# EVALUACIÓN
# IM SOFTWARE

# Reporte de
# Examen Práctico .Net

# 3 Partes
# T-SQL + Web Api + Vista MVC

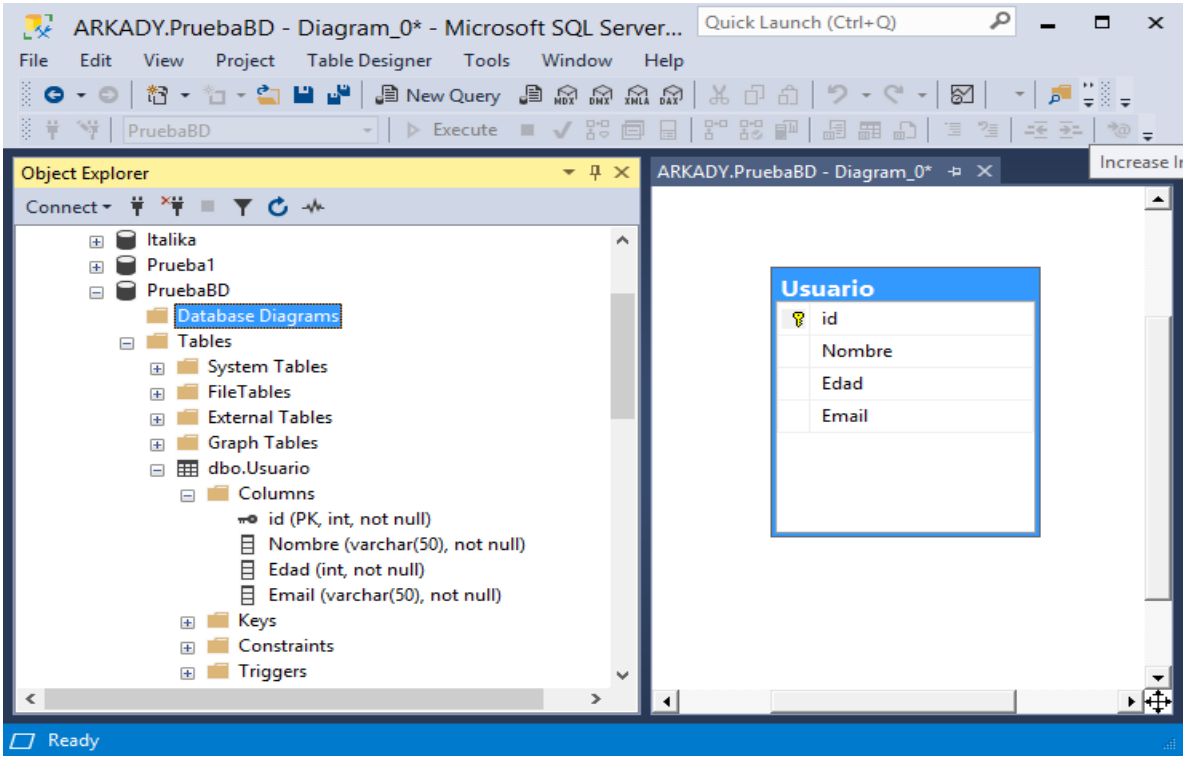
***Desarrollador: Luis Javier Vázquez Estrada***
***ing.luis.vazquez.e@outlook.com***

***Cel. 55 4943 6290***

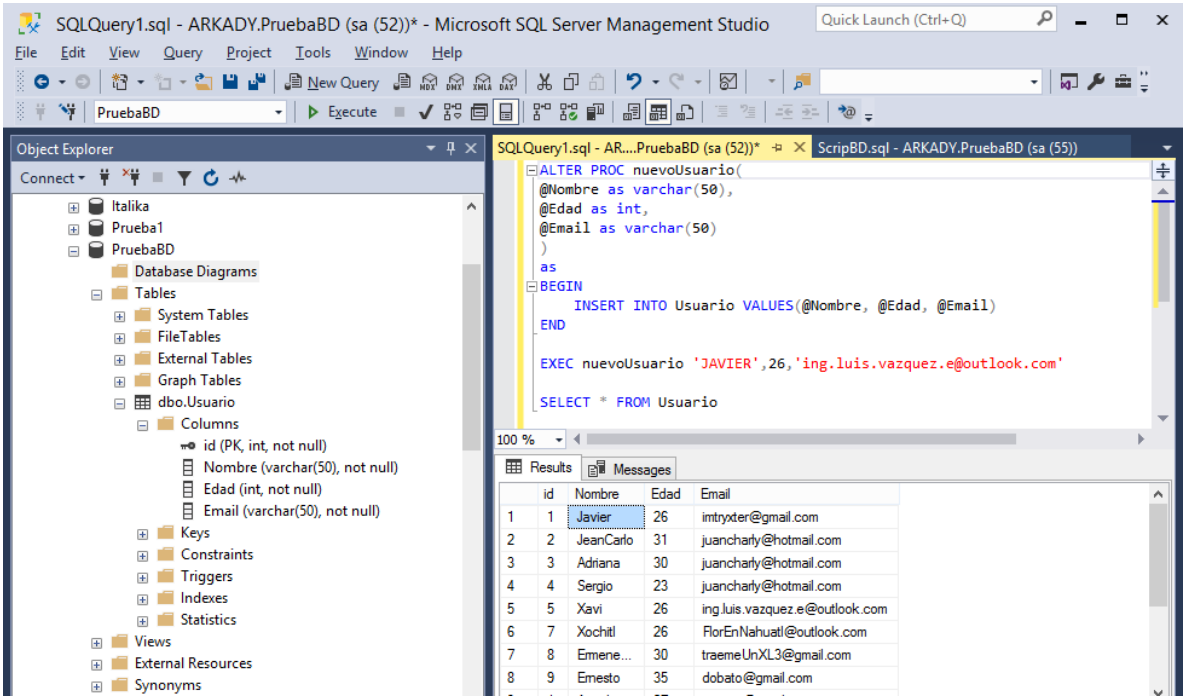
# Contenido

## BD

Se creo una Base de Datos en Sql Server 2019 una tabla Usuarios que representa a las Personas con características básicas como Nombre, Edad y Email además del ID respectivo como llave primaria.



