

Web Services with Web Api – Part 2/2

Web Api

Web API has been around for some years now. It is a very efficient and lightweight technology to build RESTful web services in .NET. Web API is very similar to .NET MVC with its controllers and routing rules. Therefore, if you are familiar with MVC then it's not too difficult to get going with Web API either.

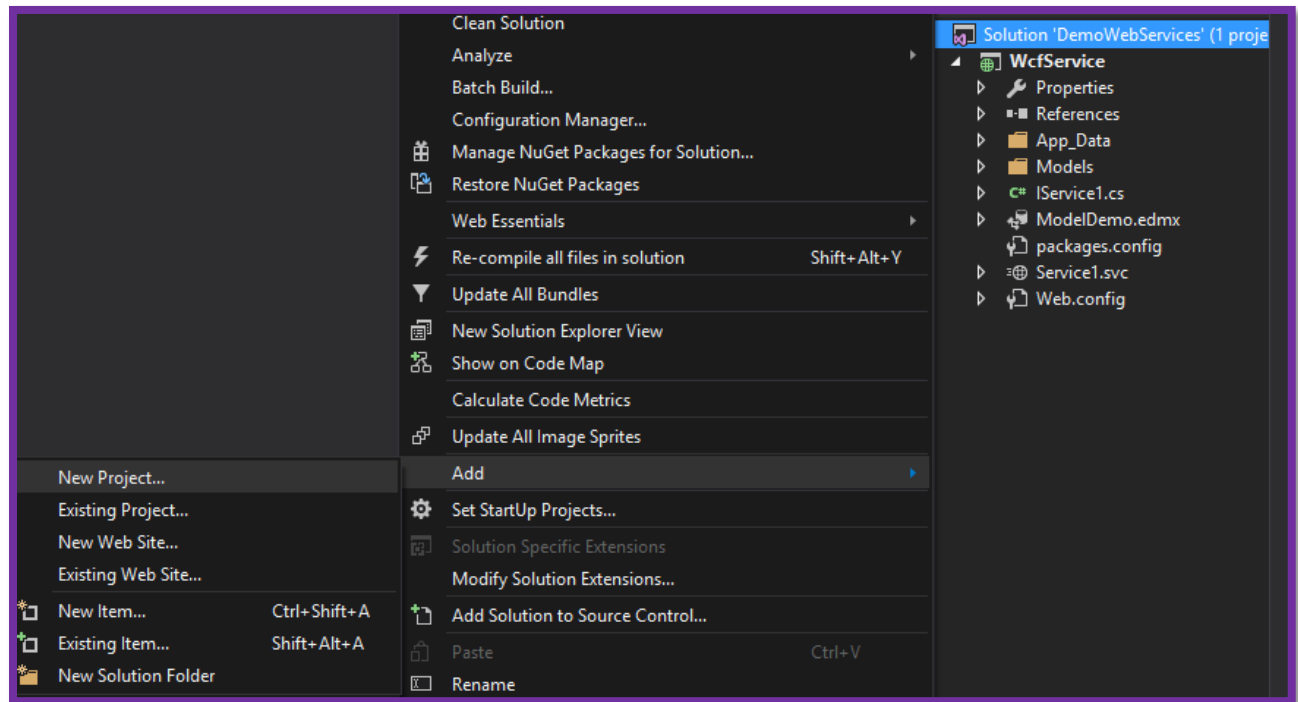
Web API is the great framework for exposing your data and service to different-different devices. Moreover, Web API is open source an ideal platform for building REST-ful services over the .NET Framework. Unlike WCF Rest service, it uses the full features of HTTP (like URIs, request/response headers, caching, versioning, various content formats) and you don't need to define any extra config settings for different devices unlike WCF Rest service. **(dotnet-tricks, 2013)**

Web Api Features

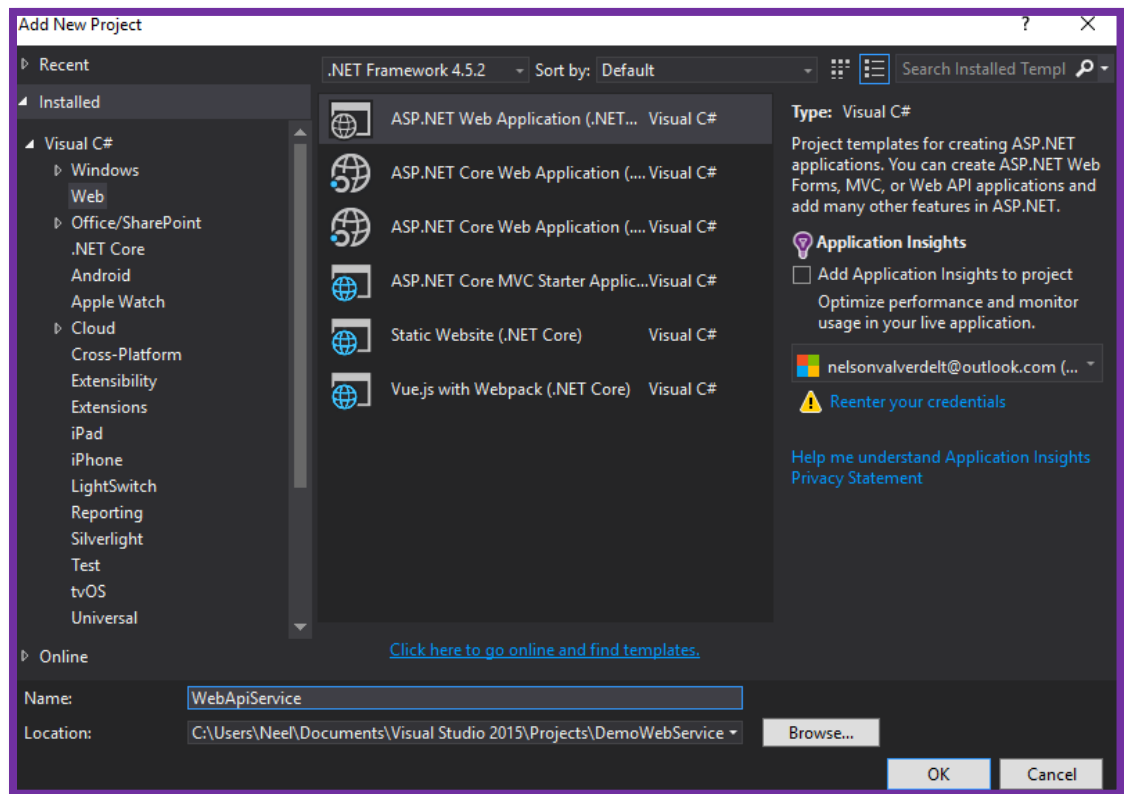
- ✓ It supports convention-based CRUD Actions since it works with HTTP verbs GET, POST, PUT and DELETE.
- ✓ Responses have an Accept header and HTTP status code.
- ✓ Responses are formatted by Web API's MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.
- ✓ It may accept and generates the content which may not be object oriented like images, PDF files etc.
- ✓ It has automatic support for OData. Hence by placing the new [Queryable] attribute on a controller method that returns IQueryable, clients can use the method for OData query composition.
- ✓ It can be hosted with in the applicaion or on IIS.
- ✓ It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection that makes it more simple and robust.

STEP 1 - CREATE A NEW PROJECT

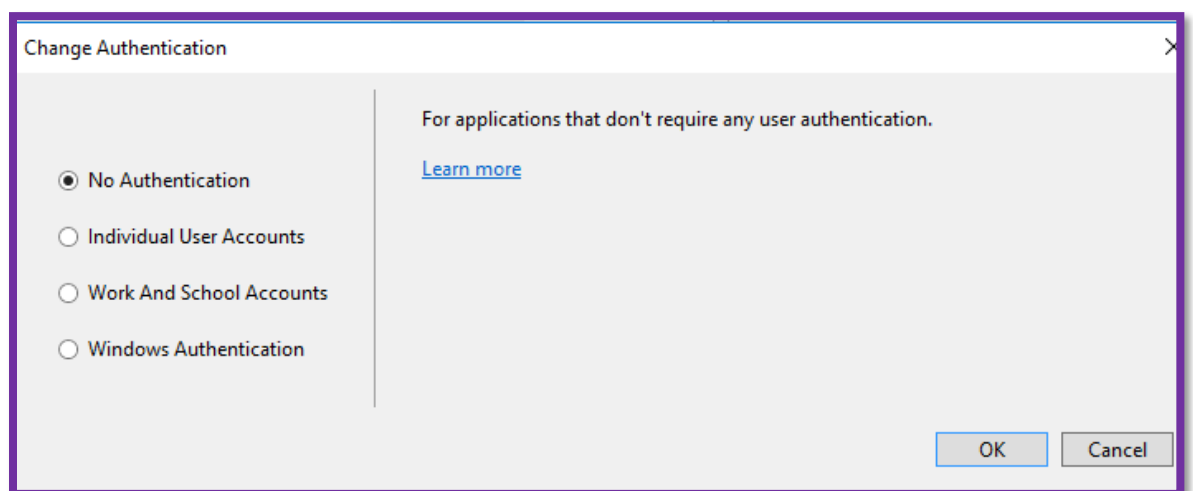
- For this example, we need continue in our project of the same solution "DemoWebServices",



