ACTIVIDAD I

GESTION DE TAREAS – API RESTFUL

CRISTIAN DAVID MEDINA HERNÁNDEZ | CÓDIGO: 90108

ELECTIVA II

PROFESOR: DIEGO FERNANDO ZÁRATE PINEDA

UNIVERSIDAD INCCA DE COLOMBIA

INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

20 ENE 2026

Índice

Introducción

Este documento describe el desarrollo de una interfaz de programación de aplicaciones (RESTful API) para la gestión de tareas personales, implementada en ASP.NET Core conforme a los principios de arquitectura REST. La solución permite a los usuarios crear, consultar, editar, eliminar y marcar tareas como completadas, utilizando almacenamiento en memoria mediante una lista estática, sin conexión a base de datos relacional.

La API fue diseñada para cumplir con una historia de usuario específica: “Como usuario de un sistema de gestión de tareas, quiero poder gestionar mis tareas a través de una API RESTful para poder realizar operaciones como crear, ver, editar y eliminar mis tareas”.

Implementación

El modelo de datos se basa en la clase `TaskModel`, que incluye las propiedades requeridas: `Id` (identificador único), `Title` (título), `Description` (descripción) y `IsCompleted` (estado booleano). El almacenamiento se simula mediante una lista estática `List<TaskModel>` declarada dentro del controlador, lo que permite persistencia temporal durante la ejecución del proceso.

Los endpoints implementados son los siguientes:

- POST `/api/tareas`: crea una nueva tarea. Devuelve 201 Created si es exitosa o 400 Bad Request si falta el título.
- GET `/api/tareas`: devuelve todas las tareas con estado 200 OK.
- PUT `/api/tareas/{id}`: actualiza título y descripción. Devuelve 200 OK o 404 Not Found si el ID no existe.
- DELETE `/api/tareas/{id}`: elimina una tarea. Devuelve 204 No Content o 404 Not Found.
- PATCH `/api/tareas/{id}/completar`: marca una tarea como completada. Devuelve 200 OK o 404 Not Found.

La API respeta los códigos de estado HTTP semánticos, facilitando la interoperabilidad con clientes como Postman.

Pruebas con Postman

A continuación se lista una serie de ejemplos correspondientes a posibles solicitudes y respuestas al utilizar la herramieta postman.

1. Crear tarea (POST /api/tareas)
   Cuerpo:
   ```
   {
       "title": "Preparar informe",
       "description": "Redactar conclusión y revisar referencias"
   }
   ```
   Respuesta esperada (201 Created):
   ```
   {
       "id": 1, "title": "Preparar informe",
       "description": "Redactar conclusión y revisar referencias",
       "isCompleted": false
   }
   ```
   Si falta el título (400 Bad Request)

2. Listar tareas (GET /api/tareas)
   Cuerpo: vacío
   Respuesta esperada (200 OK):
   ```
   [
       {
           "id": 1,
           "title": "Preparar informe",
           "description": "Redactar conclusión y revisar referencias",
           "isCompleted": false
       }
   ]
   ```

3. Editar tarea (PUT /api/tareas/1)
   Cuerpo:
   {
   "title": "Preparar informe final",
   "description": "Incluir gráficos y enviar al tutor"
   }
   Respuesta esperada (200 OK)
   Si “id” no existe (404 Not Found)

