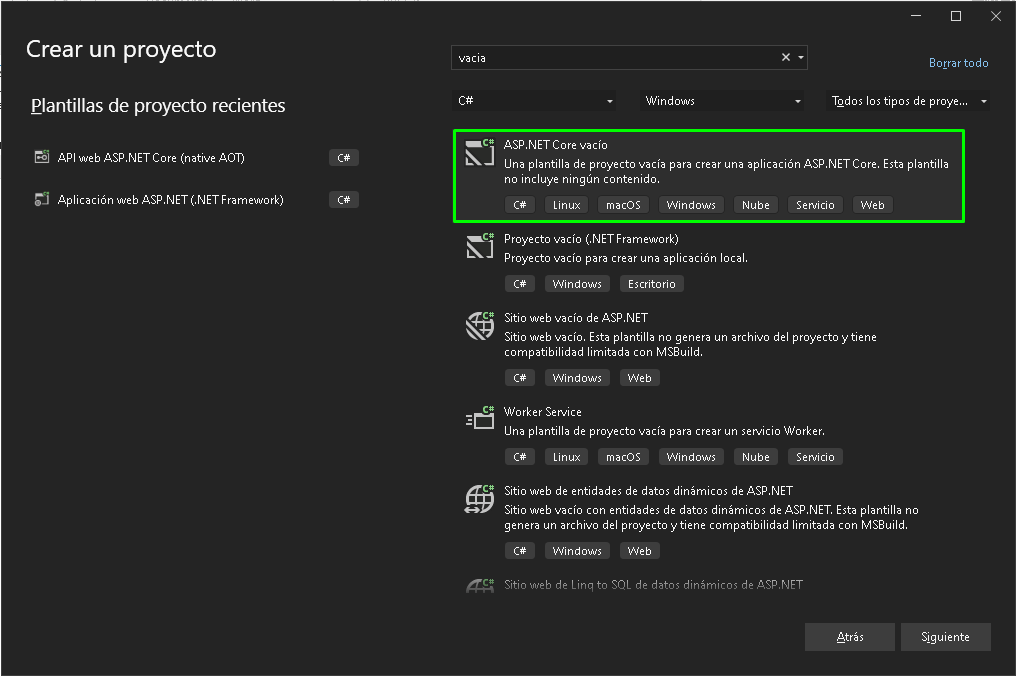
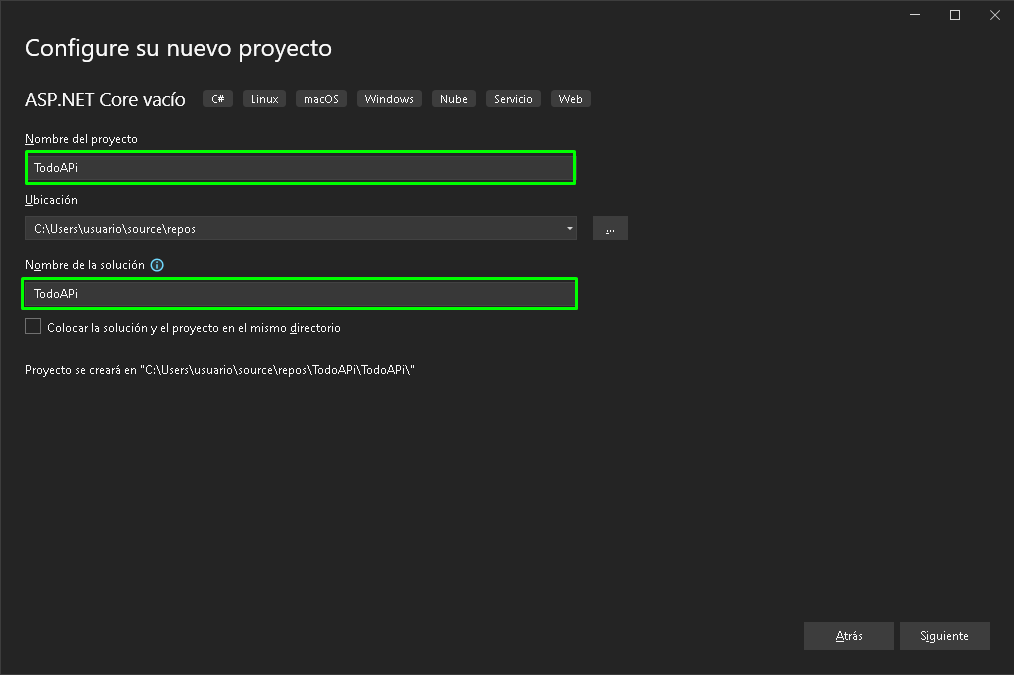
Elegimos el tipo de proyecto

## Creación del proyecto

Elegimos ASP.NET Core vacío.

