

TRABAJO FINAL



Título: SISTEMAS DE INFORMACIÓN Y SISTEMAS INTELIGENTES (Dao Postgre Update Select)
Autor: -
Fecha: 18/10/2022
Carrera: Ing. De Sistemas
Asignatura: Módulo 2. Sistemas de Información y Sistemas Inteligentes
Grupo: A
Docente: ROGER ORLANDO RUIZ ESCOBAR
Periodo Académico: 2 - 2022
Subsede: Santa Cruz de la Sierra

Copyright © (2022) por (Ronald Saucedo Tito, Jesus Moisés Chavarria Martínez, Jose Ricardo Gonzales Patiño Marcial Mamani Yujra, Jhoel Jorge Uria Porcel, José María Gomes Peralta.). Todos los derechos reservados.

Tabla De Contenidos

Introducción.....	4
Desarrollo de la Investigación	5
1.3.1 PostgreSQL	5
1.3.2 .Red	5
1.3.3 IDE.....	5

1.3.4 Implementación de Proyecto	5
Bibliografía	18

Lista de Figuras

Figura 1	5
Figura 2	6
Figura 3	7
Figura 4	8
Figura 5	9

Figura 6.....	9
Figura 7.....	10
Figura 8.....	10
Figura 9.....	11
Figura 10.....	12
Figura 11.....	17
Figura 12.....	17

Introducción

Es indudable que la llegada de las Nuevas Tecnologías, han revolucionado, para un desarrollador web moderno, saber cómo trabajar con API facilita la comunicación entre sistemas de software es primordial, nuestro objetivo es permitir operaciones CRUD, GET, POST, PUT y DELETE, en la API, que ejecutará los comandos de base de datos correspondientes. Para ello, configuraremos una ruta para cada extremo y una función para cada consulta.

Desarrollo de la Investigación

1.3.1 PostgreSQL



Figura 1

PostgreSQL es un potente sistema de base de datos relacional de objetos de código abierto que utiliza y amplía el lenguaje SQL y combina muchas funciones para almacenar y escalar de forma segura las cargas de trabajo de datos más complejas.

1.3.2 .Net

.NET es una plataforma de desarrollo gratuita, multiplataforma y de código abierto para crear muchos tipos diferentes de aplicaciones. Con .NET, puede usar múltiples lenguajes, editores y bibliotecas para crear aplicaciones web, móviles, de escritorio, juegos e IoT.

1.3.3 IDE

Se mostrará como implementar el proyecto en:

- Visual Studio 2022

1.3.4 Implementación de Proyecto

Creando nuevo proyecto

Elija la aplicación web ASP.net Core.

