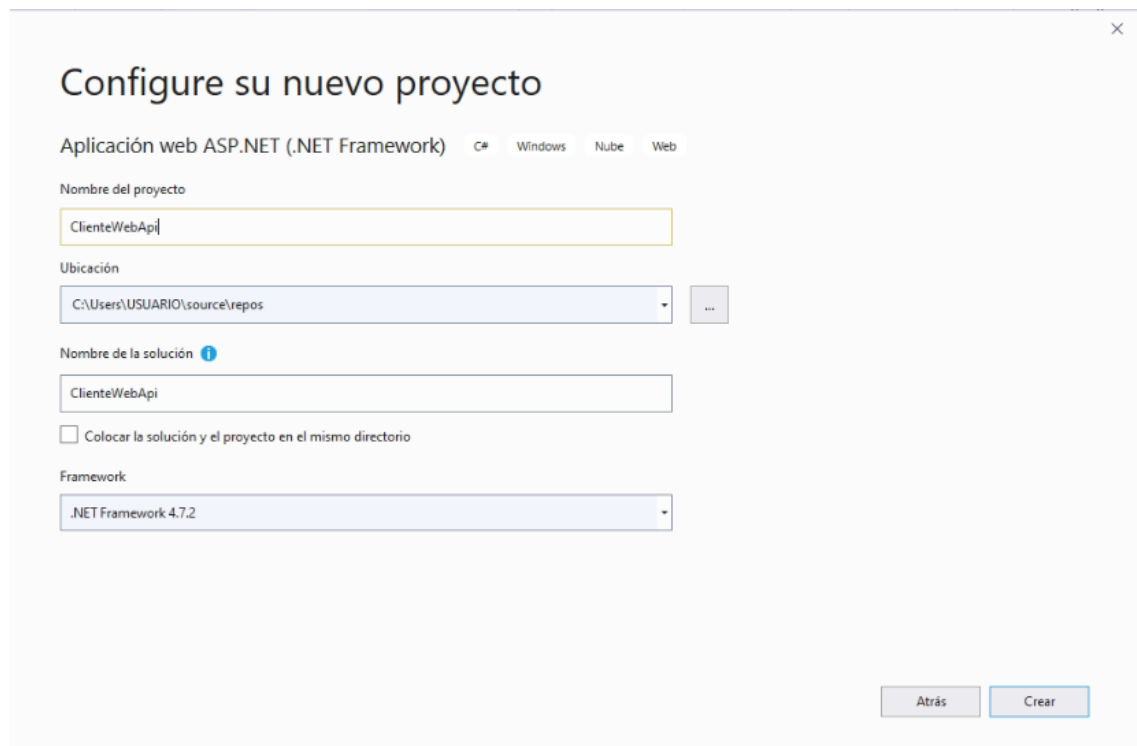
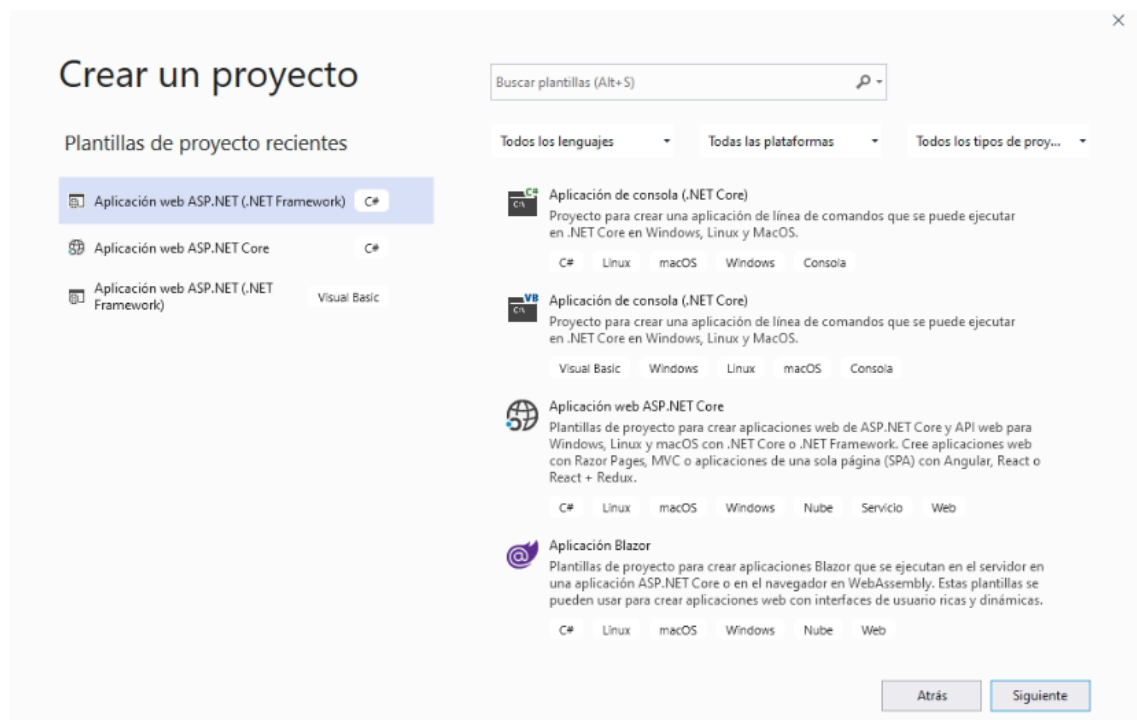



TALLER CREAR UN CLIENTE PARA CONSUMIR UN CRUD WEB SERVICE C#


Objetivo: Crear un cliente para consumir un web service en C#




Crear una aplicación web ASP.NET


Vacio


Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.


Web Forms


Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.


MVC

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.


API web

Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.


Aplicación de página única

Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación
Sin autenticación
[Cambiar](#)


Agregar carpetas y referencias principales
☐ Formularios Web Forms
☒ MVC
☐ API web

Avanzado
☒ Configurar para HTTPS
☐ Compatibilidad con Docker
(Requiere [Docker Desktop](#))
☐ Crear también un proyecto para pruebas unitarias


[Atrás](#)
[Crear](#)

Activar Wi


Crear una aplicación web ASP.NET


Vacio


Una plantilla de proyecto vacía para crear aplicaciones ASP.NET. Esta plantilla no tiene contenido.


Web Forms


Una plantilla de proyecto para crear aplicaciones de ASP.NET Web Forms. ASP.NET Web Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlados por la interfaz de usuario y con acceso a datos.


MVC

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.


API web

Plantilla de proyecto para crear servicios HTTP REST que pueden llegar a una amplia gama de clientes, como, por ejemplo, exploradores y dispositivos móviles.


Aplicación de página única

Una plantilla de proyectos para crear aplicaciones HTML5 atractivas controladas por JavaScript del lado cliente mediante ASP.NET Web API. Las aplicaciones de una sola página proporcionan una experiencia de usuario atractiva que incluye interacciones del lado cliente mediante HTML5, CSS3 y JavaScript.

Autenticación
Sin autenticación
[Cambiar](#)

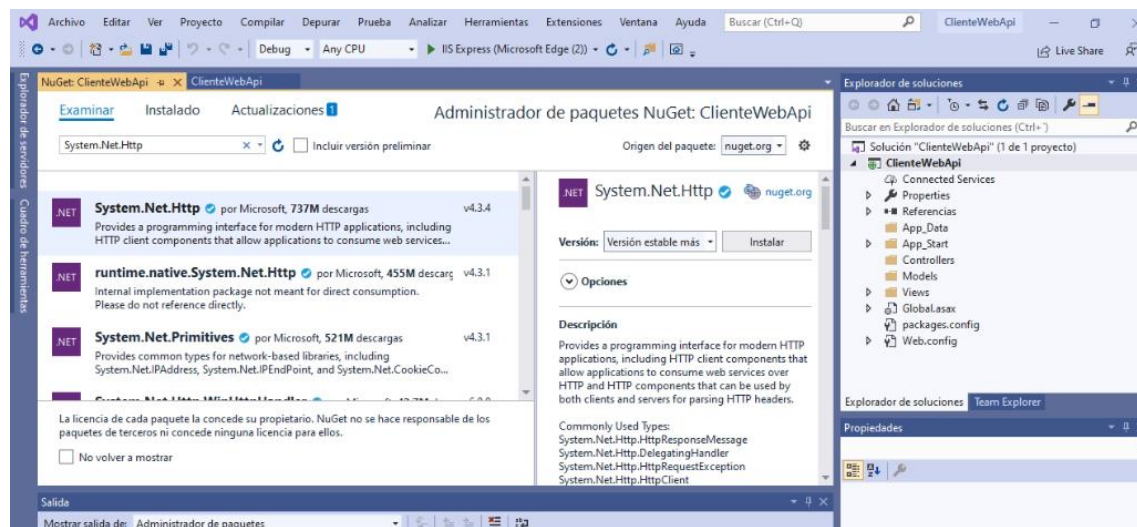
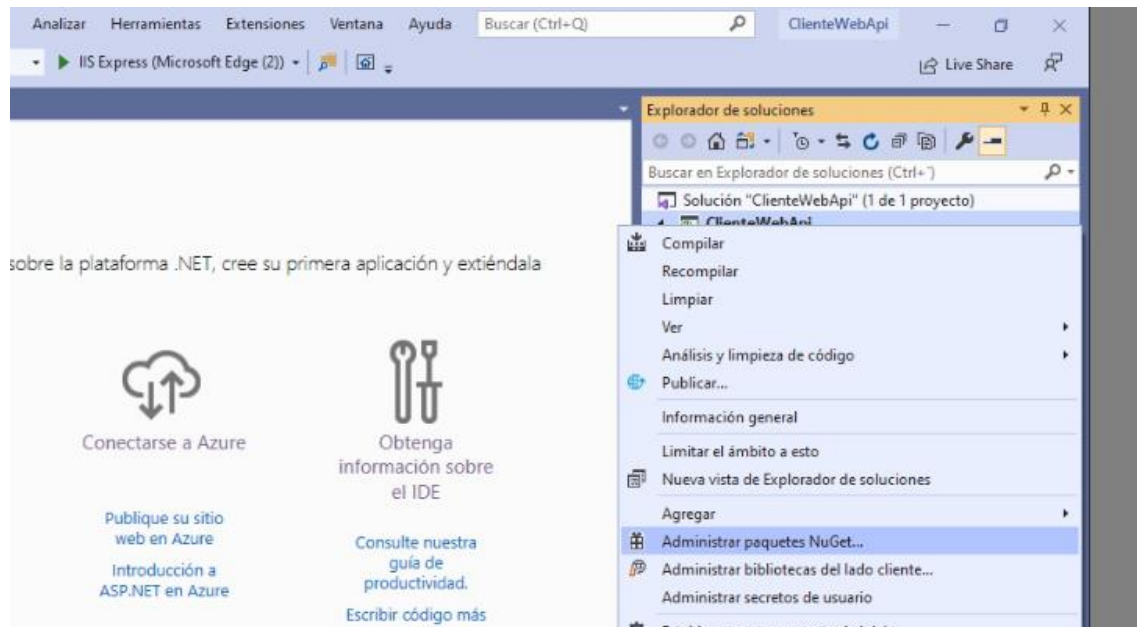
Agregar carpetas y referencias principales
☐ Formularios Web Forms
☒ MVC
☐ API web

