Informe de Implementación: API REST de Gestión de Tareas

Fecha: 17 de enero de 2026

Materia: Desarrollo de Aplicaciones Web

Entregable: Actividad 1 - API REST

Estudiante: José Daniel Avendaño Morales

Docente: Diego Fernando Zárate Pineda

Programa: Ingeniería de Sistemas

1. Introducción e Historia de Usuario

El presente proyecto detalla el desarrollo de una interfaz de programación de aplicaciones (API) bajo el estilo arquitectónico REST para la administración de tareas pendientes.

Historia de Usuario:

"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".

2. Arquitectura y Stack Tecnológico

- **Framework:** ASP.NET Core (Web API).
- **Lenguaje:** C# 12.0.
- **Servidor:** Kestrel (Puerto local 5063).
- **Persistencia:** Almacenamiento volátil en memoria mediante una colección List<TaskModel> estática.
- **Cliente de Pruebas:** Postman (v10+).

3. Implementación del Modelo y Controlador

A. Modelo de Datos (TaskModel.cs)

Se definió un modelo robusto con inicialización de cadenas para evitar advertencias de nulabilidad durante la compilación.

- **Id:** Identificador entero auto-incremental.
- **Title:** Nombre de la tarea.

- **Description:** Detalle de la actividad.
- **IsCompleted:** Estado lógico de la tarea.

B. Controlador de Tareas (TareasController.cs)

El controlador utiliza enrutamiento basado en atributos ([Route("api/tareas")]). Se implementaron los siguientes verbos HTTP para garantizar un CRUD completo:

1. **POST:** Recibe un JSON, asigna un ID único y almacena la tarea.
2. **GET:** Recupera la colección completa de tareas para su visualización.
3. **PUT:** Permite la edición integral de los campos Title y Description de una tarea existente.
4. **PATCH:** Realiza una actualización parcial para marcar una tarea como completada (IsCompleted = true).
5. **DELETE:** Elimina un registro de la memoria basado en el ID proporcionado.

4. Resolución de Conflictos y Depuración

Durante la puesta en marcha, se identificaron y resolvieron los siguientes obstáculos técnicos:

- **Corrección de Error 404 (Not Found):** Se detectó que el servidor no mapeaba las rutas del controlador debido a una configuración de "Minimal API" por defecto en Program.cs. Se corrigió añadiendo los servicios builder.Services.AddControllers() y el middleware app.MapControllers().
- **Corrección de Error 405 (Method Not Allowed):** Se solventó en la petición PUT asegurando que el parámetro de ruta {id} en el decorador [HttpPut("{id}")] coincidiera con la variable del método, permitiendo que el servidor aceptara el verbo PUT en la URL específica.
- **Estrategia de Persistencia:** Para evitar que la lista se reiniciara en cada petición (dado el ciclo de vida transitorio de los controladores), se utilizó el modificador static en la lista de tareas.

---

5. Guía de Validación (Manual de Postman)

Para replicar las pruebas satisfactorias, utilice la siguiente configuración:

Método	Endpoint	Headers	Body (Raw JSON)	Resultado
POST	http://localhost:5063/api/tareas	Content-Type: application/json	{"title": "Tarea 1", "description": "Prueba"}	201 Created
GET	http://localhost:5063/api/tareas	N/A	N/A	200 OK
PUT	http://localhost:5063/api/tareas/1	Content-Type: application/json	{"title": "Editada", "description": "Nueva"}	200 OK
PATCH	http://localhost:5063/api/tareas/1/completar	N/A	N/A	200 OK
DELETE	http://localhost:5063/api/tareas/1	N/A	N/A	204 No Content

README.md (Documentación del Repositorio)

Markdown

 # Gestión de Tareas API - ASP.NET Core

Proyecto desarrollado para la administración de tareas mediante una arquitectura RESTful.

Requisitos

- .NET 8.0 SDK o superior.

- Postman para la ejecución de pruebas.

Configuración y Ejecución

1. Abra una terminal en la raíz del proyecto.

2. Ejecute el comando:

```bash
dotnet run
```

3. **El servidor iniciará en:** http://localhost:5063.