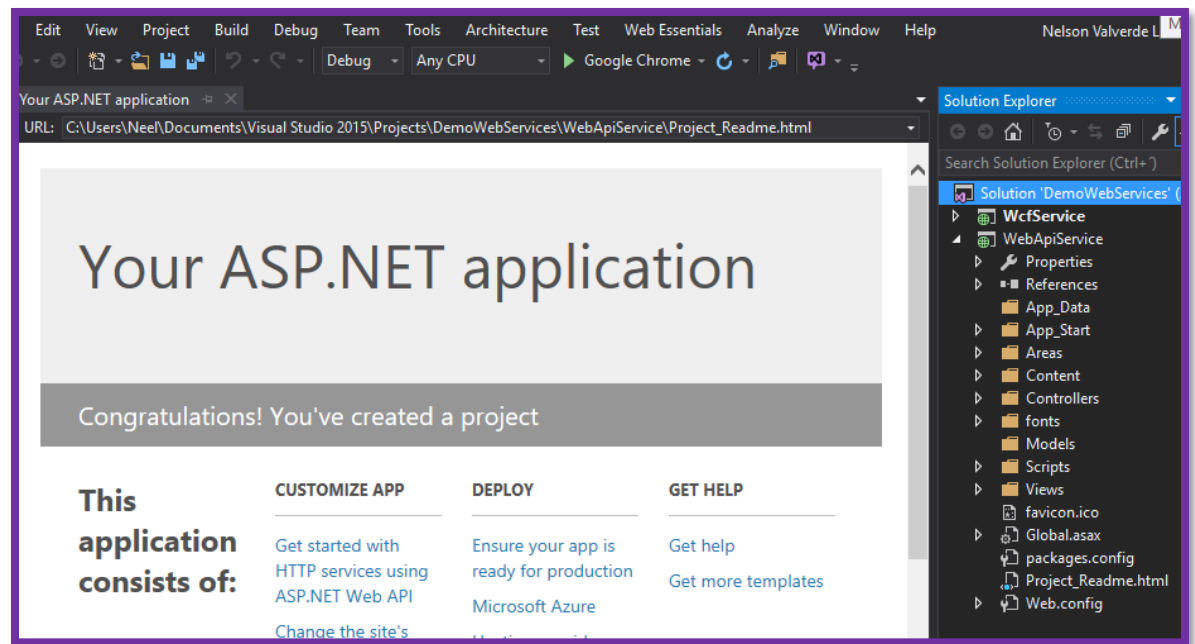
- Then we need add a new project with the name "WebApiService"



- Select **Web Api**, then we select **Change authentication**.

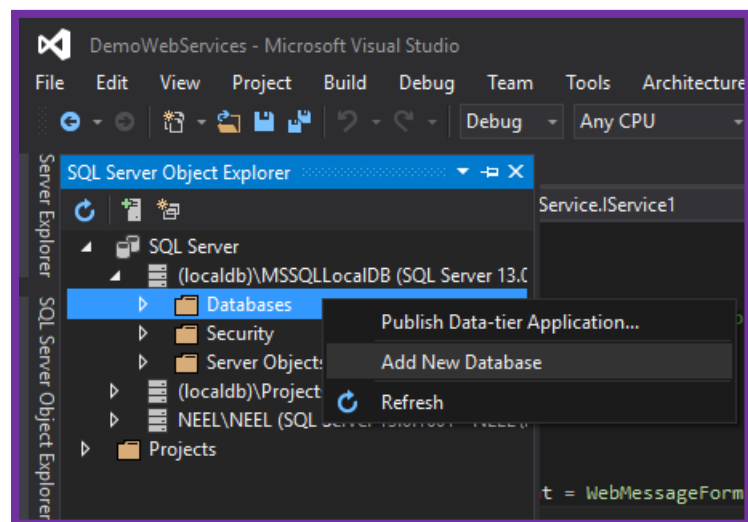


- And we are ready to start our project with Web Api.

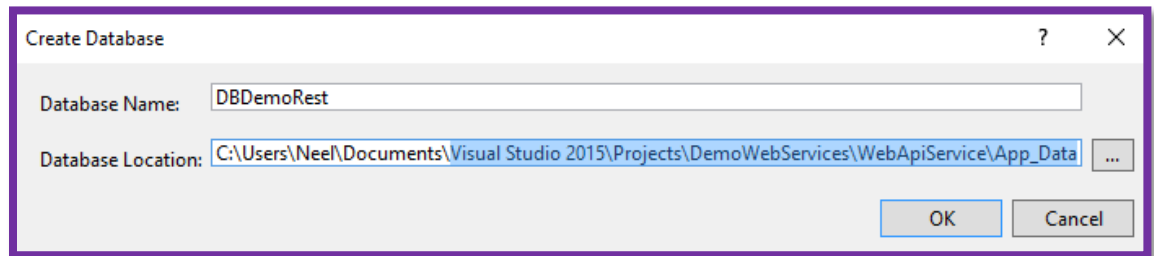


STEP 2 – CREATE OUR DATA BASE LOCAL

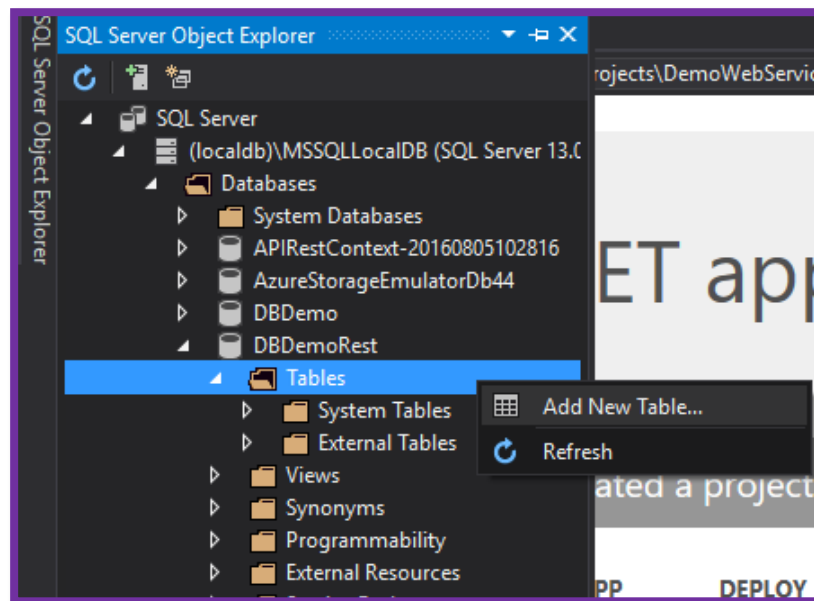
- Select SQL Server Object Explorer and create your local Database.



- Select your Database location, this having objective create a file local database with extension ".mdf", usually I select my file **App_Data** of project (**recommend**).



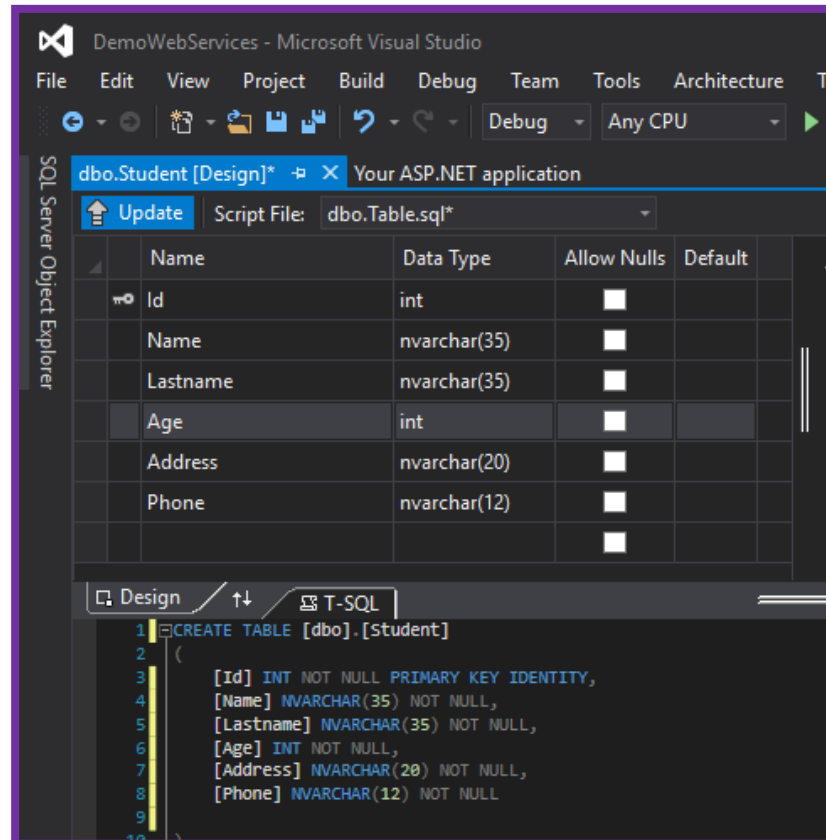
- Select our table file and add new table



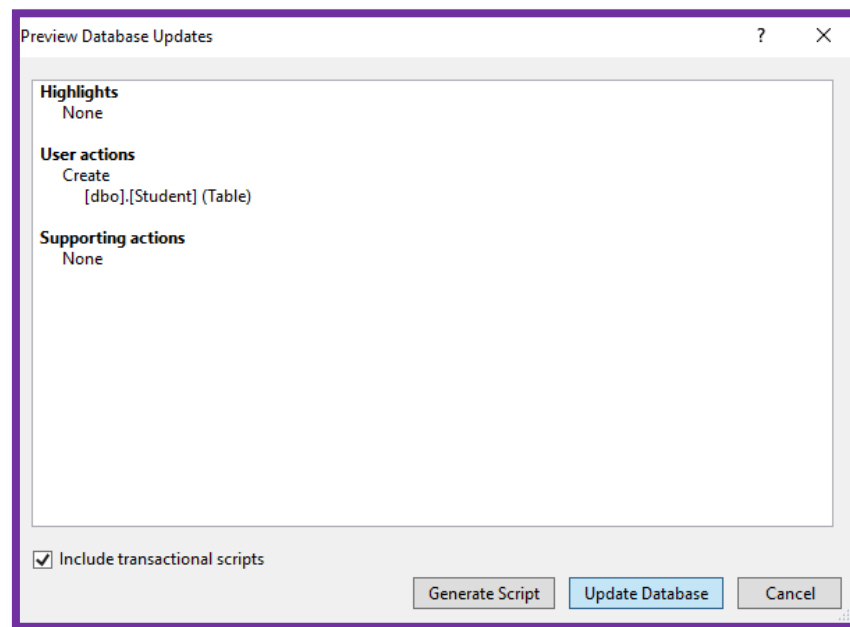
- Add a table "Employees" for this example, Add our attributes:

[Id, Name, Lastname, Address, Phone] and "update" database.

