

Casos de prueba:

Diagnosticos

Test de POST Pulmones

Parámetros:

puntada_lateral= true

fiebre= true

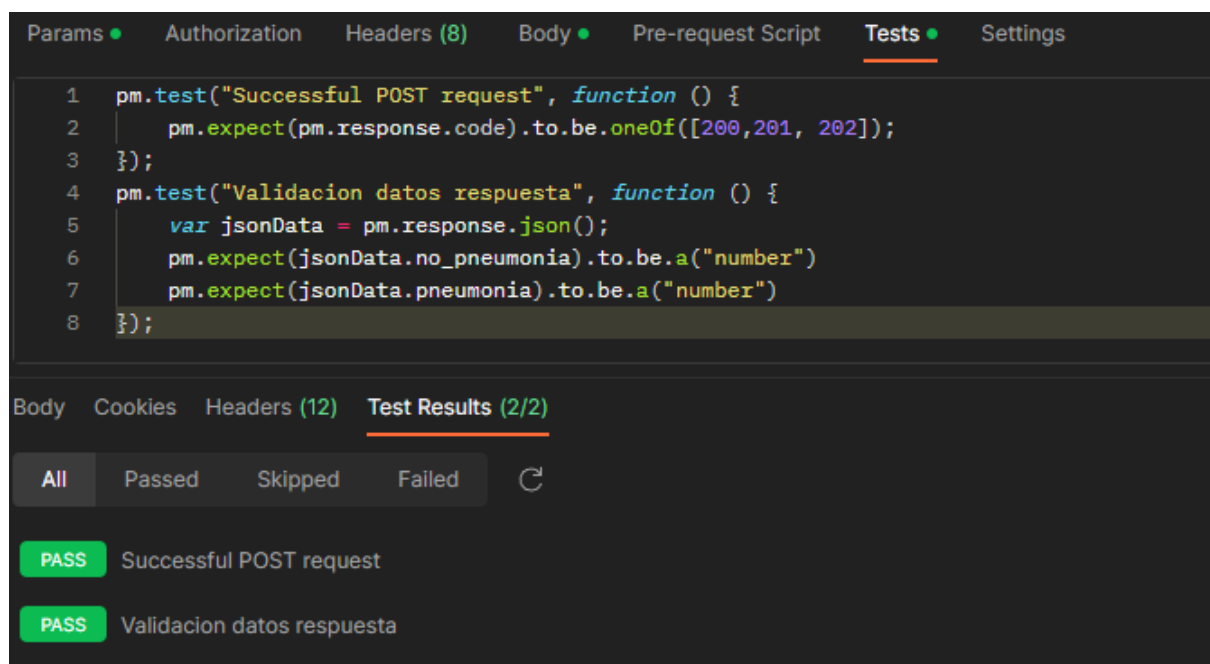
dificultad_respiratoria= true

id_usuario= 1

id_medico= 1

Dentro del Body una imagen de prueba

Casos de prueba:



Test de GET Historial

Parametros:

id_usuario = 1

rol_id = 1

Casos de prueba:

The screenshot shows the Postman interface for a GET request. The URL is `https://api-resultados.onrender.com/Diagnosticos/historial?id_usuario=1&rol_id=1`. The 'Tests' tab is active, displaying the following code:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 > var schema = {
50 };
51 pm.test("VALIDAR DATOS JSON", function () {
52   var jsonData = pm.response.json();
53
54   pm.expect(jsonData.historial).to.be.a("array");
55   for (var i=0; i<jsonData.historial.length;i++)
56     pm.expect(tv4.validate(jsonData.historial[i], schema)).to.be.true;
57
58 });
```

The 'Test Results' tab shows two passed tests:

- PASS** Status code is 200
- PASS** VALIDAR DATOS JSON

Schema:

```
var schema = {
```

```
  "type": "object",
```

```
  "properties": {
```

```
    "id": {"type": "integer"},
```

```
    "imagen": {"type": "string"},
```

```
    "datos_complementarios": {"type": "string"},
```

```
    "fecha": {"type": "string"},
```

```
    "usuario_id": {"type": "integer"},
```

```
    "usuario_medico_id": {"type": "integer"},
```

```
    "modelo_id": {"type": "integer"},
```

```
    "nombre_usuario": {"type": "string"},
```

```
    "modelo_nombre": {"type": "string"},
```

```

"nombre_medico": {"type": "string"}
},
"required": ["id", "imagen", "datos_complementarios", "fecha", "usuario_id", "usuario_medico_id",
"modelo_id", "nombre_usuario", "modelo_nombre", "nombre_medico"]
};

```

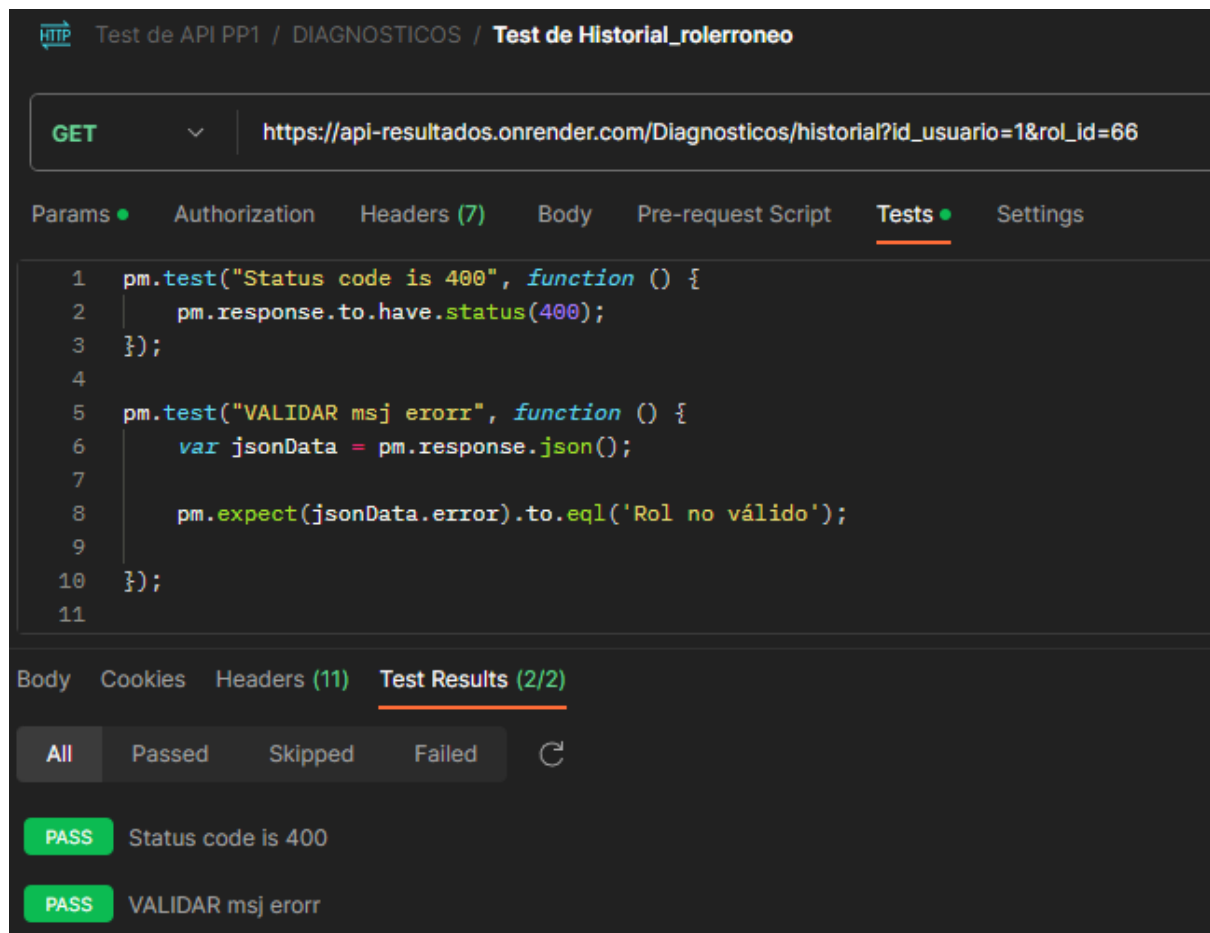
Rol erróneo

Parámetros:

id_usuario = 1

rol_id = 66

Casos de prueba:



Test de API PP1 / DIAGNOSTICOS / **Test de Historial_rolerroneo**

GET https://api-resultados.onrender.com/Diagnosticos/historial?id_usuario=1&rol_id=66

Params • Authorization Headers (7) Body Pre-request Script **Tests •** Settings

