

# Taller de sistemas de información .NET

## Tarea – 02\_A

**Autor: Lucas Techera**

**Docente: Gabriel Aramburu**

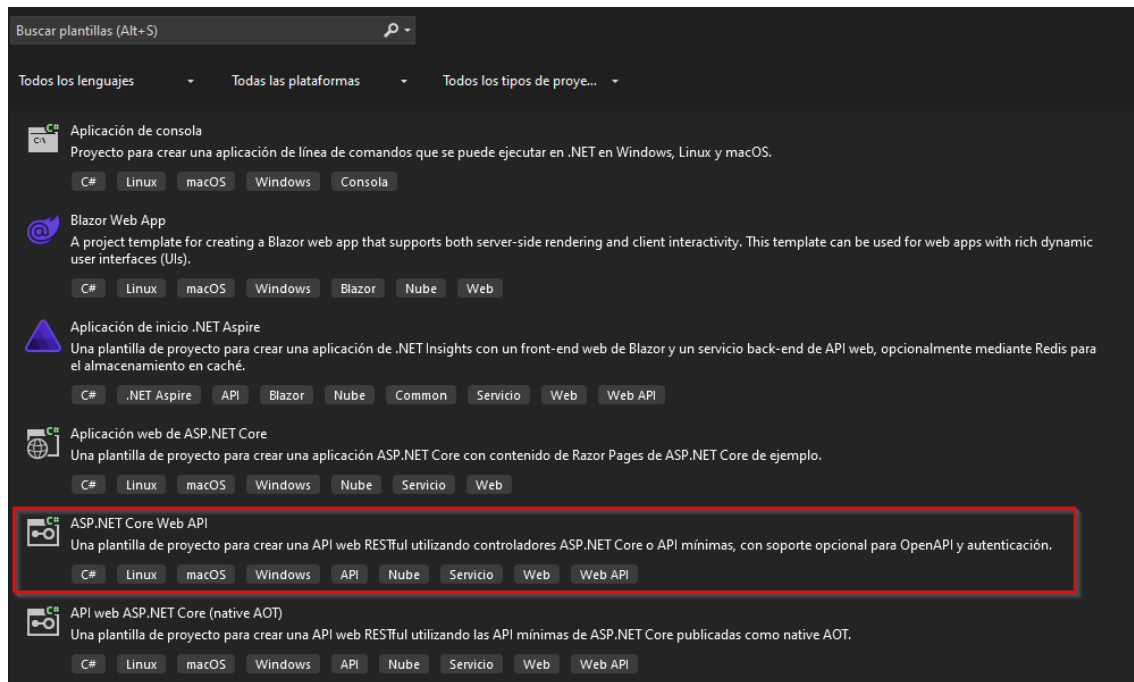
**CI: 5.295.981-3**

**Contacto: [fidel.techera@estudiantes.utec.edu.uy](mailto:fidel.techera@estudiantes.utec.edu.uy)**

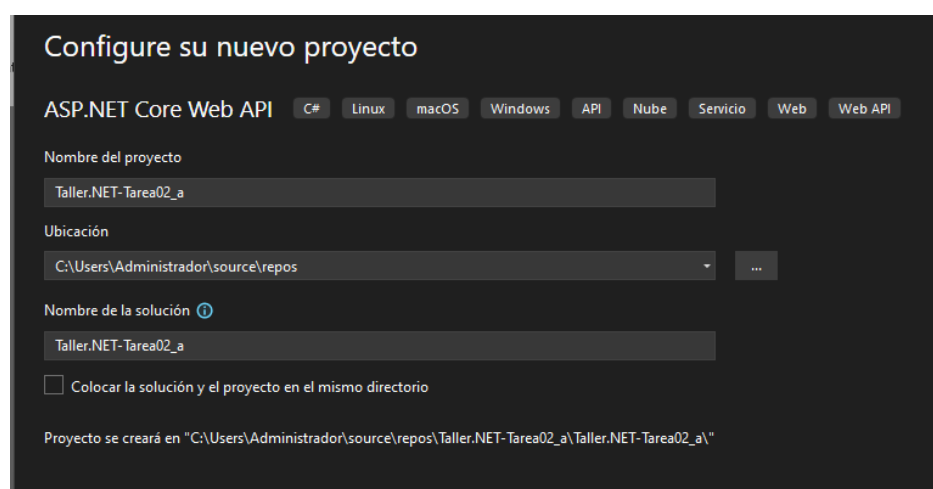
## Consigna

1. Crear un proyecto utilizando la plantilla correspondiente

Para crear el proyecto adecuadamente seleccionaremos una plantilla, como se muestra a continuación.



