

UNIVERSIDAD PRIVADA DE TACNA



INGENIERIA DE SISTEMAS

TEMA:

Informe Sesión de Laboratorio Nro 01

CURSO:

BASE DE DATOS II

DOCENTE(ING):

Patrick Jose Cuadros Quiroga

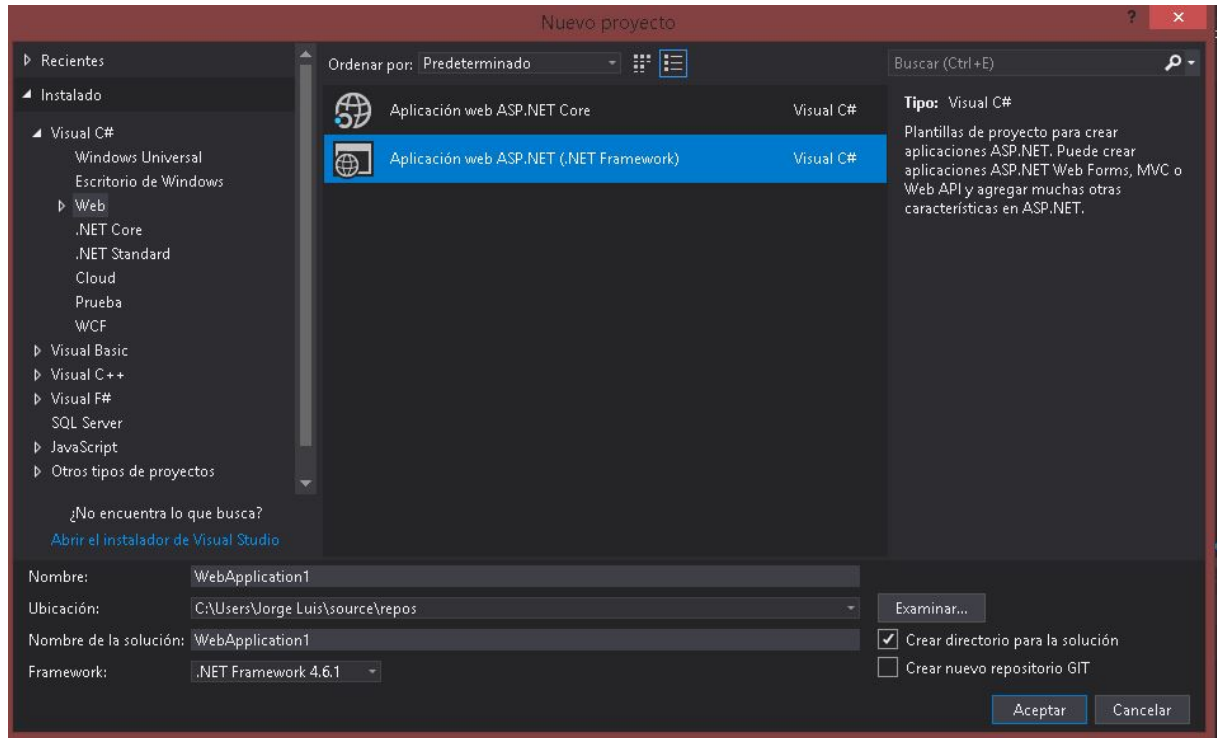
Integrantes:

Marko Antonio RIVAS RIOS	(2016055461)
Jorge Luis MAMANI MAQUERA	(2016055236)
Andree Ludwed VELASCO SUCAPUCA	(2016055286)
Yofer Nain CATARI CABRERA	(2017059289)
Adnner Sleyder ESPERILLA RUIZ	(2015050543)
Jesus ESCALANTE ALANOCA	(2015050641)