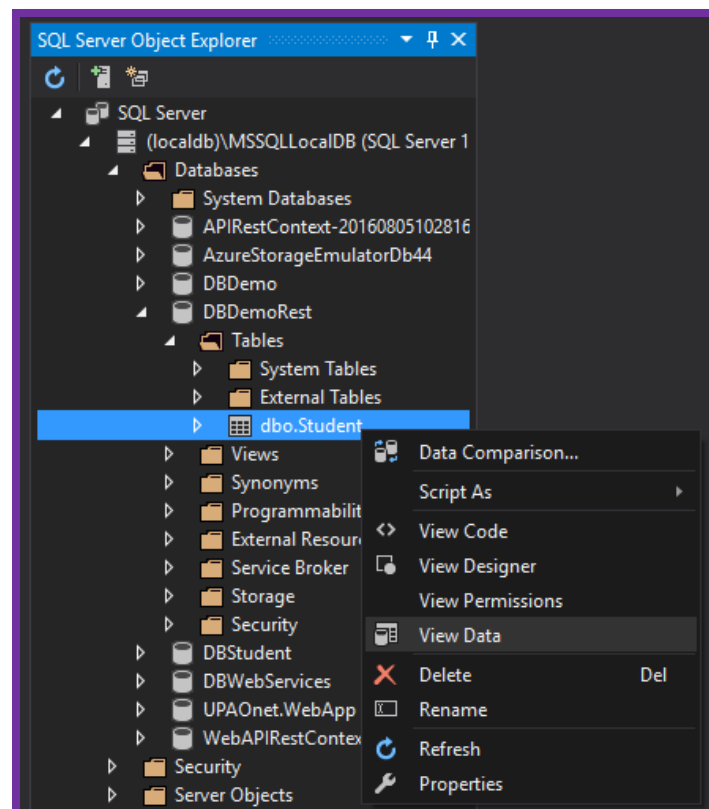
And last but not least select **Update**



- This is a preview of the new update our database and finish with **Update Database**



- Now we need add data in our table of database



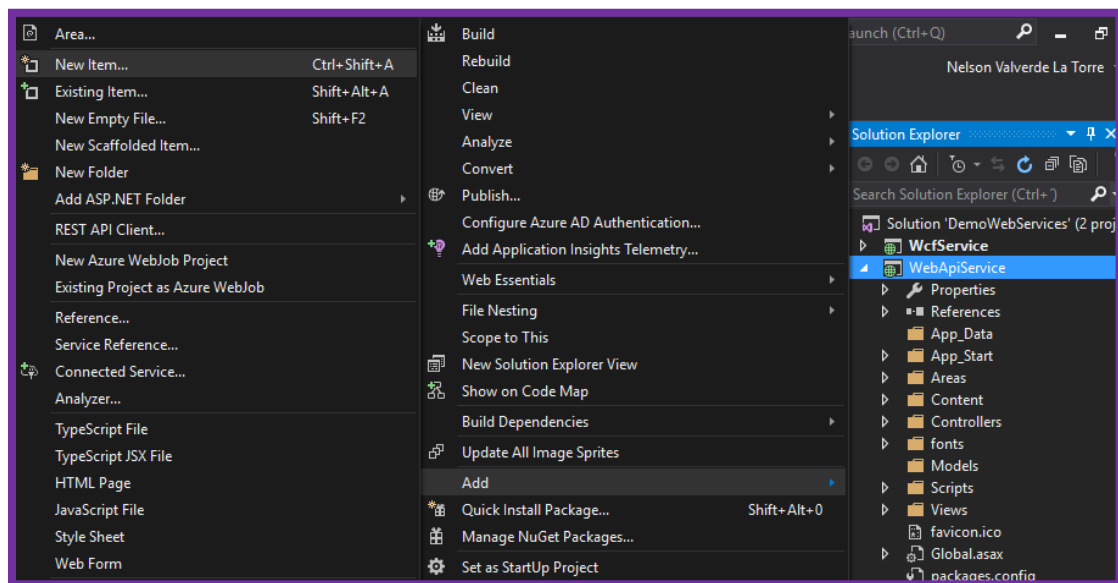
- Complete our table with data

Visual Studio interface showing the 'dbo.Student [Data]' table. The table has 6 columns: Id, Name, Lastname, Age, Address, and Phone. The data is as follows:

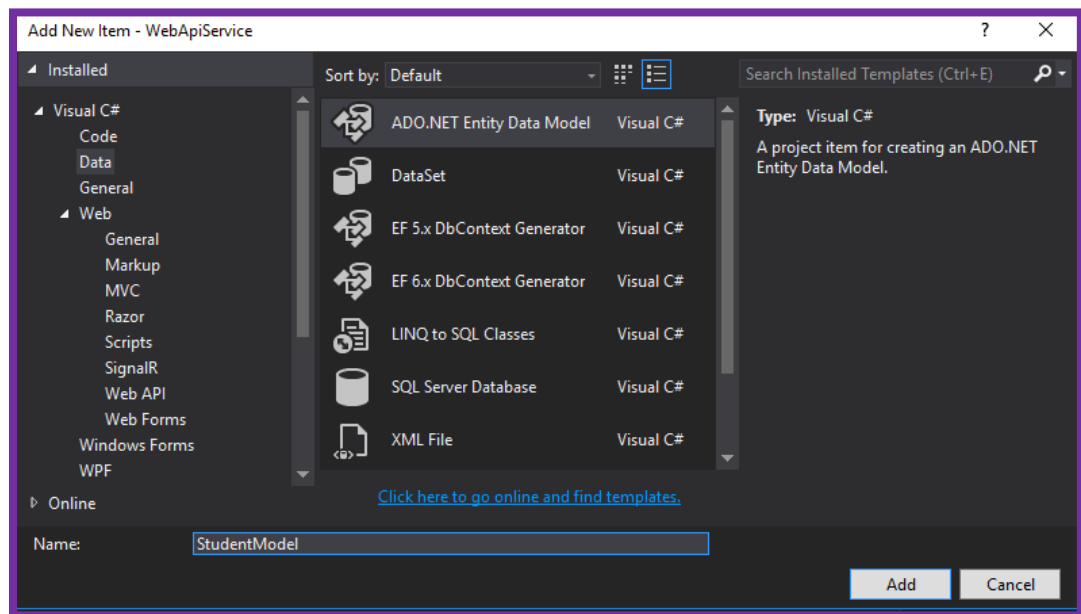
	Id	Name	Lastname	Age	Address	Phone
	1	Nelson	Valverde	24	Av.Pedro Garci...	45-57-987
	2	Yordi	Agustin	18	Av.Pumacahua ...	12-12-12
	3	Javier	Escobar	19	Vallejos	45-54-54
	4	Fredy	Guibert	26	Av.Piedas Gordas	45-57-78
	NULL	NULL	NULL	NULL	NULL	NULL

STEP 3 – CREATE OUR DATA MODEL WITH ENTITY FRAMEWORK

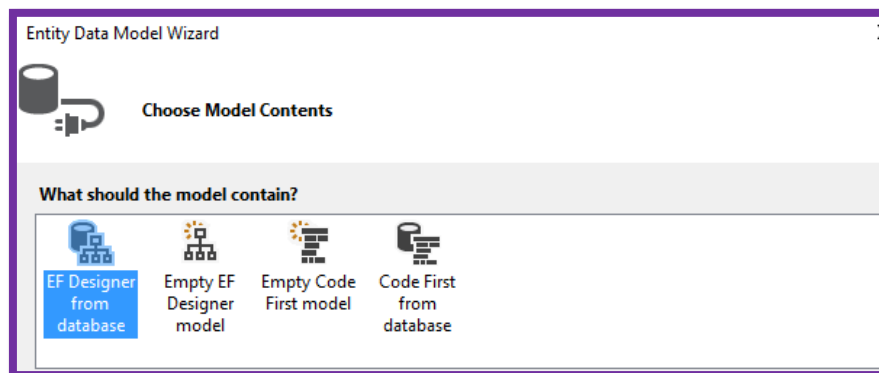
- We create a new Item



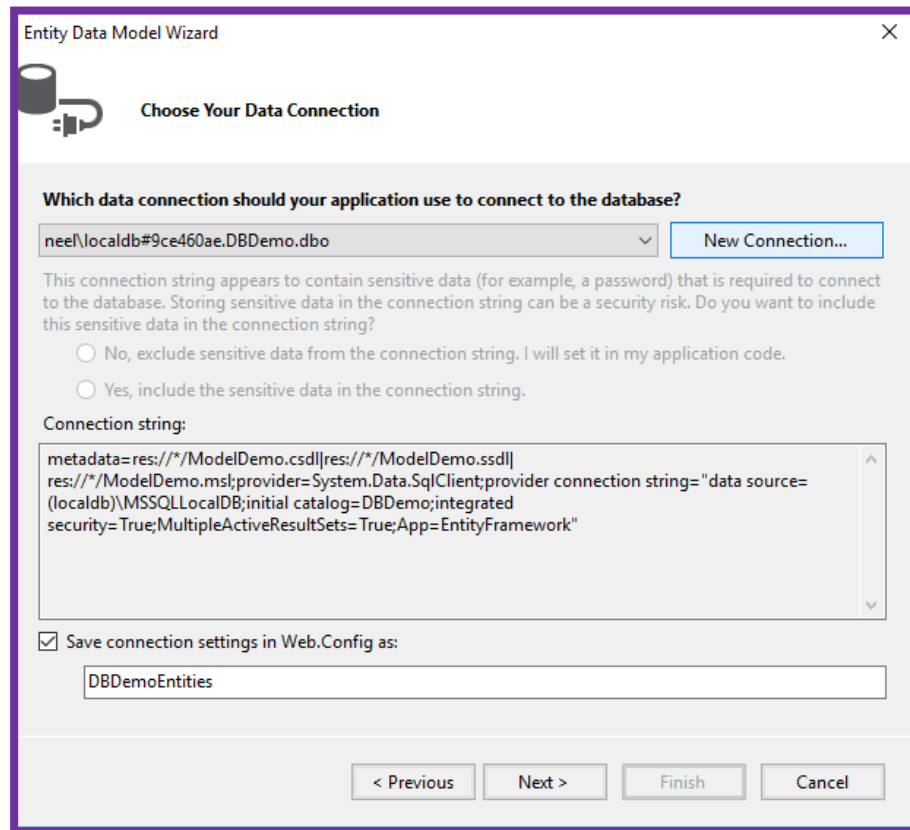
- Select ADO.NET Entity Data Model and add the name **StudentModel**, ADO.NET its located in **Visual C# / Data**,