Pruebas de Endpoints

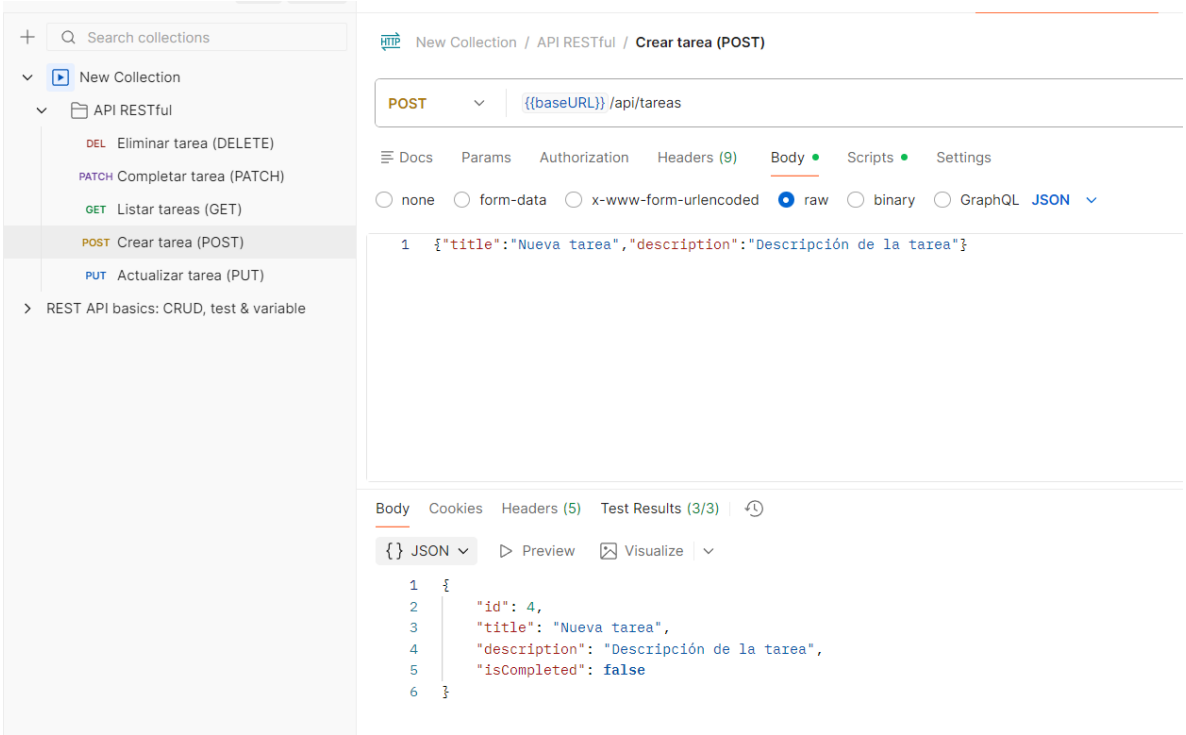
- **POST /api/tareas:** Crea una tarea (requiere JSON con title y description).
- **GET /api/tareas:** Obtiene todas las tareas en memoria.
- **PUT /api/tareas/{id}:** Actualiza los datos de una tarea.
- **PATCH /api/tareas/{id}/completar:** Marca una tarea como finalizada.
- **DELETE /api/tareas/{id}:** Elimina la tarea especificada.

Nota: *Al usar almacenamiento en memoria (static list), los datos se pierden al detener el proceso del servidor.*