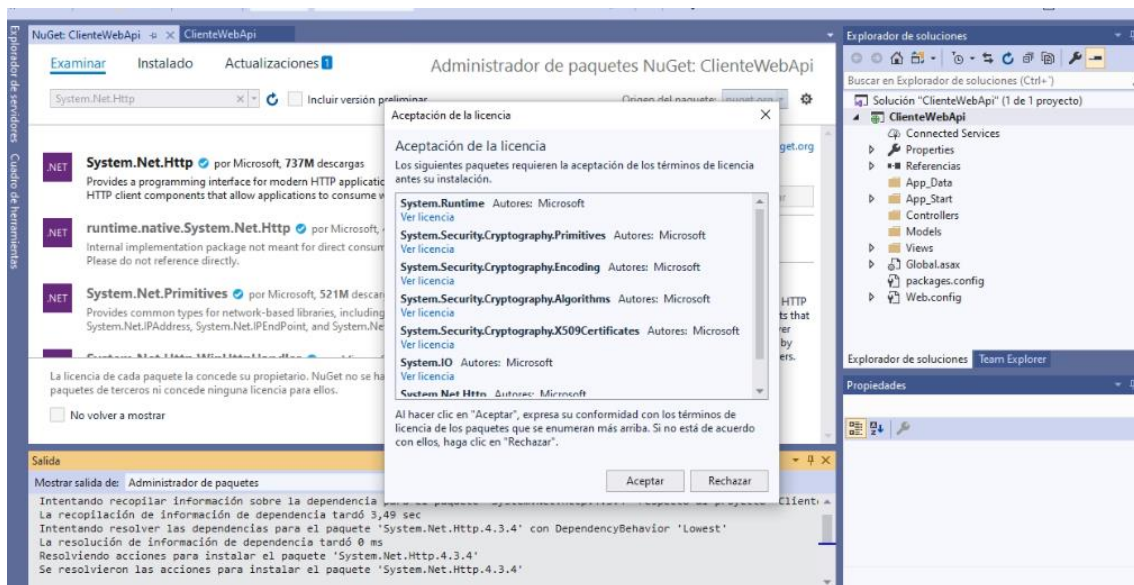
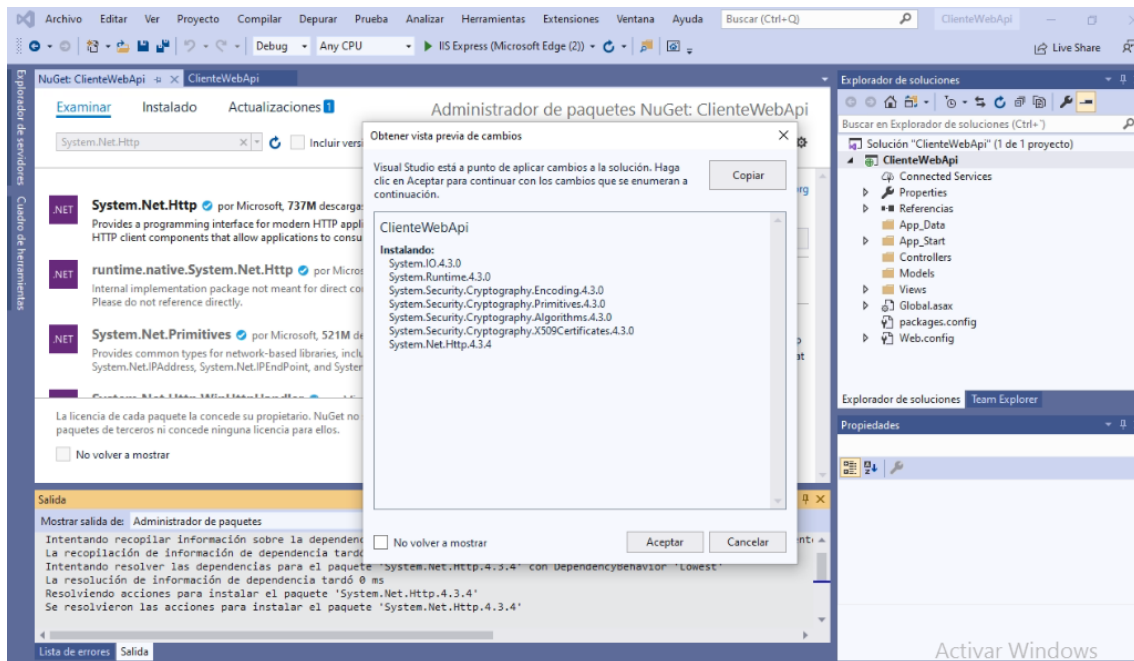
Avanzado
☒ Configurar para HTTPS
☐ Compatibilidad con Docker
(Requiere [Docker Desktop](#))
☐ Crear también un proyecto para pruebas unitarias

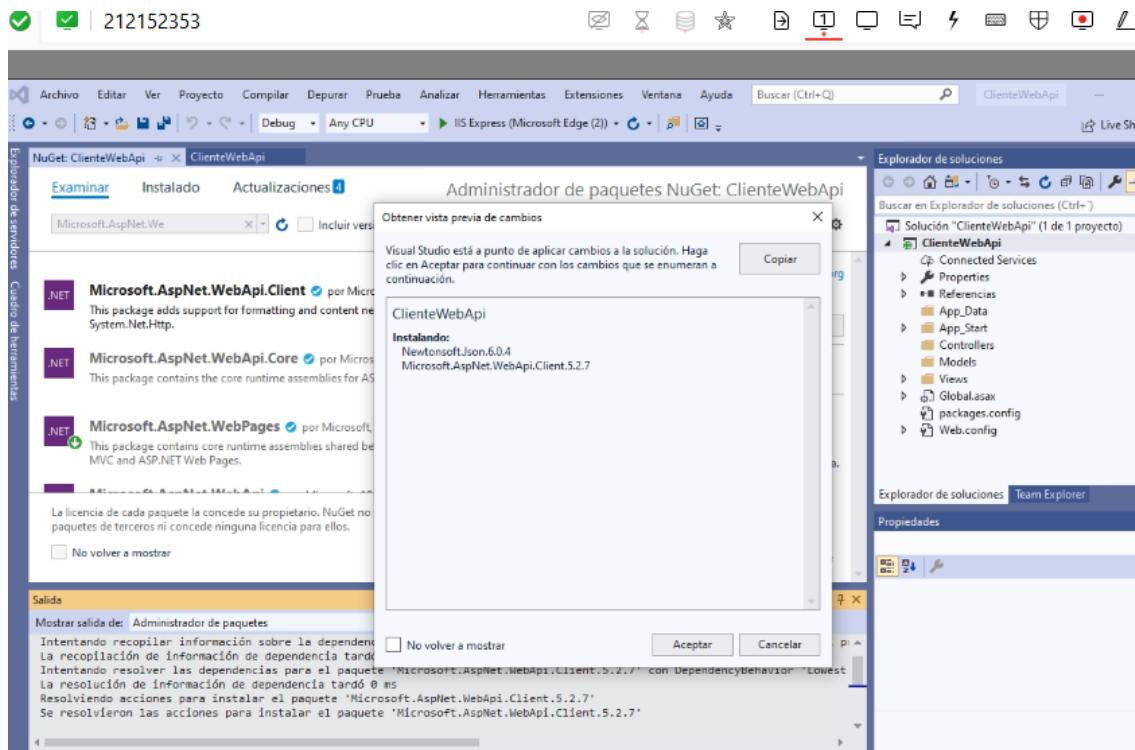
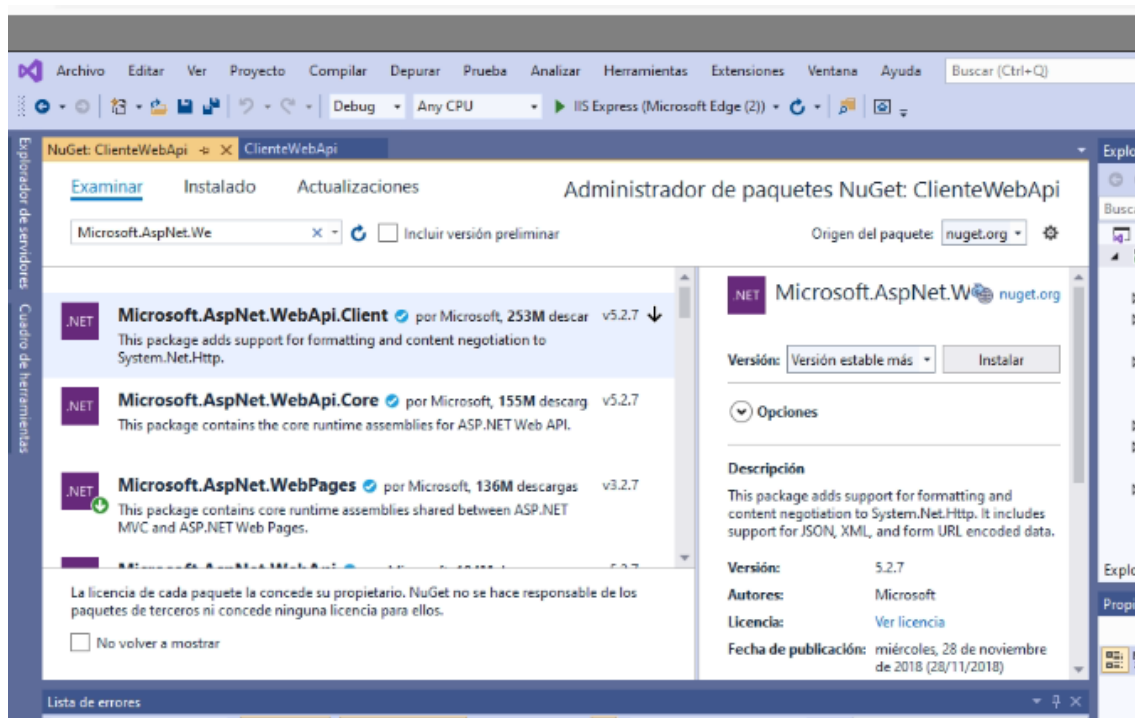
[Atrás](#)
[Crear](#)