- Select **EF Designer From Database** and Next



- ✓ Select **New Connection**, this allows generate a new connection with our database local



The image shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Data Connection' step. The title bar reads 'Entity Data Model Wizard' with a close button. Below the title bar is a database icon and the text 'Choose Your Data Connection'. The main area asks 'Which data connection should your application use to connect to the database?'. A dropdown menu shows 'neel\localdb#9ce460ae.DBDemo.dbo' and a 'New Connection...' button is to its right. Below this, a warning message states: 'This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?'. There are two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' and 'Yes, include the sensitive data in the connection string.'. Below the radio buttons is a section labeled 'Connection string:' with a text area containing the following text: 'metadata=res://*/ModelDemo.csdl|res://*/ModelDemo.ssdl|res://*/ModelDemo.msl;provider=System.Data.SqlClient;provider connection string="data source=(localdb)\MSSQLLocalDB;initial catalog=DBDemo;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"'. At the bottom, there is a checkbox 'Save connection settings in Web.Config as:' which is checked, and a text box containing 'DBDemoEntities'. At the very bottom are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

neel\localdb#9ce460ae.DBDemo.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

metadata=res://*/ModelDemo.csdl|res://*/ModelDemo.ssdl|res://*/ModelDemo.msl;provider=System.Data.SqlClient;provider connection string="data source=(localdb)\MSSQLLocalDB;initial catalog=DBDemo;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"

☒ Save connection settings in Web.Config as:

DBDemoEntities

< Previous Next > Finish Cancel

- Now, we need to make some settings in connection properties

Connection Properties

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:
(localdb)\MSSQLLocalDB Refresh

Log on to the server

Authentication: Windows Authentication

User name:

Password:

☐ Save my password

Connect to a database

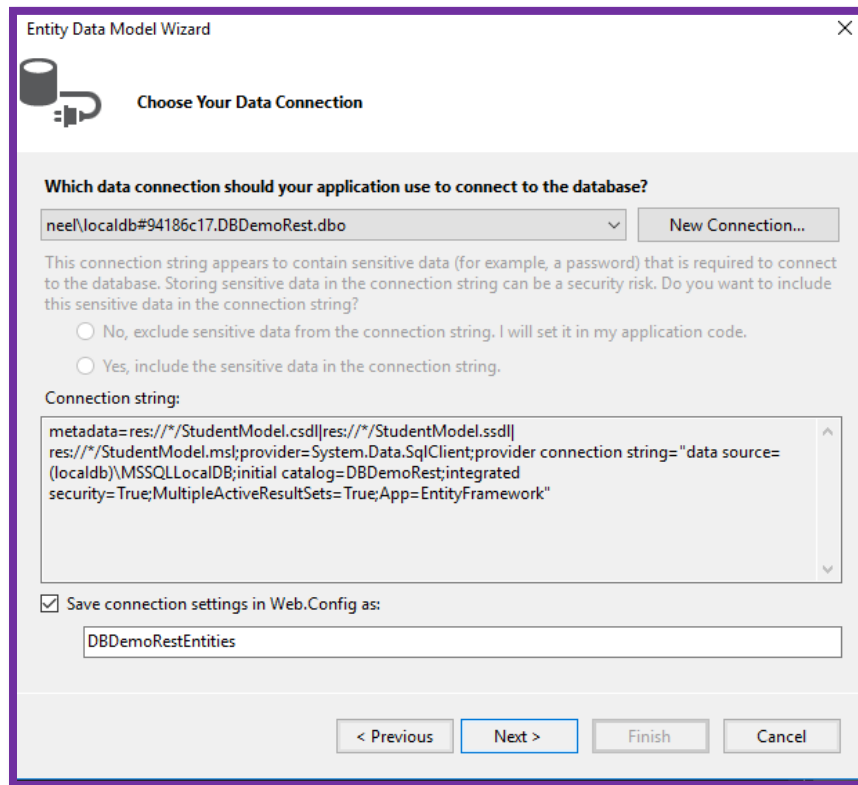
☒ Select or enter a database name:
DBDemoRest

☐ Attach a database file:
 Browse...
Logical name:

Advanced...

Test Connection OK Cancel

- Click Next →



The screenshot shows the 'Entity Data Model Wizard' window with the title 'Choose Your Data Connection'. It features a database icon and a question: 'Which data connection should your application use to connect to the database?'. A dropdown menu shows 'neel\localdb#94186c17.DBDemoRest.dbo' with a 'New Connection...' button. Below this is a warning about sensitive data in the connection string and two radio buttons: 'No, exclude sensitive data from the connection string. I will set it in my application code.' (selected) and 'Yes, include the sensitive data in the connection string.'. A 'Connection string:' label is followed by a text area containing a complex string. At the bottom, there is a checkbox 'Save connection settings in Web.Config as:' which is checked, with a text box containing 'DBDemoRestEntities'. Navigation buttons at the bottom are '< Previous', 'Next >' (highlighted), 'Finish', and 'Cancel'.

Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

neel\localdb#94186c17.DBDemoRest.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

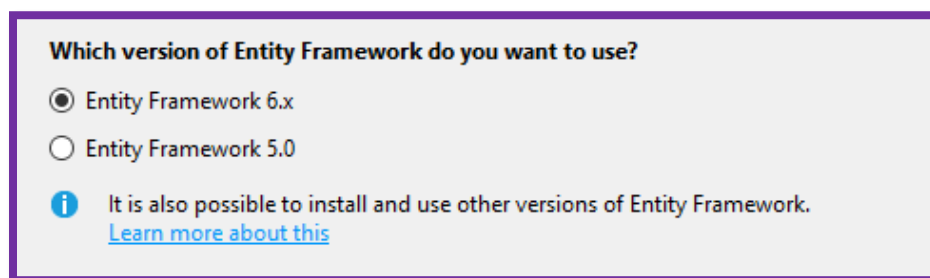
metadata=res://*/StudentModel.csdl|res://*/StudentModel.ssdl|
res://*/StudentModel.msl;provider=System.Data.SqlClient;provider connection string="data source=
(localdb)\MSSQLLocalDB;initial catalog=DBDemoRest;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"

☒ Save connection settings in Web.Config as:

DBDemoRestEntities

< Previous Next > Finish Cancel

- Select Entity Framework 6.x and Next



The screenshot shows a dialog box titled 'Which version of Entity Framework do you want to use?'. It has two radio buttons: 'Entity Framework 6.x' (selected) and 'Entity Framework 5.0'. Below the buttons is an information icon and text: 'It is also possible to install and use other versions of Entity Framework. [Learn more about this](#)'.

Which version of Entity Framework do you want to use?

