

Universidad ORT Uruguay

Facultad de Ingeniería

Evidencia de la ejecución de las pruebas de la API con Postman

URL Repositorio: <https://github.com/ORT-DA2/Obligatorio-DDA2>

Diego Billares – 232666

Emiliano Russo – 227209

Tutores:

Gonzalo Méndez

Ignacio Valle Dubé

Santiago Nahuel Mendez Varela

2021

Nosotros, Diego Billaes y Emiliano Russo, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos el obligatorio de diseño de aplicaciones 2;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Diego Billaes

Emiliano Russo

Abstract

Mediante imágenes se presentan las evidencias de pruebas realizadas en postman hacia la capa de servicio de la api restful, mencionando en ciertos puntos las consideraciones necesarias para entender cómo actúa la api rest en base a cierto conjunto de peticiones que se quieran realizar.

También se deja evidencia, de la alteración de la base de datos en base a cierta consulta realizada.

Cabe mencionar que se evidenciaron los casos bordes para las funcionalidades más importantes de la solución y por supuesto el caso correspondiente.

Palabras clave

Token: es una llave que se le da a un usuario en un sistema al ingresar para generar una sesión.

Get: Palabra (en inglés) muy utilizada en los lenguajes de programación considerada una convención para traer datos del sistema.

Index: método inicial al crear una clase controlador en .net Core.

Postman: herramienta que ayuda a probar, modificar y crear aplicaciones web.

ID: es la abreviación de identificador en inglés. Es una palabra reservada para marcar un identificador en tablas de bases de datos.

Null o Nula: es una expresión que sirve para asignar a una variable en lenguajes de programación denominado "sin valor".

Índice

| | |
|--------------------------------------|-----------|
| 1. Pruebas Postman Psicología | 7 |
| 1.1. Token | 7 |
| 1.2. Token Invalido Mensaje Error | 7 |
| 1.3. Registrar Psicólogo | 8 |
| 1.3.1. Datos correctos | 8 |
| 1.3.2. Alta Psicólogo Existente | 8 |
| 1.4. Eliminar Psicólogo | 8 |
| 1.4.1 Psicólogo Eliminado | 9 |
| 1.5. Modificar Psicólogo | 9 |
| 1.5.1. Modificacion Dirección Urbana | 10 |
| 1.6. Agendar Cita | 12 |
| 1.6.1. Datos Correctos | 12 |
| 2. Pruebas Postman Música | 13 |
| 2.1. Registrar Canción | 13 |
| 2.1.1. Datos Correctos | 13 |
| 2.2. Borrar Canción | 14 |
| Canción Borrada | 14 |
| 2.3. Get Canción | 15 |
| 2.3.1. Datos Correctos | 15 |
| 2.4. Index | 15 |
| 2.4.1. Único Curso | 15 |
| 2.5. GetPlayList | 16 |
| 2.5.1. ID existente | 16 |
| 2.5.2. ID Inexistente | 16 |

| | |
|--------------------------------------|-----------|
| 3. Referencias bibliográficas | 17 |
| 4. Anexo | 17 |
| 4.1. Registrar Psicólogo incorrecto | 18 |
| 4.1.1. Nombre Vacío | 18 |
| 4.1.2. Dirección Nula | 19 |
| 4.1.3. Omisión del Nombre | 20 |
| 4.1.3. Body Vacío | 21 |
| 4.2. Eliminar Psicólogo incorrecto | 22 |
| 4.2.1. ID no existente | 22 |
| 4.2.2. Body Vacío | 22 |
| 4.2.3. Null | 24 |
| 4.3. Modificar psicólogo incorrecto | 25 |
| 4.3.1. Nombre null | 25 |
| 4.3.2. Body Vacío | 26 |
| 4.3.3. Dirección Urbana Vacía | 27 |
| 4.3.4. ID no existente | 28 |
| 4.4. Agendar Cita incorrecta | 29 |
| 4.4.1. Nombre Null | 29 |
| 4.4.2. Body Vacío | 30 |
| 4.4.3. Omisión del Número de celular | 31 |
| 4.5. Registrar Canción incorrecta | 32 |
| 4.5.1. Título Null | 32 |
| 4.5.2. Body Vacío | 33 |
| 4.5.3. Omisión de descripción | 34 |
| 4.6. Borrar Canción incorrecta | 35 |
| 4.6.1. Body Vacío | 35 |
| 4.6.2. Título Null | 35 |

| | |
|-----------------------------|----|
| 4.6.3. Datos Inexistentes | 37 |
| 4.7. Get Canción incorrecta | 38 |
| 4.7.1. Omisión Autor | 38 |
| 4.8. GetPlayList Incorrecta | 39 |
| 4.8.1. ID Inexistente | 39 |

