

Resumen	2
Programas	2
InteliJ IDE	2
Postman	4
ADMIN	4
USER	4
Configuración	6
Roles	6
Data Pre-Cargada	6
Token	7
Rol ADMIN	8
POST - Token	8
POST - crearUsuario	9
GET - listarUsuarios	9
GET - obtenerUsuario	10
PUT - actualizarUsuario	10
PATCH - activarUsuario	10
PATCH - desactivarUsuario	11
DELETE - eliminarUsuario	11
Rol ADMIN – CREAR UN USUARIO Y SACAR TOKEN	12
POST - Token	12
Validar correo	13
Validar contraseña	13
Rol USER	14
POST - Token	14
POST - crearUsuario	15
GET - listarUsuarios	15
GET - obtenerUsuario	16
PUT - actualizarUsuario	16
PATCH - activarUsuario	17
PATCH - desactivarUsuario	17
DELETE - eliminarUsuario	18

Resumen

Manual de utilización de api con postman junto a pruebas unitarias. El alcance principal es que debe seguir los pasos de forma secuencial para probar el app. El app cuenta con acceso tanto para roles configurado para administrador y usuarios.

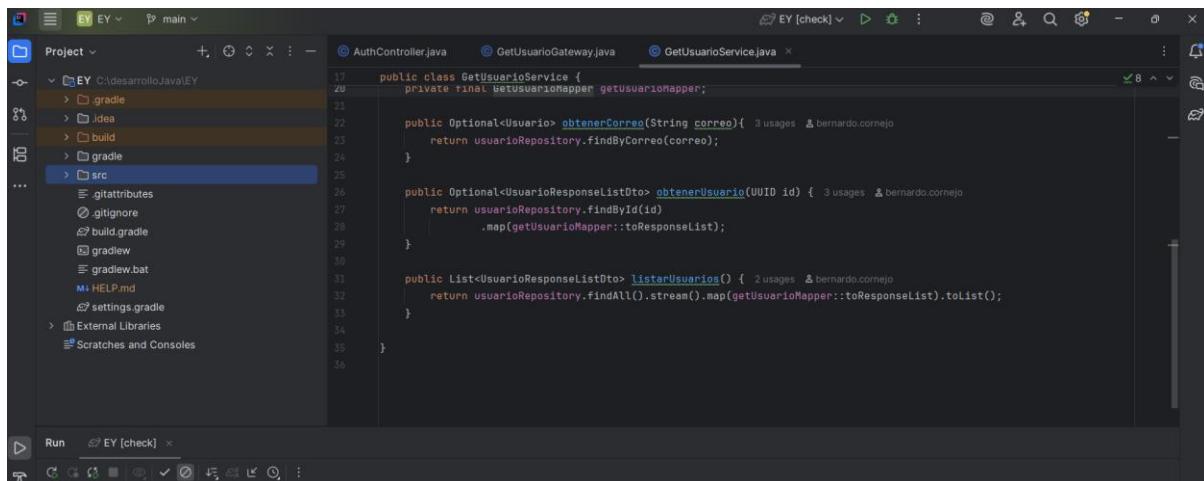
Programas

Para gestionar la prueba de la API en java debe tener Postman y Intelij IDE

Postman	Intelij IDE
	

Intelij IDE

Debe abrir Intelij IDE y seleccionar el proyecto EY



Debe presionar el botón Open para abrir el proyecto y esperar que cargar las dependencias de gradle. Terminada de carga de las dependencias y constructor del proyecto, ir a Main de java y ejecutar el app.

Se revisan los test

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project:** EY
- Run Tab:** Shows a green checkmark indicating 41 tests passed in 1 second and 798 milliseconds.
- Gradle Tool Window:** Shows the build configuration with the 'application' task highlighted.

Se ejecuta dos veces para iniciar el app. El proyectos se ejecutara en el puerto 8080

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project:** EY
- Run Tab:** Shows a green checkmark indicating the build was successful.
- Gradle Tool Window:** Shows the build configuration with the 'bootRun' task highlighted.