☒ Entity Framework 6.x

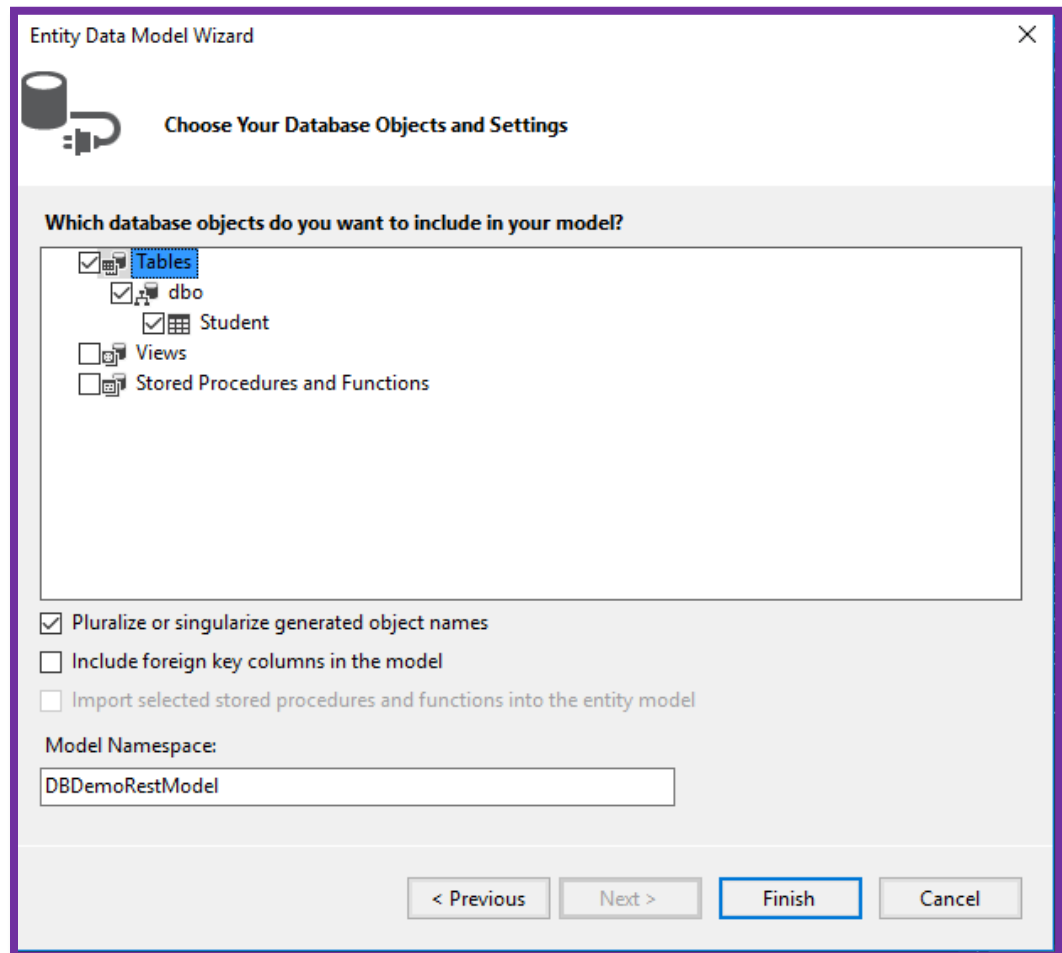
☐ Entity Framework 5.0

i It is also possible to install and use other versions of Entity Framework.
[Learn more about this](#)

- Select our table

Select the table for the mapping into our file .edmx and assign a name.

An **.edmx** file contains the the conceptual model, as well as a storage model and the mappings between them.



The image shows the 'Entity Data Model Wizard' dialog box, specifically the 'Choose Your Database Objects and Settings' step. The dialog has a title bar with 'Entity Data Model Wizard' and a close button. Below the title bar is a database icon and the text 'Choose Your Database Objects and Settings'. The main area is titled 'Which database objects do you want to include in your model?'. It contains a tree view with the following items: 'Tables' (checked), 'dbo' (checked), 'Student' (checked), 'Views' (unchecked), and 'Stored Procedures and Functions' (unchecked). Below the tree view are three checkboxes: 'Pluralize or singularize generated object names' (checked), 'Include foreign key columns in the model' (unchecked), and 'Import selected stored procedures and functions into the entity model' (unchecked). At the bottom, there is a 'Model Namespace:' label and a text box containing 'DBDemoRestModel'. The bottom right corner has four buttons: '< Previous', 'Next >', 'Finish' (highlighted with a blue border), and 'Cancel'.

Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

- ☒ Tables
 - ☒ dbo
 - ☒ Student
- ☐ Views
- ☐ Stored Procedures and Functions

☒ Pluralize or singularize generated object names

☐ Include foreign key columns in the model

☐ Import selected stored procedures and functions into the entity model

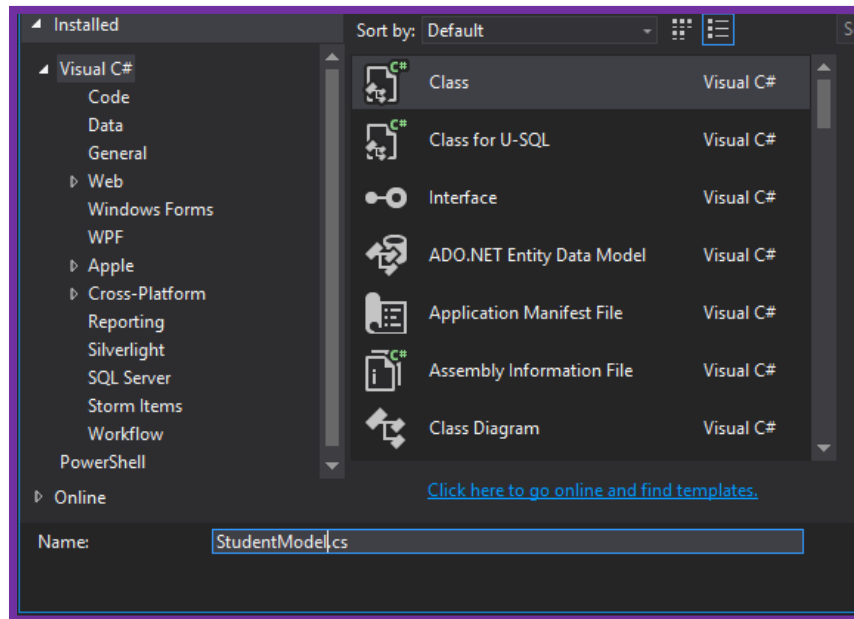
Model Namespace:

DBDemoRestModel

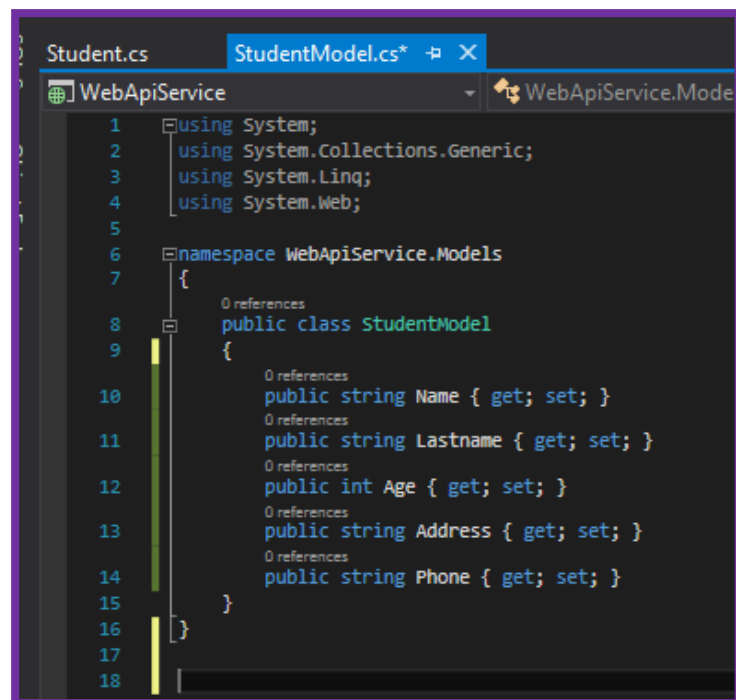
< Previous Next > **Finish** Cancel

STEP 4 – CREATE OUR MODEL

- Create our model in the folder **“Models”** and assign a name **“StudentModel”**

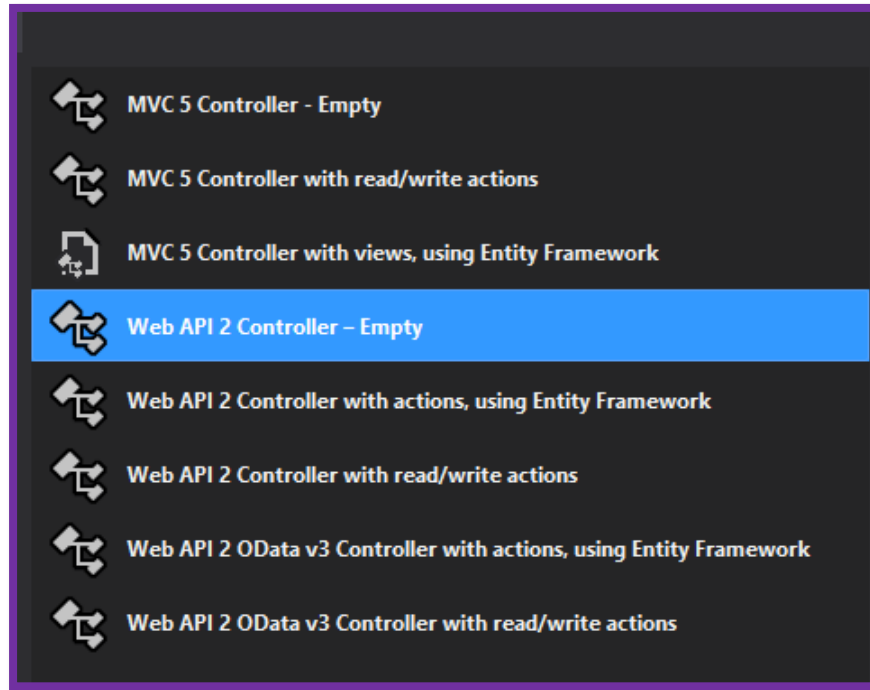


- Add our attributes or the data we need

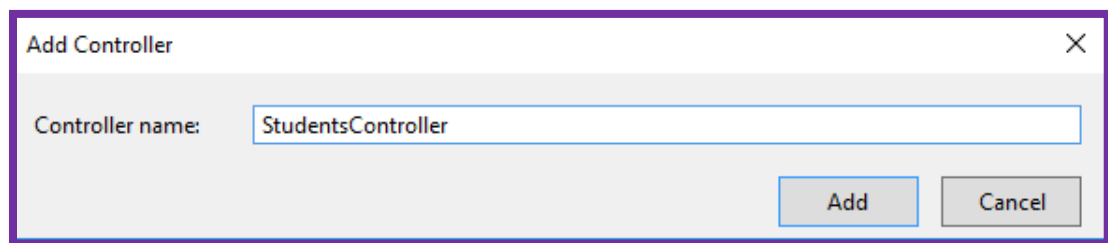


STEP 5 – CREATE OUR CONTROLLER WITH API CONTROLLER

- Create our Controller in the folder “**Controllers**”



- Assign a Name



- Add **"Using"** for this controller

```
// New usings
using System.Threading.Tasks;
using WebApiService.Models;
using System.Web.Script.Serialization;
```

