

## Ejercicio: Sistema de Gestión de Tareas

### 1. Frontend (AngularJS 1.8)

Inconcluso: Dificultad para realizar el punto por falta de conocimiento sobre la tecnología.

### 2. Backend (.NET Framework 4.8)

Reto: Dificultad para realizar el punto por falta de conocimiento sobre la tecnología.

Creación de la clase TAREAS como modelo de datos

```
1  namespace EJERCICIO_BACK.Models
2  {
3      using System;
4      using System.Collections.Generic;
5      using System.ComponentModel.DataAnnotations;
6      using System.ComponentModel.DataAnnotations.Schema;
7      using System.Data.Entity;
8      using System.Data.Entity.Spatial;
9
10     9 referencias
11     public partial class TAREAS
12     {
13         3 referencias
14         public int ID { get; set; }
15
16         [Required]
17         [StringLength(100)]
18         0 referencias
19         public string TITULO { get; set; }
20
21         [StringLength(500)]
22
23         public string DESCRIPCION { get; set; }
24
25         [Required]
26         0 referencias
27         public DateTime FECHA_CREACION { get; set; }
28
29         [Required]
30         0 referencias
31         public bool ESTADO { get; set; }
32     }
33 }
```

Creación de contexto de datos TaskManagerContext que herede de DbContext.

```
using System;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.Entity;
using System.Linq;

namespace EJERCICIO_BACK.Models
{
    3 referencias
    public partial class TaskManagerContext : DbContext
    {
        1 referencia
        public TaskManagerContext()
            : base("name=TaskManagerContext")
        {
        }

        6 referencias
        public virtual DbSet<TAREAS> TAREAS { get; set; }

        0 referencias
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
        }
    }
}
```

Implementación de controladores API para manejar las peticiones HTTP.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Data;
4 using System.Data.Entity;
5 using System.Data.Entity.Infrastructure;
6 using System.Linq;
7 using System.Net;
8 using System.Net.Http;
9 using System.Web.Http;
10 using System.Web.Http.Description;
11 using EJERCICIO_BACK.Models;
12
13 namespace EJERCICIO_BACK.Controllers
14 {
15     0 referencias
16     public class TAREASController : ApiController
17     {
18         private TaskManagerContext db = new TaskManagerContext();
19
20         // GET: api/TAREAS
21         0 referencias
22         public IQueryable<TAREAS> GetTAREAS()
23         {
24             return db.TAREAS;
25         }
26     }
27 }
```

```

24
25 // GET: api/TAREAS/5
26 [ResponseType(typeof(TAREAS))]
27 0 referencias
28 public IHttpActionResult GetTAREAS(int id)
29 {
30     TAREAS tAREAS = db.TAREAS.Find(id);
31     if (tAREAS == null)
32     {
33         return NotFound();
34     }
35     return Ok(tAREAS);
36 }
37
38 // PUT: api/TAREAS/5
39 [ResponseType(typeof(void))]
40 0 referencias
41 public IHttpActionResult PutTAREAS(int id, TAREAS tAREAS)
42 {
43     if (!ModelState.IsValid)
44     {
45         return BadRequest(ModelState);
46     }

```

```

47     if (id != tAREAS.ID)
48     {
49         return BadRequest();
50     }
51
52     db.Entry(tAREAS).State = EntityState.Modified;
53
54     try
55     {
56         db.SaveChanges();
57     }
58     catch (DbUpdateConcurrencyException)
59     {
60         if (!TAREASExists(id))
61         {
62             return NotFound();
63         }
64         else
65         {
66             throw;
67         }
68     }
69
70     return StatusCode(HttpStatusCode.NoContent);

```