Postman

Abrir el app postman y presionar Import :

- Desafio-EY-ADMIN.postman_collection.json
- Desafio-EY-USER.postman_collection.json

Para una prueba completa se sugiere abrir los dos

ADMIN

The screenshot shows the Postman interface with the 'ADMIN' collection imported. On the left, the collection tree is visible with various endpoints like POST token, POST crearUsuario, etc. A red box highlights this area. On the right, a specific POST request for 'token' is selected. The 'Body' tab is active, showing a raw JSON payload:

```
1 {
2   "correo": "juan@example.com",
3   "contraseña": "Secret@123.$"
```

A red arrow points from the left side towards this JSON input field.

USER

The screenshot shows the Postman interface with the 'USER' collection imported. On the left, the collection tree is visible with various endpoints like POST token, POST crearUsuario, etc. A red box highlights this area. On the right, a specific POST request for 'token' is selected. The 'Body' tab is active, showing a raw JSON payload:

```
1 {
2   "correo": "elias.cornejo@coagra.cl",
3   "contraseña": "Eli@sC123.$"
```

A red arrow points from the left side towards this JSON input field.

Se cargaran todo los endpoint

Metodo	Nombre	Descripción
POST	Token	Genera token con autentificación JWT

POST	crearUsuario	Crea un nuevo usuario
GET	obtenerUsuario	Recupera un usuario por UUID
GET	listarUsuarios	Lista todos los usuarios
PUT	actualizarUsuario	Actualiza un usuario por UUID
PATCH	activarUsuario	Activa usuario por UUID
PATCH	desactivarUsuario	Desactiva usuario por UUID
DEL	eliminarUsuario	Elimina usuario por UUID

En el post Token, la clave se establece de forma automática en la variable. Esto permite generar una prueba de no impacto.

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Desafio-EY-ADMIN' and 'Desafio-EY-USER'. The main area shows a 'POST token' request to 'http://localhost:8080/auth/login'. The request body is a JSON object:

```
1 {  
2   "correo": "juan@example.com",  
3   "contraseña": "Secret@123.$"  
4 }
```

The response tab shows a 200 OK status with a response body containing user information:

```
1 {  
2   "data": {  
3     "id": "11111111-1111-1111-1111-111111111111",  
4     "creado": "2025-08-14T22:01:05.528523",  
5     "modificado": "2025-08-14T22:01:05.528523",  
6     "ultimoLogin": "2025-08-14T22:13:51.1258683",  
7     "token": "eyJzdWJ0IiOiJkVYQ1LcJhbcc1OIJUzI1N1j9.",  
8     "activo": true  
9   },  
10  "mensaje": null  
11 }
```

The screenshot shows the Postman interface with the following details:

- Header:** Authorization: Bearer
- Method:** GET
- URL:** http://localhost:8080/api/usuarios
- Body:** JSON (empty)
- Response:** Status: 200 OK - 192 ms - 1.46 KB
- Preview:** Shows two user objects in JSON format.

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Date: Mon, 10 Apr 2023 19:19:20 GMT
Content-Length: 143

[{"id": "23111111-1111-1111-1111-111111111111", "nombre": "Juan Pérez", "correo": "juan@example.com", "telefono": "+521 55555555", "creado": "2023-04-10T19:11:47.197600", "modificado": "2023-04-10T19:11:47.197609", "ultimoLogin": "2023-04-10T19:11:47.197699", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiMwIiwidHlwZSI6ImdwdXNlX2F1dGhvcml0eSIsInRpdGxlIjoiMTc0MjcyODEsImlhdCI6MTc0NDMyNzIxMjQ.", "activo": true, "telefonos": []}, {"id": "22222222-2222-2222-2222-222222222222", "nombre": "Pedro Ramírez", "correo": "pedro@gmail.com", "telefono": "+521 55555555", "creado": "2023-04-10T19:11:47.197600", "modificado": "2023-04-10T19:11:47.197609", "ultimoLogin": "2023-04-10T19:11:47.197699", "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiMwIiwidHlwZSI6ImdwdXNlX2F1dGhvcml0eSIsInRpdGxlIjoiMTc0MjcyODEsImlhdCI6MTc0NDMyNzIxMjQ.", "activo": true, "telefonos": []}]
```

Para validar si la autentificación es válida se debería sacar la variable {{authorization}} o sacar el establecimiento de la variable.