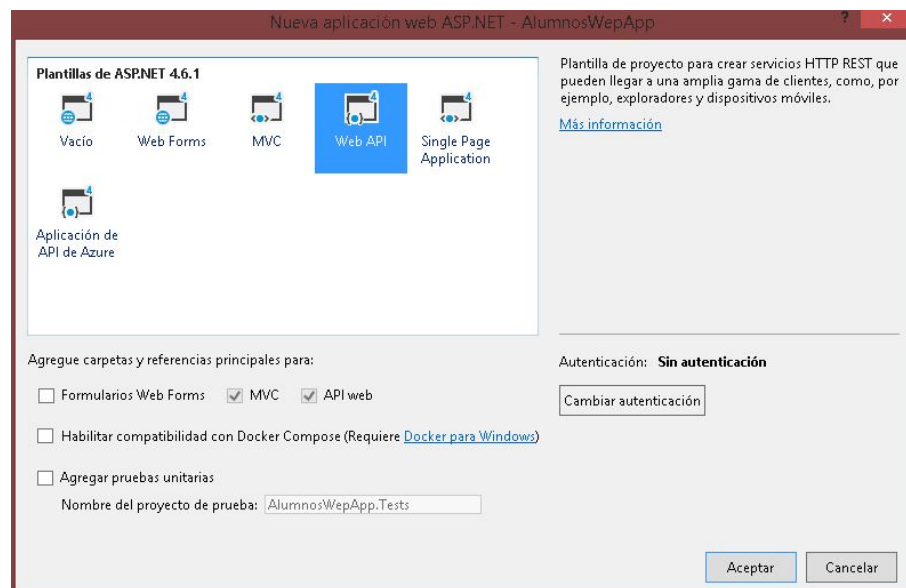
1. Informe Web API

Parte1: Creación del servicio REST Web API

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. Agregar las clases en la carpeta Models. Estas clases representan las tablas que vamos a agregar a nuestra base de datos. Las propiedades representan los campos de la tabla. El código es el siguiente:

```
namespace AlumnosWepApp.Models
{
    0 referencias
    public class Alumno
    {
        0 referencias
        public int Id { get; set; }
        0 referencias
        public string Nombre { get; set; }
        0 referencias
        public string FotoURL { get; set; }
        0 referencias
        public string Usuario { get; set; }
        0 referencias
        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:

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

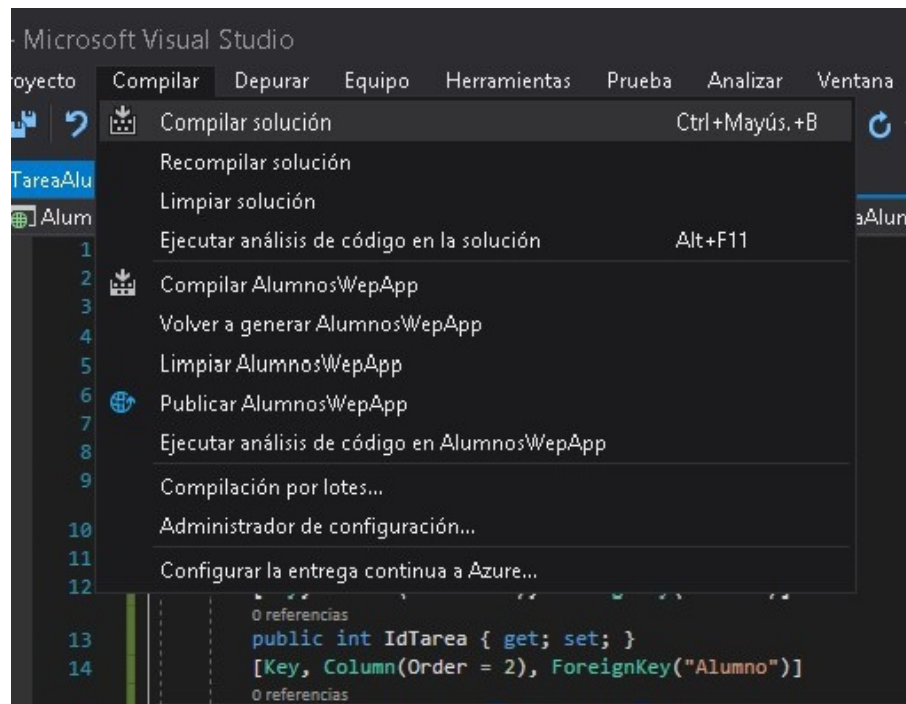
- Paso 5. Como última tabla, en nuestra carpeta Modelos agrega una clase llamada TareaAlumno, la cual representa una relación de N a N entre las tablas anteriores (Alumno y Tarea). Esto lo representaremos con los atributos ForeignKey y Key en las propiedades de ID, así como un elemento virtual para navegación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

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

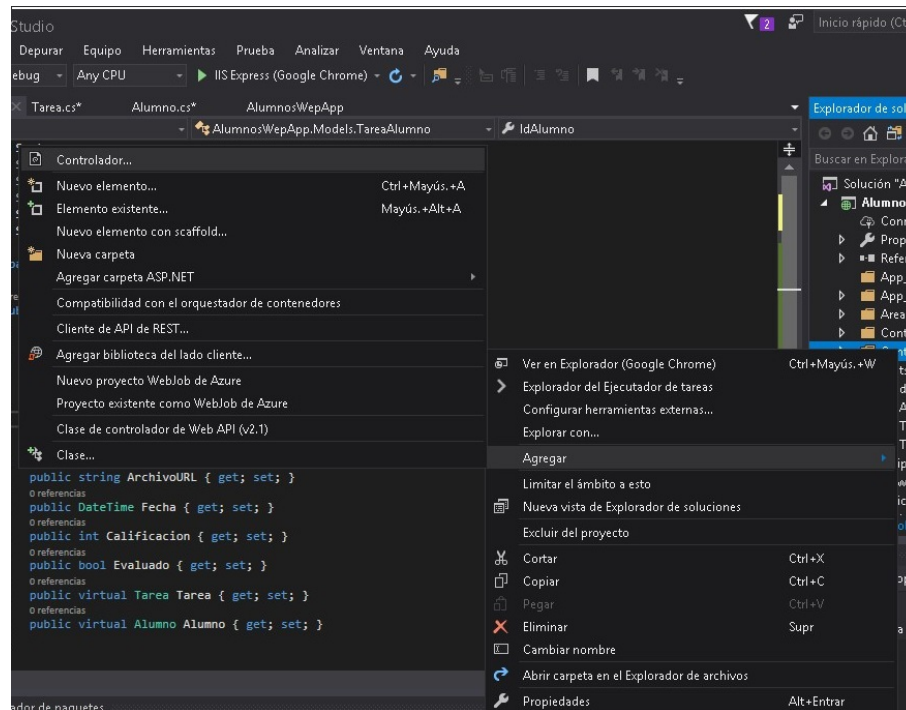
namespace AlumnosWepApp.Models
{
    0 referencias
    public class TareaAlumno
    {
        [Key, Column(Order = 1), ForeignKey("Tarea")]
        0 referencias
        public int IdTarea { get; set; }
        [Key, Column(Order = 2), ForeignKey("Alumno")]
        0 referencias
        public int IdAlumno { get; set; }
        0 referencias
        public string Mensaje { get; set; }
        0 referencias
        public string ArchivoURL { get; set; }
        0 referencias
        public DateTime Fecha { get; set; }
        0 referencias
        public int Calificacion { get; set; }
        0 referencias
        public bool Evaluado { get; set; }
        0 referencias
        public virtual Tarea Tarea { get; set; }
        0 referencias
        public virtual Alumno Alumno { get; set; }
    }
}
```