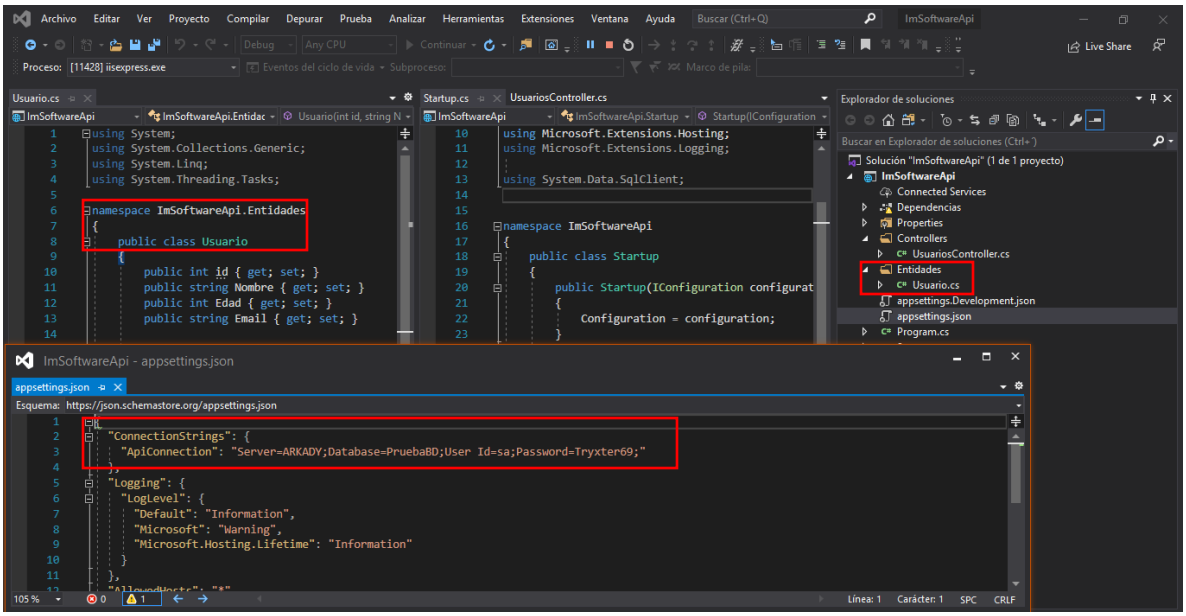

Se setearon algunos datos y se generó un Stored Procedure para facilitar la inserción más adelante



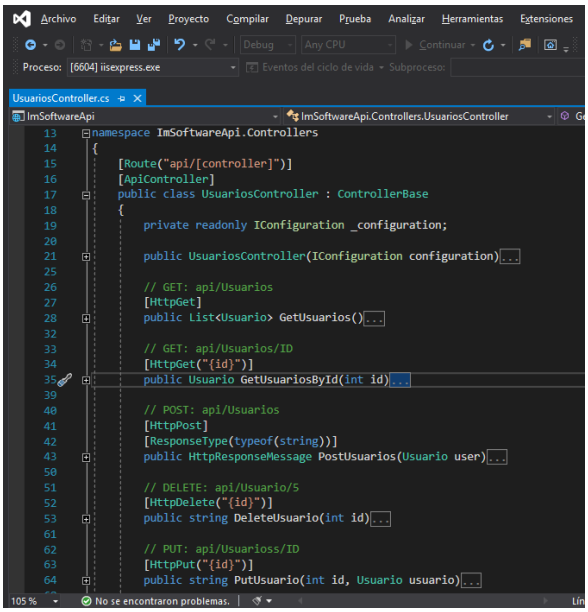
# Web Api

Para la manipulación de los datos de forma asíncrona, se creó un nuevo proyecto Web Api en el Framework de .Net Core para administrar los datos de la base de datos previamente creada. Evitando usar la herramienta ORM Entity Framework se realizó el proceso de administración de datos de la Base de Datos con la tecnología ADO.

Para ello se necesito configurar tanto la cadena de conexión, como las dependencias, y los modelos en la carpeta de Entidades.




Ya con esas configuraciones se generó el controlador principal para la tabla conectada por ADO con todas las consultas de verbos HTTP más usadas para un CRUD.