- Add our instance, this allow connect a database and create a method **GetStudents**

```
namespace WebApiService.Controllers
{
    0 references
    public class StudentsController : ApiController
    {
        //Instantance Database
        DBDemoRestEntities db = new DBDemoRestEntities();

        //GET --> api/Students
        0 references
        public List<StudentModel> GetStudents()
        {
            var query = from s in db.Students
                        select new StudentModel()
                        {
                            Name = s.Name,
                            Lastname = s.Lastname,
                            Age = s.Age,
                            Address = s.Address,
                            Phone = s.Phone
                        };

            return query.ToList();
        }
    }
}
```


- Create a method **GetStudents** with a parameter id of type integer

```
//GET --> api/Students/{id}
0 references
public StudentModel GetStudents(int id)
{
    var query = from s in db.Students
                where s.Id == id
                select new StudentModel()
                {
                    Name = s.Name,
                    Lastname = s.Lastname,
                    Age = s.Age,
                    Address = s.Address,
                    Phone = s.Phone
                };

    return query.FirstOrDefault();
}
```

- Create a method PostStudents, this need a parameter **HttpRequestMessage**, this allow obtain a request JSON

```
//POST --> api/Students
0 references
public async Task<bool> PostStudents(HttpRequestMessage request)
{
    // The ModelState represents a collection of name and value pairs that were submitted
    // to the server during a POST
    if (ModelState.IsValid)
    {
        //Read pur Json
        var jsonString = await request.Content.ReadAsStringAsync();

        //Instance Serializer
        var jss = new JavaScriptSerializer();

        //Deserialize our json
        StudentModel student = jss.Deserialize<StudentModel>(jsonString);

        //save the deserialized data in our Model generated with Entity Framework
        var _student = new Student()
        {
            Name = student.Name,
            Lastname = student.Lastname,
            Age = student.Age,
            Address = student.Address,
            Phone = student.Phone
        };

        //Add a new element
        db.Students.Add(_student);
        //Save changes of our table
        db.SaveChanges();

        return true;
    }
    else
    {
        return false;
    }
}
```

- Create a method **PutStudents** and need two parameters

Id: Parameter in url.

HttpRequestMessage; this allow obtain a request JSON

```
//PUT --> api/Students/{id}
0 references
public async Task<string> PutStudents(int id, HttpRequestMessage request)
{
    //Read pur Json
    var jsonString = await request.Content.ReadAsStringAsync();

    //Instance Serializer
    var jss = new JavaScriptSerializer();

    //Deserialize our json
    StudentModel student = jss.Deserialize<StudentModel>(jsonString);

    var query = from st in db.Students
                where st.Id == id
                select st;

    //Iterate and assign a new data for our table
    foreach (Student item in query)
    {
        item.Name = student.Name;
        item.Lastname = student.Lastname;
        item.Age = student.Age;
        item.Address = student.Address;
        item.Phone = student.Phone;
    };

    try
    {
        //Save changes
        db.SaveChanges();
        return "OK";
    }catch(Exception ex)
    {
        return ex.Message;
    }
}
```

- Create a method **DeleteStudents** with a parameter **id**

```
//Delete --> api/GetStudents/{id}
0 references
public string DeleteStudents(int id)
{
    try
    {
        var query = from st in db.Students
                     where st.Id == id
                     select st;

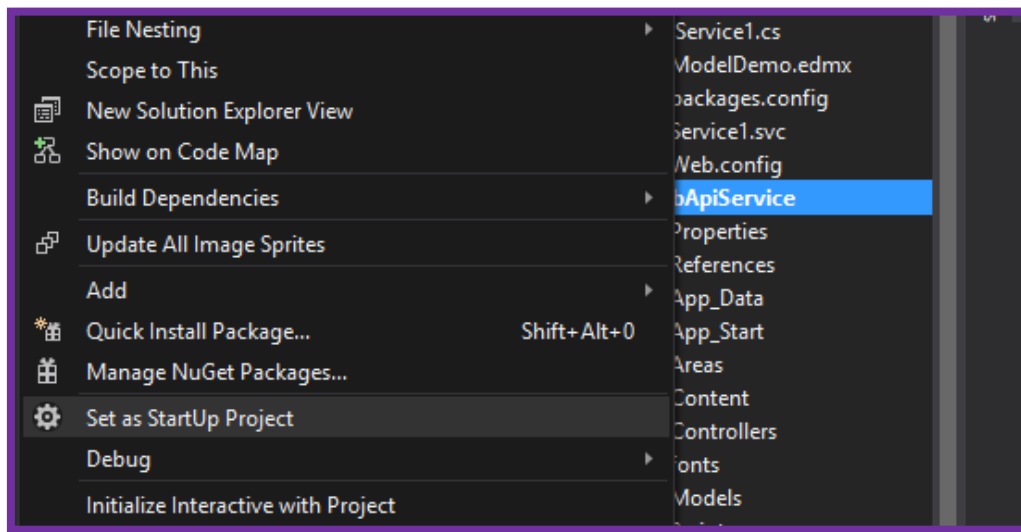
        foreach (var item in query)
        {
            db.Students.Remove(item);
        }

        db.SaveChanges();

        return "OK";
    } catch (Exception ex)
    {
        return ex.Message;
    }
}
}
```

STEP 6 – WE PREPARE OUR PROJECT

- 1) First we need select your project and set as StartUp Project, then we need have open our application web services or F5,



Look this example: we have this direction url:

<http://localhost:12102/api/Students/1>

http://localhost → This is la direction local of project

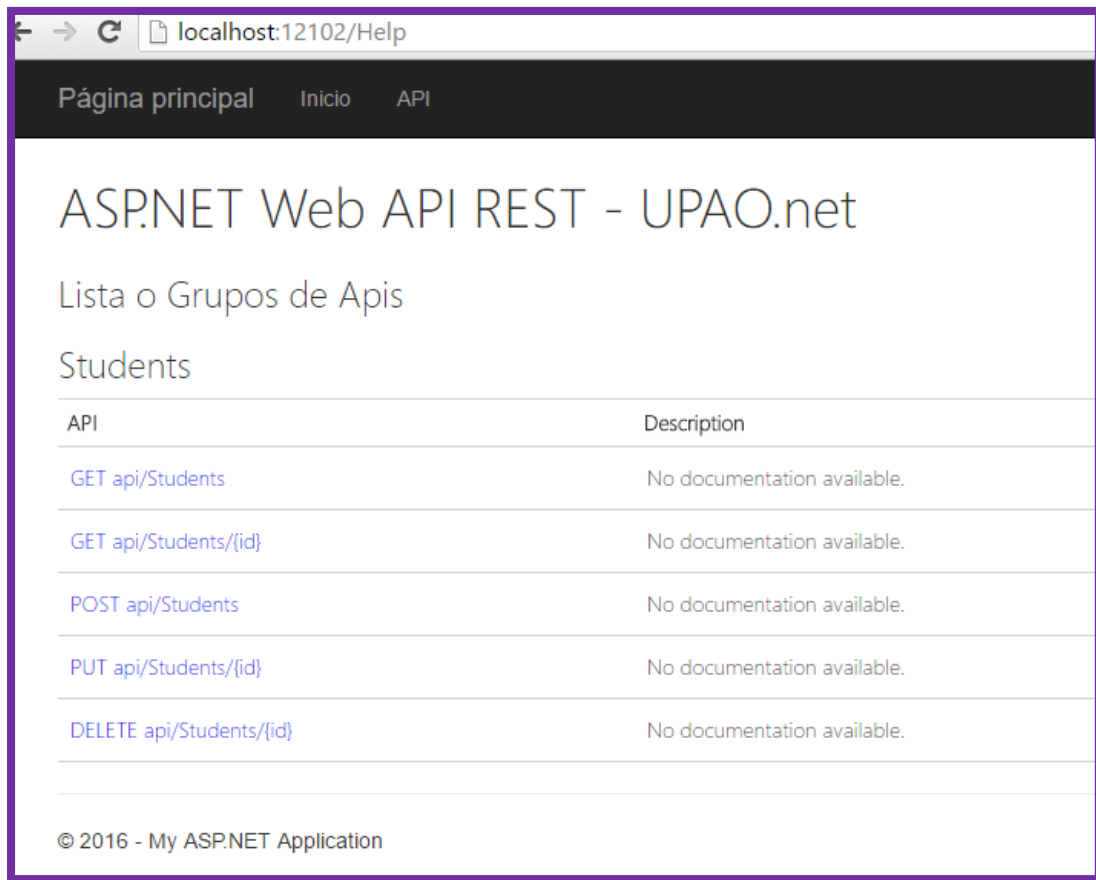
12102 → This is port local, this varies for each project

Api → this segment of our direction is your controller

Students → This is our method of the controller

1 → This is simply the parameter we need for obtain information

2) Then of this example, open our browser, add **help** the end of the direction URL, look this



This example we have 5 Apis:

1. GET → api/Students

It is equivalent to → <http://localhost:12102/api/Students>

2. GET → api/Students/{id}

It is equivalent to → <http://localhost:12102/api/Students/{id}>

3. POST → api/Students

It is equivalent to → <http://localhost:12102/api/Students>

4. PUT → api/Students/{id}

It is equivalent to → <http://localhost:12102/api/Students/{id}>

5. DELETE → api/Students/{id}

It is equivalent to → <http://localhost:12102/api/Students/{id}>

Important:

GET: Obtain or retrieve something

POST: Create a resource.

