# Creación nuevo proyecto

* En el menú Archivo, seleccione Nuevo>Proyecto.
* Escriba API web en el cuadro de búsqueda.
* Seleccione la plantilla API web de ASP.NET Core y seleccione Siguiente.
* En el cuadro de diálogo Configurar el nuevo proyecto, asigne al proyecto el nombre TodoApi y seleccione Siguiente.
* En el cuadro de diálogo Información adicional:
  + Confirme que el Marco es .NET 9.0 (Soporte en un plazo estándar).
  + Confirme que la casilla Habilitar compatibilidad con OpenAPI está activada.
  + Confirme que la casilla Usar controladores (desactivar para usar API mínimas) está activada.
  + Seleccione Crear.

# Adición de un paquete NuGet

Se debe agregar un paquete NuGet para admitir la base de datos que se usa en este tutorial.

* En el menú Herramientas, seleccione Administrador de paquetes NuGet > Administrar paquetes NuGet para la solución.
* Seleccione la pestaña Examinar.
* Escriba Microsoft.EntityFrameworkCore.InMemory en el cuadro de búsqueda y, después, seleccione Microsoft.EntityFrameworkCore.InMemory.
* Active la casilla Proyecto en el panel derecho y, después, seleccione Instalar.

# Crear clases y DTOs

Creamos una carpeta “Models” y dentro dos subcarpetas “entities” y “DTOs”

public class Animals

{

public long Id { get; set; }

public string Name { get; set; }

public string Race { get; set; }

public DateTime? birthdate { get; set; }

public string Sex { get; set; }

public string OwnerDni { get; set; }

}

public class Attentions

{

public long Id { get; set; }

public int AnimalId { get; set; }

public string ReasonForConsultation { get; set; }

public string Treatment { get; set; }

public string Medications { get; set; }

public DateTime Date { get; set; }

}

public class Owners

{

public int Dni { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

}

Los DTOs contienen lo mismo.

# Incorporación de un contexto de base de datos

* Haga clic con el botón derecho en la carpeta Models y seleccione Agregar>Clase. Asigne a la clase el nombre VetContext y haga clic en Agregar.

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using TpAPIRest\_Controladores.Models.Entities;

namespace TpAPIRestControladores.Models.Entities;

public class TodoContext : DbContext

{

public TodoContext(DbContextOptions<TodoContext> options)

: base(options)

{

}

public DbSet<Animals> Animals { get; set; } = null!;

public DbSet<Attentions> Attentions { get; set; } = null!;

public DbSet<Owners> Owners { get; set; } = null!;

}

# Registro del contexto de base de datos

En ASP.NET Core, los servicios (como el contexto de la base de datos) deben registrarse con el contenedor de [inserción de dependencias (DI)](https://learn.microsoft.com/es-es/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-9.0). El contenedor proporciona el servicio a los controladores.

Actualice Program.cs con el siguiente código:

using Microsoft.EntityFrameworkCore;

using TpAPIRestControladores.Models.Entities;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<VetContext>(opt =>

opt.UseInMemoryDatabase("VetControl"));

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

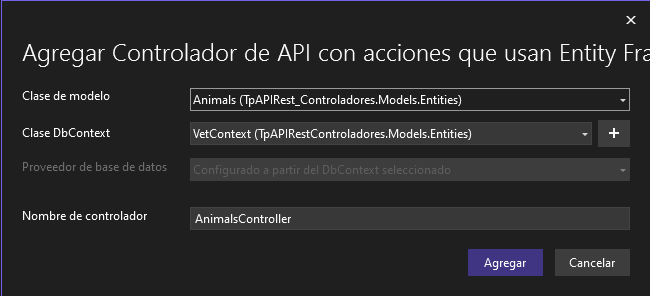
app.UseAuthorization();

app.MapControllers();

app.Run();

# Scaffolding controladores

* Haga clic con el botón derecho en la carpeta Controllers.
* Seleccione Agregar>New Scaffolded Item.
* Seleccione Controlador de API con acciones mediante Entity Framework y, después, seleccione Agregar.