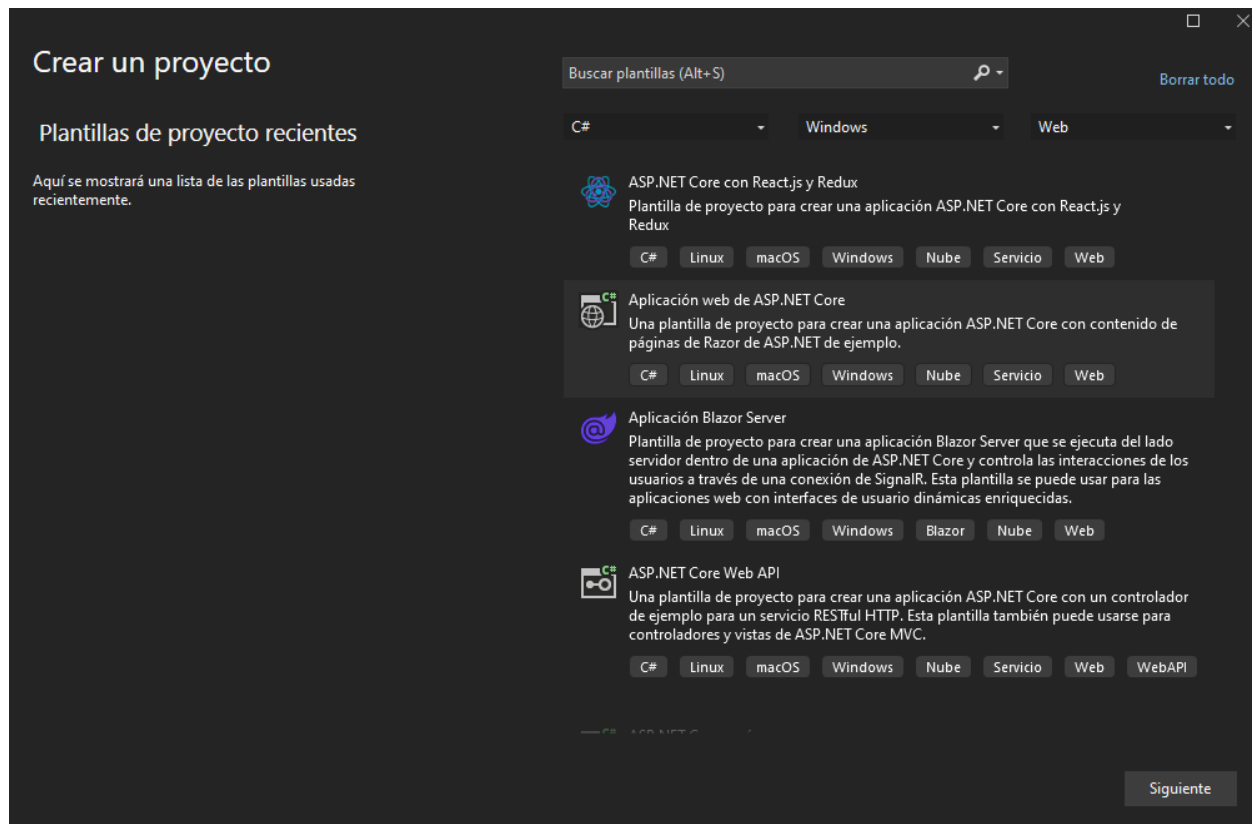


Figura 2

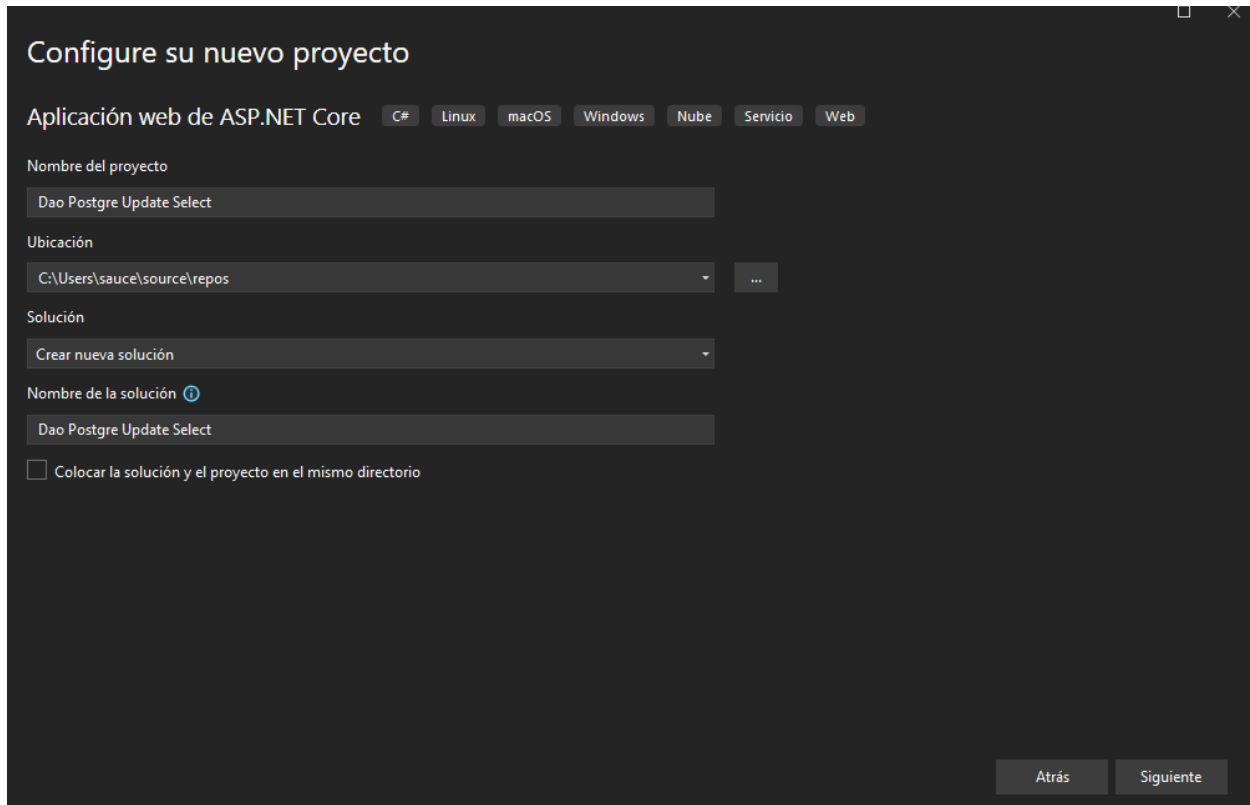


Figura 3

Haga clic con el botón derecho en Project Solution, seleccione Manage Nuget Package e instale los siguientes paquetes.

