

### **Práctica 3: API RESTful con PostgreSQL y Flask**

LUIS FELIPE MONGUI 20231020039

PROGAMACIÓN AVANZADA GRUPO 20-084

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C

24-06-2025

# 1. Introducción

El objetivo de esta práctica es diseñar e implementar una API RESTful para la gestión de tareas (CRUD) migrando el almacenamiento en memoria a una base de datos PostgreSQL. Se emplea Flask como framework backend y SQLAlchemy como ORM para asegurar una comunicación robusta entre aplicación y base de datos.

---

## 2. Desarrollo de la solución

### 2.1 Estructura del proyecto

- `app.py`: único módulo que contiene la configuración, definición de modelo y rutas.

### 2.2 Configuración de la base de datos

- Creación de la base de datos `tareasdb` y el usuario en PostgreSQL.
- Uso de `python-dotenv` para cargar credenciales desde `.env`.
- Configuración de `SQLALCHEMY_DATABASE_URI` en `app.config`.

### 2.3 Definición del modelo

```
class Tarea(db.Model):
    __tablename__ = 'tareas'
    id = db.Column(db.Integer, primary_key=True)
    nombre = db.Column(db.String(200), nullable=False)
    hecha = db.Column(db.Boolean, default=False)
```

### 2.4 Rutas y lógica CRUD

- **GET** `/tareas`: obtiene todas las tareas.
- **POST** `/tareas`: crea una nueva tarea.
- **GET** `/tareas/<id>`: obtiene tarea por ID (404 si no existe).
- **PUT** `/tareas/<id>`: actualiza campos `nombre` y `hecha`.
- **DELETE** `/tareas/<id>`: elimina la tarea.

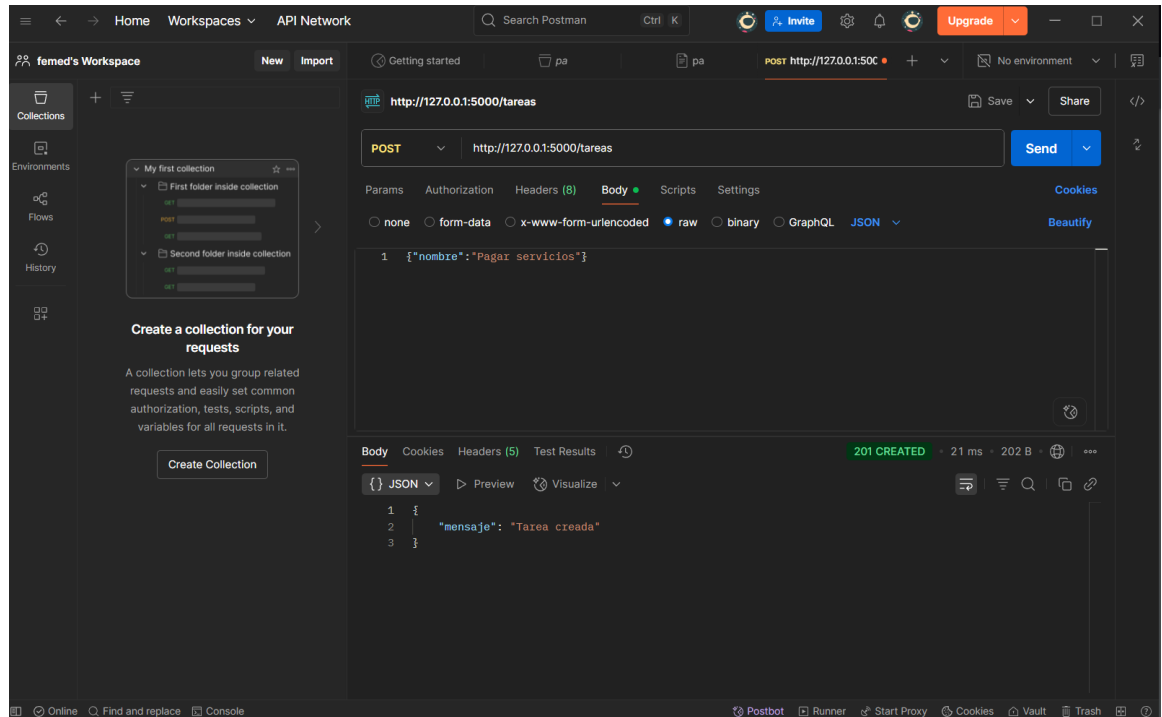
### 2.5 Rutas adicionales

- **GET** `/tareas/completadas`: filtra tareas con `hecha = true`.
  - **GET** `/tareas/pendientes`: filtra tareas con `hecha = false`.
  - **GET** `/tareas/buscar/<palabra>`: busca por texto en `nombre` (uso de `ilike`).
-