- [HttpGet]
  public List<Usuario> GetUsuarios(){}

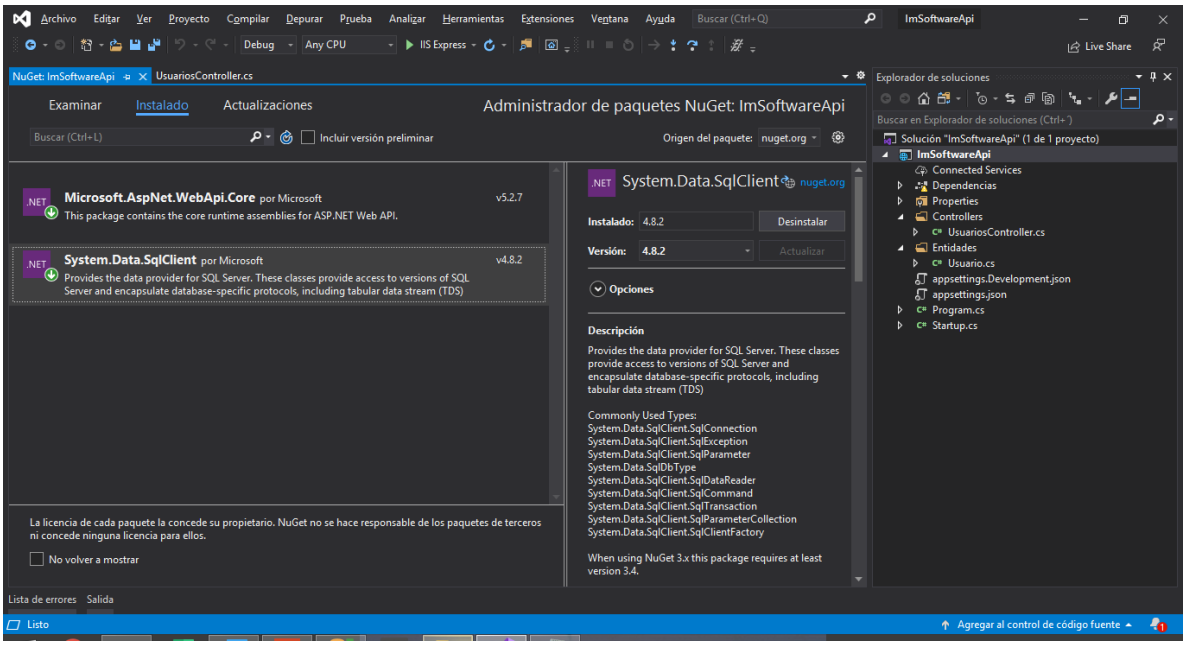
- [HttpGet("{id}")]
  public Usuario GetUsuariosById(int id){}

- [HttpPost]
  public HttpResponseMessage PostUsuarios(Usuario user)

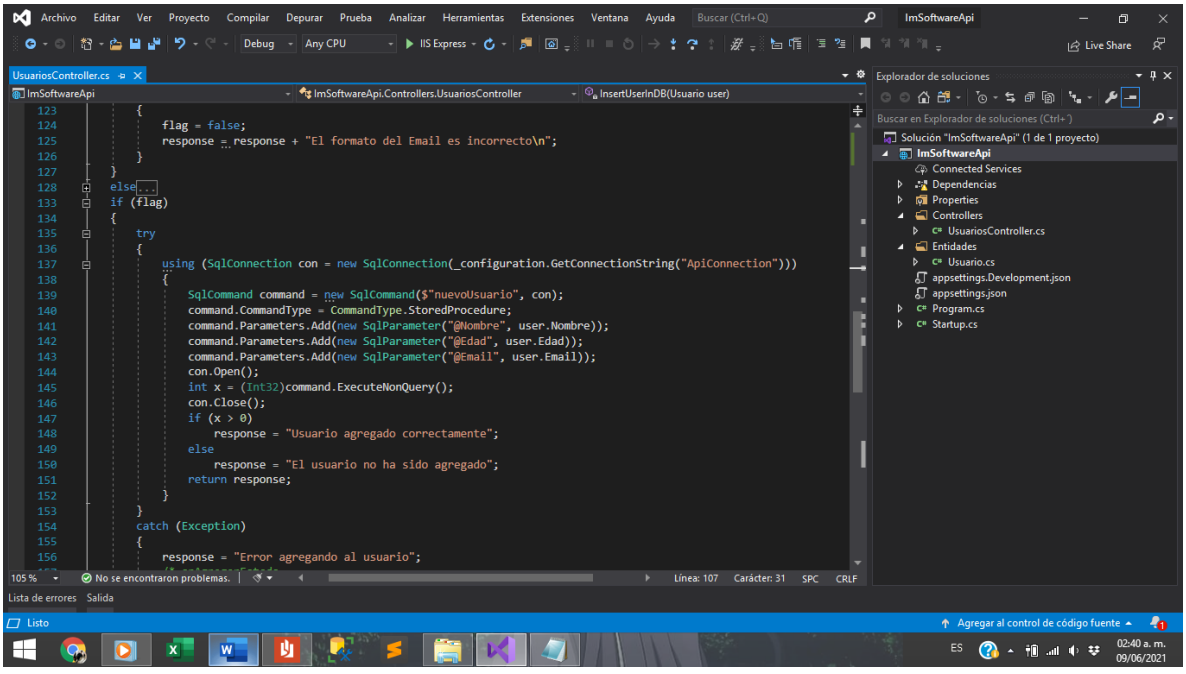
- [HttpDelete(“{id}”)]
  public string DeleteUsuario(int id)

- [HttpPut(“{id}”)]
  public string PutUsuario(int id, Usuario usuario)

Cabe mencionar la instalación de los paquetes Nuget adecuados, SQL.Data.Client que nos proporciona Visual Studio para poder administrar la base de datos con ADO




Así como el ADO en al menos uno de las consultas, principalmente la Inserción “Post” donde se usa el Store Procedure antes mencionado y generado.