```

    var data = JSON.parse(responseBody);
    if(data.data.token == null){
        console.log("token generado : " + data.data.token);
        collectionVariables.set("authorization",data.data.token);
    }else{
        console.log("No token generated");
    }

```

Response Body:

```

    {
      "data": {
        "id": "11111111-1111-1111-1111-111111111111",
        "creado": "2025-04-10T19:11:47.197669",
        "modificado": "2025-04-10T19:11:47.197669",
        "ultimo_login": "2025-04-10T19:20:47.726799",
        "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1cHA1OjE3NDQzMjcyODExMlhdCi0MTc0NDMyNzIyMX0.L99xZ",
        "activo": true
      }
    }

```

Configuración

Usuarios: Se tiene registros 3 usuarios que se encuentra en un data.sql

En la tabla usuarios tenemos los siguientes datos:

Código	Correo	Contraseña	Rol
1	juan@example.com	Secret@123.\$	ADMIN
2	pedro@gmail.com	Se-ret@098.\$	ADMIN
3	elias.cornejo@coagra.cl	Eli@sC123.\$	USER

Roles

Los roles USER corresponden a usuarios à sólo puede gestionar lectura de los datos, es decir, sólo tiene Acceso a los Endpoint GET

Los roles ADMIN corresponden a usuario à pueden gestionar todo los endpoint, es decir, POST, PUT, PATCH, GET, DEL

Data Pre-Cargada

```

    INSERT INTO usuario (
        id, nombre, correo, contraseña, token, activo,
        creado, modificado, ultimo_login, role
    ) VALUES (
        '11111111-1111-1111-1111-111111111111',
        'Juan Pérez',
        'juan@example.com',
        'Secret@123.$',
        'token-abc-123',
        true,
        CURRENT_TIMESTAMP,
        CURRENT_TIMESTAMP,
        CURRENT_TIMESTAMP,
        'ADMIN'
    );
    INSERT INTO usuario (
        id, nombre, correo, contraseña, token, activo,
        creado, modificado, ultimo_login, role
    ) VALUES (
        '22222222-2222-2222-2222-222222222222',
        'Pedro Ramírez',
        'pedro@gmail.com',
        'Se-ret@098.$',
        null,
        null,
        null,
        null,
        null,
        null
    );

```

The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure for a Java application named 'EY'. The code editor contains a SQL script named 'data.sql' with the following content:

```
16    INSERT INTO usuario (
17        id, nombre, correo, contraseña, token, activo,
18        creado, modificado, ultimo_login, role
19    ) VALUES (
20        '22222222-2222-2222-2222-222222222222',
21        'Pedro Ramirez',
22        'pedro@mail.com',
23        'Se-ret@098.$',
24        'token-abc-123',
25        true,
26        CURRENT_TIMESTAMP,
27        CURRENT_TIMESTAMP,
28        CURRENT_TIMESTAMP,
29        'ADMIN'
30    );
31    INSERT INTO usuario (
32        id, nombre, correo, contraseña, token, activo,
33        creado, modificado, ultimo_login, role
34    ) VALUES (
35        '33333333-3333-3333-3333-333333333333',
36        'Elias Cornejo',
37        'elias.cornejo@coagra.cl',
38        'Eli@C123.$',
39        'token-abc-123',
40        true,
41        CURRENT_TIMESTAMP,
42        CURRENT_TIMESTAMP,
43        CURRENT_TIMESTAMP,
44        'USER'
45    );
```

The bottom status bar indicates the file has 38 lines, is in CRLF format, and is using UTF-8 encoding with 4 spaces.

This screenshot shows the same IntelliJ IDEA interface, but the code editor now displays a different part of the 'data.sql' script:

```
28    CURRENT_TIMESTAMP,
29    'ADMIN'
30    );
31    INSERT INTO usuario (
32        id, nombre, correo, contraseña, token, activo,
33        creado, modificado, ultimo_login, role
34    ) VALUES (
35        '33333333-3333-3333-3333-333333333333',
36        'Elias Cornejo',
37        'elias.cornejo@coagra.cl',
38        'Eli@C123.$',
39        'token-abc-123',
40        true,
41        CURRENT_TIMESTAMP,
42        CURRENT_TIMESTAMP,
43        CURRENT_TIMESTAMP,
44        'USER'
45    );
```

The bottom status bar indicates the file has 45 lines, is in CRLF format, and is using UTF-8 encoding with 4 spaces.

Token

El token dura aproximadamente 1 minuto, está utilizando postman y pasa el minuto deberá gestionar nuevamente el token para volver a utilizar los endpoint

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, Flows, History.
- Header:** Home, Workspaces, API Network, Search Postman, Ctrl + K, Invite, Bell icon, Upgrade.
- Current Collection:** Desafio-EY-ADMIN
- Request:** POST token
- URL:** http://localhost:8080/auth/login
- Method:** POST
- Body:** JSON (Raw) - { "correo": "juan@example.com", "contraseña": "Secret@123" }
- Response:** 200 OK, 8 ms, 740 B
- Logs:** All Logs, Clear, 401, 5 ms; 401, 7 ms; 200, 13 ms; 403, 7 ms; 200, 563 ms; 200, 14 ms.

SI pasa más del minuto expira la sesión y está obligado a solicitar nuevamente el token para acceder a los endpoint

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, Flows, History, and a pinned item "Prueba AWS Local".
- Header:** Home, Workspaces, API Network, Search Postman, Invite, Upgrade.
- Request URL:** http://localhost:8080/api/usuarios/1fdb30b0-bb10-4904-a8b7-8281db72d970/desactivar
- Method:** PATCH
- Authorization Tab:** Headers (8) - Bearer Token
- Body Tab:** JSON (highlighted with a red arrow)
- Response Body:** {"data": null, "mensaje": "Sesión expirada o no autorizado"}
- Status:** 401 Unauthorized

Role ADMIN

POST - Token

Genero token para acceder a crear usuarios

Código	Correo	Contraseña	Rol
--------	--------	------------	-----

1	juan@example.com	Secret@123.\$	ADMIN
2	pedro@gmail.com	Se-ret@098.\$	ADMIN
3	elias.cornejo@coagra.cl	Eli@sC123.\$	USER

Postman screenshot showing a successful POST request to `/token`. The body contains:

```

1 {
2   "correo": "juan@example.com",
3   "contraseña": "Secret@123.$"
4 }

```

The response status is `200 OK`.

POST - crearUsuario

Crear un nuevo usuario

Postman screenshot showing a successful POST request to `/crearUsuario`. The body contains:

```

1 {
2   "nombre": "Juan Rodriguez",
3   "correo": "pe22233281rdz67go.com",
4   "contraseña": "3112.rwqq",
5   "telefonos": [
6     {}
7   ]
8 }

```

The response status is `201 Created`.

GET - listarUsuarios

Recupera los usuarios registrados y saco el UUID

Postman screenshot showing a successful GET request to `/listarUsuarios`. The response body contains:

```

1 [
2   {
3     "token": "token-abc-123",
4     "activo": true,
5     "role": "USER",
6     "telefonos": []
7   },
8   {
9     "id": "c03b32ab-a0c7-4d18-9475-7604af4bc1aa",
10    "nombre": "Juan Rodriguez",
11    "correo": "pe22233281rdz67go.com",
12    "contraseña": "3112.rwqq",
13    "telefonos": [
14      {}
15    ],
16    "creado": "2025-04-10T19:41:37.544675",
17    "ultimoLogin": "2025-04-10T19:41:37.544675",
18    "token": "c03b32ab-a0c7-4d18-9475-7604af4bc1aa",
19    "ultimoLogin": "2025-04-10T19:41:37.544675",
20    "ultimoLogin": "2025-04-10T19:41:37.544675",
21    "ultimoLogin": "2025-04-10T19:41:37.544675"
22 }
23 ]

```

The response status is `200 OK`.

GET - obtenerUsuario

Buscar el usuario por el UUID

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/usuarios/c03b32ab-a0c7-45f8-9476-17d44f4bc1aa`. The response body is a JSON object representing a user:

```
1 {
  "data": {
    "id": "c03b32ab-a0c7-45f8-9476-17d44f4bc1aa",
    "nombre": "Juan Rodriguez",
    "correo": "juanrodriguez@correo.com",
    "contraseña": "3112.reeQ09",
    "creado": "2025-04-10T19:41:37.544675",
    "modificado": "2025-04-10T19:41:37.544675",
    "ultimoLogin": "2025-04-10T19:41:37.544675",
    "token": "ey30eXA1013Kv12Hy11nMgH3L013Jg1t03n9xN209NGY0YmXyWElLCJleHA10jE3ND0zMjg1NTcsImIhdCI6MTc0NDMyODQ5N30",
    "Sb_alkxd9M4en051Nzpodz0m6ZpNoK8gRhn0c5et0E",
    "activo": true,
    "rol": "USER",
    "telefonos": [
      {
        "numero": "1234567",
        "codigoCiudad": "1",
        "codigoPais": "57"
      }
    ]
  }
}
```

PUT - actualizarUsuario

Se actualiza la tarea por UUID

The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/usuarios/c03b32ab-a0c7-45f8-9476-17d44f4bc1aa`. The response body is a JSON object representing the updated user:

```
1 {
  "data": {
    "id": "c03b32ab-a0c7-45f8-9476-17d44f4bc1aa",
    "nombre": "Juan Rodriguez",
    "correo": "juanrodriguez.org",
    "contraseña": "3112.reeQ09",
    "telefonos": [
      {
        "numero": "1234567",
        "codigoCiudad": "1",
        "codigoPais": "57"
      }
    ],
    "numero": "1234567",
  }
}
```

Se revisa la modificación

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/usuarios`. The response body is a JSON array of users, including the one just updated:

```
1 [
  {
    "id": "c03b32ab-a0c7-45f8-9476-17d44f4bc1aa",
    "nombre": "Juan Rodriguez",
    "correo": "juanrodriguez.org",
    "contraseña": "3112.reeQ09",
    "creado": "2025-04-10T19:41:37.544675",
    "modificado": "2025-04-10T19:46:06.010456",
    "ultimoLogin": "2025-04-10T19:41:37.544675",
    "token": "ey30eXA1013Kv12Hy11nMgH3L013Jg1t03n9xN209NGY0YmXyWElLCJleHA10jE3ND0zMjg1NTcsImIhdCI6MTc0NDMyODQ5N30",
    "Sb_alkxd9M4en051Nzpodz0m6ZpNoK8gRhn0c5et0E",
    "activo": true,
    "rol": "USER",
    "telefonos": [
      {
        "numero": "1234567",
        "codigoCiudad": "1",
        "codigoPais": "57"
      }
    ]
  }
]
```

PATCH - activarUsuario

Se actualiza el usuario por UUID

PATCH - desactivarUsuario

Se actualiza el usuario por UUID

The screenshot shows the Postman interface with the following details:

- Project:** NTtData
- Collection:** NTTData-01
- Request:** PATCH /api/usuarios/{id}/desactivar
- Method:** PATCH
- URL:** http://localhost:8080/api/usuarios/11111111-1111-1111-1111-111111111111/desactivar
- Params:** Authorization (set to Bearer token)
- Body:** JSON (deactivatedUser)