1. Pruebas Postman Psicología

1.1. Token

Params Authorization Headers (9) Body ● Pre-request Script Tests

Headers Hide auto-generated headers

| KEY | VALUE |
|---|-----------------------------------|
| <input checked="" type="checkbox"/> Postman-Token | <calculated when request is sent> |
| <input checked="" type="checkbox"/> Content-Type | application/json |
| <input checked="" type="checkbox"/> Content-Length | <calculated when request is sent> |
| <input checked="" type="checkbox"/> Host | <calculated when request is sent> |
| <input checked="" type="checkbox"/> User-Agent | PostmanRuntime/7.28.0 |
| <input checked="" type="checkbox"/> Accept | */* |
| <input checked="" type="checkbox"/> Accept-Encoding | gzip, deflate, br |
| <input checked="" type="checkbox"/> Connection | keep-alive |
| <input checked="" type="checkbox"/> token | 7WALU4FAPQ00 |

1.2. Token Invalido Mensaje Error

Params Authorization Headers (9) Body ● Pre-request Script Tests Set

Headers Hide auto-generated headers

| KEY | VALUE |
|---|-----------------------------------|
| <input checked="" type="checkbox"/> Postman-Token | <calculated when request is sent> |
| <input checked="" type="checkbox"/> Content-Type | application/json |
| <input checked="" type="checkbox"/> Content-Length | <calculated when request is sent> |
| <input checked="" type="checkbox"/> Host | <calculated when request is sent> |
| <input checked="" type="checkbox"/> User-Agent | PostmanRuntime/7.28.0 |
| <input checked="" type="checkbox"/> Accept | */* |
| <input checked="" type="checkbox"/> Accept-Encoding | gzip, deflate, br |
| <input checked="" type="checkbox"/> Connection | keep-alive |
| <input checked="" type="checkbox"/> token | 7WALU4FAPQ00 |

Body Cookies Headers (5) Test Results

511 Network Authentication Required 39 ms 257 B

Pretty Raw Preview Visualize JSON

```
1
2  "codigo": 511,
3  "mensaje": "No existe la sesion"
4
```


1.3. Registrar Psicólogo

1.3.1. Datos correctos

The screenshot shows a REST client interface with a POST request to `https://localhost:44306/Psicologia/RegistrarPsicologo`. The request body is a JSON object with the following fields: `Nombre` (Federico), `FormatoConsulta` (0), `DireccionUrbana` (Avenida X 123), and `DolenciasQueTrata` (an array of two objects, each with a `Dolencia` field). The response status is 200 OK, and the response body is a JSON object with `codigo` (200) and `mensaje` (Psicologo guardado).

```
POST https://localhost:44306/Psicologia/RegistrarPsicologo

{
  "Nombre": "Federico",
  "FormatoConsulta": 0,
  "DireccionUrbana": "Avenida X 123",
  "DolenciasQueTrata": [
    {
      "Dolencia": 1
    },
    {
      "Dolencia": 2
    }
  ]
}
```

```
{
  "codigo": 200,
  "mensaje": "Psicologo guardado"
}
```

1.3.2. Alta Psicólogo Existente

Nuestra solución no considera la duplicación de datos en Psicólogos. En conclusión si se permite registrar un Psicólogo con datos iguales a otro Psicólogo.

1.4. Eliminar Psicólogo

Evidencia de Psicólogo registrado en la base de datos

The screenshot shows a database table with the following columns: ID, Nombre, FormatoConsulta, and DireccionUrbana. The table contains 7 rows of data, with the last row highlighted in yellow.

| | ID | Nombre | FormatoConsulta | DireccionUrbana |
|---|----|----------|-----------------|--------------------|
| 1 | 1 | Silvia | 0 | Blvr 1234 |
| 2 | 3 | Marcos | 0 | Blvr 1111 |
| 3 | 4 | Violeta | 0 | Avenida Pintos 178 |
| 4 | 5 | Raul | 0 | Avenida Pintos 701 |
| 5 | 6 | Fernando | 1 | Avenida Pintos 101 |
| 6 | 7 | Federico | 0 | Avenida X 123 |