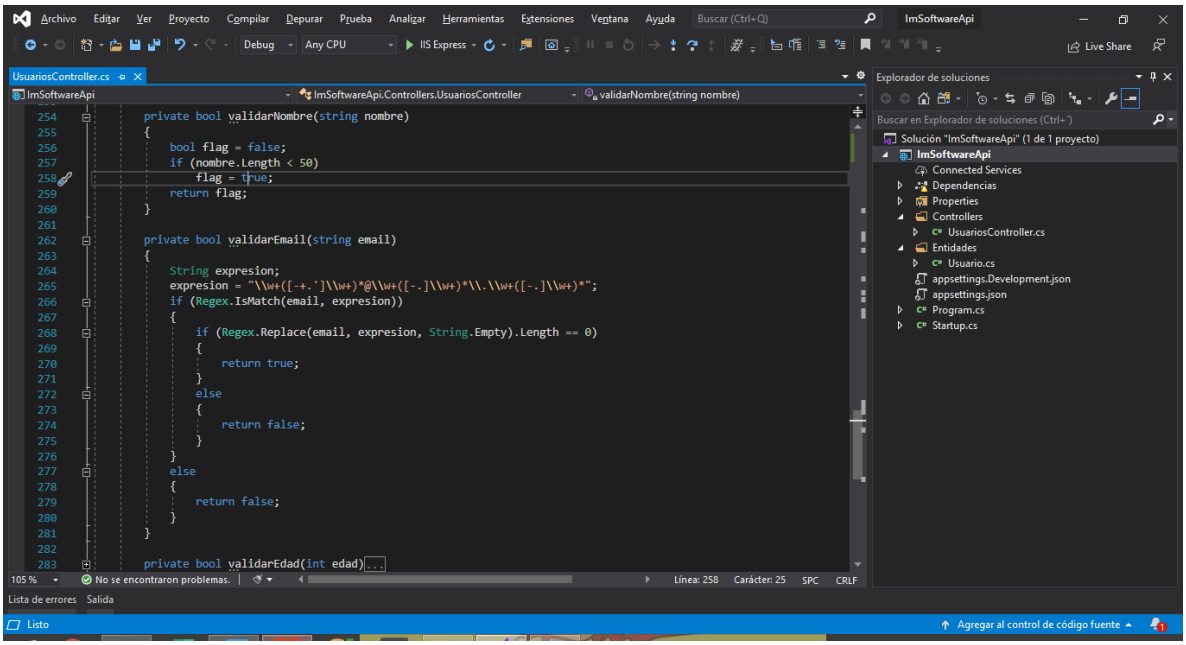
```
{
    flag = false;
    response = response + "El formato del Email es incorrecto\n";
}
}
else...
if (flag)
{
    try
    {
        using (SqlConnection con = new SqlConnection(_configuration.GetConnectionString("ApiConnection")))
        {
            SqlCommand command = new SqlCommand($"nuevoUsuario", con);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.Add(new SqlParameter("@Nombre", user.Nombre));
            command.Parameters.Add(new SqlParameter("@Edad", user.Edad));
            command.Parameters.Add(new SqlParameter("@Email", user.Email));
            con.Open();
            int x = (Int32)command.ExecuteNonQuery();
            con.Close();
            if (x > 0)
                response = "Usuario agregado correctamente";
            else
                response = "El usuario no ha sido agregado";
            return response;
        }
    }
    catch (Exception)
    {
        response = "Error agregando al usuario";
```

Parte del script es el siguiente:

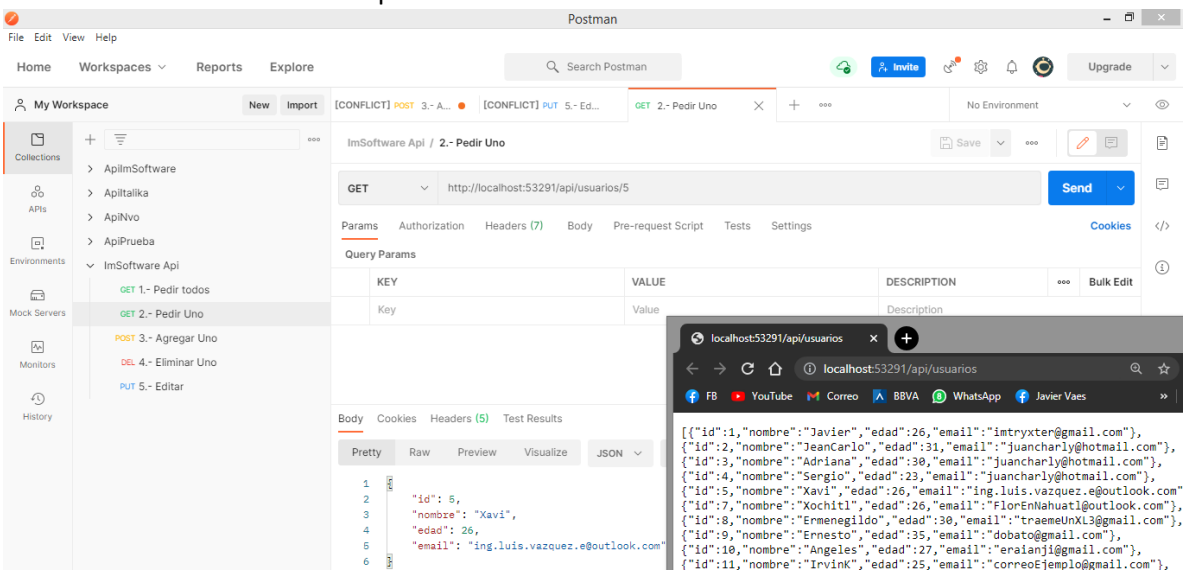
```
try
{
using (SqlConnection con = new SqlConnection(_configuration.GetConnectionString("ApiConnection")))
            {
               SqlCommand command = new SqlCommand($"nuevoUsuario", con);
               command.CommandType = CommandType.StoredProcedure;
               command.Parameters.Add(new SqlParameter("@Nombre", user.Nombre));
               command.Parameters.Add(new SqlParameter("@Edad", user.Edad));
               command.Parameters.Add(new SqlParameter("@Email", user.Email));
               con.Open();
               int x = (Int32)command.ExecuteNonQuery();
               con.Close();
               if (x > 0)
                  response = "Usuario agregado correctamente";
               else
                  response = "El usuario no ha sido agregado";
               return response;
            }
}
catch (Exception)
{
      response = "Error agregando al usuario";
}
```

También podemos visualizar las subrutinas del punto 3 – Las Validaciones de la Web Api.

