PRÁCTICA 3: Servicios RESTful con Python

1. Diseño del servicio Restful

Para el diseño del servicio, se han definido los siguientes recursos, con el objetivo de poder realizar las operaciones requeridas:

- Obtener una lista de todas las piezas de la colección

URI	http://127.0.0.1:5000/api/pieces	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y piezas
	415	Unsupported Media Type

- Añadir una pieza nueva

URI	http://127.0.0.1:5000/api/pieces	
Método	POST	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	400	Bad Request
	415	Unsupported Media Type

- Editar una pieza de la colección

URI	http://127.0.0.1:5000/api/pieces/<id>	
Método	PUT	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	404	Not Found
	415	Unsupported Media Type

- Eliminar una pieza de la colección

URI	http://127.0.0.1:5000/api/pieces/[pieceId]	
Método	DELETE	
Devuelve	204	No Content
	404	Not Found

- Obtener una lista de todas las productoras

URI	http://127.0.0.1:5000/api/studios	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y productoras
	415	Unsupported Media Type

- Añadir una productora nueva

URI	http://127.0.0.1:5000/api/studios	
Método	POST	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	400	Bad Request
	415	Unsupported Media Type

- Editar una productora

URI	http://127.0.0.1:5000/api/studios/<id>	
Método	PUT	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	404	Not Found
	415	Unsupported Media Type

- Eliminar una productora solo si no hay ninguna pieza asociada

URI	http://127.0.0.1:5000/api/studios/<id>	
Método	DELETE	
Devuelve	204	No Content
	400	Bad Request
	404	Not Found

- Publicar una nueva evaluación

URI	http://127.0.0.1:5000/api/evaluations	
Método	POST	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	400	Bad Request
	415	Unsupported Media Type

- Editar una evaluación

URI	http://127.0.0.1:5000/api/evaluations/<id>	
Método	PUT	
Cuerpo de la petición	application/xml, application/json	
Devuelve	202	Accepted
	404	Not Found
	415	Unsupported Media Type

- Eliminar una evaluación

URI	http://127.0.0.1:5000/api/evaluations/<id>	
Método	DELETE	
Devuelve	204	No Content
	404	Not Found

- Obtener una lista de todas las evaluaciones de una pieza y filtrar esa lista por fecha o limitar la cantidad de información obtenida (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

URI	http://127.0.0.1:5000/api/pieces/<id_piece>/evaluations?date=fecha&start=a&end=b	
Método	GET	
Cadena de Consulta	date=	Fecha de la que se desean obtener las evaluaciones (por defecto 2022-12-13)
	start=	Posición de inicio de las evaluaciones deseadas (por defecto 0)
	end=	Posición final de las evaluaciones deseadas (por defecto 100)
Devuelve	202	Accepted y evaluaciones
	404	Not Found
	415	Unsupported Media Type

- Obtener el número de piezas dado una productora

URI	http://127.0.0.1:5000/api/studios/<id_studio>/pieces	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y lista de piezas
	401	Unauthorized
	404	Not Found

- Obtener la lista de evaluaciones que contienen un determinado texto

URI	http://127.0.0.1:5000/api/evaluations?pattern=text	
Método	GET	
Cadena de Consulta	?pattern=	Texto que deseamos buscar en las evaluaciones
Devuelve	202	Accepted y lista de evaluaciones
	401	Unauthorized
	404	Not Found

Además, para que las peticiones funcionasen de forma correcta, ha sido necesario implementar las siguientes funciones auxiliares:

- Obtener una pieza por id

URI	http://127.0.0.1:5000/api/pieces/<id>	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y lista de productoras
	401	Unauthorized
	404	Not Found

- Obtener una productora por id

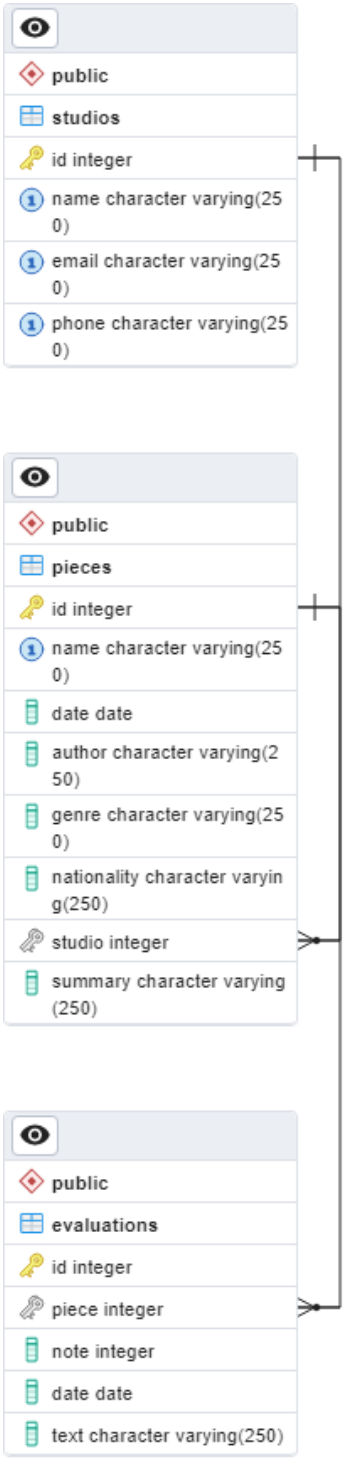
URI	http://127.0.0.1:5000/api/studios/<id>	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y productora
	401	Unauthorized
	404	Not Found

- Obtener una evaluación por id

URI	http://127.0.0.1:5000/api/evaluations/<id>	
Método	GET	
Cadena de Consulta		
Devuelve	202	Accepted y lista de productoras
	401	Unauthorized
	404	Not Found

Para asegurar la persistencia de los datos en el servicio, se ha diseñado una base de datos en PostgreSQL, denominada upmvideo, cuyo usuario es *"postgres"* y su contraseña es *"flaskroot"*.

Esta base de datos se compone de tres tablas (pieces, studios y evaluations), cuyas columnas son los atributos de los modelos definidos en el API y que se relacionan de la siguiente forma:



Las filas de todas las tablas se encuentran vacías en un primer momento, ya que los datos de prueba se insertan directamente desde el archivo initAlchemy.py. Para relacionar la base de datos con el código escrito en Python se ha usado la herramienta SQLAlchemy, tomando como ejemplo el código aportado por el profesor.

El código SQL necesario para la creación de la base de datos es el siguiente:

```
-- This script was generated by a beta version of the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/new if you find any bugs, including reproduction steps.
BEGIN;


CREATE TABLE IF NOT EXISTS public.evaluations
(
    id integer NOT NULL DEFAULT nextval('evaluations_id_seq'::regclass),
    piece integer,
    note integer NOT NULL,
    date date NOT NULL,
    text character varying(250) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT evaluations_pkey PRIMARY KEY (id)
);


CREATE TABLE IF NOT EXISTS public.pieces
(
    id integer NOT NULL DEFAULT nextval('pieces_id_seq'::regclass),
    name character varying(250) COLLATE pg_catalog."default" NOT NULL,
    date date NOT NULL,
    author character varying(250) COLLATE pg_catalog."default" NOT NULL,
    genre character varying(250) COLLATE pg_catalog."default" NOT NULL,
    nationality character varying(250) COLLATE pg_catalog."default" NOT NULL,
    studio integer,
    summary character varying(250) COLLATE pg_catalog."default",
    CONSTRAINT pieces_pkey PRIMARY KEY (id),
    CONSTRAINT pieces_name_key UNIQUE (name)
);


CREATE TABLE IF NOT EXISTS public.studios
(
```

```
    id integer NOT NULL DEFAULT nextval('studios_id_seq'::regclass),
    name character varying(250) COLLATE pg_catalog."default" NOT NULL,
    email character varying(250) COLLATE pg_catalog."default" NOT NULL,
    phone character varying(250) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT studios_pkey PRIMARY KEY (id),
    CONSTRAINT studios_email_key UNIQUE (email),
    CONSTRAINT studios_name_key UNIQUE (name),
    CONSTRAINT studios_phone_key UNIQUE (phone)
);

ALTER TABLE IF EXISTS public.evaluations
    ADD CONSTRAINT evaluations_piece_fkey FOREIGN KEY (piece)
    REFERENCES public.pieces (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;

ALTER TABLE IF EXISTS public.pieces
    ADD CONSTRAINT pieces_studio_fkey FOREIGN KEY (studio)
    REFERENCES public.studios (id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION;

END;
```

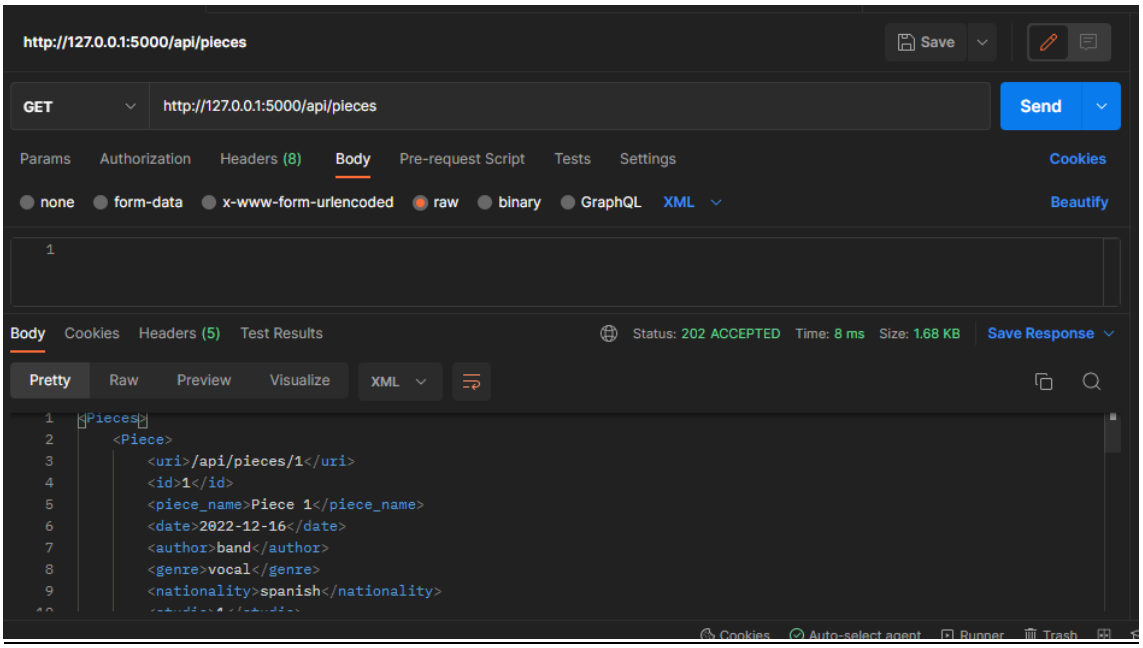
El servicio empleado para la base de datos ha sido PostgreSQL (versión 15) y el diseño se ha llevado acabo a través de pgAdmin 4.

