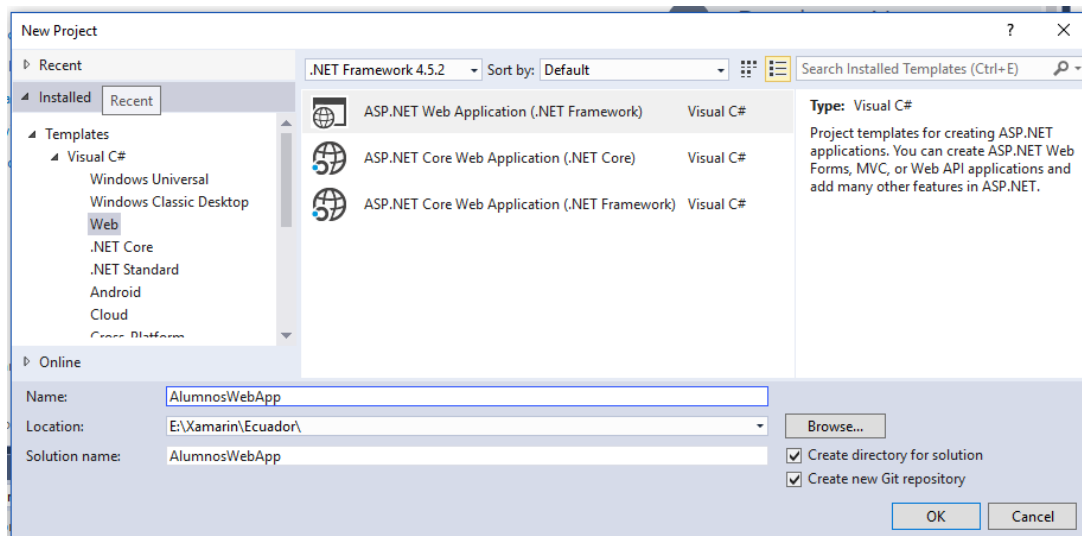


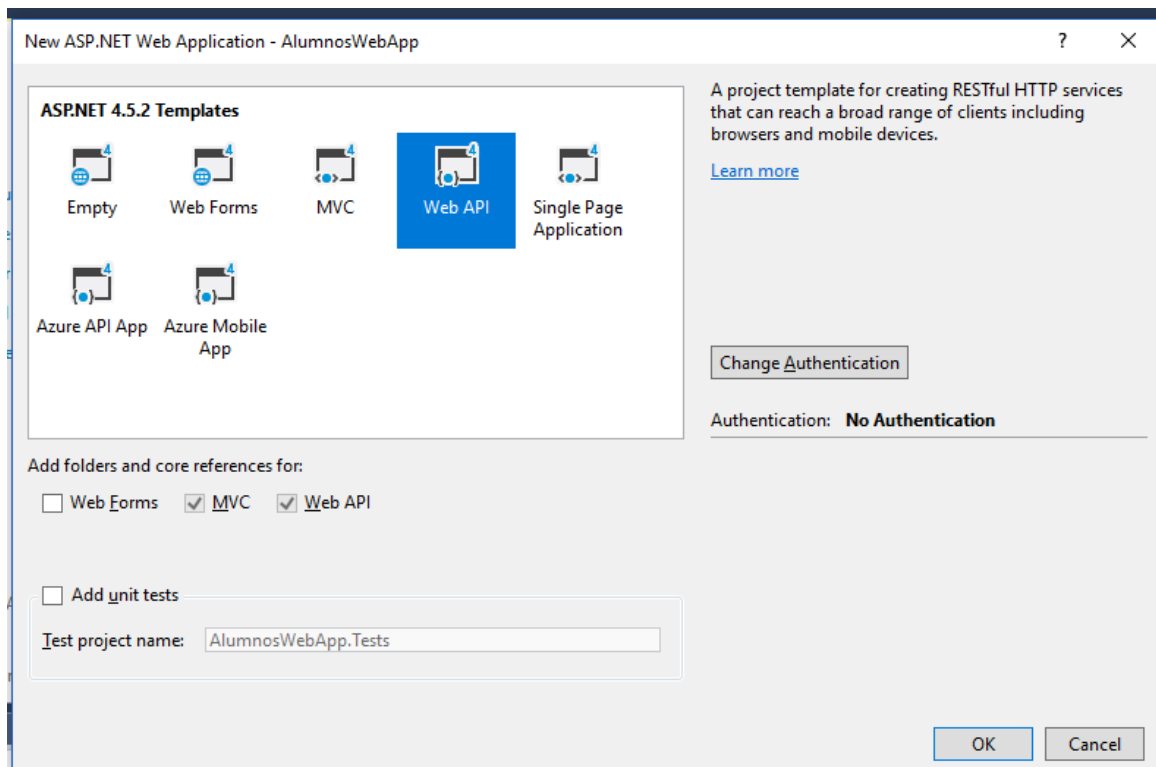
Parte 1: Creación del servicio REST Web API – Autor: Luis Beltrán

Para completar esta parte requieres instalar la herramienta **Postman** (<https://www.getpostman.com/>) a fin de visualizar el resultado de las peticiones web que realizaremos al servicio REST que construiremos.

Paso 1. De la categoría **Web**, selecciona el proyecto de tipo **ASP.NET Web Application (.NET Framework)** y coloca el nombre **AlumnosWebApp**.



Paso 2. Selecciona el template **Web API** y da clic en **OK**.



