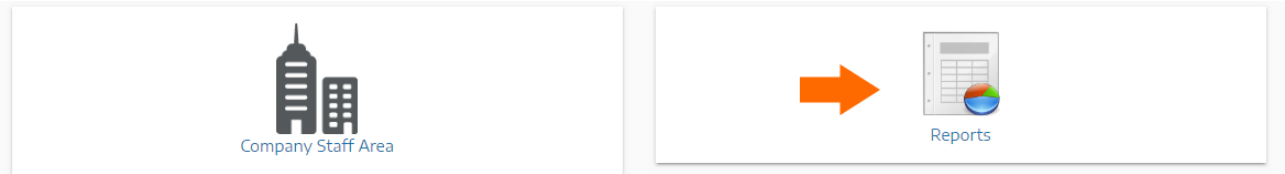



Finally, it is necessary to assign to this company the newly created staff user, by clicking on “Assign company” (on picture: arrow 1). Once clicked, the result will be visible in the last column on the right (on picture: arrow 2).


In this way the enterprise staff user will be eventually capable to access its reserved area.

# How to Build Reports and distribute them to Staff Users

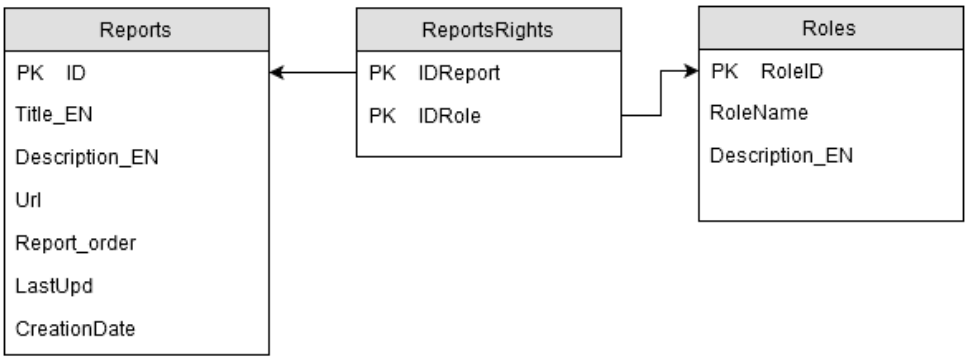
Every staff user can access to the report area of the website. This area contains several reports which are useful to check the situation of the platform from several points of view, for instance the number of registered students or the number of completed questionnaires.




Every report which is visible into the platform is a SQL file that is physically stored into the folder `App_Data/customSql` in the application.

All the classed that manage the report management are part of the `Report` namespace.

This is a resume of the tables which are used for the report handling:




To see correctly the report into the report area, it is also necessary to insert a record into the table Reports, which mainly indicates a multilingual version of the link and the URL of the report (the link of `App_Data/customSql` physical file).

It is also necessary to give the permission for every report to a given role, via indicating a correct couple (`IdReport`,`IdRole`) into the table `ReportsRights`.

Every report is built by using several special placeholders that let the usage of specific function for visualization.

## Example report

The example report that we analyze is the **Report by faculty and course of study with details for each student.**

This is the query that is run for the report (please notice the particular red options):

```
SELECT C.DegreeInstitute As Faculty, C.DegreeTitle_en + ' (' +
CAST(C.DegreeLevel AS VARCHAR(1)) + ')' As Course,
S.Name,S.Surname,ISNULL(S.[NationalTaxNumber],SUBSTRING(S.[identitydocument],3,L
EN(S.[identitydocument]))) AS IdentificationCode,
```

```
ISNULL(S.email,'') AS Email,ISNULL(S.telephone1,'') AS
Telephone,ISNULL(S.mobile1,'') AS Mobile,
ISNULL(dec_Country.Description_en,'') AS [Residence
Country],ISNULL(tbProvince.description_en,'') AS [Residence Province],
ISNULL(S.ResiPlace,'') AS [Residence Place],ISNULL(S.ResiAddress_EN,'') AS
[Residence Address],ISNULL(S.ResiZip,'') AS [Residence Zip],
SUM(D.QuesComplete) AS [Completed questionnaires],
CASE WHEN (SUM(CASE WHEN Q.UltimaPagina>0 THEN 1 ELSE 0 END) -
SUM(D.QuesComplete)) < 0 THEN 0
ELSE SUM(CASE WHEN Q.UltimaPagina>0 THEN 1 ELSE 0 END) - SUM(D.QuesComplete) END
AS [Questionnaires only started],
SUM(CASE WHEN D.IdStatus=1 THEN 1 ELSE 0 END) AS [Certificate degree]
FROM Stud As S
INNER JOIN Users AS U ON S.UserId = U.UserName
INNER JOIN CV_DiplomasDegrees As D ON S.idStud=D.idStud
INNER JOIN tbCourseOfStudy AS C ON C.DegreeCode = D.CodeCourse
LEFT JOIN dec_Country ON dec_Country.Code=S.ResiCountry
LEFT JOIN tbProvince ON tbProvince.code=S.ResiProv AND tbProvince.valid=1
LEFT JOIN Ques_Compilazioni AS Q ON Q.IdStud = S.idStud AND
D.CodeCourse=Q.CodCorso AND Q.StatoCompilazione=1
WHERE D.IdStatus IN (1,3) AND QuesComplete<>-1 AND S.IsTest=0 AND U.isApproved=1
AND <whereClause>D.CodeUniv IN (%%Univ:Univ:Univ%%)</whereClause>
AND <whereClause>D.CodeInstitute IN (%%Faculty:Faculty:Faculty%%)</whereClause>
GROUP BY S.idstud, S.Name,S.Surname,S.email,S.telephone1,S.mobile1,
dec_Country.Description_en,tbProvince.description_en,S.ResiPlace,S.ResiAddress_E
N,S.ResiZip,
ISNULL(S.[NationalTaxNumber],SUBSTRING(S.[identitydocument],3,LEN(S.[identitydoc
ument]))), C.DegreeInstitute, C.DegreeTitle_en + ' (' + CAST(C.DegreeLevel AS
VARCHAR(1)) + ')', D.CodeUniv, D.CodeCourse
ORDER BY D.CodeUniv, C.DegreeInstitute, C.DegreeTitle_en + ' (' +
CAST(C.DegreeLevel AS VARCHAR(1)) + ')',S.surname,S.name
```

## How do report placeholders work?

**`whereClause`**: the presence of the tags **`<whereClause>`** and **`</whereClause>`** indicates that there is a placeholder within them that needs to be replaced.

In particular, there can be tags in the form **%%firstToken:secondToken:thirdToken%%** , where**:**

- `firstToken`
  has a purely mnemonic utility (for example Univ indicates that this is a placeholder for a University code)
- `secondToken`
  contains a string mapped in code application (possible values are listed later)
- `thirdToken`
  is the label that appears next to the field in the form of the page.

The secondToken placeholder can be of these types:

1. `Univ`
   this placeholder is replaced with the University code of the logged user (table `Staff`, column `CodeUniversity`)

2. `Faculty`
   it adds a drop-down on the web page, containing the list of the Faculties that belong to the user's University.
3. `Data`
   it adds a form calendar on the web page
4. `Select-List`
   it adds a drop-down list on the web pages, containing the values indicated in the `thirdToken` (in the `thirdToken` these values must be separated by |)

**Example for a custom TOP N results dropdown**

```
%%TOP:Select-List:View:Top 50|TOP 50;Top 100|TOP 100;Top 1000|TOP 1000;All|%%
```