HOW TO CREATE A CRUD CON WEB API EN ASP NET CORE

Proyecto ubicado en Maquina Virtual :

|  |
| --- |
| C:\www\7 Apis\WebApplication1 |

Create a AspNet Core project

Graphical user interface, text, application, email

Description automatically generated

Agregue un ConnectionStrings en AppSetting

"ConnectionStrings":

{

"ConexionTest": "**Server**=localhost;**Database**=CrudWebApi;User **ID**=crud;**password**=123456;**Trusted\_Connection**=true;**MultipleActiveResultSets**=true"

}

Graphical user interface, application

Description automatically generated

Agregue un modelo llamado contacto.cs

|  |
| --- |
| public class Contacto  {  public int Id { get; set; }  [Required]  public string Nombre { get; set; }  public string Telefono { get; set; }  public string Direccion { get; set; }  public string Email { get; set; }  public bool Verificado { get; set; }  }// |

Graphical user interface, application, Word

Description automatically generated

Ahora cree una clase contexto

|  |
| --- |
| public class ContactoContexto: DbContext  {  public ContactoContexto(DbContextOptions <ContactoContexto> options):base(options)  {    }//constructor  //crear nuestro diseño  public DbSet<Contacto> ContactoItems { get; set; }  }// |

Graphical user interface, application, Word

Description automatically generated

Ahora abra la consola Nuget y corra el siguiente comando:

**add-migration MigracionInicial**

Graphical user interface, application

Description automatically generated

Este error se solucionado agregando en ConfigureServices la línea:

services.AddDbContext<**ContactoContexto**>(options => options.UseSqlServer(Configuration.GetConnectionString("ConexionTest")));

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<**ContactoContexto**>(options => options.UseSqlServer(Configuration.GetConnectionString("ConexionTest")));

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

}

Graphical user interface, text, application

Description automatically generated

|  |
| --- |
| Si sale error    Luego de la instalación |

En la consola corra nuevamente el comando :

**PM> add-migration MigracionInicial**

Y esto crea una carpeta Migrations

Graphical user interface, application

Description automatically generated

Clase MigracionInicial

Graphical user interface, text, application, email

Description automatically generated

Ahora para crear la base de datos, en la consola de Nuget corra el siguiente comando:

PM> update-database

Graphical user interface, text, application

Description automatically generated

Text, application

Description automatically generated

Ahora en SSMS debe aparecer la base de datos creada con el EntityFrameWork

A picture containing text

Description automatically generated

Agregar un controlador en blanco llamado contacto

Graphical user interface, application

Description automatically generated

Antes de crear nuestro primer metodo debemos llamar nuestro contexto para eso creamos una variable tipo contexto

|  |
| --- |
| namespace WebApplication1.Controllers  {  [Route("api/[Controller]")]  [ApiController]  public class ContactosController : Controller  {  private readonly ContactoContexto \_context;  public ContactosController(ContactoContexto contexto)  {  \_context = contexto;  }  }//end controller  } |

Graphical user interface, text, application

Description automatically generated

Ahora creamos nuestro primer método HTTPGET asincrono **GetContactoItems**

|  |
| --- |
| [HttpGet]  public async Task<ActionResult<**IEnumerable<Contacto**>>> GetContactoItems()  {  return await \_context.ContactoItems.ToListAsync();  } |

Graphical user interface, application

Description automatically generated

Quien es \_context.ContactoItems : es el Dbset de la clase contacto:

|  |
| --- |
|  |

Ahora en el archivo launchsettings de properties hacemos el cambio api/contactos para que llame en el navegador a éste método

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Ahora método **GetContactoItem**

|  |
| --- |
| //Peticion tipo get: un solo registro : api/contactos/4  [HttpGet("{idrecibido}")]  public async Task<ActionResult<Contacto>> GetContactoItem(int idrecibido)  {  var contactoitem = await \_context.ContactoItems.FindAsync(idrecibido);  if (contactoitem == null)  {  return NotFound();  }  return contactoitem;  } |

Graphical user interface, text, application

Description automatically generated

Si das F5 y buscas por el id = 3 obtendras un error porque aún no hay data

Graphical user interface, text, application

Description automatically generated

Cambio el tipo de dato de id en el método:

Graphical user interface, text, application, email

Description automatically generated

F5 nuevamente

Graphical user interface, text, application, email

Description automatically generated

Crear método Peticion tipo Post

|  |
| --- |
| //Peticion de tipo POST : api/contactos para insertar  [HttpPost]  public async Task<ActionResult<Contacto>> PostContactoItem(Contacto item)  {  \_context.ContactoItems.Add(item);  await \_context.SaveChangesAsync();  return CreatedAtAction(nameof(GetContactoItem),new {idrecibido=item.Id },item);  //llamo al método GetContactoItem(int idrecibido)  } |

Graphical user interface, application

Description automatically generated

El POST se prueba desde PostMan

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

La URL del Post es la misma del GET, proceda a configurar la petición como se indica:

Graphical user interface, text, application, website

Description automatically generated

Establezca los **headers** values

Graphical user interface, application

Description automatically generated

Establezco los valores en el **Body**

Graphical user interface, text, application

Description automatically generated

Click en send y verifique

Graphical user interface, text, application, email

Description automatically generated

Llamando a contactos:

<https://localhost:44356/api/contactos>

Graphical user interface, text, application

Description automatically generated

Consulta a todos con GET desde POSTMAN

Graphical user interface, text, application, email

Description automatically generated

UPDATE

|  |
| --- |
| //Peticion Tipo PUT: api/contactos/2  [HttpPut("id")]  public async Task<ActionResult> PutContactoItem(int id, Contacto item)  {  if (id !=item.Id)  {  return BadRequest();  }  \_context.Entry(item).State = EntityState.Modified;  await \_context.SaveChangesAsync();  return NoContent();  } |

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Se modifican los datos en el Postman:

Graphical user interface, application

Description automatically generated

Envie los datos

Graphical user interface, text, application, email

Description automatically generated

Verifique en base de datos

Table

Description automatically generated

DELETE

|  |
| --- |
| //PETICION DELETE PARA BORRAR : api/contactos/2  [HttpDelete("{id}")]  public async Task<ActionResult> DeleteContactoItem(int id)  {  var contactoitem = await \_context.ContactoItems.FindAsync(id);  if (contactoitem ==null)  {  return NotFound();  }  \_context.ContactoItems.Remove(contactoitem);  await \_context.SaveChangesAsync();  return NoContent();  } |

|  |
| --- |
|  |

Ahora en el Postman

Graphical user interface, text, application, email

Description automatically generated

Verifique en base de datos

Table

Description automatically generated

# Tablas relacionadas Video 17 Udemy

|  |
| --- |
|  |

Solución: Ciclo de objetos detectado cuando trae un libro trae tambien la data del autor por el include

return await context.Libros.Include(x => x.Autor).FirstOrDefaultAsync(x => x.Id == id);

|  |
| --- |
|  |

Solución: En la clase Statup.cs agregue el código resaltado en rojo dentro de la función ConfigureServices

|  |
| --- |
|  |

|  |
| --- |
| services.AddControllers().AddJsonOptions(x =>  x.JsonSerializerOptions.ReferenceHandler = ReferenceHandler.Preserve); |