Mario Parra Castiñeiras

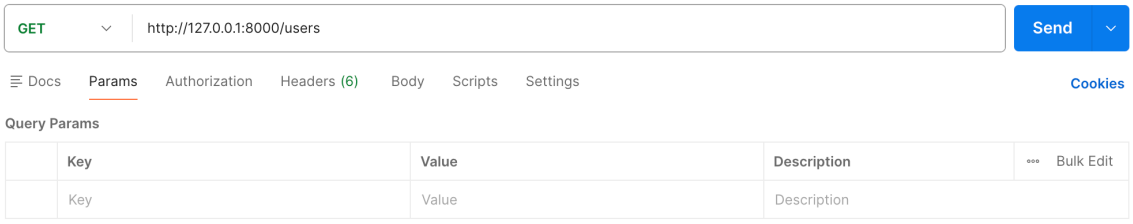
Desarrollo de una API Completa – DOCUMENTACIÓN GRAFICA

Endpoints de la API:

GET /users

```python
@app.get("/users")
async def get_user_list():
    mydb = DataBaseConnection(host="localhost", user="root", password="123123123", database="ENTREGA1")
    mydb_conn=mydb.get_connection()
    mycursor = mydb_conn.cursor()
    mycursor.execute("SELECT * FROM users")
    data=mycursor.fetchall()
    mydb_conn.close()
    return data
```

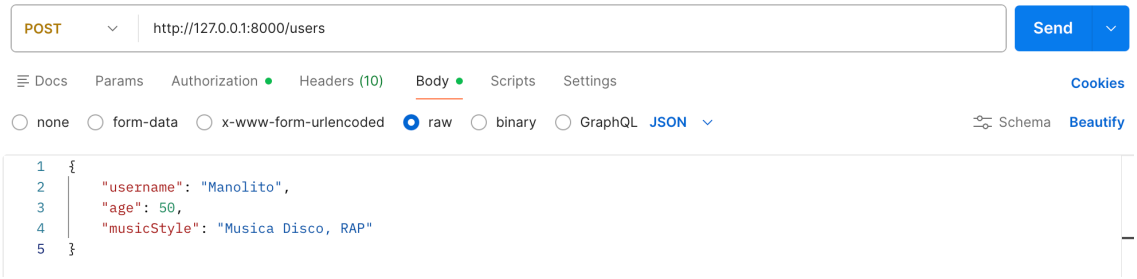
Este endpoint está diseñado para consultar en la base de datos conectada la tabla de usuarios. En esta tabla se mostrará el ID, Nombre, Edad y estilos musicales del usuario.



POST /users

```python
@app.post("/users")
async def post_users(request: Request):
    mydb = DataBaseConnection(host="localhost", user="root", password="123123123", database="ENTREGA1")
    mydb_conn = mydb.get_connection()
    request = await request.json()
    username = request['username']
    age = request['age']
    musicStyle = request['musicStyle']
    mycursor = mydb_conn.cursor()
    mycursor.execute(f"INSERT INTO users (username, age, musicStyle) VALUES ('{username}', {age}, '{musicStyle}')")
    mydb_conn.commit()
    return JSONResponse(content={"message": "User added successfully"}, status_code=201)
```

El objetivo de este endpoint es poder añadir usuarios mediante la API, para ello se deberá proporcionar los siguientes datos: Nombre, Edad y estilo/s de música.