1.4.1 Psicólogo Eliminado

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

1 7

Body Cookies Headers (5) Test Results 200 OK 1691 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "codigo": 200,
3   "mensaje": "Psicologo eliminado"
4 }
```

Base de datos

| Results | | Messages | | |
|---------|----|----------|-----------------|--------------------|
| | ID | Nombre | FormatoConsulta | DireccionUrbana |
| 1 | 1 | Silvia | 0 | Blvr 1234 |
| 2 | 3 | Marcos | 0 | Blvr 1111 |
| 3 | 4 | Violeta | 0 | Avenida Pintos 178 |
| 4 | 5 | Raul | 0 | Avenida Pintos 701 |
| 5 | 6 | Fernando | 1 | Avenida Pintos 101 |

1.5. Modificar Psicólogo

Psicólogo Fernando en base de datos

| | ID | Nombre | FormatoConsulta | DireccionUrbana |
|---|----|----------|-----------------|--------------------|
| 1 | 1 | Silvia | 0 | Blvr 1234 |
| 2 | 3 | Marcos | 0 | Blvr 1111 |
| 3 | 4 | Violeta | 0 | Avenida Pintos 178 |
| 4 | 5 | Raul | 0 | Avenida Pintos 701 |
| 5 | 6 | Fernando | 1 | Avenida Pintos 101 |

Dolencias

| Results | | Messages | |
|---------|----|-------------|------------|
| | ID | IDPsicologo | IDDolencia |
| 1 | 11 | 1 | 0 |
| 2 | 12 | 1 | 4 |
| 3 | 13 | 3 | 2 |
| 4 | 14 | 3 | 3 |
| 5 | 15 | 3 | 1 |
| 6 | 16 | 4 | 5 |
| 7 | 17 | 4 | 6 |
| 8 | 18 | 5 | 1 |
| 9 | 19 | 5 | 6 |
| 10 | 20 | 6 | 1 |
| 11 | 21 | 6 | 2 |

1.5.1. Modificacion Dirección Urbana

Base de datos antes

| Results | | Messages | | |
|---------|----|----------|-----------------|--------------------|
| | ID | Nombre | FormatoConsulta | DireccionUrbana |
| 1 | 1 | Silvia | 0 | Blvr 1234 |
| 2 | 3 | Marcos | 0 | Blvr 1111 |
| 3 | 4 | Violeta | 0 | Avenida Pintos 178 |
| 4 | 5 | Raul | 0 | Avenida Pintos 701 |
| 5 | 6 | Fernando | 1 | Avenida Pintos 101 |

Postman

The screenshot shows the Postman interface for a PATCH request to `https://localhost:44306/Psicologia/ModificarPsicologo`. The request body is a JSON object with the following structure:

```
1 {
2   "ID": 6,
3   "Nombre": "Fernando",
4   "FormatoConsulta": 1,
5   "DireccionUrbana": "18 de Julio 172",
6   "DolenciasQueTrata": [
7     {
8       "Dolencia": 1
9     },
10    {
11      "Dolencia": 2
12    }
13  ]
14 }
```

The response is a 200 OK status with the following JSON body:

```
1 {
2   "codigo": 200,
3   "mensaje": "Psicologo modificado"
4 }
```

Base de datos después

| Results | | | | |
|---------|----|----------|-----------------|--------------------|
| | ID | Nombre | FormatoConsulta | DireccionUrbana |
| 1 | 1 | Silvia | 0 | Blvr 1234 |
| 2 | 3 | Marcos | 0 | Blvr 1111 |
| 3 | 4 | Violeta | 0 | Avenida Pintos 178 |
| 4 | 5 | Raul | 0 | Avenida Pintos 701 |
| 5 | 6 | Fernando | 1 | 18 de Julio 172 |

1.6. Agendar Cita

Fecha de nacimiento

POST ▼ https://localhost:44306/Psicologia/PedirCita

Params Authorization Headers (9) Body ● Pre-request Script ● Tests Settings

```
1 var fechaNacimiento = new Date(1990,12,20);
2 postman.setEnvironmentVariable("fechaNacimiento", fechaNacimiento.toISOString())
3 ;
```

1.6.1. Datos Correctos

POST ▼ https://localhost:44306/Psicologia/PedirCita

Params Authorization Headers (9) Body ● Pre-request Script ● Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼

```
1 {
2   "Nombre": "Anderson",
3   "Apellido": "Perrone",
4   "FechaNacimiento": "{{fechaNacimiento}}",
5   "Email": "anderson@gmail.com",
6   "NumeroCelular": "092111777",
7   "Dolencia": 6
8 }
```

Body Cookies Headers (5) Test Results 🚩 Status: 200 OK Time: 108 ms

Pretty Raw Preview Visualize JSON ▼ 🔗

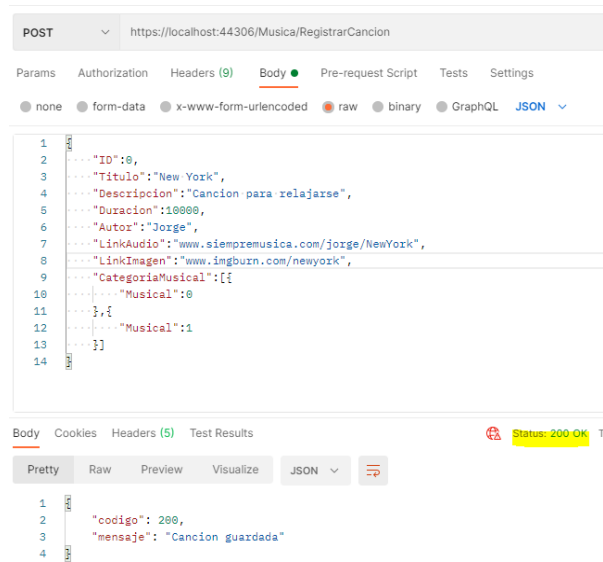
```
1 {
2   "codigo": 200,
3   "mensaje": {
4     "id": 5,
5     "idPsicologo": 4,
6     "nombrePsicologo": "Violeta",
7     "formatoConsulta": 0,
8     "direccionConsulta": "https://bettercalm.com.uy/4/CP75S",
9     "fechaConsulta": "2021-04-29T16:13:07.378517-03:00"
10  }
11 }
```

| Results Messages | | | | | | |
|------------------|----|-------------|-----------------|-----------------|-----------------------------------|-----------------------------|
| | ID | IdPsicologo | NombrePsicologo | FormatoConsulta | DireccionConsulta | FechaConsulta |
| 1 | 1 | 1 | Silvia | 0 | https://bettercalm.com.uy/1/9CAWK | 2021-04-22 21:38:29.4396206 |
| 2 | 2 | 3 | Marcos | 1 | Blvr 1111 | 2021-04-22 21:39:38.5640587 |
| 3 | 3 | 1 | Silvia | 0 | https://bettercalm.com.uy/1/XMNQV | 2021-04-23 19:36:14.5756894 |
| 4 | 4 | 3 | Marcos | 0 | https://bettercalm.com.uy/3/2PW4U | 2021-04-23 19:38:39.6738877 |
| 5 | 5 | 4 | Violeta | 0 | https://bettercalm.com.uy/4/CP75S | 2021-04-29 16:13:07.3785170 |

2. Pruebas Postman Música

2.1. Registrar Canción

2.1.1. Datos Correctos



Base de datos

| | ID | Titulo | Descripcion | Duracion | Autor | LinkAudio | LinkImagen |
|---|----|----------|-------------------------------------|----------|-------|-------------------------------------|------------------------|
| 1 | 1 | Oro | cancion relajante y emotiva | 100 | Jorge | www.siempremusica.com/jorge/Oro | |
| 2 | 2 | WorkHard | cancion motivacional para ejercitar | 100 | Hugo | www.siempremusica.com/Hugo/WorkHard | |
| 3 | 3 | Dash | cancion motivacional para correr | 100 | Rodri | www.siempremusica.com/Rodri/Dash | |
| 4 | 4 | Hola | Cancion para relajarse | 8000 | Jorge | www.siempremusica.com/jorge/Hola | |
| 5 | 5 | New York | Cancion para relajarse | 10000 | Jorge | www.siempremusica.com/jorge/NewYork | www.imgbum.com/newyork |

2.2. Borrar Canción

Canción Borrada

The screenshot displays a REST client interface with the following components:

- Header:** "Coleccion de pruebas 2021 / Musical / BorrarCancion" and a "Save" button.
- Method and URL:** A dropdown menu set to "DELETE" and the URL "https://localhost:44306/Musica/BorrarCancion".
- Tabs:** "Params", "Authorization", "Headers (9)", "Body", "Pre-request Script", "Tests", and "Settings". The "Body" tab is selected.
- Body Type:** A row of radio buttons for "none", "form-data", "x-www-form-urlencoded", "raw", "binary", and "GraphQL". The "JSON" option is selected, indicated by a blue checkmark.
- Request Body:** A code editor showing a JSON object:

```
1 {  
2   "Autor": "Jorge",  
3   "Titulo": "New York"  
4 }
```
- Response Section:** Tabs for "Body", "Cookies", "Headers (5)", and "Test Results". The "Body" tab is selected.
 - Status Bar:** Shows a globe icon, "Status: 200 OK", "Time: 83 ms", and "Size: 216 B".
 - Response Body:** A code editor showing the JSON response:

```
1 {  
2   "codigo": 200,  
3   "mensaje": "Cancion borrada"  
4 }
```

2.3. Get Canción

2.3.1. Datos Correctos

GET

▼

https://localhost:44306/Musica/GetCancion?titulo=Hola&autor=Jorge

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

| | KEY | VALUE | DESCRIPTION |
|-------------------------------------|--------|-------|-------------|
| <input checked="" type="checkbox"/> | titulo | Hola | |
| <input checked="" type="checkbox"/> | autor | Jorge | |
| | Key | Value | Description |

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 5.21 s


Pretty

Raw

Preview

Visualize

JSON