Luego tenemos que nombrar nuestro proyecto y seleccionar donde queremos almacenar el mismo.



Como ultimo paso debemos asegurarnos de seleccionar opciones de configuración del mismo, en este caso se utilizan las siguientes:



**Información adicional**

ASP.NET Core Web API C# Linux macOS Windows API Nube Servicio Web Web API

Framework ⓘ  
 .NET 8.0 (Compatibilidad a largo plazo)

Authentication de campo ⓘ  
 Ninguno

☒ Configurar para HTTPS ⓘ  
☐ Habilitar compatibilidad con el contenedor ⓘ

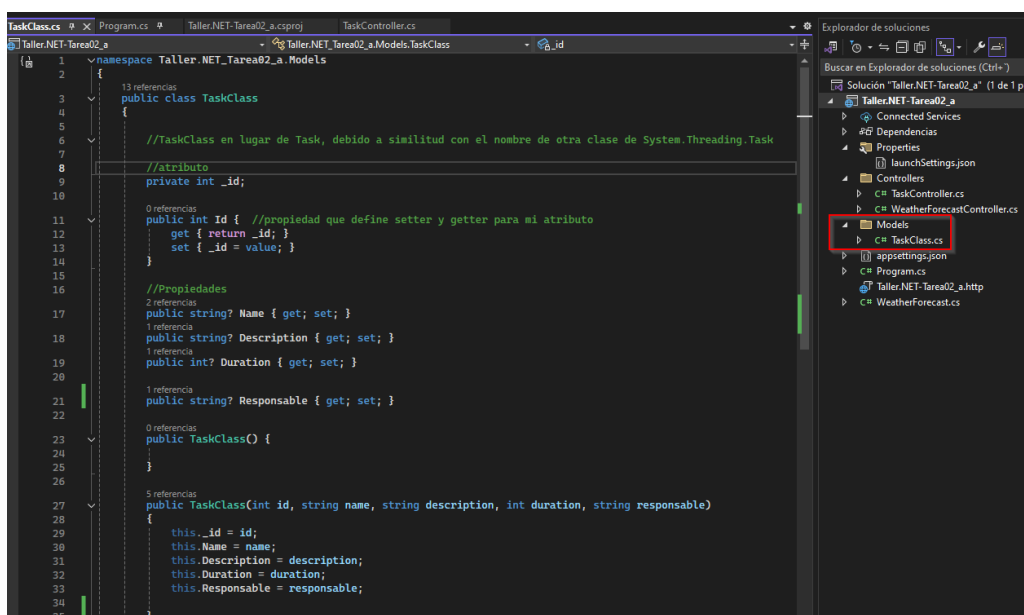
SO del contenedor ⓘ  
 Linux

Tipo de compilación de contenedor ⓘ  
 Dockerfile

☒ Habilitar compatibilidad con OpenAPI ⓘ  
☐ No usar instrucciones de nivel superior ⓘ  
☒ Utilizar controladores ⓘ  
☐ Inscribirse en la orquestación de .NET Aspire ⓘ

2. Implementar, dentro de la carpeta Models, una clase de dominio que represente una Tarea (id, nombre, desc, duración horas, responsable)

Se crea dentro de la carpeta Models una clase dominio llamada **TaskClass.cs**, la cual contiene una simple clase Tarea con la siguiente estructura:



```

1 namespace Taller.NET_Tarea02_a.Models
2 {
3     public class TaskClass
4     {
5         //TaskClass en lugar de Task, debido a similitud con el nombre de otra clase de System.Threading.Task
6
7         //atributo
8         private int _id;
9
10        0 referencias
11        public int Id { //propiedad que define setter y getter para mi atributo
12            get { return _id; }
13            set { _id = value; }
14        }
15
16        //Propiedades
17        2 referencias
18        public string? Name { get; set; }
19        1 referencia
20        public string? Description { get; set; }
21        1 referencia
22        public int? Duration { get; set; }
23
24        1 referencia
25        public string? Responsable { get; set; }
26
27        0 referencias
28        public TaskClass() {
29        }
30
31        5 referencias
32        public TaskClass(int id, string name, string description, int duration, string responsable)
33        {
34            this._id = id;
35            this.Name = name;
36            this.Description = description;
37            this.Duration = duration;
38            this.Responsable = responsable;
39        }
40    }
  
```

### 3. Incluir atributos de tipo string y entero

Se puede observar en la imagen del punto anterior, como se asignan atributos de tipo entero (**Id, Duration**) y de tipo string (**Name, Description, Responsable**)

