USERS CREDENTIALS SERVICE

This service runs on port **8019**.

This service allows to manage users. It contains CRUD operations like **add** or **get**.
Information are returned with **REST API**.

REST API are located in **UserController.java** class.

- **getAllUser( )** → http://localhost:8019/api/users/get/all


```
[
    {
        "userid": "Felice",
        "password": "01234",
        "active": true,
        "role": "USER"
    },
    {
        "userid": "Admin",
        "password": "87960",
        "active": true,
        "role": "ADMIN"
    }
]
```

- **getUserByUserID( )** → http://localhost:8019/api/users/get/user/{userID}


```
{
    "code": "200 ok",
    "message": "user Felice found"
}
```

- **addNewUser( )** → http://localhost:8019/api/add


```
{
    "userid": "Felice",
    "password": "01234",
    "active": true,
    "role": "USER"
}
```

- **addNewUsers( )** → http://localhost:8019/api/users/add

```
[
    {
        "userid": "Felice",
        "password": "01234",
        "active": true,
        "role": "USER"
    },
    {
       "userid": "Admin",
       "password": "87960",
       "active": true,
       "role": "ADMIN"
    }
]
```

Other methods returns user model to manage application security and in fact are invoked by **CustomUserDetailsService** with R**estTemplate**.

- **fetchUserByUserId( )** → http://localhost:8019/api/users/fetch/user/{userId}
- **fetchUsers( )** → http://localhost:8019/api/users/fetch/users

STUDENTS WEB SERVICE

This service runs on port **8080**.

This service allows to manage students. It contains CRUD operations and information are returned with **REST API**.

All endpoints/APIs are protected by **Spring Security** framework and then it is necessary to provide user credentials (Basic Auth)
Authorizations and roles are specified in WebSecurityConfiguration class.
Users data are retrieved by CustomUserDetailsService class which uses rest template to invoke users credentials service Rest APIs.

REST API are located in **StudentController.java** class.

- **getStudents( )** → http://localhost:8080/api/students/get/all

- **addStudent( )** → http://localhost:8080/api/students/post/student

```
{
   …
}
```

Basic Auth
Username: Admin
Password: 87960

- **getStudentByID( )** → http://localhost:8080/api/students/get/student/id/{id}

```
{
   "id": 61270,
   "name": "Filippo",
   "surname": "Angrisani",
   "birthdate": "1994-06-02T22:00:00.000+00:00",
   "university": "Salerno",
   "active": true
}
```

Basic Auth
Username: Felice
Password: 01234

UNIVERSITY AUTH SERVER JWT

This service runs on port **9210**.

This server manges Token system.
REST API are located in **JwtAuthController.java** class and the **endpoint (REST API)** which returns token is:

- **createAuthToken( )** → http://localhost:9210/auth

```
{
   "username": "Felice",
   "password": "01234",
}
```

```
{
      “token”: “eyJhb…”
}
```

REST APIs, in this controller class, make use of **custom user details service** containing methods to access, retrieve and manage user information. The user information are located in a persistence layer, e.g. PostgreSQL.

In addition, it make use of class called **JwtTokenUtil** which contains methods like **generateToken()**.

This service will be called to get valid token to access to endpoints which are protected by Spring Security – JWT Auth.

COURSE WEB SERVICE

This service runs on port **8021**.

Firstly, to get information about courses, you need to get valid token through the **University Auth Server JWT**.

Once you get the token, if it is valid, you can access to courses endpoints.
The class which verifies the validity of token is **JwtTokenAuthFilter.java**.

REST API are located in **CourseController.java** class.

- **getCourses( ) →** http://localhost:8080/api/courses/get/all

```
[
   {                                                    Bearer Token
     "name": CLOUD COMPITING,                           Token: eyJhbG...
      "hours": "46",
      "prof": "2"
   },

   {
     ....
   }

]
```

- **addCourse( ) →** http://localhost:8080/api/courses/post/course/add

```
{                                                       Bearer Token
      "name": CLOUD COMPITING,                          Token: eyJhbG...
       "hours": "46",
       "prof": "2"
},
```

- **addCourses( ) →** http://localhost:8080/api/courses/post/courses/add

…

DATABASES

As databases we have different PostgreSQL.

To access to PostgreSQL client PGAdmin, go to url **http://localhost:89**.
Number of port is specified in the **docker-compose** and in this case we specified 89.