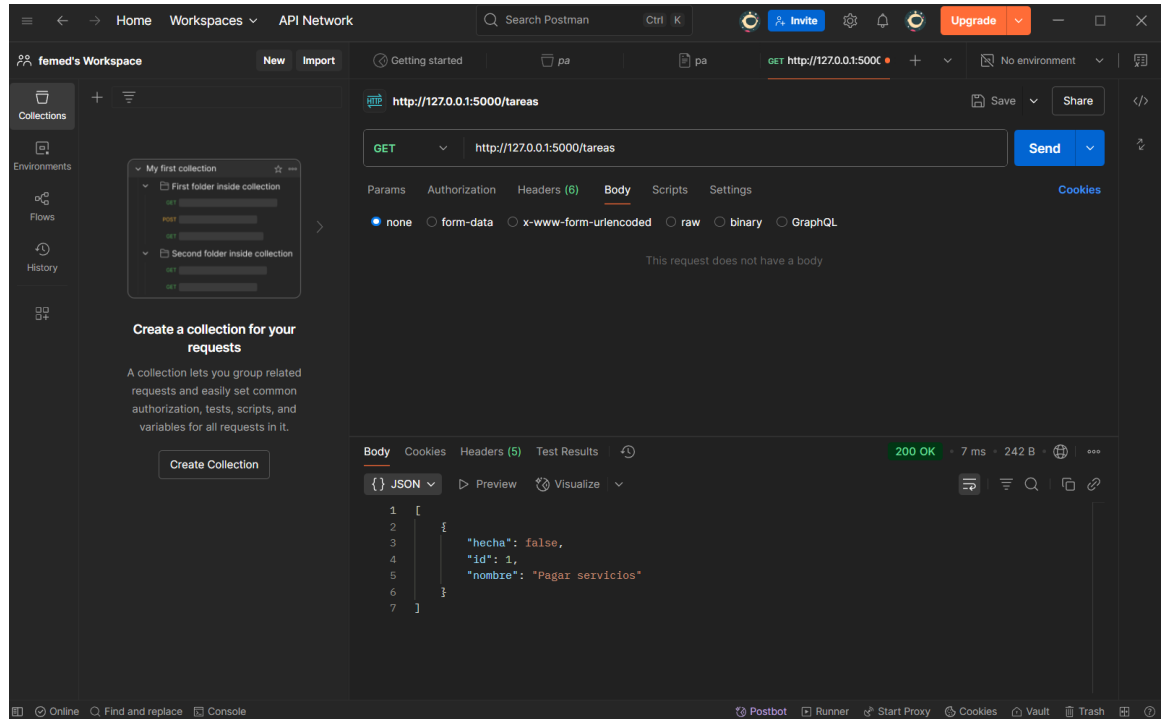
### 3. Resultados y pruebas

Se validó cada endpoint usando Postman con los siguientes resultados:

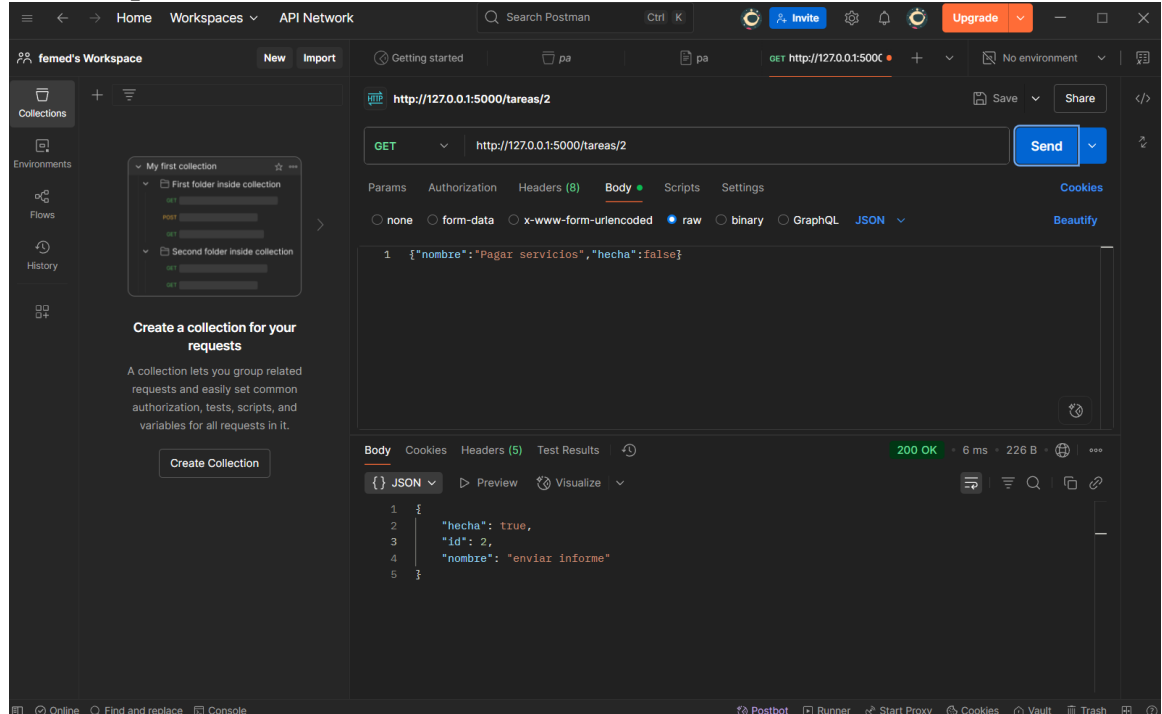
1. **Crear tarea** (POST /tareas): retorna 201 Created y mensaje de confirmación.



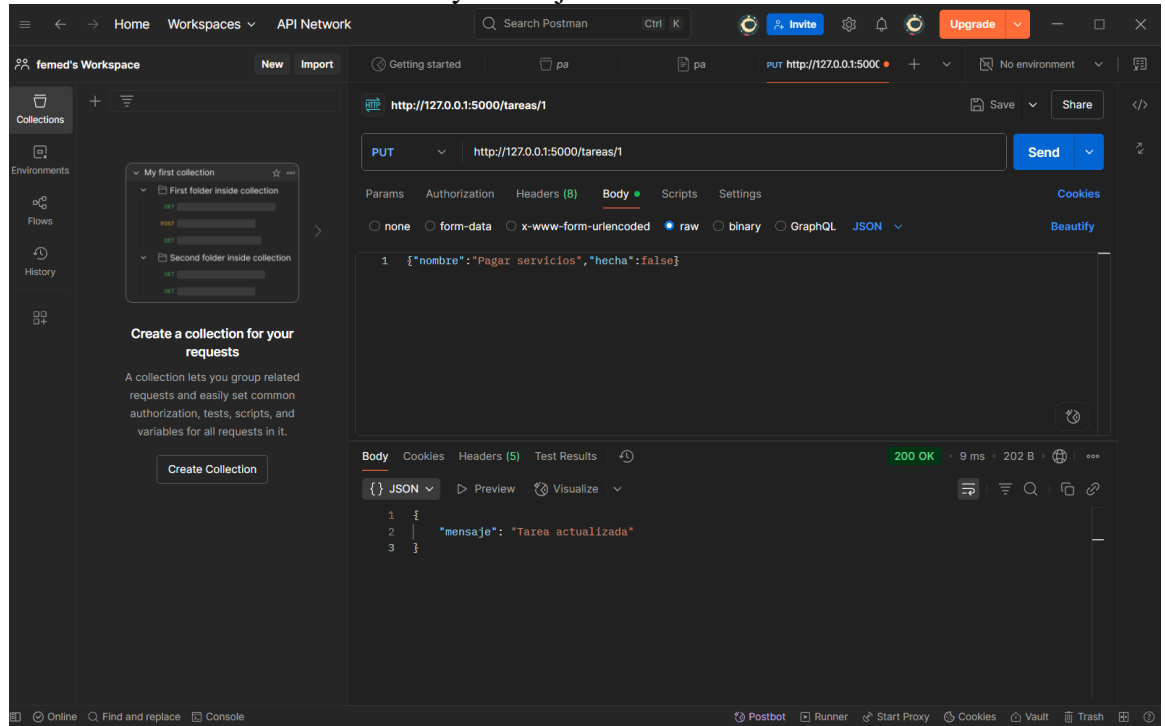
2. **Listar tareas** (GET /tareas): retorna 200 OK con array de objetos.



