

## Mini-dossier técnico – Modulo Persona API

**Proyecto:** Codium

**Desarrolladores:** Eberson Guayllas – Wilmer Pardo – Santiago Tamayo

### 1. Introducción

El modulo persona da los endpoints utilizados en la gestión de usuarios dentro del sistema Codium. Se realizan operaciones CRUD (crear, leer, actualiza, eliminar) sobre los registros de los usuarios que utilizan la plataforma, del mismo modo se puede obtener información paginada o individual.

Los endpoints están implementados mediante Flask Blueprints y se comunican con el modelo PersonaModel, este realiza consulta a la base de datos MySQL con la conexión db.py.

Este modulo resulta importante en el backend ya que se encarga de administrar la información base de los usuarios, esto para su autenticación y otras funcionalidades.

### 2. Endpoints RESTful

*GET /personas*

Obtiene una lista paginada de todas las personas activas en la base de datos . La ruta para obtener el listado es: <http://127.0.0.1:5000/api/personas>.

Los códigos de respuesta pueden ser 200 (lista de personas obtenidas correctamente) o 500 (error interno del servidor)

Ejemplo de solicitud: GET <http://127.0.0.1:5000/api/personas>.

Ejemplo de respuesta:

```
[
  {
    "apellidos": "Tamayo",
    "correo": "santiago@example.com",
    "id_persona": 2,
    "id_rol": 2,
    "nombre": "Santiago",
    "nombre_usuario": "stamayo",
    "num_retos_resueltos": 0,
    "puntaje_total": 0
  }
]
```

*GET /personas/<id\_persona>*

Este método obtiene la información de una persona específica mediante su ID. La ruta requerida es: [http://127.0.0.1:5000/api/personas/<id\\_persona/](http://127.0.0.1:5000/api/personas/<id_persona/). Y esta nos puede generar los códigos de respuesta 200, 404 (persona no encontrada) y 500 (error interno del servidor)

El parámetro a utilizar sería el id\_persona (int)

Ejemplo de solicitud: GET <http://127.0.0.1:5000/api/personas/5/>

Ejemplo de respuesta:

```
{
  "apellidos": "Pardo",
  "correo": "wilmer.pardo @correo.com",
  "id_persona": 1,
  "id_rol": 2,
  "nombre": "Wilmer",
  "nombre_usuario": "wpardo",
  "num_retos_resueltos": 0,
  "puntaje_total": 0
}
```

*POST /personas*

Crea un registro de usuario en la base de datos, asegurando que la contraseña se encripta antes de ser almacenada en la base de datos para proteger la información del usuario. La ruta para registrar una persona es: <http://127.0.0.1:5000/api/personas>.

Los parámetros a utilizar son los presentes en el siguiente body, donde todos los campos son strings. Los códigos que se pueden generar son 201, 400 y 500.

Ejemplo de solicitud:

POST <http://127.0.0.1:5000/api/personas>

```
{
  "nombre": "Wilmer",
  "apellidos": "Pardo",
  "correo": "wilmer.pardo@correo.com ",
  "contrasena_plana": "passWilner123",
  "nombre_usuario": "wpardo"
}
```

Ejemplo de respuesta:

```
{
  "id_persona": 1,
  "message": "Persona creada exitosamente"
}
```

*PUT /personas/<id\_persona>*

Actualiza los datos permitidos de una persona (nombre, apellidos, nombre\_usuario, token\_refresco). Su ruta para actualizar una persona es: <http://127.0.0.1:5000/api/personas/<idPersona>>

Ejemplo de solicitud:

PUT <http://127.0.0.1:5000/api/personas/3>

```
{
  nombre: "daniel"
}
```

Ejemplo de respuesta

```
{
  message: "Persona actualizada"
}
```

*DELETE /personas/<id\_persona>*

Este método no elimina la persona de forma definitiva, sino la desactiva realizando un “**borrado lógico**”, cambiando su estado a inactivo (esta\_activo = FALSE). La ruta para eliminar una persona es: <http://127.0.0.1:5000/api/personas/<idPersona>>. Y el parámetro es id\_persona (int), esta solicitud puede generar los códigos 200 y 500.