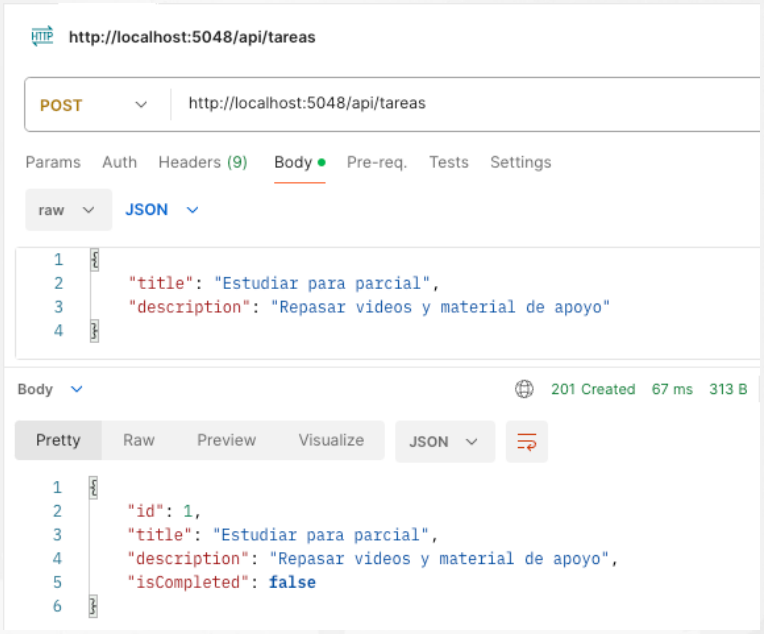
4. Marcar una tarea como completada (PATCH /api/tareas/1/completar)
   Cuerpo: vacío
   Respuesta esperada (200 OK):
   {
   "id": 1,
   "title": "Preparar informe final",
   "description": "Incluir gráficos y enviar al tutor",
   "isCompleted": true
   }
   Si “id” no existe (404 Not Found)

5. Eliminar tarea (DELETE /api/tareas/1)
   Cuerpo: vacío
   Respuesta esperada (204 No Content)
   Si “id” no existe (404 Not Found)

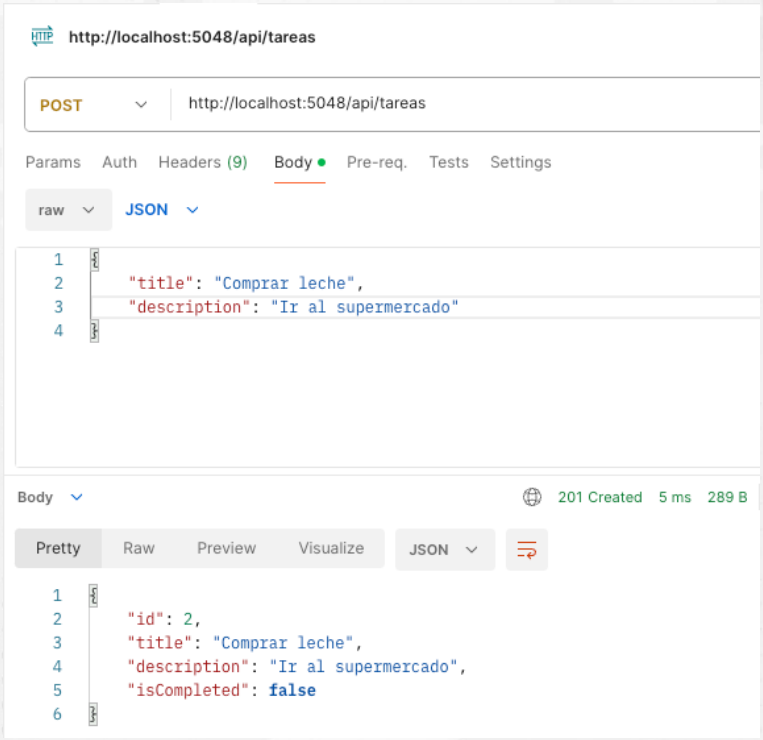
Se realizaron pruebas exhaustivas utilizando Postman para validar cada endpoint. Las pruebas incluyeron casos de éxito y manejo de errores (por ejemplo, enviar un título vacío o un ID inexistente). Todas las respuestas incluyeron los códigos de estado correctos y cuerpos JSON coherentes con el modelo `TaskModel`.