# Actualización del método create de PostAnimals, PostOwners, PostAttentions

// POST: api/Animals

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Animals>> PostAnimals(Animals animals)

{

\_context.Animals.Add(animals);

await \_context.SaveChangesAsync();

// return CreatedAtAction("GetAnimals", new { id = animals.Id }, animals);

return CreatedAtAction(nameof(GetAnimals), new { id = animals.Id }, animals);

}

—----------------------------------------------------------------------------------------------------------------------------

// POST: api/Owners

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Owners>> PostOwners(Owners owners)

{

\_context.Owners.Add(owners);

await \_context.SaveChangesAsync();

// return CreatedAtAction("GetOwners", new { id = owners.Id }, owners);

return CreatedAtAction(nameof(GetOwners), new { id = owners.Id }, owners);

}

—----------------------------------------------------------------------------------------------------------------------------

// POST: api/Attentions

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Attentions>> PostAttentions(Attentions attentions)

{

\_context.Attentions.Add(attentions);

await \_context.SaveChangesAsync();

// return CreatedAtAction("GetAttentions", new { id = attentions.Id }, attentions);

return CreatedAtAction(nameof(GetAttentions), new { id = attentions.Id }, attentions);

}

El método [CreatedAtAction](https://learn.microsoft.com/es-es/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.createdataction) realiza las acciones siguientes:

* Devuelve un [código de estado HTTP 201](https://developer.mozilla.org/docs/Web/HTTP/Status/201) cuando se ha ejecutado correctamente. HTTP 201 es la respuesta estándar de un método HTTP POST que crea un recurso en el servidor.
* Agrega un encabezado [Location](https://developer.mozilla.org/docs/Web/HTTP/Headers/Location) a la respuesta. El encabezado Location especifica el [URI](https://developer.mozilla.org/docs/Glossary/URI) de la tarea pendiente recién creada. Para obtener más información, consulte [10.2.2 201 creado](https://www.rfc-editor.org/rfc/rfc9110.html#section-10.2.2).
* Hace referencia a la acción GetAnimals (O la clase que sea) para crear el identificador URI del encabezado Location. La palabra clave nameof de C# se usa para evitar que se codifique de forma rígida el nombre de acción en la llamada a CreatedAtAction.

# Prueba de PostAnimals

* Seleccione Ver>Otras ventanas>Explorador de puntos de conexión.
* Haga clic con el botón derecho en el punto de conexión POST y seleccione Generar solicitud.

JSON:

{

"Name" : "Patroclo",

"Race" : "Blacky",

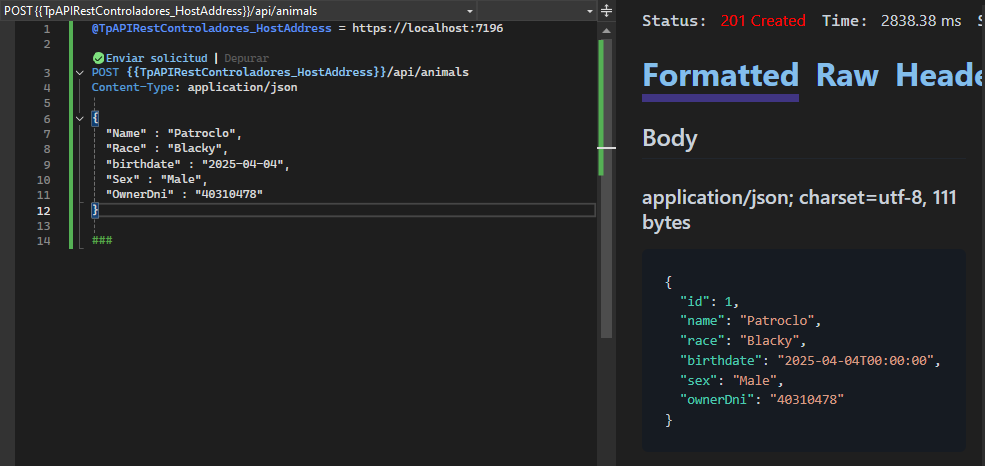
"birthdate" : "2025-04-04",

"Sex" : "Male",

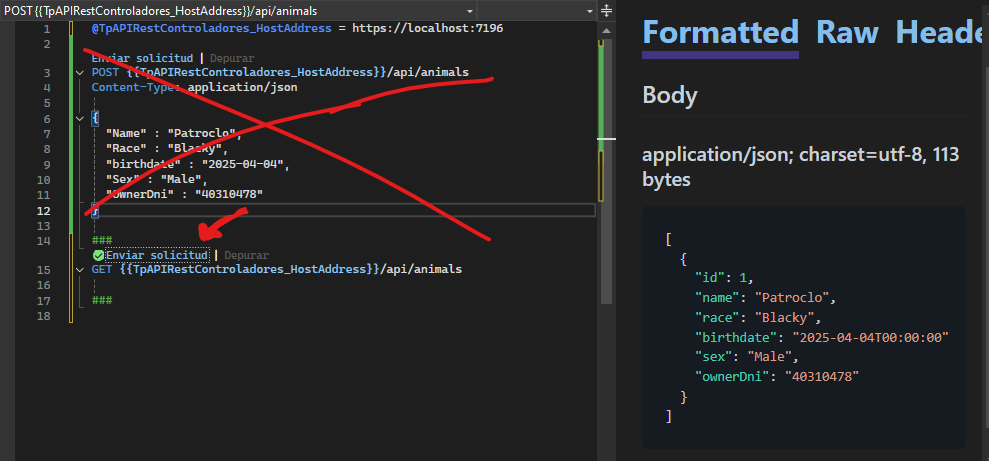
"OwnerDni" : "40310478"

}

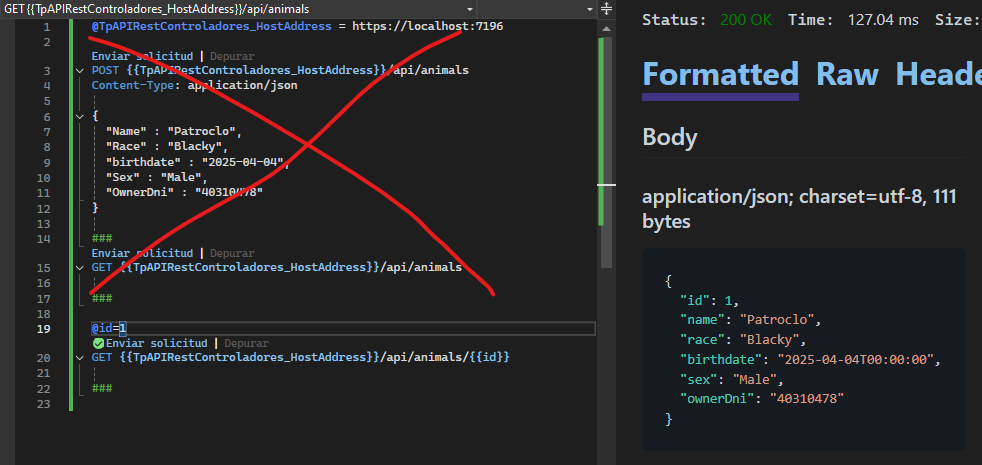
Seleccione el vínculo Enviar solicitud situado encima de la línea de solicitud POST.



