

## Universidad de la Fuerzas Armadas "ESPE"

<b>Nombre:</b> Mateo Jarén García Galarza	<b>Docente:</b> Diego Sosa
<b>NRC:</b> 2267	<b>Fecha:</b> 2025-01-12
<b>Asignatura:</b> Sistemas de Bases de Datos	<b>Calificación:</b>

Objetivo: Construir un servicio API (Application Programming interface) REST con HTTP, mediante el uso del framework Flask y el uso de MONGODB y MongoDB Atlas con la finalidad de implementar un recurso practicop para bases de datos.

**Indicador:** Caso de estudio de REST API en Python utilizando Flask y MongoDB:

### Funcionalidades incluidas:

**Obtener todos los usuarios** (GET /usuarios).

**Obtener un usuario por ID** (GET /usuarios/<id>).

**Crear un usuario** (POST /usuarios).

**Actualizar un usuario por ID** (PUT /usuarios/<id>).

**Eliminar un usuario por ID** (DELETE /usuarios/<id>).

### Integración:

Las credenciales fueron asignadas en un archivo .env para seguridad.

Como primer punto se inicia la coneccion a la base de datos remota y configuraciones del framework

```
1  #@author: Mateo Garcia
2
3  #dependencies
4
5  from dotenv import load_dotenv
6  from flask import Flask, request
7  from flask_pymongo import PyMongo
8  from urllib.parse import quote_plus
9  from werkzeug.security import generate_password_hash, check_password_hash
10 from bson.objectid import ObjectId
11 from constants import *
12 from validator import *
13 import os
14
15
16
17 # Settings
18
19 app = Flask(__name__)
20
21 # Load environment variables
22
23 load_dotenv()
24
25 USERNAME = os.getenv('USERNAME')
26 PASSWD = os.getenv('PASSWORD')
27
28 app.config['MONGO_URI'] = f"mongodb+srv://{USERNAME}:{PASSWD}@cluster0.4vufa.mongodb.net/espe?retryWrites=true&w=majority&appName=Cluster0&
29
30 # Initialize PyMongo to work with MongoDB
31 mongoConnection = PyMongo(app)
```

### Rutas:

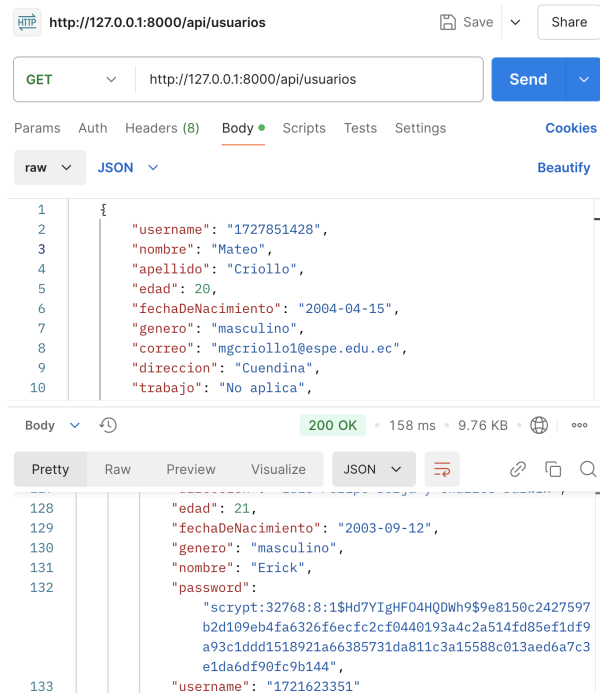
```
1 #Constants for server
2 test_db_connection = '/test-db-connection'
3 all_users = '/api/usuarios'
4 user_by_id = '/api/usuarios/<id>'
5
```

## Servicios:

1. Obtener todos los usuarios (GET /usuarios).

```
1 @app.route(all_users, methods=['GET'])
2 def get_all_users():
3     try:
4         users = mongoConnection.db.users.find()
5         response = []
6         for user in users:
7             user['_id'] = str(user['_id'])
8             response.append(user)
9         return {
10             'data': response,
11             'status': 200
12         }
13     except :
14         return internal_server_error()
```

## Consumo:



http://127.0.0.1:8000/api/usuarios

GET http://127.0.0.1:8000/api/usuarios

Send

Params Auth Headers (8) Body Scripts Tests Settings Cookies

raw JSON Beautify

```
1 {
2   "username": "1727851428",
3   "nombre": "Mateo",
4   "apellido": "Criollo",
5   "edad": 20,
6   "fechaDeNacimiento": "2004-04-15",
7   "genero": "masculino",
8   "correo": "mgcriollo1@espe.edu.ec",
9   "direccion": "Cuendina",
10  "trabajo": "No aplica",
11}
```

Body 200 OK 158 ms 9.76 KB

Pretty Raw Preview Visualize JSON

```
128 "edad": 21,
129 "fechaDeNacimiento": "2003-09-12",
130 "genero": "masculino",
131 "nombre": "Erick",
132 "password":
    "script:32768:8:1$Hd7YIghF04HQDWh9$9e8156c2427597
    b2d109eb4fa6326f6ecfc2cf0440193a4c2a514fd85ef1df9
    a93c1ddd1518921a66385731da811c3a15588c013aed6a7c3
    e1da6df90fc9b144",
133 "username": "1721623351"
```