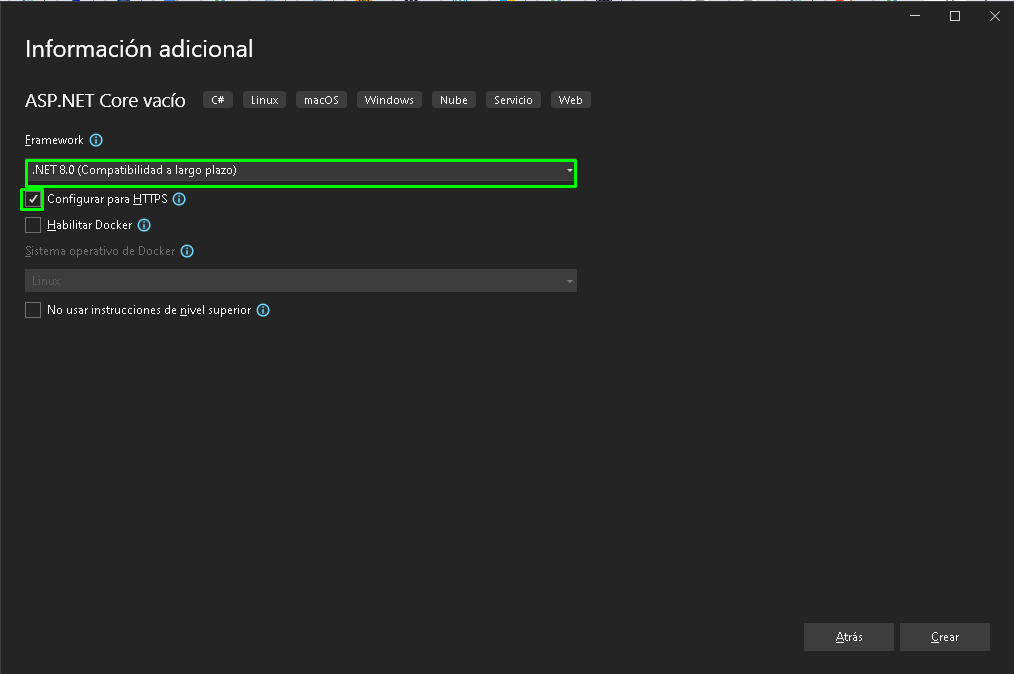


Escribimos el nombre del proyecto **TodoApi**



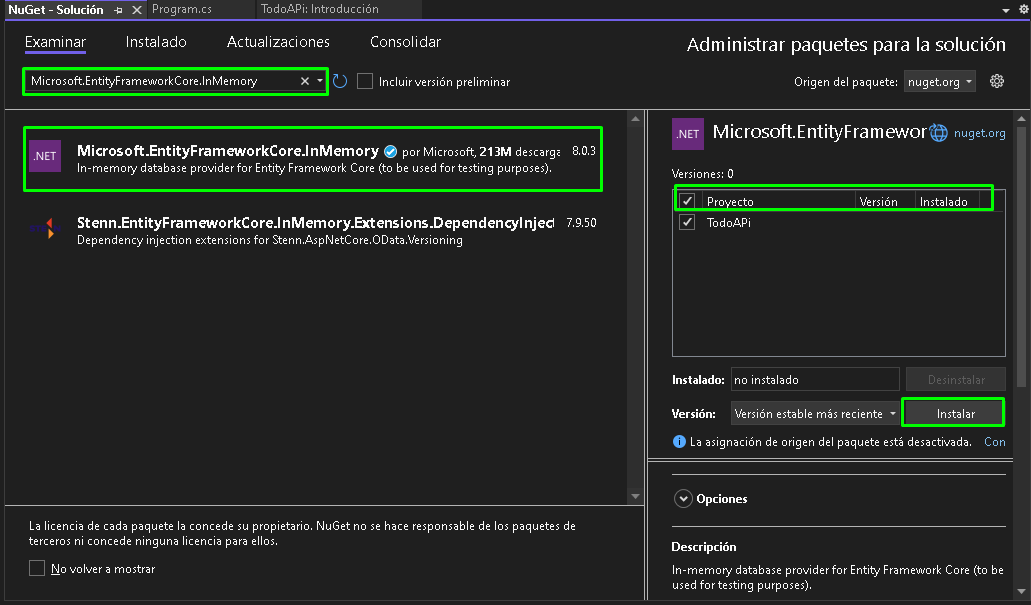
En el cuadro de diálogo **Información adicional**:

* Seleccione **.NET 8.0 (Compatibilidad a largo plazo)**
* Anule la sección de **No usar instrucciones de nivel superior**.
* Seleccione **Crear**

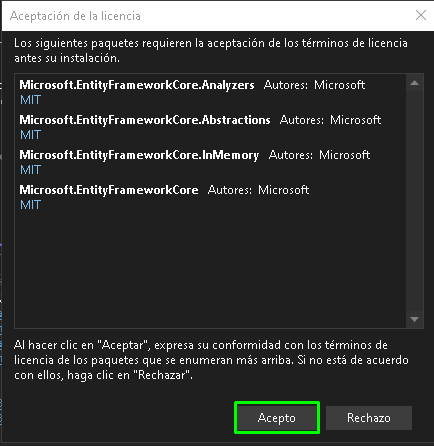
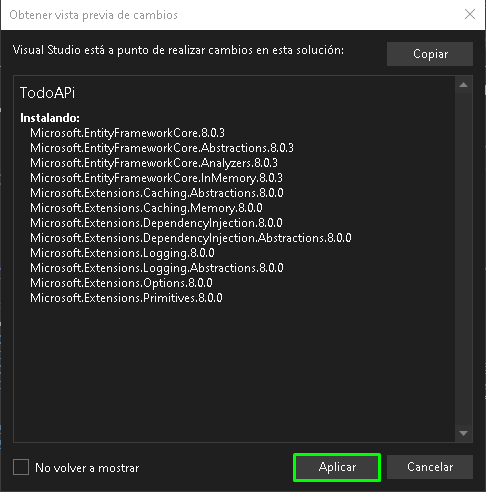