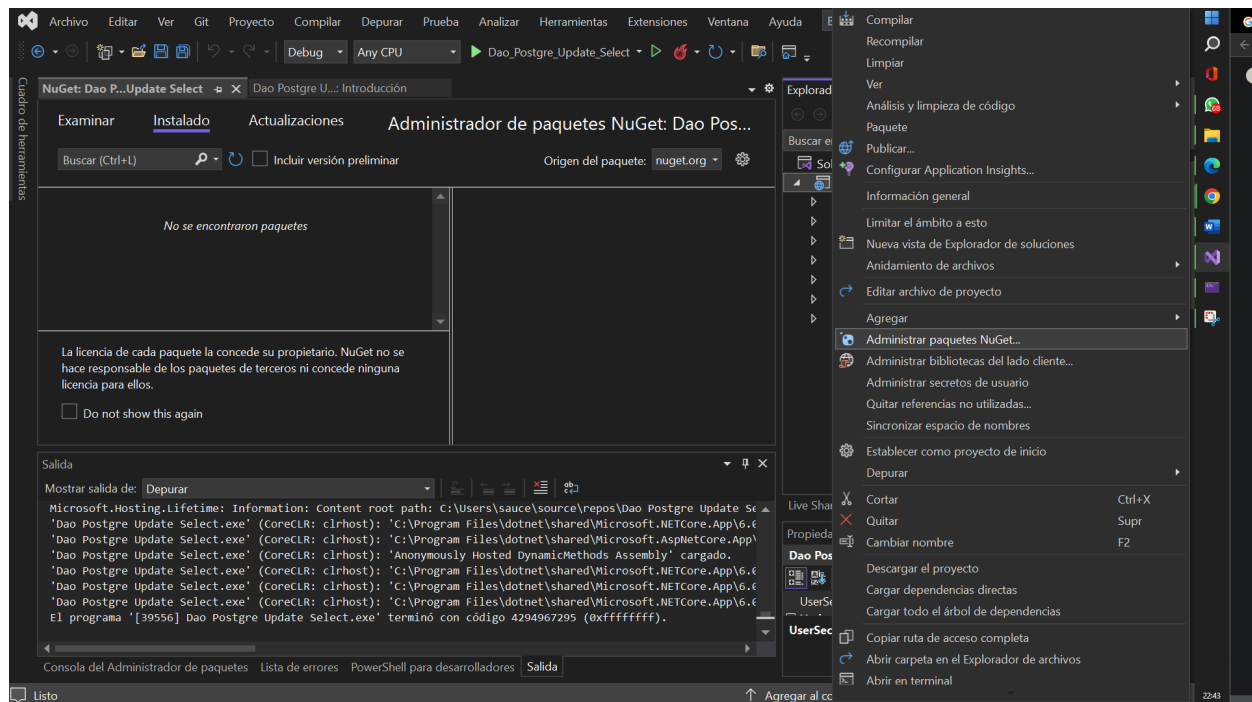







Figura 4

- Npgsql.EntityFrameworkCore.PostgreSQL
- Microsoft.EntityFrameworkCore.Design
- Microsoft.EntityFrameworkCore.Tools

	Microsoft.AspNetCore.All by Microsoft	v2.0.9
	Microsoft.AspNetCore.All	v2.2.4
	Microsoft.EntityFrameworkCore.Design by Microsoft	v2.2.4
	Shared design-time components for Entity Framework Core tools.	
	Microsoft.EntityFrameworkCore.Tools by Microsoft	v2.2.4
	Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.	
	Microsoft.NETCore.App by Microsoft	v2.0.0
	A set of .NET APIs that are included in the default .NET Core application model. e8b8861ac7faf042c87a5c2f9f2d04c98b69f28d	
	Npgsql.EntityFrameworkCore.PostgreSQL by Shay Rojansky, Austin Drenski, Yoh Deadfall	v2.2.0
	PostgreSQL/Npgsql provider for Entity Framework Core.	

Configuración

Después de la instalación, vaya a Startup.cs . En el método ConfigureServices, necesitamos agregar PostgreSQL al proyecto. Agregue Database context (en mi caso,

PostgreSQLConnection) y el nombre de la cadena de conexión (ConnectionString se agrega en appsettings.json) (en mi caso, PostgreSQLConnection).

