



Curso: CI-0126

## Ingeniería de software

### Laboratorio 4: Creando un proyecto de backend conectado a base de datos.

#### Resumen

El objetivo de este laboratorio es poder crear un proyecto de backend que pueda conectarse a una base de datos.

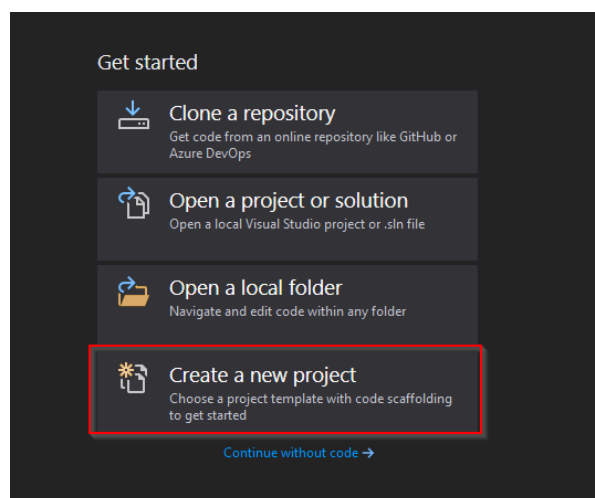
#### Primera parte - Creación del proyecto de API

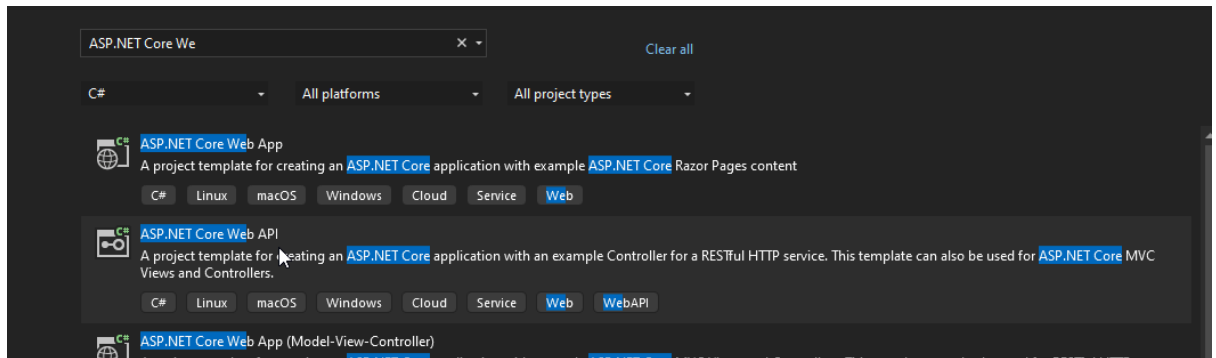
Primero es necesario entender que es un web API y que opciones ofrece .NET. Para esto se le recomienda al menos ver la sección 1 y 2 del siguiente curso de ASP.NET <https://learn.microsoft.com/es-mx/training/modules/build-web-api-aspnet-core/>

También tome el tiempo para investigar sobre las RESTFul APIs.

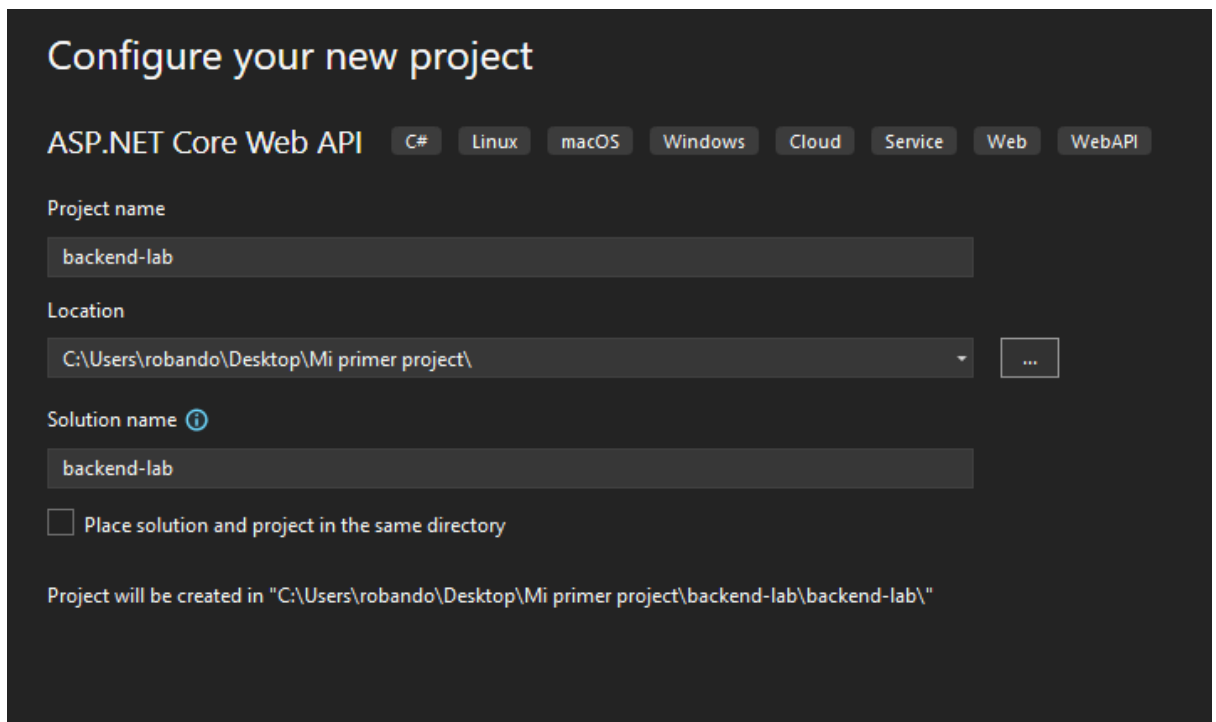
Una vez comprendida esta información vamos a proceder a crear el proyecto.

