Versión 1.1
El sistema deberá gestionar permisos/accesos a recursos de información de los usuarios.

```
let myResourceHandler = new resourceHandler();

let myAccesHandler = new accesHandler();
```

```
let myUserhandler =  new userHandler();

let myGrouphandler =  new groupHandler();

user.username = "pedrito",
user.password = "12345",

userData.name = "Kevin",
userData.surname = "Taylor",
userData.dni = "42354009",
userData.telephone = "2234561788",
userData.gender = "Male",
userData.address = "9 de julio 5650",
userData.email = "kevintylo@hotmail.com",
userData.isActive = true,

group.name =  "Textil"
```

```
resource.id = 1,
resource.name = 'Leer usuario',
resource.url = 'localhost/seminario/test/userhanlder/readuser'
```

La api contiene unos objetos “mock” para probar las distintas funcionalidades para gestionar usuarios y grupos.

userHandler: Cargar un usuario a la base de datos, actualizar los datos de un usuario en la base de datos, ingresar información personal de un usuario en la base de datos, borrar un usuario de la base de datos, mostrar un usuario guardado en la base de datos, mostrar todos los usuarios guardados en la base de datos, mostrar a que grupos pertenece el usuario.
groupHandler: Agregar un grupo a la base de datos, borrar un grupo de la base de datos, actualizar un grupo de la base de datos, mostrar la información de un grupo guardado en la base de datos, agregar un usuario a un grupo, borrar a un usuario de un grupo.
resourceHandler: Agregar un recurso a la base de datos, actualizar información de un recurso a la

base de datos, eliminar un recurso de la base de datos, leer la información de un recurso en la base de datos.
accesHandler: Dar accesos de un grupo a un recurso, quitar accesos de un grupo a un recurso, mostrar a que recursos puede acceder un grupo, mostrar cuantos grupos tienen acceso a un recurso, mostrar que usuarios tienen acceso a un recurso.

```
//myUserhandler.insertUserInDB(user);
//updateUser(user,2)
//myUserhandler.showAllUsers();
//myUserhandler.deleteUser(3);
//myUserhandler.showAllUsers();
//myUserhandler.insertUserDataInDB(user,1);
//myUserhandler.readUser(1);
//myUserhandler.getGroupMembership(1);


//myGrouphandler.insertGroupInDB(group);
//myGrouphandler.deleteGroup(2);
//myGrouphandler.updateGroup(group,1);
//myGrouphandler.readGroup(1);
//myGrouphandler.addUserInGroup(2,1);
//myGrouphandler.removeUserFromGroup(1,2);
```

```
 //myResourceHandler.insertResourceInDB(resource);
 //myResourceHandler.readResource(resource.id);
 //myResourceHandler.updateResource(resource,2);
 //myResourceHandler.deleteResource(2);

 //myAccesHandler.addAccesToGroup(2,2);
 //myAccesHandler.removeAccesFromGroup(1);
 //myAccesHandler.getGroupAcces(1);
 //myAccesHandler.getResourceAcces(2)
 //myAccesHandler.getUserAcces(user.id);
```

Estas funcionalidades pueden ser utilizadas des-comentando el método que desee probar.

```
const {user, userData, group, resource} = require('./objects.js')
const {userHandler} = require('./userhandler.js')
const {groupHandler} = require('./grouphandler.js');
const {resourceHandler} = require('./resourcehandler.js');
const {accesHandler} = require('./acceshandler.js');
```

Los objetos utilizados están guardados en objects.js y se los llama para asignarles valores. El userHandler se encarga de todas las funcionalidades referidas a la gestión de usuarios. El groupHandler se encarga de todas las funcionalidades referidas a la gestión de grupos.

Se agrega el objeto “recurso”, también se agregan el resourceHandler, encargado de gestionar a los recursos que pueden acceder los usuarios, y también se agrega al accesHandler, encargado de gestionar los accesos a los diferentes recursos.