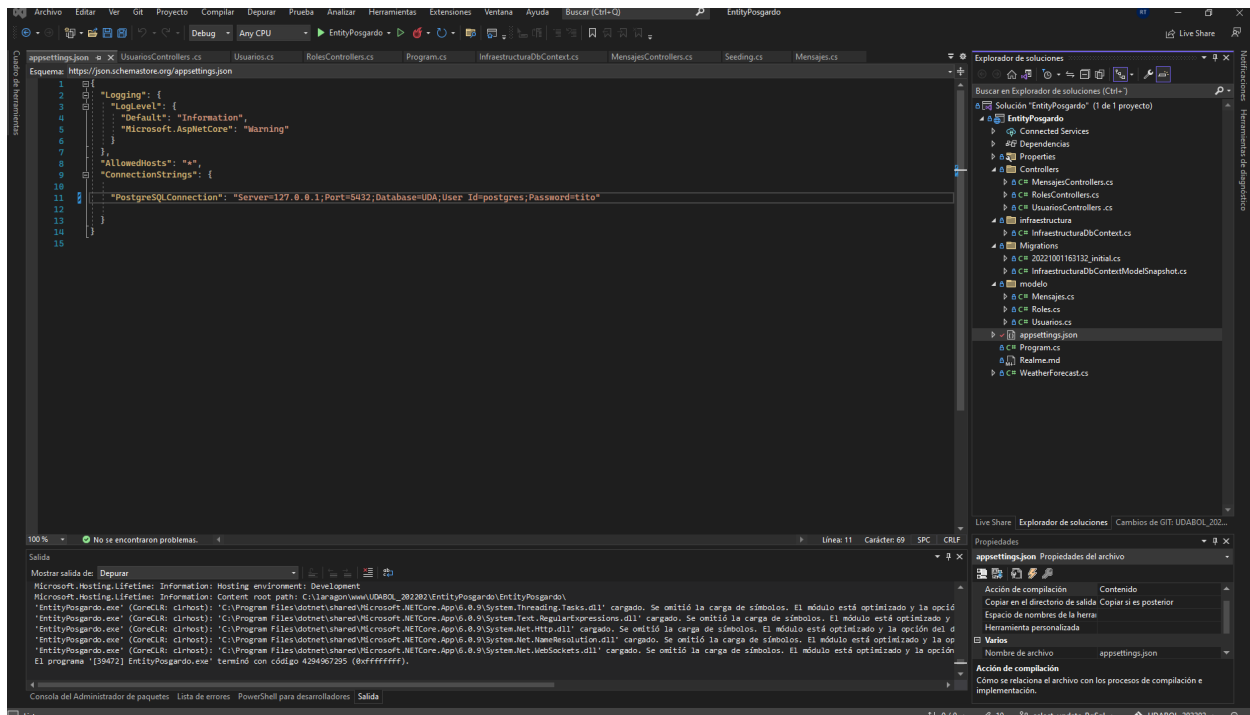


Figura 5

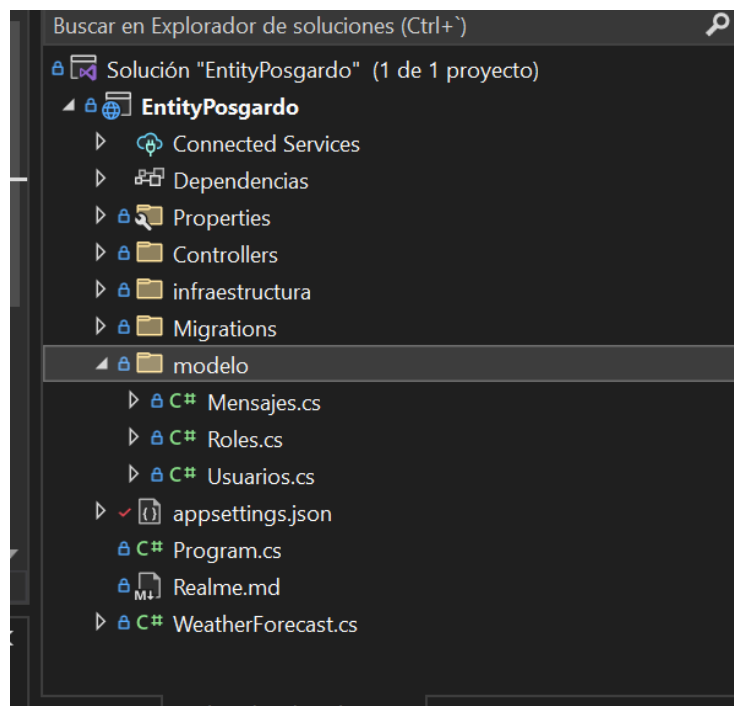


Figura 6

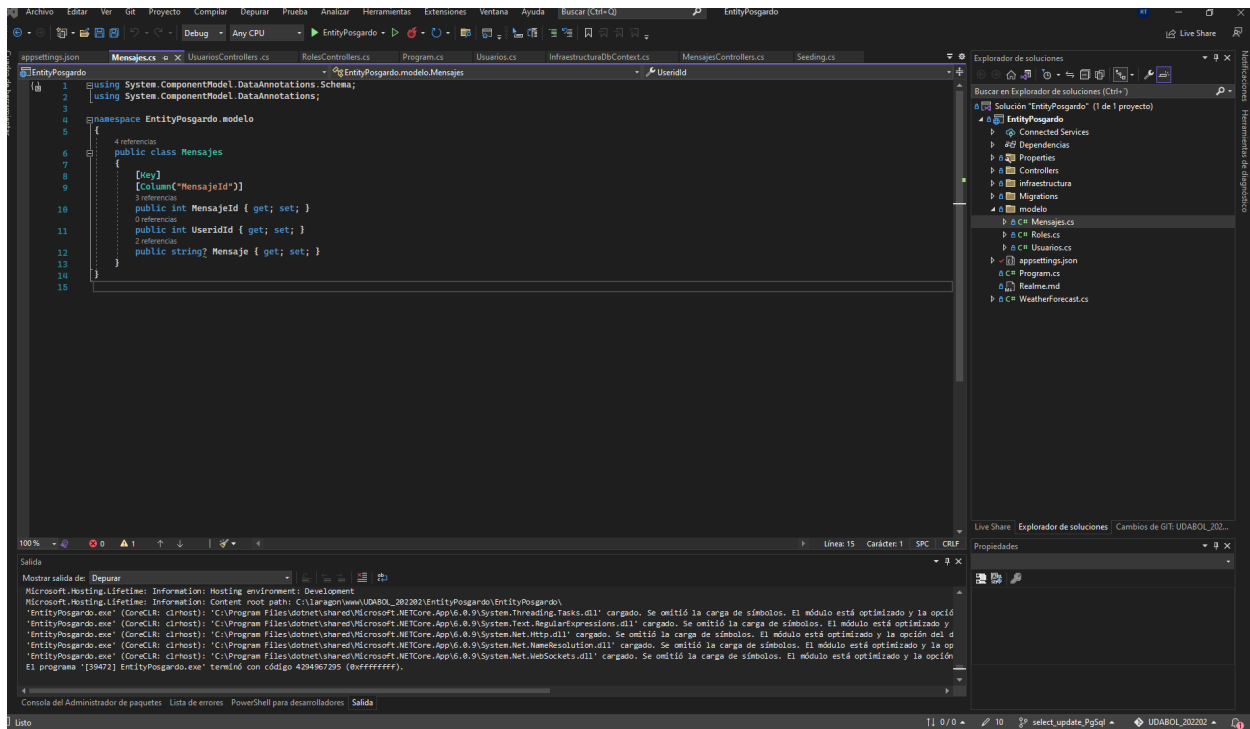


Figura 7

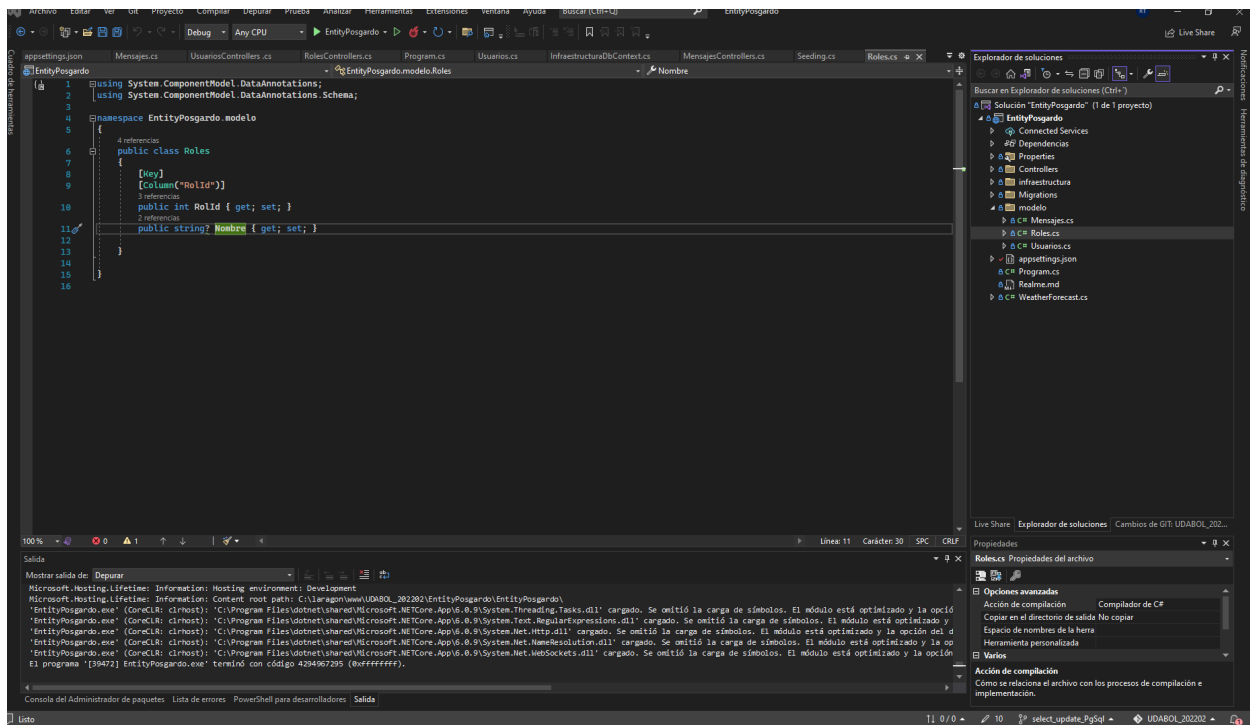


Figura 8

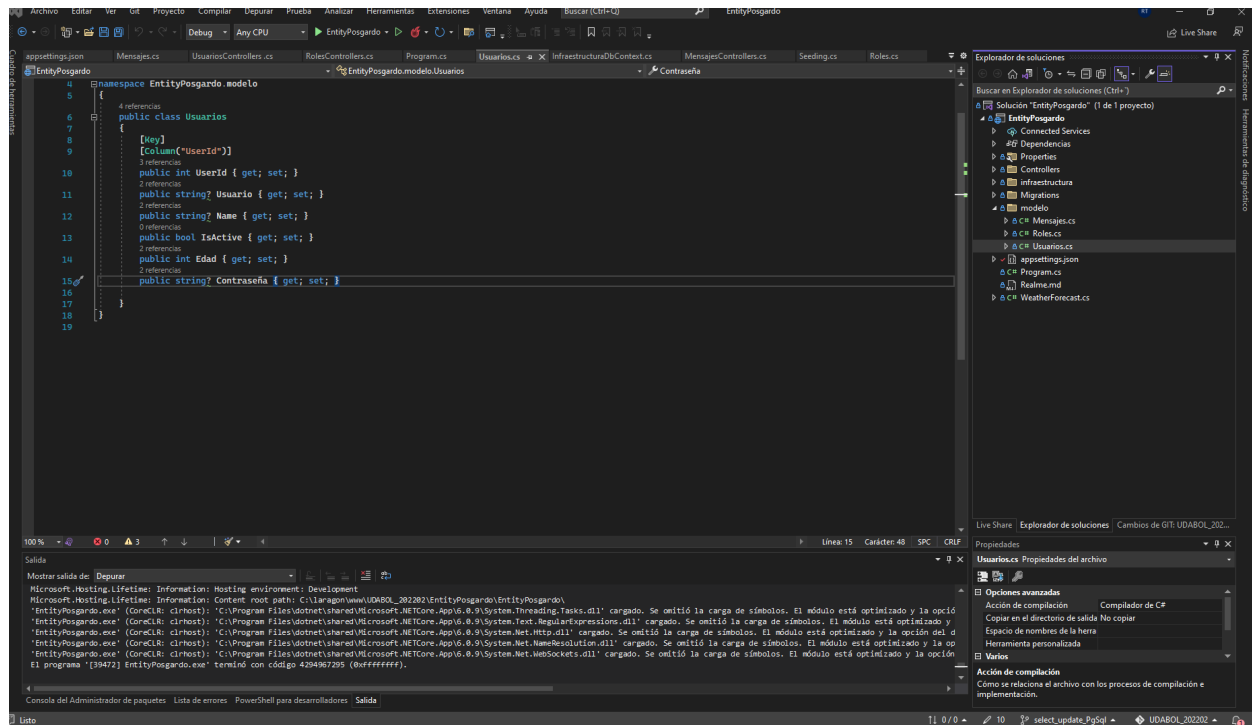


Figura 9

Ahora, cree un DbContext (en mi caso, lo nombré PostgreSQLConnection). Este contexto de base de datos se hereda de DbContext.