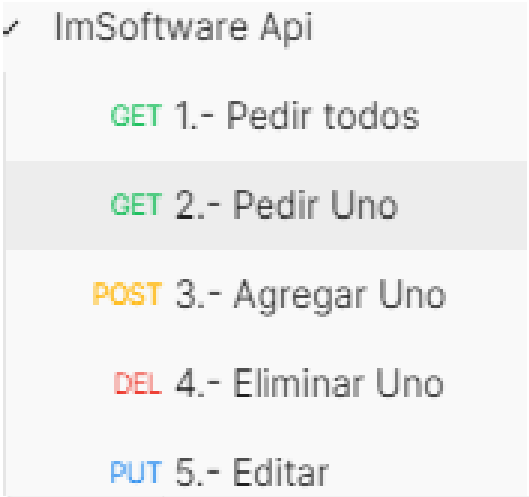
- **private bool validarNombre(string nombre)**
- **private bool validarEmail(string email)**
- **private bool validarEdad(int edad)**



Ya con la compilación correcta y la ejecución de la Web Api usamos **Postman** para probar la funcionalidad correcta de cada petición.




Cada verbo básico para un CRUD ha sido evaluado.



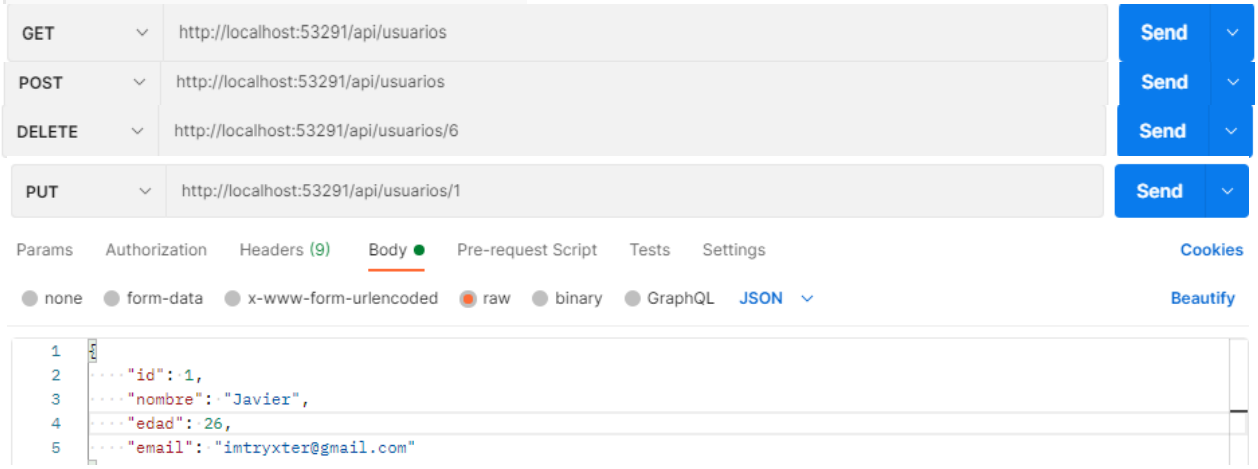

http://localhost:53291/api/usuarios

http://localhost:53291/api/usuarios/5

http://localhost:53291/api/usuarios