1. Primero abra Visual Studio, de clic en crear un nuevo proyecto. Va a seleccionar del tipo de proyecto **ASP.NET Core Web API**. Como en las siguientes imágenes:



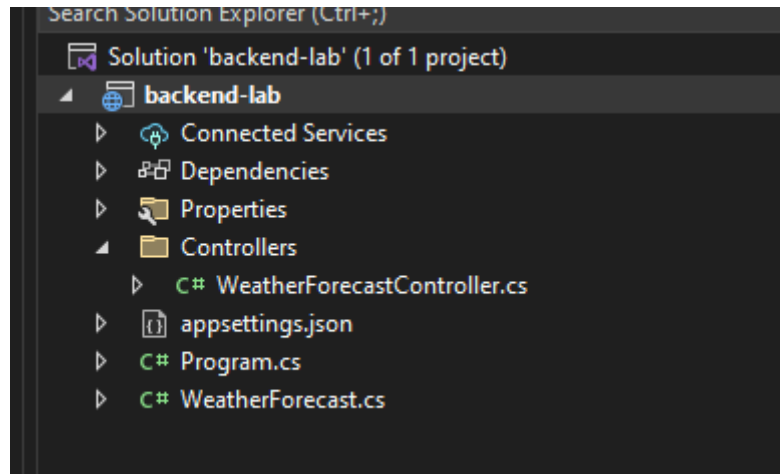



2. Al proyecto puede ponerle de backend-lab-carné vincule este proyecto a un nuevo repositorio en github (el mismo que el del front end del lab anterior).

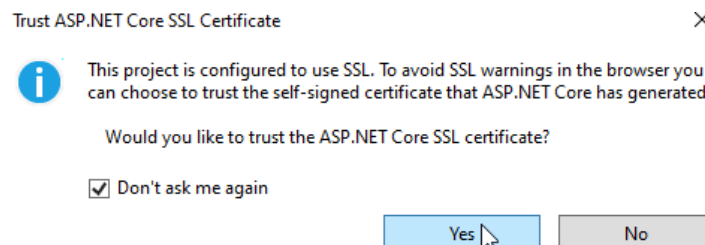




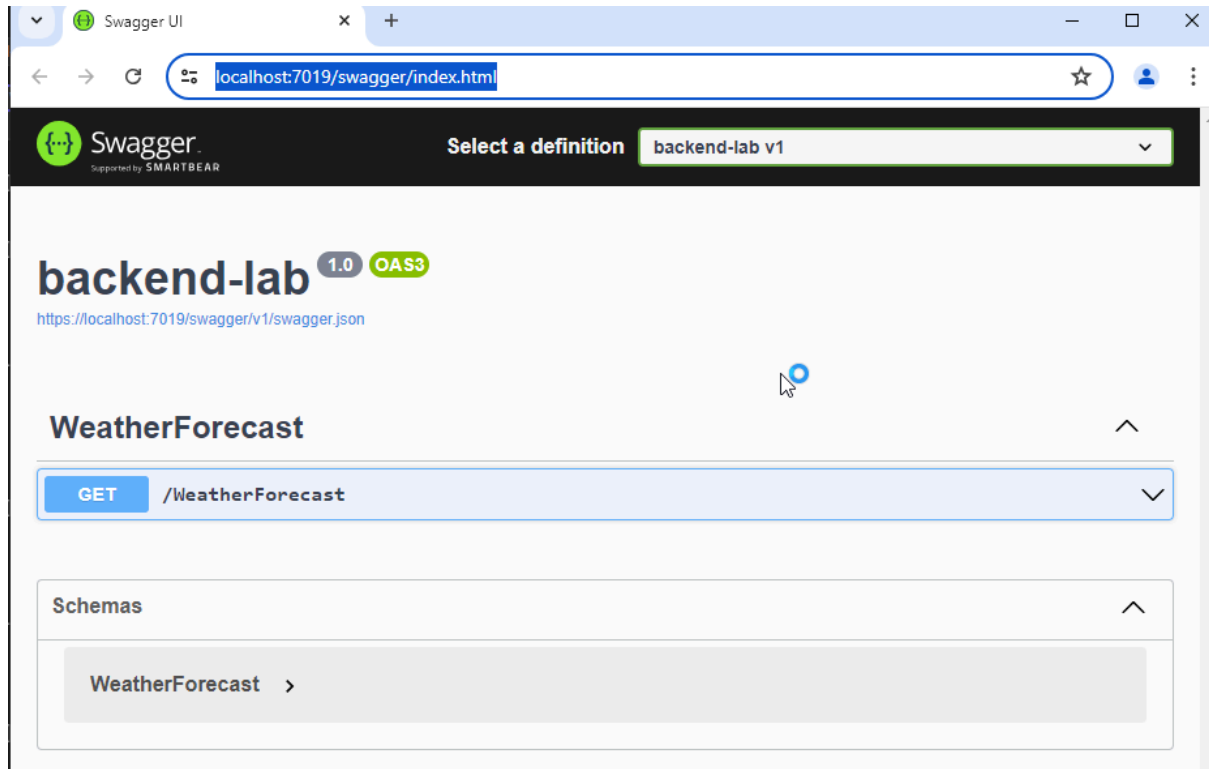
La solución va a tener una estructura similar a esta:



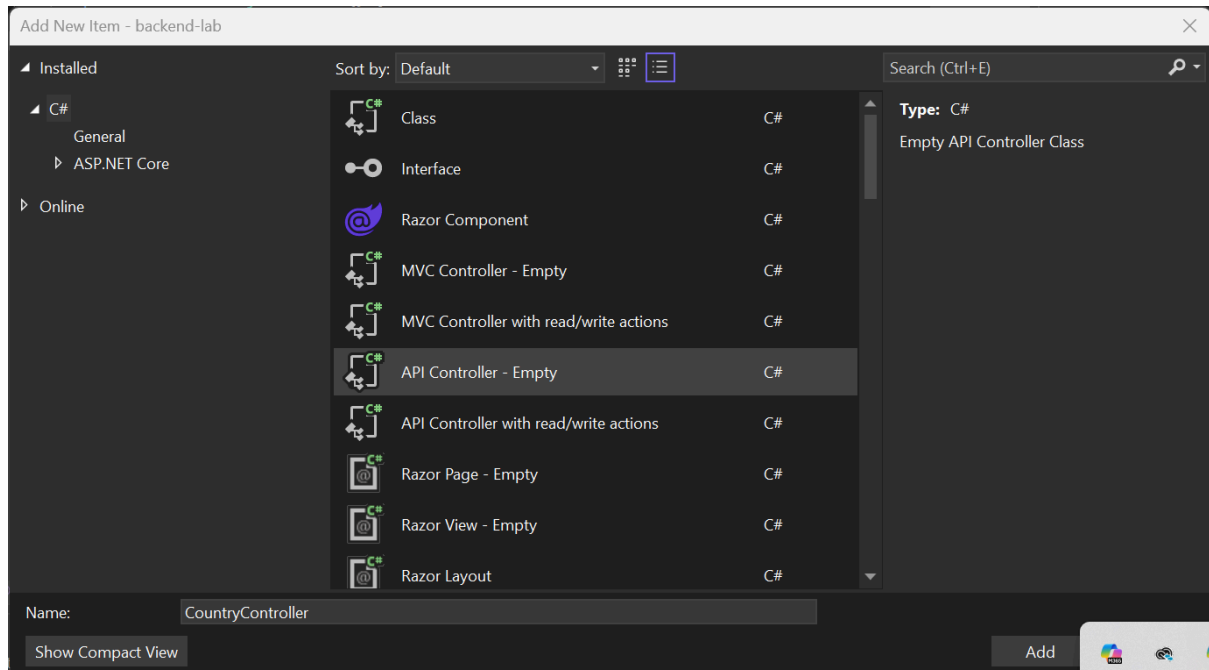
3. Por defecto Visual Studio va a crear la clase WeatherForecast.cs y WeatherForecastController.cs. Estas son clases de ejemplo de cómo es un modelo y un controlador. Si da clic en hacer build de la solución  backend\_lab, es posible que obtenga el siguiente mensaje. A este dele clic en “don't ask me again” y finalmente “Yes”.



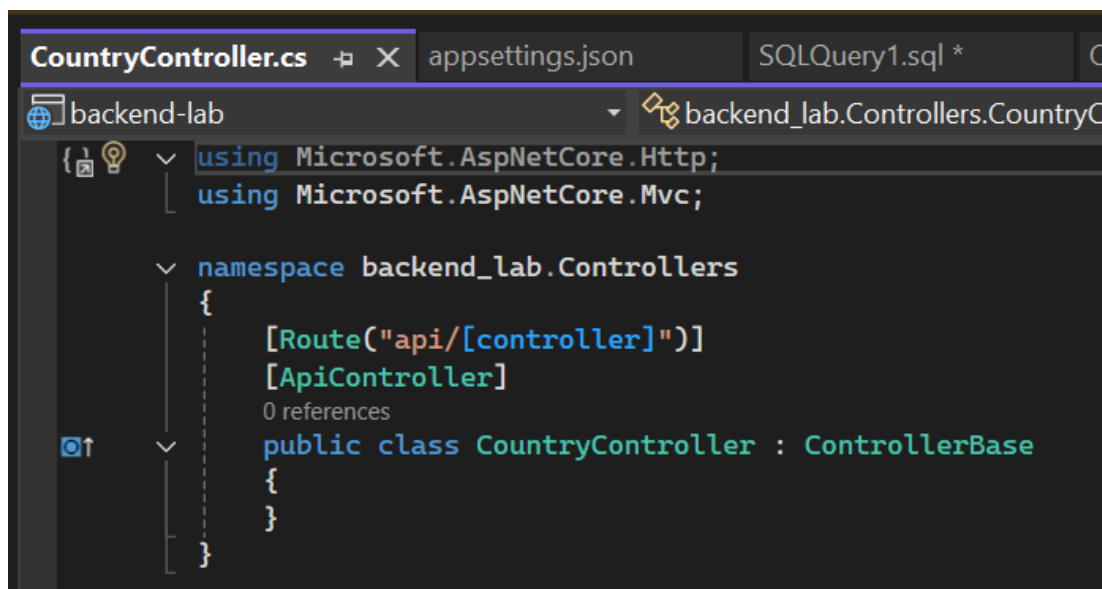
4. Una vez que levante la aplicación, va a poder ingresar a la página de swagger del API. Swagger es la librería de APIs disponibles para ser consumidos por los clientes del API, ya sea un proyecto web, móvil o cualquier otro sistema que necesite interactuar con nuestro backend.



5. Elimine los archivos WeatherForecastController.cs y WeatherForecast.cs.
6. De clic derecho sobre la carpeta de controller y agregue un nuevo **Empty API controller** con el nombre de CountryController y de clic en Add. Como en la siguiente imagen



Por defecto Visual Studio generará el siguiente código:



El tag **[ApiController]** le hace saber a ASP.NET Core que esa clase es un controlador, además es un punto de acceso y salida de información. Por otro lado el tag **[Route("api/[controller]")]** quiere decir que la URL para acceder a este



controlador específico es por medio de su nombre en otras palabras en el controller de country será:

None

<https://localhost:7019/api/Country>

## 7. Dentro del CountryController cree el siguiente método

```
[HttpGet]
public string Get() {
    return "Hola Mundo";
}
```

Este método lo que hace es crear un endpoint de tipo GET donde solo retorna un string que dice “Hola mundo”. Si corremos una vez más la aplicación. Obtendremos una nueva sección o endpoint en swagger donde encontraremos un único método llamado GET.

### Country

GET

/api/Country

Parameters

Try it out

No parameters

Responses

Code	Description	Links
200	Success	No links

Media type

text/plain

Controls Accept header.

Example Value | Schema

string



8. Haga clic en el botón **try it out** y luego **execute** y podrá ver el resultado de llamar este endpoint.

The screenshot shows a REST client interface with the following sections:

- GET /api/Country**: The method and endpoint.
- Parameters**: A section with a "Cancel" button and the text "No parameters".
- Execute**: A blue button highlighted with a red box.
- Clear**: A button to clear the request.
- Responses**: A section containing:
  - Curl**: A code block with the command: `curl -X 'GET' \ 'https://localhost:7019/api/Country' \ -H 'accept: text/plain'`. A red arrow points to the URL.
  - Request URL**: A text field containing `https://localhost:7019/api/Country`. A red arrow points to the URL.
  - Server response**: A section with a table showing the response code and body.