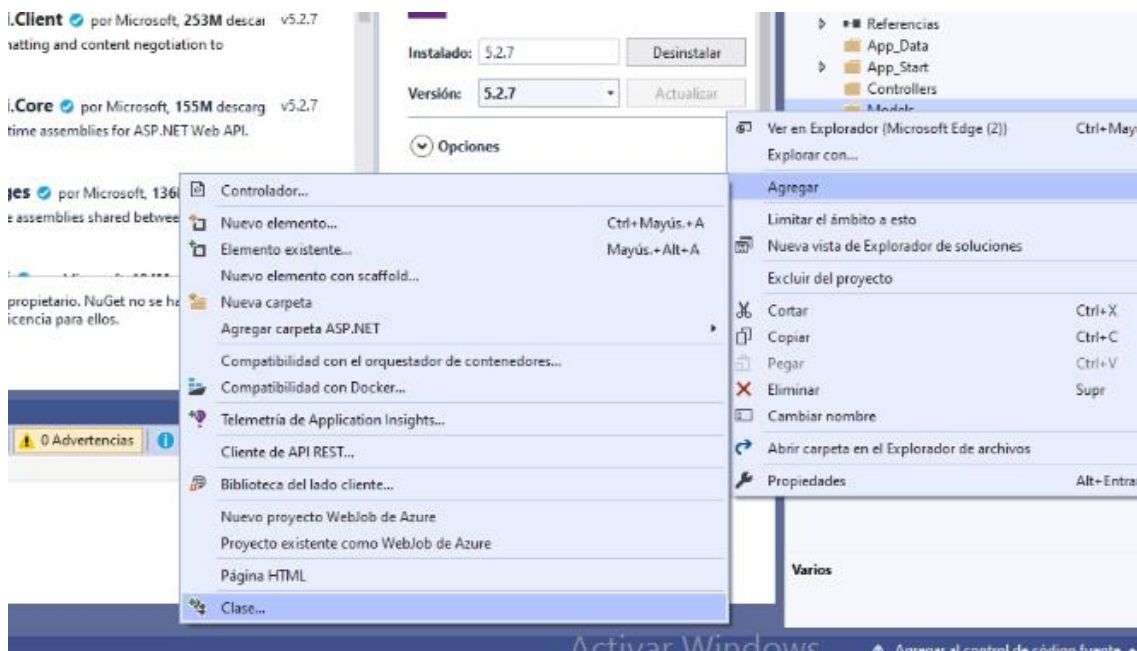
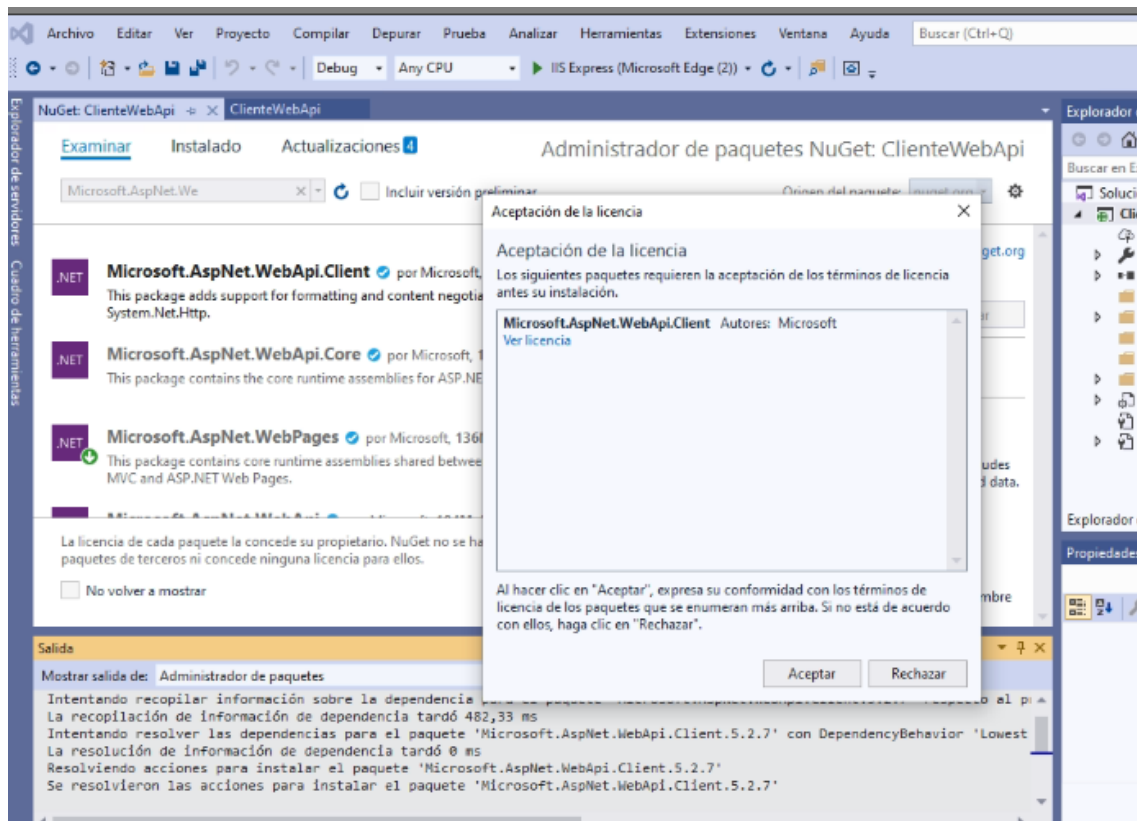
Activar Wi

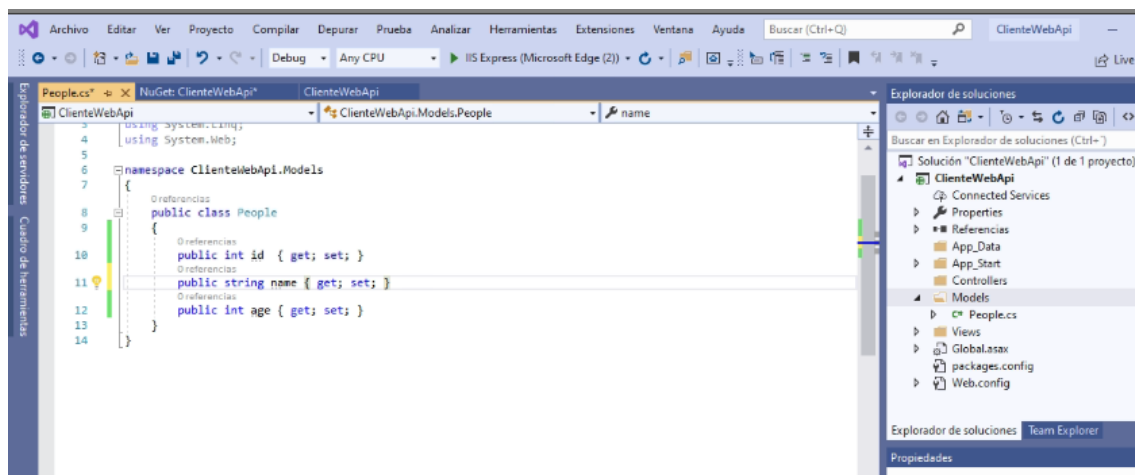
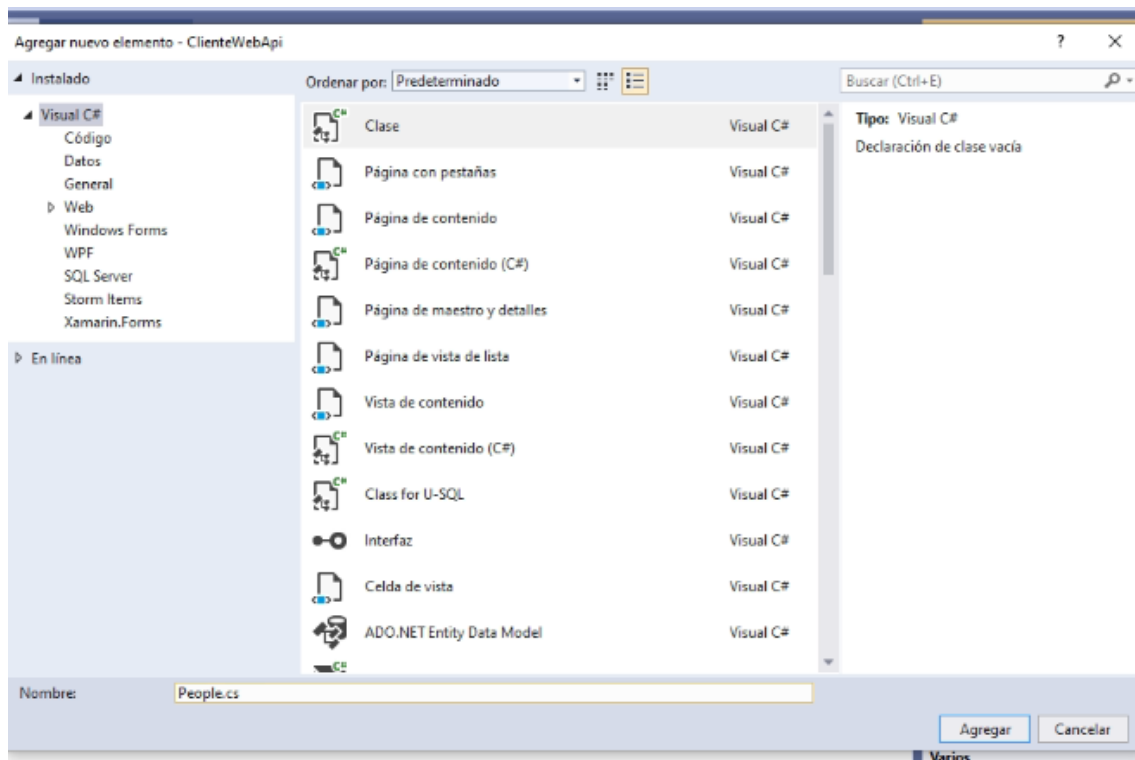
Ir al administrador de paquetes