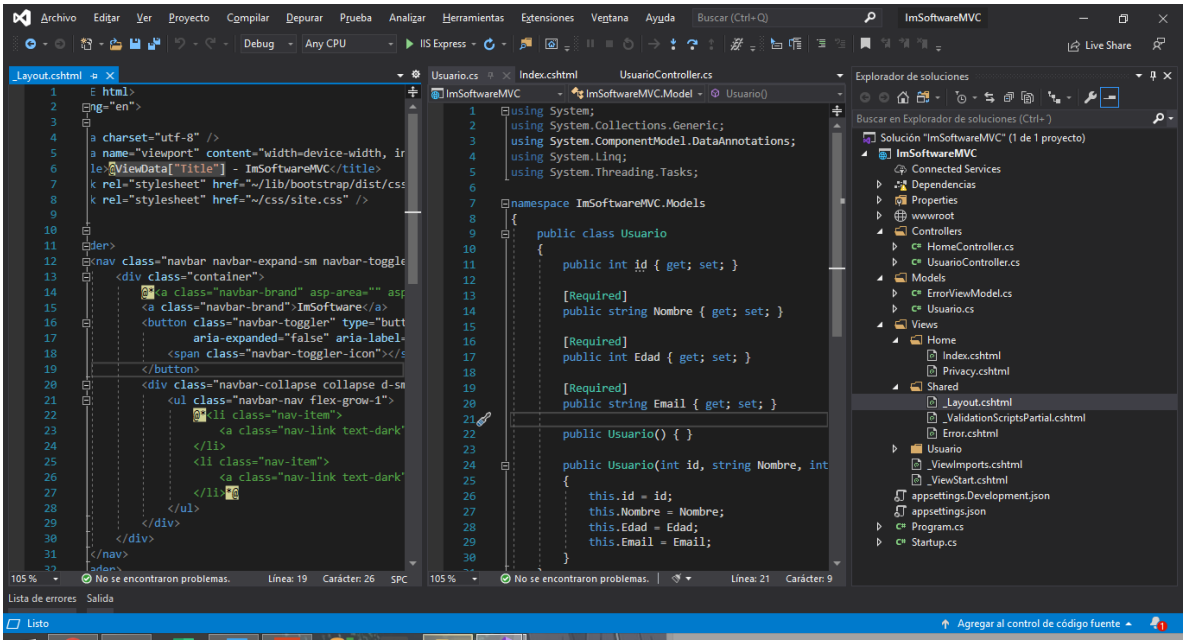
http://localhost:53291/api/usuarios/6

http://localhost:53291/api/usuarios/1

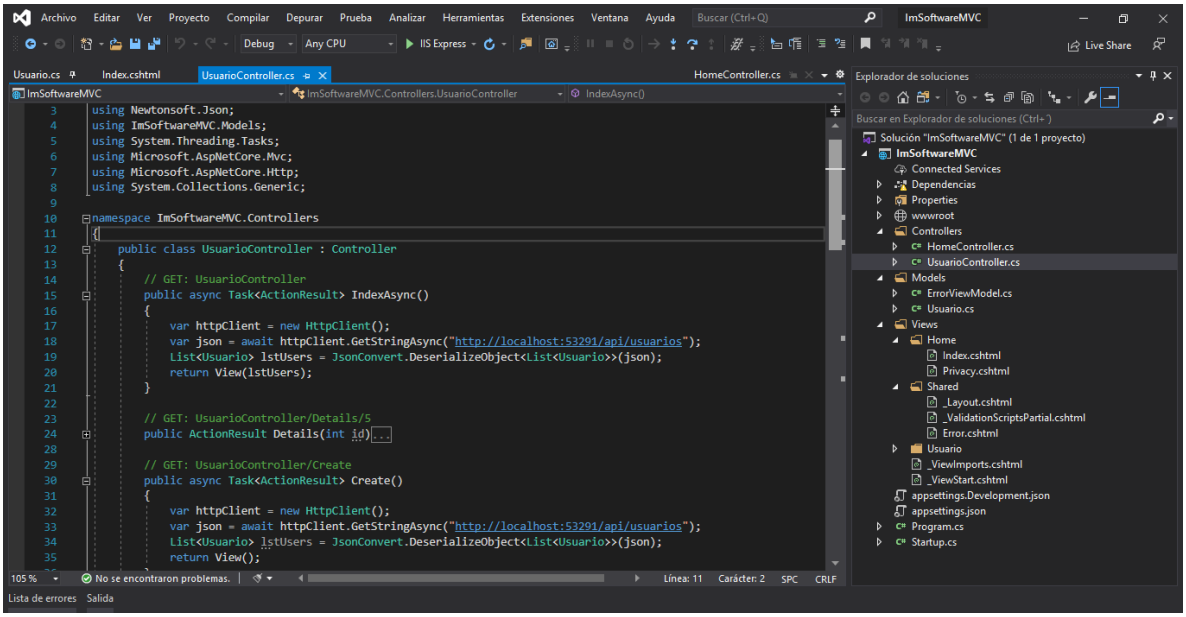


# MVC

En la primera parte del proyecto se desarrollan los modelos y se edita el Layout Maestro.



Se genera el Controlador de lectura y escritura Api, en base al modelo usuario de la capa de presentación y la serialización con el paquete Newton.

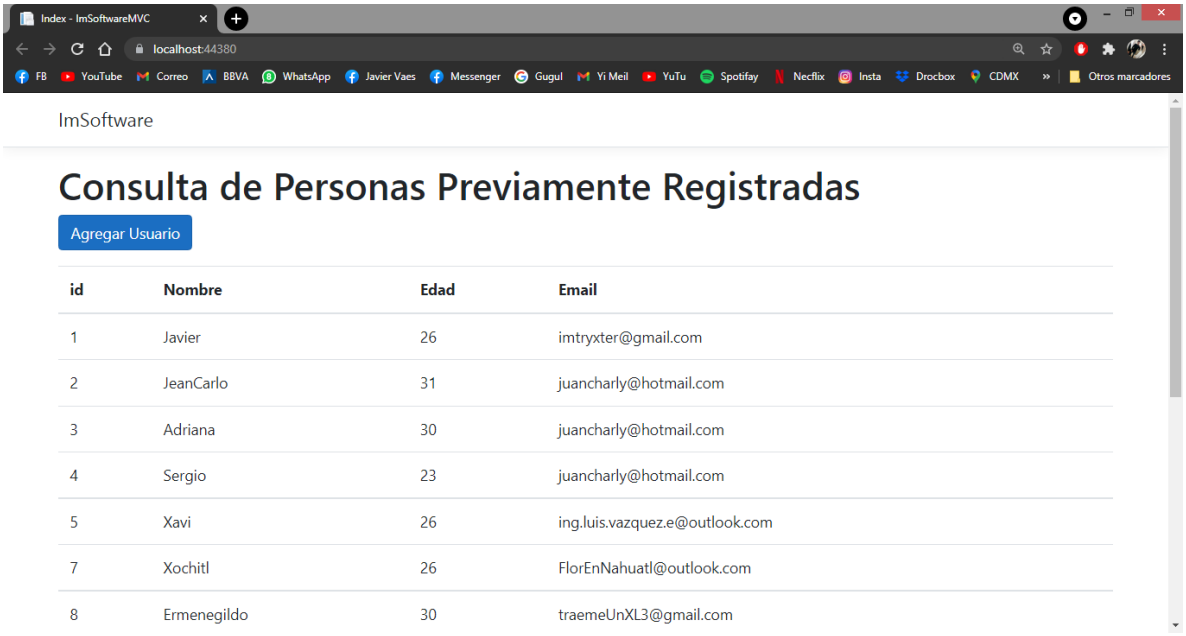


Se generan las vistas pertinentes solicitadas en el punto 5;