Code	Details
200	<p><b>Response body</b></p> <p>Hola Mundo</p> <p><b>Response headers</b></p> <p>content-type: text/plain; charset=utf-8 date: Tue, 12 Aug 2025 04:42:24 GMT server: Kestrel</p>

Aquí podrá ver como se llama el endpoint, la respuesta del servidor y el código de respuesta. El código es **200** lo que significa que la llamada fue procesada con éxito.

## Segunda parte - Conexión con la base de datos

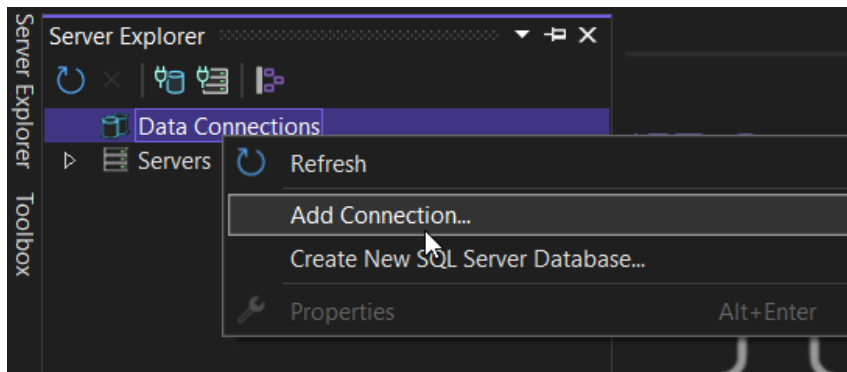
Existen diferentes maneras de conectarse a una base de datos. Puede ser conectado a una base de datos local, es decir, que está en su propia computadora o bien puede conectarse a una base de datos que está alojada en un servidor remoto, como la base de datos que se le asignó al equipo en el curso de PI, a la



cual ingresa mediante una dirección IP con una conexión vía VPN a la ECCI. En ambos casos, la manera de establecer la conexión es similar.

## Conectando a una base de datos en un servidor remoto

1. Conéctese a la VPN de la ECCI.
2. En visual studio abra una el Server explorer. De clic derecho sobre Data connections -> Add new connection



3. Seleccione la opción de data source: **Microsoft SQL Server**
4. Llene el formulario utilizando la siguiente información:
  - a. Server name: utilice alguna de las direcciones IP que se le proporcionaron en el curso de bases de datos
  - b. Authentication: seleccione la opción que se muestra en la siguiente figura.
  - c. Ingrese sus credenciales: su usuario y su respectiva contraseña
  - d. Select or enter database name: cuando presione el menú desplegable, aquí se mostrarán todas las bases de datos a las cuales tiene acceso (lectura y/o escritura) dentro del servidor, seleccione la base que le fue asignada
  - e. Finalmente de clic en Test Connection





Add Connection ? X

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:  
Microsoft SQL Server (SqlClient) Change...

Server name:  
Refresh

Log on to the server

Authentication: SQL Server Authentication

User name: my user name

Password: my password

☐ Save my password

Connect to a database

☒ Select or enter a database name:

☐ Attach a database file:

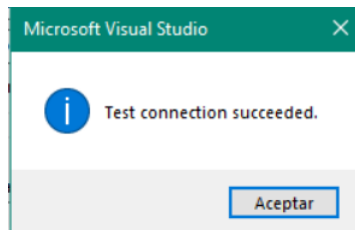
Browse...

Logical name:

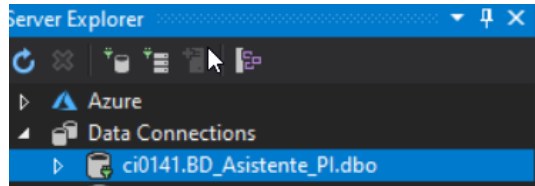
Advanced...

Test Connection OK Cancel

Debe obtener un mensaje de "Test Connection succeeded"



Ahora, presione el botón Ok para agregar la conexión. Una vez hecho esto, en el Server Explorer deberá aparecer su base de datos.



Para cerrar su conexión, presione clic derecho sobre su base de datos y seleccione la opción Close Connection, para volver a abrir la conexión basta con dar doble clic sobre la base de datos. Hasta este momento lo único que se ha hecho es añadir la conexión de la base de datos que está en el servidor remoto, pero para empezar a hacer consultas y demás es necesario realizar otros pasos que se presentan más adelante.

## Conectando a una base de datos local

Para este caso no ocupa estar conectado a la VPN de la ecci sin embargo el encargado de crear la base de datos será usted. Para conectar la base de datos al proyecto solo debe ingresar al Server Explorer y dar clic derecho sobre Data Connections y seleccione la opción Create New SQL Server Database. Se va abrir una ventana como la que se muestra en la figura a continuación.

Nota: si obtiene un error con el server name HOSTNAME intente con localhost



En el server name debe poner el nombre de su máquina (es el mismo nombre que va a aparecer en SQL server management studio cuando instalo SQL server). En el nombre de new database name ponga "Countries"