Paso 3. Agrega una nueva clase en la carpeta **Models** llamada **Alumno**. Esta clase representa una tabla que vamos a agregar a nuestra base de datos. Las propiedades representan los campos de la tabla. El código es el siguiente:

```
namespace AlumnosWebApp.Models
{
    public class Alumno
    {
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string FotoURL { get; set; }
        public string Usuario { get; set; }
        public string Password { get; set; }
    }
}
```

Paso 4. En la carpeta **Modelos** agrega otra clase llamada **Tarea**. Al igual que en el caso anterior, ésta es otra tabla que incluiremos en nuestra base de datos, con el código siguiente:

```
using System;

namespace AlumnosWebApp.Models
{
    public class Tarea
    {
        public int Id { get; set; }
        public string Titulo { get; set; }
        public string ArchivoURL { get; set; }
        public DateTime FechaPublicacion { get; set; }
        public DateTime FechaLimite { get; set; }
    }
}
```

Paso 5. Como última tabla, en nuestra carpeta **Modelos** agrega una clase llamada **TareaAlumno**, la cual representa una relación de N a N entre las tablas anteriores (Alumno y Tarea). Esto lo representaremos con los atributos **ForeignKey** y **Key** en las propiedades de ID, así como un elemento **virtual** para navegación estilo EntityFramework.

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace AlumnosWebApp.Models
{
    public class TareaAlumno
    {
        [Key, Column(Order = 1), ForeignKey("Tarea")]
        public int IdTarea { get; set; }

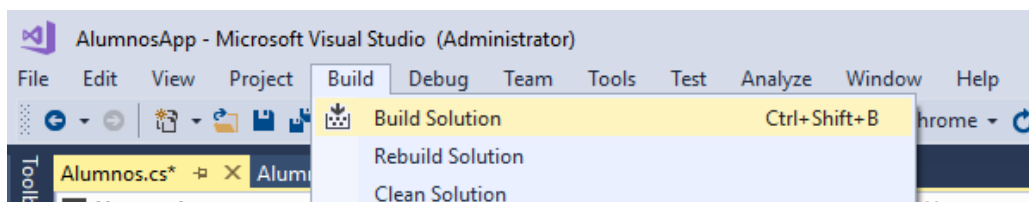
        [Key, Column(Order = 2), ForeignKey("Alumno")]
        public int IdAlumno { get; set; }

        public string Mensaje { get; set; }
        public string ArchivoURL { get; set; }
        public DateTime Fecha { get; set; }
        public int Calificacion { get; set; }
        public bool Evaluado { get; set; }