PostgreSQLConnection > El nombre del contexto de la base de datos se agrega en el constructor.

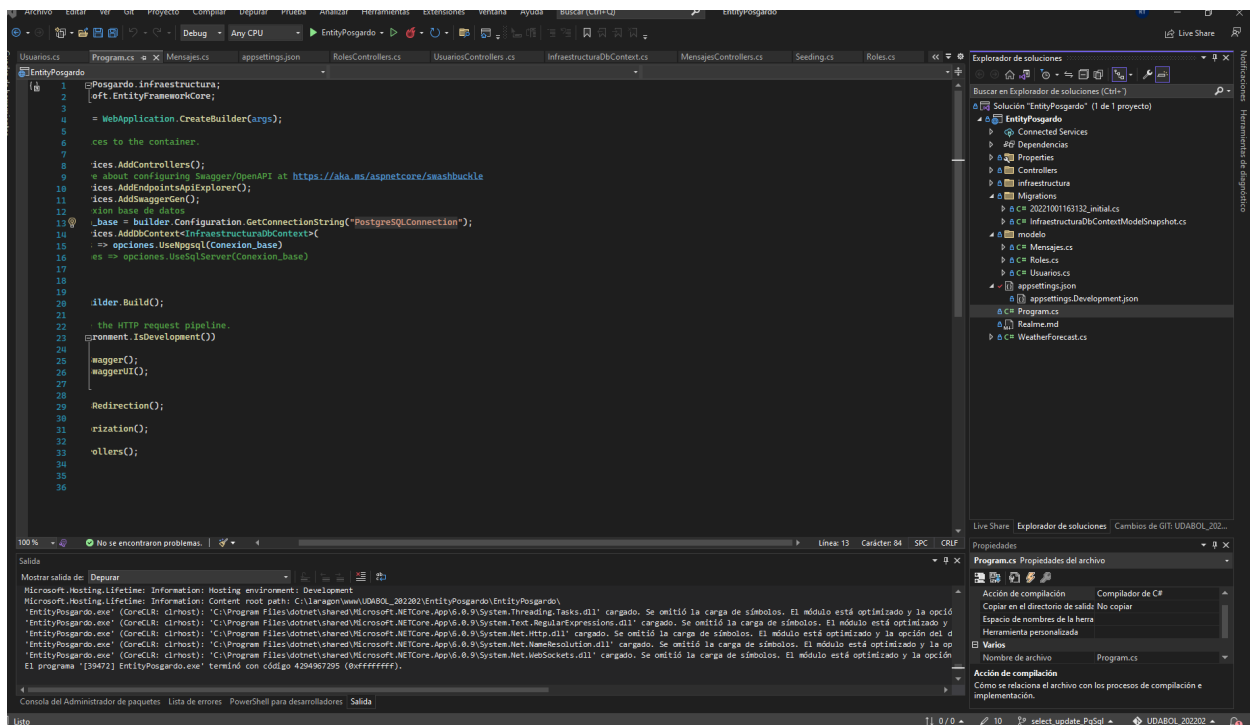


Figura 10

PM> enable-migrations

PM> add-migration initial

PM> update-database

enable-migration habilitará la migración en el proyecto. Add-migration initial creará la primera confirmación en las migraciones. Puede encontrar la migración en la carpeta de migración en el árbol del proyecto. actualizar la base de datos actualizará la base de datos de acuerdo con los modelos recientes.

```
using EntityPosgado.infraestructura;
using EntityPosgado.modelo;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
```

```
namespace EntityPosgado.Controllers
{
    [ApiController]
    [Route("api/mensajes")]
    public class MensajesControllers : ControllerBase
    {