```

1 pm.test("Status code is 400", function () {
2   pm.response.to.have.status(400);
3 });
4
5 pm.test("VALIDAR msj error", function () {
6   var jsonData = pm.response.json();
7
8   pm.expect(jsonData.error).to.eql('Rol no válido');
9
10  });
11

```

Body Cookies Headers (11) **Test Results (2/2)**

All Passed Skipped Failed ↻

PASS Status code is 400

PASS VALIDAR msj error

Usuario no valido

Parametros:

id_usuario = 166

rol_id = 1

Casos de prueba:

The screenshot shows a REST client interface with the following details:

- URL:** `https://api-resultados.onrender.com/Diagnosticos/historial?id_usuario=166&rol_id=1`
- Method:** GET
- Tests Tab:** Contains two test cases:
 - Line 1: `pm.test("Status code is 200", function () {`
 - Line 2: `pm.response.to.have.status(200);`
 - Line 3: `});`
 - Line 5: `pm.test("VALIDAR msj error", function () {`
 - Line 6: `var jsonData = pm.response.json();`
 - Line 7: `pm.expect(jsonData.historial).to.be.a("array");`
 - Line 8: `pm.expect(jsonData.historial.length).to.eql(0);`
 - Line 9: `});`
 - Line 10: `});`
- Test Results Tab:** Shows two passed tests:
 - PASS** Status code is 200
 - PASS** VALIDAR msj error

Test de POST predecir cerebro

Parametros:

perdida_visual=true

debilidad_focal=true

convulsiones= true

id_usuario= 1

id_medico= 1

Dentro del Body una imagen de prueba.

Casos de prueba:

Test de API PP1 / DIAGNOSTICOS / **Test de predecir cerebro**

POST https://api-resultados.onrender.com/Diagnosticos/predecir/cerebro?perdida_visual=true&debilidad_focal=true&convulsiones=true&id_usuario=1&id_medico=1

Params • Authorization Headers (8) Body • Pre-request Script **Tests •** Settings

```
1 pm.test("Successful POST request", function () {
2   pm.expect(pm.response.code).to.be.oneOf([200, 201, 202]);
3 });
4 pm.test("Validacion datos respuesta", function () {
5   var jsonData = pm.response.json();
6   pm.expect(jsonData.glioma).to.be.a("number")
7   pm.expect(jsonData.meningioma).to.be.a("number")
8   pm.expect(jsonData.no_tumor).to.be.a("number")
9   pm.expect(jsonData.pituitary).to.be.a("number")
10 });
```

Body Cookies Headers (12) **Test Results (2/2)**

All Passed Skipped Failed C

PASS Successful POST request

PASS Validacion datos respuesta

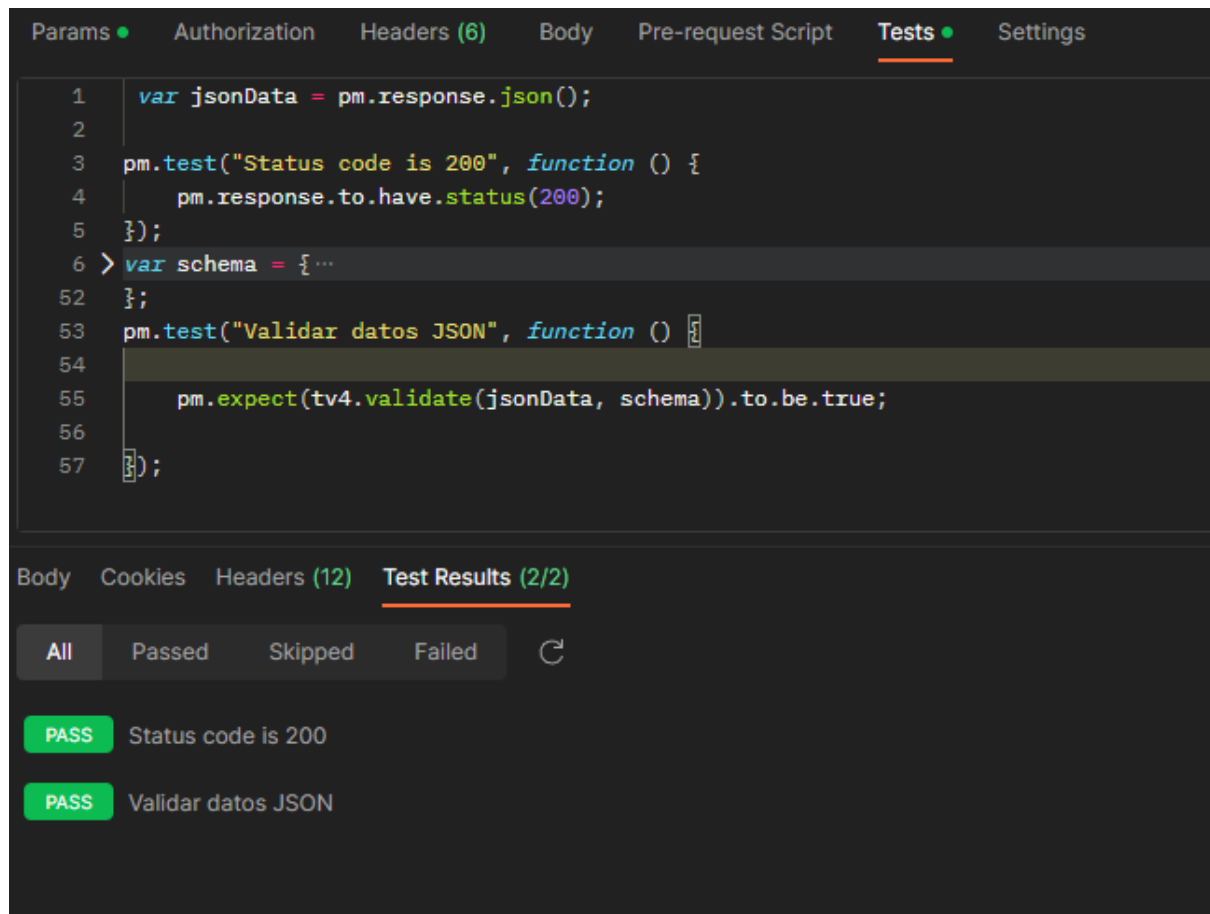
Test de GET Diagnósticos

Parametros:

rol_id:1

id:26

Casos de prueba:



Schema utilizado:

```
var schema = {
  "type": "object",
  "properties": {
    "id": {"type": "integer"},
    "imagen": {"type": "string" },
    "datos_complementarios": {"type": "string"},
    "fecha": {"type": "string"},
    "usuario_id": {"type": "integer" },
    "usuario_medico_id": {"type": "integer"},
    "modelo_id": {"type": "integer"},
    "nombre_usuario": {"type": "string" },
    "modelo_nombre": {"type": "string"},
    "nombre_medico": {"type": "string"}
  },
}
```

```
"required": ["id", "imagen", "datos_complementarios", "fecha", "usuario_id", "usuario_medico_id",  
"modelo_id", "nombre_usuario", "modelo_nombre", "nombre_medico"]  
};
```

Usuarios:

Test de POST AdminAlta

Parametros:

nombre= test

dni= test

email= test@test.com

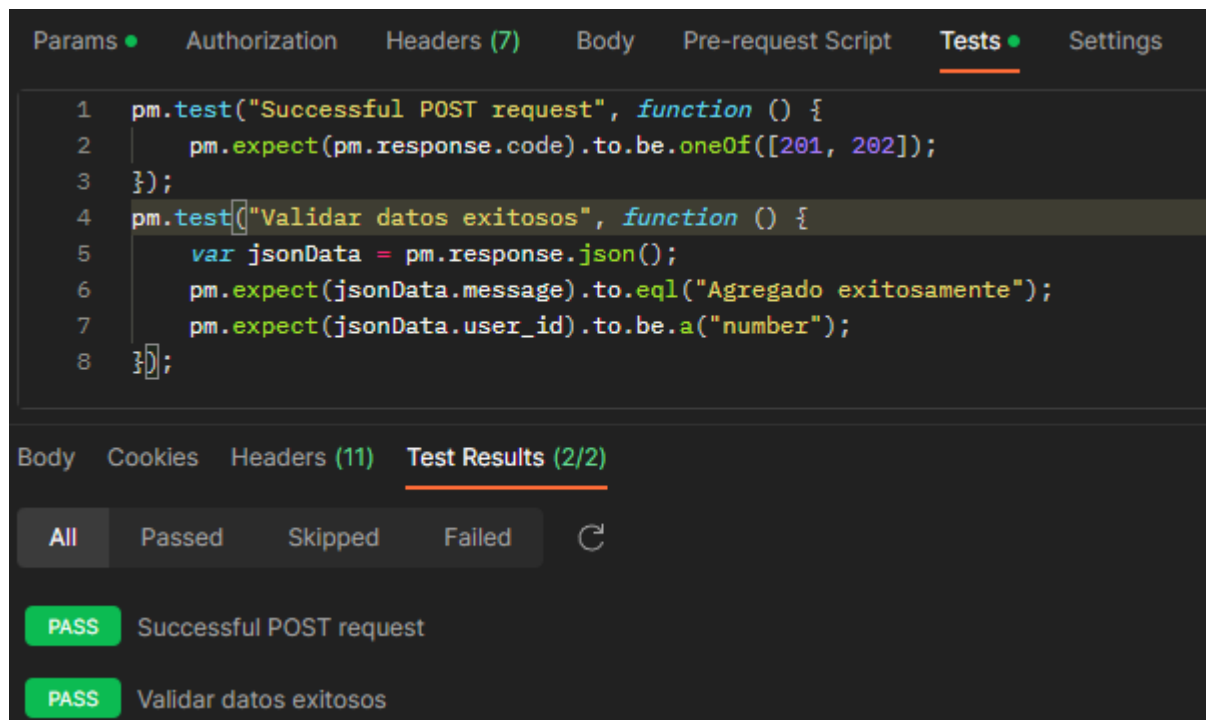
password= testtest

rol_id= 1

establecimiento_id= 1

especialidad= test

Casos de prueba:



para comprobar que el usuario esta creado, usaremos el GET UsuarioID con el ID que nos devuelve la API, en este caso fue ID= 85. Al consultarlo, nos devuelve el usuario recién creado:

```
{
  "id": 85,
  "nombre": "test",
  "dni": "test",
  "email": "test@test.com",
  "password": "testtest",
  "rol_id": 1,
  "establecimiento_id": 1,
  "fecha_ultima_password": "2023-10-23 13:59:57",
  "especialidad": "test"
}
```

Test de PATCH update-user-informacion

Para probar la actualización de datos del usuario creado anteriormente usaremos el campo "dni" para que lo encuentre y le modificaremos el nombre en esta ocasión

Parametros:

nombre:test2

dni:test

email:test@test.com

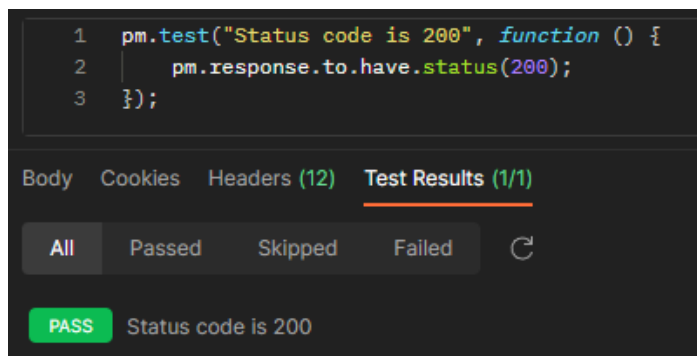
password:testtest

rol_id:1

establecimiento_id:1

especialidad:test

Para el caso de prueba corroboramos que el status code sea 200.



Una vez ejecutado el test, corroboramos la info del usuario con el GET UsuarioID

200

Response body

```
{
  "id": 85,
  "nombre": "test2",
  "dni": "test",
  "email": "test@test.com",
  "password": "testtest",
  "rol_id": 1,
  "establecimiento_id": 1,
  "fecha_ultima_password": "2023-10-23 14:20:09",
  "especialidad": "test"
}
```

Test de GET usuarios Médicos

Como la el endpoint devuelve la lista de todos los médicos, primero se corrobora que llegue la lista y luego corroboramos que la estructura de cada elemento de la lista cumpla el siguiente schema:

```
var schema = {
  "type": "object",
  "properties": {
    "id": {"type": "integer"},
    "nombre": {"type": "string"},
    "rol_id": {"type": "integer"},
    "establecimiento_id": {"type": "integer"},
    "establecimiento_nombre": {"type": "string"},
    "establecimiento_direccion": {"type": "string"}
  },
  "required": ["id", "nombre", "rol_id", "establecimiento_id", "establecimiento_nombre", "establecimiento_direccion"]
};
```

Casos de prueba:

The screenshot shows the Postman interface with the 'Tests' tab selected. The test scripts are as follows:

```
1 pm.test("Status code is 200", function () {
2   pm.response.to.have.status(200);
3 });
4 > var schema = { ...
34 };
35 pm.test("Validar datos respuesta", function () {
36   var jsonData = pm.response.json();
37   pm.expect(jsonData).to.be.a("array")
38   for(var i=0;i<jsonData.length;i++){
39     pm.expect(tv4.validate(jsonData[i], schema)).to.be.true;
40   }
41 });
```

Below the scripts, the 'Test Results (2/2)' tab is active, showing two passed tests:

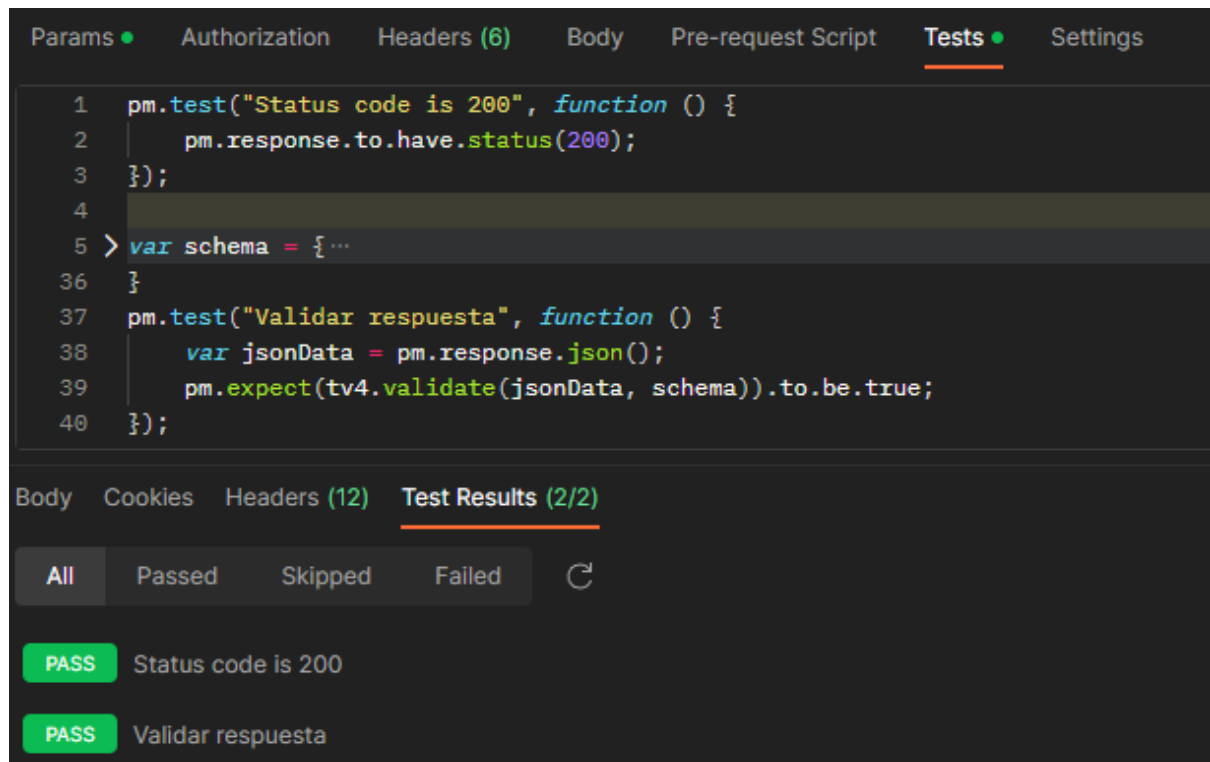
- PASS** Status code is 200
- PASS** Validar datos respuesta

Test de GET Usuario ID

Parámetros:

id=85

Casos de prueba:



Schema utilizado:

```
var schema = {  
  "type": "object",  
  
  "properties": {  
    "id": {"type": "integer"},  
    "nombre": {"type": "string"},  
    "dni": {"type": "string"},  
    "email": {"type": "string"},  
    "password": {"type": "string"},  
    "rol_id": {"type": "integer"},  
    "establecimiento_id": {"type": "integer"},  
    "fecha_ultima_password": {"type": "string"},  
    "especialidad": {"type": "string"}  
  },  
}
```

Comprobamos que el usuario es el que creamos en primer lugar:

```

1  {
2    "id": 85,
3    "nombre": "test2",
4    "dni": "test",
5    "email": "test@test.com",
6    "password": "testtest",
7    "rol_id": 1,
8    "establecimiento_id": 1,
9    "fecha_ultima_password": "2023-10-23 14:20:09",
10   "especialidad": "test"
11 }
```