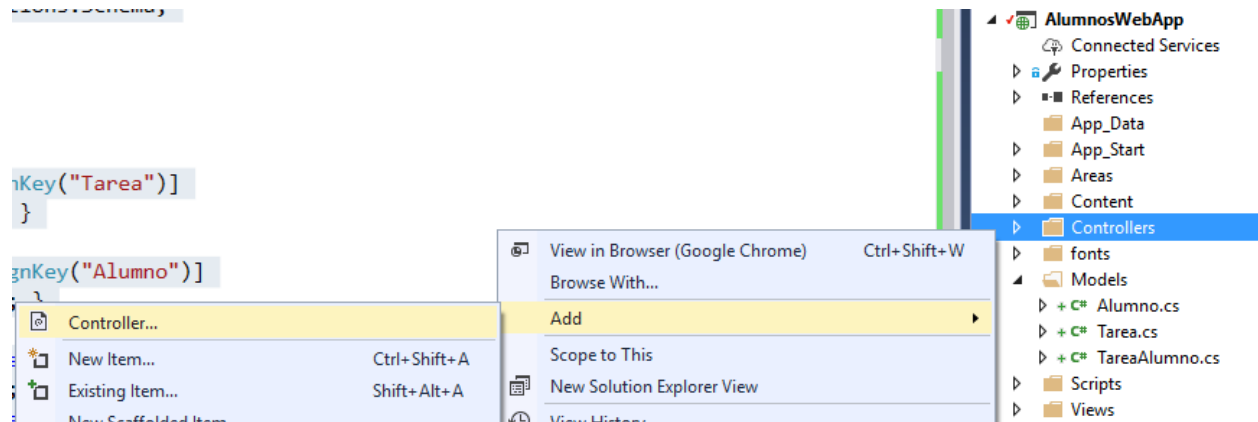
        public virtual Tarea Tarea { get; set; }
        public virtual Alumno Alumno { get; set; }
    }
}
```

Como puedes intuir, las clases creadas son modelos de tablas que serán generadas en una base de datos.

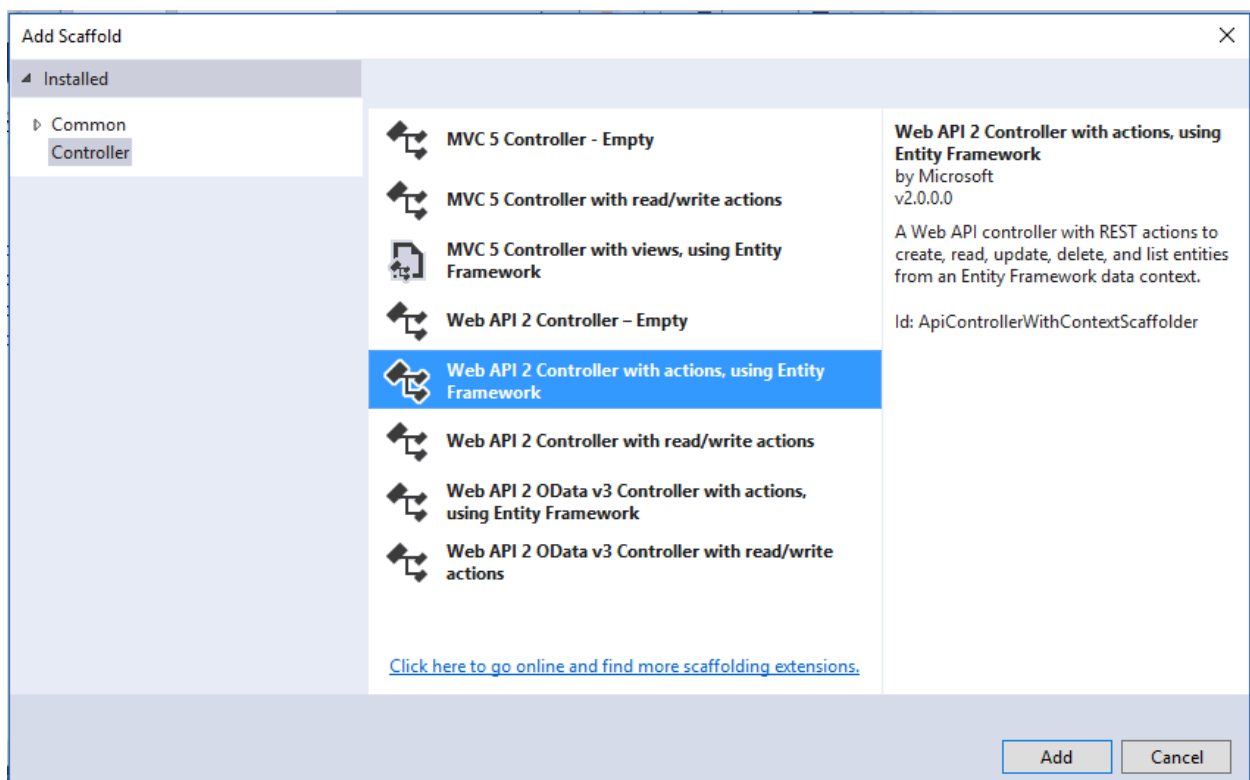
Paso 6. Después de crear tus modelos, **compila la solución.**



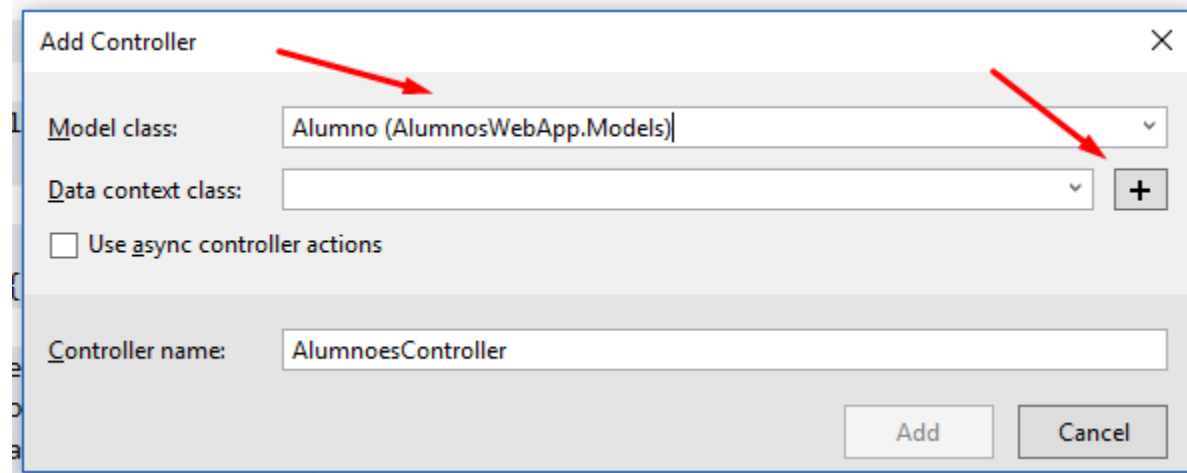
Paso 7. Ahora da clic derecho en la carpeta **Controllers** y agrega un nuevo **Controlador**.



Paso 8. Selecciona **Web API 2 Controller with actions, using EntityFramework** en la categoría de **Scaffolds**



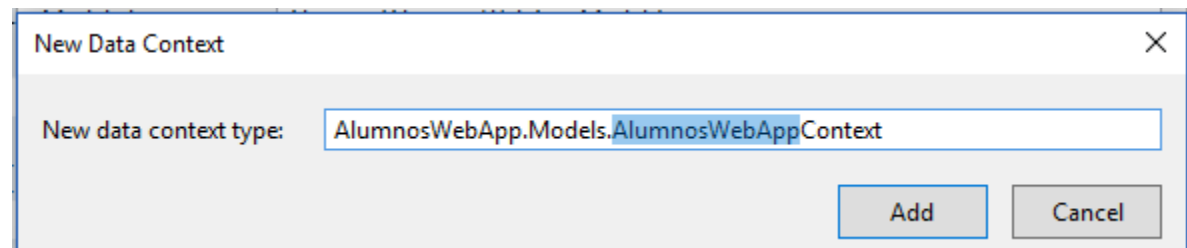
Paso 9. En clase de **Modelo** selecciona **Alumno** y da clic en el botón + para agregar una clase de contexto (conexión).



Dialog box titled "Add Controller".

- Model class: Alumno (AlumnosWebApp.Models)
- Data context class: [Empty] +
- ☐ Use async controller actions
- Controller name: AlumnoesController