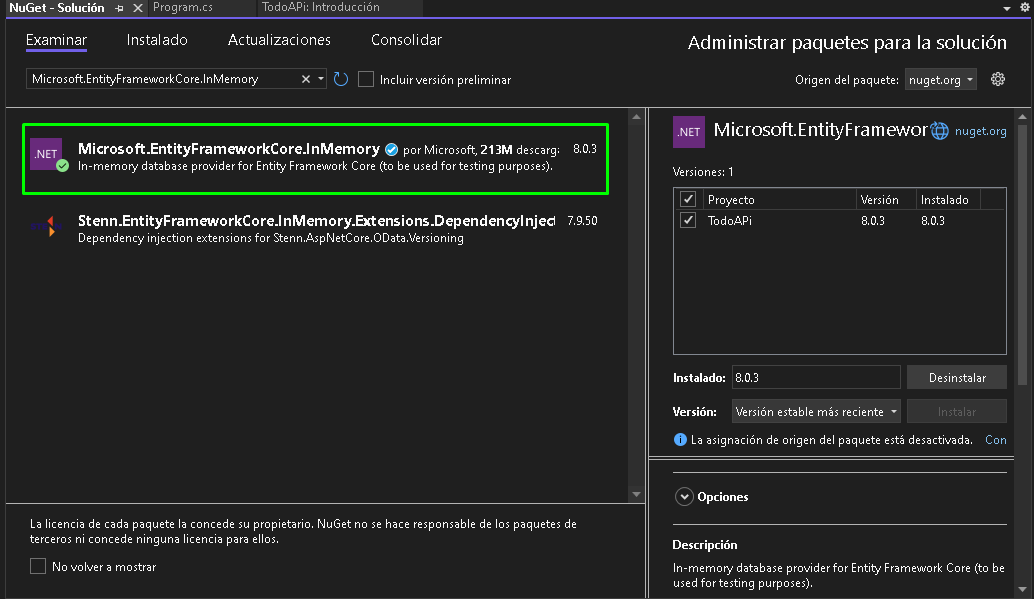
**Agregando paquetes nugget**

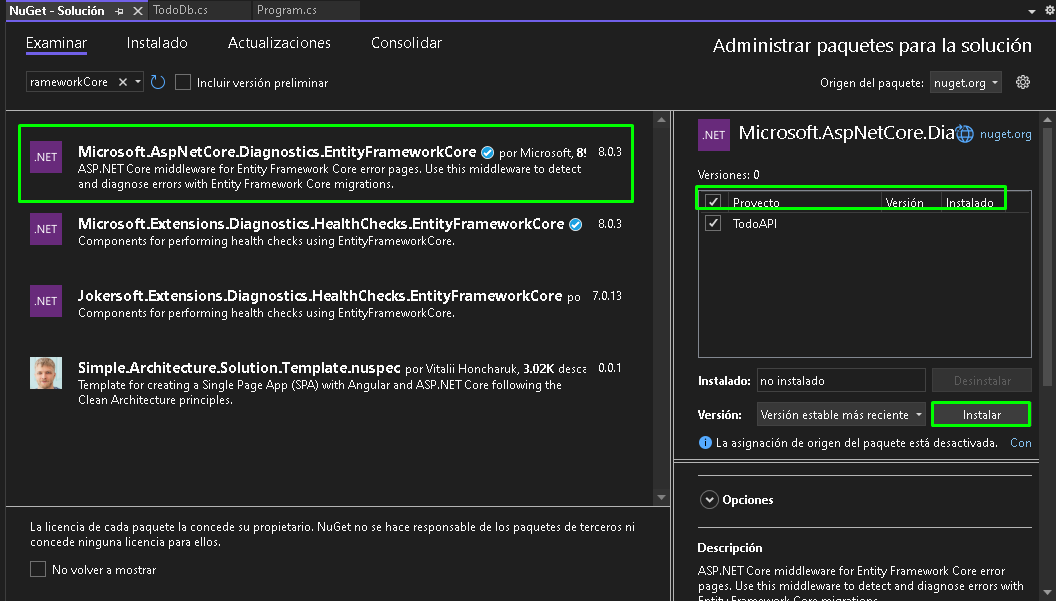
* En el menú **Herramientas**, seleccione **Administrador de paquetes NuGet > Administrar paquetes NuGet para la solución**.
* Seleccione la pestaña **Examinar**.
* Escriba **Microsoft.EntityFrameworkCore.InMemory** en el cuadro de búsqueda y, después, seleccione Microsoft.EntityFrameworkCore.InMemory.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

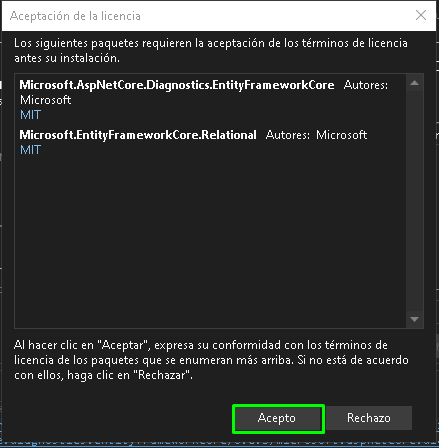
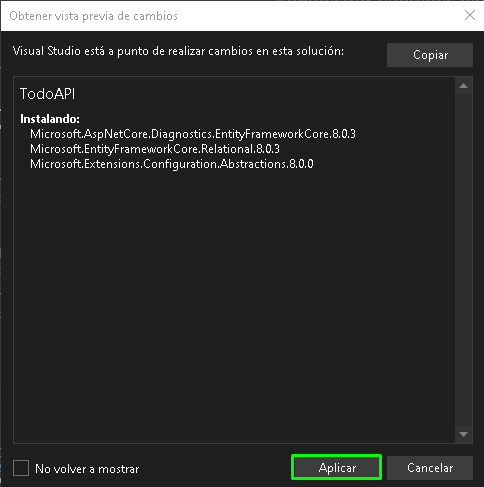