```
1 {
2     "data": {
3         "id": "11111111-1111-1111-1111-111111111111",
4         "creado": "2025-04-10T18:45:10.081Z",
5         "modificado": "2025-04-10T18:54:19.556Z",
6         "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImlhbmNpZ24gZWxvbmUifQ.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImlhbmNpZ24gZWxvbmUifQ",
7         "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImlhbmNpZ24gZWxvbmUifQ.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6ImlhbmNpZ24gZWxvbmUifQ",
8         "activo": false
9     },
10     "mensaje": null
11 }
```

DELETE - eliminarUsuario

Eliminar el usuario por el UUID

The screenshot shows the Postman application interface. At the top, there are tabs for 'data.sql' and 'NttDataApplication.java'. Below the tabs, the 'Maven' tab is selected. The main workspace is titled 'My Workspace' and contains a collection named 'NttData-01'. This collection includes several API endpoints: 'POST token', 'POST crearUsuario', 'GET listarUsuarios', 'GET obtenerUsuario', 'PUT actualizarUsuario', 'PATCH activarUsuario', and 'PATCH desactivarUsuario'. A specific endpoint, 'DEL eliminarUsuario', is highlighted with a gray background. To the right of the collection list, there is a detailed view of the 'eliminarUsuario' endpoint. It shows a 'DELETE' method pointing to the URL 'http://localhost:8080/api/usuarios/22222222-2222-2222-2222-222222222222'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Cookies (1)', 'Headers (9)', 'Test Results', and 'Settings'. The 'Body' tab is selected and contains the value '1'. The 'Headers' tab shows '(9)' entries. The 'Test Results' tab indicates a '204 No Content' response with a duration of '46 ms' and a size of '282 B'. There are also buttons for 'Save', 'Send', and 'Cookies'.

Revisar que si está eliminado

The screenshot shows the Postman interface with a collection named 'NTTData-01'. A specific endpoint, 'obtenerUsuario', is selected. The request method is set to 'GET' and the URL is 'http://localhost:8080/api/usuarios/22222222-2222-2222-2222-222222222222'. The response status is '404 Not Found' with a message: 'data: null, mensaje: "Usuario no encontrado"'.

Rol ADMIN – CREAR UN USUARIO Y SACAR TOKEN

POST - Token

Genero token para acceder a crear usuarios

Código	Correo	Contraseña	Rol
1	juan@example.com	Secret@123.\$	ADMIN
2	pedro@gmail.com	Se-ret@098.\$	ADMIN
3	elias.cornejo@coagra.cl	Eli@sC123.\$	USER

The screenshot shows the Postman interface with a collection named 'NTTData-01'. An endpoint named 'token' is selected. The request method is 'POST' and the URL is 'http://localhost:8080/auth/login'. The response status is '200 OK' with a message: 'data: null'.

Creo el usuario

Postman API Network - NTTData-desafio-spring-boot-manual

My Workspace

NTTData-01

POST /crearUsuario

Body (JSON)

```
{
  "nombre": "Juan Rodriguez",
  "correo": "Pe222332810d267go.com",
  "contraseña": "3112.rewQq",
  "telefonos": [
    {
      "numero": "1234567",
      "codigoCiudad": "1",
      "codigoPais": "57"
    }
  ]
}
```

201 Created

Se inicia sesión con el nuevo usuario

Postman API Network - NTTData-desafio-spring-boot-manual

My Workspace

NTTData-01

POST /token

Body (JSON)

```
{
  "correo": "Pe222332810d267go.com",
  "contraseña": "3112.rewQq"
}
```

200 OK

Validar correo

No tiene formato correo

Postman API Network - Desafio-EY-ADMIN

My Workspace

Desafio-EY-ADMIN

POST /crearUsuario

Body (JSON)