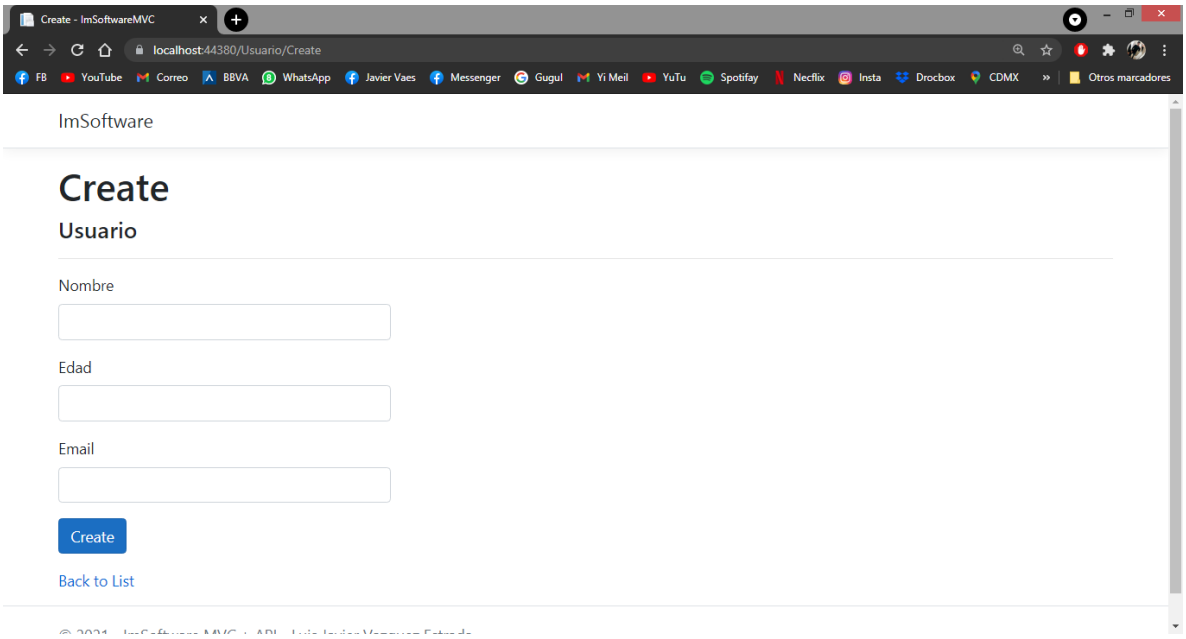
● Únicamente una vista donde se pueda visualizar una pantalla de registro que apunte al servicio de registro de personas de la Web API.

● Y una vista donde se pueda visualizar una consulta de las personas previamente registradas en la Web API.

Así se ve finalmente el Index.