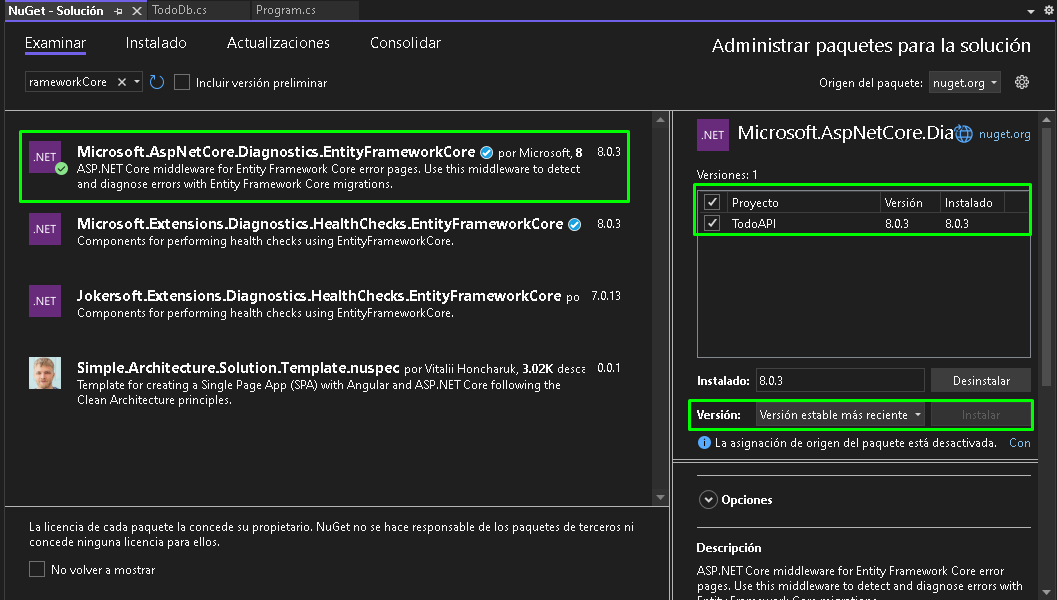
Clic en instalar y se mostraran las siguientes pantallas.









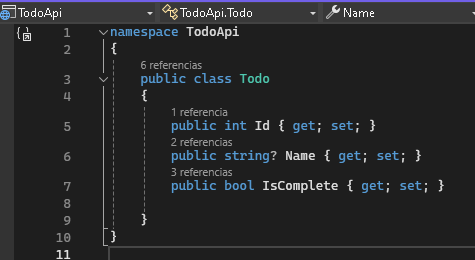


Una vez concluida la instalación realizamos los siguientes pasos.

Se crearán 2 archivos de clase.

**Clases de contexto de base de datos y modelo**

Se crea un archivo llamado TodoDb.cs con el código siguiente:

namespace TodoApi

{

public class Todo

{

public int Id { get; set; }

public string? Name { get; set; }

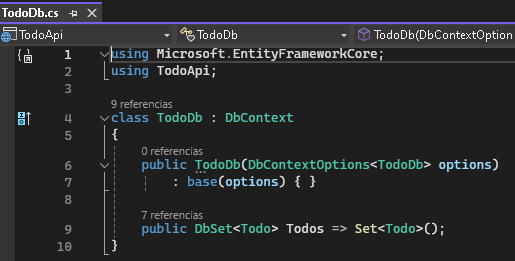
public bool IsComplete { get; set; }

}

}

El código anterior crea el modelo para esta aplicación. Un modelo es una clase que representa los datos que la aplicación administra.

Se crea un archivo llamado TodoDb.cs con el código siguiente:

using Microsoft.EntityFrameworkCore;

using TodoApi;

class TodoDb : DbContext

{

public TodoDb(DbContextOptions<TodoDb> options)

: base(options) { }

public DbSet<Todo> Todos => Set<Todo>();

}

Se reemplaza el contenido del archivo Program.cs por el código siguiente:

using Microsoft.EntityFrameworkCore;

using TodoApi;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<TodoDb>(opt => opt.UseInMemoryDatabase("TodoList"));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

var app = builder.Build();

var todoItems = app.MapGroup("/todoitems");

todoItems.MapGet("/", async (TodoDb db) =>

await db.Todos.ToListAsync());

todoItems.MapGet("/complete", async (TodoDb db) =>

await db.Todos.Where(t => t.IsComplete).ToListAsync());

todoItems.MapGet("/{id}", async (int id, TodoDb db) =>

await db.Todos.FindAsync(id)

is Todo todo

? Results.Ok(todo)

: Results.NotFound());

todoItems.MapPost("/", async (Todo todo, TodoDb db) =>

{

db.Todos.Add(todo);

await db.SaveChangesAsync();

return Results.Created($"/todoitems/{todo.Id}", todo);