### 4. Crear un nuevo controlador que exponga siguiendo los lineamientos de una API RESTful las siguientes operaciones:

Al estar implementada la persistencia en memoria se cargarán datos de prueba en una lista de tareas, como se muestra a continuación:

```
//atributos
private readonly ILogger<TaskController> _logger;
private IList<TaskClass> _taskList; //lista de tareas a manipular en memoria.

0 referencias
public TaskController(ILogger<TaskController> logger)
{
    this._logger = logger; //inyeccion instancia logger mediante el constructor.

    this._taskList = new List<TaskClass>(); //inicializo lista

    //carga datos de prueba en lista.
    //id, nombre, descripcion, duracion, responsable.

    this._taskList.Add(new TaskClass(0, "Tarea 1", "Mirar correos", 2, "Pepe"));
    this._taskList.Add(new TaskClass(1, "Tarea 2", "Hacer reporte", 2, "Milagros"));
    this._taskList.Add(new TaskClass(2, "Tarea 3", "Terminar maqueta", 2, "Felipe"));
    this._taskList.Add(new TaskClass(3, "Tarea 4", "Actualizar perfil", 2, "José"));
    this._taskList.Add(new TaskClass(4, "Tarea 5", "Chequear actualizaciones", 2, "Nicolás"));
}
```

También se puede observar el uso de un logger primitivo para loggear información básica cuando se invoque alguna de las operaciones.

A continuación, se mostrarán imágenes del código de los endpoints solicitados.

a. Insertar.

```

/// <summary>
/// Crear tarea
/// </summary>
/// <param name="task"> Utiliza los parametros del body para crear el objeto </param>
/// <returns>Una lista de tareas con la tarea agregada</returns>
/// <remarks>
/// Sample request:
///
/// {
///     "id": 5,
///     "name": "Revisar Tarea",
///     "description": "Tareas de API's en .NET",
///     "duration": 2,
///     "responsable": "Gabriel",
///     "creationDate": "2024-04-30"
/// }
/// </remarks>
/// <response code="200"> Retorna una lista de tarea, con la nueva tarea</response>
/// <response code="500"> Si no se pudo crear la tarea </response>
[HttpPost]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
0 referencias
public ActionResult newTask([FromBody] TaskClass task) //Obtengo la data del body. No es necesario FromBody en caso de tener: [ApiController]
{
    try
    {
        _taskList.Add(task);
        _logger.LogInformation("Añadiendo nueva tarea: " + task.Name + " a la lista de Tareas");

        return Ok(_taskList.ToList());
    }
    catch (Exception ex)
    {
        _logger.LogError("Error al añadir la tarea: " + ex.Message);
        return StatusCode(500, "Se produjo un error al añadir la tarea.");
    }
}

```

b. Eliminar

```

/// <summary>
/// Eliminar Tarea
/// </summary>
/// <param name="id"> Utiliza el identificador para buscar y eliminar la tarea </param>
/// <returns> Devuelve una lista de tareas, sin la tarea eliminada </returns>
/// <response code="200"> Retorna una lista de tareas con la tarea eliminada</response>
/// <response code="404"> Si no se pudo eliminar la tarea </response>
[HttpDelete("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
0 referencias
public ActionResult DeleteTaskById(int id) {

    _logger.LogInformation("Eliminando tarea con id: " + id);

    if (_taskList != null) {
        this._taskList.RemoveAt(id);
        return Ok(_taskList.ToList());
    }

    return NotFound("No se encontró una tarea con el id: " + id);
}

```

c. consultar todos

```
/// <summary>
/// Obtiene todas las tareas del sistema
/// </summary>
/// <returns> Devuelve una lista de tareas </returns>
/// <response code="200"> Retorna una lista de todas las tareas</response>
/// <response code="404"> Si la lista de tareas esta vacía </response>
[HttpGet]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
0 referencias
public ActionResult<IList<TaskClass>> GetAllTasks()
{
    _logger.LogInformation("Devolviendo lista de tareas");

    if (_taskList != null)
    {
        return Ok(_taskList.ToList());
    }

    return NotFound("No hay usuarios en el sistema");
}
```