```

71 }
72
73 // POST: api/TAREAS
74 [ResponseType(typeof(TAREAS))]
75 0 referencias
76 public IHttpActionResult PostTAREAS(TAREAS tAREAS)
77 {
78     if (!ModelState.IsValid)
79     {
80         return BadRequest(ModelState);
81     }
82
83     db.TAREAS.Add(tAREAS);
84     db.SaveChanges();
85
86     return CreatedAtRoute("DefaultApi", new { id = tAREAS.ID }, tAREAS);
87 }
88
89 // DELETE: api/TAREAS/5
90 [ResponseType(typeof(TAREAS))]
91 0 referencias
92 public IHttpActionResult DeleteTAREAS(int id)
93 {
94     TAREAS tAREAS = db.TAREAS.Find(id);
95     if (tAREAS == null)

```

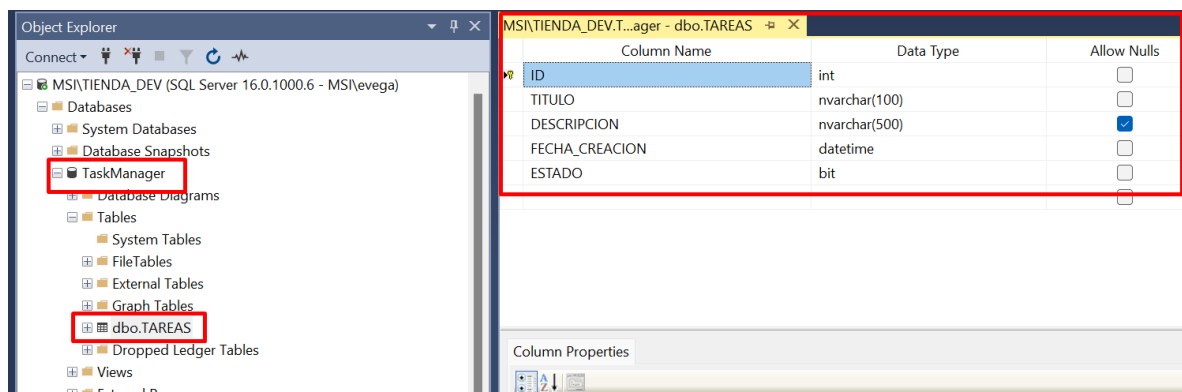
```

93         if (tAREAS == null)
94         {
95             return NotFound();
96         }
97
98         db.TAREAS.Remove(tAREAS);
99         db.SaveChanges();
100
101         return Ok(tAREAS);
102     }
103
104     0 referencias
105     protected override void Dispose(bool disposing)
106     {
107         if (disposing)
108         {
109             db.Dispose();
110         }
111         base.Dispose(disposing);
112     }
113
114     1 referencia
115     private bool TAREASExists(int id)
116     {
117         return db.TAREAS.Count(e => e.ID == id) > 0;
118     }

```

### 3. Base de Datos (SQL Server con Entity Framework)

Se creó la base de datos de acuerdo a las indicaciones de la actividad.

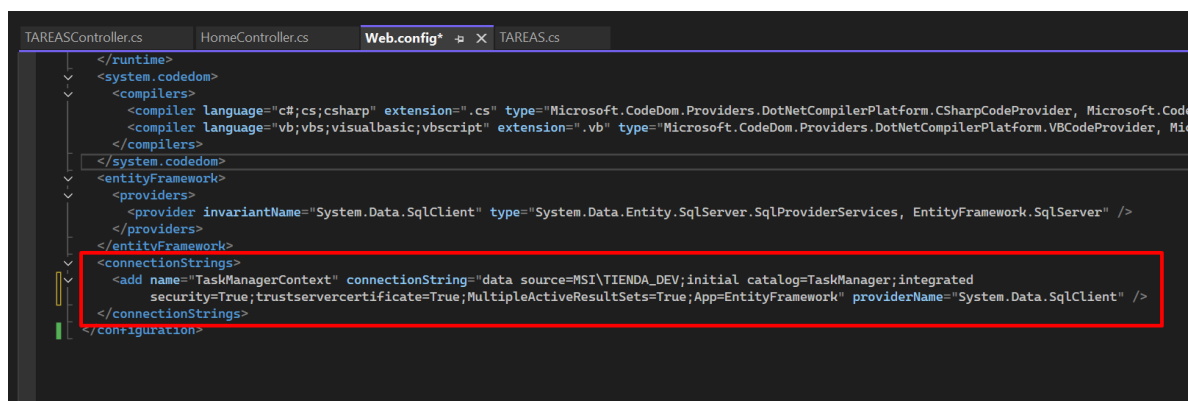


The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Object Explorer' pane displays the 'MSI\TIENDA\_DEV' database, with 'TaskManager' and 'dbo.TAREAS' highlighted. On the right, the 'Table Designer' for 'dbo.TAREAS' is open, showing the following columns:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
TITULO	nvarchar(100)	<input type="checkbox"/>
DESCRIPCION	nvarchar(500)	<input checked="" type="checkbox"/>
FECHA_CREACION	datetime	<input type="checkbox"/>
ESTADO	bit	<input type="checkbox"/>

### 4. Conexión con Entity Framework

Se creó la conexión con entity framework



The screenshot shows the 'Web.config' file in a code editor. The configuration is as follows:

```

<?xml version='1.0' encoding='utf-8'>
  <runtime>
    <system.codedom>
      <compilers>
        <compiler language="c#;cs;csharp" extension=".cs" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.CSharpCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform" />
        <compiler language="vb;vbs;visualbasic;vbscript" extension=".vb" type="Microsoft.CodeDom.Providers.DotNetCompilerPlatform.VBCodeProvider, Microsoft.CodeDom.Providers.DotNetCompilerPlatform" />
      </compilers>
    </system.codedom>
    <entityFramework>
      <providers>
        <provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
      </providers>
    </entityFramework>
    <connectionStrings>
      <add name="TaskManagerContext" connectionString="data source=MSI\TIENDA_DEV;initial catalog=TaskManager;integrated security=True;trustservercertificate=True;MultipleActiveResultSets=True;App=EntityFramework" providerName="System.Data.SqlClient" />
    </connectionStrings>
  </runtime>
</configuration>

```

## Pruebas y Validaciones

Las pruebas se ejecutaron desde la aplicación POSTMAN.

### Prueba de operación GET

The screenshot shows a Postman interface with a GET request to `{{Base_URL}}/api/TAREAS`. The response is a 200 OK status with a response time of 64 ms and a body size of 619 B. The response body is displayed in JSON format:

```
1 [
2   {
3     "ID": 1,
4     "TITULO": "TAREA_PRUEBA",
5     "DESCRIPCION": "PRUEBA",
6     "FECHA_CREACION": "2024-09-12T16:00:16.907",
7     "ESTADO": false
8   },
9 ]
```

### Prueba de operación POST

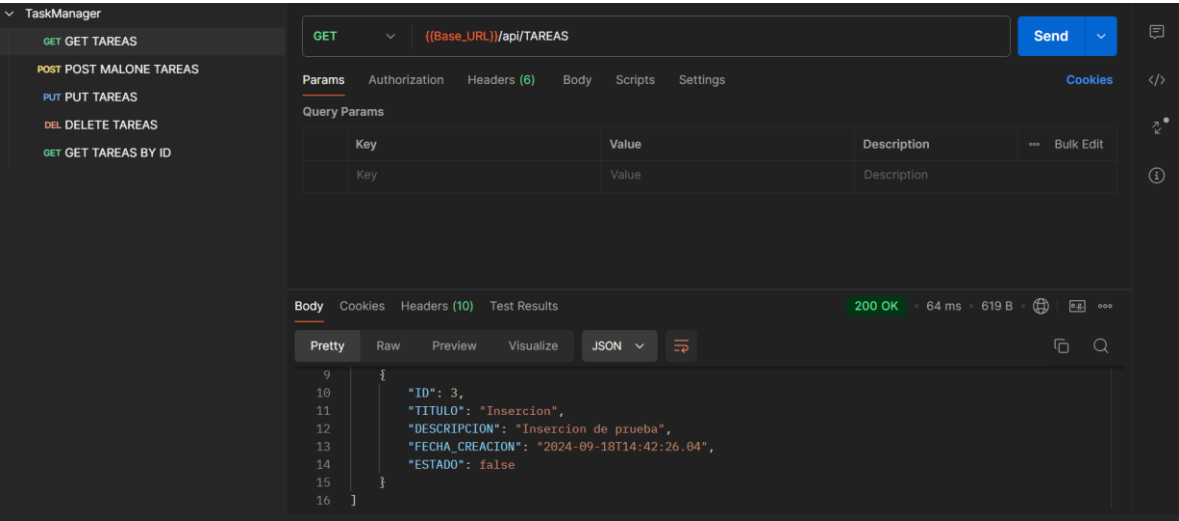
The screenshot shows a Postman interface with a POST request to `{{Base_URL}}/api/TAREAS`. The request body is a JSON object:

```
1 {
2   "TITULO": "Insercion",
3   "DESCRIPCION": "Insercion de prueba",
4   "FECHA_CREACION": "2024-09-18T14:42:26.0414814-06:00",
5   "ESTADO": false
6 }
```

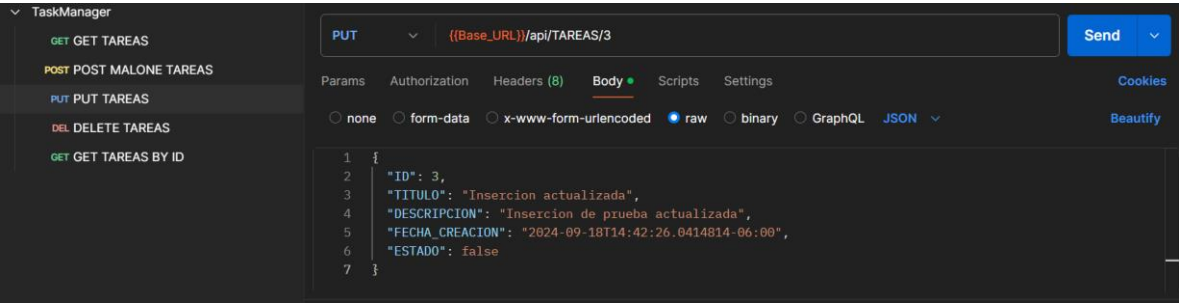
The response is a 201 Created status with a response time of 838 ms and a body size of 566 B. The response body is displayed in JSON format:

```
1 {
2   "ID": 3,
3   "TITULO": "Insercion",
4   "DESCRIPCION": "Insercion de prueba",
5   "FECHA_CREACION": "2024-09-18T14:42:26.0414814-06:00",
6   "ESTADO": false
7 }
```

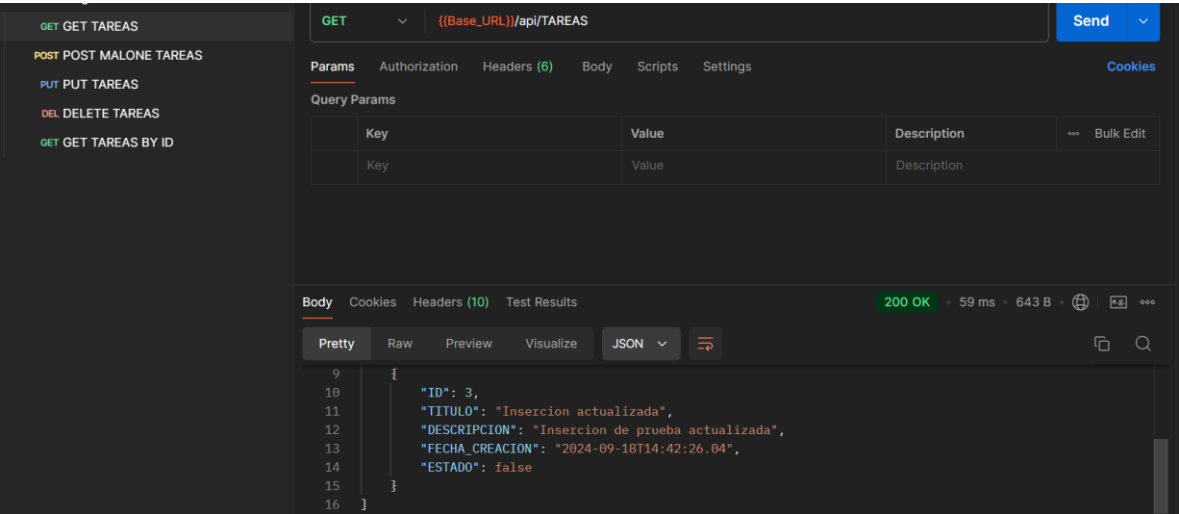
Consulta de validación:



## Prueba de operación PUT



## Consulta de validación



## Prueba de operación DELETE

Se enviará el id 3 en la URL para eliminar ese registro

The screenshot shows a Postman interface with a DELETE request to `{{Base_URL}}/api/TAREAS/3`. The response is a 200 OK status with a JSON body containing task details for ID 3.

**Request:**

- Method: DELETE
- URL: `{{Base_URL}}/api/TAREAS/3`
- Body: 1

**Response:**

- Status: 200 OK
- Time: 149 ms
- Size: 531 B
- Body (JSON):

```
{  "ID": 3,  "TITULO": "Insercion actualizada",  "DESCRIPCION": "Insercion de prueba actualizada",  "FECHA_CREACION": "2024-09-18T14:42:26.04",  "ESTADO": false}
```

## Consulta de validación

The screenshot shows a GET request to `{{Base_URL}}/api/TAREAS/3` resulting in a 404 Not Found status.

**Request:**

- Method: GET
- URL: `{{Base_URL}}/api/TAREAS/3`

**Response:**

- Status: 404 Not Found
- Time: 69 ms
- Size: 343 B
- Body: 1