Ejemplo de solicitud:

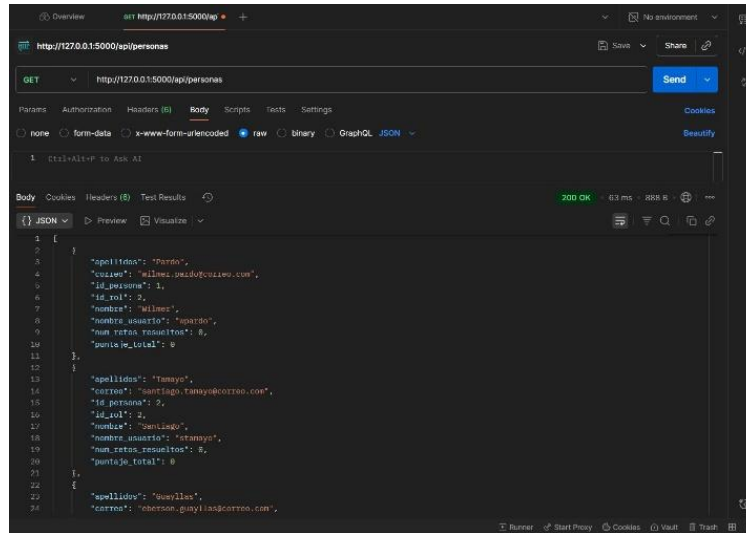
DELETE <http://127.0.0.1:5000/api/personas/3>

Ejemplo de respuesta:

```
{
  "message": "Persona desactivada"
}
```

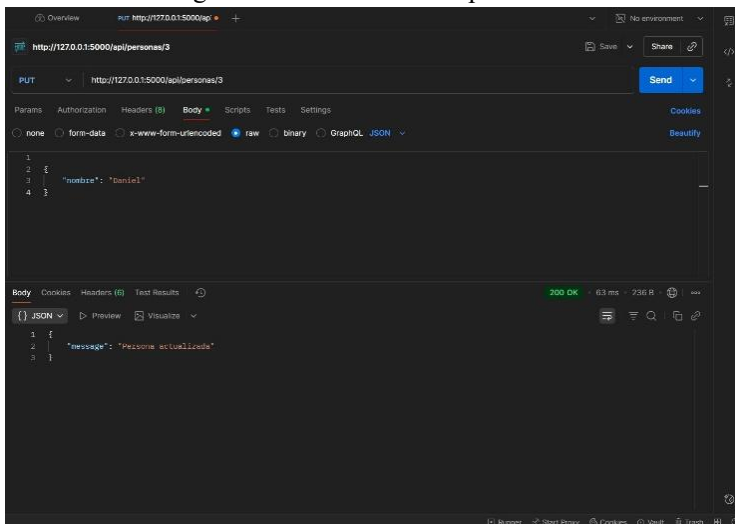
### 3. Capturas de Postman/ Swagger

Get /personas

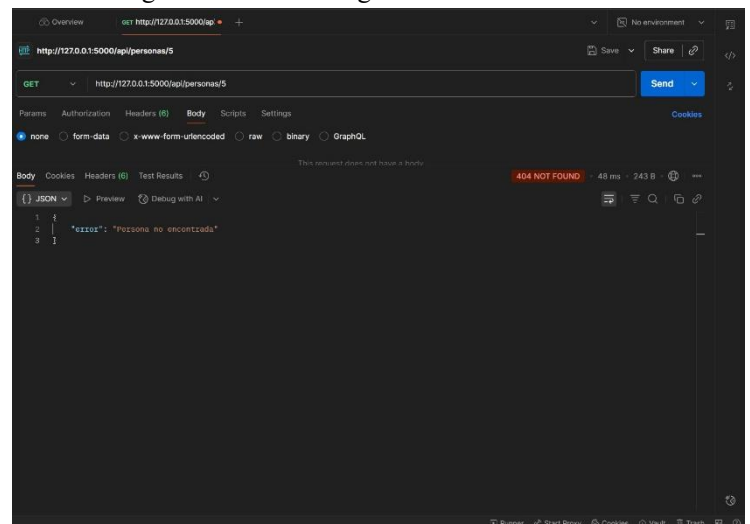


GET /personas/<id\_persona>

Código 200 al encontrar una persona:

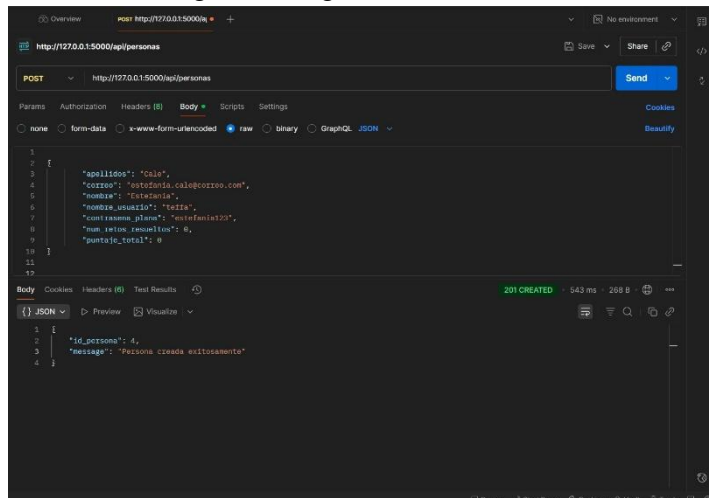


Código 404 cuando se ingrese un id incorrecto:

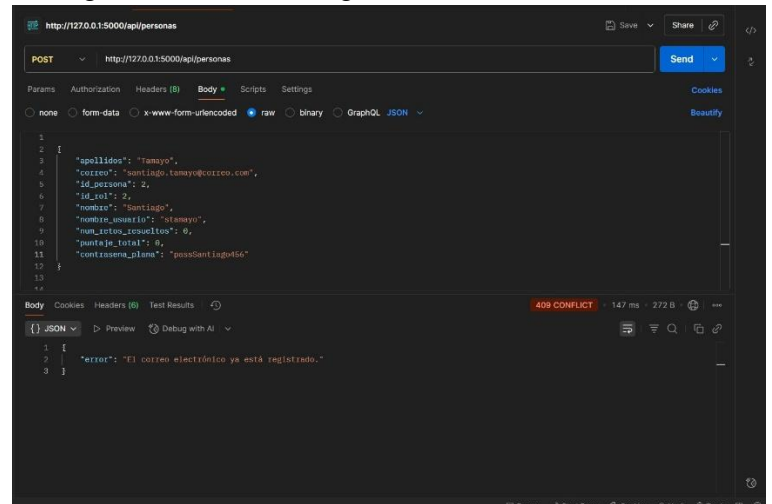


POST /personas

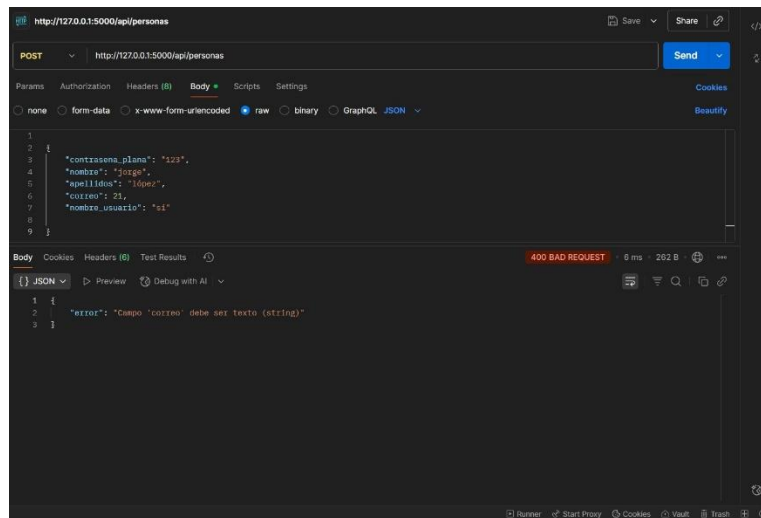
Código 201 al ingresar un usuario a la BD:



Código 409 al violar la integridad de los datos:

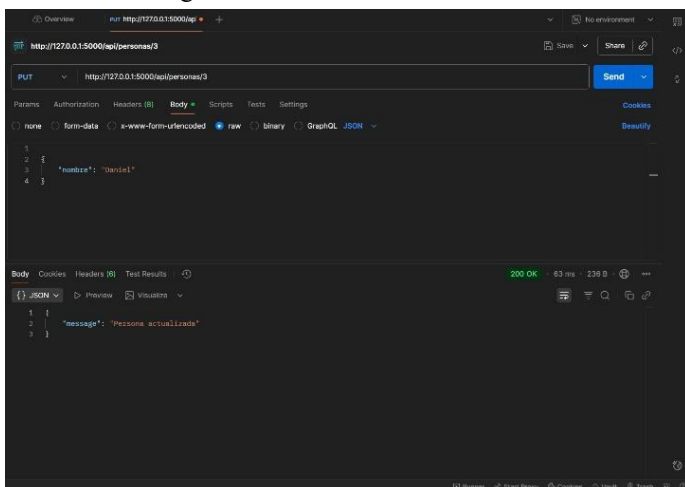


Datos erróneos al ingresar un usuario:

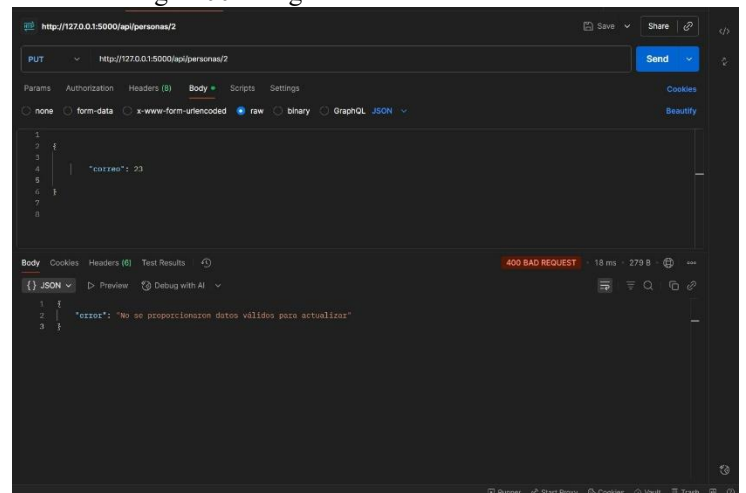


*PUT /personas/<id\_persona>*

Código 200 al actualizar el dato de un usuario:

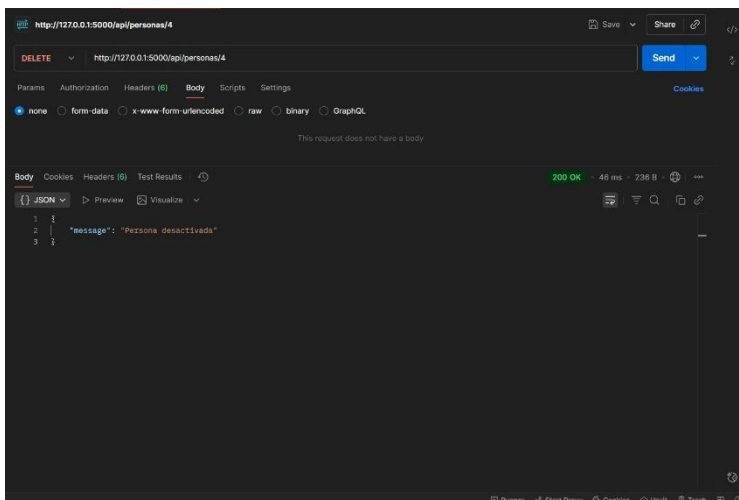


Código 400 al ingresar un dato erróneo:

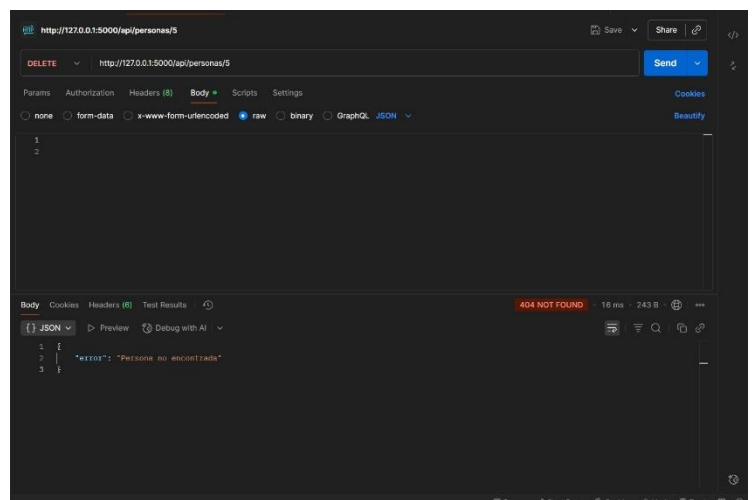


*DELETE /personas/<id\_persona>*

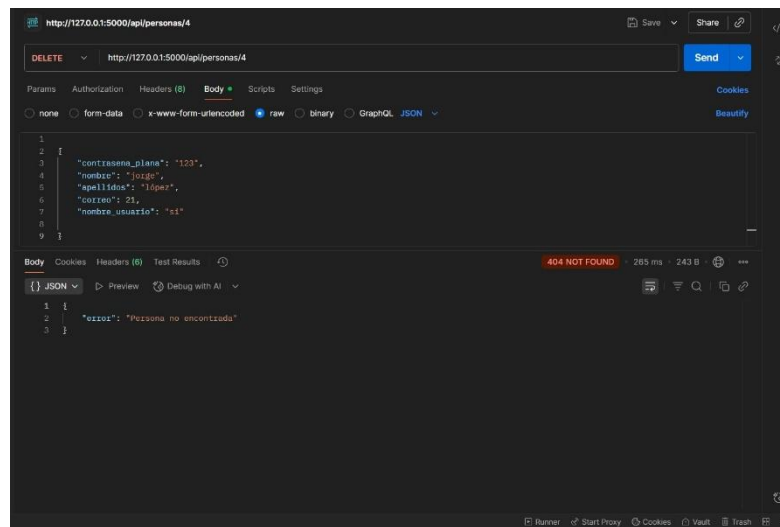
Desactivación de un usuario activo:



Intento de desactivar un usuario inexistente:



Error al desactivar una cuenta inactiva:



#### 4. Referencias

Repositorio: <https://github.com/SantiagoTamayo2002/codium>