});

todoItems.MapPut("/{id}", async (int id, Todo inputTodo, TodoDb db) =>

{

var todo = await db.Todos.FindAsync(id);

if (todo is null) return Results.NotFound();

todo.Name = inputTodo.Name;

todo.IsComplete = inputTodo.IsComplete;

await db.SaveChangesAsync();

return Results.NoContent();

});

todoItems.MapDelete("/{id}", async (int id, TodoDb db) =>

{

if (await db.Todos.FindAsync(id) is Todo todo)

{

db.Todos.Remove(todo);

await db.SaveChangesAsync();

return Results.NoContent();

}

return Results.NotFound();

});

app.Run();

El código resaltado siguiente agrega el contexto de base de datos al contenedor de inserción de dependencias (ID) y permite mostrar excepciones relacionadas con la base de datos:

var builder = WebApplication.CreateBuilder(args);

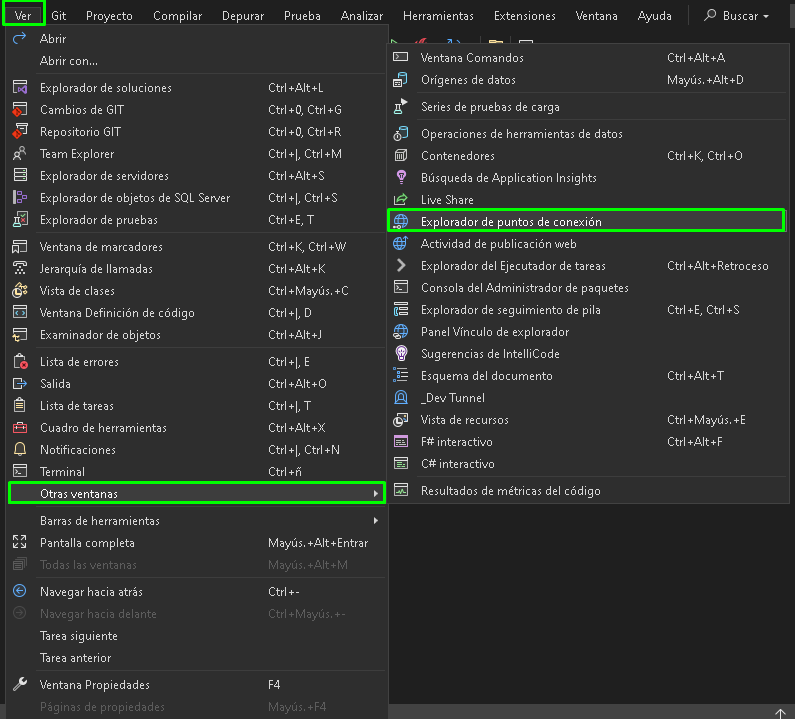
builder.Services.AddDbContext<TodoDb>(opt => opt.UseInMemoryDatabase("TodoList"));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();

var app = builder.Build();

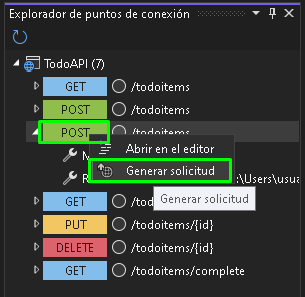
## Publicación de datos

* Seleccione **Ver**>**Otras ventanas**>**Explorador de puntos de conexión**.

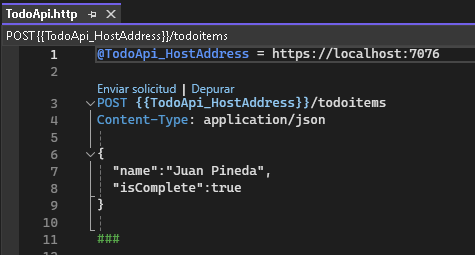


Se visualizará la siguiente ventana realizamos lo siguiente

* Haga clic con el botón derecho en el punto de conexión **POST** y seleccione **Generar solicitud**.