2. Capturas de la ejecución de las operaciones anteriormente referidas desde un REST Client (Postman)

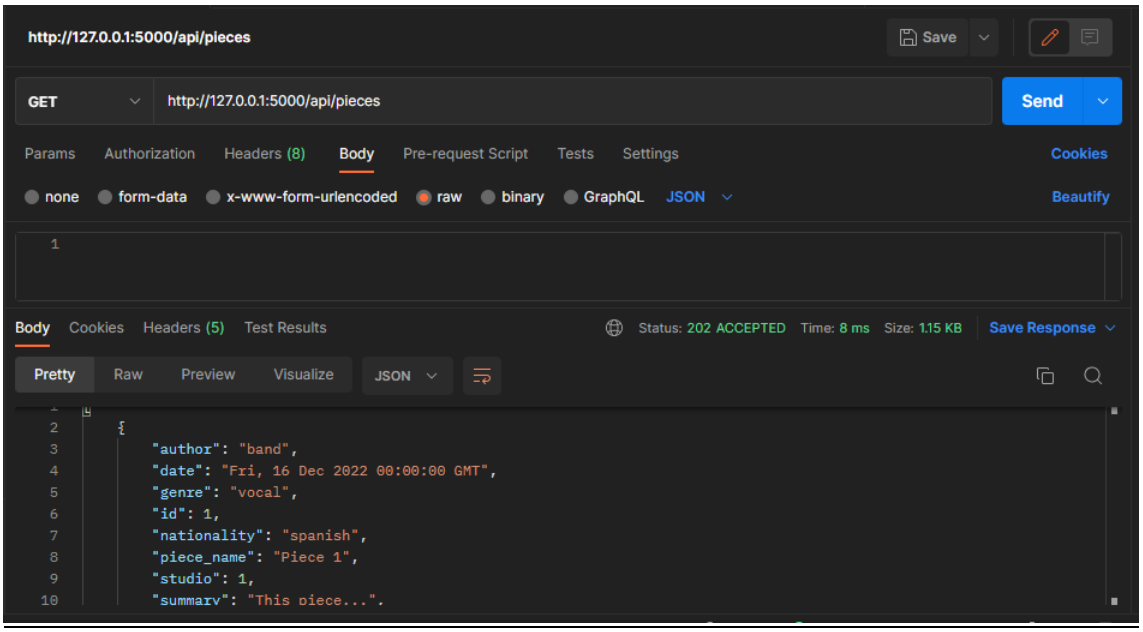
Se han introducido unos datos de prueba en la base de datos para poder llevar acabo las operaciones propuestas, operando los datos tanto en formato XML como JSON.