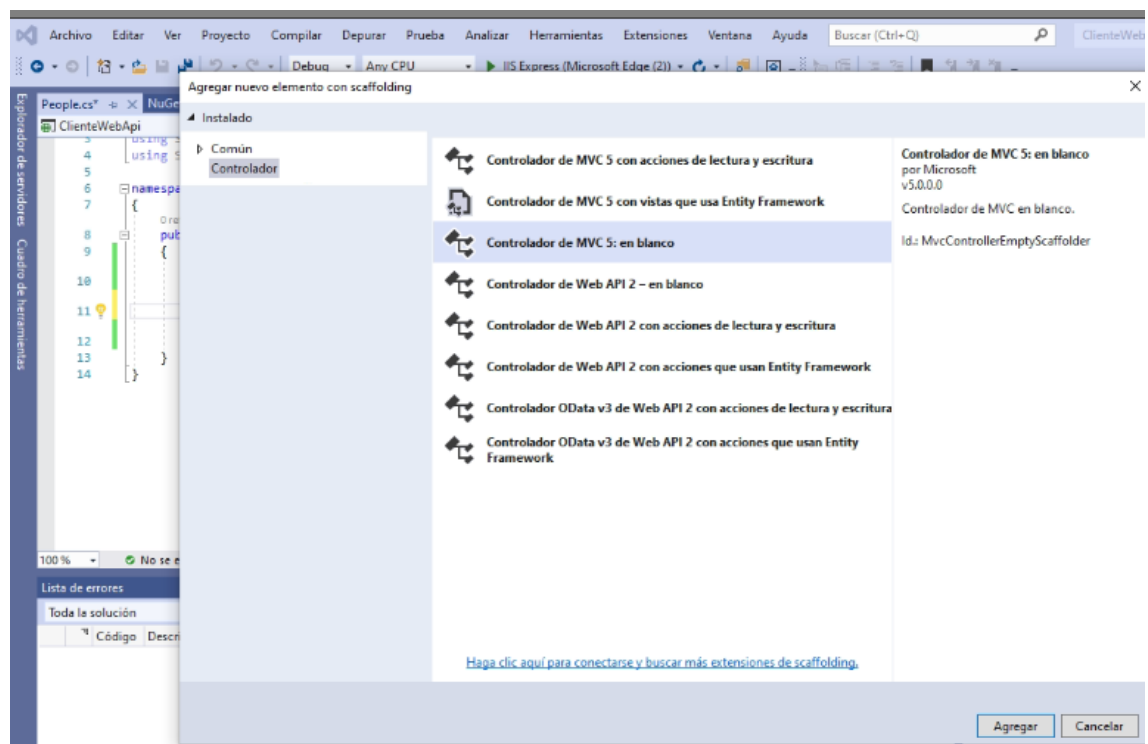
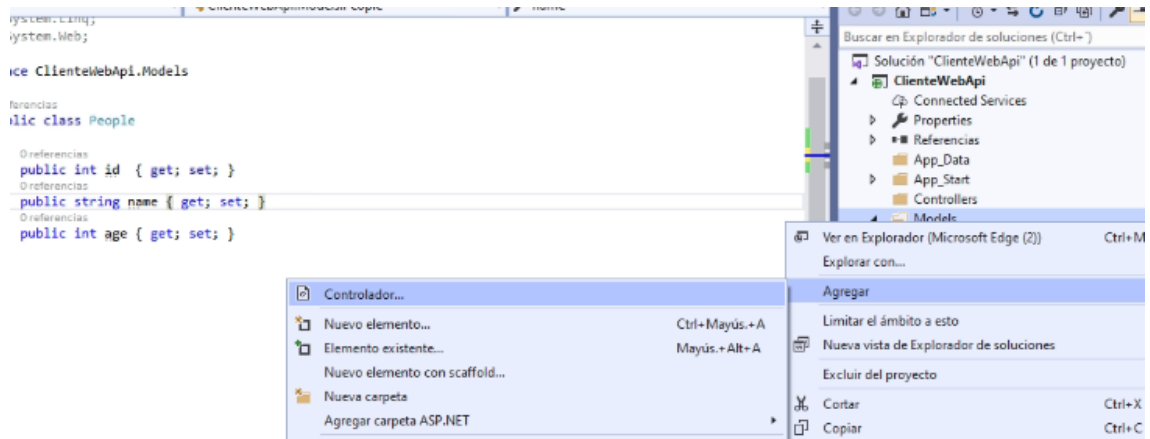


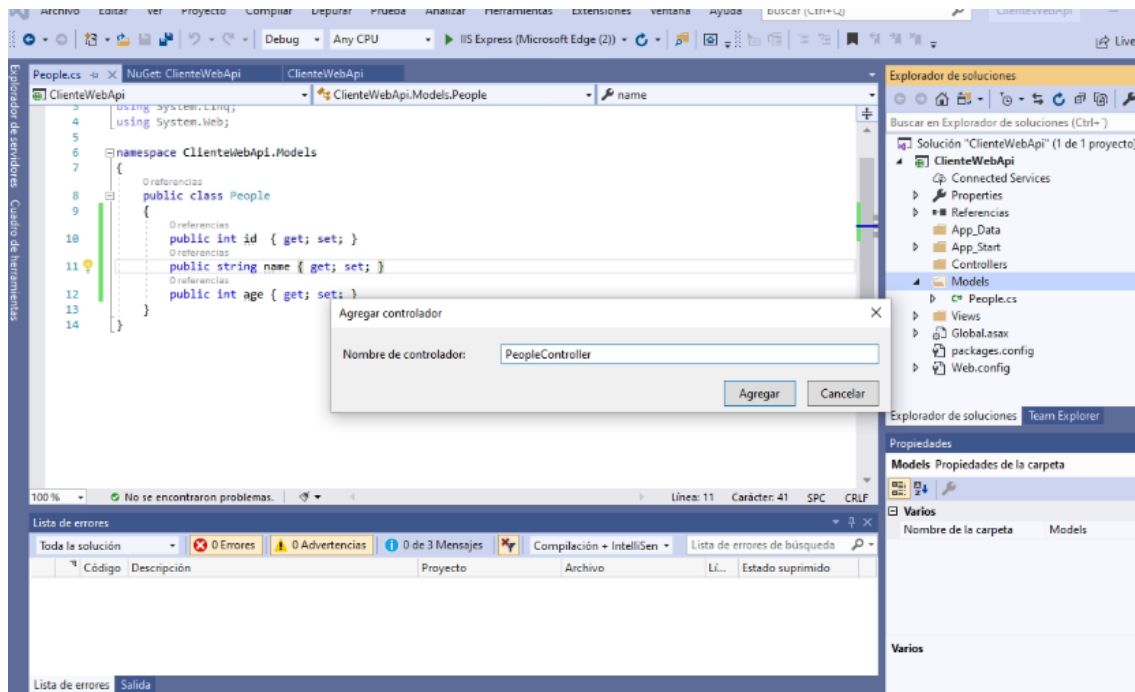




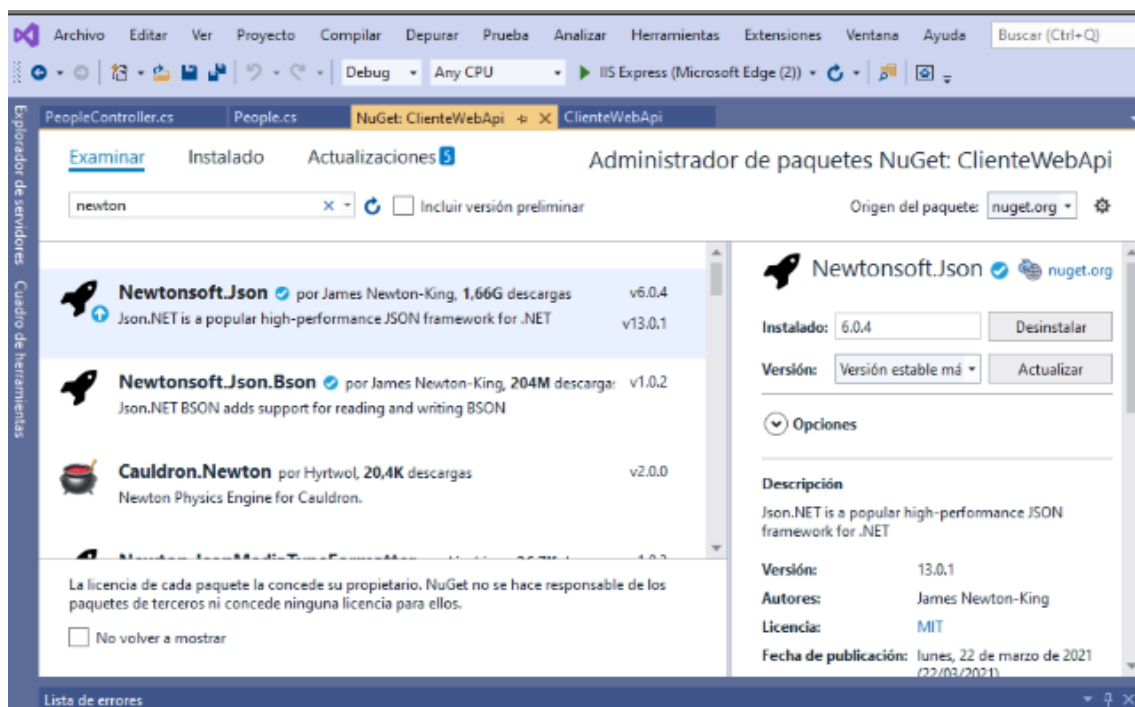


Agregar un controlador





Agregar o actualizar paquete json



Dentro del controlador de la tabla añadimos las librerías adecuadas y el código siguiente.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

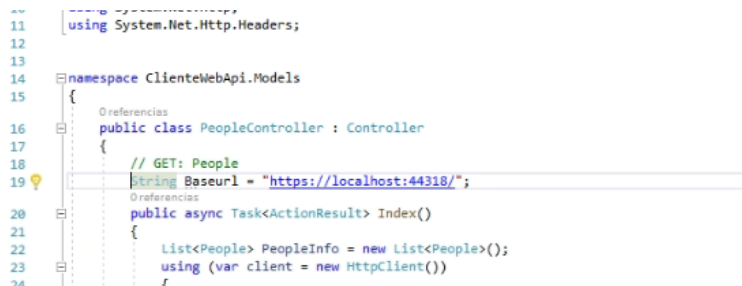
```

using System.Web.Mvc;
using ClienteWebApi.Models;
using System.Threading.Tasks;
using Newtonsoft.Json;
using System.Net.Http;
using System.Net.Http.Headers;

namespace ClienteWebApi.Controllers
{
    public class PeopleController : Controller
    {
        // GET: People
        String Baseurl = "http://localhost:1696/";
        public async Task<ActionResult> Index()
        {
            List<People> PeopleInfo = new List<People>();
            using (var client = new HttpClient()) {
                client.BaseAddress = new Uri(Baseurl);
                client.DefaultRequestHeaders.Clear();
                client.DefaultRequestHeaders.Accept.Add(new
MediaTyewithQualityHeaderValue("application/json"));

                HttpResponseMessage Res = await client.GetAsync("api/people/");
                if (Res.IsSuccessStatusCode)
                {
                    var PeopleResponse = Res.Content.ReadAsStringAsync().Result;
                    PeopleInfo =
JsonConvert.DeserializeObject<List<People>>(PeopleResponse);
                }
            }
            return View(PeopleInfo);
        }
    }
}

