UNIVERSIDAD PRIVADA-DE-TACNA

INGENIERIA DE SISTEMAS

TITULO:

INFORME DE LABORATORIO No 01

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Cuadros Quiroga

Integrantes:

Orlando Antonio Acosta Ortiz	(2015052775)
Orestes Ramirez Ticona	(2015053236)
Nilson Laura Atencio	(2015053846)
Roberto Zegarra Reyes	(2010036175)
Richard Cruz Escalante	(2013047247)

Índice

1. INFORMACIÓN GENERAL

1.1. Objetivos:

- Conocer los fundamentos del servicio REST Web Api.
- Saber usar los controladores usando Entity Framework.
- Poder intercambiar datos a la base de datos con el formato JSON usando Postman.

1.2. Equipos, materiales, programas y recursos utilizados:

- Computadora con sistema operativo Windows 10.
- Microsoft Visual Studio 2017
- Postman 7.0.7 para (x86) o (x64)

2. MARCO TEORICO

2.1. Postman:

- gestiona y construye tus APIs rápidamente,Postman surgió originariamente como una extensión para el navegador Google Chrome. A día de hoy dispone de aplicaciones nativas para MAC y Windows y están trabajando en una aplicación nativa para Linux (disponible en versión beta).

- Está compuesto por diferentes herramientas y utilidades gratuitas (en la versión free) que permiten realizar tareas diferentes dentro del mundo API REST: creación de peticiones a APIs internas o de terceros, elaboración de tests para validar el comportamiento de APIs, posibilidad de crear entornos de trabajo diferentes (con variables globales y locales), y todo ello con la posibilidad de ser compartido con otros compañeros del equipo de manera gratuita (exportación de toda esta información mediante URL en formato JSON).

- Además, dispone de un modo cloud colaborativo (de pago) para que equipos de trabajo puedan desarrollar entre todos colecciones para APIs sincronizadas en la nube para una integración más inmediata y sincronizada.

2.2. JSON :

- es un formato ligero de intercambio de datos.

- está constituído por dos estructuras: Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o sequencias.

2.3. API REST :

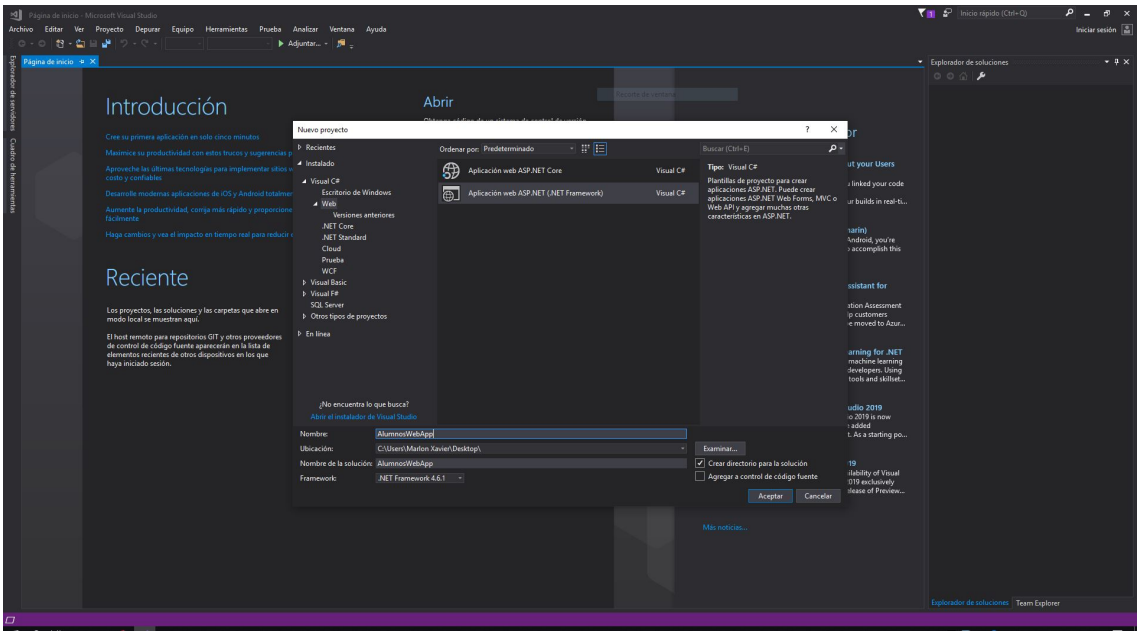
- El término REST (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación HTTP. Un servicio REST no es una arquitectura software, sino un conjunto de restricciones con las que podemos crear un estilo de arquitectura software, la cual podremos usar para crear aplicaciones web respetando HTTP.

-Algunas características de una API REST -Las operaciones más importantes que nos permitirán manipular los recursos son cuatro: GET para consultar y leer, POST para crear, PUT para editar y DELETE para eliminar.

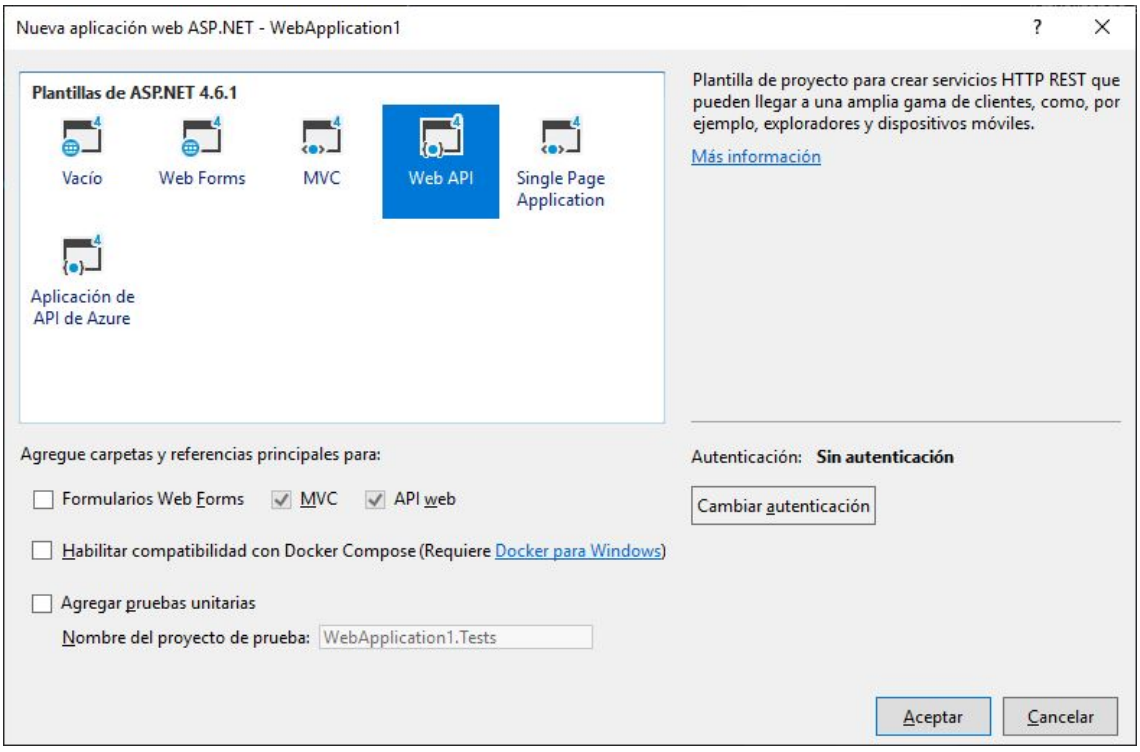
-El uso de hipermedios (término que en el ámbito de las páginas web define el conjunto de procedimientos para crear contenidos que contengan texto, imagen, vídeo, audio y otros métodos de información) para permitir al usuario navegar por los distintos recursos de una API REST a través de enlaces HTML.