PUT: Update or Modify a resource

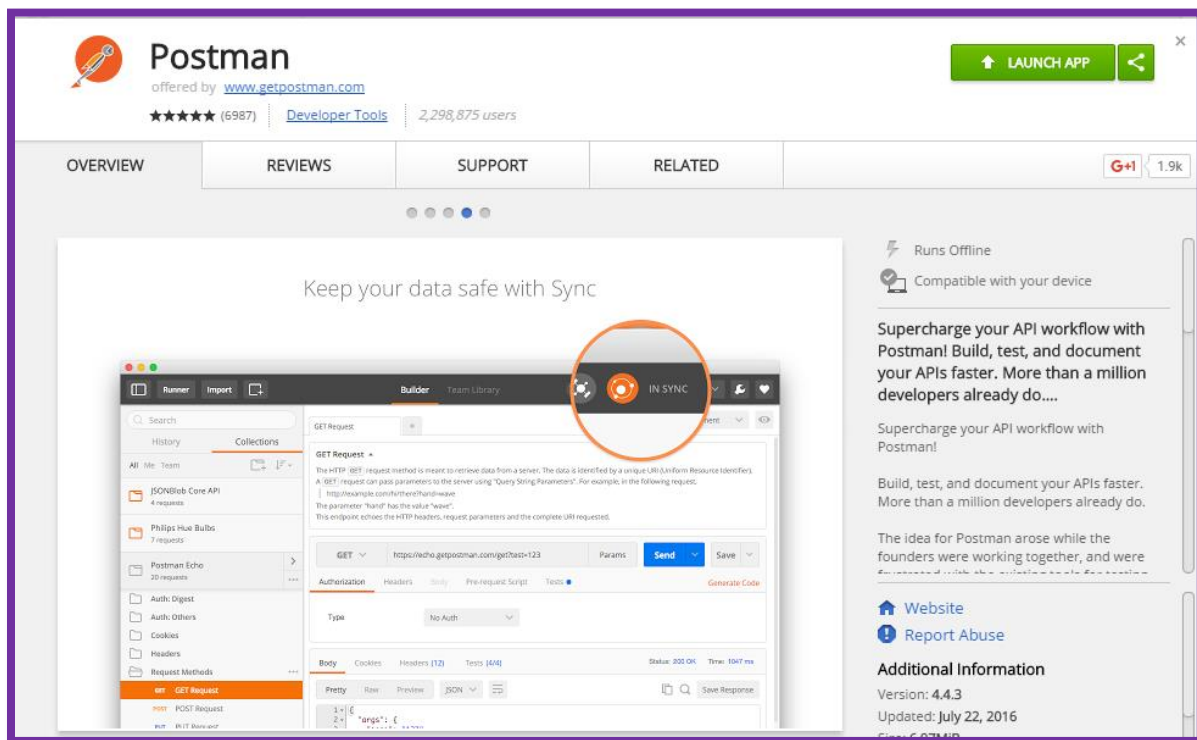
DELETE: Delete a resource

WE PREPARE OUR TOOL

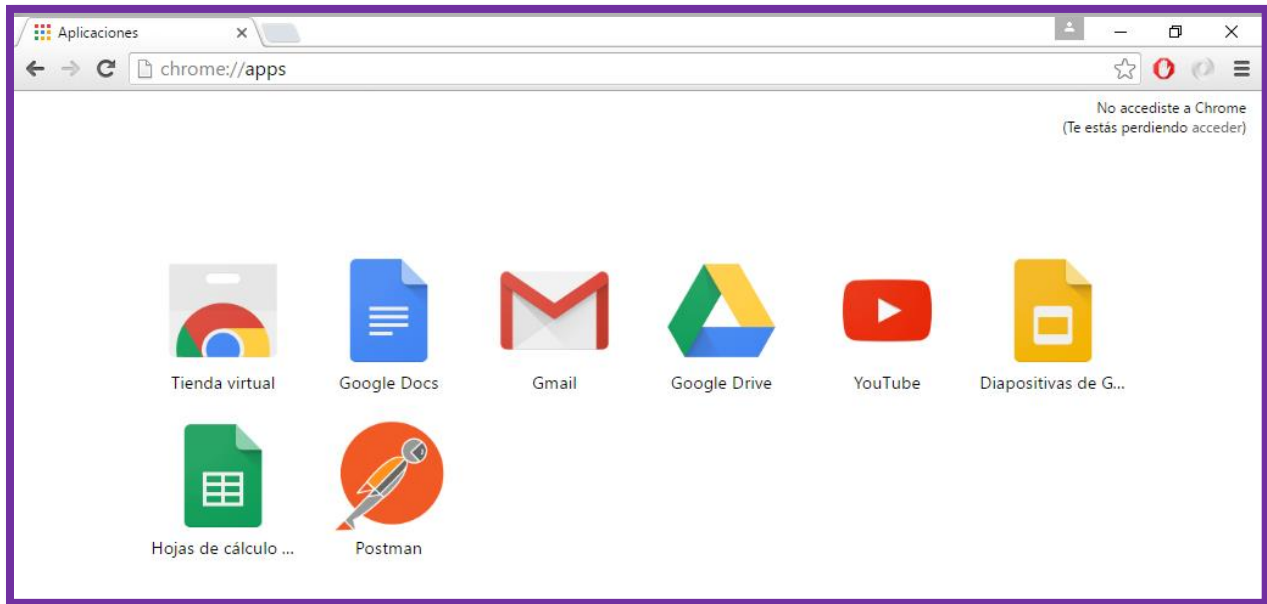
1) Now we need Add an extension in Chrome

Link:

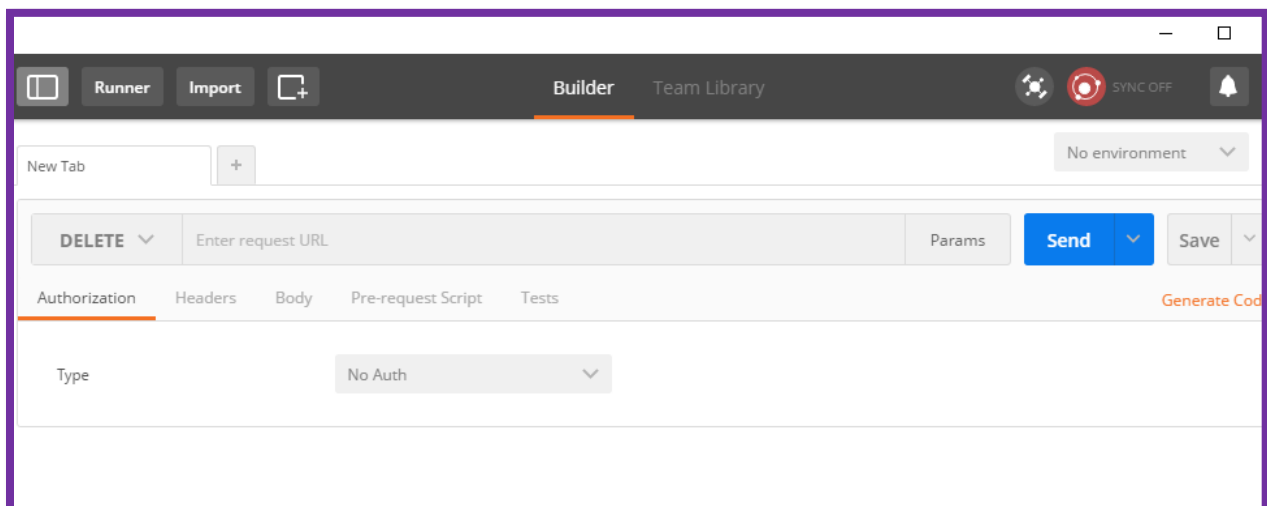
<https://www.getpostman.com/>



2) Now we need verify in our extension of chrome and we:



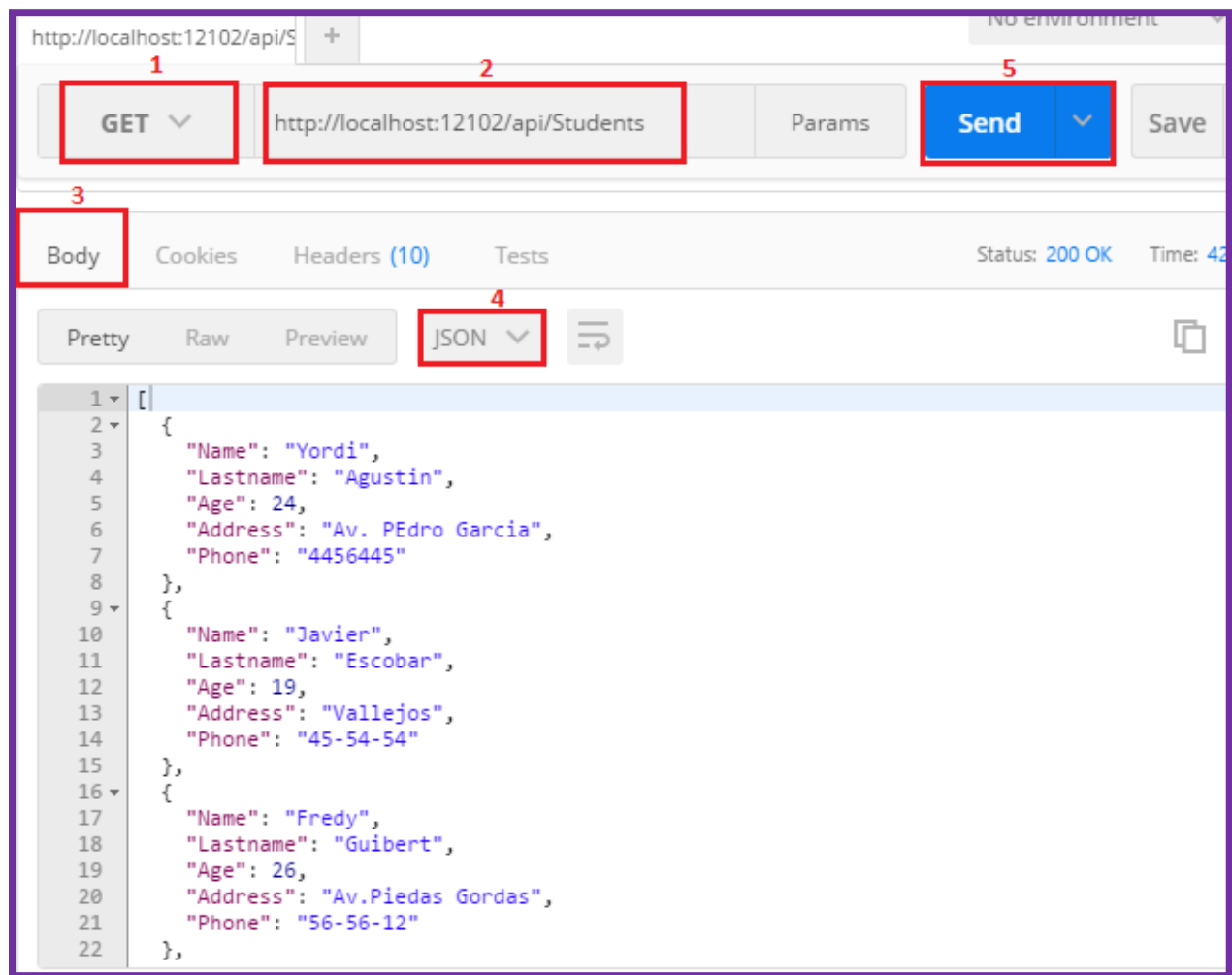
3) We open application and we find something like this:



4) We open out first api of type **GET**:

In my case it is:

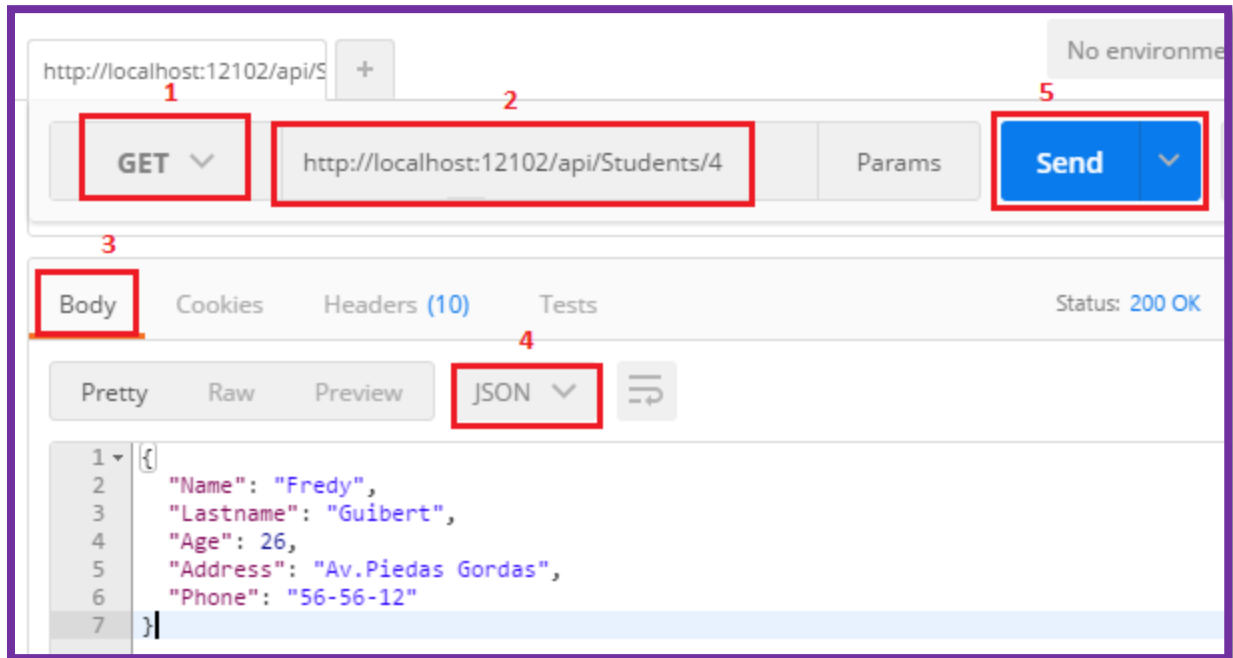
<http://localhost:12102/api/Students>



5) We open our second api of type **GET** with a parameter {id}:

In my case it is:

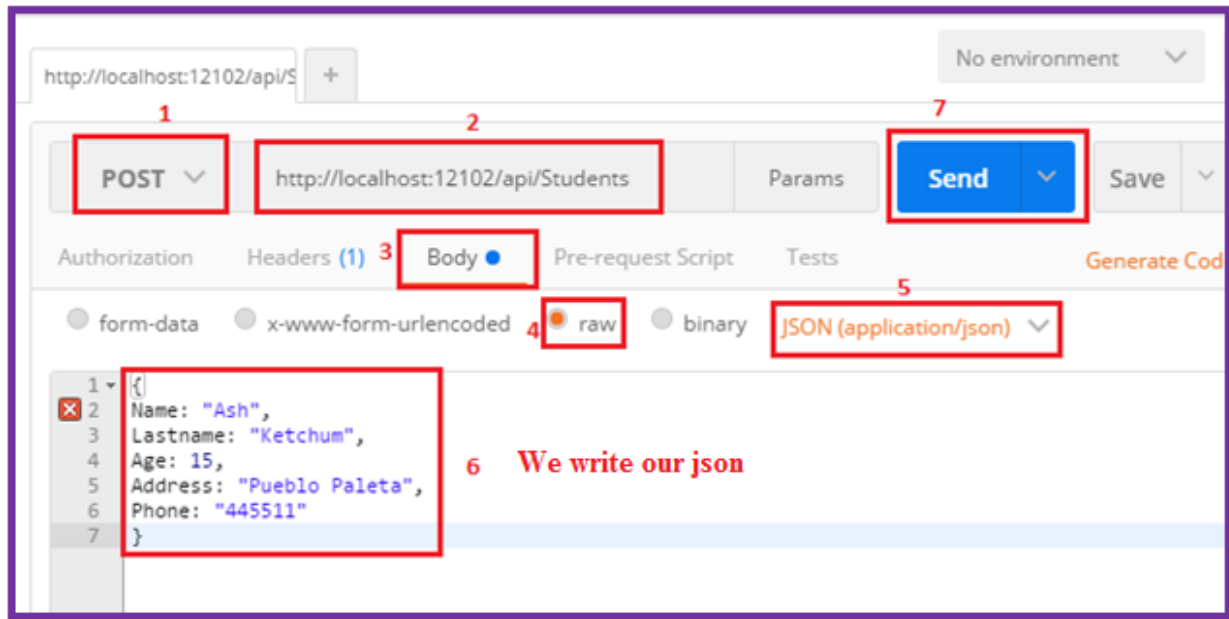
<http://localhost:12102/api/Students/{id}> → {id} = 1,2,3,4,5,6,7



6) We open out third api of type **POST**:

In my case it is:

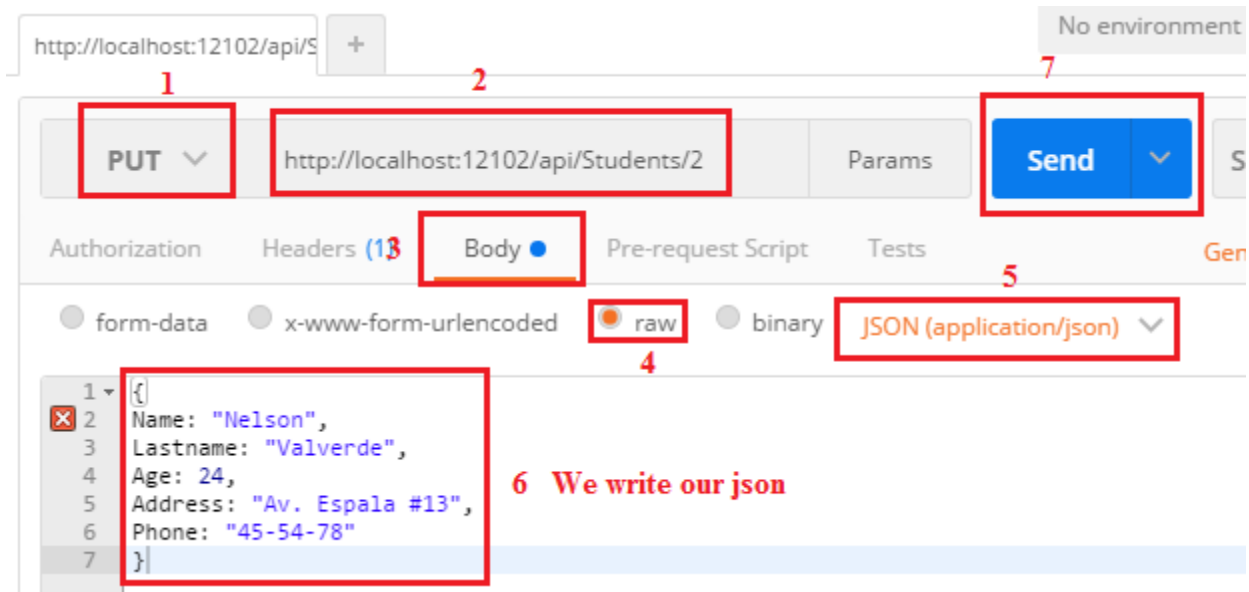
<http://localhost:12102/api/Students>



7) We open out fourth api of type **PUT**:

In my case it is:

<http://localhost:12102/api/Students/{id}> → {id} = 1,2,3,4,5,6,7



8) We open out fifth api of type **DELETE**:

In my case it is:

<http://localhost:12102/api/Students/{id}> → {id} = 1,2,3,4,5,6,7