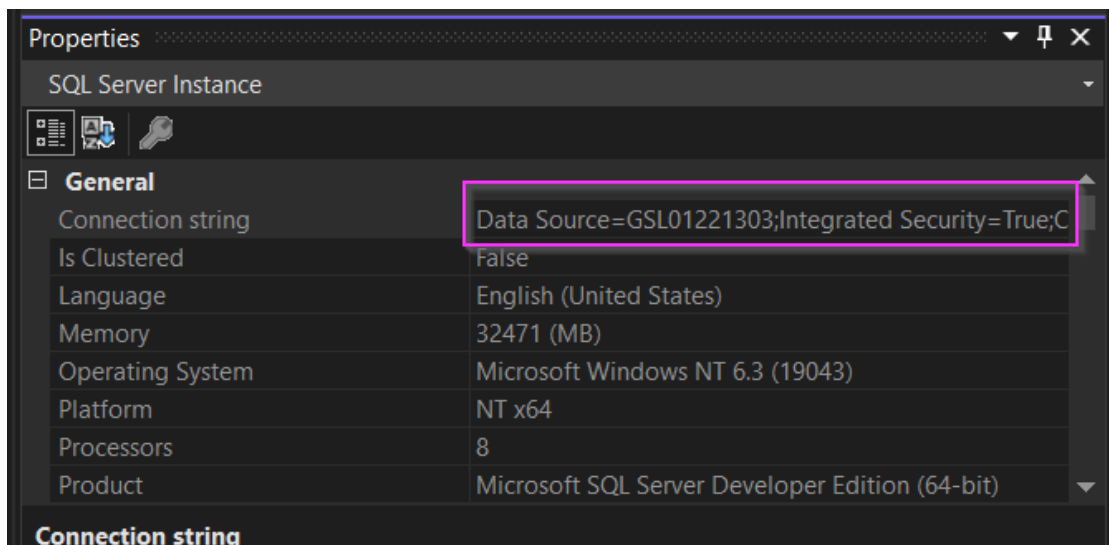
## Connection strings

Otra manera de agregar una conexión a la base de datos es por medio de un **connection string**. El Connection String se agrega y configura en el archivo appsettings.json el cual se puede acceder desde el Solution Explorer. El Connection String en sí es la manera de especificar al programa la fuente de los datos, con lo cual se puede establecer la conexión.

Usar el Connection String de la manera que se va mostrar a continuación, tiene la ventaja que cuando usted desea realizar consultas a una base de datos, no necesita repetir el proceso de ingresar esta hilera a cada uno de los comandos de SQL que se detallarán más adelante. Además, el Connection String no es único, usted puede tener varios en la configuración de su proyecto y nombrarlos de manera diferente, de modo que usted puede tener conexión con su base de datos del servidor remoto y local al mismo tiempo, incluso con otras bases de datos un mismo servidor o de distintos servidores.

### ¿Cómo obtener el connection string?

Desde el Server Explorer dé clic derecho sobre la base de datos Countries recién creada y seleccione la opción Properties. La siguiente figura muestra el resultado de seleccionar esta opción, usted observará una fila que justamente se llama Connection String, la cual en la columna de al lado tiene una hilera, copie la hilera completa



Ingresa al Solution Explorer y ubique el archivo llamado appsettings.json, abra el archivo y al final del código, agregue el connection string de la siguiente manera:

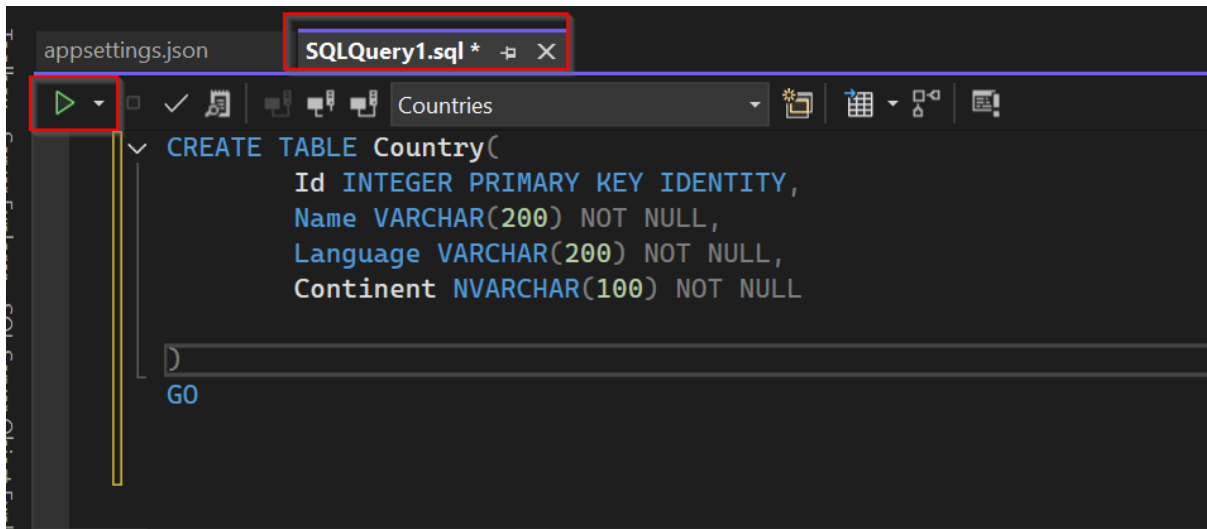


```
appsettings.json  SQLQuery1.sql *
Schema: https://www.schemastore.org/appsettings.json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "CountryContext": "Data Source=REBE-PC;Initial Catalog=Countries;Integrated Security=True"
  }
}
```

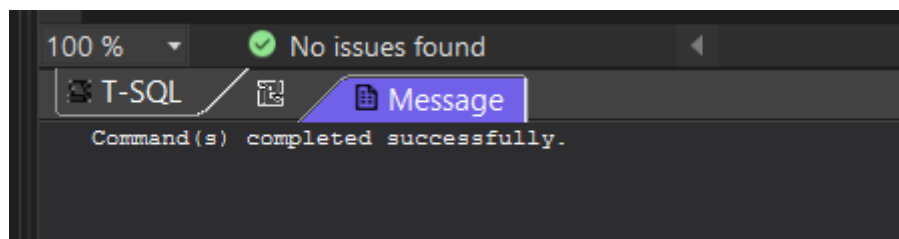
## Creando una tabla en la base de datos

En este apartado no se dará mayor detalle, debido a que únicamente se creará una tabla con algunos atributos de la manera que usted ya aprendió en el curso de bases de datos. Dé clic derecho sobre su base de datos Countries y seleccione la opción New query, copie el siguiente código:

```
CREATE TABLE Country(
    Id INTEGER PRIMARY KEY IDENTITY,
    Name VARCHAR(200) NOT NULL,
    Language VARCHAR(200) NOT NULL,
    Continent NVARCHAR(100) NOT NULL
)
GO
```



Presione el botón verde que está en la parte superior izquierda de la pantalla (figura anterior) y posteriormente recibirá un mensaje que refleja éxito en la creación de la tabla.