```
1  {
2    "codigo": 200,
3    "mensaje": {
4      "id": 4,
5      "titulo": "Hola",
6      "descripcion": "Cancion para relajarse",
7      "duracion": "02h:13m",
8      "autor": "Jorge",
9      "linkAudio": "www.siempremusica.com/jorge/Hola",
10     "linkImagen": "",
11     "categoriasMusicales": [
12       "Dormir",
13       "Meditar"
14     ]
15   }
16 }
```


2.5. GetPlayList

2.5.1. ID existente

GET ▼ https://localhost:44306/Musica/GetPlayList?id=1

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

| | KEY | VALUE | DESCRIPTION |
|-------------------------------------|-----|-------|-------------|
| <input checked="" type="checkbox"/> | id | 1 | |
| | Key | Value | Description |

body Cookies Headers (5) Test Results 🚫 Status: 200 OK Time: 83 ms Size: 551 B

Pretty Raw Preview Visualize JSON ▼ ≡

```
1  {
2    "codigo": 200,
3    "mensaje": {
4      "id": 1,
5      "nombre": "Playlist para Dormir",
6      "descripcion": "Solo canciones relajantes",
7      "canciones": [
8        {
9          "id": 1,
10         "titulo": "Oro",
11         "descripcion": "cancion relajante y emotiva",
12         "duracion": "01m:40s",
13         "autor": "Jorge",
14         "linkAudio": "www.siempremusica.com/jorge/Oro",
15         "linkImagen": "",
16         "categoriasMusicales": [
17           "Dormir",
18           "Meditar",
19           "Musica"
20         ]
21       }
22     ],
23     "listaCategorias": [
24       "Dormir"
```

2.5.2. ID Inexistente

GET ▼ https://localhost:44306/Musica/GetPlayList?id=777

Params ● Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

| | KEY | VALUE | DESCRIPTION |
|-------------------------------------|-----|-------|-------------|
| <input checked="" type="checkbox"/> | id | 777 | |
| | Key | Value | Description |

body Cookies Headers (5) Test Results 🚫 Status: 400 Bad Request Time: 53 ms Size: 239 B

Pretty Raw Preview Visualize JSON ▼ ≡

```
1  {
2    "codigo": 400,
3    "mensaje": "No existe la Playlist"
4  }
```

3. Referencias bibliográficas

[1] B. Mulloy. Web API Design: Crafting Interfaces that Developers Love. Apigee ed.

4. Anexo

4.1. Registrar Psicólogo incorrecto

4.1.1. Nombre Vacío

The screenshot displays a REST client interface with a POST request to `https://localhost:44306/Psicologia/RegistrarPsic`. The request body is a JSON object with the following fields:

```
1 {
2   "Nombre": "",
3   "FormatoConsulta": 0,
4   "DireccionUrbana": "Avenida X 123",
5   "DolenciasQueTrata": [
6     {
7       "Dolencia": 1
8     },
9     {
10      "Dolencia": 2
11    }
12  ]
13 }
```

The response shows a `400 Bad Request` status with a response time of 100 ms and a size of 255 B. The response body is a JSON object:

```
1 {
2   "codigo": 400,
3   "mensaje": "No pueden haber psicologos sin nombre"
4 }
```

4.1.2. Dirección Nula

The screenshot shows a REST client interface with a POST request to `https://localhost:44306/Psicologia/RegistrarPsicologo`. The request body is a JSON object with the following fields: `Nombre` (Federico), `FormatoConsulta` (0), `DireccionUrbana` (null), and `DolenciasQueTrata` (an array of two objects with `Dolencia` values 1 and 2). The `DireccionUrbana` field is highlighted in yellow. The response status is `400 Bad Request`, also highlighted in yellow. The response body is a JSON object with `codigo` (400) and `mensaje` ("Un psicologo no puede tener una direccion nula").

```
POST https://localhost:44306/Psicologia/RegistrarPsicologo


Params Authorization Headers (9) Body ● Pre-request Script Tests Settings
● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL JSON ▼


1 2
2  ... "Nombre": "Federico",
3  ... "FormatoConsulta": 0,
4  ... "DireccionUrbana": null,
5  ... "DolenciasQueTrata": [
6    {
7      "Dolencia": 1
8    },
9    {
10     "Dolencia": 2
11   }
12 ]
13








Body Cookies Headers (5) Test Results 400 Bad Request
Pretty Raw Preview Visualize JSON ▼

1 2
2  "codigo": 400,
3  "mensaje": "Un psicologo no puede tener una direccion nula"
4  2
```


4.1.3. Omisión del Nombre



POST  https://localhost:44306/Psicologia/RegistrarPsicologo

Params Authorization Headers (9) **Body**  Pre-request Script Tests Settings

 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** 

```
1  {
2    "FormatoConsulta":0,
3    "DireccionUrbana":"Avenida X 123",
4    "DolenciasQueTrata":[
5      {
6        "Dolencia":1
7      },
8      {
9        "Dolencia":2
10     }
11   ]
12 }
```

Body Cookies Headers (5) Test Results  400 Bad Request

Pretty Raw Preview Visualize **JSON**  