• Obtener una lista de todas las piezas de la colección

XML

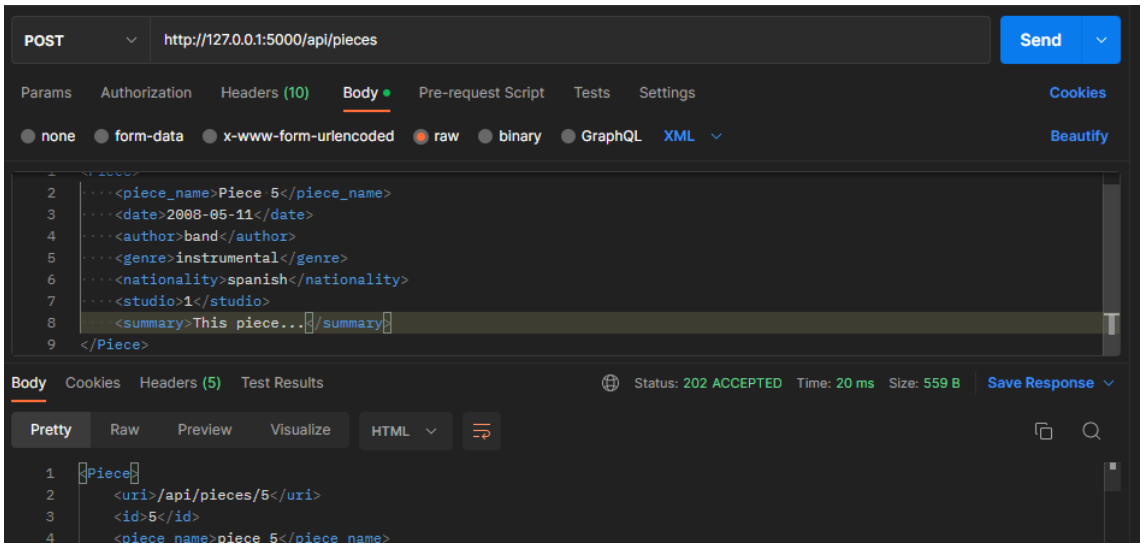


JSON

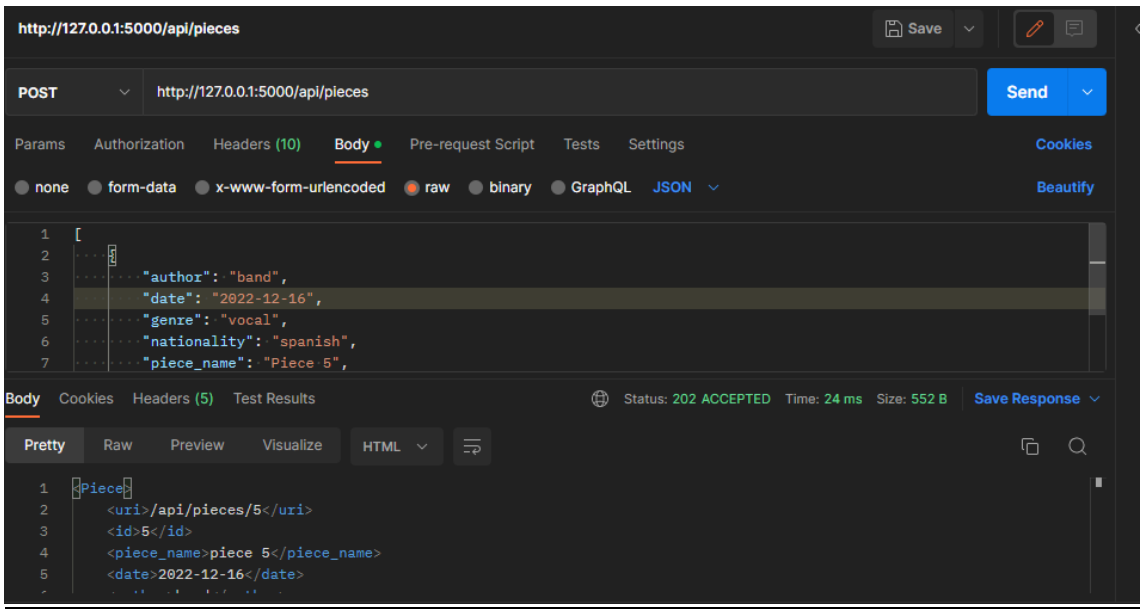


- Añadir una pieza nueva

XML

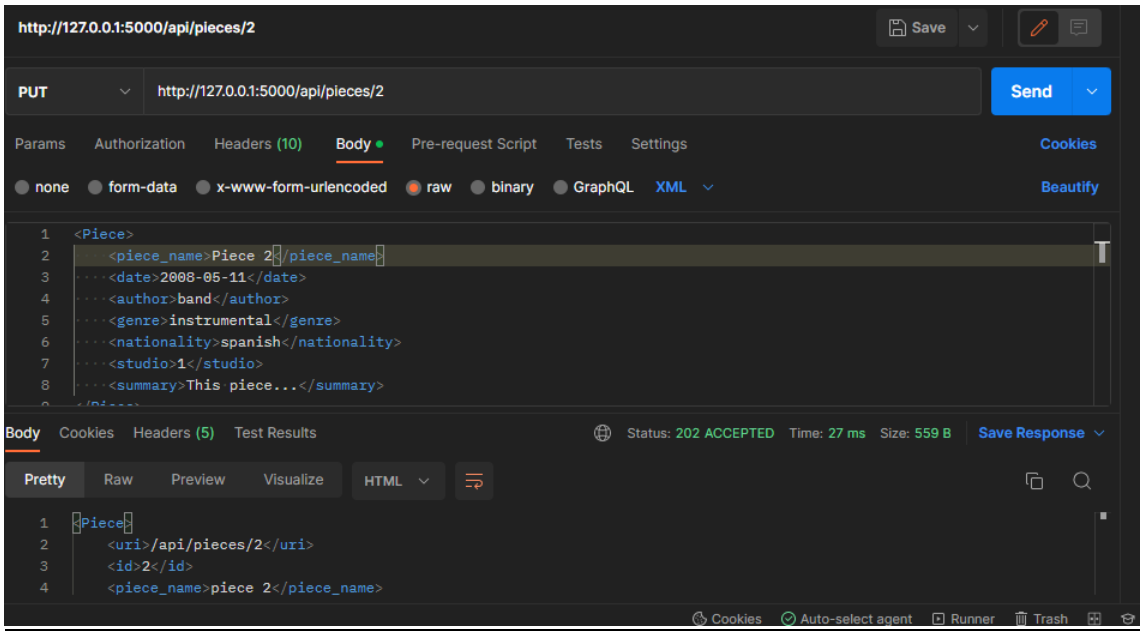


JSON

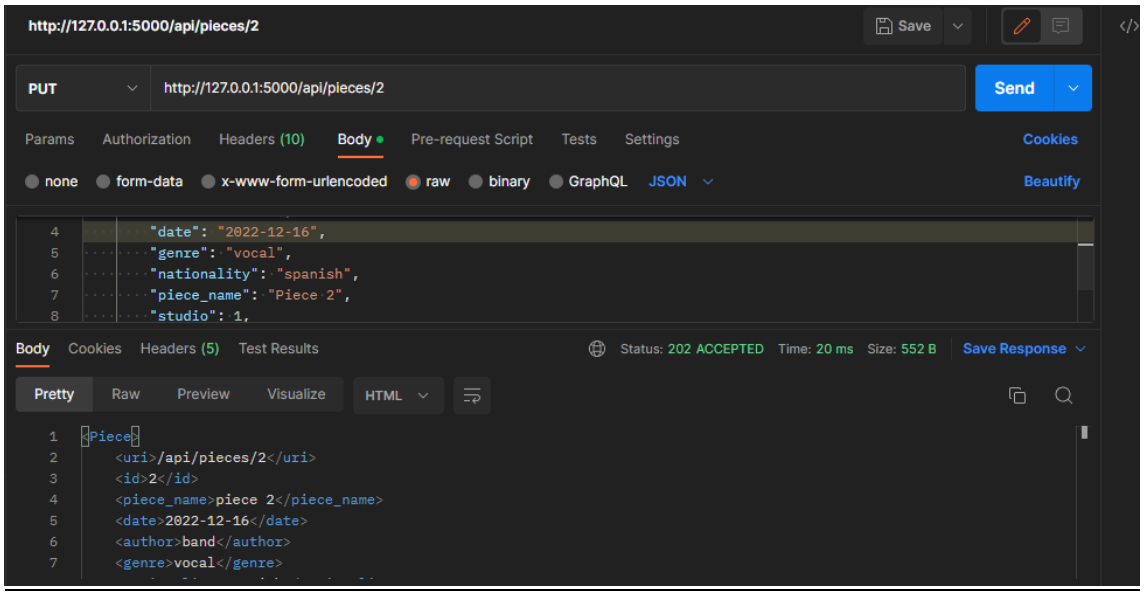


- Editar una pieza de la colección

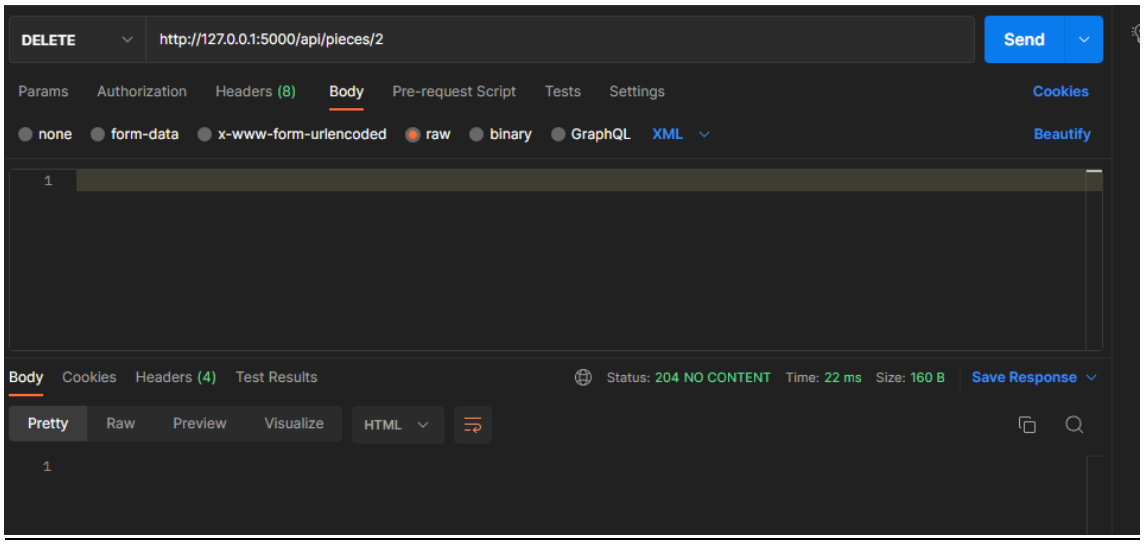
XML



JSON

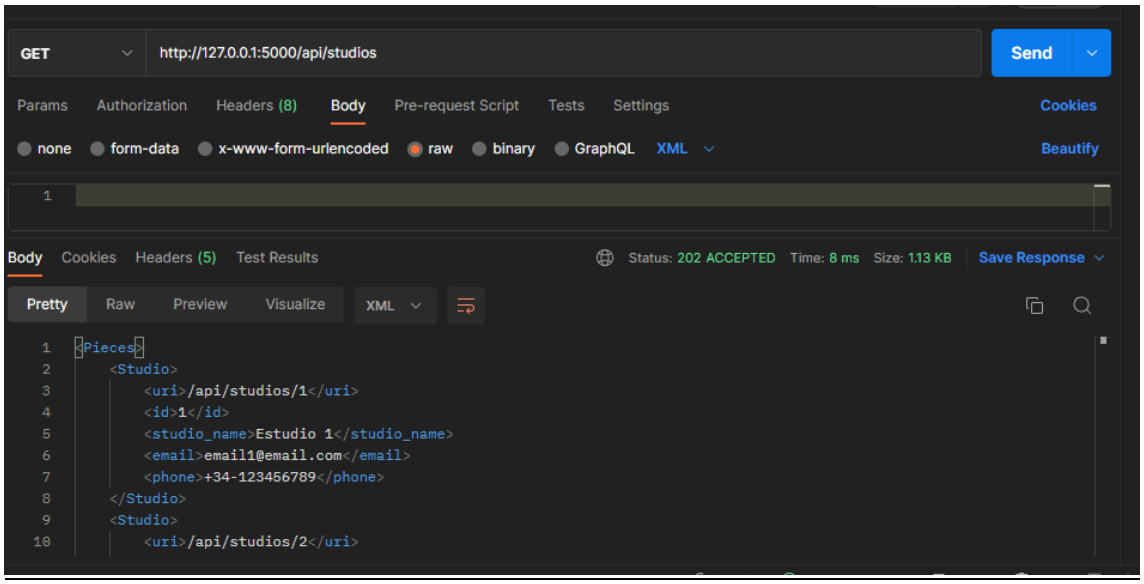