d. consultar por id

```
/// <summary>
/// Buscar tarea por id
/// </summary>
/// <param name="id"> Utiliza el identificador para buscar la tarea</param>
/// <returns> Devuelve una lista en particular </returns>
/// <response code="200"> Retorna una tarea </response>
/// <response code="404"> Si no se pudo encontrar la tarea </response>
[HttpGet]
[Route("{id}")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
0 referencias
public ActionResult<TaskClass> GetTaskById(int id)
{
    _logger.LogInformation("Devolviendo " + $"tarea con id: {id}");

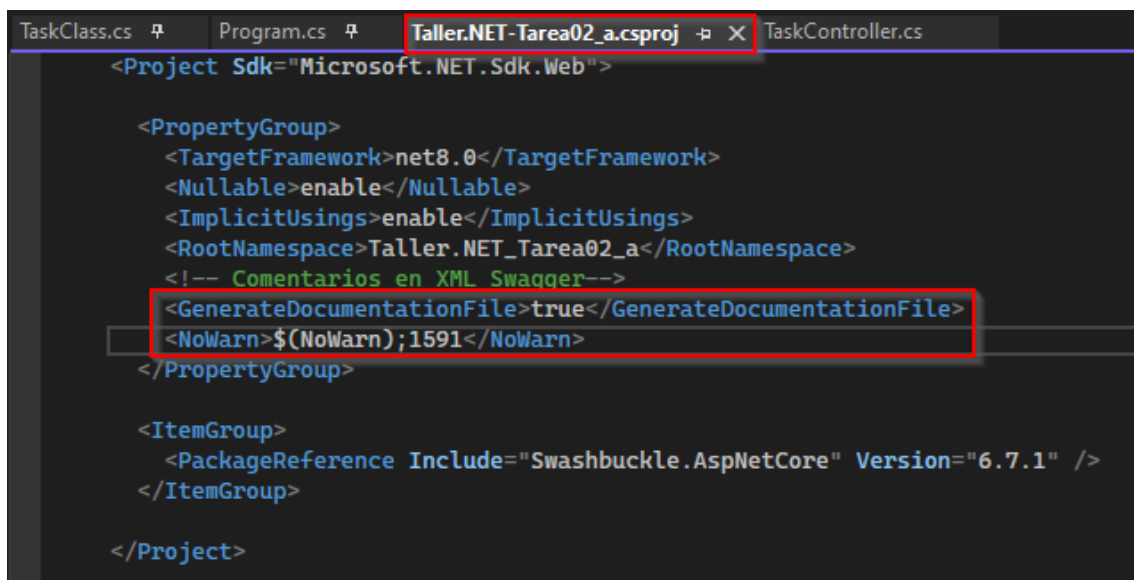
    if (_taskList != null)
    {
        return Ok(_taskList[id]);
    }

    return NotFound("No se encontró una tarea con el id: " + id);
}
```

5. Realizar las pruebas con Swagger.

Al momento de realizar pruebas con Swagger se realizaron una serie de modificaciones en la implementación del mismo para habilitar la documentación en XML, dichas modificaciones son las siguientes:

Habilitado de documentación XML y supresión de warnings.



```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <RootNamespace>Taller.NET_Tarea02_a</RootNamespace>
    <!-- Comentarios en XML Swagger -->
    <GenerateDocumentationFile>true</GenerateDocumentationFile>
    <NoWarn>$(NoWarn);1591</NoWarn>
  </PropertyGroup>

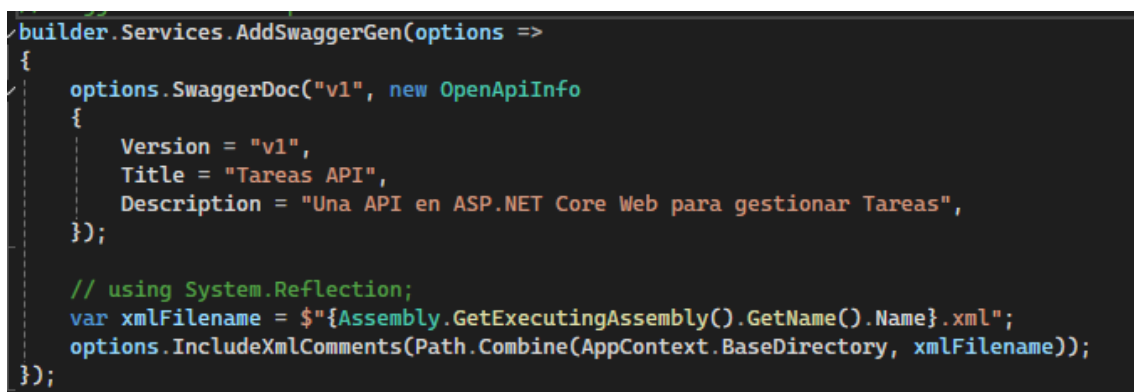
  <ItemGroup>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.7.1" />
  </ItemGroup>

</Project>
```

Se puede observar que dichos cambios se realizan en el archivo .csproj

A continuación, se modifica la manera en la que se genera Swagger para que habilite los formatos XML.