- Buttons: Add, Cancel

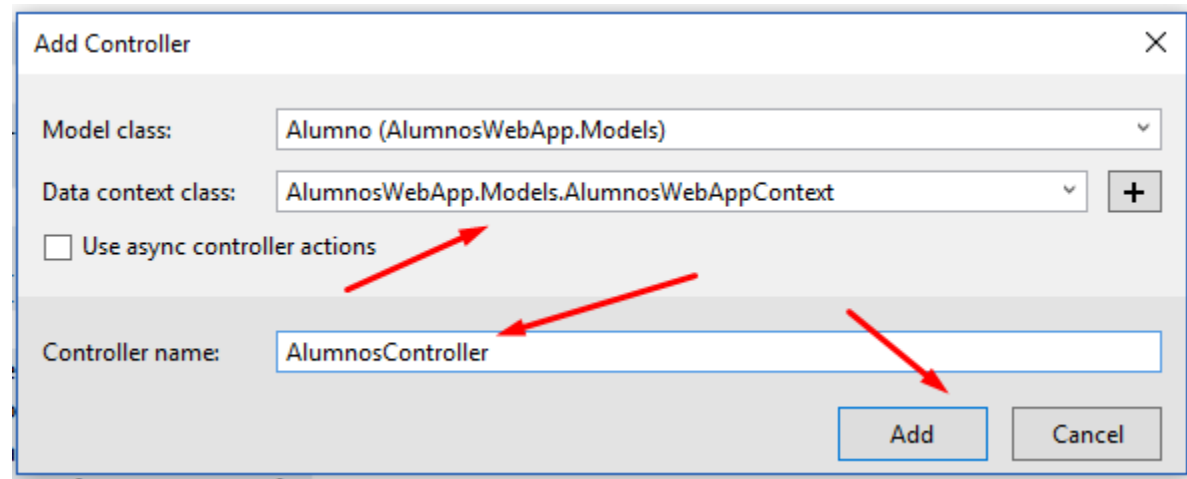
Paso 10. Da clic en **Agregar** (el nombre de la clase de contexto no es relevante, se recomienda dejarlo como aparece):



Dialog box titled "New Data Context".

- New data context type: AlumnosWebApp.Models.AlumnosWebAppContext
- Buttons: Add, Cancel

Paso 11. De regreso en la pantalla de **Agregar Cotrolador**, modifica el nombre del controlador a **AlumnosController** y da clic en **Agregar**.



Dialog box titled "Add Controller".

- Model class: Alumno (AlumnosWebApp.Models)
- Data context class: AlumnosWebApp.Models.AlumnosWebAppContext +
- ☐ Use async controller actions
- Controller name: AlumnosController
- Buttons: Add, Cancel

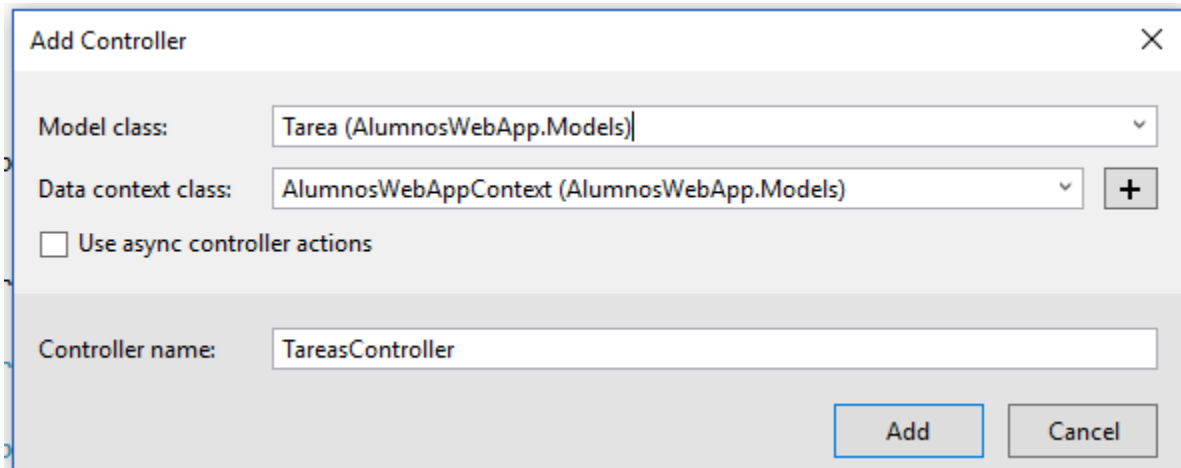
Paso 12. Modifica o agrega los siguientes métodos dentro del controlador:

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

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

    return Ok(alumno);
}
```