```

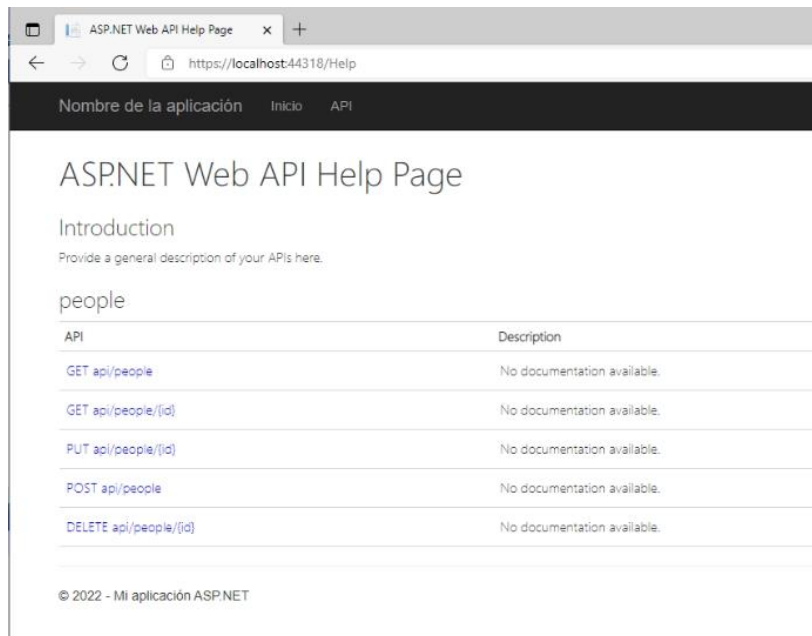


```

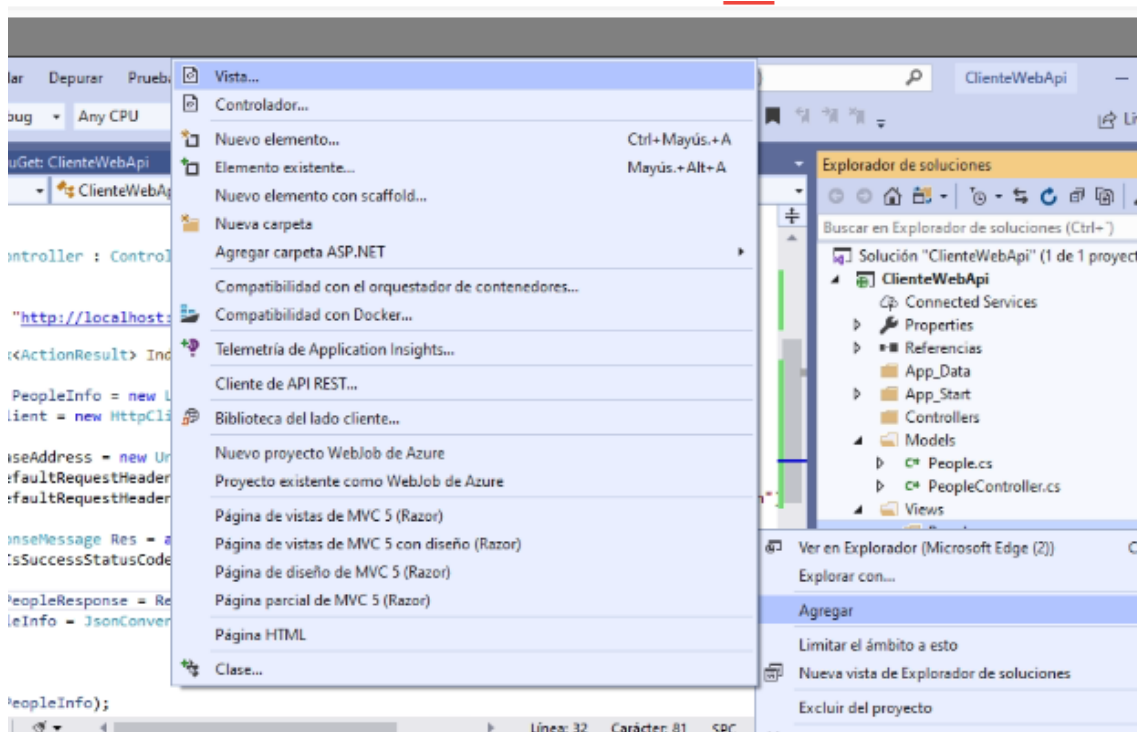
11 using System.Net.Http.Headers;
12
13
14 namespace ClienteWebApi.Models
15 {
16     public class PeopleController : Controller
17     {
18         // GET: People
19         String Baseurl = "https://localhost:44318/";
20         public async Task<ActionResult> Index()
21         {
22             List<People> PeopleInfo = new List<People>();
23             using (var client = new HttpClient())
24             {

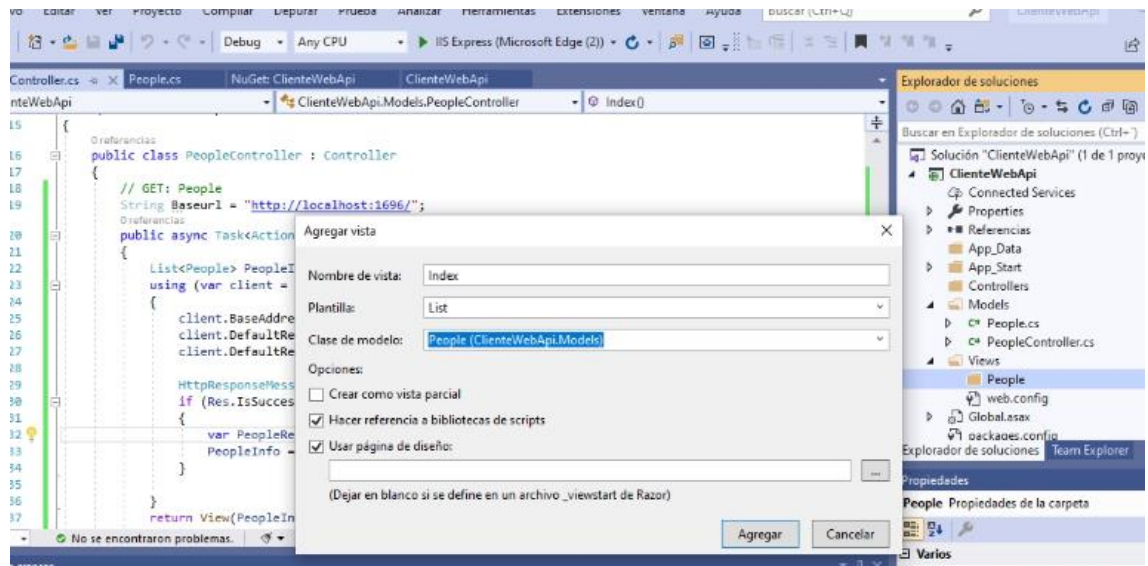
```

La base url de penderá de la url del web service

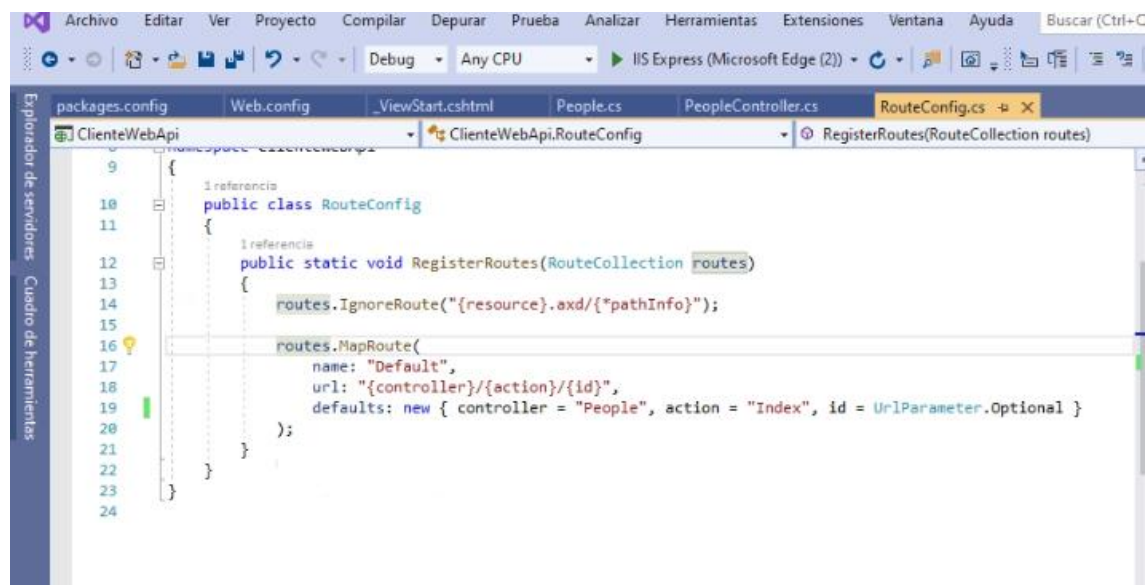


Añadir una vista.

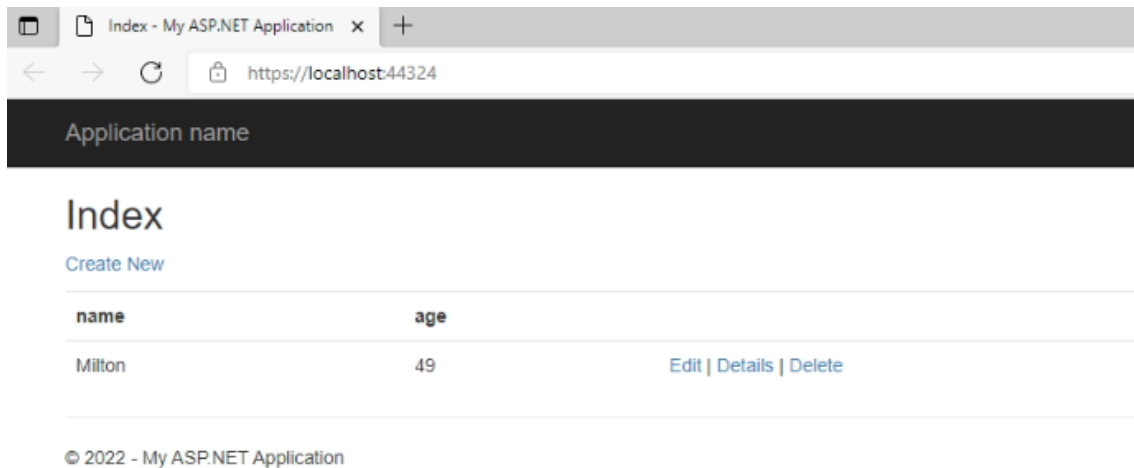




Reemplazar People por Home en el routeConfig



Probar la aplicación



Modificar el controlador para añadir el créate.

```
using System;
```

```
using System.Collections.Generic;
```

```
using System.Linq;
```

```
using System.Web;
```

```
using System.Web.Mvc;
```

```
using ClienteWebApi.Models;
```

```
using System.Threading.Tasks;
```

```
using Newtonsoft.Json;
```

```
using System.Net.Http;
```

```
using System.Net.Http.Headers;
```

```
namespace ClienteWebApi.Models
```

```
{
```

```
    public class PeopleController : Controller
```

```
    {
```

```
        // GET: People
```

```
        String Baseurl = "https://localhost:44318/";
```

```
        public async Task<ActionResult> Index()
```

```

{
    List<People> PeopleInfo = new List<People>();
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri(Baseurl);
        client.DefaultRequestHeaders.Clear();
        client.DefaultRequestHeaders.Accept.Add(new
MediaTypesWithQualityHeaderValue("application/json"));

        HttpResponseMessage Res = await client.GetAsync("api/people/");
        if (Res.IsSuccessStatusCode)
        {
            var PeopleResponse = Res.Content.ReadAsStringAsync().Result;
            PeopleInfo = JsonConvert.DeserializeObject<List<People>>(PeopleResponse);
        }

    }

    return View(PeopleInfo);
    //return View("People / Index");

}

public ActionResult create()
{

    return View();
}

[HttpPost]
public ActionResult create(People people)
{
    using (var client = new HttpClient())
    {