3. PROCEDIMIENTO

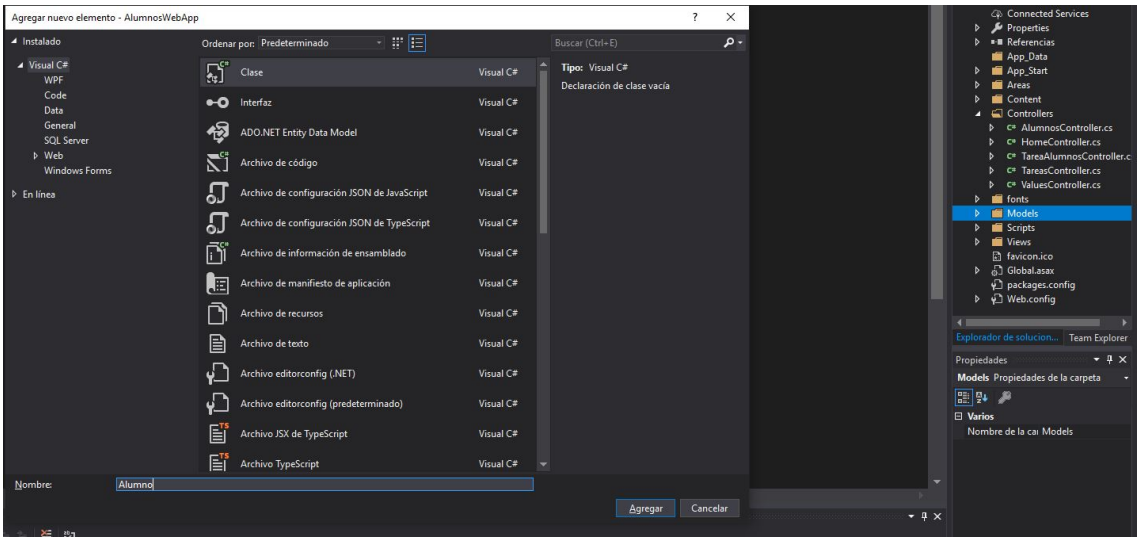
– Paso 1: Abrir Visual Studio, Crear una solucion Web [Aplicacion web ASP.NET(.NET Framework)]



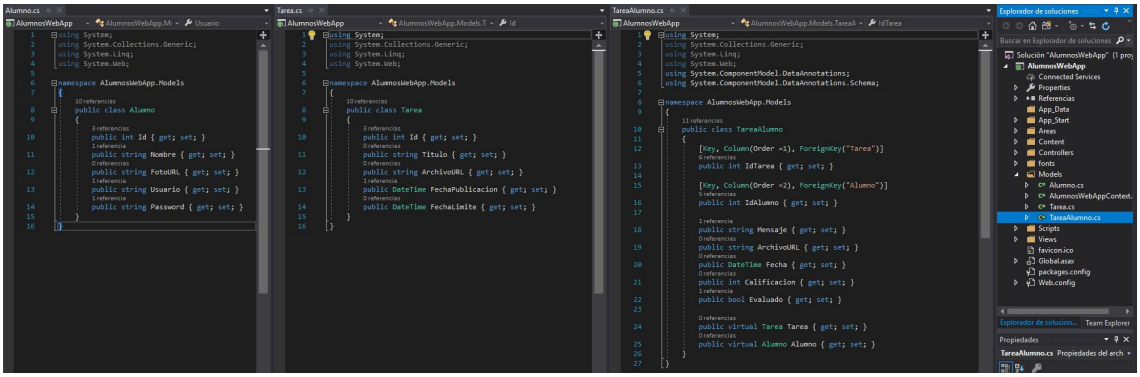
– Paso 2: Seleccionar la plantilla (Web API) con las opciones (MVC y API web) marcadas