```
{
  "nombre": "Juan Rodriguez",
  "correo": "Pe22210d267go.com",
  "contraseña": "3112.rewQq",
  "telefonos": [
    {
      "numero": "1234567",
      "codigoCiudad": "1",
      "codigoPais": "57"
    }
  ]
}
```

400 Bad Request

Validar contraseña

Iniciar sesión

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:8080/auth/login`. The response body is:

```
1 {  
2   "correo": "juan@example.com",  
3   "contraseña": "Secret123-$"  
4 }
```

The status bar indicates `200 OK`, `26 ms`, and `739 B`.

Contraseña no adapta. Se sugiere cambiar una más fuerte.

Una contraseña válida debe tener al menos:

- Una letra mayúscula (A-Z)
 - Un número (0-9)
 - Seis caracteres o más

The screenshot shows the Postman application interface. On the left sidebar, under 'My Workspace', there are sections for 'Collections', 'Environments', 'Flows', 'History', and a list of recent projects: 'Prueba AWS Local', 'Prueba AWS Web', 'Prueba Feign', 'REST API basics: CRUD, test & variable', and 'Test-01'. The main workspace is titled 'Desafío-EY-ADMIN / crearUsuario'. It shows a 'POST' request to 'http://localhost:8080/api/usuarios'. The 'Body' tab is selected, displaying the following JSON payload:

```
1 {  
2     "nombre": "Juan Rodriguez",  
3     "correo": "Pe222332@ioidz267go.com",  
4     "contraseña": "31Qq",  
5     "telefonos": [  
6         {  
7             "numero": "1234567",  
8             "codigoCiudad": "1",  
9             "codigoPais": "57"  
10        }  
11    ]  
12}
```

The 'Test Results' tab shows a '400 Bad Request' response with a message: "data: null, mensaje: 'Contraseña no válida'".

Role User

POST - Token

Genero token para acceder a crear tareas

Código	Correo	Contraseña	Rol
1	juan@example.com	Secret@123.\$	ADMIN
2	pedro@gmail.com	Se-ret@098.\$	ADMIN

3 elias.cornejo@coagra.cl Eli@sC123.\$ USER

Iniciar sesión

POST - crearUsuario

No tienen autorización

The screenshot shows the Postman application interface. On the left, the sidebar has sections for Collections, Environments, Flows, and History. The main workspace is titled "My Workspace" and contains a collection named "Desafio-EY-USER". Under this collection, there are several items: "POST token", "POST crearUsuario" (which is selected), "GET listarUsuarios", "GET obtenerUsuario", "PUT actualizarUsuario", "PATCH activarUsuario", "PATCH desactivarUsuario", and "DEL eliminarUsuario". Below these are three collapsed sections: "Prueba AWS Local", "Prueba AWS Web", and "Prueba Feign". Further down are sections for "REST API basics: CRUD, test & variable" and "Test-01".

The central area shows a POST request to "http://localhost:8080/api/usuarios". The "Body" tab is selected, showing the following JSON payload:

```
1 {  
2   "nombre": "Juan Rodriguez",  
3   "correo": "Pe@r1odr267go.com",  
4   "contraseña": "3112.rwQ",  
5   "telefonos": [  
6     {  
7       "numero": "1234567",  
8       "codigoCiudad": "1",  
9       "codigoPais": "57"  
10    }  
11  ]  
12 }
```

The response status is "403 Forbidden" with a message: "data: null, mensaje: 'Acceso denegado: Access is denied'".

GFT - listarUsuarios

Recupera los usuarios registrados