Paso 13. Agrega otro controlador con la siguiente información:

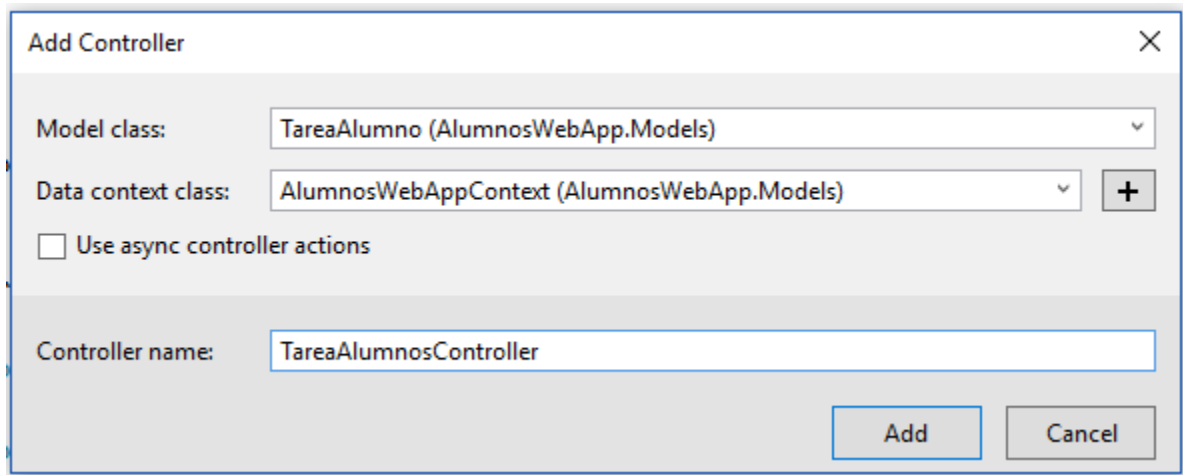


The screenshot shows the 'Add Controller' dialog box. The 'Model class' is 'Tarea (AlumnosWebApp.Models)'. The 'Data context class' is 'AlumnosWebAppContext (AlumnosWebApp.Models)'. The 'Use async controller actions' checkbox is unchecked. The 'Controller name' is 'TareasController'. The 'Add' button is highlighted.

Paso 14. Modifica el método GetTareas:

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

Paso 15. Agrega otro controlador con la siguiente información:



The screenshot shows the 'Add Controller' dialog box. It has a title bar with a close button. Inside, there are two dropdown menus: 'Model class' with 'TareaAlumno (AlumnosWebApp.Models)' selected, and 'Data context class' with 'AlumnosWebAppContext (AlumnosWebApp.Models)' selected. Below these is an unchecked checkbox labeled 'Use async controller actions'. At the bottom, there is a text box for 'Controller name' containing 'TareaAlumnosController'. At the bottom right are 'Add' and 'Cancel' buttons.

Paso 16. El código completo del controlador se muestra a continuación:

```
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
{
    public class TareaAlumnosController : ApiController
    {
        private AlumnosWebAppContext db = new AlumnosWebAppContext();

        // GET: api/TareaAlumnos
        public IQueryable<TareaAlumno> GetTareaAlumnos()
        {
            return db.TareaAlumnos;
        }

        [Route("api/TareaAlumnos/GetTareaAlumnosByEval/{evaluado}")]
        public List<TareaAlumno> GetTareaAlumnosByEval(bool evaluado)
        {
            var data = db.TareaAlumnos.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}")]
public IActionResult GetTareaAlumno(int idTarea, int idAlumno)
{
    try
    {
        TareaAlumno tareaAlumno = db.TareaAlumnos.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}")]
public IActionResult PutTareaAlumno(int idTarea, int idAlumno, TareaAlumno
tareaAlumno)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    if (idTarea != tareaAlumno.IdTarea || idAlumno != tareaAlumno.IdAlumno)
        return BadRequest();

    db.Entry(tareaAlumno).State = EntityState.Modified;

    try
    {
        db.SaveChanges();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!TareaAlumnoExists(idTarea, idAlumno))
            return NotFound();
        else
            throw;
    }

    return StatusCode(HttpStatusCode.NoContent);
}

// POST: api/TareaAlumnos
[ResponseType(typeof(TareaAlumno))]
public IActionResult PostTareaAlumno(TareaAlumno tareaAlumno)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    db.TareaAlumnos.Add(tareaAlumno);

    try
    {