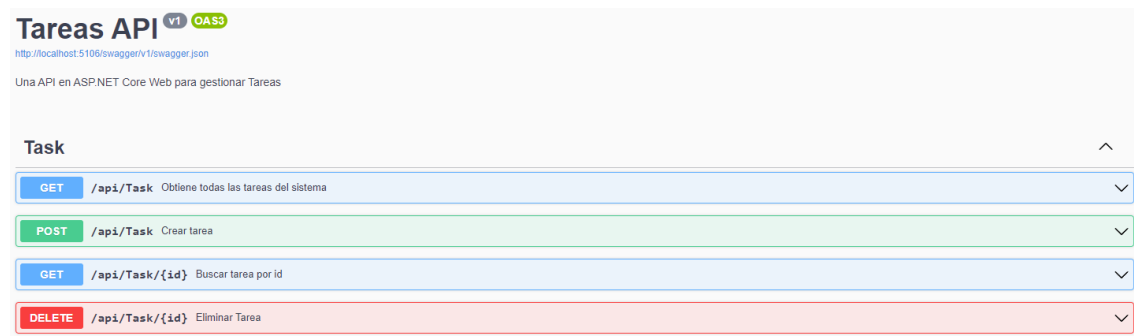
Modificación **builder.Services.AddSwaggerGen**:



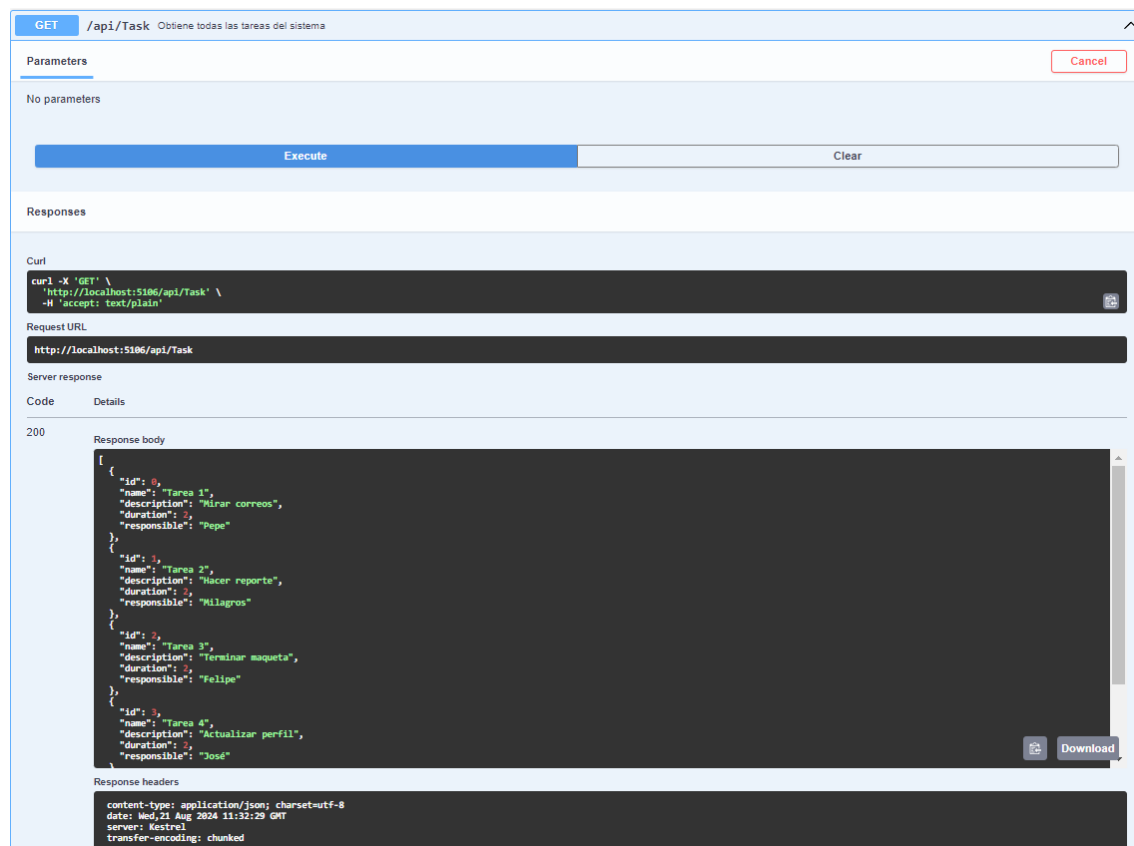
```
builder.Services.AddSwaggerGen(options =>
{
    options.SwaggerDoc("v1", new OpenApiInfo
    {
        Version = "v1",
        Title = "Tareas API",
        Description = "Una API en ASP.NET Core Web para gestionar Tareas",
    });

    // using System.Reflection;
    var xmlFilename = $"{Assembly.GetExecutingAssembly().GetName().Name}.xml";
    options.IncludeXmlComments(Path.Combine(AppContext.BaseDirectory, xmlFilename));
});
```