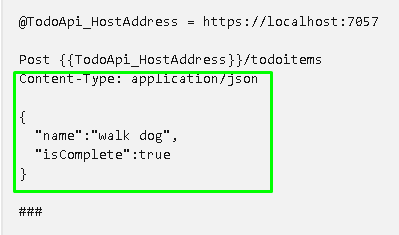


Se creará el siguiente archivo TodoApi.http

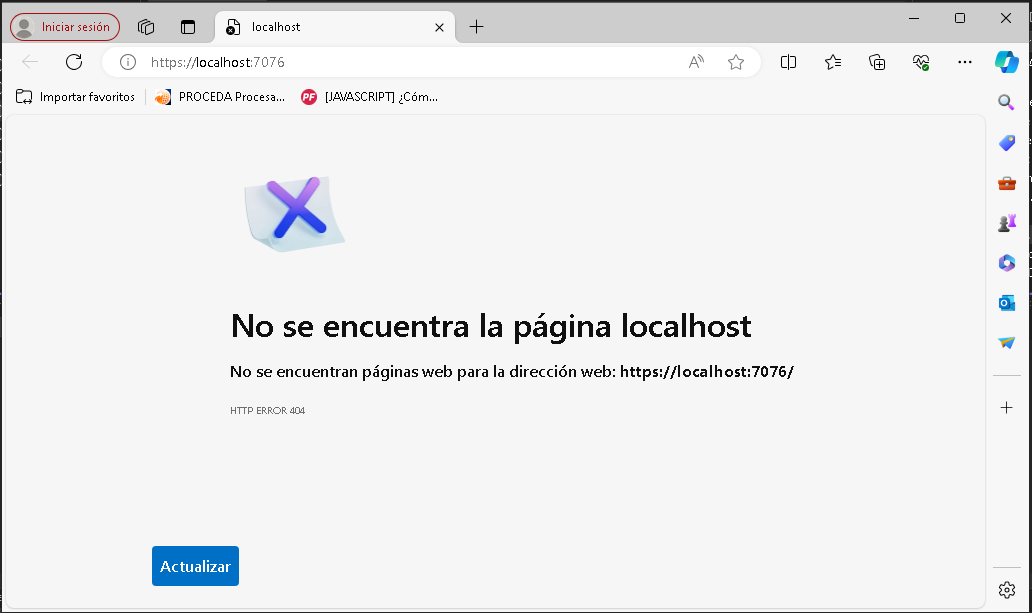


Explicación del contenido del archivo.

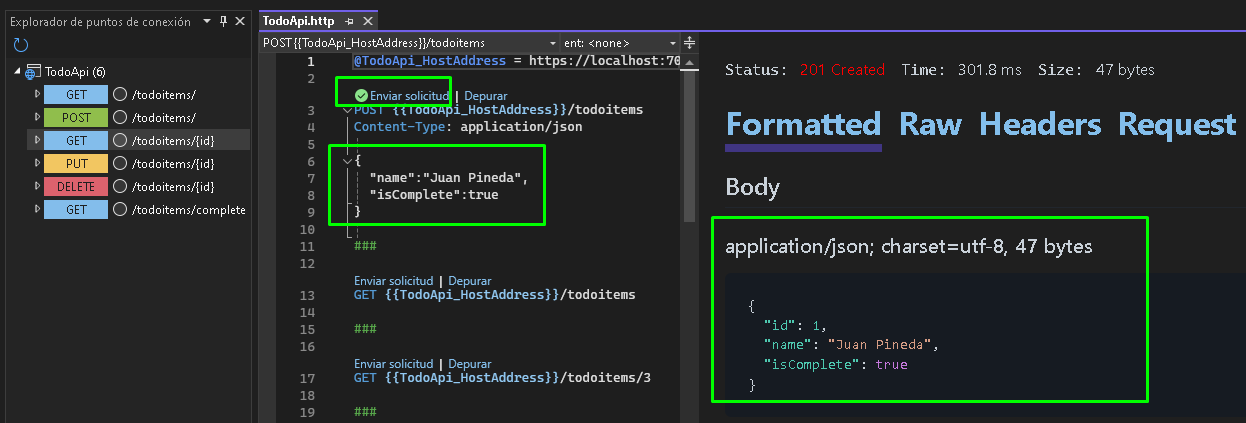
* + La primera línea crea una variable que se usa para todos los puntos de conexión.
  + La siguiente línea define una solicitud POST.
  + La línea triple hashtag (###) es un delimitador de solicitud: lo que viene después es para una solicitud diferente.
* La solicitud POST necesita encabezados y un cuerpo. Para definir esas partes de la solicitud, agregue las líneas siguientes inmediatamente después de la línea de solicitud POST:



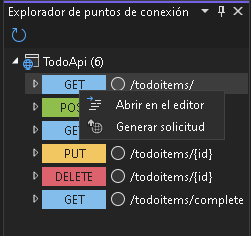
Ejecutamos la aplicación Control + F5, la cosultaremos desde el archivo TodoApi.hhtp con sus diferentes metodos



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



Realizamos clic con cada uno de los puntos de conexión.



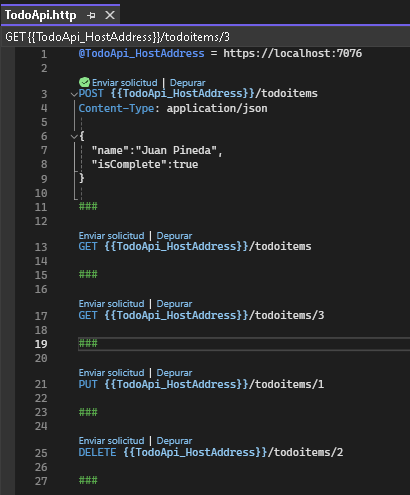
El archivo debe de quedar de la siguiente manera. TodoApi.http

@TodoApi\_HostAddress = https://localhost:7076

POST {{TodoApi\_HostAddress}}/todoitems

Content-Type: application/json

{

 "name":"Juan Pineda",

"isComplete":true

}

###

GET {{TodoApi\_HostAddress}}/todoitems

###

GET {{TodoApi\_HostAddress}}/todoitems/3

###

PUT {{TodoApi\_HostAddress}}/todoitems/1

###

DELETE {{TodoApi\_HostAddress}}/todoitems/2

###

Contenido

|  |  |  |  |
| --- | --- | --- | --- |
| **API** | **Descripción** | **Cuerpo de la solicitud** | **Cuerpo de la respuesta** |
| GET /todoitems | Obtener todas las tareas pendientes | None | Matriz de tareas pendientes |
| GET /todoitems/complete | Obtener tareas pendientes completadas | None | Matriz de tareas pendientes |
| GET /todoitems/{id} | Obtener un elemento por identificador | None | Tarea pendiente |
| POST /todoitems | Incorporación de un nuevo elemento | Tarea pendiente | Tarea pendiente |
| PUT /todoitems/{id} | Actualizar un elemento existente | Tarea pendiente | None |
| DELETE /todoitems/{id} | Eliminar un elemento | None | None |