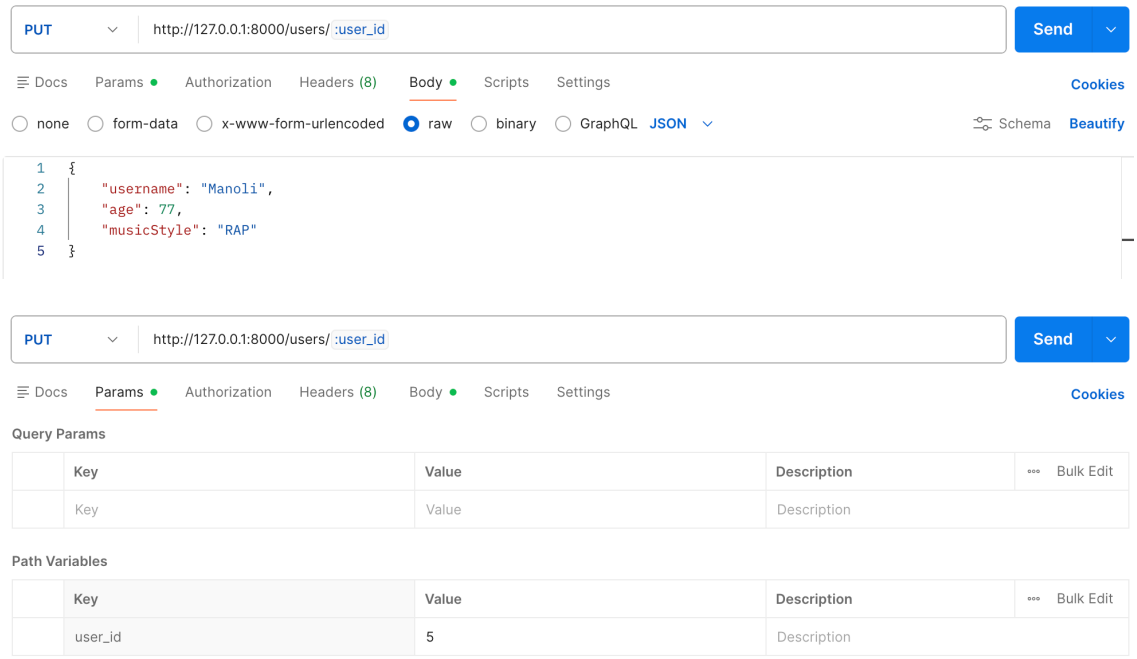


Mario Parra Castiñeiras

PUT /users

```python
@app.put("/users/{user_id}")
async def put_users(user_id: int, request: Request):
    #Connexion to dataBase
    mydb = DataBaseConnection(host="localhost", user="root", password="123123123", database="ENTREGA1")
    mydb_conn = mydb.get_connection()
    #datos a json
    request = await request.json()
    username = request['username']
    age = request['age']
    musicStyle = request['musicStyle']

    #cursor
    mycursor = mydb_conn.cursor()

    mycursor.execute(f"UPDATE users SET username = %s, age=%s, musicStyle=%s WHERE id=%s", (username, age, musicStyle, user_id))
    mydb_conn.commit()
    return JSONResponse(content={"message": "User updated successfully"}, status_code=201)
```

Para este endpoint se busca poder ser capaces de modificar cualquiera de los parámetros de la base de datos (menos el ID que se asigna automáticamente).

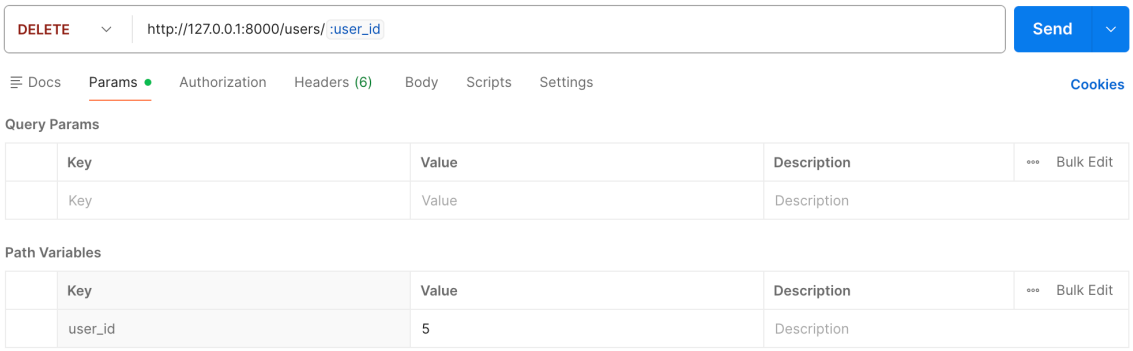


Mario Parra Castiñeiras

DELETE /users/{user_id}

```python
@app.delete("/users/{user_id}")
async def delete_users(user_id: int, request: Request):

    mydb = DataBaseConnection(host="localhost", user="root", password="123123123", database="ENTREGA1")
    mydb_conn = mydb.get_connection()

    mycursor = mydb_conn.cursor()
    mycursor.execute(f"DELETE FROM users WHERE id=%s", (user_id,))
    mydb_conn.commit()

    return JSONResponse(content={"message": "User deleted successfully"}, status_code=201)
```

Para poder eliminar usuarios se ha implementado de la misma manera que en casos anteriores solo que ahora para poder eliminar un usuario deberemos de ser conocedores de su ID y proporcionarlo como un Path Param.