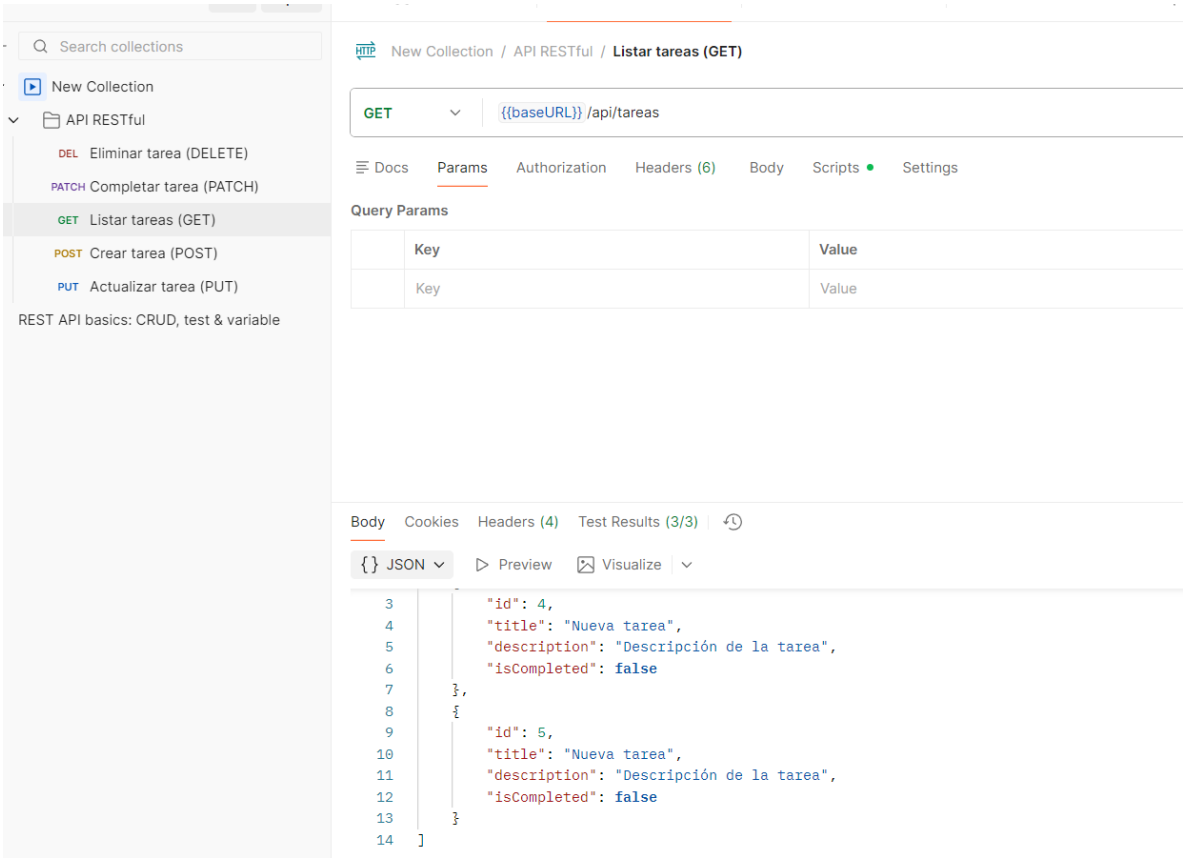
ANEXOS PRUEBAS

- **Evidencias**: Se adjunta la colección de Postman exportada con los casos de prueba mencionados

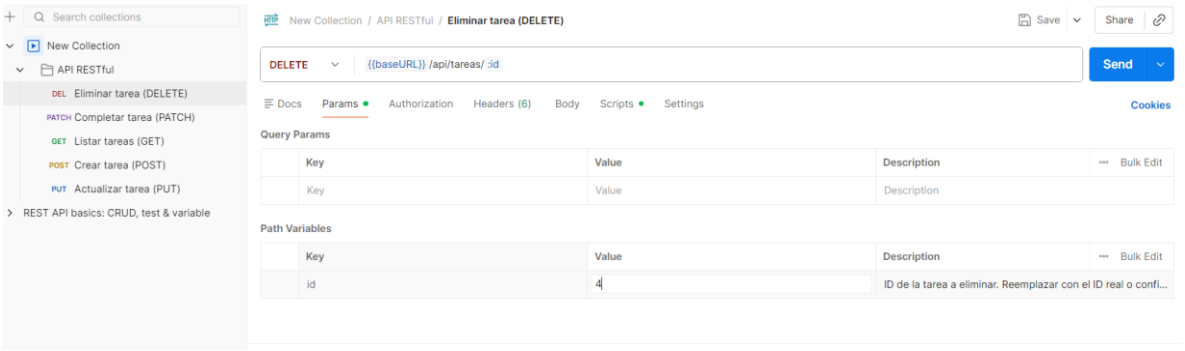
POST:

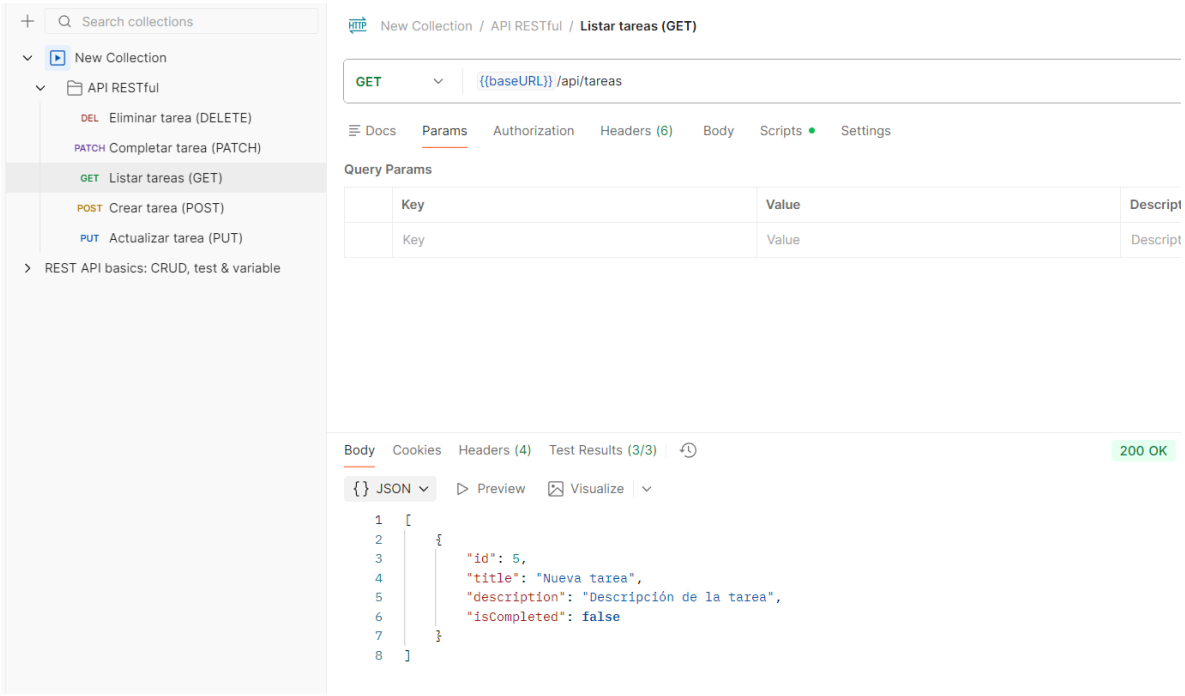


GET:

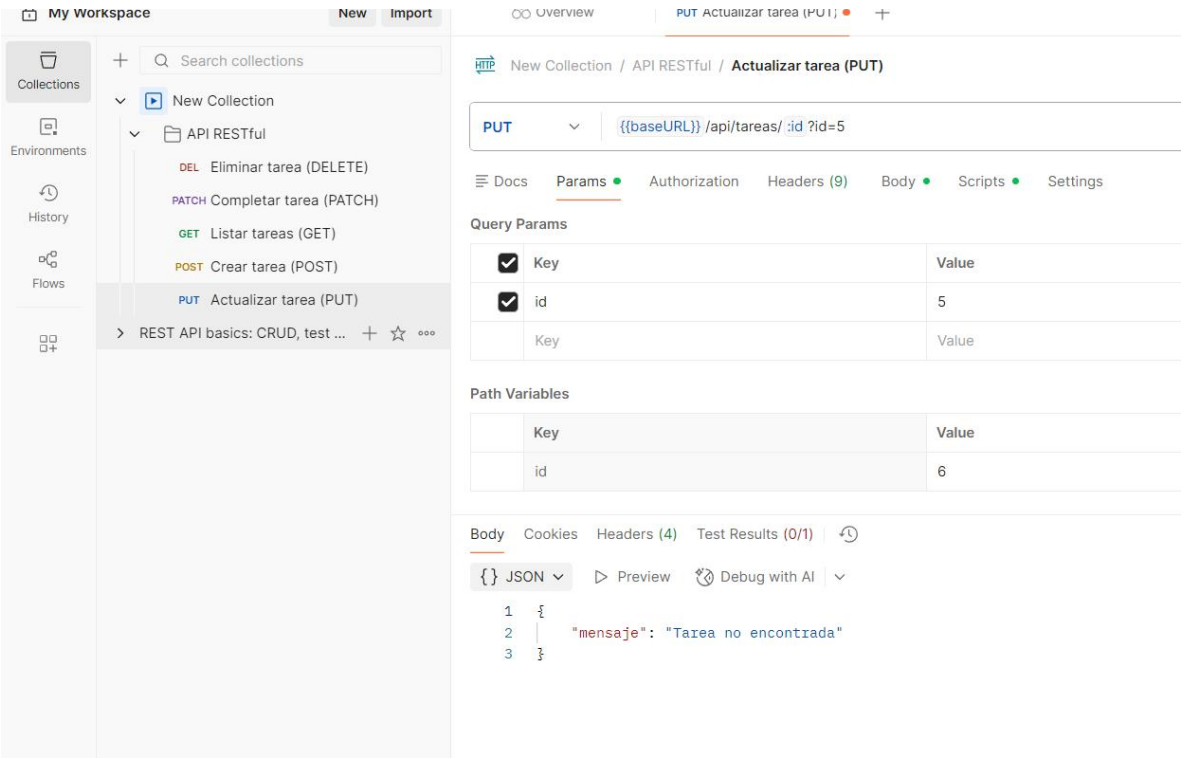

Search collections
New Collection
API RESTful
DEL Eliminar tarea (DELETE)
PATCH Completar tarea (PATCH)
GET Listar tareas (GET)
POST Crear tarea (POST)
PUT Actualizar tarea (PUT)
REST API basics: CRUD, test & variable
New Collection / API RESTful / Listar tareas (GET)
GET
{{baseURL}} /api/tareas
Docs
Params
Authorization
Headers (6)
Body
Scripts
Settings
Query Params
Key
Value
Key
Value
Body
Cookies
Headers (4)
Test Results (3/3)
JSON
Preview
Visualize
"id": 4,
"title": "Nueva tarea",
"description": "Descripción de la tarea",
"isCompleted": false
},
{
"id": 5,
"title": "Nueva tarea",
"description": "Descripción de la tarea",
"isCompleted": false
}
]

DELETE:

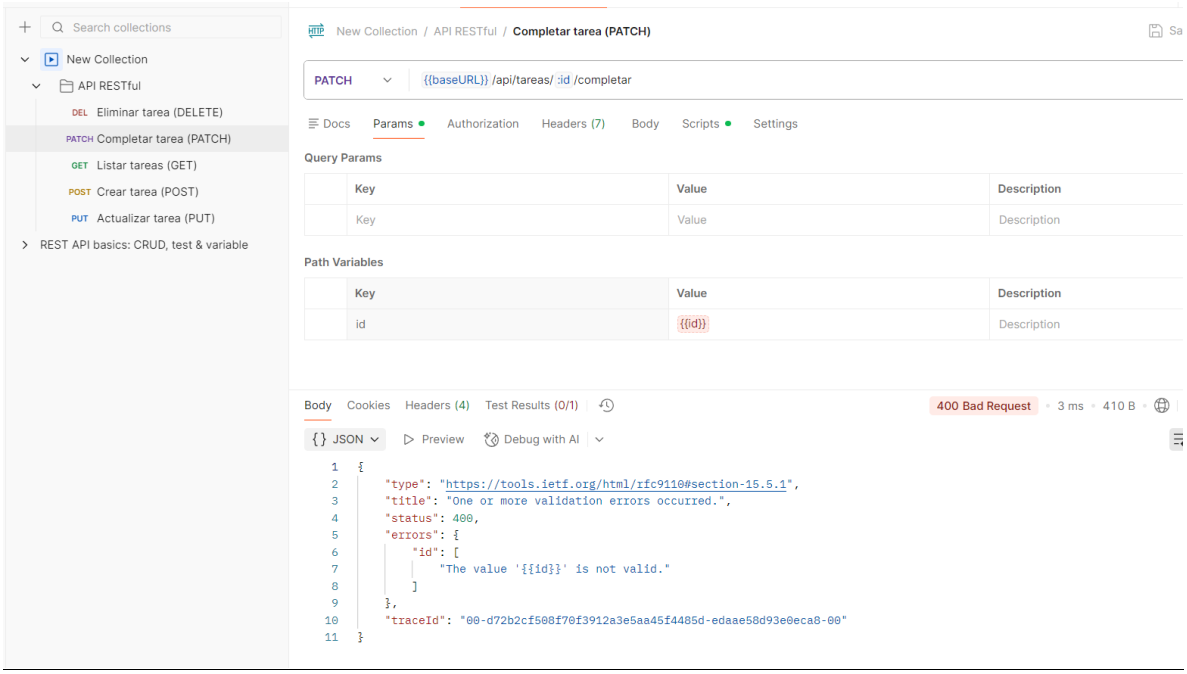

Search collections
New Collection
API RESTful
DEL Eliminar tarea (DELETE)
PATCH Completar tarea (PATCH)
GET Listar tareas (GET)
POST Crear tarea (POST)
PUT Actualizar tarea (PUT)
REST API basics: CRUD, test & variable
New Collection / API RESTful / Eliminar tarea (DELETE)
Save
Share
DELETE
{{baseURL}} /api/tareas/ :id
Send
Docs
Params
Authorization
Headers (6)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Key
Value
Description
Path Variables
Key
Value
Description
Bulk Edit
id
4
ID de la tarea a eliminar. Reemplazar con el ID real o confi...


Search collections
New Collection
API RESTful
DEL Eliminar tarea (DELETE)
PATCH Completar tarea (PATCH)
GET Listar tareas (GET)
POST Crear tarea (POST)
PUT Actualizar tarea (PUT)
REST API basics: CRUD, test & variable
New Collection / API RESTful / Listar tareas (GET)
GET
{{baseURL}} /api/tareas
Docs
Params
Authorization
Headers (6)
Body
Scripts
Settings
Query Params
Key
Value
Descript
Body
Cookies
Headers (4)
Test Results (3/3)
200 OK
JSON
Preview
Visualize
[
{
"id": 5,
"title": "Nueva tarea",
"description": "Descripción de la tarea",
"isCompleted": false
}
]

PUT:


My Workspace
New
Import
Overview
PUT Actualizar tarea (PUT)
Collections
Environments
History
Flows
Search collections
New Collection
API RESTful
DEL Eliminar tarea (DELETE)
PATCH Completar tarea (PATCH)
GET Listar tareas (GET)
POST Crear tarea (POST)
PUT Actualizar tarea (PUT)
REST API basics: CRUD, test ...
New Collection / API RESTful / Actualizar tarea (PUT)
PUT
{{baseURL}} /api/tareas/ :id ?id=5
Docs
Params
Authorization
Headers (9)
Body
Scripts
Settings
Query Params
Key
Value
id
5
Path Variables
Key
Value
id
6
Body
Cookies
Headers (4)
Test Results (0/1)
JSON
Preview
Debug with AI
{
"mensaje": "Tarea no encontrada"
}

PATCH:



Instrucciones para la Entrega

- **Código Fuente**: Actualizado en el repositorio público de Git con un README.md que detalla los cambios de la Actividad 1.

  Link: https://github.com/vallrack/apirestfullparte1

- **Documentación**: El presente archivo PDF resume todas las modificaciones técnicas.