- Eliminar una pieza de la colección

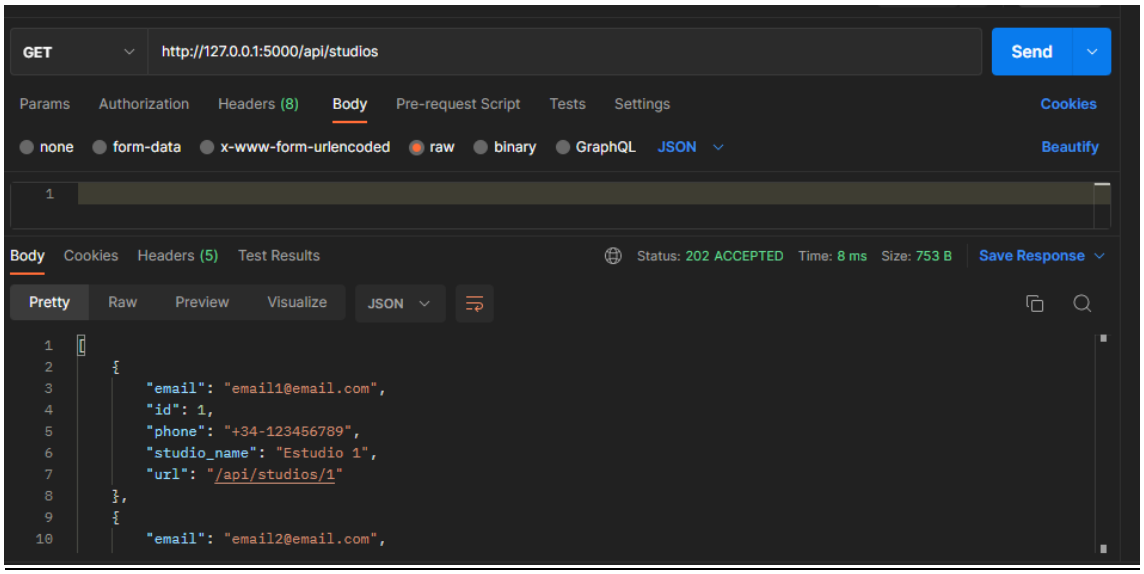


- Obtener una lista de todas las productoras

XML

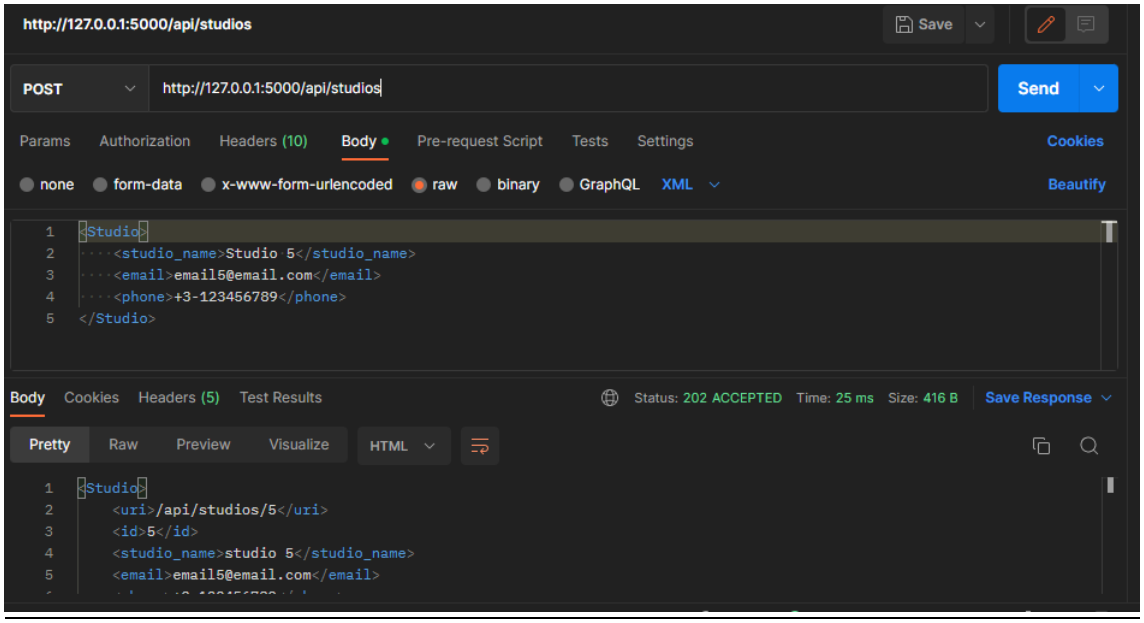


JSON

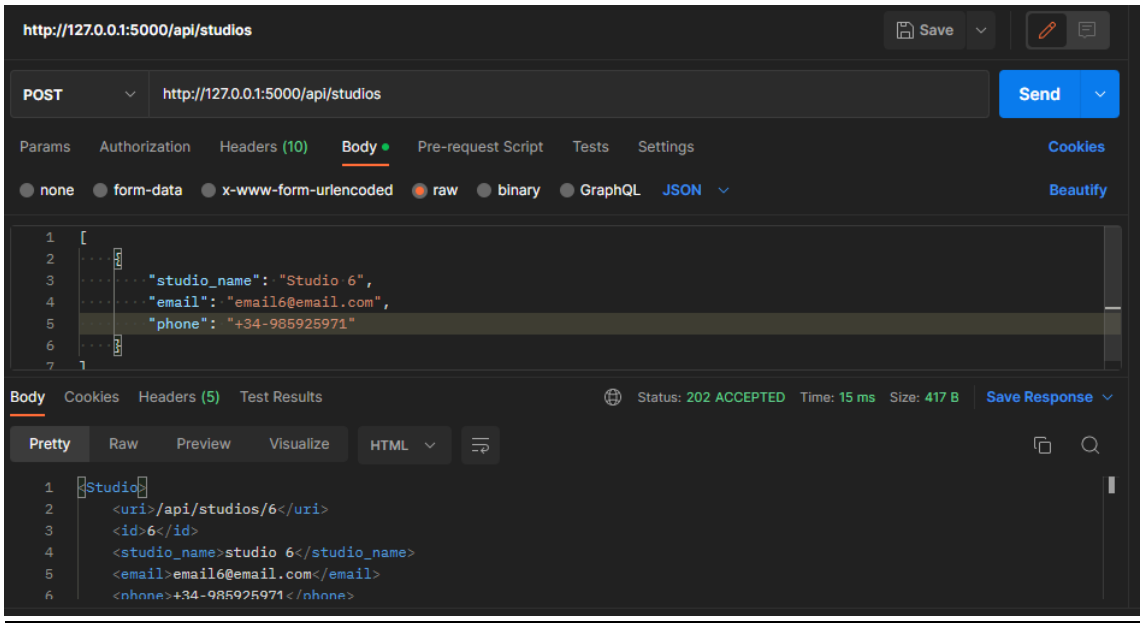


- Añadir una productora nueva

XML

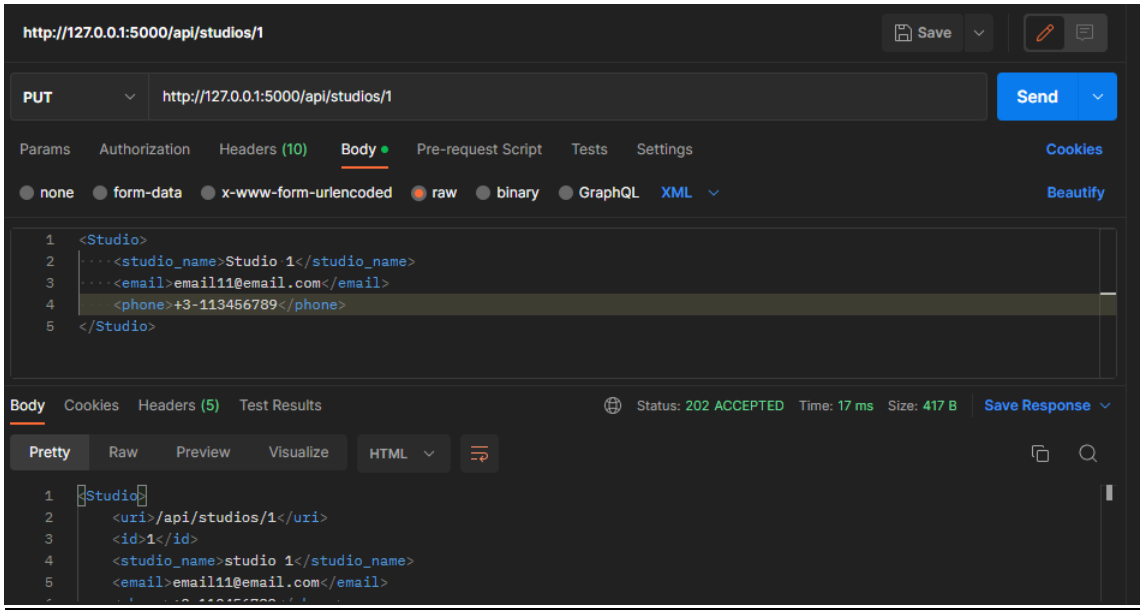


JSON

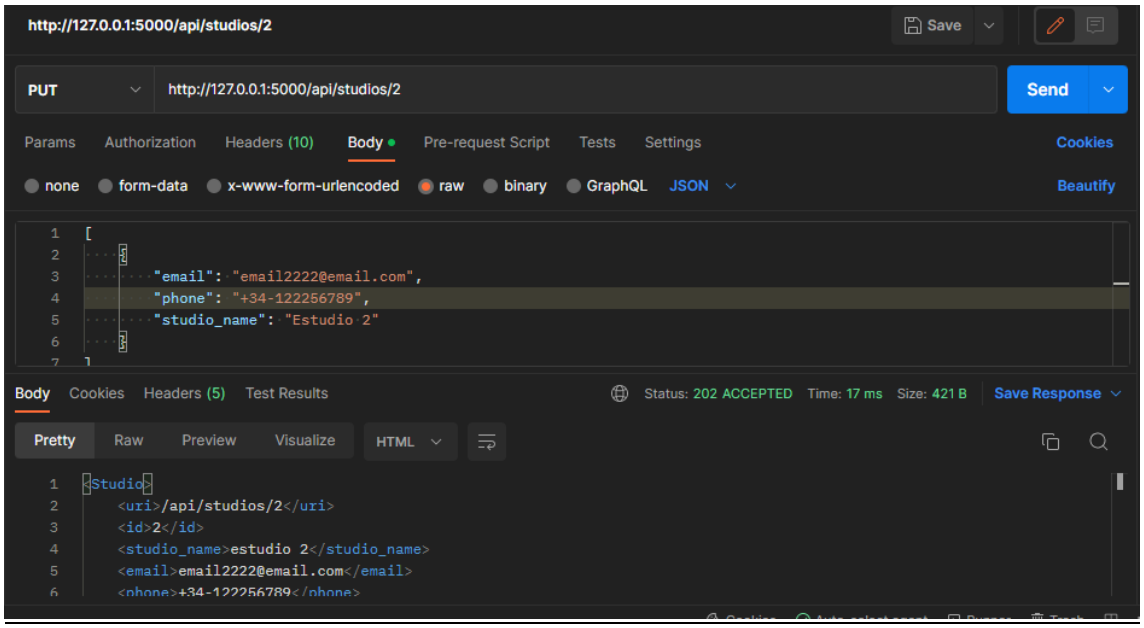


- Editar una productora

XML