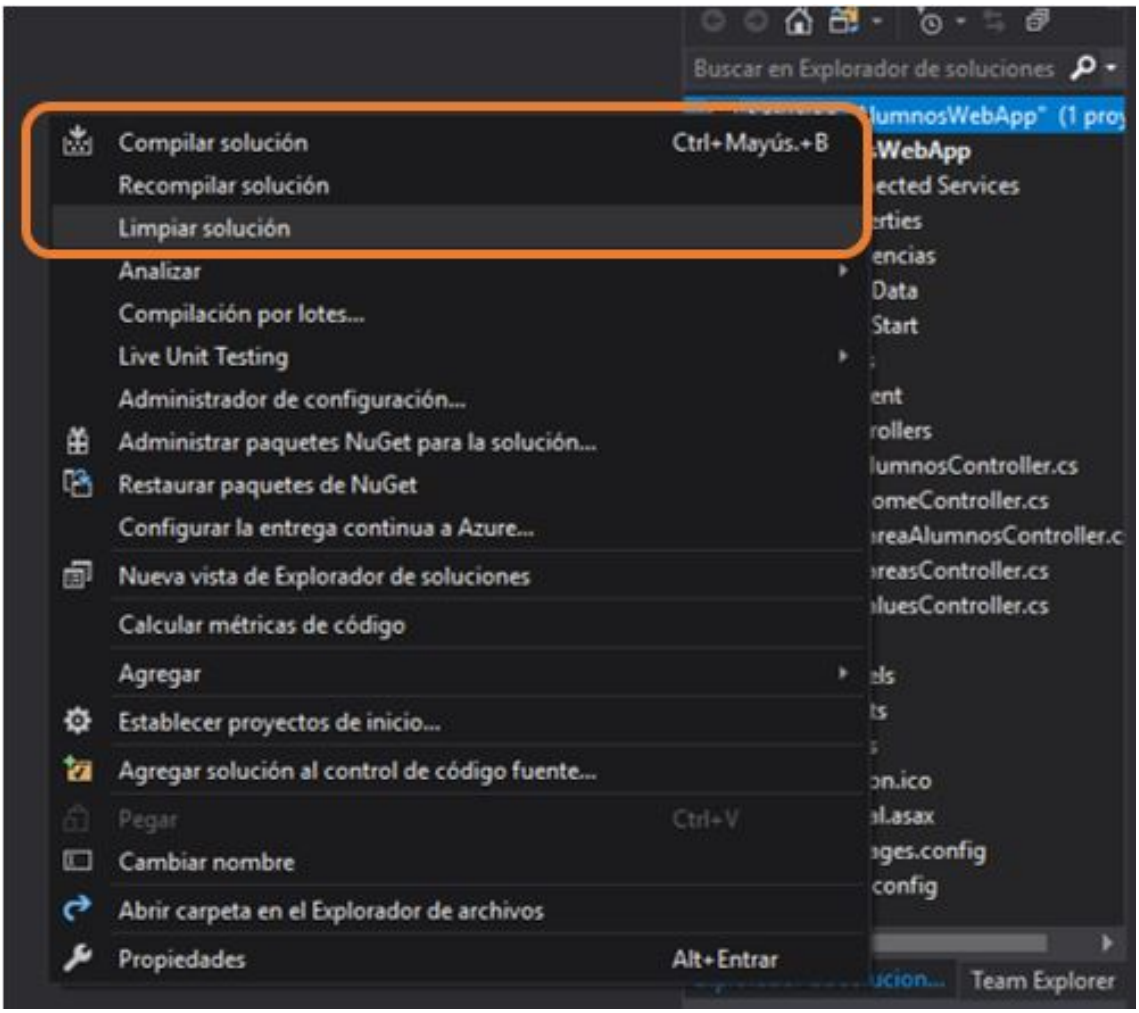
– Paso 3: Una vez creada la solución, dentro de la carpeta (Models)
Agregar 3 clases llamadas (Alumno, Tarea, TareaAlumno)



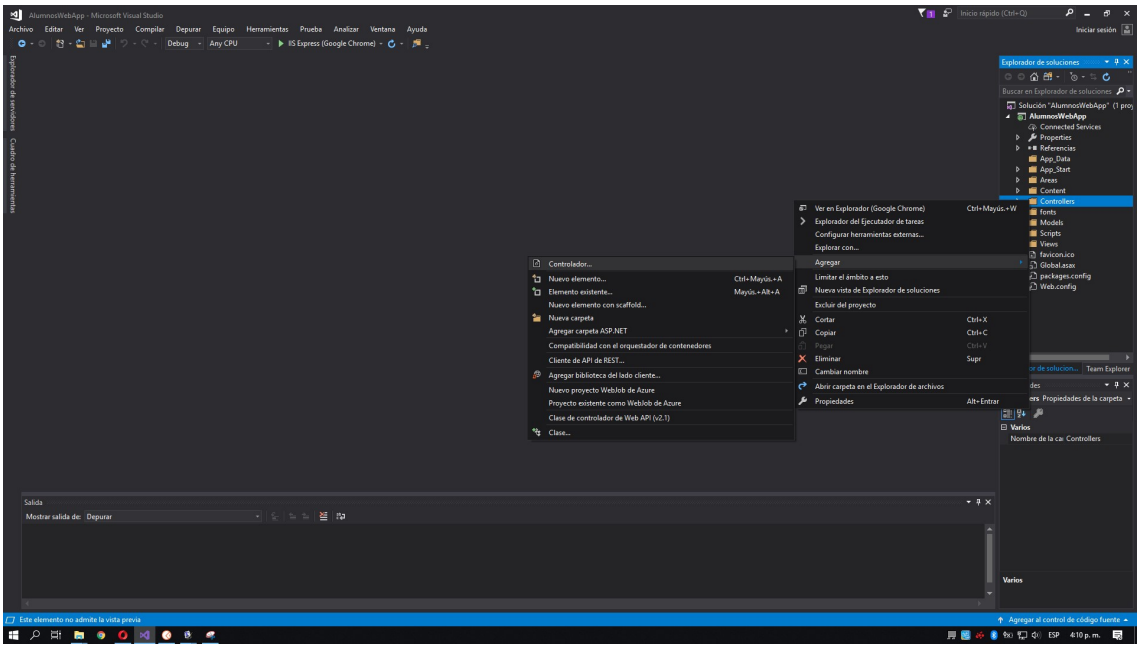
– Paso 4:Dentro de cada clase ingresar el codigo correspondiente a la siguiente imagen (tener en cuenta agregar en la clase TareaAlumno:
using System.Component.Model.DataAnnotations;
using System.Component.Model.DataAnnotations.Schema;)