Mario Parra Castiñeiras

POST /Spotify/token

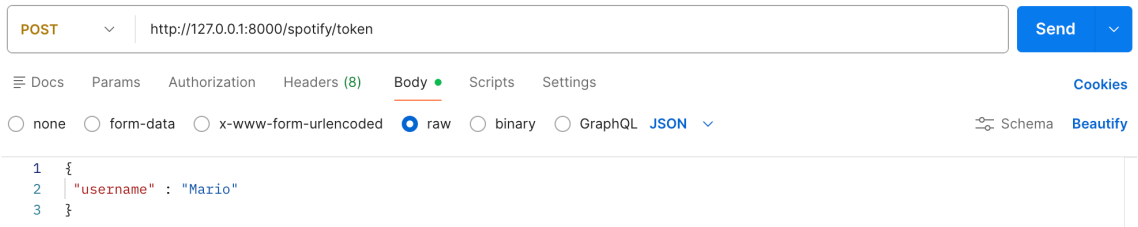
```python
@app.post("/spotify/token")
async def spotifytoken(request: Request):
    global SPOTIFY_TOKEN_GLOBAL

    data = await request.json()
    username = data.get("username")

    if not username:
        raise HTTPException(status_code=400, detail="You must provide a username")

    mydb = DataBaseConnection(host="localhost", user="root", password="123123123", database="ENTREGA1")
    mydb_conn = mydb.get_connection()
    mycursor = mydb_conn.cursor()

    mycursor.execute("SELECT * FROM users WHERE username=%s", (username,))
    user = mycursor.fetchone()
    mydb_conn.commit()

    if not user:
        raise HTTPException(status_code=404, detail="User not registered. Please register to get access")

    SPOTIFY_TOKEN_GLOBAL = get_Token(SPOTIFY_CLIENT_ID, SPOTIFY_CLIENT_SECRET)
    if "error" in SPOTIFY_TOKEN_GLOBAL:
        raise HTTPException(status_code=404, detail="Token couldn't be generated")
    return JSONResponse(content=SPOTIFY_TOKEN_GLOBAL, status_code=200)
    print(SPOTIFY_TOKEN_GLOBAL)
```

Para poder acceder a la API Publica de Spotify es necesario que generarnos un Token el cual permitirá acceder a la información pública (es decir toda aquella que su endpoint no contenga un /me/), esta es la función de este endpoint.

El planteamiento ha sido el siguiente, este endpoint solo será accesible para aquellos usuarios que consten en la base de datos, en caso de que no consten se les devolverá un mensaje conforme deben registrarse previamente.

En caso de que el usuario conste en la base de datos se le devolverá el token, el cual se guardara en una base de datos global para poder utilizarlo en los siguientes endpoints.

La idea de hacerlo en este endpoint es para evitar que usuarios no registrados puedan acceder a las peticiones contra la API Publica de Spotify.

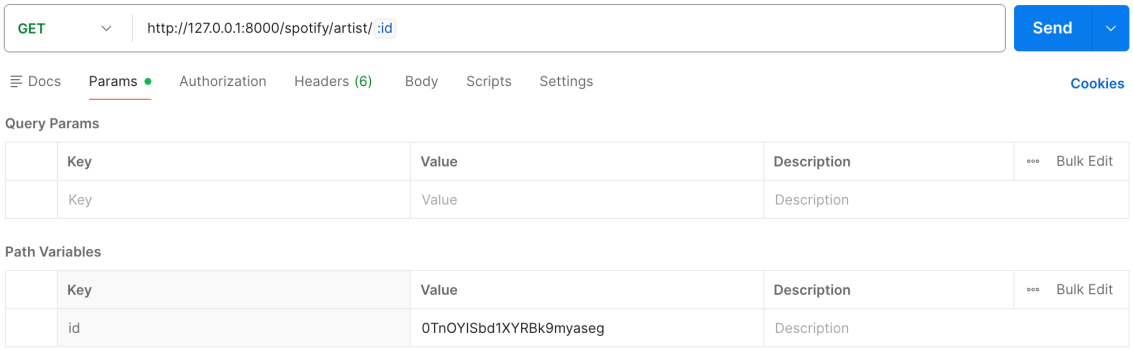


Mario Parra Castiñeiras

GET /spotify/artists/{id}

```python
@app.get("/spotify/artist/{id}")
async def artistinfo(id: str):
    global SPOTIFY_TOKEN_GLOBAL
    artist_data=get_Artists(SPOTIFY_TOKEN_GLOBAL, id)
    if "error" in artist_data:
        raise HTTPException(status_code=404, detail="Artist Not found")
    if "token_error" in artist_data:
        raise HTTPException(status_code=401, detail="Invalid token. Please try again with a new token")
    return JSONResponse(content=artist_data, status_code=200)
    print(artist_data)
```

Mediante este endpoint se podrá consultar la información del artista escogido, para ello se deberá de tener un token en vigor y el ID del artista a consultar. Este ID se pasará mediante un Path Param. Si por algún motivo estuviera caducado o no existiera el token se devolverá un error al usuario para que obtenga uno nuevo.



GET /Spotify/releases

```python
@app.get("/spotify/releases")
async def newreleases():
    global SPOTIFY_TOKEN_GLOBAL
    releases_data = get_NewReleases(SPOTIFY_TOKEN_GLOBAL)

    if "error" in releases_data:
        raise HTTPException(status_code=404, detail="Unable to load latest data, try again later")
    if "token_error" in releases_data:
        raise HTTPException(status_code=401, detail="Invalid token. Please try again with a new token")
    return JSONResponse(content=releases_data, status_code=200)
    print(releases_data)
```

El objetivo de este endpoint es obtener información sobre los últimos lanzamientos.