```

```

private readonly InfraestructuraDbContext context;

public MensajesControllers(InfraestructuraDbContext context)
{
    this.context = context;
}

[HttpGet("get")]
public async Task<IEnumerable<Mensajes>> Get()
{
    return await context.Mensajes.ToListAsync();
}

[HttpPost("post")]
public async Task<ActionResult> Post(Mensajes mensaje)
{
    context.Add(mensaje);
    await context.SaveChangesAsync();
    return Ok();
}

[HttpPut]
public async Task<ActionResult> update(Mensajes request)
{
    var mensaje = await context.Mensajes.FirstOrDefaultAsync(r => r.Mensajeld == request.Mensajeld);

    mensaje.Mensaje = request.Mensaje;
    context.Update(mensaje);
    await context.SaveChangesAsync();
    return Ok();
}

[HttpDelete("delete/{id:int}")]
public async Task<ActionResult> delete(int id)
{
    var mensaje = await context.Mensajes.FirstOrDefaultAsync(r => r.Mensajeld == id);
    if (mensaje is null)
    {
        return NotFound();
    }
    context.Remove(mensaje);
    await context.SaveChangesAsync();
    return Ok();
}
}
}

```

```
using EntityPosgardo.infraestructura;
using EntityPosgardo.modelo;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EntityPosgardo.Controllers
{
    [ApiController]
    [Route("api/roles")]
    public class RolesControllers : ControllerBase
    {
        private readonly InfraestructuraDbContext context;

        public RolesControllers(InfraestructuraDbContext context)
        {
            this.context = context;
        }

        [HttpGet("get")]
        public async Task<IEnumerable<Roles>> Get()
        {
            return await context.Roles.ToListAsync();
        }

        [HttpPost("post")]
        public async Task<ActionResult> Post(Roles roles)
        {
            context.Add(roles);
            await context.SaveChangesAsync();
            return Ok();
        }

        [HttpPut]
        public async Task<ActionResult> update(Roles request)
        {
            var rol = await context.Roles.FirstOrDefaultAsync(r => r.RolId == request.RolId);

            rol.Nombre = request.Nombre;
            context.Update(rol);
            await context.SaveChangesAsync();
            return Ok();
        }

        [HttpDelete("delete/{id:int}")]
        public async Task<ActionResult> delete(int id)
        {
            var roles = await context.Roles.FirstOrDefaultAsync(r => r.RolId == id);
```

```

        if (roles is null)
        {
            return NotFound();
        }
        context.Remove(roles);
        await context.SaveChangesAsync();
        return Ok();
    }
}
}

```

```

using EntityPosgardo.infraestructura;
using EntityPosgardo.modelo;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;

namespace EntityPosgardo.Controllers
{
    [ApiController]
    [Route("api/usuarios")]
    public class UsuariosControllers : ControllerBase
    {
        private readonly InfraestructuraDbContext context;

        public UsuariosControllers(InfraestructuraDbContext context)
        {
            this.context = context;
        }

        [HttpGet("get")]
        public async Task<IEnumerable<Usuarios>> Get()
        {
            return await context.Usuarios.ToListAsync();
        }

        [HttpPost("post")]
        public async Task<ActionResult> Post(Usuarios user)
        {
            context.Add(user);
            await context.SaveChangesAsync();
            return Ok();
        }
    }
}

```

```
[HttpPut]
public async Task<ActionResult> update(Usuarios request)
{
    var user = await context.Usuarios.FirstOrDefaultAsync(r => r.UserId == request.UserId);

    if (user == null)
        return BadRequest("Usuario no encontrado");

    user.Usuario = request.Usuario;
    user.Name = request.Name;
    user.Edad = request.Edad;
    user.Contraseña = request.Contraseña;
    context.Update(user);
    await context.SaveChangesAsync();
    return Ok();
}

[HttpDelete("delete/{id:int}")]
public async Task<ActionResult> delete(int id)
{
    var usuarios = await context.Usuarios.FirstOrDefaultAsync(r => r.UserId == id);
    if (usuarios is null)
    {
        return NotFound();
    }
    context.Remove(usuarios);
    await context.SaveChangesAsync();
    return Ok();
}
}
```