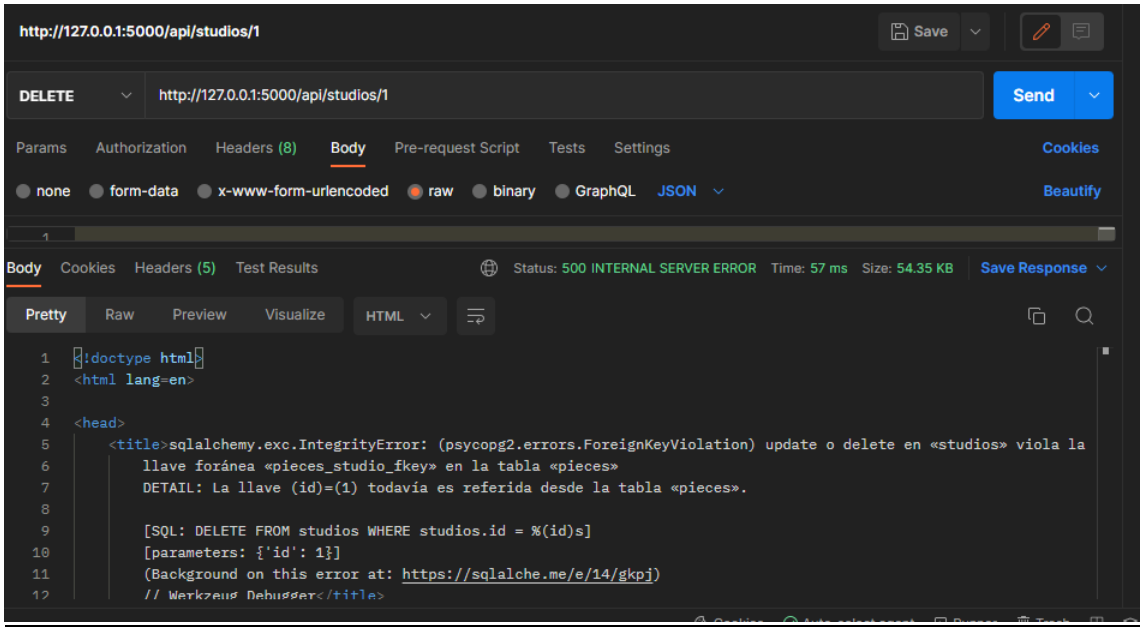


JSON

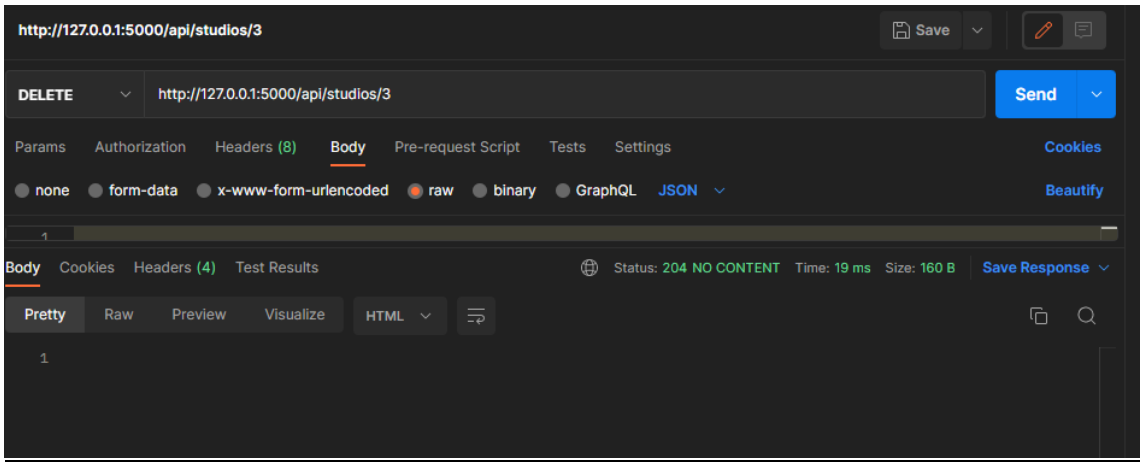


- Eliminar una productora solo si no hay ninguna pieza asociada

Con pieza asociada

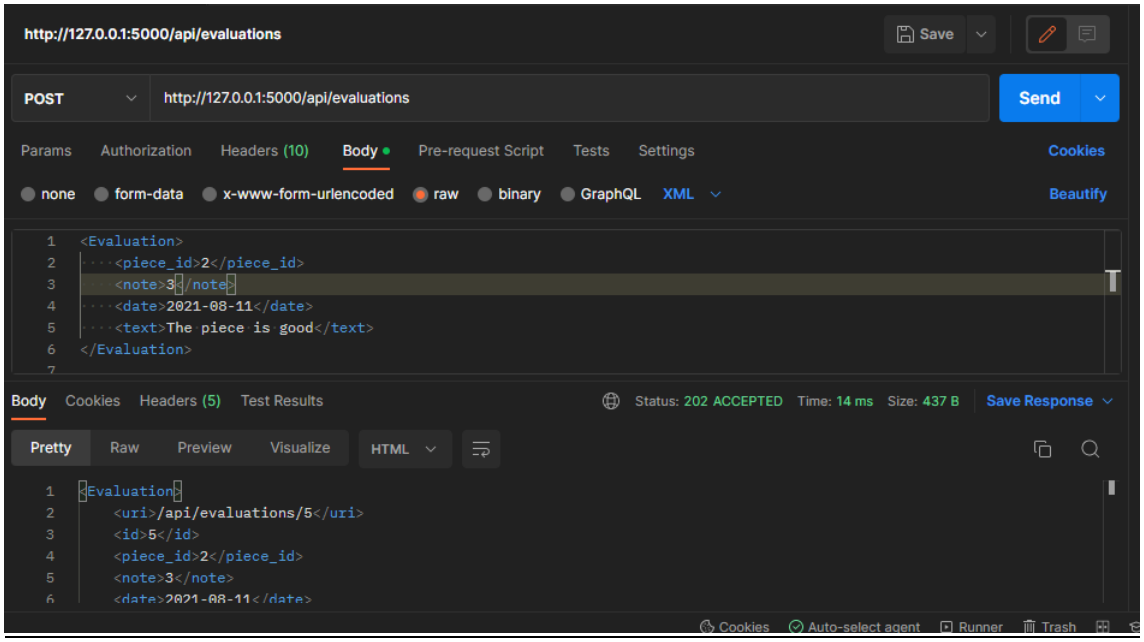


Sin pieza asociada

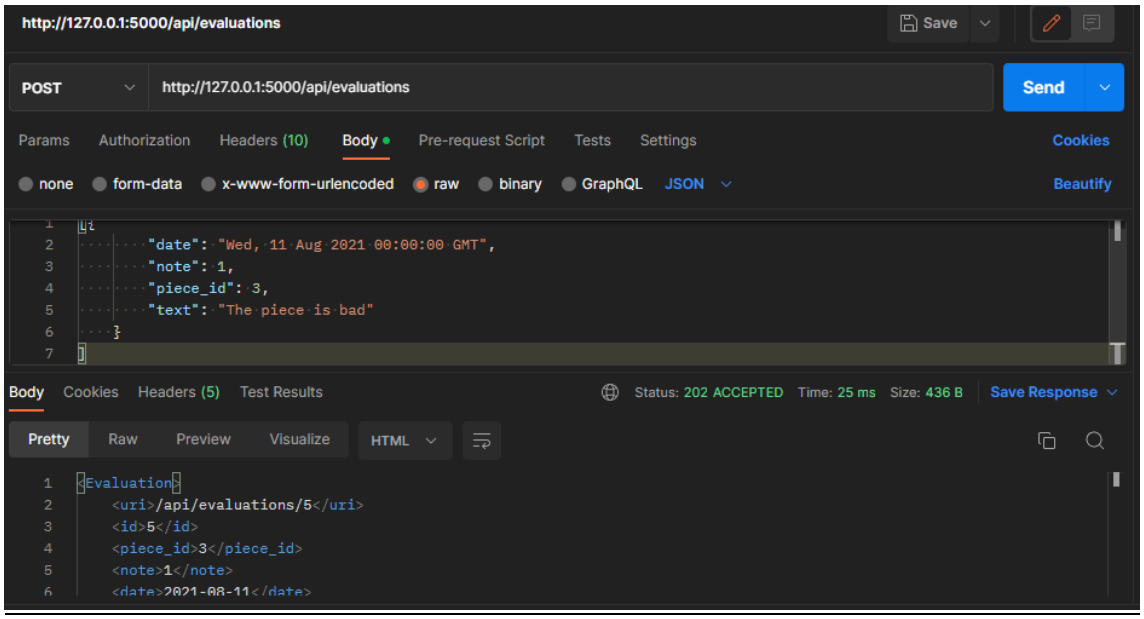


- Publicar una nueva evaluación

XML

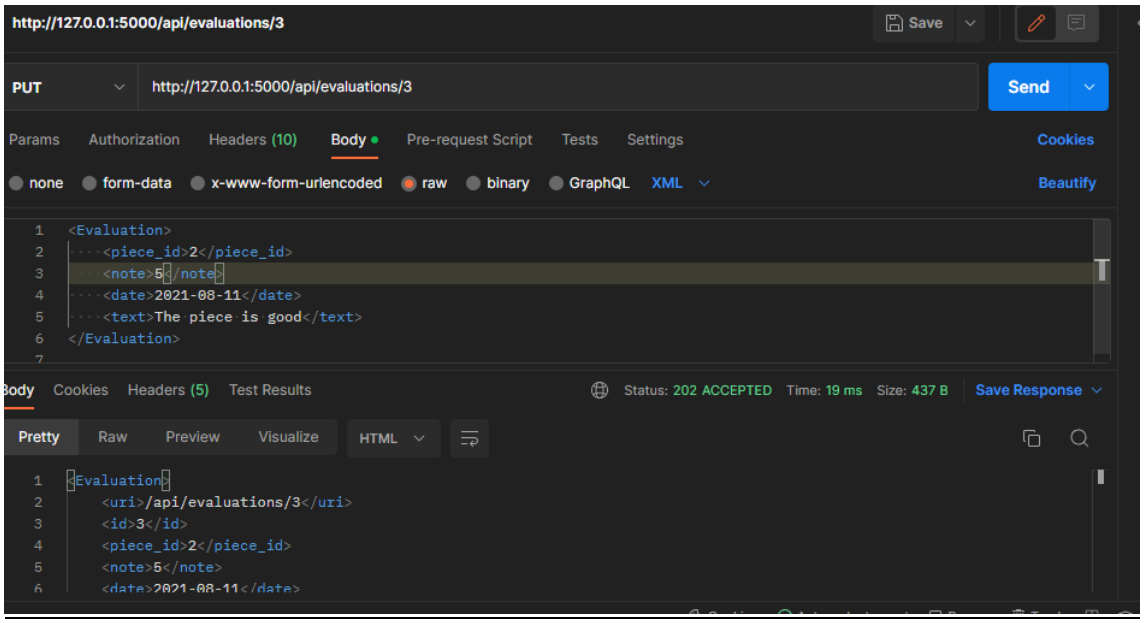


JSON

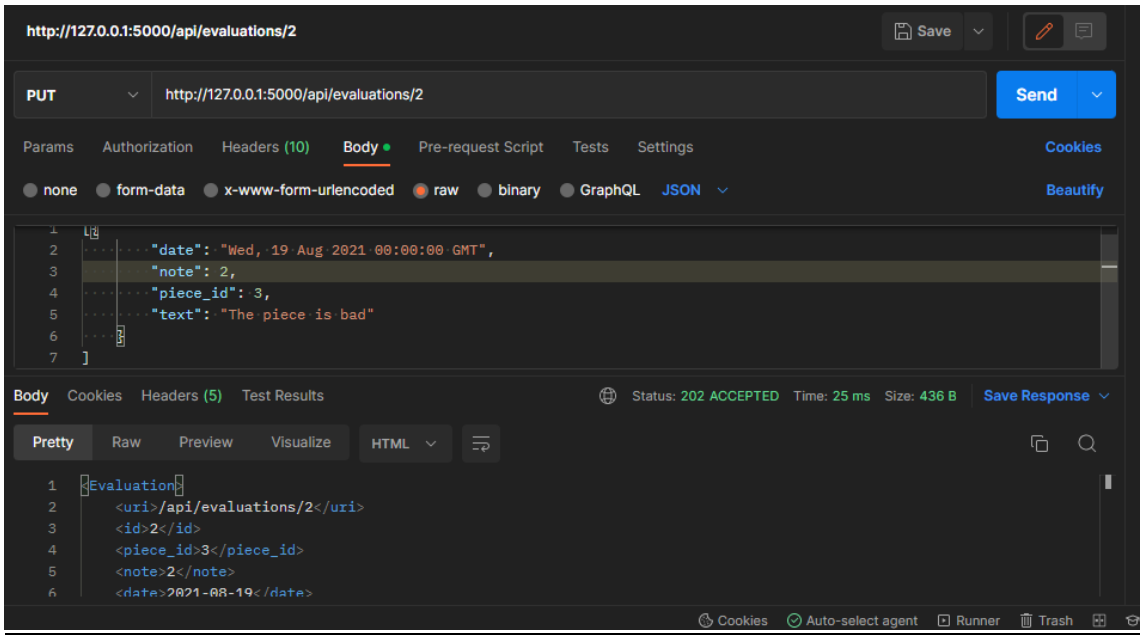