Con dichas modificaciones agregadas se procede a probar con Swagger los endpoints implementados.



Obtener todas las tareas:



Se obtiene una respuesta positiva y se devuelve una lista con todas las tareas que se encuentran ingresadas en el sistema.



## Crear Tarea:

POST /api/Task Crear tarea

Sample request:

```
{
  "id": 5,
  "name": "Revisar Tarea",
  "description": "Tareas de API's en .NET",
  "duration": 2,
  "responsable": "Gabriel"
}
```

Parameters

No parameters

Request body

application/json

Utiliza los parametros del body para crear el objeto

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "duration": 0,
  "responsible": "string"
}
```

Responses

Code	Description	Links
200	Retorna una lista de tarea, con la nueva tarea	No links
500	Si no se pudo crear la tarea	No links

En este caso nos ayudaremos de la petición de prueba para crear una tarea rápidamente.

Request body

application/json

Utiliza los parametros del body para crear el objeto

```
{
  "id": 5,
  "name": "Revisar Tarea",
  "description": "Tareas de API's en .NET",
  "duration": 2,
  "responsable": "Gabriel"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  http://localhost:5106/api/Task \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 5,
    "name": "Revisar Tarea",
    "description": "Tareas de API\'s en .NET",
    "duration": 2,
    "responsable": "Gabriel"
  }'
```

Request URL

http://localhost:5106/api/Task

A continuación, se observa el resultado de la operación:

Server response

Code	Details
200	<p>Response body</p> <pre> {   "name": "Tarea 3",   "description": "Terminar maqueta",   "duration": 2,   "responsible": "Felipe" }, {   "id": 3,   "name": "Tarea 4",   "description": "Actualizar perfil",   "duration": 2,   "responsible": "José" }, {   "id": 4,   "name": "Tarea 5",   "description": "Chequear actualizaciones",   "duration": 2,   "responsible": "Nicolás" }, {   "id": 5,   "name": "Revisar Tarea",   "description": "Tareas de API's en .NET",   "duration": 2,   "responsible": "Gabriel" } </pre> <p>Response headers</p> <pre> content-type: application/json; charset=utf-8 date: Wed, 21 Aug 2024 11:41:33 GMT server: Kestrel transfer-encoding: chunked </pre>

Responses

Code	Description	Links
200	Retorna una lista de tarea, con la nueva tarea	No links
500	Si no se pudo crear la tarea	No links

Se puede observar en el último lugar la nueva tarea ingresada.

Buscar tarea por id:

GET /api/Task/{id} Buscar tarea por id

Parameters

Try it out

Name	Description
<b>id</b> <small>required</small>	Utiliza el identificador para buscar la tarea

Responses

Code	Description	Links
200	Retorna una tarea	No links

Media type:

Controls Accept header:

Example Value | Schema

```

{
  "id": 0,
  "name": "string",
  "description": "string",
  "duration": 0,
  "responsible": "string"
}

```

## Resultado de buscar tarea por id:

GET /api/Task/{id} Buscar tarea por id

Parameters

Cancel

Name	Description
id * required	Utiliza el identificador para buscar la tarea

integer(\$int32) (path)

2

Execute Clear

Responses

Curl

```
curl -X 'GET' \  
  'http://localhost:5106/api/Task/2' \  
  -H 'accept: text/plain'
```

Request URL

http://localhost:5106/api/Task/2

Server response

Code	Details
200	<p>Response body</p> <pre>{   "id": 2,   "name": "Tarea 3",   "description": "Terminar maqueta",   "duration": 2,   "responsible": "Felipe" }</pre> <p>Response headers</p> <pre>content-type: application/json; charset=utf-8 date: Wed, 21 Aug 2024 11:53:38 GMT server: Kestrel transfer-encoding: chunked</pre>

Se puede observar en el Response Body la tarea con id 2 que fue devuelta.

## Eliminar Tarea:

DELETE /api/Task/{id} Eliminar Tarea

Try it out

Name	Description
id * required	Utiliza el identificador para buscar y eliminar la tarea

integer(\$int32) (path)

id

Responses

Code	Description	Links
200	Retorna una lista de tareas con la tarea eliminada	No links
404	Si no se pudo eliminar la tarea	No links

Media type

text/plain

Example Value | Schema