```
1  {
2    "codigo": 400,
3    "mensaje": "Un psicologo no puede tener un nombre nulo"
4  }
```

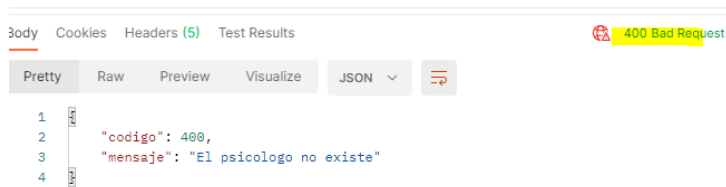
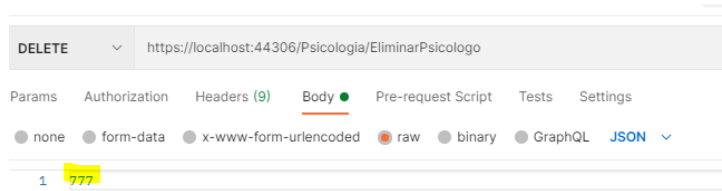
4.1.3. Body Vacío

The screenshot shows a REST client interface. At the top, a POST request is configured to `https://localhost:44306/Psicologia/RegistrarPsicologo`. The 'Body' tab is selected, and the content type is set to 'JSON'. The body is empty. Below the request area, the response status is '400 Bad Request' with a response time of '57 ms' and a size of '260 B'. The 'Body' tab of the response is selected, showing the following JSON:

```
1 {  
2   "codigo": 400,  
3   "mensaje": "Un psicologo no puede tener un nombre nulo"  
4 }
```

4.2. Eliminar Psicólogo incorrecto

4.2.1. ID no existente



4.2.2. Body Vacío

The screenshot displays a REST client interface with a DELETE request to `https://localhost:44306/Psicologia/EliminarPsicologo`. The 'Body' tab is selected, showing an empty body. The response is a 400 Bad Request, with a status bar indicating a duration of 4.36 s and a size of 240 B. The response body is shown in JSON format, indicating an error.

Request:

- Method: DELETE
- URL: `https://localhost:44306/Psicologia/EliminarPsicologo`
- Body: Empty

Response:

- Status: 400 Bad Request
- Duration: 4.36 s
- Size: 240 B
- Body (JSON):

```
1 {  
2   "codigo": 400,  
3   "mensaje": "El psicologo no existe"  
4 }
```


4.2.3. Null

The screenshot displays a REST client interface. At the top, a request bar shows the method **DELETE** and the URL `https://localhost:44306/Psicologia/EliminarPsicologo`. Below this, a tab bar includes **Params**, **Authorization**, **Headers (9)**, **Body** (selected), **Pre-request Script**, **Tests**, and **Settings**. The **Body** tab shows a single line of text: `1 null`. Below the request editor, a status bar indicates a **400 Bad Request** response, with a duration of **3.97 s** and a size of **240 B**. The **Body** tab is active, showing the response in **JSON** format. The response body is displayed in a code editor with line numbers 1 through 4, containing the following JSON object:

```
1 {
2   "codigo": 400,
3   "mensaje": "El psicologo no existe"
4 }
```

4.3. Modificar psicólogo incorrecto

4.3.1. Nombre null

The screenshot shows a REST client interface with a PATCH request to `https://localhost:44306/Psicologia/ModificarPsicologo`. The request body is a JSON object with the following structure:

```
1 {
2   ... "ID": 6,
3   ... "Nombre": null,
4   ... "FormatoConsulta": 1,
5   ... "DireccionUrbana": "Avenida X-777",
6   ... "DolenciasQueTrata": [
7     {
8       ... "Dolencia": 1
9     },
10    {
11      ... "Dolencia": 2
12    }
13  ]
14 }
```

The response is a 400 Bad Request with the following JSON body:

```
1 {
2   "codigo": 400,
3   "mensaje": "Un psicologo no puede tener un nombre nulo"
4 }
```

4.3.2. Body Vacío

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

- Method:** PATCH
- URL:** `https://localhost:44306/Psicologia/ModificarPsicologo`
- Body:** Empty (indicated by a green dot and the word "Body" in the tabs).
- Response:** 400 Bad Request (highlighted in yellow). Status: 1647 ms, 260 B.
- Response Body (JSON):**

```
1 {  
2   "codigo": 400,  
3   "mensaje": "Un psicologo no puede tener un nombre nulo"  
4 }
```

4.3.3. Dirección Urbana Vacía

The screenshot shows a REST client interface with a PATCH request to `https://localhost:44306/Psicologia/ModificarPsicologo`. The request body is a JSON object with the following structure:

```
1 {
2   "ID": 3,
3   "Nombre": "Marcos",
4   "FormatoConsulta": 0,
5   "DireccionUrbana": "",
6   "DolenciasQueTrata": [
7     {
8       "Dolencia": 2
9     },
10    {
11      "Dolencia": 3
12    }, {
13      "Dolencia": 1
14    }
15  ]
16 }
```

The response is a 400 Bad Request with the following JSON body:

```
1 {
2   "codigo": 400,
3   "mensaje": "No pueden haber psicologos sin direccion"
4 }
```

The status bar indicates a 400 Bad Request, 112 ms, 258 B, and a 'Save Response' button.

4.3.4. ID no existente

PATCH ▼ https://localhost:44306/Psicologia/ModificarPsicologo

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼

```
1  {
2    "ID": 888,
3    "Nombre": "Gerardo",
4    "FormatoConsulta": 0,
5    "DireccionUrbana": "Blvr 1111",
6    "DolenciasQueTrata": [
7      {
8        "Dolencia": 2
9      },
10     {
11       "Dolencia": 3
12     },
13     {
14       "Dolencia": 1
15     }
16   ]
17 }
```

Body Cookies Headers (5) Test Results 🌐 400 Bad Request 74 ms 232 B

Pretty Raw Preview Visualize **JSON** ▼

```
1  {
2    "codigo": 400,
3    "mensaje": "El psicologo no existe"
4  }
```

4.4. Agendar Cita incorrecta

4.4.1. Nombre Null

POST ▼ https://localhost:44306/Psicologia/PedirCita

Params Authorization Headers (9) **Body** ● Pre-request Script ● Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   ... "Nombre": null,
3   ... "Apellido": "Rodriguez",
4   ... "FechaNacimiento": "{{fechaNacimiento}}",
5   ... "Email": "juan@gmail.com",
6   ... "NumeroCelular": "092741369",
7   ... "Dolencia": 1
8 }
```

Body Cookies Headers (5) Test Results ⚠ Status: 400 Bad Request Time: 100ms

Pretty Raw Preview Visualize **JSON** ▼ ⌵

```
1 {
2   "codigo": 400,
3   "mensaje": "Datos Paciente Invalidos"
4 }
```

4.4.2. Body Vacío

The screenshot displays a REST client interface for a POST request to `https://localhost:44306/Psicologia/PedirCita`. The 'Body' tab is selected, showing an empty body. The 'JSON' format is chosen. The response status is '400 Bad Request' with a time of 51 ms. The response body is shown in the 'Body' tab, displaying the error message in JSON format.

Request:

- Method: POST
- URL: `https://localhost:44306/Psicologia/PedirCita`
- Body: Empty

Response:

- Status: 400 Bad Request
- Time: 51 m
- Body (JSON):

```
1 {  
2   "codigo": 400,  
3   "mensaje": "Datos Paciente Invalidos"  
4 }
```

4.4.3. Omisión del Número de celular

POST ▼ https://localhost:44306/Psicologia/PedirCita

Params Authorization Headers (9) **Body** ● Pre-request Script ● Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1 {
2   ... "Nombre": "Juan",
3   ... "Apellido": "Rodriguez",
4   ... "FechaNacimiento": "{{fechaNacimiento}}",
5   ... "Email": "juan@gmail.com",
6   ... "Dolencia": 1
7 }
```

Body Cookies Headers (5) Test Results Status: **400 Bad Request** Time: 43 ms Size: 242 B

Pretty Raw Preview Visualize **JSON** ▼

```
1 {
2   "codigo": 400,
3   "mensaje": "Datos Paciente Invalidos"
4 }
```


4.5. Registrar Canción incorrecta

4.5.1. Título Null

POST ▼ https://localhost:44306/Musica/RegistrarCancion

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1  {
2    "ID":0,
3    "Titulo":null,
4    "Descripcion":"Cancion para relajarse",
5    "Duracion":100,
6    "Autor":"Jorge",
7    "LinkAudio":"www.siempremusica.com/jorge/Hola",
8    "LinkImagen":"",
9    "CategoriaMusical":[{"
10     "Musical":0
11   },{
12     "Musical":1
13   }]
14 }
```

Body Cookies Headers (5) Test Results 🚫 Status: 400 Bad Request Time: 87 ms Size: 244 B

Pretty Raw Preview Visualize **JSON** ▼ ≡

```
1  {
2    "codigo": 400,
3    "mensaje": "Datos de cancion invalidos"
4  }
```

4.5.2. Body Vacío

The screenshot shows a REST client interface. At the top, a POST request is configured to `https://localhost:44306/Musica/RegistrarCancion`. The 'Body' tab is selected, and the body type is set to 'JSON'. The body content is empty. Below the request configuration, the response is displayed. The status is '400 Bad Request' (highlighted in yellow), with a time of 56 ms and a size of 236 B. The response body is shown in JSON format, indicating an invalid request.