Las capturas de pantalla de las pruebas se incluyen como evidencia en la sección siguiente. Cada imagen muestra la URL, método HTTP, cuerpo de la solicitud (si aplica), código de estado y respuesta JSON.

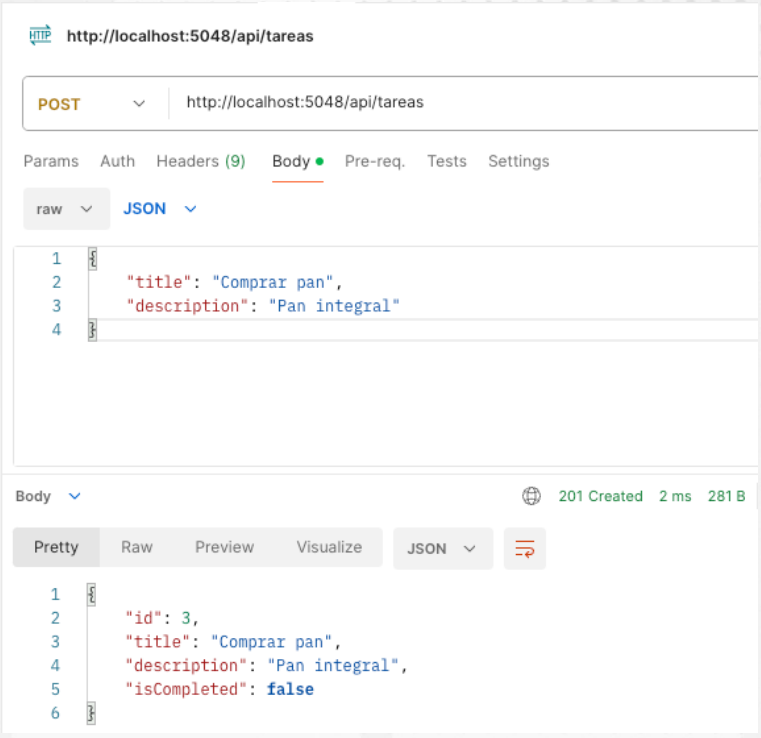
Capturas de Pantalla



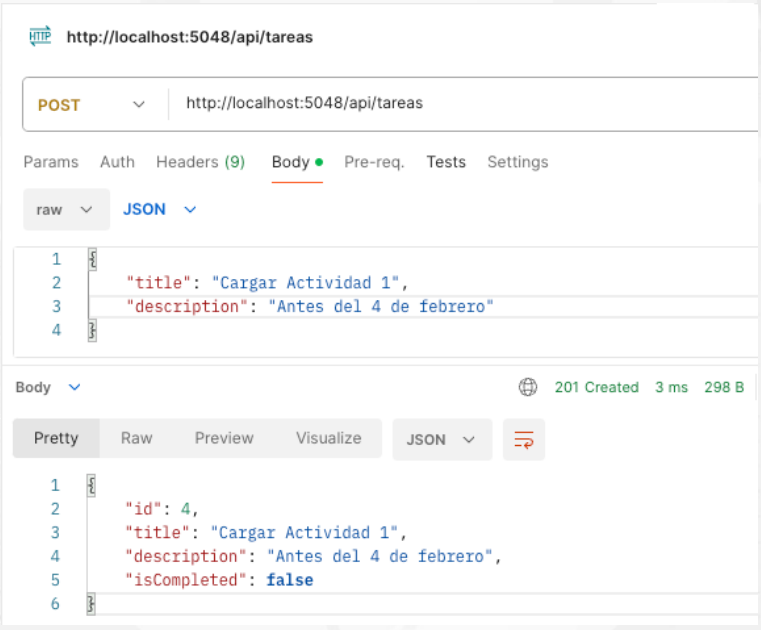
Captura 1 - Prueba POST: creación de tarea