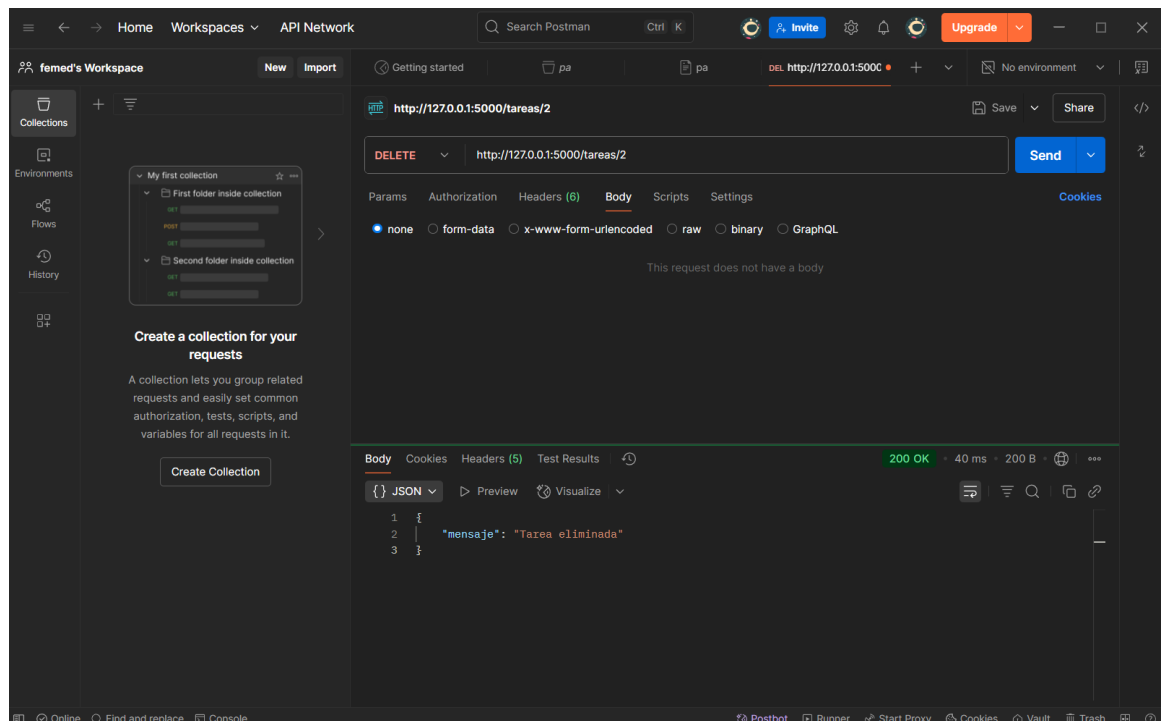
3. **Obtener por ID**: retorna 200 OK o 404 Not Found si no existe.



4. **Actualizar tarea:** retorna 200 OK y mensaje de actualización.



5. **Eliminar tarea:** retorna 200 OK y mensaje de eliminación.



6. **Filtros** (completadas, pendientes, buscar): funcionan según lo esperado.

## Completadas

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar is visible with a collection named 'My first collection'. The main workspace displays a GET request to 'http://127.0.0.1:5000/areas/completadas'. The request is successful, returning a 200 OK status. The response body is a JSON array with two objects. The first object has 'hecha' as true, 'id' as 4, and 'nombre' as 'Comprar pan'. The second object is partially visible.