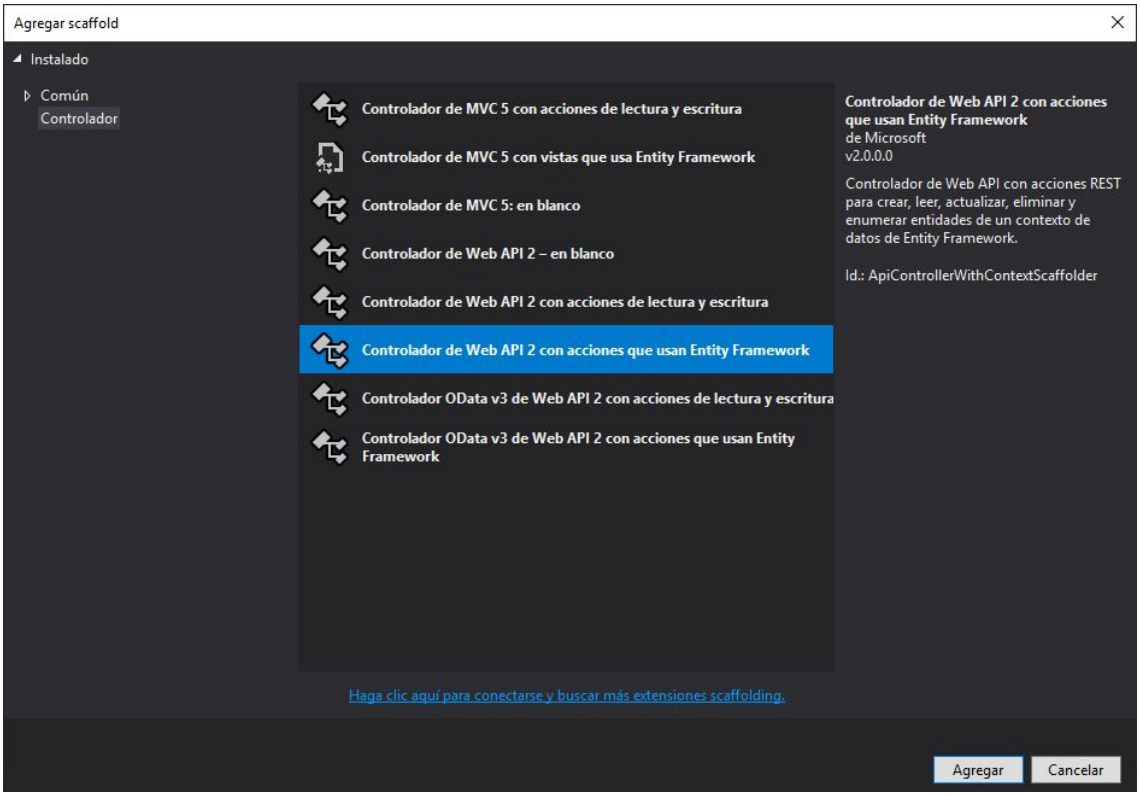
– Paso 5: Luego de terminar con las clases en Models, Pulsamos click derecho en la raiz de la solucion y por seguridad seleccionar las 3 primeras opciones(Para evitar errores al continuar con los siguientes pasos)



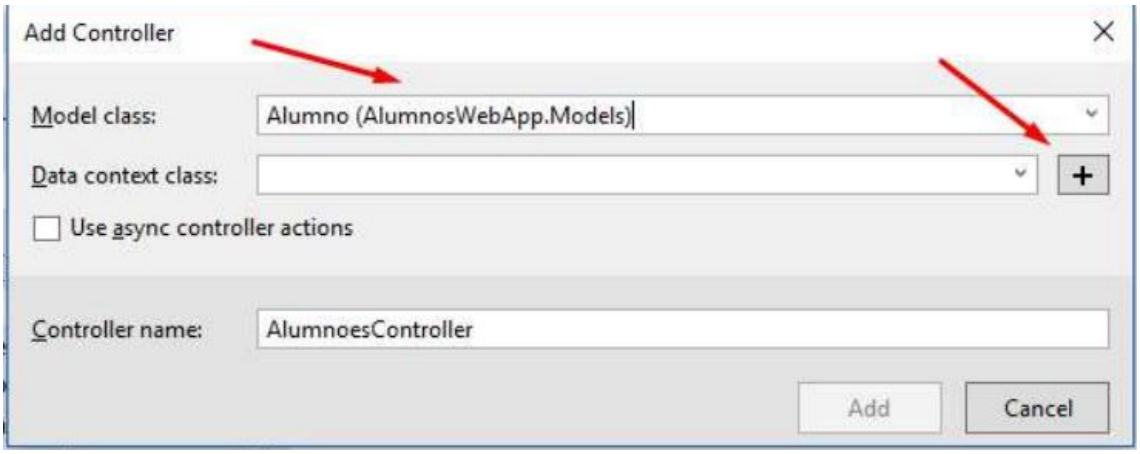
– Paso 6: Ahora hay que dirigirnos a la carpeta Controllers
En la carpeta le damos a Agregar - Controlador



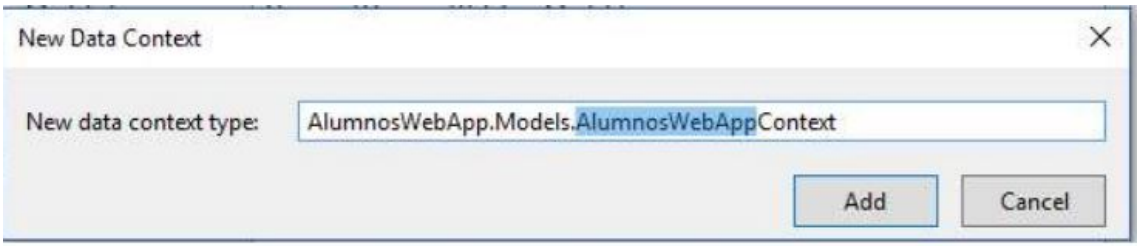
– Paso 7: En la ventana que nos aparece, Seleccionar la Opcion
Controlador de Web API 2 con acciones que usan Entity Framework



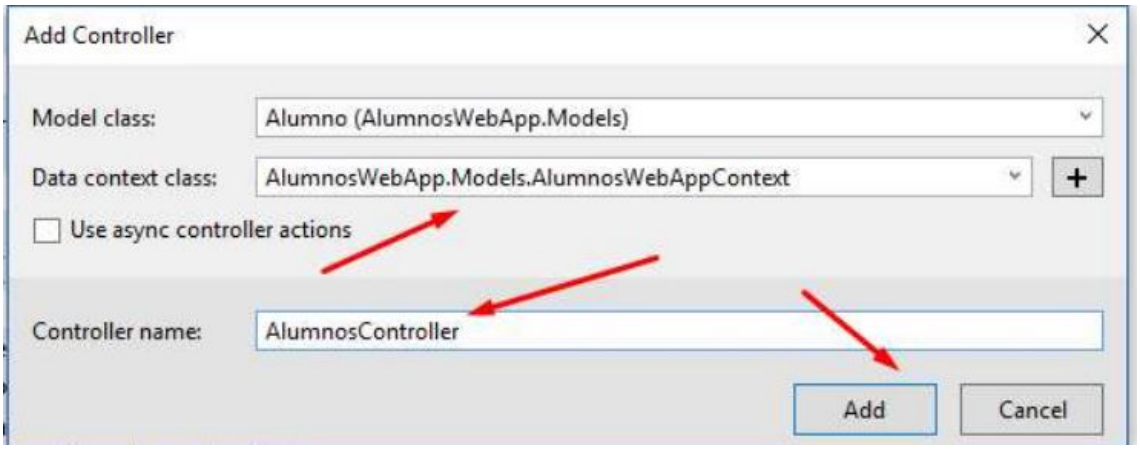
– Paso 8: Aparecerá una ventana para Agregar Controllador
hacer lo mostrado en la imagen