# Prueba del URI del encabezado de ubicación



Con id



# Enrutamiento y rutas URL

En AnimalsController:

[Route("api/[controller]")]

[ApiController]

public class AnimalsController : ControllerBase

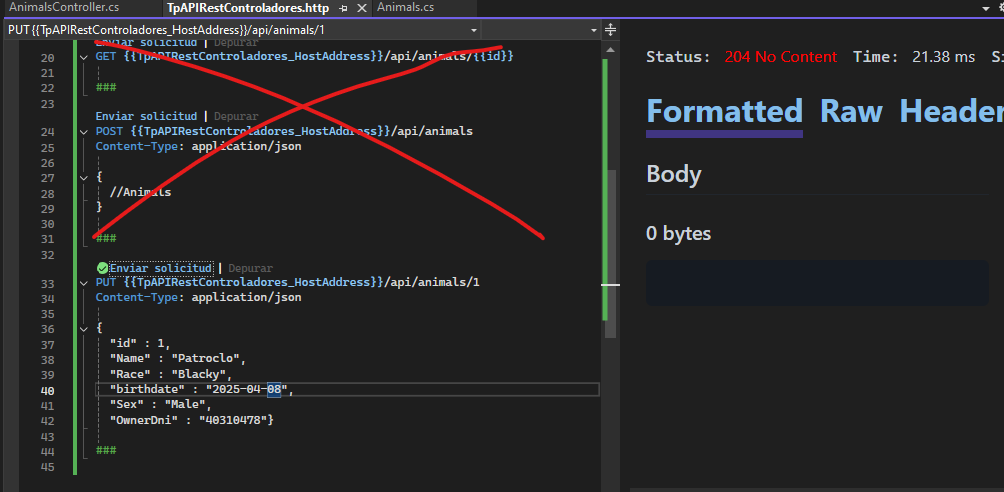
cambiamos por:

[Route("api/Animals")]

[ApiController]

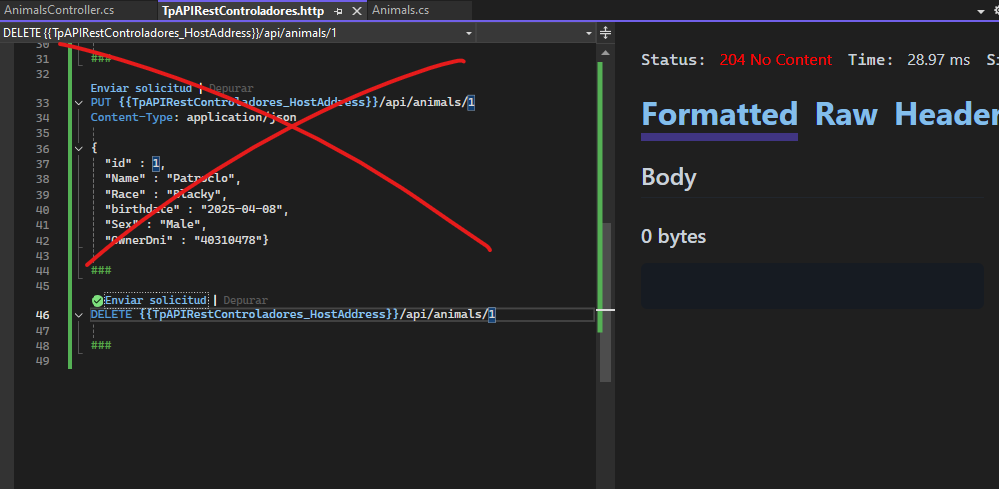
public class AnimalsController : ControllerBase