Request:

- Method: POST
- URL: `https://localhost:44306/Musica/RegistrarCancion`
- Body Type: JSON
- Body Content: (Empty)

Response:

- Status: 400 Bad Request
- Time: 56 ms
- Size: 236 B
- Body (JSON):

```
{  "codigo": 400,  "mensaje": "Datos de cancion invalidos"}
```

4.5.3. Omisión de descripción

POST ▼ https://localhost:44306/Musica/RegistrarCancion

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● **raw** ● binary ● GraphQL **JSON** ▼

```
1  {
2    "ID":0,
3    "Titulo":"Hola",
4    "Duracion":100,
5    "Autor":"Jorge",
6    "LinkAudio":"www.siempremusica.com/jorge/Hola",
7    "LinkImagen":"",
8    "CategoriaMusical":[{"
9      "Musical":0
10   },{
11     "Musical":1
12   }]
13 }
```

Body Cookies Headers (5) Test Results Status: **400 Bad Request** Time: 53 ms Size: 236 B

Pretty Raw Preview Visualize **JSON** ▼

```
1  {
2    "codigo": 400,
3    "mensaje": "Datos de cancion invalidos"
4  }
```

4.6. Borrar Canción incorrecta

4.6.1. Body Vacío

The screenshot displays a REST client interface. At the top, a **DELETE** request is configured for the URL `https://localhost:44306/Musica/BorrarCancion`. The **Body** tab is selected, showing an empty body. Below the request configuration, the **Response** tab is active, showing a **400 Bad Request** status. The response body is displayed in JSON format, indicating an error with the message "Nombre Autor Incorrecto".

Request Configuration:

- Method: DELETE
- URL: `https://localhost:44306/Musica/BorrarCancion`
- Body: Empty

Response Details:

- Status: 400 Bad Request
- Time: 64 ms
- Size: 241 B

Response Body (JSON):

```
1 {
2   "codigo": 400,
3   "mensaje": "Nombre Autor Incorrecto"
4 }
```

4.6.2. Titulo Null

The screenshot shows a REST client interface with a DELETE request to `https://localhost:44306/Musica/BorrarCancion`. The request body is a JSON object with `"Autor": "Jorge"` and `"Titulo": null`. The response is a 400 Bad Request with the message `"Nombre Autor Incorrecto"`.

Request:

```
DELETE https://localhost:44306/Musica/BorrarCancion
```

Body:

```
1 {
2   "Autor": "Jorge",
3   "Titulo": null
4 }
```

Response:

```
1 {
2   "codigo": 400,
3   "mensaje": "Nombre Autor Incorrecto"
4 }
```

Status: 400 Bad Request **Time:** 42 ms

4.6.3. Datos Inexistentes

The screenshot shows a REST client interface. At the top, the method is set to **DELETE** and the URL is `https://localhost:44306/Musica/BorrarCancion`. Below the URL bar, there are tabs for **Params**, **Authorization**, **Headers (9)**, **Body** (which is selected), **Pre-request Script**, **Tests**, and **Settings**. Under the **Body** tab, there are radio buttons for **none**, **form-data**, **x-www-form-urlencoded**, **raw** (which is selected), **binary**, and **GraphQL**. To the right of these is a dropdown menu showing **JSON**. The main area displays the raw JSON body of the request:

```
1 {
2   ... "Autor": "asdasd",
3   ... "Titulo": "asdasdasdasd"
4 }
```

Below the request body, there are tabs for **Body** (selected), **Cookies**, **Headers (5)**, and **Test Results**. To the right of these tabs, there is a status icon and the text **Status: 400 Bad Request**. Below the tabs, there are buttons for **Pretty**, **Raw**, **Preview**, **Visualize**, and a dropdown menu showing **JSON**. The main area displays the raw JSON response:

```
1 {
2   "codigo": 400,
3   "mensaje": "No existe la cancion"
4 }
```

4.7. Get Canción incorrecta

4.7.1. Omisión Autor

GET

https://localhost:44306/Musica/GetCancion?titulo=Hola

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

| | KEY | VALUE | DESC |
|-------------------------------------|--------|-------|------|
| <input checked="" type="checkbox"/> | titulo | Hola | |
| | Key | Value | Desc |

Body

Cookies

Headers (5)

Test Results

Status: 400 Bad Request

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "codigo": 400,
3   "mensaje": "Nombre Autor Incorrecto"
4 }
```

4.8. GetPlayList Incorrecta

4.8.1. ID Inexistente

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

- Method:** GET
- URL:** `https://localhost:44306/Musica/GetPlayList?id=777`
- Params:** Authorization, Headers (6), Body, Pre-request Script, Tests, Settings
- Query Params:**

| | KEY | VALUE | DESCRIPTION |
|-------------------------------------|-----|-------|-------------|
| <input checked="" type="checkbox"/> | id | 777 | |
| | Key | Value | Description |
- Body:** Cookies, Headers (5), Test Results
- Status:** 400 Bad Request (highlighted in yellow), Time: 53 ms, Size: 239 B
- Response Body (JSON):**

```
1 {
2   "codigo": 400,
3   "mensaje": "No existe la PlayList"
4 }
```