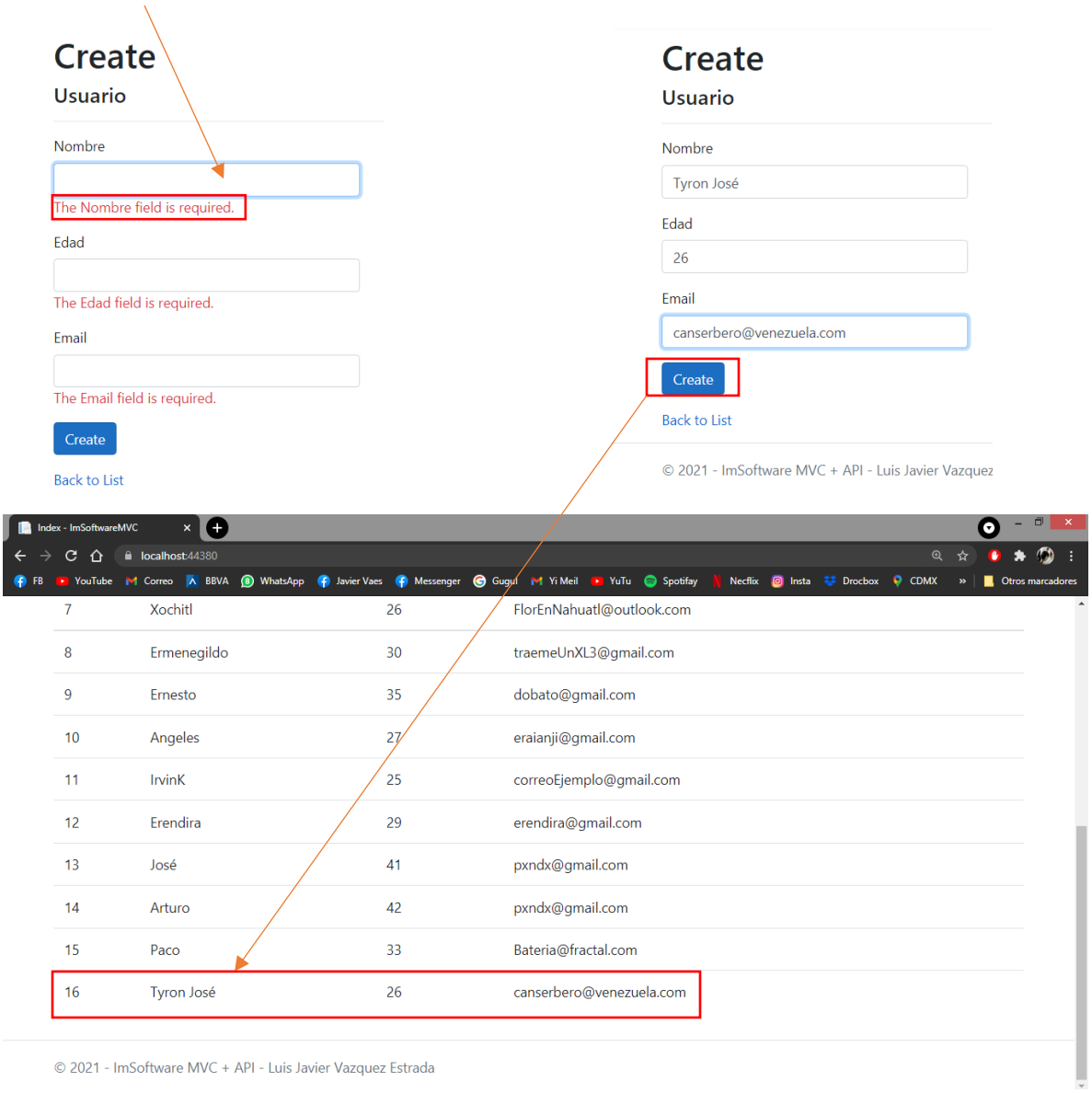



Y la vista de Agregar Usuario



Creación de un nuevo Usuario

VALIDACIÓN



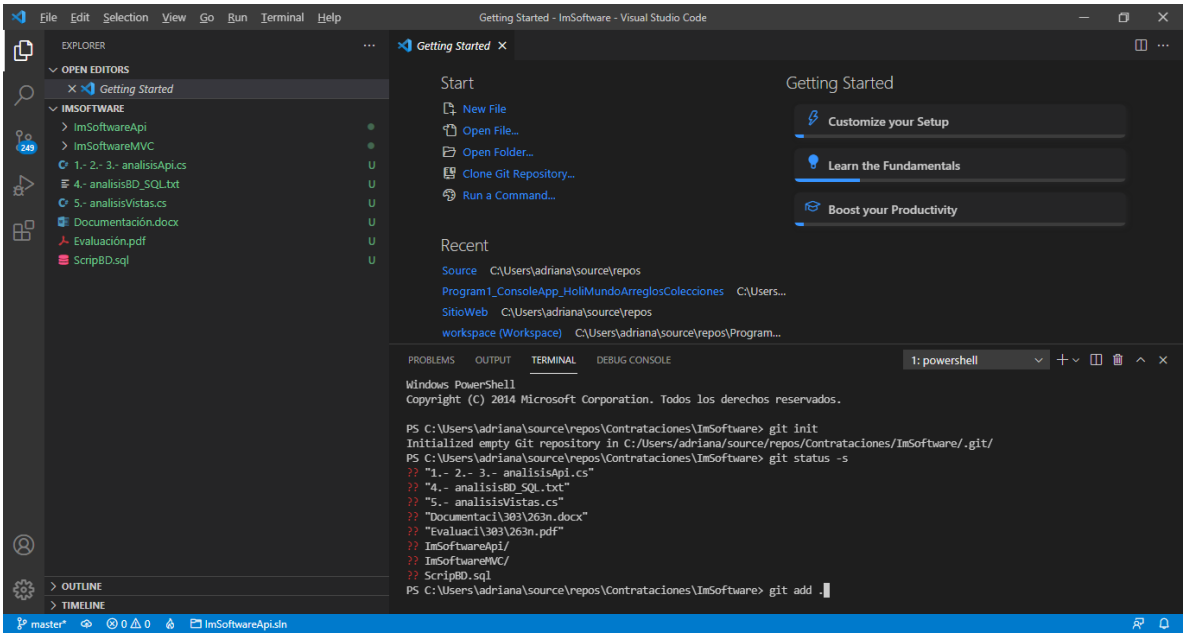

Redireccionamiento y presentación del nuevo registro al fondo de la tabla

Así bien vemos como se refresca de forma correcta, dinámicamente y Asíncrona

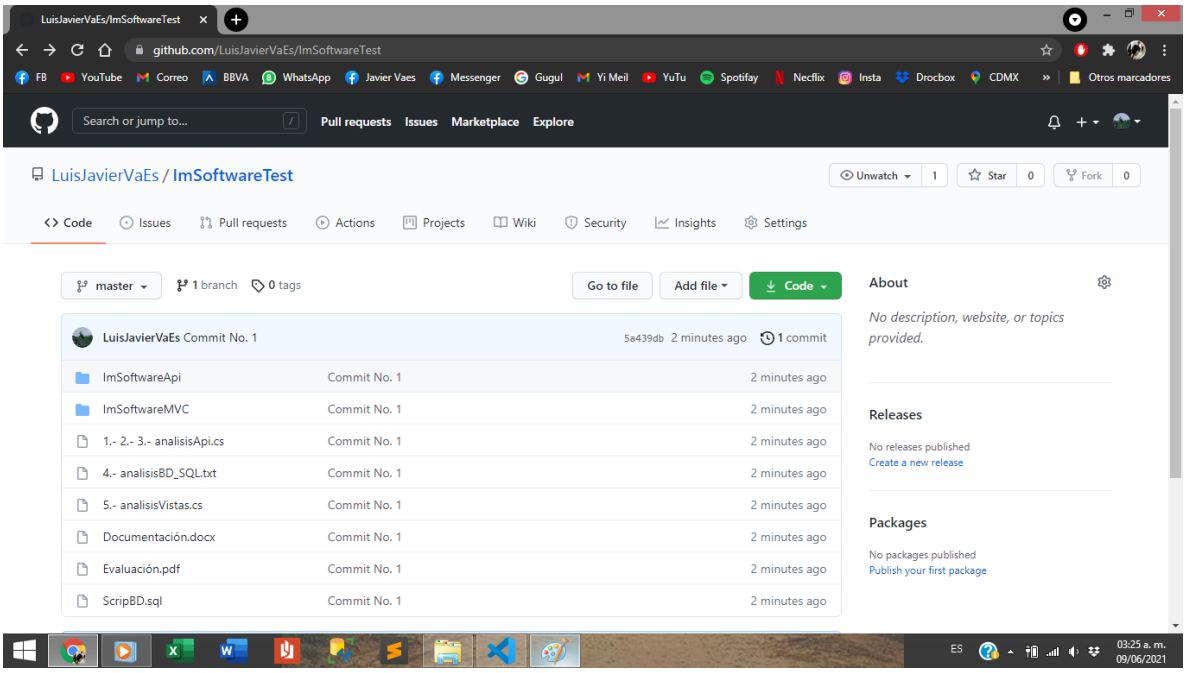
# Git

Instalación y subida con Visual Studio Code, comandos Git y respaldo a GitHub

***GIT***




***GITHUB***




Repositorio: https://github.com/LuisJavierVaEs/ImSoftwareTest.git