```



```

        db.SaveChanges();
    }
    catch (DbUpdateException)
    {
        if (TareaAlumnoExists(tareaAlumno.IdTarea, tareaAlumno.IdAlumno))
            return Conflict();
        else
            throw;
    }

    return CreatedAtRoute("DefaultApi", new { id = tareaAlumno.IdTarea },
tareaAlumno);
}

// DELETE: api/TareaAlumnos/5/6
[ResponseType(typeof(TareaAlumno))]
[Route("api/TareaAlumnos/{idTarea}/{idAlumno}")]
public IHttpActionResult DeleteTareaAlumno(int idTarea, int idAlumno)
{
    TareaAlumno tareaAlumno = db.TareaAlumnos.Where(x => x.IdTarea == idTarea &&
x.IdAlumno == idAlumno).FirstOrDefault();

    if (tareaAlumno == null)
        return NotFound();

    db.TareaAlumnos.Remove(tareaAlumno);
    db.SaveChanges();

    return Ok(tareaAlumno);
}

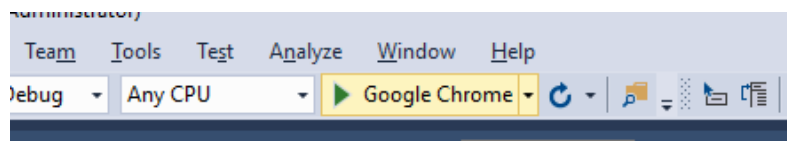
protected override void Dispose(bool disposing)
{
    if (disposing)
        db.Dispose();

    base.Dispose(disposing);
}

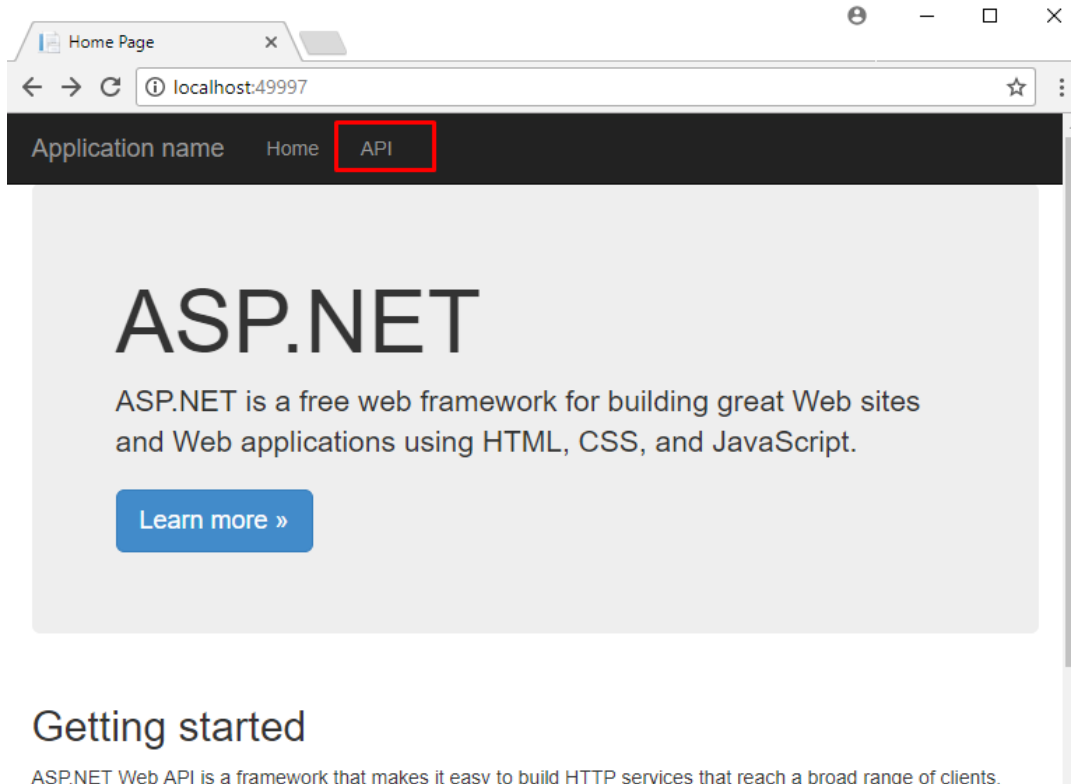
private bool TareaAlumnoExists(int idTarea, int idAlumno)
{
    return db.TareaAlumnos.Count(e => e.IdTarea == idTarea && e.IdAlumno ==
idAlumno) > 0;
}
}
}

```

Paso 17. Compila y ejecuta la aplicación:



Paso 18. Da clic en el enlace API a fin de observar la descripción de los controladores de nuestra aplicación:



Paso 19. Localiza el controlador **Alumnos** y sus métodos. Da clic en **POST Alumnos** para agregar un registro:

Alumnos

API	Description
GET api/Alumnos/GetAlumnoByCredentials/{usuario}/{password}	No documentation available.
GET api/Alumnos	No documentation available.
GET api/Alumnos/{id}	No documentation available.
PUT api/Alumnos/{id}	No documentation available.
POST api/Alumnos	No documentation available.
DELETE api/Alumnos/{id}	No documentation available.

Paso 20. Observa el formato de la cadena que debe ser enviada y cópiala:

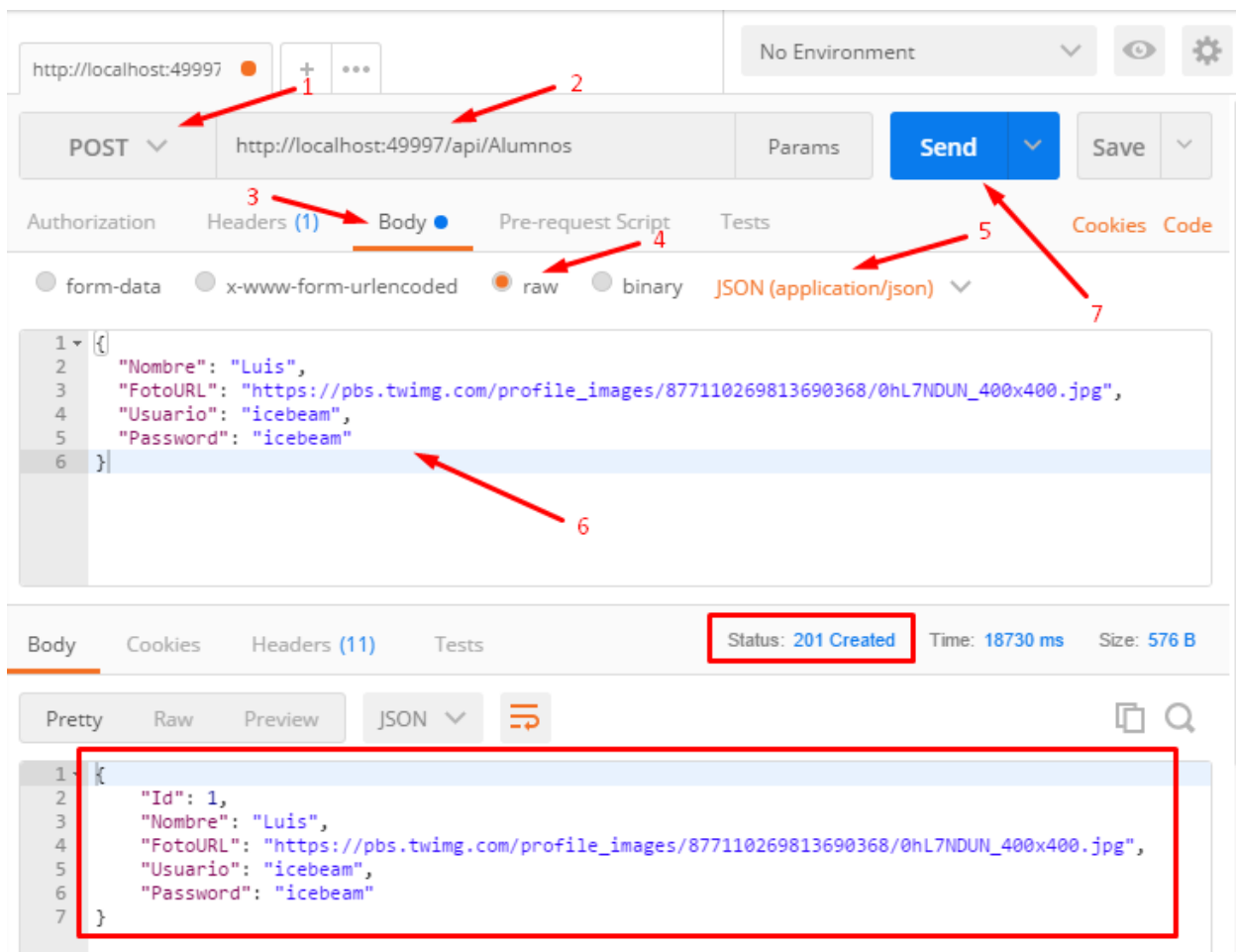
Request Formats

application/json, text/json

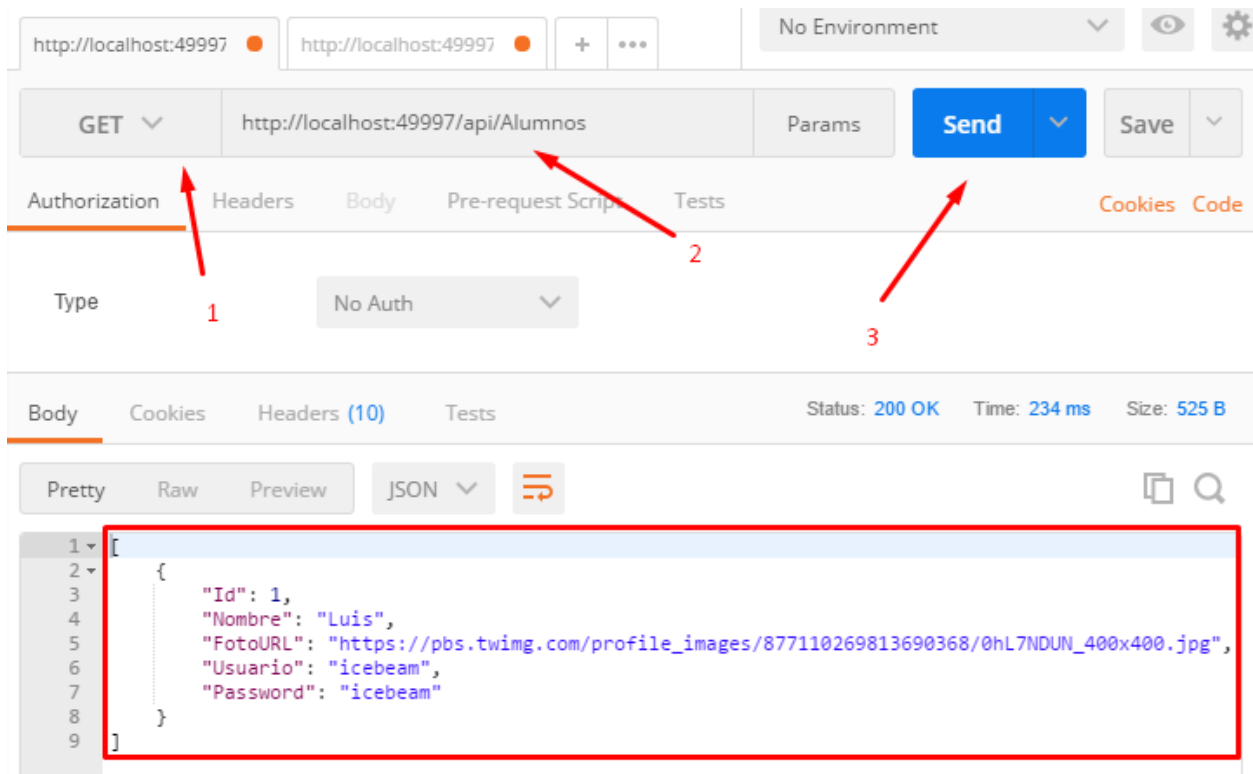
Sample:

```
{
  "Id": 1,
  "Nombre": "sample string 2",
  "FotoURL": "sample string 3",
  "Usuario": "sample string 4",
  "Password": "sample string 5"
}
```

Paso 21. Ahora abre la aplicación Postman y sigue los pasos para hacer una petición POST al servicio, lo cual nos permitirá agregar un nuevo alumno en nuestra base de datos:



Paso 22. Si hacemos una petición GET al api de Alumnos, podemos comprobar que el registro ha sido agregado con éxito:



Paso 23. Ahora agrega una nueva **Tarea**. Primero damos clic en el POST del api de Tareas

Tareas

API	Description
GET api/Tareas	No documentation available.
GET api/Tareas/{id}	No documentation available.
PUT api/Tareas/{id}	No documentation available.
POST api/Tareas	No documentation available.
DELETE api/Tareas/{id}	No documentation available.