- Editar una evaluación

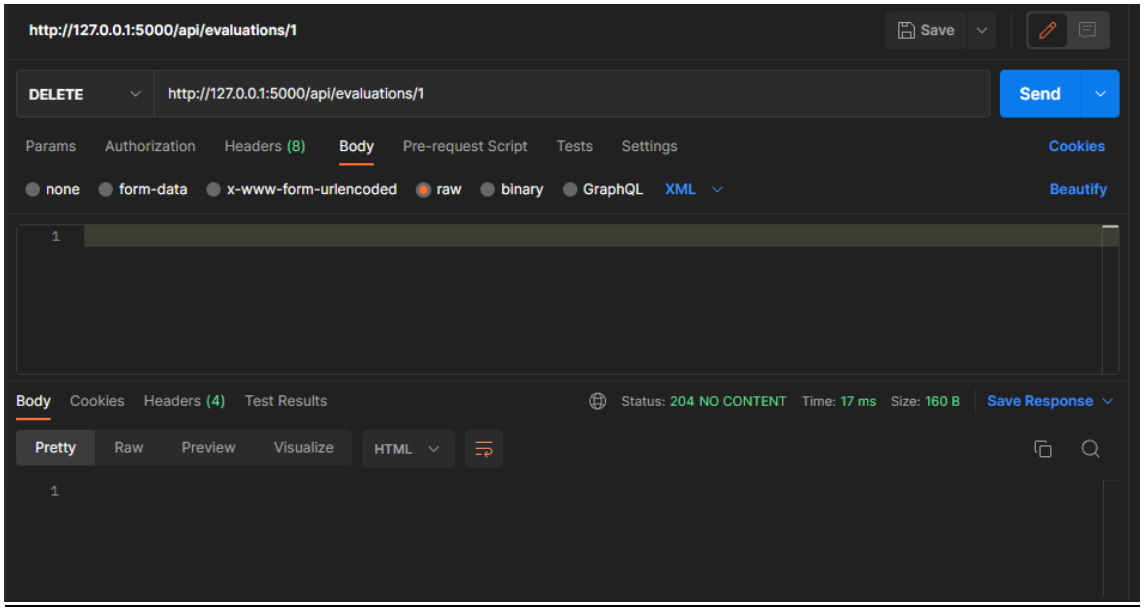
XML



JSON

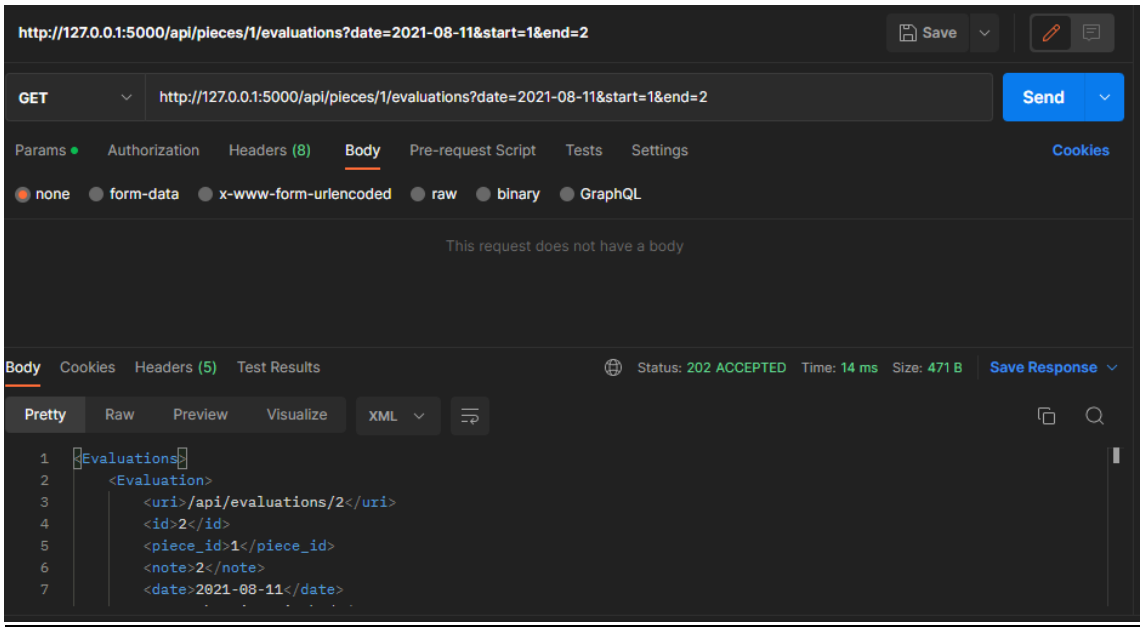


- Eliminar una evaluación

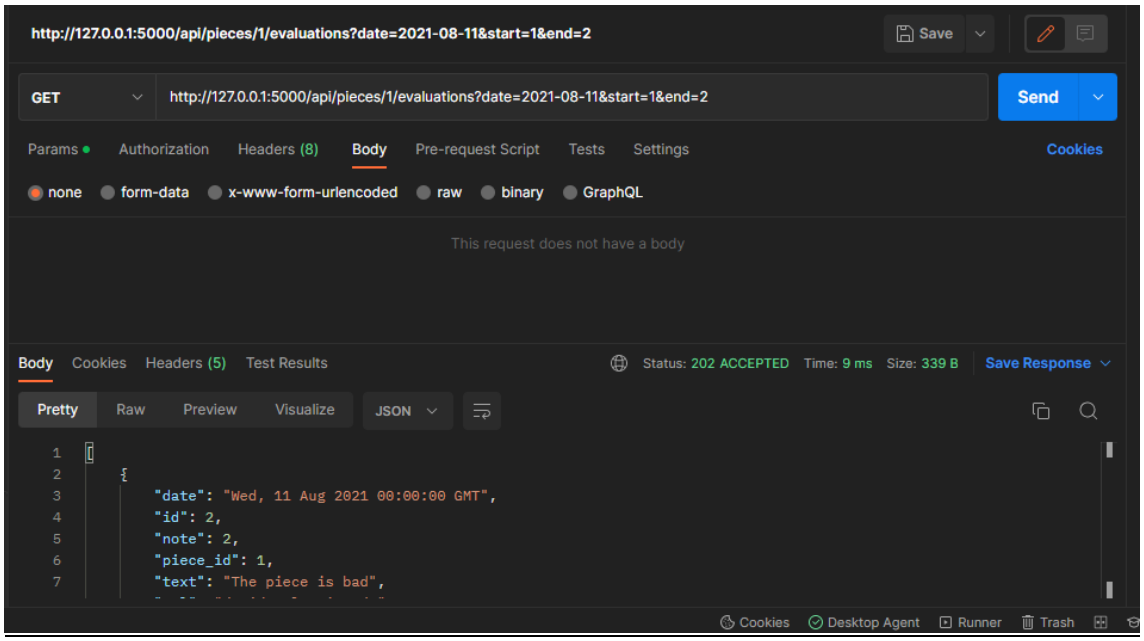


- Obtener una lista de todas las evaluaciones de una pieza y filtrar esa lista por fecha o limitar la cantidad de información obtenida (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