– Paso 9: Por defecto nos genera el nombre para el Nuevo Contexto de Datos, pulsamos Agregar o Add



– Paso 10: Cambiamos el Nombre del Controlador de (AlumnoesController a AlumnosController) y Pulsamos Add



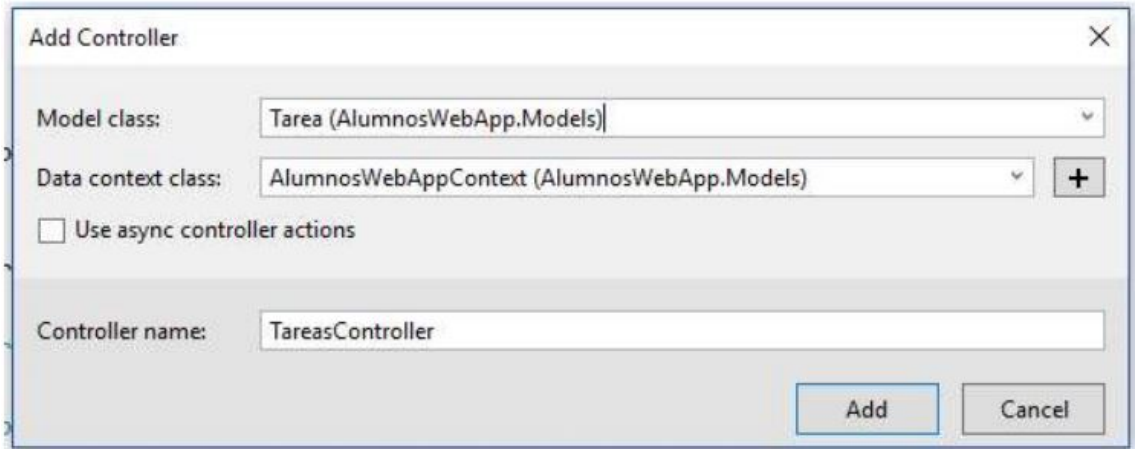
– Paso 11: Una vez creado el controlador AlumnosController
Modificamos y agregamos condigo dentro de //GET: api/Alumnos

```
        // GET: api/Alumnos
        public IQueryable<Alumno> GetAlumnoes()
        {
            return db.Alumnoes.OrderBy(x => x.Nombre);
        }


        [ResponseType(typeof(Alumno))]
        [Route("api/Alumnos/GetAlumnoByCredentials/{usuario}/{password}")]
        public IHttpActionResult GetAlumnoByCredentials(string usuario, string password)
        {
            Alumno alumno = db.Alumnoes.Where(x => x.Usuario == usuario && x.Password ==
password).FirstOrDefault();
            if (alumno == null)
            {
                return NotFound();
            }

            return Ok(alumno);
        }
```

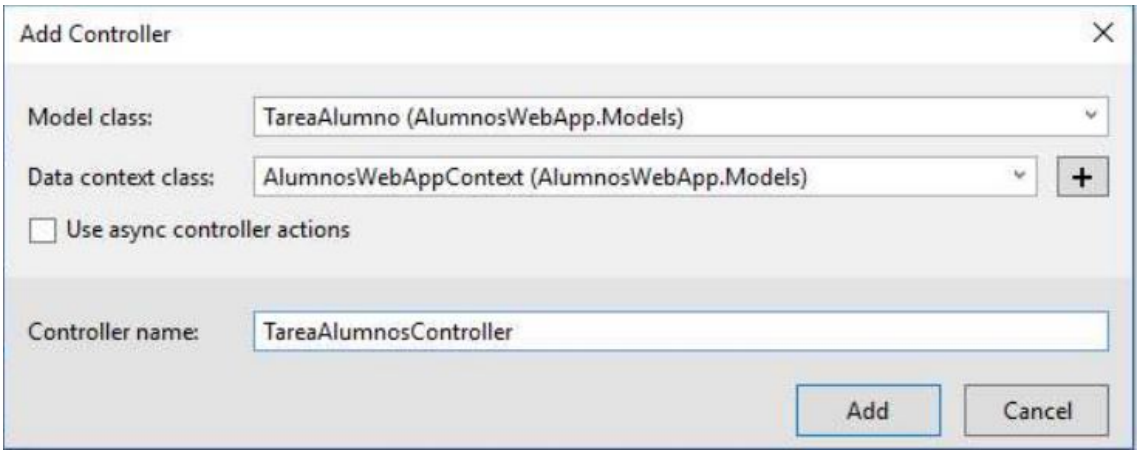
– Paso 12: Al igual que el anterior controllador, Agregamos un controlador nuevo llamado TareasController



– Paso 13: Ahora dentro del controlador TareasController solo modificamos en // GET: api:Tareas la linea (return db.Tareas)