Paso 24. Observamos el formato de la cadena JSON a introducir:

Request Formats

application/json, text/json

Sample:

```
{
  "Id": 1,
  "Titulo": "sample string 2",
  "ArchivoURL": "sample string 3",
  "FechaPublicacion": "2017-08-16T20:13:30.2287444+02:00",
  "FechaLimite": "2017-08-16T20:13:30.2287444+02:00"
}
```

Paso 25. Haz una petición POST al servicio para agregar un nuevo registro en Tareas:

The screenshot shows the Postman interface for a POST request. The URL is `http://localhost:49997/api/Tareas`. The request body is a JSON object:

```
{
  "Titulo": "Ecuaciones de 2º grado",
  "ArchivoURL": "https://drive.google.com/open?id=0ByGXRtB_vuMwclWmMG5oLXR0d1E",
  "FechaPublicacion": "2017-08-16",
  "FechaLimite": "2017-08-30"
}
```

The response status is **201 Created**. The response body is a JSON object:

```
{
  "Id": 1,
  "Titulo": "Ecuaciones de 2º grado",
  "ArchivoURL": "https://drive.google.com/open?id=0ByGXRtB_vuMwclWmMG5oLXR0d1E",
  "FechaPublicacion": "2017-08-16T00:00:00",
  "FechaLimite": "2017-08-30T00:00:00"
}
```

Paso 26. Finalmente, comprobamos que también funcione el api de **TareaAlumnos**. Para ello, damos clic en **POST** dicho api:

TareaAlumnos

API	Description
GET api/TareaAlumnos/GetTareaAlumnosByEval/{evaluado}	No documentation available.
GET api/TareaAlumnos/{idTarea}/{idAlumno}	No documentation available.
PUT api/TareaAlumnos/{idTarea}/{idAlumno}	No documentation available.
DELETE api/TareaAlumnos/{idTarea}/{idAlumno}	No documentation available.
GET api/TareaAlumnos	No documentation available.
POST api/TareaAlumnos	No documentation available.

Paso 27: Verificamos la cadena JSON, centrándonos únicamente en las propiedades que no son de navegación:

Request Formats

application/json, text/json

Sample:

```
{
  "IdTarea": 1,
  "IdAlumno": 2,
  "Mensaje": "sample string 3",
  "ArchivoURL": "sample string 4",
  "Fecha": "2017-08-18T14:32:58.995101+02:00",
  "Calificacion": 6,
  "Evaluado": true,
  "Tarea": {
    "Id": 1,
    "Titulo": "sample string 2",
    "ArchivoURL": "sample string 3",
    "FechaPublicacion": "2017-08-18T14:32:58.9961022+02:00",
    "FechaLimite": "2017-08-18T14:32:58.9961022+02:00"
  },
  "Alumno": {
    "Id": 1,
    "Nombre": "sample string 2",
    "FotoURL": "sample string 3",
    "Usuario": "sample string 4",
    "Password": "sample string 5"
  }
}
```

Paso 28: Hacemos la petición **POST** al api de **TareaAlumnos** en **Postman**:

The screenshot displays the Postman interface for a POST request. The URL bar shows `http://localhost:49997` and the environment is set to `No Environment`. The request method is `POST` and the URL is `http://localhost:49997/api/TareaAlumnos`. The request body is in `JSON (application/json)` format with the following content:

```
{
  "IdTarea": 1,
  "IdAlumno": 1,
  "Mensaje": "Esta es la solución a mi tarea",
  "ArchivoURL": "https://drive.google.com/file/d/0ByGXrTB_vuMwV1vVVFfYkItTmM/view?usp=sharing",
  "Fecha": "2017-08-16",
  "Calificacion": 0
}
```

The response status is `201 Created`, with a time of `107 ms` and a size of `668 B`. The response body is in `JSON` format and contains the following content:

```
{
  "IdTarea": 1,
  "IdAlumno": 1,
  "Mensaje": "Esta es la solución a mi tarea",
  "ArchivoURL": "https://drive.google.com/file/d/0ByGXrTB_vuMwV1vVVFfYkItTmM/view?usp=sharing",
  "Fecha": "2017-08-16T00:00:00",
  "Calificacion": 0,
  "Tarea": null,
  "Alumno": null
}
```

Paso 29: Comprobamos que ha sido agregado un nuevo registro en la tabla:

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

- URL:** `http://localhost:49997/Api/TareaAlumnos/`
- Method:** `GET`
- Status:** `200 OK`
- Time:** `111 ms`
- Size:** `959 B`

The response body is a JSON array containing two objects. The second object is highlighted with a red box:

```
[
  {
    "Alumno": {
      "Id": 1,
      "Nombre": "Luis",
      "FotoURL": "https://pbs.twimg.com/profile_images/877110269813690368/0hL7NDUN_400x400.jpg",
      "Usuario": "icebeam",
      "Password": "icebeam"
    },
    "Tarea": {
      "Id": 1,
      "Titulo": "Ecuaciones de 2° grado",
      "ArchivoURL": "https://drive.google.com/open?id=0ByGXrTB_vuMwclVvmMG5oLXR0d1E",
      "FechaPublicacion": "2017-08-16T00:00:00",
      "FechaLimite": "2017-08-30T00:00:00"
    },
    "IdTarea": 1,
    "IdAlumno": 1,
    "Mensaje": "Esta es la solución a mi tarea",
    "ArchivoURL": "https://drive.google.com/file/d/0ByGXrTB_vuMwclVlVVFfYkItTmM/view?usp=sharing",
    "Fecha": "2017-08-16T00:00:00",
    "Calificacion": 0
  }
]
```

Hemos concluido con la parte 1 de esta sesión. En la siguiente parte, aprenderás a publicar tu servicio en Azure.