## Tercera parte - Manipulando los datos

La manipulación de los datos almacenados en la base de datos se puede interpretar de distintas maneras como, por ejemplo: insertar nuevos datos, modificar los datos existentes, desplegar algunos datos o eliminarlos. En esta parte del laboratorio se va a centrar en la visualización y creación de los datos

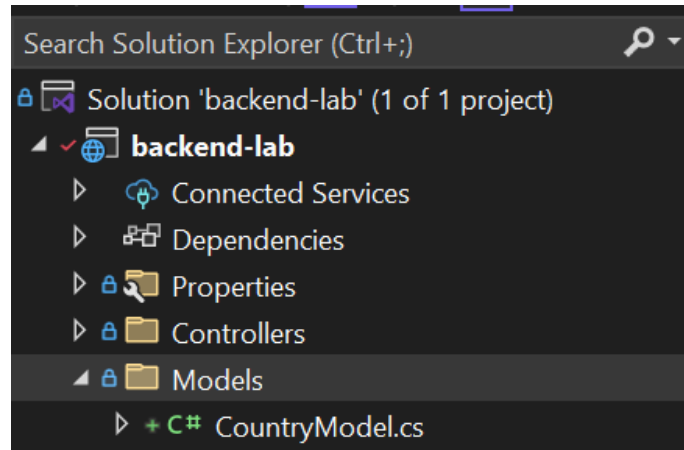
### Cree un modelo

Para lograr lo que se pretende, es necesario tener una manera de representar los datos en el sistema y para eso se utilizará un modelo. Básicamente es un objeto que contiene las características de una entidad de negocio. Para esto siga los siguientes pasos:

1. Dé clic derecho sobre su archivo de proyecto y seleccione la opción Add para crear una nueva carpeta, nombre la carpeta como Models.
2. Dé clic derecho sobre la carpeta Models y agregue una nueva clase de C#, nombre la clase como CountryModel.



La estructura de la solución debe ser similar a esta:



Ahora use el código de la siguiente imagen en el CountryModel:

```
namespace backend_lab.Models
{
    7 references
    public class CountryModel
    {
        0 references
        public int Id { get; set; }
        1 reference
        public string Name { get; set; }
        1 reference
        public string Continent { get; set; }
        1 reference
        public string Language { get; set; }
    }
}
```

**Nota:** en este caso la base de datos solo cuenta con la tabla country, es pertinente destacar que por esta razón el único modelo existente es el modelo country. No obstante, cuando existen más tablas, esto no implica que se deba crear un modelo por cada tabla, lo que se hace es unificar los datos de las distintas tablas en un modelo de manera que se logre una representación del objeto en cuestión o bien si para un mismo objeto existen variantes, se crean modelos que responden a las necesidades según los datos que desean presentar.



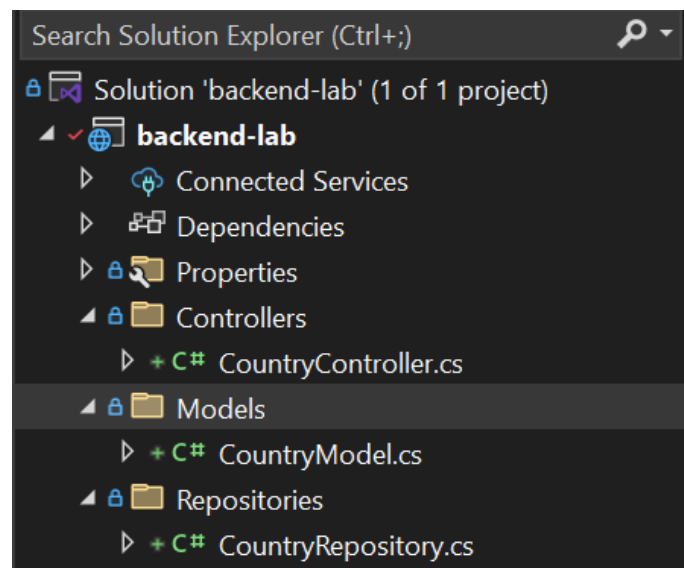
## Establecer la capa de negocios y conexión con la base de datos

En esta sección vamos a trabajar en crear una pequeña arquitectura del sistema utilizando 3 capas, presentación(controllers), negocio(services) y el manejo de datos (repositories). El tema de arquitectura de sistemas será parte del curso, por ahora vamos a trabajar una arquitectura simple que nos ayude a completar el laboratorio y darle una mejor separación de conceptos. Sin embargo esto no significa que esta es la única arquitectura o LA arquitectura que debe usar en todo problema.

Vamos a empezar con la capa de datos, va a crear una clase que va servir para hacer la conexión a la base de datos, y por medio de los métodos ejecutar consultas, insertar datos y demás operaciones con los datos.

1. Dé clic derecho sobre su archivo de proyecto y seleccione la opción Add para crear una nueva carpeta, nombre la carpeta como Repositories.
2. Dé clic derecho sobre la carpeta Repositories y agregue una nueva clase de C#, nombre la clase como CountryRepository . En esta clase se realizará lo necesario para establecer la conexión a la base de datos y manipularla.

Como resultado obtendrá la siguiente estructura en su solución:



Ahora vamos a agregar el siguiente código a la clase CountryRepository, este código nos permitirá conectar a la base de datos.

```
using backend_lab.Models;
using System.Data;
using System.Data.SqlClient;

namespace backend_lab.Handlers
{
```