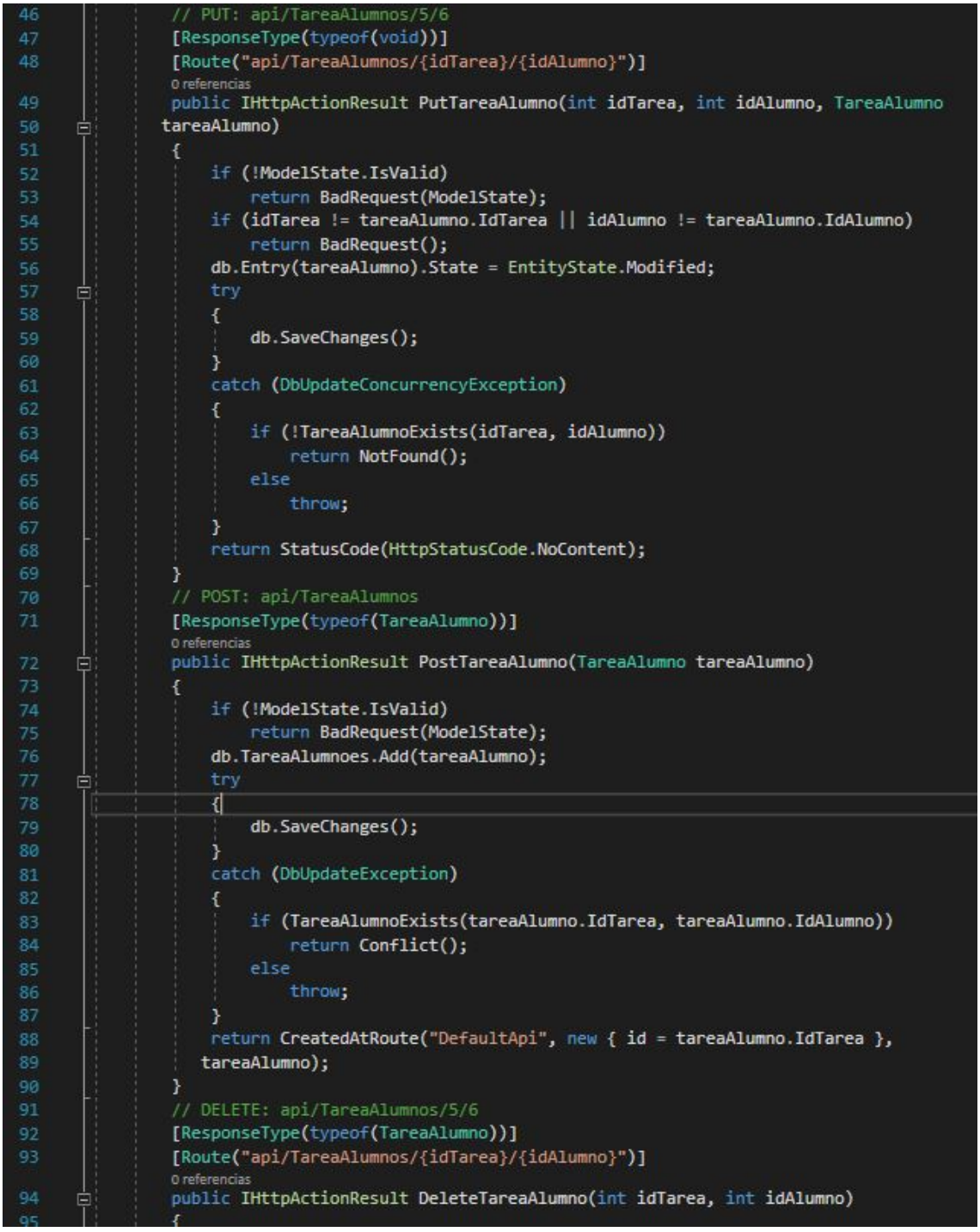
```
// GET: api/Tareas
//modificar return db.Tareas
0 referencias
public IQueryable<Tarea> GetTareas()
{
    return db.Tareas.OrderByDescending(x => x.FechaPublicacion);
}
```

– Paso 14: Agregamos un ultimo controlador, Este se llamara TareaAlumnosController



– Paso 15: Reemplazamos todo el código con las siguientes imagenes

```
using System;
using System.Data;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Linq;
using System.Net;
using System.Web.Http;
using System.Web.Http.Description;
using AlumnosWebApp.Models;
using System.Collections.Generic;
namespace AlumnosWebApp.Controllers
{
    0 referencias
    public class TareaAlumnosController : ApiController
    {
        private AlumnosWebAppContext db = new AlumnosWebAppContext();
        // GET: api/TareaAlumnos
        0 referencias
        public IQueryable<TareaAlumno> GetTareaAlumnoes()
        {
            return db.TareaAlumnoes;
        }
        [Route("api/TareaAlumnos/GetTareaAlumnosByEval/{evaluado}")]
        0 referencias
        public List<TareaAlumno> GetTareaAlumnosByEval(bool evaluado)
        {
            var data = db.TareaAlumnoes.Where(x => x.Evaluado == evaluado).ToList();
            db.Configuration.LazyLoadingEnabled = false;
            return data;
        }
        // GET: api/TareaAlumnos/5/6
        [ResponseType(typeof(TareaAlumno))]
        [Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
        0 referencias
        public IHttpActionResult GetTareaAlumno(int idTarea, int idAlumno)
        {
            try
            {
                TareaAlumno tareaAlumno = db.TareaAlumnoes.Where(x => x.IdTarea ==
               idTarea && x.IdAlumno == idAlumno).FirstOrDefault();
                if (tareaAlumno == null)
                    return NotFound();
                return Ok(tareaAlumno);
            }
            catch (Exception ex)
            {
                return Ok(new TareaAlumno() { Mensaje = ex.Message });
            }
        }
        // PUT: api/TareaAlumnos/5/6
        [ResponseType(typeof(void))]
        [Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
        0 referencias
        public IHttpActionResult PutTareaAlumno(int idTarea, int idAlumno, TareaAlumno
       tareaAlumno)
        {
```