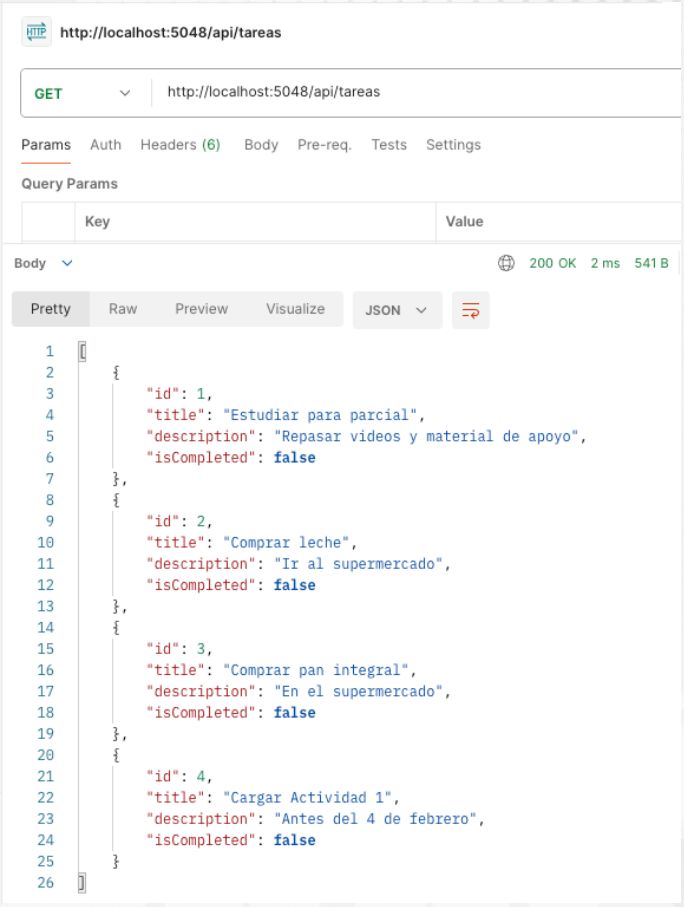
Captura 2 - Prueba POST: creación de tarea



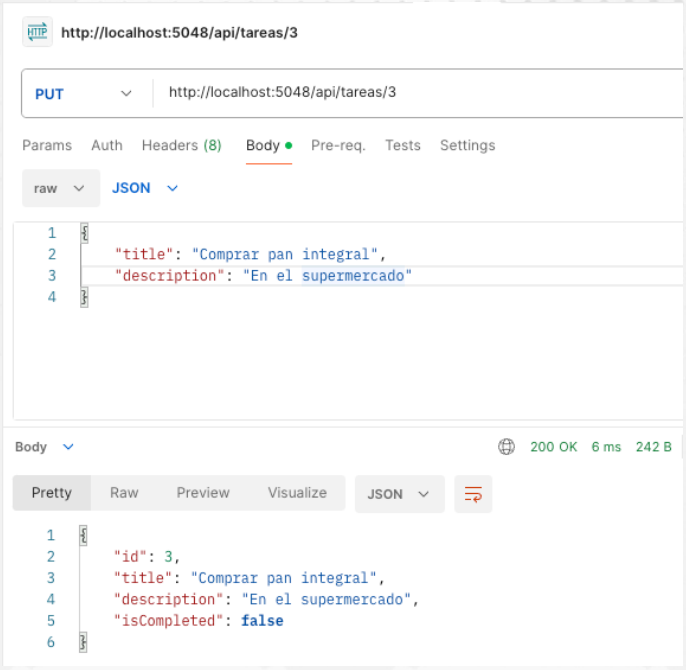
Captura 3 - Prueba POST: creación de tarea