```



```

        client.BaseAddress = new Uri("http://localhost:1696/api/people");

        var postTask = client.PostAsJsonAsync<People>("people", people);

        postTask.Wait();

        var result = postTask.Result;

        if (result.IsSuccessStatusCode)
        {
            return RedirectToAction("Index");
        }

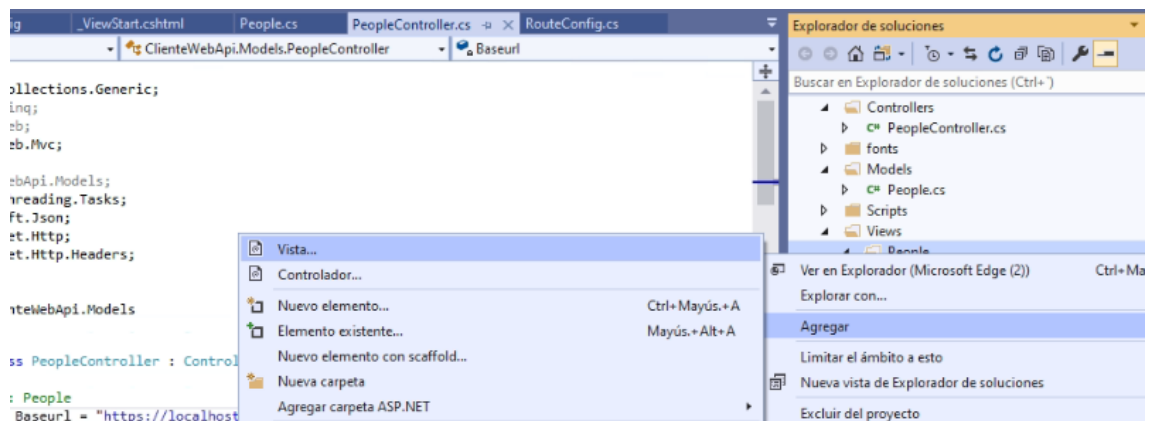
        ModelState.AddModelError(string.Empty, "Error contacta al administador");

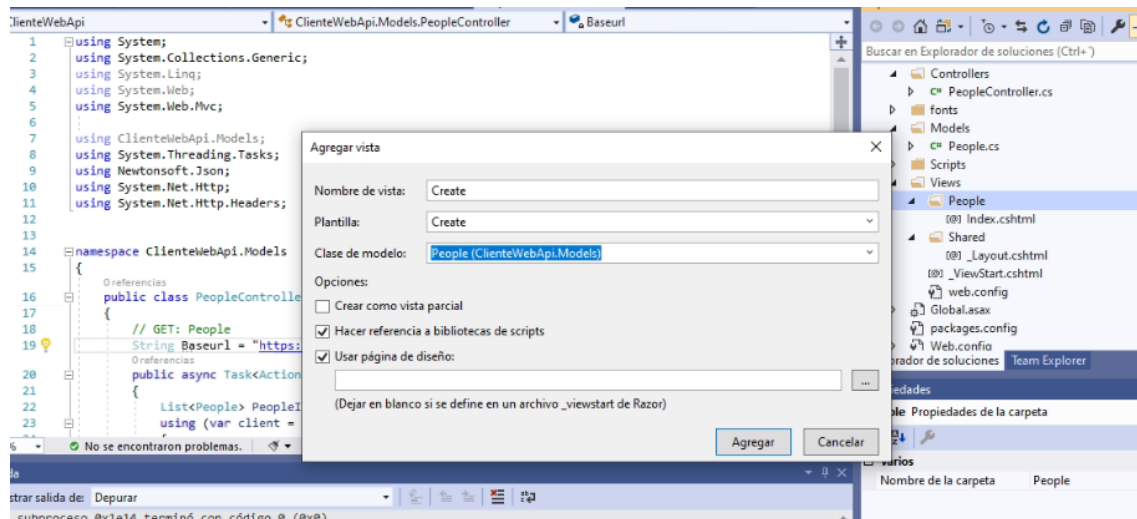
        return View(people);
    }

}
}

```

Añadir la vista.





Verificar el código de la vista

@model ClienteWebApi.Models.People

@{

 ViewBag.Title = "Create";

}

<h2>Create</h2>

@using (Html.BeginForm())

{

 @Html.AntiForgeryToken()

<div class="form-horizontal">

 <h4>People</h4>

 <hr />

 @Html.ValidationSummary(true, "", new { @class = "text-danger" })

 <div class="form-group">

 @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label col-md-2" })

```

        <div class="col-md-10">

            @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class =
"form-control" } })

            @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-
danger" })

        </div>

    </div>

    <div class="form-group">

        @Html.LabelFor(model => model.age, htmlAttributes: new { @class = "control-label col-
md-2" })

        <div class="col-md-10">

            @Html.EditorFor(model => model.age, new { htmlAttributes = new { @class = "form-
control" } })

            @Html.ValidationMessageFor(model => model.age, "", new { @class = "text-danger"
})

        </div>

    </div>

    <div class="form-group">

        <div class="col-md-offset-2 col-md-10">

            <input type="submit" value="Create" class="btn btn-default" />

        </div>

    </div>

</div>
}

<div>

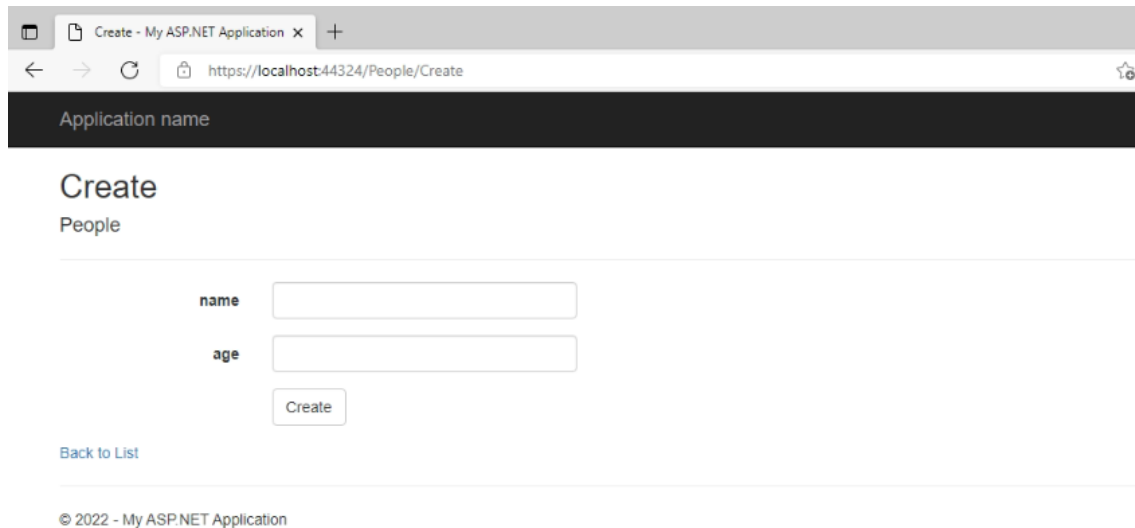
    @Html.ActionLink("Back to List", "Index")

</div>

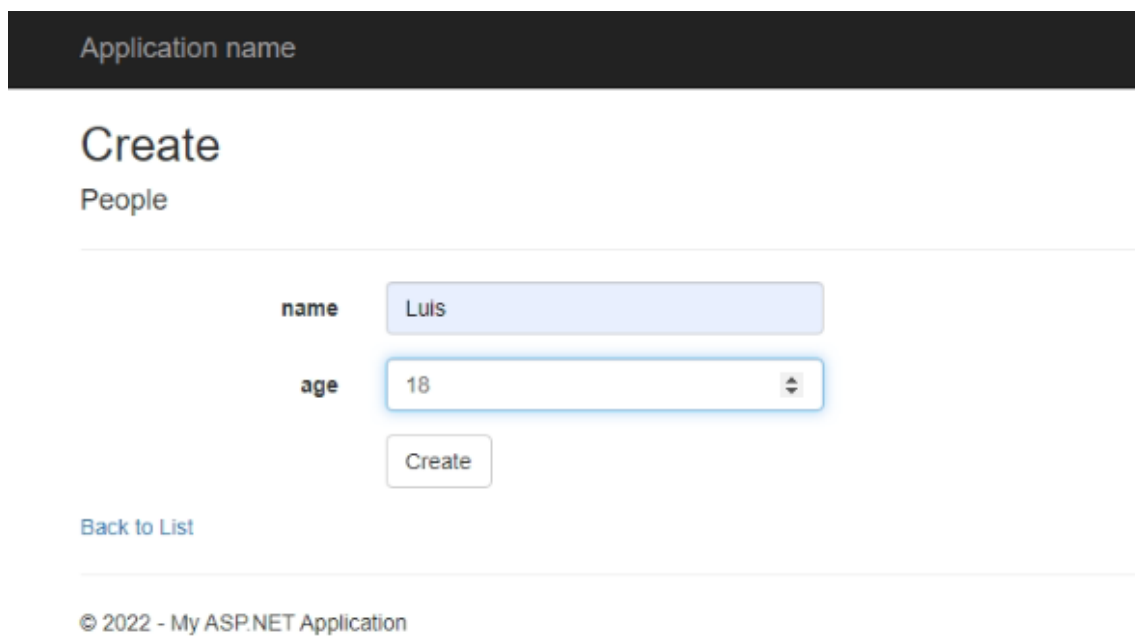
<script src="~/Scripts/jquery-3.4.1.min.js"></script>
<script src="~/Scripts/jquery.validate.min.js"></script>