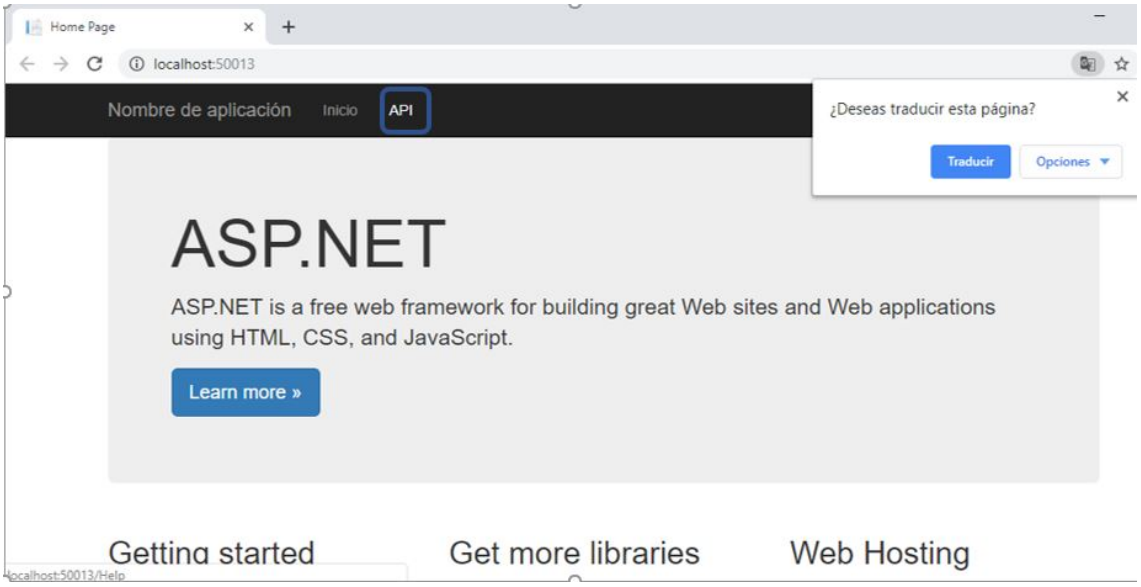


```
46          // PUT: api/TareaAlumnos/5/6
47          [ResponseType(typeof(void))]
48          [Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
            0 referencias
49          public IHttpActionResult PutTareaAlumno(int idTarea, int idAlumno, TareaAlumno
50      tareaAlumno)
51          {
52              if (!ModelState.IsValid)
53                  return BadRequest(ModelState);
54              if (idTarea != tareaAlumno.IdTarea || idAlumno != tareaAlumno.IdAlumno)
55                  return BadRequest();
56              db.Entry(tareaAlumno).State = EntityState.Modified;
57              try
58              {
59                  db.SaveChanges();
60              }
61              catch (DbUpdateConcurrencyException)
62              {
63                  if (!TareaAlumnoExists(idTarea, idAlumno))
64                      return NotFound();
65                  else
66                      throw;
67              }
68              return StatusCode(HttpStatusCode.NoContent);
69          }
70          // POST: api/TareaAlumnos
71          [ResponseType(typeof(TareaAlumno))]
            0 referencias
72          public IHttpActionResult PostTareaAlumno(TareaAlumno tareaAlumno)
73          {
74              if (!ModelState.IsValid)
75                  return BadRequest(ModelState);
76              db.TareaAlumnoes.Add(tareaAlumno);
77              try
78              {
79                  db.SaveChanges();
80              }
81              catch (DbUpdateException)
82              {
83                  if (TareaAlumnoExists(tareaAlumno.IdTarea, tareaAlumno.IdAlumno))
84                      return Conflict();
85                  else
86                      throw;
87              }
88              return CreatedAtRoute("DefaultApi", new { id = tareaAlumno.IdTarea },
89            tareaAlumno);
90          }
91          // DELETE: api/TareaAlumnos/5/6
92          [ResponseType(typeof(TareaAlumno))]
93          [Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
            0 referencias
94          public IHttpActionResult DeleteTareaAlumno(int idTarea, int idAlumno)
95          {
```

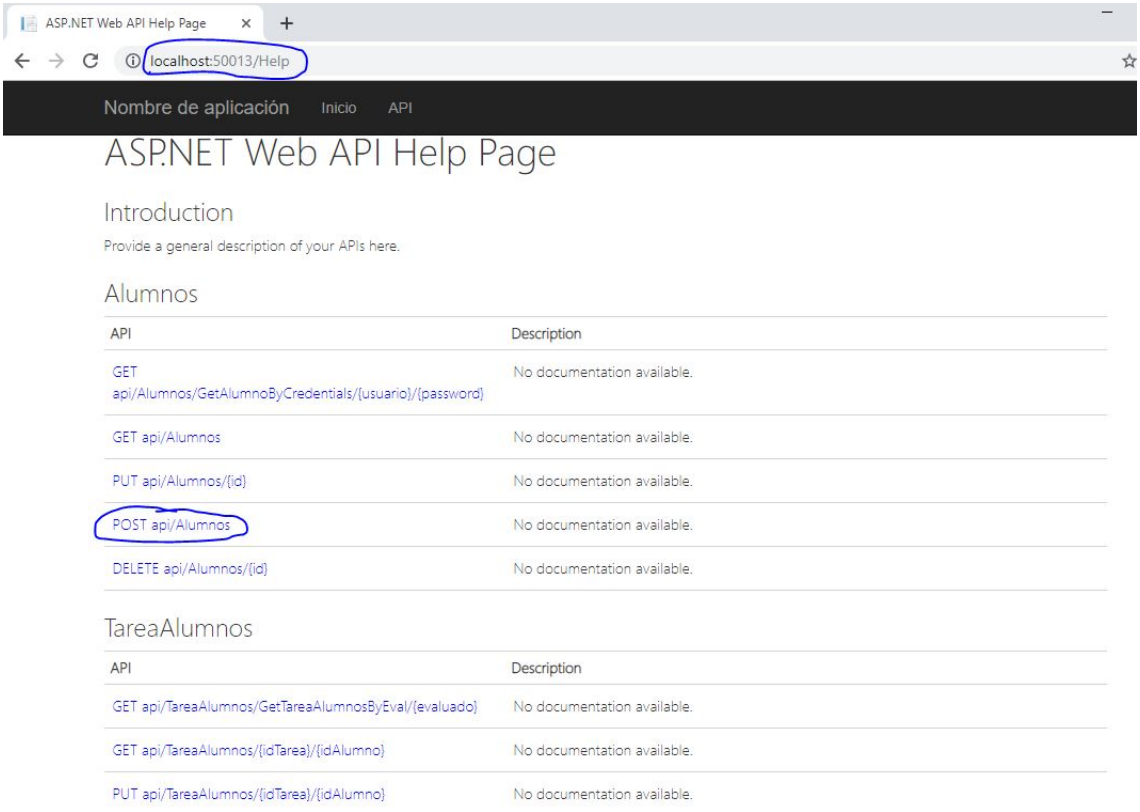
```
        }
        // DELETE: api/TareaAlumnos/5/6
        [ResponseType(typeof(TareaAlumno))]
        [Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
        0 referencias
        public IHttpActionResult DeleteTareaAlumno(int idTarea, int idAlumno)
        {
            TareaAlumno tareaAlumno = db.TareaAlumnoes.Where(x => x.IdTarea == idTarea &&
            x.IdAlumno == idAlumno).FirstOrDefault();
            if (tareaAlumno == null)
                return NotFound();
            db.TareaAlumnoes.Remove(tareaAlumno);
            db.SaveChanges();
            return Ok(tareaAlumno);
        }
        2 referencias
        protected override void Dispose(bool disposing)
        {
            if (disposing)
                db.Dispose();
            base.Dispose(disposing);
        }
        2 referencias
        private bool TareaAlumnoExists(int idTarea, int idAlumno)
        {
            return db.TareaAlumnoes.Count(e => e.IdTarea == idTarea && e.IdAlumno ==
            idAlumno) > 0;
        }
    }
}
```

