NEST Duro de Roer - Usuarios

- Seguimos la misma estrategia que en la lección anterior
- Creo la carpeta modules, dentro de la carpeta creo el módulo users con el CLI de Nest
- Creo también el servicio y el controlador
- En el controller coloco api/v1

Creando el dto (Data Transfer Object)

- Es una clase común

```
export class UserDto{

    id: number
    name: string
    email:string
    birthDate: Date
}
```

- Para validar uso class-validator y class-tarnsformer (lo instalo)

> npm i class-validator class-transformer

- Añado los decoradores
- El @Type es de class-transformer. Transforma la data
- Le voy a pasar un string con una data en el body. Quiero que la transforme a tipo Date

```
import {IsNumber, IsNotEmpty, IsString, IsEmail, IsDate} from 'class-validator'
import {Type} from 'class-transformer'

export class UserDto{

    @IsNumber()
    @IsNotEmpty()
    id: number

    @IsString()
    @IsNotEmpty()
    name: string

    @IsEmail()
    @IsNotEmpty()
    email:string

    @IsDate()
    @IsNotEmpty()
```

```
    @Type(()=> Date)
    birthDate: Date
}
```

- Para que haga las validaciones uso ValidationPipe en el main antes del app.listen

```
app.useGlobalPipes(
    new ValidationPipe({
        whitelist: true,            //si hay data que no está en el dto la
ignorará
        forbidNonWhitelisted: true, //muestra error si le mando data que no está
en el dto
        transform: true             //permite transformar la data
    })
)
```

Creando Usuarios

- Como no uso una db creo el array en el servicio
- service

```
@Injectable()
export class UsersService{
    private _users: userDto[]

    constructor(){
        this._users = []
    }
}
```

- En el controller está inyectado el servicio en el constructor
- controller

```
@Post()
createUser(@Body() user: UserDto){
    return this.usersService.createUser(user)
}
```

- Creo el servicio createUser
- service

```
createUser(user: UserDto){
    const userFound = this._users.find(u=> u.id === user.id)

    if(!userFound){
        this._user.push(user)
        return true
    }else{
        return false
    }
}
```

Obteniendo usuarios filtrados

- Vamos a filtrar los usuarios por la fecha de nacimiento
- Como puse el transform en true en el ValidationPipe, lo va a transformar a date si es un string con el que puede hacerlo
- Creo el GET en el controlador
- controller

```
@Get()
getUsers(@Query('start') start: Date, @Query('end') end: Date){
    return this.usersService.getUsers(start, end)
}
```

- En el service uso el isNaN para validar si la fecha es válida o no
- Si es una fecha correcta isNaN me va a devolver false
- Puedo negarlo con ! para evaluar lo contrario, es decir **SI SI ES UN NUMERO**
- Tendríamos los casos en el que me pasan las dos fechas, solo una de las dos o ninguna de las dos
- service

```
getUsers(start: Date, end: Date){

    if(!isNaN(start.getTime()) && !isNaN(end.getTime())){
        return this._users.filter(u => u.birthDate.getTime() >= start.getTime() &&
                                       u.birthDate.getTime() <= end.getTime())

    }else if(!isNaN(start.getTime()) && isNaN(end.getTime())){
            return this._users.filter(u => u.birthDate.getTime() >=
start.getTime())

    }else if(isNaN(start.getTime()) && !isNaN(end.getTime())){
            return this._users.filter(u => u.birthDate.getTime() <= end.getTime())
    }else{
        return this._users
    }
}
```

Actualizando Usuarios

- En el caso de que el id no exista no tiene que dar el fallo, tiene que crearse
- En el caso de que exista debe actualizarse
- Se suele hacer así
- Creo el endpoint en el controlador
- controller

```
@Put()
updateUser(@Body() user: UserDto){
    return this.usersService.updateUser(user)
}
```

- En el servicio uso createUser
- Si el usuario existe me va a devolver un falso, con lo que hay que actualizar
- En caso de que exista vamos a buscar el índice en el que se encuentra
- findIndex en lugar de devolver el objeto devuelve el index
- service

```
updateUser(user: userDto){

    const userAdded = this.createUser(user)

    if(!userAdded){
        const index = this._users.findIndex(u => u.id === user.id)

        this._users[index] = user
    }

    return true
}
```

Eliminando usuarios

- controller

```
@Delete('/:id')
deleteUser(@Param('id') id: number){
    return this.usersService.deleteUser(+id)
}
```

- Voy al servicio
- Si index es distinto de -1 significa que existe el elemento. devuelve -1 cuando no lo encuentra
- splice, dado un indice borra n elementos
- service

```
deleteUser(id: number){
    const index = this._users.findIndex(u => u.id === id)

    if(index != -1){
        this._users.splice(index, 1)
        return true
    }
    return false      //si llega aquí es que no existe
}
```

Documentar Dto (ApiProperty)

```
import {IsNumber, IsNotEmpty, IsString, IsEmail, IsDate} from 'class-validator'
import {Type} from 'class-transformer'
import {ApiProperty} from '@nestjs/swagger'

export class UserDto{

    @ApiProperty({
        name: 'id',
        type: 'number',
        description: 'Identificador del usuario'
    })
    @IsNumber()
    @IsNotEmpty()
    id: number

    @ApiProperty({
        name: 'name',
        type: String,
        description: 'Nombre del usuario',
        required: true
    })
    @IsString()
    @IsNotEmpty()
    name: string

     @ApiProperty({
        name: 'email',
        type: String,
        description: 'Email del usuario',
        required: true
    })
    @IsEmail()
```

```
    @IsNotEmpty()
    email:string

    @ApiProperty({
        name: 'birthDate',
        type: Date,
        description: 'Fecha de nacimiento del usuario',
        required: true
    })
    @IsDate()
    @IsNotEmpty()
    @Type(()=> Date)
    birthDate: Date
}
```

- Pongo el POST del controlador como ejemplo de documentación

```
@Post()
@ApiOperation({
    description: 'Crea un usuario'
})
@ApiBody({
    description: 'Crea un usuario mediante un Dto. devuelve true si ha tenido
éxito',
    type: UserDto,
    examples:{
        value:{
            "id": 1,
            "name": "Migue",
            "email": "email@gmail.com",
            "birthDate": "1981-20-01"
        }
    }
})
```