```

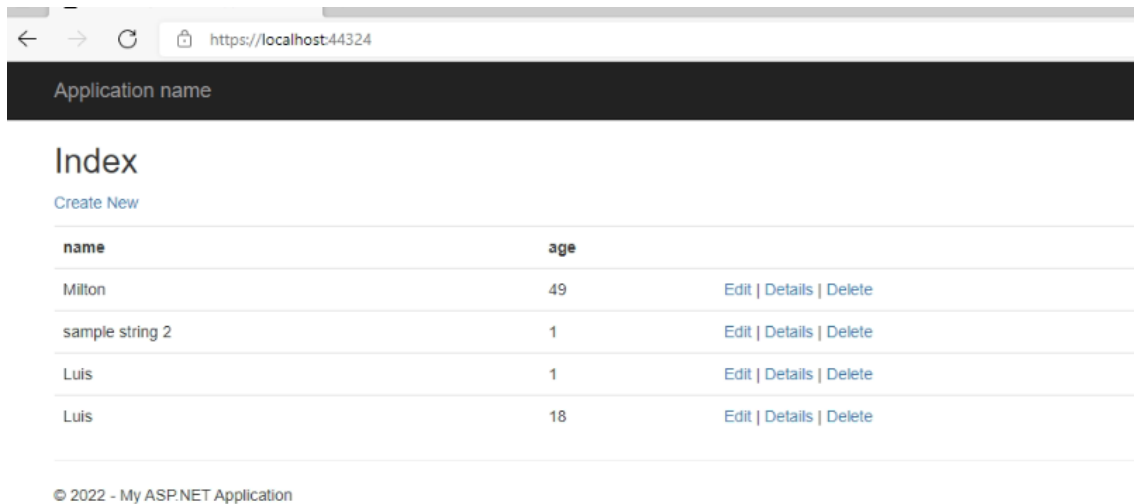
```
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```



The screenshot shows a web browser window with the title 'Create - My ASP.NET Application'. The address bar displays 'https://localhost:44324/People/Create'. The page has a dark header bar with the text 'Application name'. Below the header, the main content area has the title 'Create' and the subtitle 'People'. There is a form with two input fields: 'name' and 'age'. The 'name' field is empty, and the 'age' field is empty. Below the fields is a 'Create' button. At the bottom of the form, there is a link 'Back to List'. The footer of the page shows '© 2022 - My ASP.NET Application'.



The screenshot shows the same web browser window as the previous one, but with form data entered. The 'name' field now contains the text 'Luis', and the 'age' field now contains the number '18'. The 'Create' button is still present. The 'Back to List' link is also visible. The footer of the page shows '© 2022 - My ASP.NET Application'.



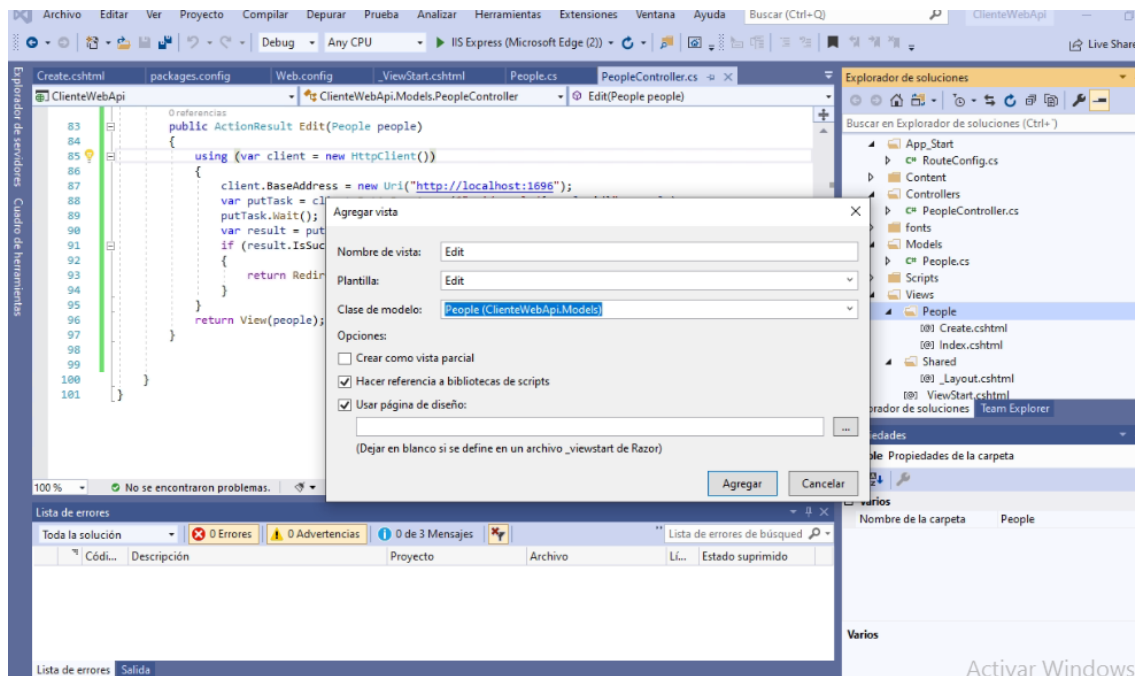
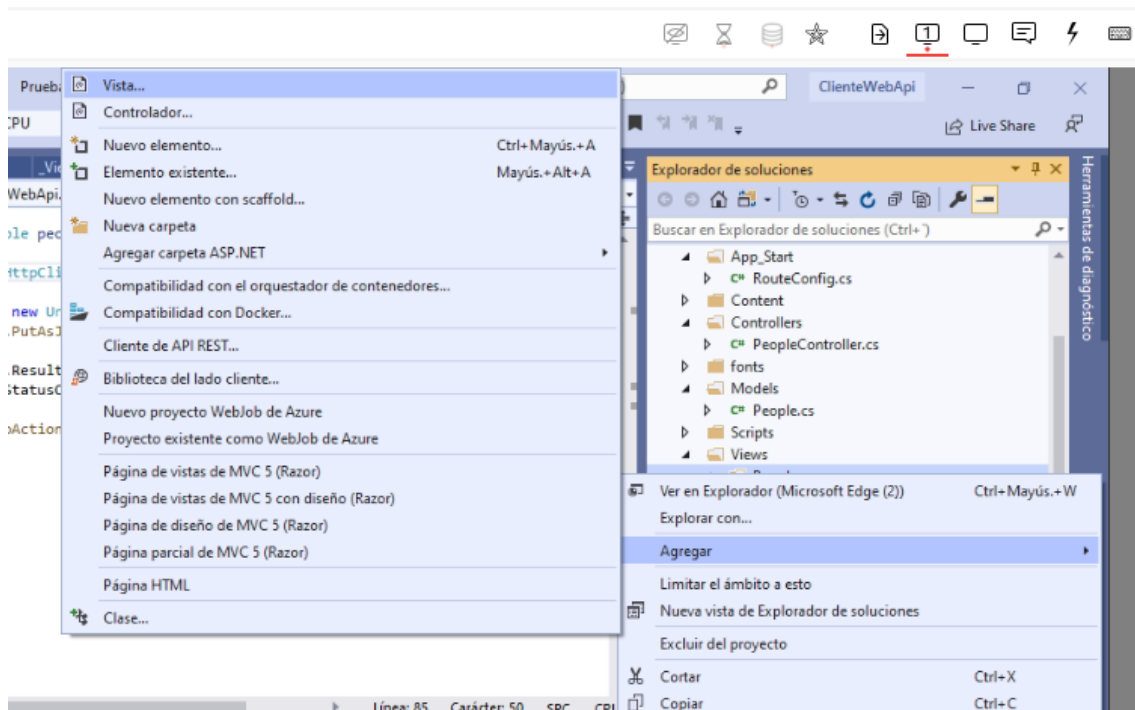
Añadir el código para actualizar en el controlador

```
public ActionResult Edit(int id) {
    People people = null;
    using (var client = new HttpClient()) {
        client.BaseAddress = new Uri("http://localhost:1696");
        var responseTask = client.GetAsync("api/people/" + id.ToString());
        responseTask.Wait();
        var result = responseTask.Result;
        if (result.IsSuccessStatusCode) {
            var readTask = result.Content.ReadAsAsync<People>();
            readTask.Wait();
            people = readTask.Result;
        }
    }

    return View(people);
}

[HttpPost]
public ActionResult Edit(People people) {
    using (var client = new HttpClient()) {
        client.BaseAddress = new Uri("http://localhost:1696");
        var putTask =
client.PutAsJsonAsync($"api/people/{people.id}", people);
        putTask.Wait();
        var result = putTask.Result;
        if (result.IsSuccessStatusCode)
        {
            return RedirectToAction("Index");
        }
    }
    return View(people);
}
```

Añadir la vista



@model ClienteWebApi.Models.People

@{

 ViewBag.Title = "Edit";

}


```
<h2>Edit</h2>
```

```
@using (Html.BeginForm())
```

```
{
```

```
    @Html.AntiForgeryToken()
```

```
<div class="form-horizontal">
```

```
    <h4>People</h4>
```

```
    <hr />
```

```
    @Html.ValidationSummary(true, "", new { @class = "text-danger" })
```

```
    @Html.HiddenFor(model => model.id)
```

```
    <div class="form-group">
```

```
        @Html.LabelFor(model => model.name, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
        <div class="col-md-10">
```

```
            @Html.EditorFor(model => model.name, new { htmlAttributes = new { @class = "form-control" } })
```

```
            @Html.ValidationMessageFor(model => model.name, "", new { @class = "text-danger" })
```

```
        </div>
```

```
    </div>
```

```
    <div class="form-group">
```

```
        @Html.LabelFor(model => model.age, htmlAttributes: new { @class = "control-label col-md-2" })
```

```
        <div class="col-md-10">
```

```
            @Html.EditorFor(model => model.age, new { htmlAttributes = new { @class = "form-control" } })
```

```
            @Html.ValidationMessageFor(model => model.age, "", new { @class = "text-danger" })
```