2. Obtener un usuario por ID (GET /usuarios/<id>).

```
1 @app.route(user_by_id, methods=['GET'])
2 def get_user_by_id(id):
3     print('buscar usuario por id', id)
4     try:
5         user = mongoConnection.db.users.find_one({'username': id})
6         if user:
7             user['_id'] = str(user['_id'])
8             return {
9                 'data': user,
10                'status': 200
11            }
12         else:
13             return not_found()
14     except:
15         return internal_server_error()
16
```

Consumo:

GET

http://127.0.0.1:8000/api/usuarios/1727851428

Send

ParamsAuthHeaders (8)BodyScriptsTestsSettingsCookies

rawJSONBeautify

1{

2"username": "1727851428",

3"nombre": "Mateo",

4"apellido": "Criollo",

5"edad": 20,

6"fechaDeNacimiento": "2004-04-15",

7"genero": "masculino",

8"correo": "mgcriollo1@espe.edu.ec",

9"direccion": "Cuendina",

10"trabajo": "No aplica",

Body200 OK • 1.38 s • 654 B •

PrettyRawPreviewVisualizeJSON

10"nombre": "Mateo",

11"password":

"script:32768:8:1\$N3yLHVg0AuVzLyWt\$d6f21b845340490760

910a6a7e172bbef5c97a435de732f33219d9b44fd592c9a7806e8

b7a94ec31596dd118ac3ff2d3596618fb70229260d31655052d04

0d84",

12"username": "1727851428"

13},

14"status": 200

15}

### 3. Crear un usuario (POST /usuarios).

```

1 @app.route(all_users, methods=['POST'])
2 def create_user():
3     print('crear usuario')
4     try:
5         username = request.json['username']
6         nombre = request.json['nombre']
7         apellido = request.json['apellido']
8         correo = request.json['correo']
9         fecha_nacimiento = request.json['fechaDeNacimiento']
10        direccion = request.json['direccion']
11        edad = request.json['edad']
12        genero = request.json['genero']
13        password = request.json['password']
14
15        #check for all fields into request to create a user
16        if not username or not nombre or not apellido or not correo or not fecha_nacimiento or not direccion or not edad or not genero or not password:
17            return bad_request()
18
19
20
21        if not validate_email(correo):
22            return bad_request()
23    except:
24        return bad_request()
25
26    # if username and email and password:
27    password = generate_password_hash(password)
28    mongoConnection.db.users.insert_one(
29        {
30            'username': username,
31            'nombre': nombre,
32            'apellido': apellido,
33            'correo': correo,
34            'fechaDeNacimiento': fecha_nacimiento,
35            'direccion': direccion,
36            'edad': edad,
37            'genero': genero,
38            'password': password
39        }
40    )
41    print('Usuario creado' + username)
42    # else:
43    #     return bad_request()
44    return {
45        'message': 'User creado con éxito',
46        'status': 201
47    }

```

Consumo:

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

- URL:** `http://127.0.0.1:8000/api/usuarios`
- Method:** `POST`
- Request Body (JSON):**

```

{
  "username": "1727851428",
  "nombre": "Mateo",
  "apellido": "Criollo",
  "edad": 20,
  "fechaDeNacimiento": "2004-04-15",
  "genero": "masculino",
  "correo": "mgcriollo1@espe.edu.ec",
  "direccion": "Cuendina",
  "password": "1234567890"
}

```
- Response Status:** `200 OK` (342 ms, 223 B)
- Response Body (JSON):**

```

{
  "message": "User creado con éxito",
  "status": 201
}