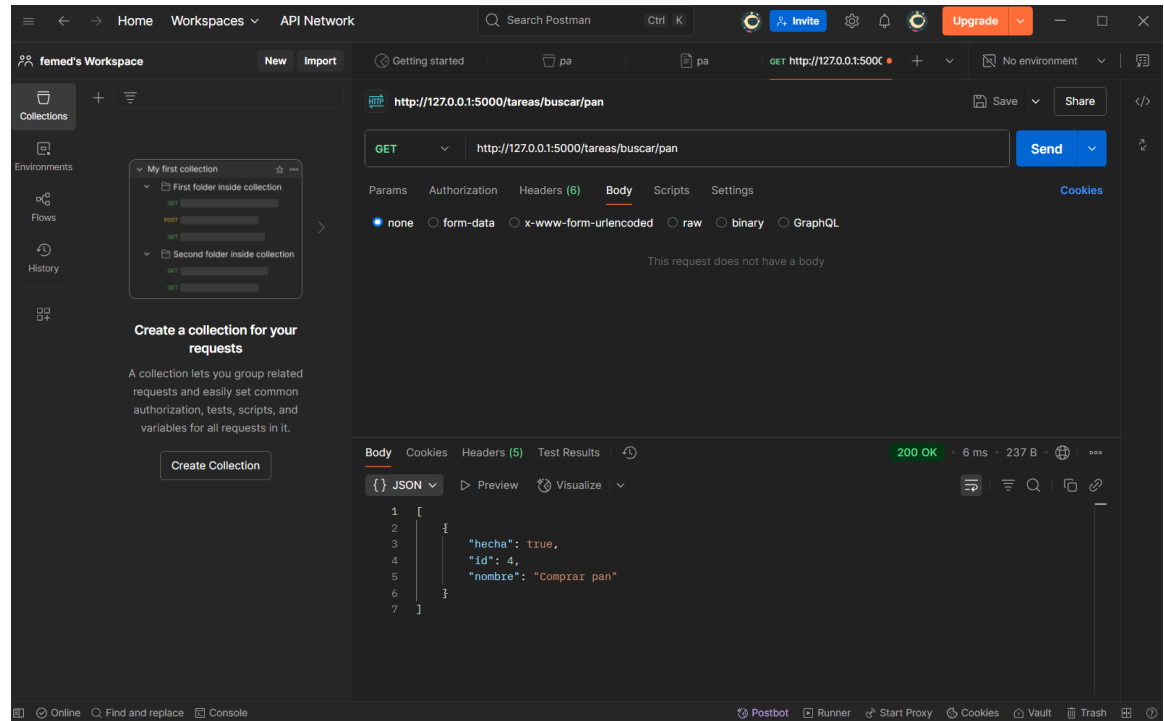
```
1 [
2   {
3     "hecha": true,
4     "id": 4,
5     "nombre": "Comprar pan"
6   }
7 ]
```

## Pendientes

The screenshot shows the Postman application interface. On the left, the 'Collections' sidebar is visible with a collection named 'My first collection'. The main workspace displays a GET request to 'http://127.0.0.1:5000/areas/pendientes'. The request is successful, returning a 200 OK status. The response body is a JSON array with one object. The object has 'hecha' as false, 'id' as 1, and 'nombre' as 'Pagar servicios'.

```
1 [
2   {
3     "hecha": false,
4     "id": 1,
5     "nombre": "Pagar servicios"
6   }
7 ]
```

## Palabra clave



---

## 4. Dificultades y soluciones

- **Conexión a PostgreSQL:** ajustes en URI y permisos de usuario.
- **Carga de variables de entorno:** instalación y configuración de `python-dotenv`.
- **Filtros `ilike`:** sintaxis de comodines `%` en consultas.

---

## 5. Conclusiones

- Comprensión del ciclo CRUD en Flask con SQLAlchemy.
- Ventajas de persistir datos en PostgreSQL y realizar consultas avanzadas.
- Recomendaciones: modularizar con Blueprints, integrar autenticación JWT y documentar con Swagger.

## 6. Carpeta compartida

<https://github.com/femd10/practica3pa.git>