```
</div>
```

```
</div>
```

```
<div class="form-group">
```

```
  <div class="col-md-offset-2 col-md-10">
```

```
    <input type="submit" value="Save" class="btn btn-default" />
```

```
  </div>
```

```
</div>
```

```
</div>
```

```
}
```

```
<div>
```

```
  @Html.ActionLink("Back to List", "Index")
```

```
</div>
```

```
<script src="~/Scripts/jquery-3.4.1.min.js"></script>
```

```
<script src="~/Scripts/jquery.validate.min.js"></script>
```

```
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
```

index - My ASP.NET Application x

← → ↻ https://localhost:44324

Application name

Index

[Create New](#)

name	age	
Milton	49	Edit Details Delete
sample string 2	1	Edit Details Delete
Luis	13	Edit Details Delete
Luis	18	Edit Details Delete

© 2022 - My ASP.NET Application

Edit - My ASP.NET Application x +

← → ↻ https://localhost:44324/People/Edit/3

Application name

Edit

People

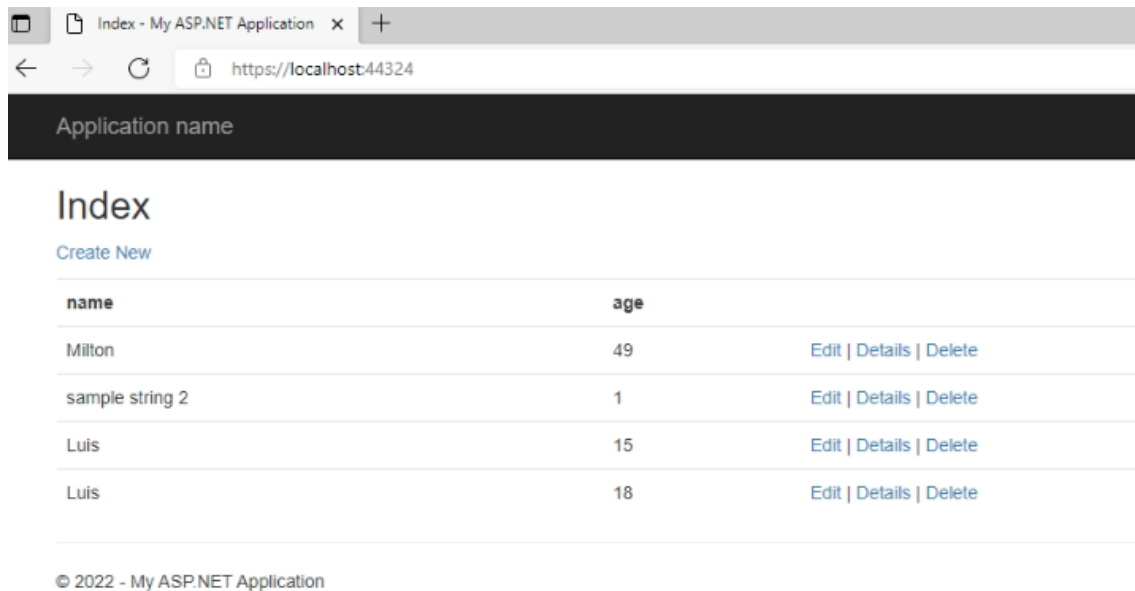
name

age

Save

[Back to List](#)

© 2022 - My ASP.NET Application



Ahora agregar el código para eliminar en el controlador

```
public ActionResult Delete(int id)
{
    People people = null;
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("https://localhost:44318");
        var responseTask = client.GetAsync("api/people/" +
id.ToString());
        responseTask.Wait();
        var result = responseTask.Result;
        if (result.IsSuccessStatusCode)
        {
            var readTask = result.Content.ReadAsAsync<People>();
            readTask.Wait();
            people = readTask.Result;
        }
    }

    return View(people);
}

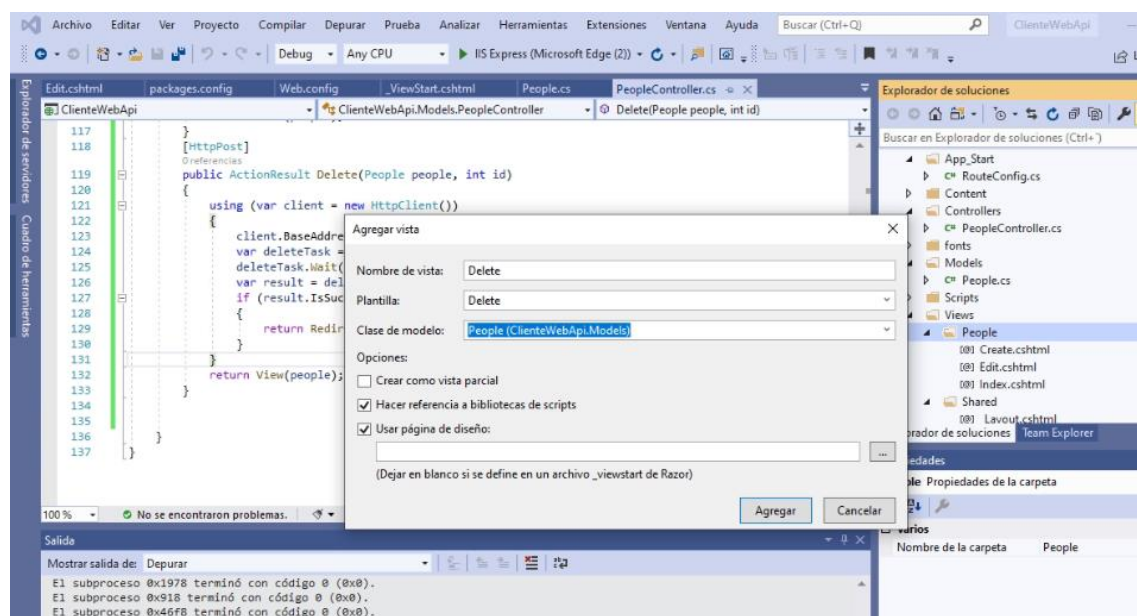
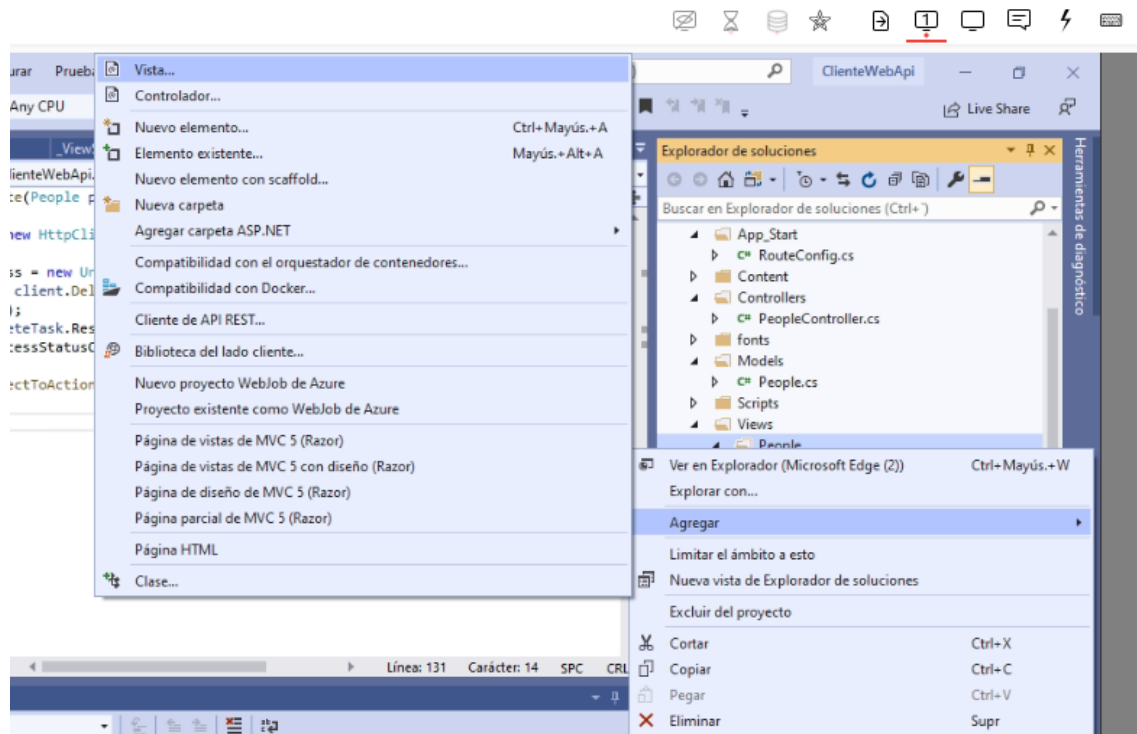
[HttpPost]
public ActionResult Delete(People people, int id)
{
    using (var client = new HttpClient())
    {
        client.BaseAddress = new Uri("https://localhost:44318");
        var deleteTask =
client.DeleteAsync($"api/people/" + id.ToString());
        deleteTask.Wait();
        var result = deleteTask.Result;
        if (result.IsSuccessStatusCode)
        {
            return RedirectToAction("Index");
        }
    }
}
```

```

    }
    return View(people);
}

```

Añadir la vista



```
@model ClienteWebApi.Models.People
```

```
@{
```

```
    ViewBag.Title = "Delete";
```

```
}
```

```
<h2>Delete</h2>
```

```
<h3>Are you sure you want to delete this?</h3>
```

```
<div>
```

```
    <h4>People</h4>
```

```
    <hr />
```

```
    <dl class="dl-horizontal">
```

```
        <dt>
```

```
            @Html.DisplayNameFor(model => model.name)
```

```
        </dt>
```

```
        <dd>
```

```
            @Html.DisplayFor(model => model.name)
```

```
        </dd>
```

```
        <dt>
```

```
            @Html.DisplayNameFor(model => model.age)
```

```
        </dt>
```

```
        <dd>
```

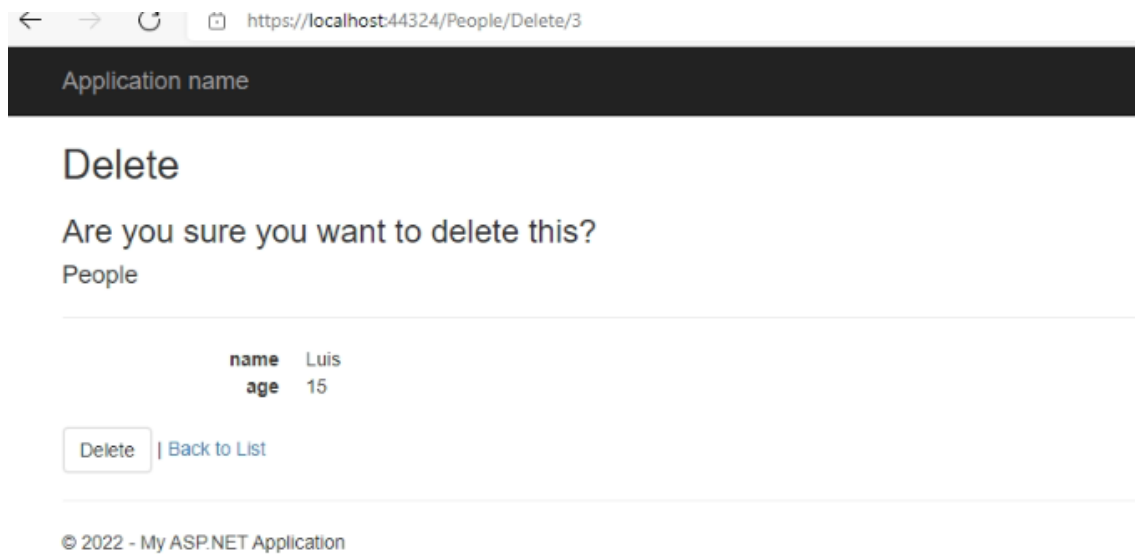
```
            @Html.DisplayFor(model => model.age)
```

```
        </dd>
```

```
</dl>
```



```
@using (Html.BeginForm()) {  
    @Html.AntiForgeryToken()  
  
    <div class="form-actions no-color">  
        <input type="submit" value="Delete" class="btn btn-default" /> |  
        @Html.ActionLink("Back to List", "Index")  
    </div>  
}  
</div>
```



Index - My ASP.NET Application

+

←

→

↻

https://localhost:44324

Application name

Index

Create New

name	age	
Milton	49	Edit Details Delete
sample string 2	1	Edit Details Delete
Luis	18	Edit Details Delete

© 2022 - My ASP.NET Application