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

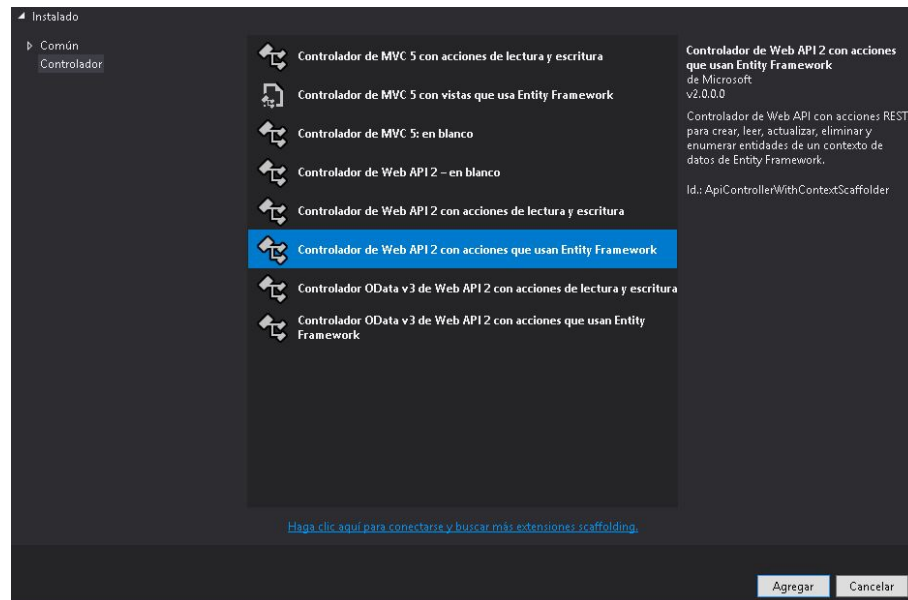
- Paso 6. Después de crear tus modelos, compila la solución.