```

1 {
  "data": [
    {
      "id": "11111111-1111-1111-1111-111111111111",
      "nombre": "Juan Pérez",
      "correo": "juan@example.com",
      "contraseña": "Secret123$",
      "creado": "2025-04-10T19:46:55.012104",
      "modificado": "2025-04-10T19:46:36.233676",
      "ultimoLogin": "2025-04-10T19:46:55.012104",
      "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.",
      "activo": true,
      "role": "ADMIN",
      "telefonos": []
    },
    {
      "id": "33333333-3333-3333-3333-333333333333",
      "nombre": "Elias Cornejo",
      "correo": "elias.cornejo@coagra.cl",
      "mensaje": null
    }
  ]
}

```

GET - obtenerUsuario

Buscar el usuario por el UUID

```

1 {
  "data": [
    {
      "id": "11111111-1111-1111-1111-111111111111",
      "nombre": "Juan Pérez",
      "correo": "juan@example.com",
      "contraseña": "Secret123$",
      "creado": "2025-04-10T19:46:55.012104",
      "modificado": "2025-04-10T19:46:36.233676",
      "ultimoLogin": "2025-04-10T19:46:55.012104",
      "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.",
      "activo": true,
      "role": "ADMIN",
      "telefonos": []
    }
  ]
}

```

PUT - actualizarUsuario

No tienen autorización

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** PUT Desafio-EY-USER / actualizarUsuario
- Method:** PUT
- URL:** http://localhost:8080/api/usuarios/1fdb30b0-bb10-4904-a8b7-8281db72d970
- Body:** JSON (raw)
- Body Content:**

```
1 {
2   "nombre": "Juan Rodriguez",
3   "correo": "juan@rodriguez.org",
4   "contraseña": "3112.rewQq",
5   "telefonos": [
6     {
7       "numero": "1234567",
8       "codigoCiudad": "1",
9       "codigoPais": "57"
10    }
11 }
```
- Response Status:** 403 Forbidden
- Response Body:**

```
1 {
2   "data": null,
3   "mensaje": "Acceso denegado: Access is denied"
4 }
```

PATCH - activarUsuario

No tienen autorización

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** PATCH Desafio-EY-USER / activarUsuario
- Method:** PATCH
- URL:** http://localhost:8080/api/usuarios/1fdb30b0-bb10-4904-a8b7-8281db72d970/activar
- Body:** JSON (raw)
- Body Content:**

```
1 Ctrl+Alt+P for Postbot
```
- Response Status:** 403 Forbidden
- Response Body:**

```
1 {
2   "data": null,
3   "mensaje": "Acceso denegado: Access is denied"
4 }
```

PATCH - desactivarUsuario

No tienen autorización

The screenshot shows the Postman application interface. On the left, the sidebar displays 'My Workspace' with collections like 'Desafio-EY-ADMIN' and 'Desafio-EY-USER'. Under 'Desafio-EY-USER', several methods are listed: POST token, POST crearUsuario, GET listarUsuarios, GET obtenerUsuario, PUT actualizarUsuario, PATCH activarUsuario, PATCH desactivarUsuario, and DEL eliminarUsuario. The 'PATCH desactivarUsuario' method is currently selected. The main workspace shows a 'PATCH' request to `http://localhost:8080/api/usuarios/1fdb30b0-bb10-4904-a8b7-8281db72d970/desactivar`. The 'Authorization' tab is selected, showing 'Bearer Token' and a placeholder `{{(authorization)}}`. The response body shows a JSON object with `"data": null` and `"mensaje": "Acceso denegado: Access is denied"`. The status bar at the bottom indicates a 403 Forbidden error.

DELETE - eliminarUsuario

No tiene autorización

This screenshot shows the same Postman interface as the previous one, but with a different request. The 'DEL eliminarUsuario' method under 'Desafio-EY-USER' is selected. The main workspace shows a 'DELETE' request to `http://localhost:8080/api/usuarios/1fdb30b0-bb10-4904-a8b7-8281db72d970`. The 'Authorization' tab is selected, showing 'Bearer Token' and a placeholder `{{(authorization)}}`. The response body shows a JSON object with `"data": null` and `"mensaje": "Acceso denegado: Access is denied"`. The status bar at the bottom indicates a 403 Forbidden error.