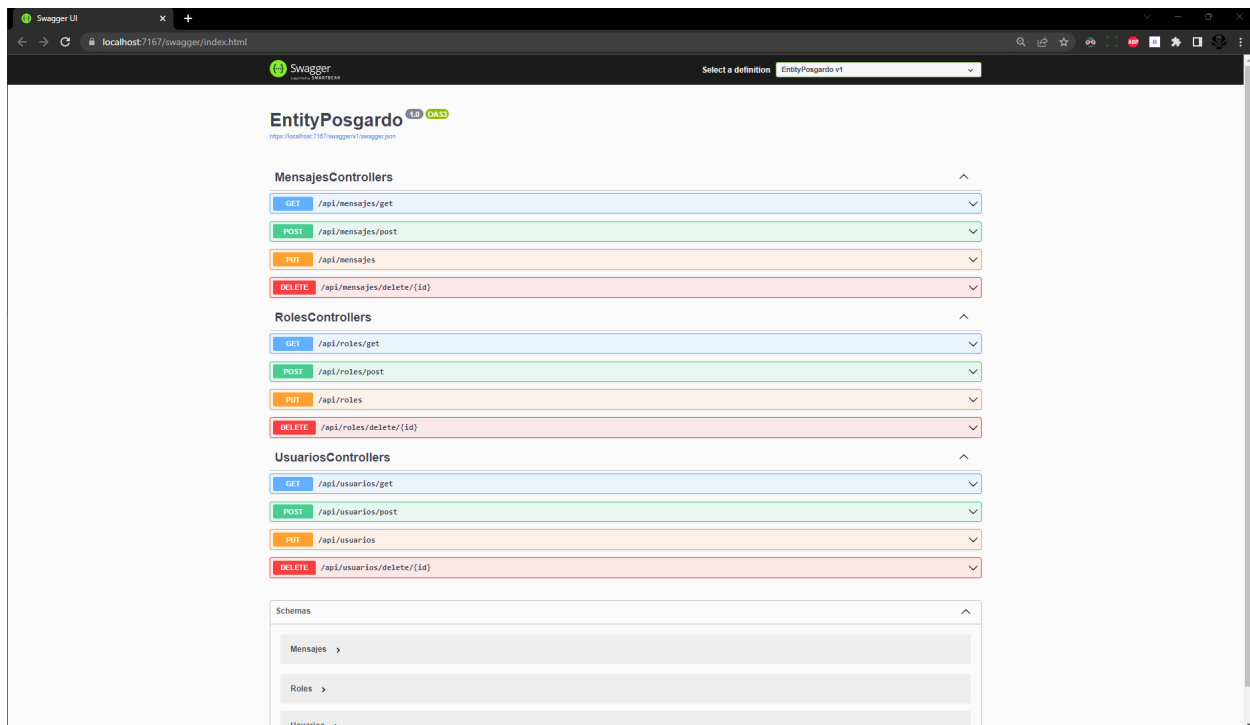



Figura 11

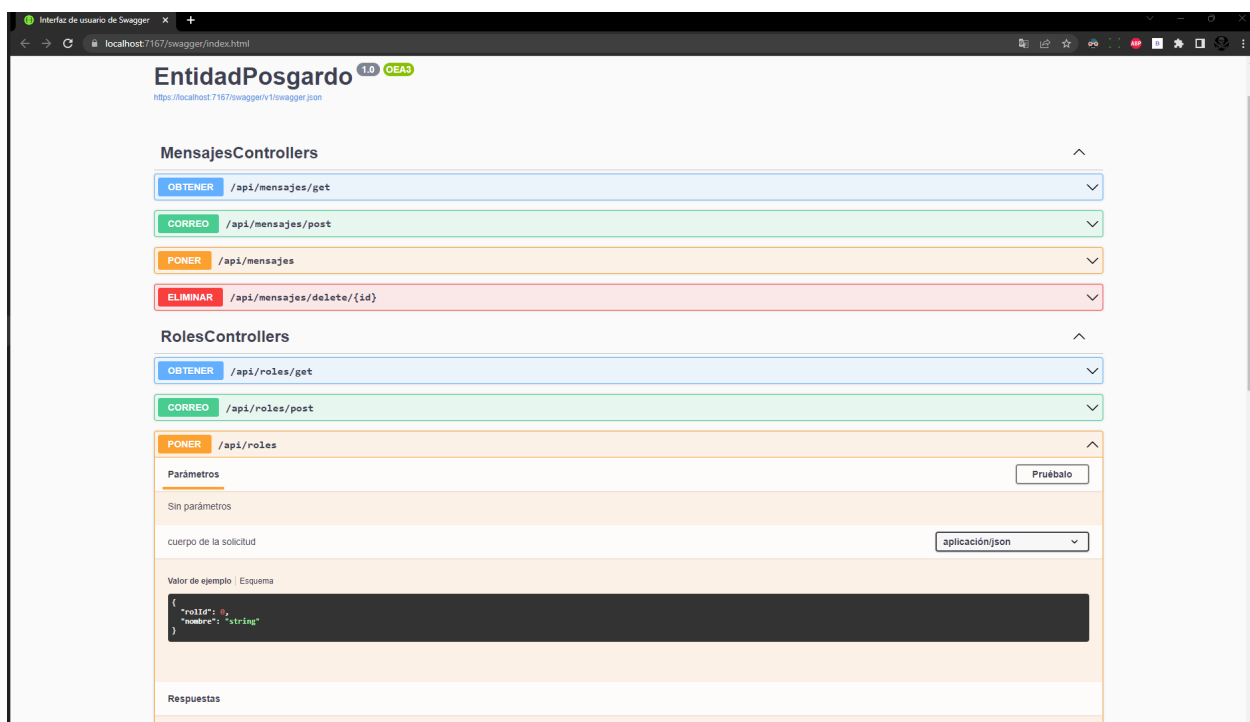


Figura 12

Bibliografía

Hidayat, A. (10 de abril de 2019). *MEDIUM*. Obtenido de ASP.NET Core, Entity Framework Core con PostgreSQL Code First: <https://faun.pub/asp-net-core-entity-framework-core-with-postgresql-code-first-d99b909796d7>

Phaiza. (23 de mayo de 2022). *LogRocket*. Obtenido de API CRUD REST con Node.js, Express y PostgreSQL: <https://blog.logrocket.com/crud-rest-api-node-js-express-postgresql/>