XML

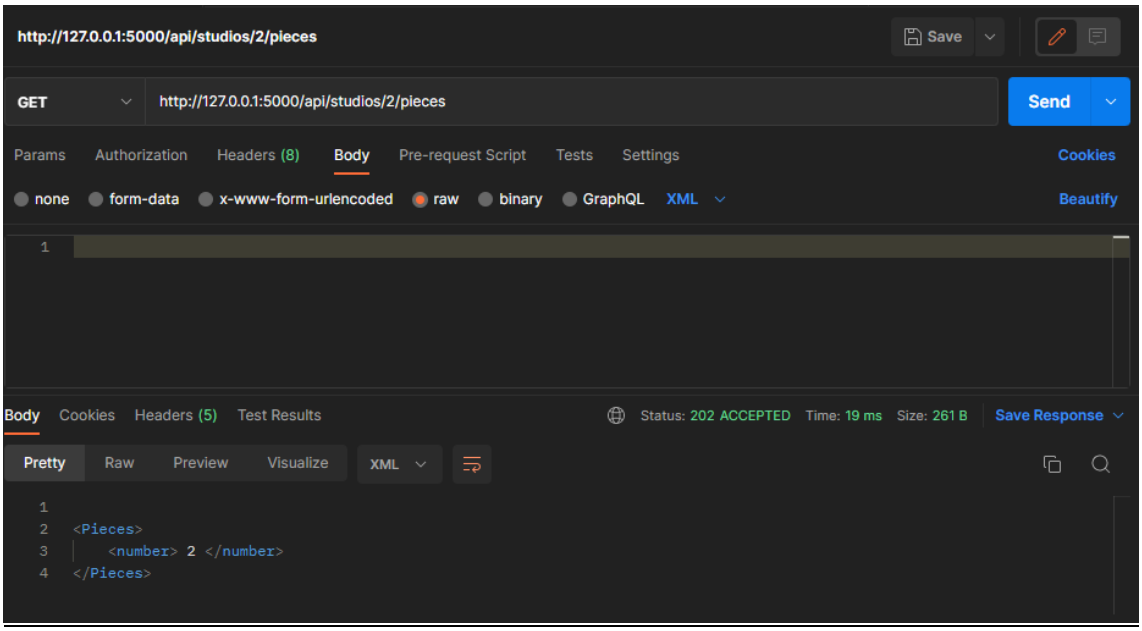


JSON

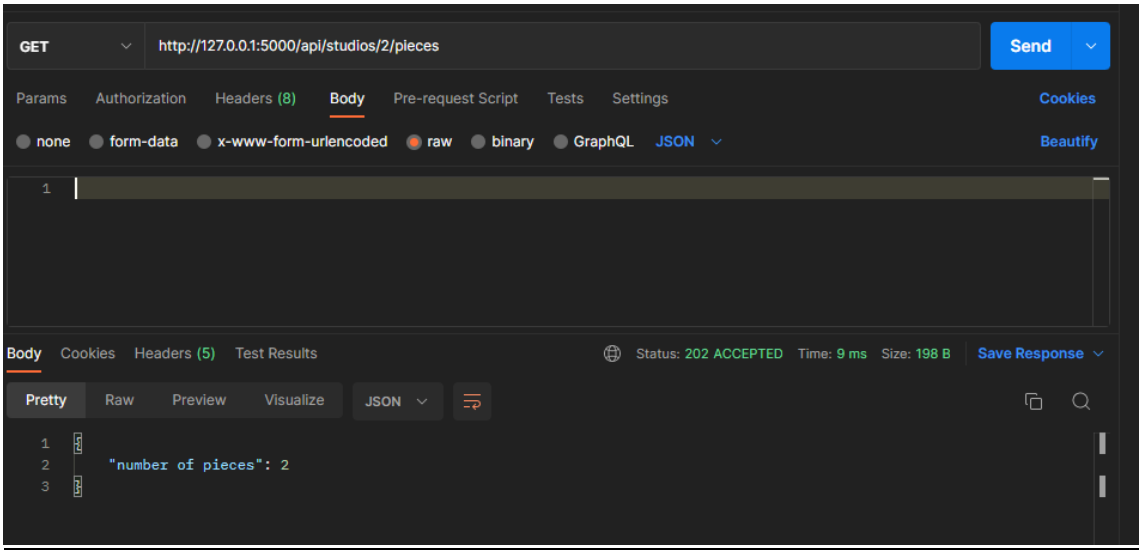


- Obtener el número de piezas dado una productora

XML

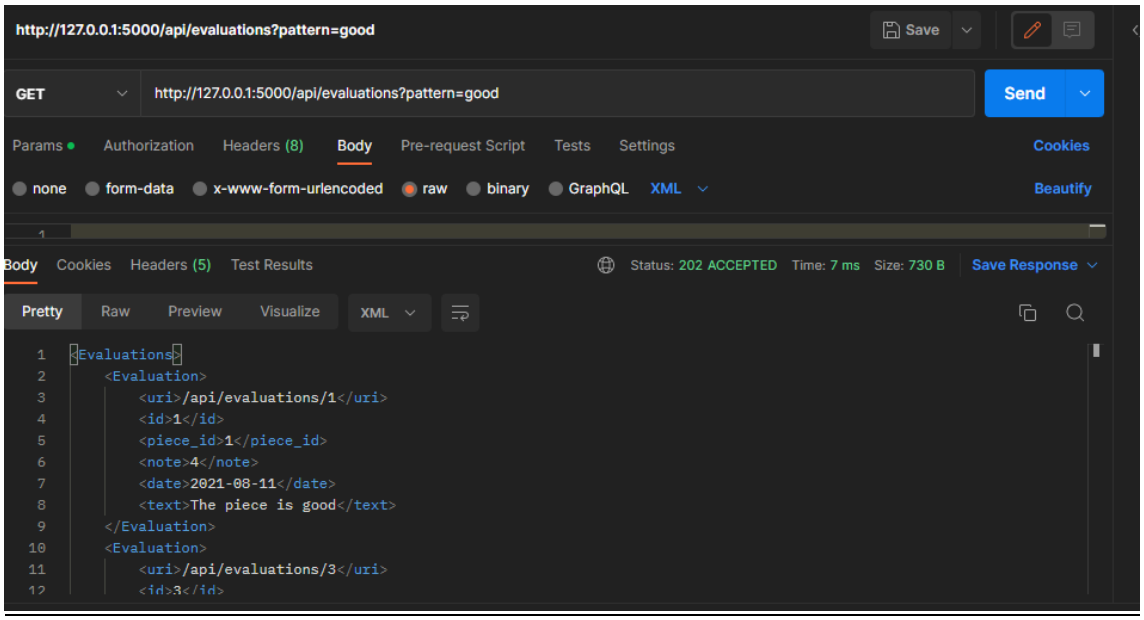


JSON

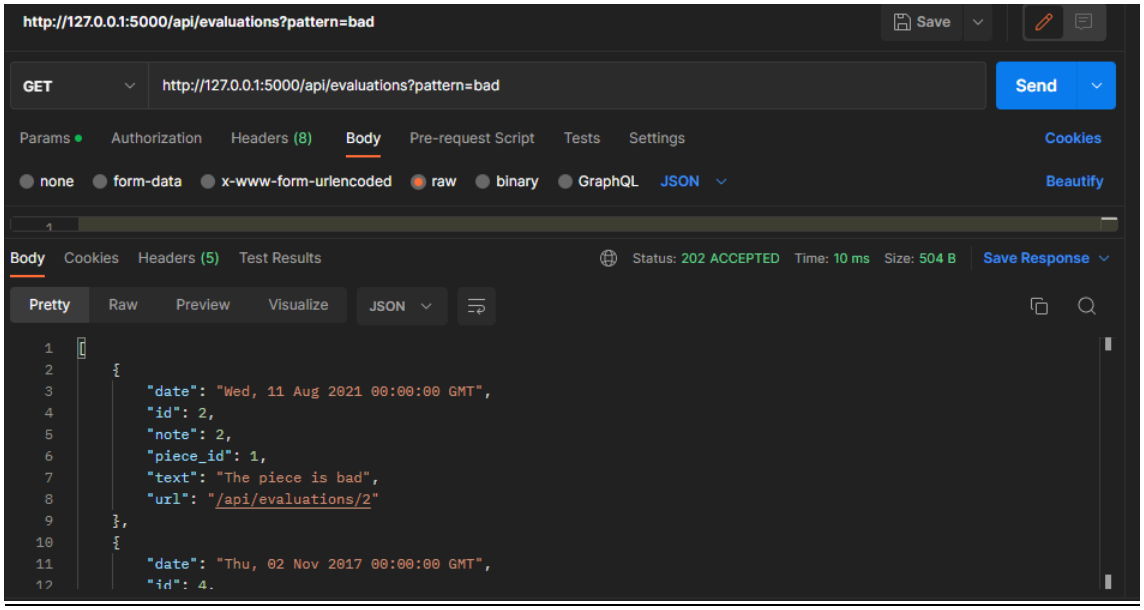


- Obtener la lista de evaluaciones que contienen un determinado texto

XML



JSON



3. Capturas de la ejecución del cliente de prueba para las operaciones anteriormente referidas

Se realizan las mismas operaciones que en el apartado anterior, pero esta vez desde nuestro cliente de prueba.