– Paso 16: Una vez terminado, Ejecutamos la solucion

– Paso 17: Se abrirá un navegador (en este caso chrome) y debemos pulsar la opcion API

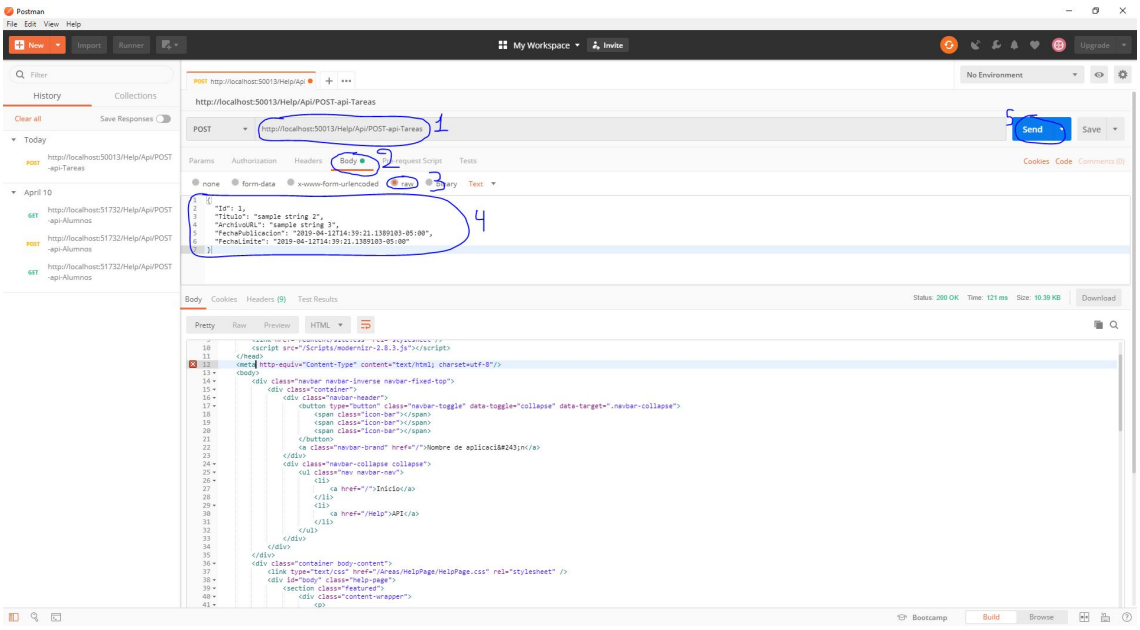


– Paso 18: Nos mandara a otra pagina en la que debemos buscar la opcion (POST api/Alumnos) y la seleccionamos



– Paso 19: Ahora hay que buscar un bloque de texto que se llame(application/json, text/json) Copiamos el ejemplo

– Paso 20: Descargamos Postman, y lo instalamos, para luego dentro de postman abrimos una pestaña nueva dentro del mismo y hacemos lo siguiente



4. ANALISIS E INTERPRETACION DE RESULTADOS

En el presente laboratorio se ha desarrollado un proyecto llamado .ªlumnosWebApp"de tipo Web Application, usando la Web API para realizar con ello el servicio de REST. Este proyecto trata acerca de calificar las tareas de los alumnos, haciendo primero el resgistro de los alumnos y sus tareas para luego relacionar las tareas con los alumnos correspondiente y de modo el cual se podrá poner una calificación. Se utilizaron tres clases, una para la crecion del .ªlumno", otra para la creacion de la "Tareaçon sus respectivos atributos y otra clase de "TareaAlumno", que esta útima se llama al atributo de IdTarea y IdAlumno, aparte de sus atributos respectivos, para que se relacionen y se califique la tarea del alumno.
Por medio de los controladores se usará la Web API por el cual podremos acceder a la web mediante el protocolo HTTP. Una ves que tenemos el codigo listo, se para a verificar que el proyecto funcione con el servicio de Rest, que nos permiten listar, crear, leer, actualizar y borrar información. Cuando solicitamos una pagina web, podemos hacer por diferentes métodos, el mas común es el GET, es el que usamos cuando digitamos una dirección en nuestro navegador, en ocasiones utilizamos POST, cuando enviamos un formulario con datos, pero las aplicaciones pueden usar otros métodos como PATCH, PUT, etc.

- Listar y leer: Usan el método GET
- Crear: Usan el método POST

Para validar el funcionamiento, primero se ejecuta el proyecto y vemos observamos que formato de la cadena hay que seguir para probar en Postman, que es una herramienta que permiten realizar tareas diferentes dentro del mundo API REST.

Realizamos primero la creacion de un alumno, siguiendo el formato de la caden. Para crear un alumno usamos la operación de POST y para verifiCar si se ha creado el alumno, realizamos la operacion de GET; y lo mismo para el caso de la tarea. Aqui se puede notar como se ha realizado correctamente el registro de ambos casos a través del servicio de REST y ahora lo que falta es vincular el alumno con su tarea y calificarla.
Para realizar la calificación de la tarea del alumno se sigue la primera parte del formato de cadena y ahi se coloca tanto el indice del alumno como de la tarea y los datos correspondientes a la calificación, usando la operación de POST.

Una vez realizada el registro de la calificación de la tarea del alumno con la operación GET podemos revisar si ha sido verdaderamente registrada. Gracias a estas operaciones de GET POST utilizadas la transferencia de datos en un sistema REST facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.

5. CONCLUSIONES

- Usar el servicio REST Web Api, se pudo notar que necesita mas código para su implementación.
- El servicio REST se basa en conocer sobre HTTP.
- Nosotros hemos usado el formato JSON pero para la implementación de REST se puede usar cualquier tipo de descripcion.

6. REFERENCIAS