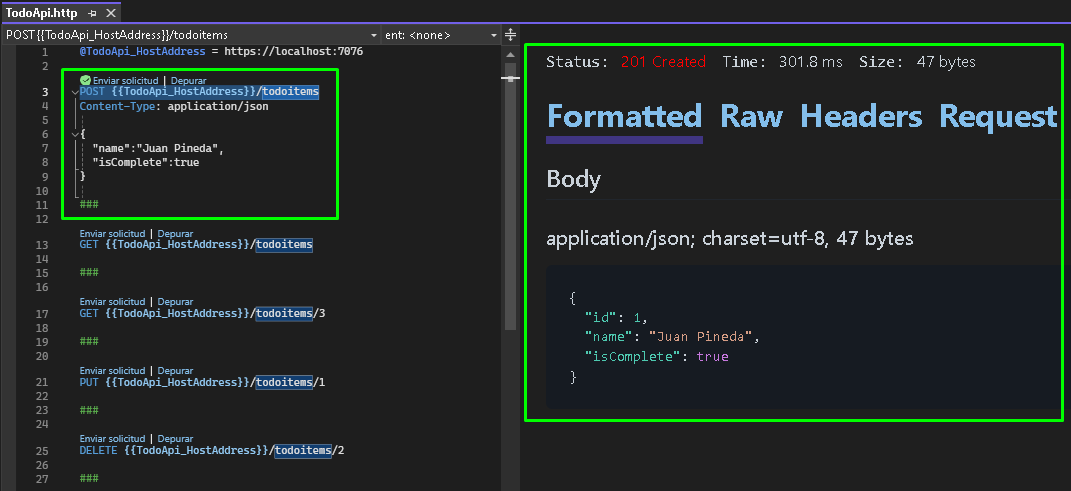
**Puntos de conexión GET**

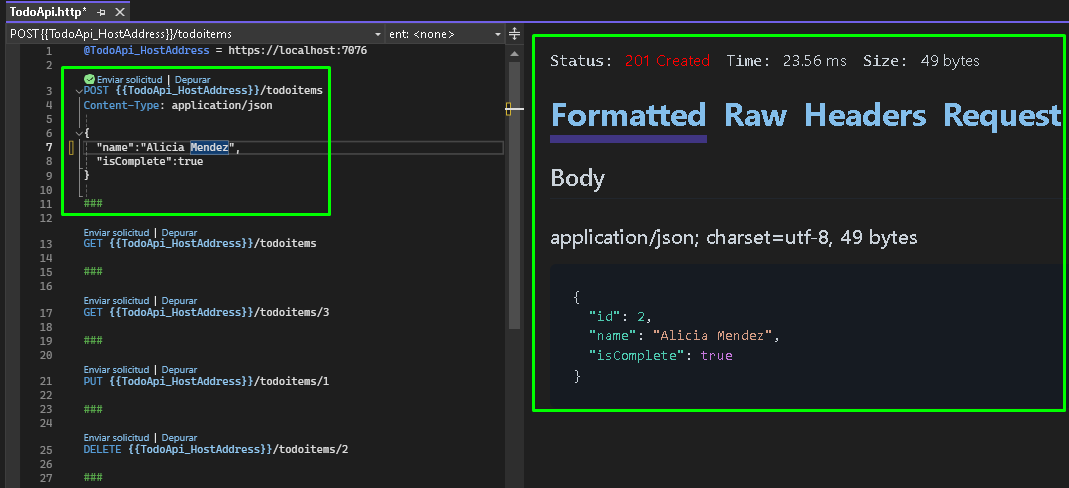
La aplicación de ejemplo implementa varios puntos de conexión GET mediante la llamada a MapGet:

|  |  |  |  |
| --- | --- | --- | --- |
| API | **Descripción** | **Cuerpo de la solicitud** | **Cuerpo de la respuesta** |
| GET /todoitems | Obtener todas las tareas pendientes | None | Matriz de tareas pendientes |
| GET /todoitems/complete | Obtención de todas las tareas pendientes completadas | None | Matriz de tareas pendientes |
| GET /todoitems/{id} | Obtener un elemento por identificador | None | Tarea pendiente |