```

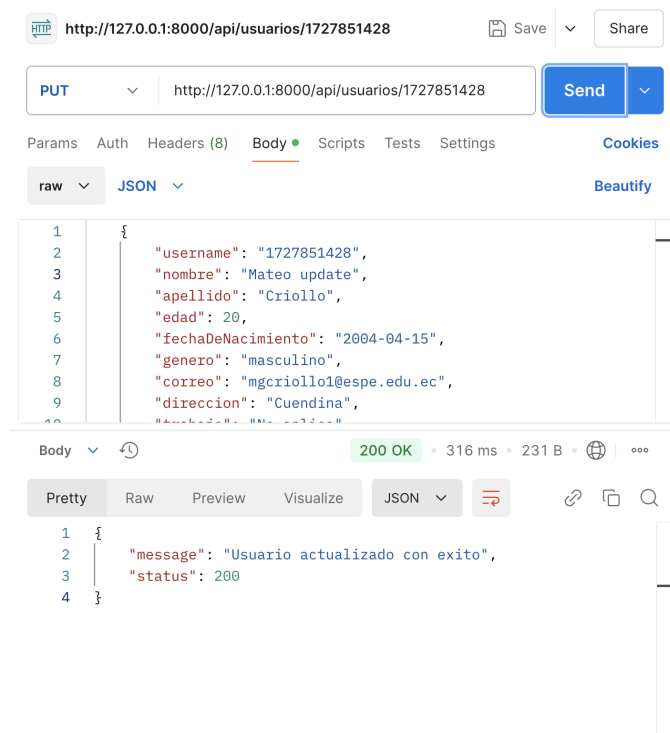
#### 4. Actualizar un usuario por ID (PUT /usuarios/<id>).

```

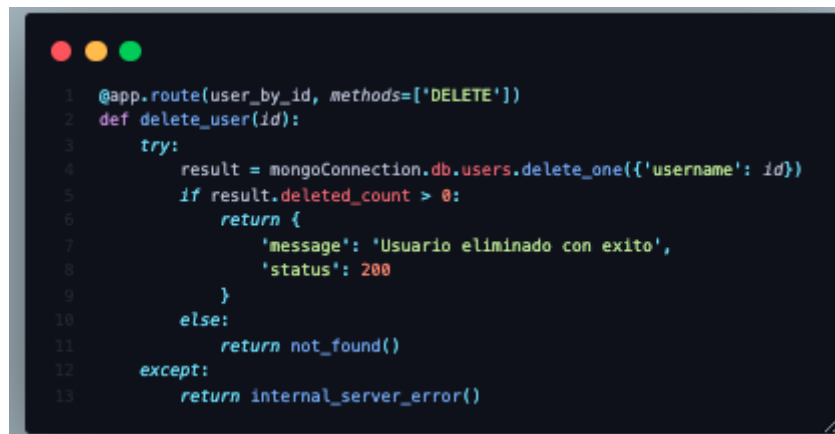
1  @app.route(user_by_id, methods=['PUT'])
2  def update_user(id):
3      try:
4          username = request.json['username']
5          nombre = request.json['nombre']
6          apellido = request.json['apellido']
7          correo = request.json['correo']
8          fecha_nacimiento = request.json['fechaDeNacimiento']
9          direccion = request.json['direccion']
10         edad = request.json['edad']
11         genero = request.json['genero']
12         password = request.json['password']
13
14
15     except:
16         return bad_request()
17
18     if username and nombre and apellido and correo and fecha_nacimiento and direccion and edad and genero and password:
19         password = generate_password_hash(password)
20         result = mongoConnection.db.users.update_one(
21             {'username': id},
22             {
23                 '$set': {
24                     'username': username,
25                     'nombre': nombre,
26                     'apellido': apellido,
27                     'correo': correo,
28                     'fechaDeNacimiento': fecha_nacimiento,
29                     'direccion': direccion,
30                     'edad': edad,
31                     'genero': genero,
32                     'password': password
33                 }
34             }
35         )
36         if result.modified_count > 0:
37             return {
38                 'message': 'Usuario actualizado con exito',
39                 'status': 200
40             }
41         else:
42             return not_found()
43     else:
44         return bad_request()

```

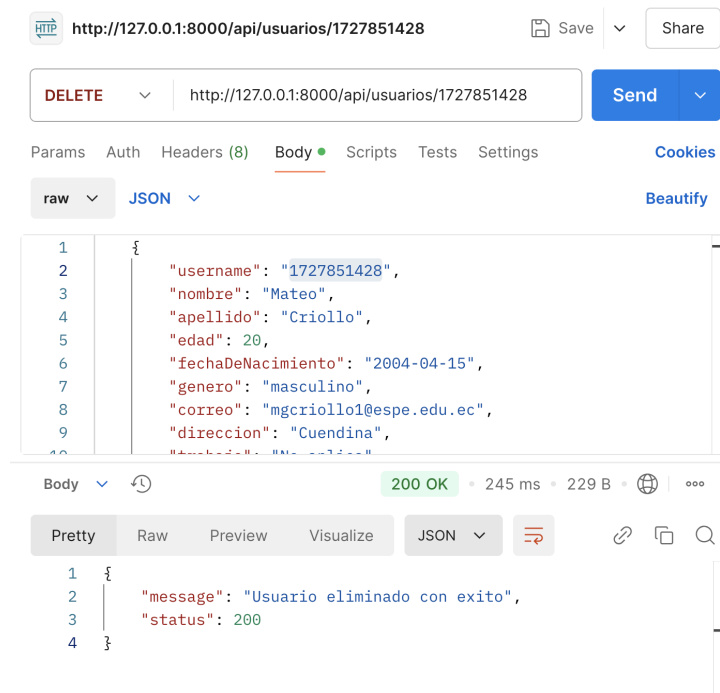
Consumo:



## 5. Eliminar un usuario por ID (DELETE /usuarios/<id>).



Consumo:



**Conclusiones:**

- Los servicios REST Http permiten la fácil manipulación de bases de datos y lógica, así mismo como la fácil integración con clientes.
- El formato JSON nos permite transferir información al protocolo HTTP de una manera sencilla, ya que cuenta con una fácil integración a lenguajes modernos como python.
- Los método GET, POST, PUT y DELETE a demas de ser estándares nos permiten llevar una mejor semántica en la construcción de un servicio rest Http

**Anexos:**

Proyecto Adjuntado a la entrega de este documento