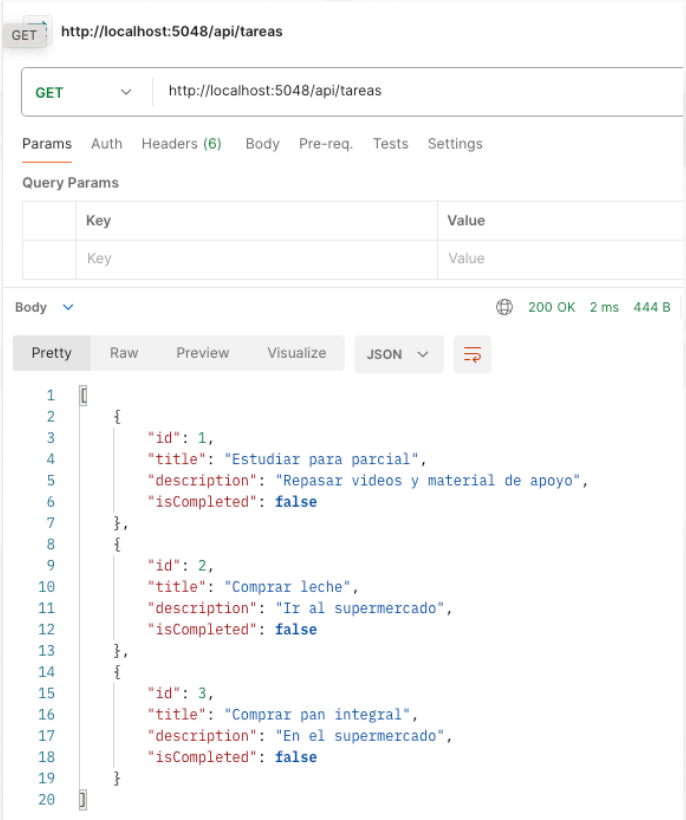
Captura 4 - Prueba POST: creación de tarea



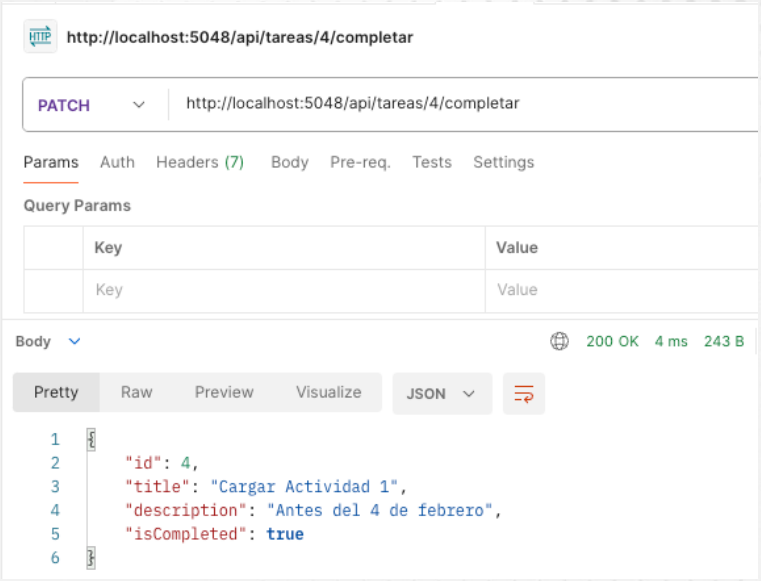
Captura 5 - Prueba GET: listado de tareas



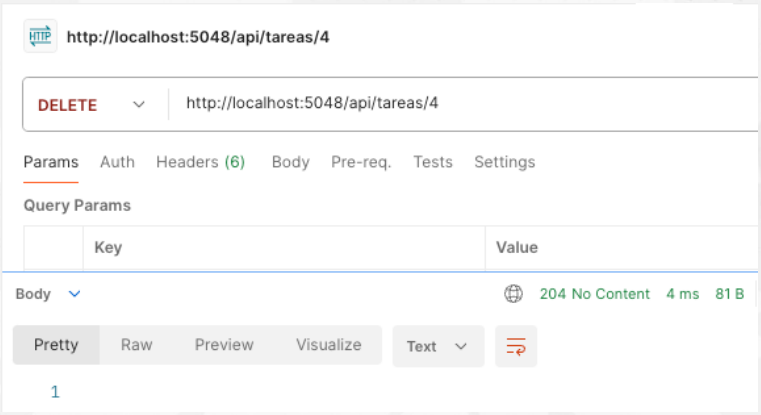
Captura 6 - Prueba PUT: edición de tarea