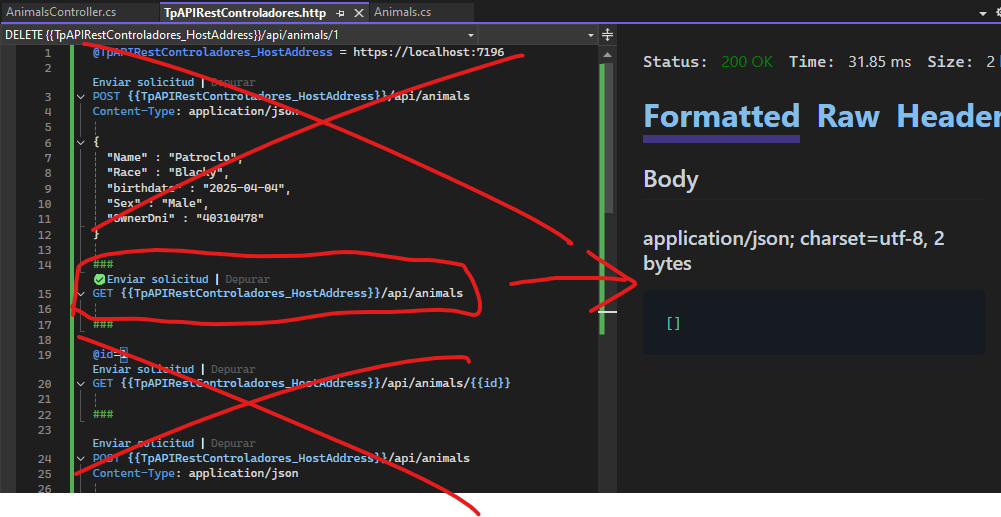
# Prueba del método PutAnimals



# 

# Prueba del método DeleteAnimals





# Buscar todos los Attentions de un animal en específico

Agregar dos Attentions:

"AnimalId": 101,

"ReasonForConsultation": "Infección ocular",

"Treatment": "Limpieza y gotas antibióticas",

"Medications": "Tobramicina",

"Date": "2024-04-03"

"AnimalId": 101,

"ReasonForConsultation": "Dolor en la pata trasera",

"Treatment": "Reposo y vendaje",

"Medications": "Ibuprofeno veterinario",

"Date": "2024-04-04"

y pruebo el controlador

// GET: api/Attentions/ByAnimal/{idAnimal}

[HttpGet("ByAnimal/{idAnimal}")]

public async Task<ActionResult<IEnumerable<Attentions>>> GetAttentionsOfAnimal(long idAnimal)

{

var attentions = await \_context.Attentions

.Where(a => a.AnimalId == idAnimal)

.ToListAsync();

if (!attentions.Any())

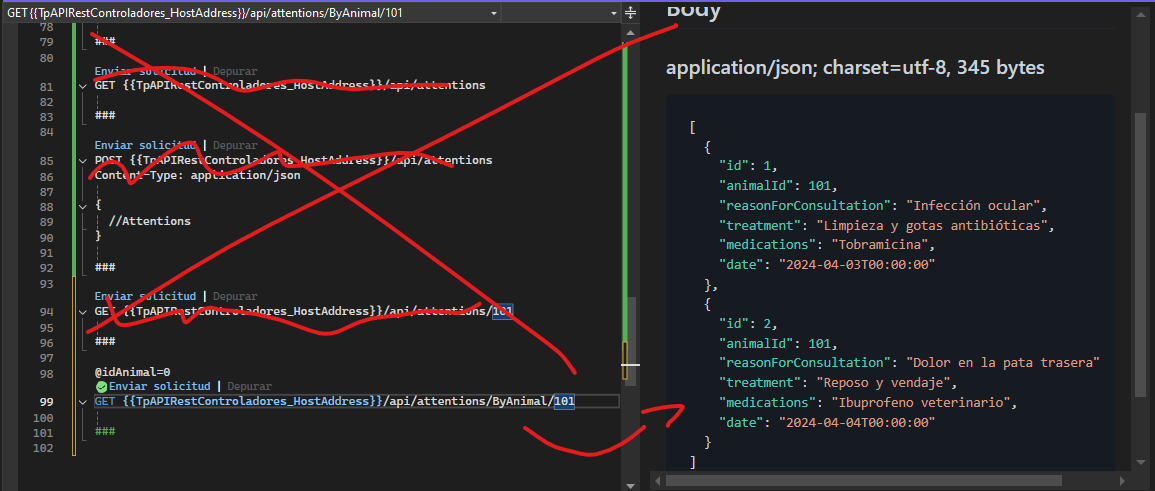
{

return NotFound();

}

return attentions;

}



# Buscar todos los Attentions de un Owner en específico

Creamos a Patroclo y:

{

"Name" : "Brunella",

"Race" : "Criollo",

"birthdate" : "2022-04-04",

"Sex" : "Female",

"OwnerDni" : "40310478"

}