```
{  
  "type": "string",  
  "title": "string",  
  "status": 0,  
  "detail": "string",  
  "instance": "string",  
  "additionalProp1": "string",  
  "additionalProp2": "string",  
  "additionalProp3": "string"  
}
```

Para este caso se eliminará la tarea con id 2 obtenida en el punto anterior.

## Resultado:

NameDescription

id required

integer(\$int32)

Utiliza el identificador para buscar y eliminar la tarea

(path)

2

Execute

Clear

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:5106/api/Task/2' \
-H 'accept: */*'
```

Request URL

http://localhost:5106/api/Task/2

Server response

Code

Details

200

Response body

```
{
  "id": 0,
  "name": "Tarea 1",
  "description": "Mirar correos",
  "duration": 0,
  "responsible": "Pepe"
},
{
  "id": 1,
  "name": "Tarea 2",
  "description": "Hacer reporte",
  "duration": 0,
  "responsible": "Milagros"
},
{
  "id": 3,
  "name": "Tarea 4",
  "description": "Actualizar perfil",
  "duration": 0,
  "responsible": "José"
},
{
  "id": 4,
  "name": "Tarea 5",
  "description": "Chequear actualizaciones",
  "duration": 0,
  "responsible": "Nicolás"
}
```

Download

Se puede observar en la lista la ausencia de la tarea de id 2, al ver como salta entre id 1 a id 3.

## 6. Agregar a la clase del negocio un atributo de tipo fecha.

### Modificación en clase de dominio:

```

//Propiedades
2 referencias
public string? Name { get; set; }
1 referencia
public string? Description { get; set; }
1 referencia
public int? Duration { get; set; }
1 referencia
public string? Responsible { get; set; }
2 referencias
public DateOnly? creationDate { get; set; }

5 referencias
public TaskClass() {
}

5 referencias
public TaskClass(int id, string name, string description, int duration, string responsible, DateOnly creationDate)
{
    this._id = id;
    this.Name = name;
    this.Description = description;
    this.Duration = duration;
    this.Responsible = responsible;
    this.creationDate = creationDate;
}
  
```

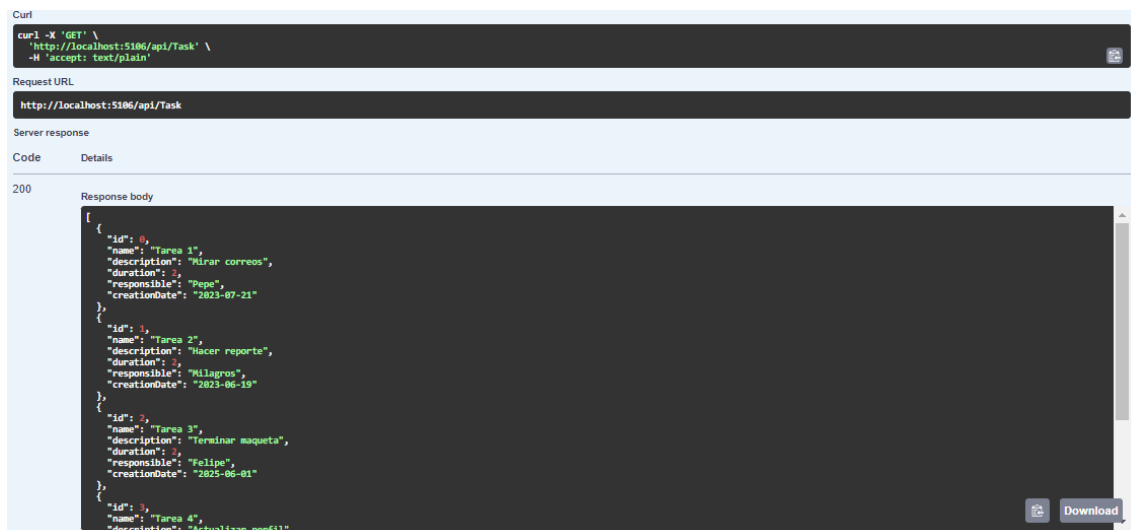
También se modificarán los datos de prueba para agregar una fecha.

```
this._taskList.Add(new TaskClass(0, "Tarea 1", "Mirar correos", 2, "Pepe", new DateOnly(2023, 7, 21)));
this._taskList.Add(new TaskClass(1, "Tarea 2", "Hacer reporte", 2, "Milagros", new DateOnly(2023, 6, 19)));
this._taskList.Add(new TaskClass(2, "Tarea 3", "Terminar maqueta", 2, "Felipe", new DateOnly(2025, 6, 1)));
this._taskList.Add(new TaskClass(3, "Tarea 4", "Actualizar perfil", 2, "José", new DateOnly(2023, 4, 2)));
this._taskList.Add(new TaskClass(4, "Tarea 5", "Chequear actualizaciones", 2, "Nicolás", new DateOnly(2024, 12, 31)));
```