- Obtener una lista de todas las piezas de la colección

XML

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/pieces
Answer:
-> Status: 202
-> Body:
<Pieces>
        <Piece>
            <uri>/api/pieces/1</uri>
            <id>1</id>
            <piece_name>Piece 1</piece_name>
            <date>2022-12-16</date>
            <author>band</author>
            <genre>vocal</genre>
            <nationality>spanish</nationality>
            <studio>1</studio>
            <summary>This piece...</summary>
        </Piece>

        <Piece>
            <uri>/api/pieces/2</uri>
            <id>2</id>
            <piece_name>Piece 2</piece_name>
            <date>2018-11-14</date>
            <author>composer</author>
            <genre>instrumental</genre>
            <nationality>spanish</nationality>
            <studio>1</studio>
            <summary>This piece...</summary>
```

JSON

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/pieces
Answer:
-> Status: 202
-> Body:
[
  {
    "author": "band",
    "date": "Fri, 16 Dec 2022 00:00:00 GMT",
    "genre": "vocal",
    "id": 1,
    "nationality": "spanish",
    "piece_name": "Piece 1",
    "studio": 1,
    "summary": "This piece...",
    "url": "/api/pieces/1"
  },
  {
    "author": "composer",
    "date": "Wed, 14 Nov 2018 00:00:00 GMT",
    "genre": "instrumental",
    "id": 2,
    "nationality": "spanish",
```

- Añadir una pieza nueva

XML

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/pieces
Answer:
-> Status: 202
-> Body:

       <Piece>
           <uri>/api/pieces/5</uri>
           <id>5</id>
           <piece_name>piece 5</piece_name>
           <date>2011-12-27</date>
           <author>band</author>
           <genre>instrumental</genre>
           <nationality>spanish</nationality>
           <studio>1</studio>
           <summary>this piece...</summary>
       </Piece>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '380', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/pieces
Answer:
-> Status: 202
-> Body:

        <Piece>
            <uri>/api/pieces/5</uri>
            <id>5</id>
            <piece_name>piece 6</piece_name>
            <date>2015-08-19</date>
            <author>composer</author>
            <genre>vocal</genre>
            <nationality>spanish</nationality>
            <studio>2</studio>
            <summary>this piece...</summary>
        </Piece>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '377', 'Connection': 'close'}
```

- Editar una pieza de la colección

XML

```
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/pieces/2
Answer:
-> Status: 202
-> Body:

       <Piece>
           <uri>/api/pieces/2</uri>
           <id>2</id>
           <piece_name>piece 2</piece_name>
           <date>2004-12-17</date>
           <author>band</author>
           <genre>vocal</genre>
           <nationality>spanish</nationality>
           <studio>2</studio>
           <summary>this piece...</summary>
       </Piece>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '373', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/pieces/2
Answer:
-> Status: 202
-> Body:

        <Piece>
            <uri>/api/pieces/2</uri>
            <id>2</id>
            <piece_name>piece 2</piece_name>
            <date>2015-04-19</date>
            <author>composer</author>
            <genre>instrumental</genre>
            <nationality>spanish</nationality>
            <studio>1</studio>
            <summary>this piece...</summary>
        </Piece>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '384', 'Connection': 'close'}
```

- Eliminar una pieza de la colección

```
Request:
-> Action: DELETE
-> URI: http://127.0.0.1:5000/api/pieces/3
Answer:
-> Status: 204
-> Body:

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Connection': 'close'}
```

- Obtener una lista de todas las productoras

XML

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/studios
Answer:
-> Status: 202
-> Body:
<Studios>
        <Studio>
            <uri>/api/studios/1</uri>
            <id>1</id>
            <studio_name>Estudio 1</studio_name>
            <email>email1@email.com</email>
            <phone>+34-123456789</phone>
        </Studio>

        <Studio>
            <uri>/api/studios/2</uri>
            <id>2</id>
            <studio_name>Estudio 2</studio_name>
            <email>email2@email.com</email>
            <phone>+34-234567891</phone>
        </Studio>

        <Studio>
            <uri>/api/studios/3</uri>
            <id>3</id>
            <studio_name>Estudio 3</studio_name>
```

JSON

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/studios
Answer:
-> Status: 202
-> Body:
[
  {
    "email": "email1@email.com",
    "id": 1,
    "phone": "+34-123456789",
    "studio_name": "Estudio 1",
    "url": "/api/studios/1"
  },
  {
    "email": "email2@email.com",
    "id": 2,
    "phone": "+34-234567891",
    "studio_name": "Estudio 2",
    "url": "/api/studios/2"
  },
  {
    "email": "email3@email.com",
    "id": 3,
```

- Añadir una productora nueva

XML

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/studios
Answer:
-> Status: 202
-> Body:

        <Studio>
            <uri>/api/studios/5</uri>
            <id>5</id>
            <studio_name>studio 5</studio_name>
            <email>email5@email.com</email>
            <phone>+99-123456789</phone>
        </Studio>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '238', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/studios
Answer:
-> Status: 202
-> Body:

        <Studio>
            <uri>/api/studios/5</uri>
            <id>5</id>
            <studio_name>studio 6</studio_name>
            <email>email6@email.com</email>
            <phone>+56-565656556</phone>
        </Studio>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '238', 'Connection': 'close'}
```

- Editar una productora

XML

```
Request:
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/studios/2
Answer:
-> Status: 202
-> Body:

        <Studio>
            <uri>/api/studios/2</uri>
            <id>2</id>
            <studio_name>studio 2</studio_name>
            <email>email2222@email.com</email>
            <phone>+22-222256789</phone>
        </Studio>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '241', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/studios/2
Answer:
-> Status: 202
-> Body:

        <Studio>
            <uri>/api/studios/2</uri>
            <id>2</id>
            <studio_name>studio 2</studio_name>
            <email>e2m2a2i2l@email.com</email>
            <phone>+56-222222222</phone>
        </Studio>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '241', 'Connection': 'close'}
```

- Eliminar una productora solo si no hay ninguna pieza asociada

```
Request:
-> Action: DELETE
-> URI: http://127.0.0.1:5000/api/studios/3
Answer:
-> Status: 204
-> Body:

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Connection': 'close'}
```

- Publicar una nueva evaluación

XML

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/evaluations
Answer:
-> Status: 202
-> Body:

        <Evaluation>
            <uri>/api/evaluations/5</uri>
            <id>5</id>
            <piece_id>1</piece_id>
            <note>4</note>
            <date>2014-09-14</date>
            <text>the piece is good</text>
        </Evaluation>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '258', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: POST
-> URI: http://127.0.0.1:5000/api/evaluations
Answer:
-> Status: 202
-> Body:

        <Evaluation>
            <uri>/api/evaluations/5</uri>
            <id>5</id>
            <piece_id>2</piece_id>
            <note>4</note>
            <date>2017-08-11</date>
            <text>the piece is good</text>
        </Evaluation>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '258', 'Connection': 'close'}
```

- Editar una evaluación

XML

```
Request:
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/evaluations/3
Answer:
-> Status: 202
-> Body:

        <Evaluation>
            <uri>/api/evaluations/3</uri>
            <id>3</id>
            <piece_id>2</piece_id>
            <note>2</note>
            <date>2015-09-14</date>
            <text>the piece is bad</text>
        </Evaluation>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Content-Length': '257', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: PUT
-> URI: http://127.0.0.1:5000/api/evaluations/3
Answer:
-> Status: 202
-> Body:

        <Evaluation>
            <uri>/api/evaluations/3</uri>
            <id>3</id>
            <piece_id>2</piece_id>
            <note>2</note>
            <date>2017-08-11</date>
            <text>the piece is bad</text>
        </Evaluation>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'text/html; cha
rset=utf-8', 'Content-Length': '257', 'Connection': 'close'}
```

- Eliminar una evaluación

```
Request:
-> Action: DELETE
-> URI: http://127.0.0.1:5000/api/evaluations/3
Answer:
-> Status: 204
-> Body:

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 15:59:33 GMT', 'Content-Type': 'text/html; char
set=utf-8', 'Connection': 'close'}
```

- Obtener una lista de todas las evaluaciones de una pieza y filtrar esa lista por fecha o limitar la cantidad de información obtenida (e.g. los 10 primeros elementos, los elementos entre el 11 y el 20, etc.)

XML

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/pieces/1/evaluations?date=2021-08-11&start=1&end=2
Answer:
-> Status: 202
-> Body:
<Evaluations>
        <Evaluation>
            <uri>/api/evaluations/2</uri>
            <id>2</id>
            <piece_id>1</piece_id>
            <note>2</note>
            <date>2021-08-11</date>
            <text>The piece is bad</text>
        </Evaluation>
         </Evaluations>
-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 16:08:03 GMT', 'Content-Type': 'application/xml
; charset=utf-8', 'Content-Length': '286', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/pieces/1/evaluations?date=2021-08-11&start=1&end=2
Answer:
-> Status: 202
-> Body:
[
  {
    "date": "Wed, 11 Aug 2021 00:00:00 GMT",
    "id": 2,
    "note": 2,
    "piece_id": 1,
    "text": "The piece is bad",
    "url": "/api/evaluations/2"
  }
]

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:15:41 GMT', 'Content-Type': 'application/js
on', 'Content-Length': '168', 'Connection': 'close'}
```

- Obtener el número de piezas dado una productora

XML

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/studios/2/pieces
Answer:
-> Status: 202
-> Body:

        <Pieces>
            <number> 2 </number>
        </Pieces>

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 16:15:32 GMT', 'Content-Type': 'application/xml
; charset=utf-8', 'Content-Length': '77', 'Connection': 'close'}
```

JSON

```
Request:
 -> Action: GET
 -> URI: http://127.0.0.1:5000/api/studios/1/pieces
 Answer:
 -> Status: 202
 -> Body:
 {
   "number of pieces": 2
 }

 -> Header:
 {'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:23:42 GMT', 'Content-Type': 'application/js
 on', 'Content-Length': '28', 'Connection': 'close'}
```

- Obtener la lista de evaluaciones que contienen un determinado texto

XML

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/evaluations?pattern=good
Answer:
-> Status: 202
-> Body:
<Evaluations>
        <Evaluation>
            <uri>/api/evaluations/1</uri>
            <id>1</id>
            <piece_id>1</piece_id>
            <note>4</note>
            <date>2021-08-11</date>
            <text>The piece is good</text>
        </Evaluation>

        <Evaluation>
            <uri>/api/evaluations/5</uri>
            <id>5</id>
            <piece_id>1</piece_id>
            <note>4</note>
            <date>2014-09-14</date>
         </Evaluations>
-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 16:16:46 GMT', 'Content-Type': 'application/xml
; charset=utf-8', 'Content-Length': '1061', 'Connection': 'close'}
```

JSON

```
Request:
-> Action: GET
-> URI: http://127.0.0.1:5000/api/evaluations?pattern=bad
Answer:
-> Status: 202
-> Body:
[
  {
    "date": "Wed, 11 Aug 2021 00:00:00 GMT",
    "id": 2,
    "note": 2,
    "piece_id": 1,
    "text": "The piece is bad",
    "url": "/api/evaluations/2"
  }
]

-> Header:
{'Server': 'Werkzeug/2.2.2 Python/3.9.6', 'Date': 'Fri, 16 Dec 2022 17:24:45 GMT', 'Content-Type': 'application/js
on', 'Content-Length': '168', 'Connection': 'close'}
```