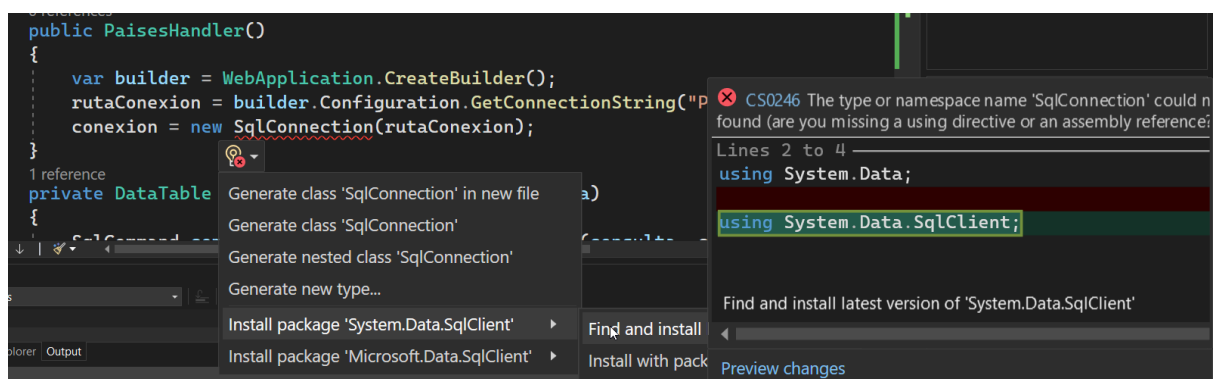


```
using backend_lab.Models;
using Dapper;
using System.Data.SqlClient;

namespace backend_lab.Repositories
{
    public class CountryRepository
    {
        private readonly string _connectionString;
        public CountryRepository()
        {
            var builder = WebApplication.CreateBuilder();
            _connectionString =
builder.Configuration.GetConnectionString("CountryContext");
        }

        public List<CountryModel> GetCountries()
        {
            using var connection = new
SqlConnection(_connectionString);
            string query = "SELECT * FROM dbo.Country";
            return connection.Query<CountryModel>(query).ToList();
        }
    }
}
```

Si tiene un problema a la hora de reconocer el paquete de SQL o Dapper dé clic en **Install package 'System.Data.SqlClient'** o **Install package 'Dapper'** sobre alguna de los nombres de clases no reconocidas como SqlConnection como lo puede observar en la siguiente imagen:







A continuación, se presenta una breve explicación del código que usted acaba de copiar:

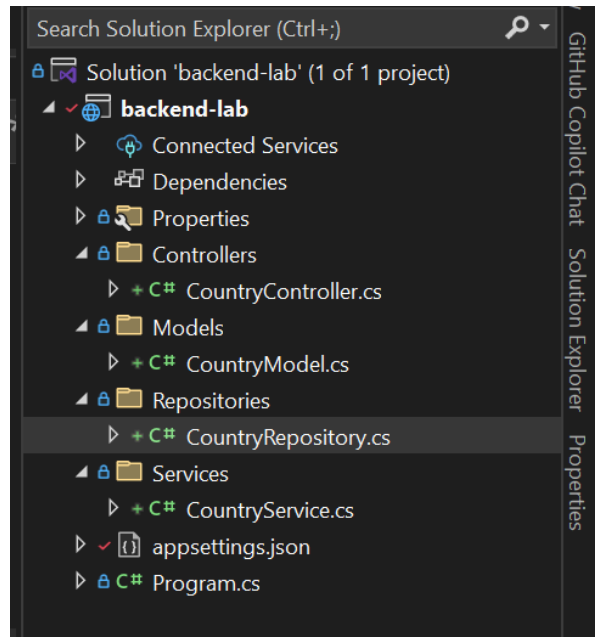
1. Observe que la clase tiene dos atributos privados, uno de ellos es el Connection String el cual se utiliza a su vez para inicializar la conexión con el servidor. Observe que en el encabezado se incluyen varias librerías, una de ellas es un cliente de SQL y es la que brinda los elementos para establecer la conexión y crear las consultas.
2. Observe que el constructor, para obtener el Connection String, se pasa por parámetro el nombre de alguno de los Connection String que fue agregado en el appsettings.json. En este caso se utiliza el de la base de datos Country, pero usted podría conectarse a cualquier otra base de datos con solo cambiar el este valor por el nombre de otro Connection String que usted haya agregado. Note además que, al inicializar un nuevo CountryRepository para manipular algunos datos, por medio del constructor, la conexión estará automáticamente creada.
3. Dapper es una librería ligera para trabajar con bases de datos en .NET. Te ayuda a ejecutar consultas SQL de forma rápida y sencilla, y a convertir los resultados en objetos de C#.
4. Con respecto al método getCountries, observe que las consultas (no es la única manera) se crean a través de un String convencional. Usted en este String puede escribir un query tan complejo como se lo pueda imaginar, aunque en este caso se ejemplifica con una consulta básica. Las herramientas de Visual Studio le permiten escribir estas hileras de manera cómoda, ya que usted puede pulsar enter al final de la primera línea y automáticamente se crea un espacio para seguir escribiendo.
5. Dapper nos ayuda con la siguientes cosas:
  - a. Ejecutarla en la base de datos.
  - b. Leer los resultados.
  - c. Crear una lista de objetos C# con esos datos.

Para mayor documentación puede usar esta pagina web:  
<https://dappertutorial.net/dapper>

Ahora bien vamos a crear el servicio que contendrá la lógica de negocio y validaciones, nuevamente más adelante en el curso veremos diseño y arquitectura de software. Siga lo siguientes pasos:

1. Dé clic derecho sobre su archivo de proyecto y seleccione la opción Add para crear una nueva carpeta, nombre la carpeta como Services.
2. Dé clic derecho sobre la carpeta Services y agregue una nueva clase de C#, nombre la clase como CountryService. En esta clase se realizará lo necesario para establecer utilizar el repositorio, validar los datos o aplicar cualquier lógica de negocio.

Como resultado obtendrá la siguiente estructura en su solución:



Ahora vamos a agregar el siguiente código a la clase `CountryService`, este código nos permitirá conectar al repositorio y obtener los datos:

```
using backend_lab.Models;
using backend_lab.Repositories;

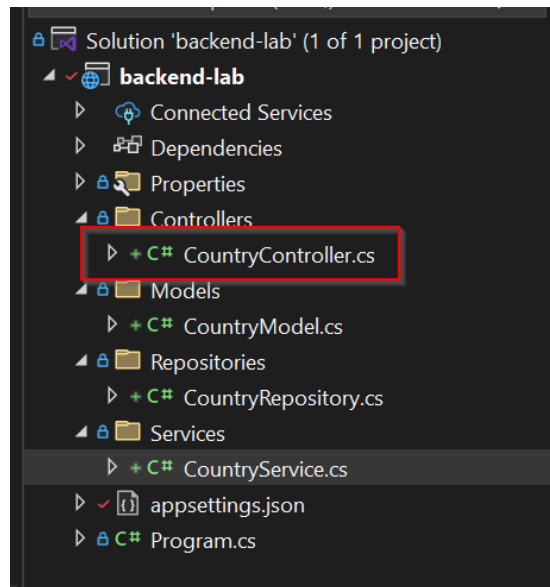
namespace backend_lab.Services
{
    public class CountryService
    {
        private readonly CountryRepository countryRepository;
        public CountryService()
        {
            countryRepository = new CountryRepository();
        }

        public List<CountryModel> GetCountries()
        {
            // Add any missing business logic when it is
            neccesary
            return countryRepository.GetCountries();
        }
    }
}
```