Captura 7 - Prueba GET: listar tareas (comprobar tarea modificada)



Captura 8 - Prueba PATCH: completar tarea



Captura 9 - Prueba DELETE: eliminar tarea

http://localhost:5048/api/tareas

GET http://localhost:5048/api/tareas

Params Auth Headers (6) Body Pre-req. Tests Settings

Query Params

Key	Value

Body 200 OK 1 ms 444 B

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "title": "Estudiar para parcial",
        "description": "Repasar videos y material de apoyo",
        "isCompleted": false
    },
    {
        "id": 2,
        "title": "Comprar leche",
        "description": "Ir al supermercado",
        "isCompleted": false
    },
    {
        "id": 3,
        "title": "Comprar pan integral",
        "description": "En el supermercado",
        "isCompleted": false
    }
]
```


Captura 10 - Prueba GET: listar tareas (comprobar tarea eliminada)

http://localhost:5048/api/tareas/4

GET http://localhost:5048/api/tareas/4

Params Auth Headers (6) Body Pre-req. Tests Settings

Query Params

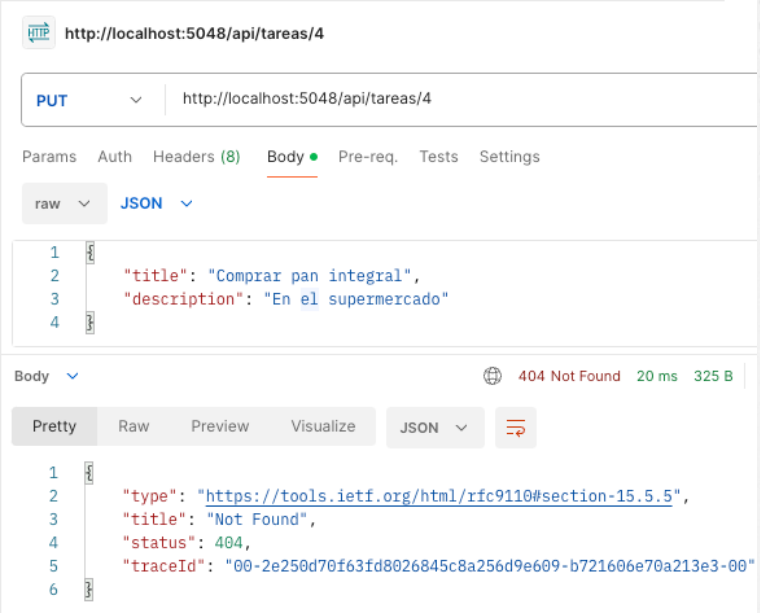
Key	Value

Body 405 Method Not Allowed 2 ms 128 B

Pretty Raw Preview Visualize Text

```
1
```

Captura 11 - Prueba Code 405: Metodo no permitido



Captura 12 - Prueba Cod 404: Endpoint inexistente (Not Found)

Conclusión

La API desarrollada cumple con todos los requisitos funcionales y técnicos establecidos en la actividad. Su diseño sigue buenas prácticas de desarrollo en ASP.NET Core, utiliza almacenamiento en memoria de forma eficiente y responde con códigos de estado HTTP adecuados. Las pruebas en Postman confirman su correcto funcionamiento.

Apendice

- Repositorio público: https://github.com/cdmedinah-unincca/TaskApi
- README.md: disponible en la raíz del repositorio
- Colección de Postman: exportada como `TaskApi.postman_collection.json`