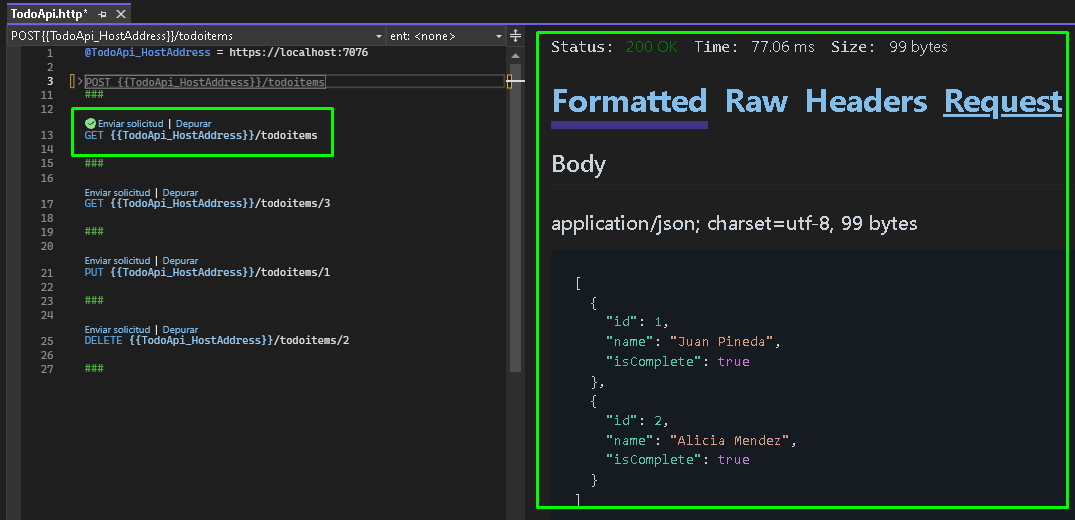
**Pruebas de los puntos de conexión**

POST enviamos dos distintos nombres para la prueba y se muestra a continuación

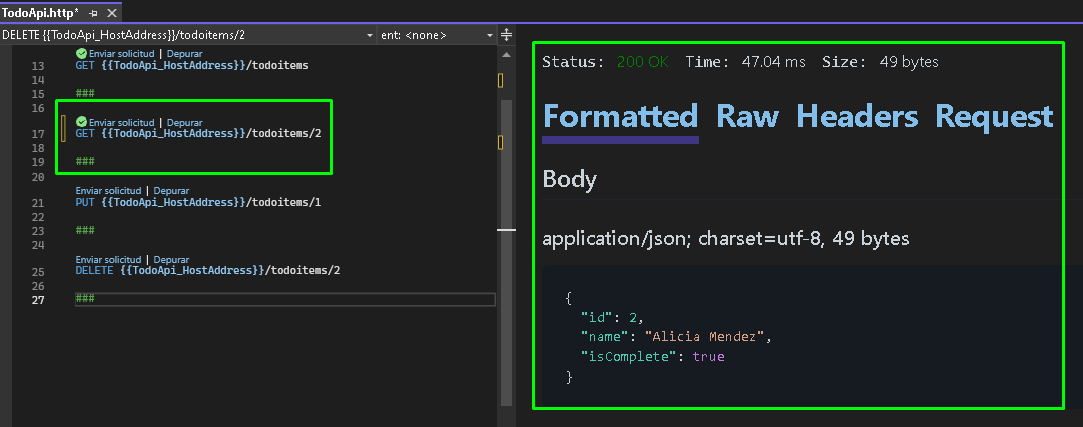




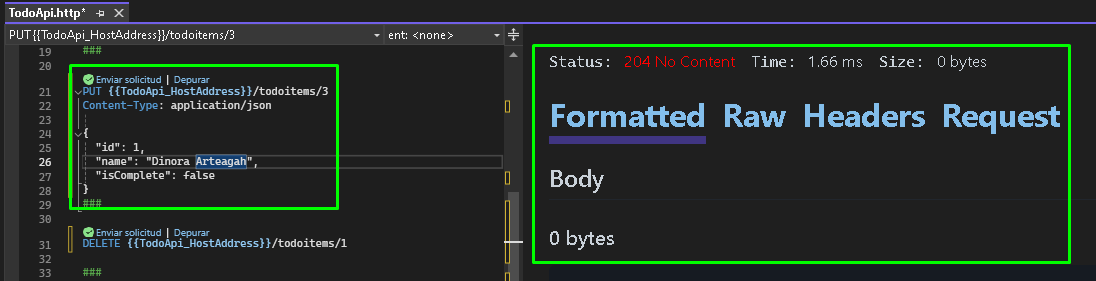
Get consultamos los 2 registros



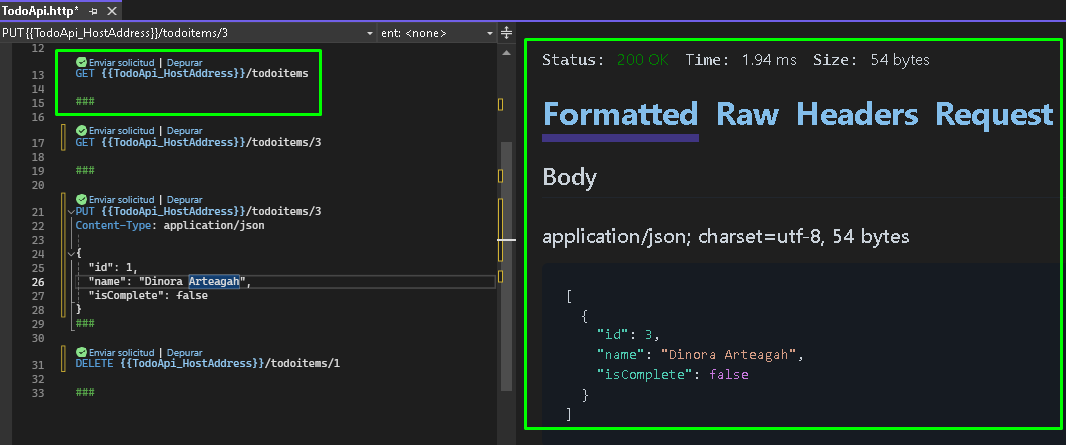
Get. Consultamos por el id=2



PUT se actualiza por completo el registro



Se consulta para ver el cambio



DELETE eliminamos los 2 registros