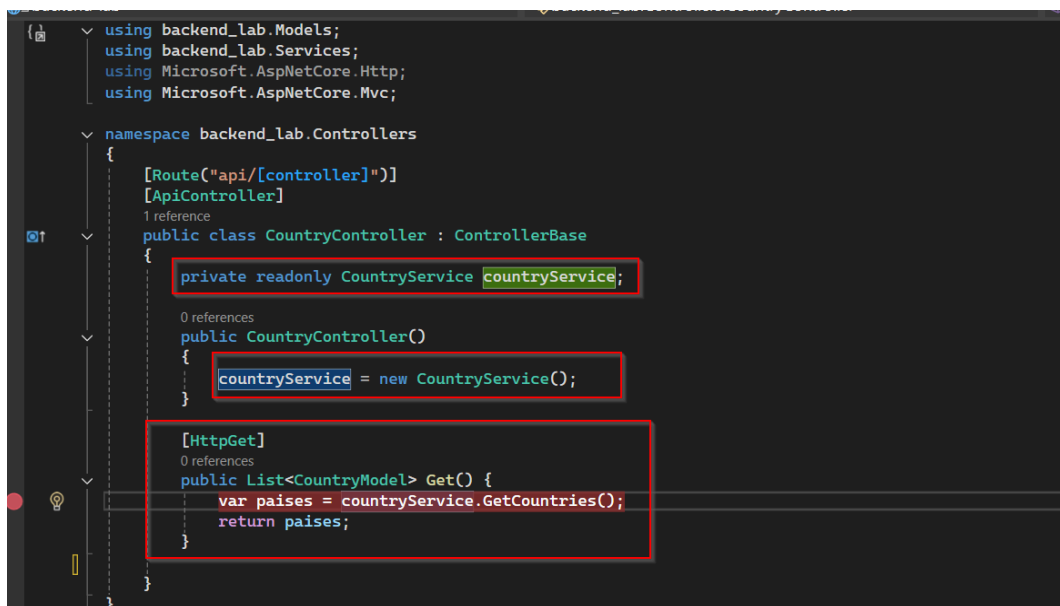


## Desplegando los datos en el endpoint

CountryService puede conectarse al repository y obtener los datos de los países y retornar modelos. Ahora es necesario desplegar estos datos, para eso vamos a crear un usar el controlador CountryController. Para este momento su solución debe lucir como la siguiente imagen:



Ahora bien vamos a cambiar el código del endpoint **GET** donde ahora devolveremos una lista de modelos tipo Country: De esta manera el código del controlador se verá algo similar a esta imagen:





Antes de correr la solución, agregue algunos datos a la tabla country de su base de datos, para esto cree un nuevo query y corralo de la misma manera que hizo para crear la tabla. Puede usar el siguiente código como ejemplo:

```
INSERT INTO [dbo].[Country] ([Name],[Language],[Continent])  
VALUES('Costa Rica', 'Español', 'América'),  
('Argentina', 'Español', 'América'),  
('Canada', 'Inglés/Frances', 'América'),  
('Francia', 'Frances', 'Europa'),  
('España', 'Español', 'Europa')
```

GO

Una vez haya hecho la inserción, verifique por medio de un Select que se hayan agregado correctamente, para ello puede borrar el código copiado en el query y escribir el Select \* o bien crear un nuevo query.

Ahora corra nuevamente la aplicación y en swagger utilice la opción try it out para el endpoint country. El resultado obtenido debe ser como el siguiente:

**Country**

GET /api/Country

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7019/api/Country' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7019/api/Country

Server response

Code Details

200

Response body

```
[  
  {  
    "id": 1,  
    "name": "Costa Rica",  
    "continent": "América",  
    "language": "Español"  
  },  
  {  
    "id": 2,  
    "name": "Argentina",  
    "continent": "América",  
    "language": "Español"  
  },  
  {  
    "id": 3,  
    "name": "Canada",  
    "continent": "América"  
  }  
]
```

**NOTA: No borre este laboratorio de su cuenta github. En el siguiente laboratorio conectaremos el front end con el back end de la aplicación.**

## Entregable

Para este laboratorio se debe compartir el repositorio usado con el asistente y la profesora. Ambos usuarios están en la página principal de mediación

**Horario y lugar de consulta:** Lunes 16:00 - 18:00, Jueves 17:00 - 18:00 La consulta se realizará presencial en oficina pendiente por avisar o virtual por la herramienta Zoom según coordinación.

**Usuarios de Github:** Profe Rebeca (**rebeca-ov**)

Cree un pequeño documento con diferentes screenshots donde demuestre:

1. Conexión exitosa a la base de datos (debe mostrar el nombre de su computadora local o la base de datos dada en el curso de bases de datos)
2. Screenshot de el **Select \* from Country** donde se enseñen que realmente se crearon los datos con éxito. Ya sea en la base de datos local o en la base de datos personal del curso de bases de datos.
3. Resultado final al correr el laboratorio en su computadora.
4. Haga una pequeña investigación sobre qué es: Entity framework y Dapper.
5. Haga una investigación de que es Postman e instalelo en su computadora, debido a que vamos a usarlo en clase.

### Tabla de calificación

Rubro	Porcentaje
Repositorio creado en github	20pts
Documento respondiendo a las preguntas solicitadas.	40pts
Aplicación completada, compilando sin problemas y funcional	40pts



UNIVERSIDAD DE  
COSTA RICA

ECCI  
Escuela de  
Ciencias de la  
Computación e  
Informática

## Referencias

- Web API con ASP.NET Core:  
<https://learn.microsoft.com/es-mx/training/modules/build-web-api-aspnet-core/>
- Dapper tutorial <https://dappertutorial.net/dapper>