**Documento de especificación de la API**

**19/4/2023 : Versión 1.0**

-El sistema deberá gestionar usuarios

-El sistema deberá gestionar grupos

Para ello mediante un análisis de dichos requerimientos se llega a la conclusión de que se necesitaran de 4 interfaces, las cuales se detallan a continuación.

**<<Interface>> User** quien contara con métodos como los siguientes:

getID(): string

getIsActive(): boolean

**<<Interface>> USERHandler** quien contara con métodos como los siguientes:

Create(UserData: data): boolean

Remove(string: id): boolean

Update(string: id):boolean

Read(string: id): UserData

getGruptMembership(string: id): list<string>

**<<Interface>> Group** quien contara con métodos como los siguientes:

getID(): string

getAmountOfUsers(): string

getIsActive(): boolean

**<<Interface>> GroupHandler** quien contara con métodos como los siguientes:

Create(UserData: data): boolean

Remove(string: id): boolean

Update(string: id):boolean

Read(string: id): UserData

addUser(string: id): boolean

removeUser(string: id): boolean

**Especificación de datos(UserData):**

ID: string

name: string

surname: string

DNI: string

phone: string

gender: string

address: string

mail: string

role: string

**Especificación de datos(GroupData):**

ID: string

name: string

isActive: boolean

**10/5/2023: Versión 1.1**

**Especificación de métodos DatabaseHandler: NOS PERMITIRA CONECTARNOS A LA BASE DE DATOS-LLAMAR SUS PROCEDIMIENTOS ALMACENADOS Y LUEGO DESCONECTARNOS**

connect()

callStoredProcedure(…args)

disconnect()

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

Se plantea de manera que necesitaremos de dos interfaces principales quienes nos darán el comportamiento necesario para acceder al recurso y una interface para interactuar directamente con el recurso en si.

Se plantea que:

**<<Interface>> Access** quien contara con métodos como los siguientes:

getID(): string

getIsActive(): boolean

**<<Interface>> AccessHandler:** quien contara con métodos como los siguientes

Create(AccessData: data): boolean

Remove(string: id): boolean

Update(string: id):boolean

Read(string: id): UserData

AuthorizeAccessToResource(string: groupID,string: resourceID): boolean

RemoveAccessToResource (string: groupID,string: resourceID): boolean

getGroupAccessByResource(string: resourceID): list<string>

getResourceAccessByGroup (string: groupID): list<string>

**<<Interface>> Resource** quien contara con métodos como los siguientes:

getID(): string

getIsActive(): boolean

**<<Interface>> ResourceHandler**:

Create(resourceData: data): boolean

Remove(string: id): boolean

Update(string: id):boolean

Read(string: id): ResourceData

Luego contaremos con dos estructuras que hasta el momento no exhiben comportamiento por si mismas.

**Especificación de datos(RESOURCEDATA):**

ID: string

name: string

isActive: boolean

resourceData: list<string>

**Especificación de datos(ACCESSDATA):**

ID: string

managmentLevel: string

name: string

**23-08-2023 version 1.2**

Para la permitir efectuar la autenticación por nombre de usuario y contraseña se plantea que deberemos contar con las siguientes clases:

**<<Interface>> SessionHandler**:

logIn(string: username, string: password): string

logOut(): boolean

***Mecanismos de autenticación y autorización***

**05-09-2023 version 1.0**

Se deberá implementar un mecanismo de cifrado irreversible a la contraseña,en este caso implementar el hash SHA-256 (Secure Hash Algorithm 256 bits),dicha hash se deberá almacenar en la base de datos en el campo contraseña remplazando el valor de esta.

A fin de manejar el cifrado se presenta la siguiente interface:

**<<Interface>> EncryptionHandler**:

encrypt(string: password): string