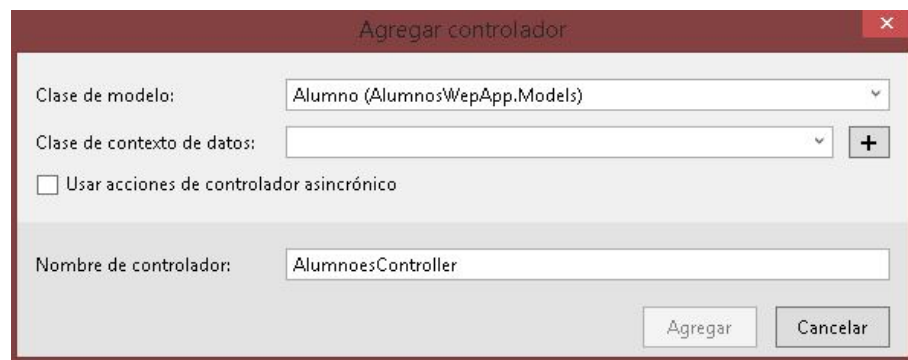
- Paso 7. Ahora da clic derecho en la carpeta Controllers y agrega un nuevo Controlador



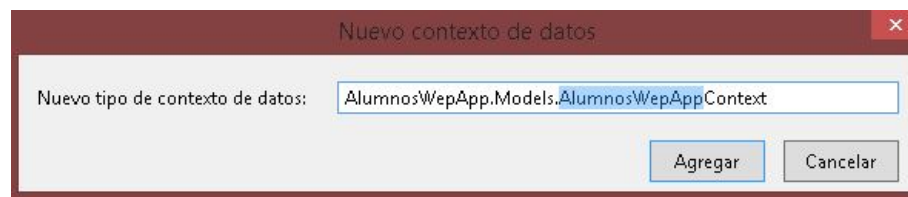
- Paso 8. Selecciona Web API 2 Controller with actions, using EntityFramework en la categoría de Scaffolds



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



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



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

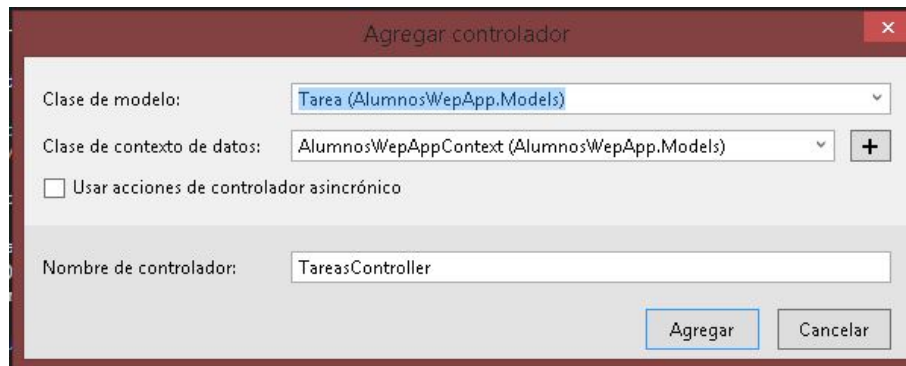


- Paso 12. Modifica o agrega los siguientes métodos dentro del controlador:

```
// 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);
}
```

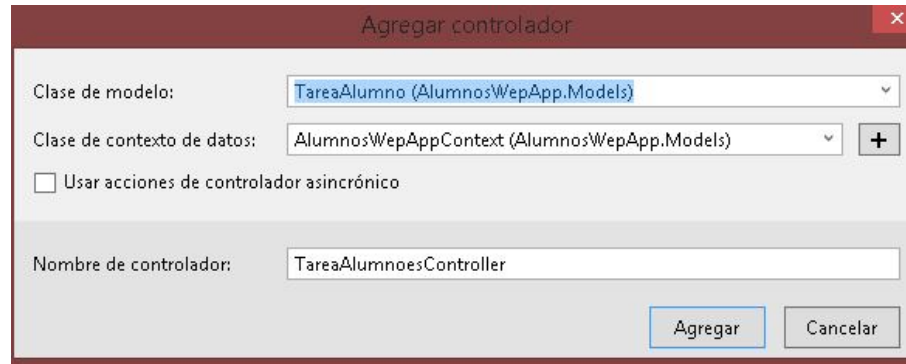
- Paso 13. Agrega otro controlador con la siguiente información:



- 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:



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 IHttpActionResult 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 IHttpActionResult 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 IHttpActionResult 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);
}

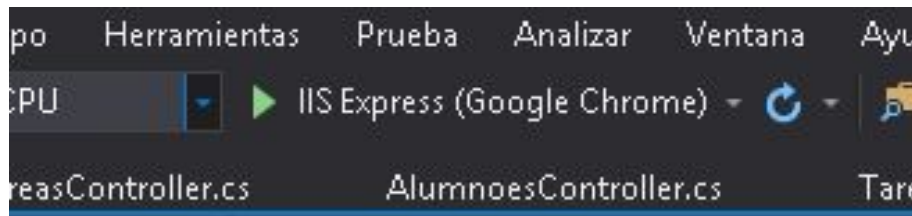
protected override void Dispose(bool disposing)
{
    if (disposing)
        db.Dispose();

    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.

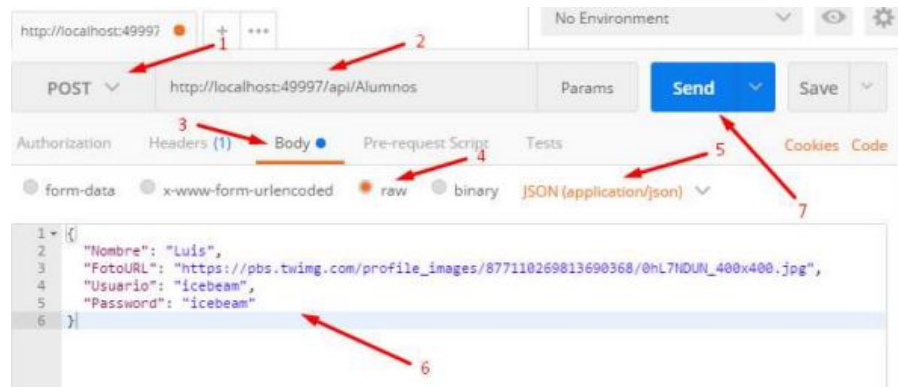
- Paso 20. Observa el formato de la cadena que debe ser enviada y cópiala:

application/json, text/json

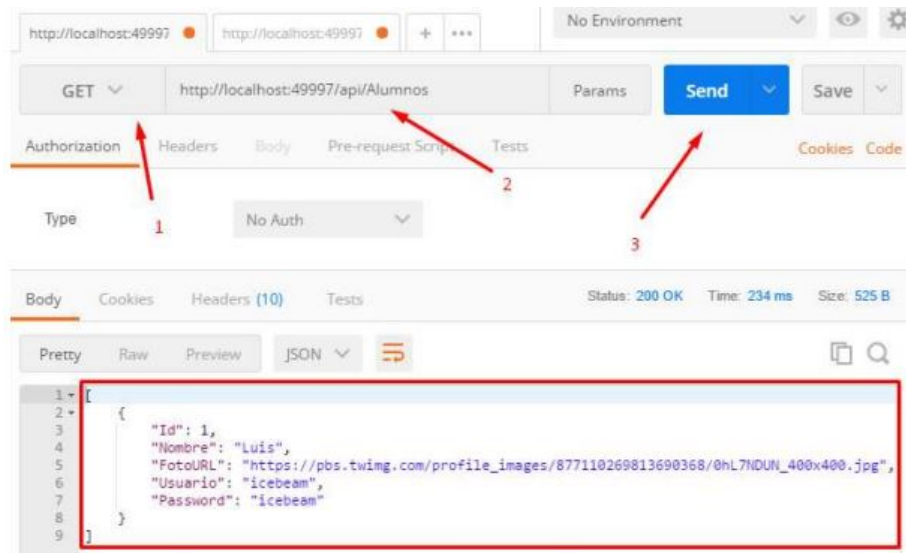
Sample:

```
{
  "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

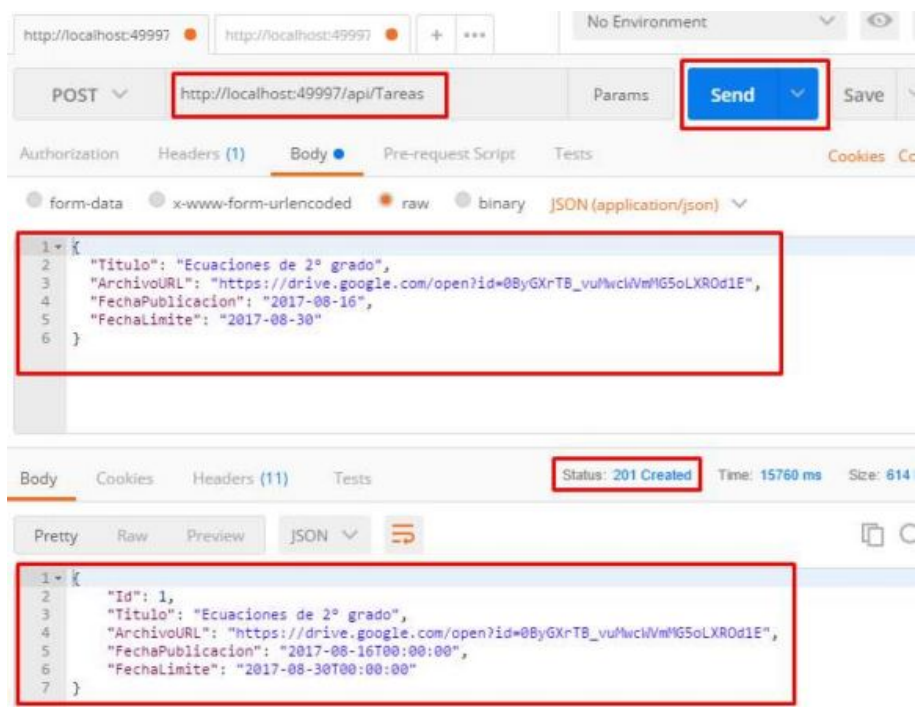
Tareas

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

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



- Paso 25. Observamos el formato de la cadena JSON a introducir:



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

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.

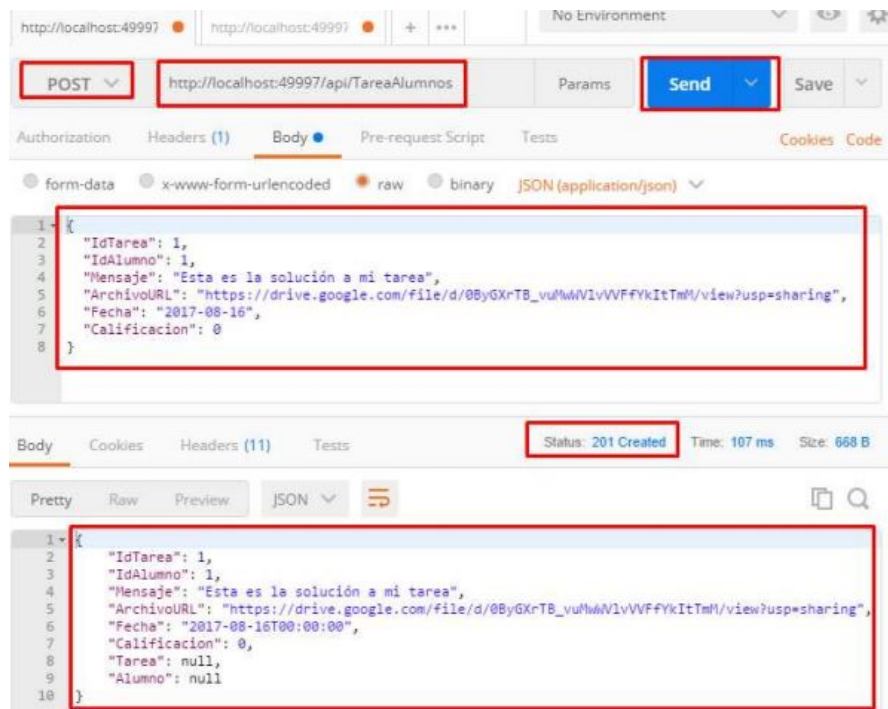
- Paso 27. Verificamos la cadena JSON, centrándonos únicamente en las propiedades que no son de navegación:

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:



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