Y una pequeña modificación en la documentación a la hora de crear tarea, para incluir este nuevo atributo.

```
/// <summary>
/// Crear tarea
/// </summary>
/// <param name="task"> Utiliza los parametros del body para crear el objeto </param>
/// <returns>Una lista de tareas con la tarea agregada</returns>
/// <remarks>
/// Sample request:
///
/// {
///   "id": 5,
///   "name": "Revisar Tarea",
///   "description": "Tareas de API's en .NET",
///   "duration": 2,
///   "responsible": "Gabriel",
///   "creationDate": "2024-04-30"
/// }
/// </remarks>
/// <response code="200"> Retorna una lista de tarea, con la nueva tarea</response>
/// <response code="500"> Si no se pudo crear la tarea </response>
[HttpPost]
```

Y se prueba obteniendo todas las tareas:



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

- Request:** curl -X 'GET' \ 'http://localhost:5106/api/Task' \ -H 'accept: text/plain'
- Request URL:** http://localhost:5106/api/Task
- Server response:** Code 200, Details
- Response body:** A JSON array of 5 tasks. The first task is:
 

```
{
    "id": 0,
    "name": "Tarea 1",
    "description": "Mirar correos",
    "duration": 2,
    "responsible": "Pepe",
    "creationDate": "2023-07-21"
  },
```

Se puede observar el nuevo atributo de tipo fecha en cada objeto Tarea.

A continuación, se creará una tarea nueva, para demostrar el correcto funcionamiento.

POST /api/Task Crear tarea

Sample request:

```
{
  "id": 5,
  "name": "Revisar Tarea",
  "description": "Tareas de API's en .NET",
  "duration": 2,
  "responsible": "Gabriel",
  "creationDate": "2024-04-30"
}
```

Parameters Try it out

No parameters

Request body application/json

Utiliza los parametros del body para crear el objeto

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "duration": 0,
  "responsible": "string",
  "creationDate": "2024-08-21"
}
```

Se utilizará la petición de prueba para crear esa Tarea.

Ejecución:

Request body application/json

Utiliza los parametros del body para crear el objeto

```
{
  "id": 5,
  "name": "Revisar Tarea",
  "description": "Tareas de API's en .NET",
  "duration": 2,
  "responsible": "Gabriel",
  "creationDate": "2024-04-30"
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5106/api/Task' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "id": 5,
    "name": "Revisar Tarea",
    "description": "Tareas de API's en .NET",
    "duration": 2,
    "responsible": "Gabriel",
    "creationDate": "2024-04-30"
  }'
```

Respuesta:

Code Details

200

Response body

```
{
  "duration": 2,
  "responsible": "Felipe",
  "creationDate": "2025-06-01"
},
{
  "id": 3,
  "name": "Tarea 4",
  "description": "Actualizar perfil",
  "duration": 2,
  "responsible": "José",
  "creationDate": "2023-04-02"
},
{
  "id": 4,
  "name": "Tarea 5",
  "description": "Chequear actualizaciones",
  "duration": 2,
  "responsible": "Nicolás",
  "creationDate": "2024-12-31"
},
{
  "id": 5,
  "name": "Revisar Tarea",
  "description": "Tareas de API's en .NET",
  "duration": 2,
  "responsible": "Gabriel",
  "creationDate": "2024-04-30"
}
```

Download

Con el correcto funcionamiento de los casos anteriores se daría por finalizada la demostración del nuevo atributo agregado, se obvia la prueba de los casos de obtener y eliminar tarea por id, ya que se considera que su comportamiento será el mismo, debido a que esas operaciones no sufren grandes modificaciones y las operaciones utilizadas cubren lo necesario.

Repositorio de GitHub con el código: [https://github.com/ElPiche/.NET\\_Exercices.git](https://github.com/ElPiche/.NET_Exercices.git)