

Programación Web .NET Core

Módulo 3

ADO.NET: Plantillas para Vistas 2

Alta de cliente

Continuamos trabajando en nuestro proyecto.

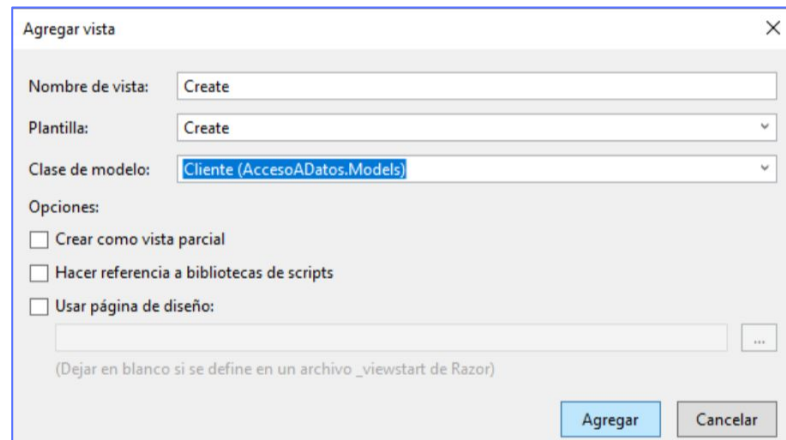
Una vez diseñada la página principal que muestra el contenido de la tabla **Clientes**, implementaremos el alta de clientes.

1. Para esto, modificaremos las acciones llamadas **Create** del **HomeController**:



```
// POST: Cliente/Create
[HttpPost]
public ActionResult Create(FormCollection collection)
{
    try
    {
        // TODO: Add insert logic here
        AdmCliente objAdmCli = new AdmCliente();
        Cliente cliente = new Cliente
        {
            Nombre = collection["nombre"],
            Apellido = collection["apellido"],
            Email = collection["email"]
        };
        objAdmCli.Alta(cliente);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

2. Crearemos la vista para la acción **Create()** haciendo clic con el botón derecho del mouse y seleccionando la opción **Agregar Vista**.
3. En siguiente ventana, ingresamos como plantilla **Create** y la clase de modelo **Cliente**:



Agregar vista

Nombre de vista: Create

Plantilla: Create

Clase de modelo: Cliente (AccesoADatos.Models)

Opciones:

☐ Crear como vista parcial

☐ Hacer referencia a bibliotecas de scripts

☐ Usar página de diseño:

(Dejar en blanco si se define en un archivo _viewstart de Razor)

Agregar Cancelar

```
@model AccesoADatos.Models.Cliente

@{
    Layout = null;
}

<!DOCTYPEhtml>

<html>
<head>
<metaname="viewport"content="width=device-width"/>
<title>Create</title>
<!-- CSS only -->
<linkhref="https://cdn.jsdelivrivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"r
el="stylesheet"integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6D0itfPri4tjfhXaWutUpFmB
p4vmVor"crossorigin="anonymous">
</head>
```

```
...  
<body>  
@using (Html.BeginForm())  
{  
@Html.AntiForgeryToken()  
  
<divclass="form-horizontal">  
<h4>Cliente</h4>  
<hr/>  
@Html.ValidationSummary(true, "", new { @class = "text-danger" })  
<!--  
<divclass="form-group">  
@Html.LabelFor(model =>model.Codigo, htmlAttributes: new { @class = "control-label  
col-md-2" })  
<divclass="col-md-10">  
@Html.EditorFor(model =>model.Codigo, new { htmlAttributes = new { @class = "form-control"  
} })  
@Html.ValidationMessageFor(model =>model.Codigo, "", new { @class = "text-danger" })  
</div>  
</div>  
-->
```



```
<divclass="form-group">
@Html.LabelFor(model =>model.Nombre, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Nombre, new { htmlAttributes = new { @class = "form-control"
} })
@Html.ValidationMessageFor(model =>model.Nombre, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.Apellido, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Apellido, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.Apellido, "", new { @class = "text-danger" })
</div>
</div>
```



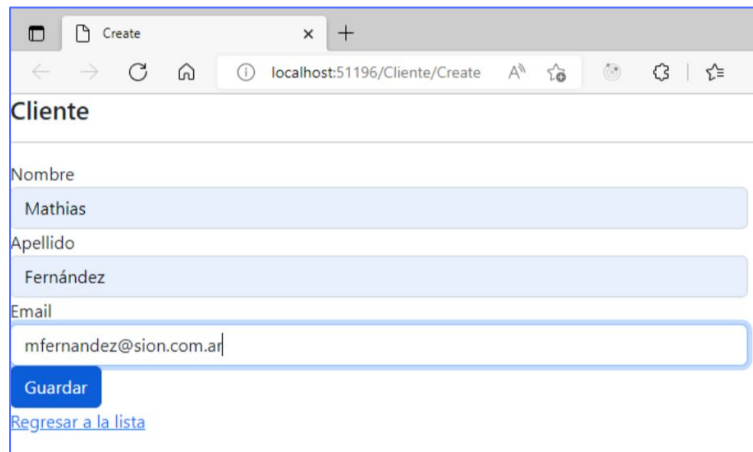
...

```
<divclass="form-group">
@Html.LabelFor(model =>model.Email, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Email, new { htmlAttributes = new { @class = "form-control"
} })
@Html.ValidationMessageFor(model =>model.Email, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
<divclass="col-md-offset-2 col-md-10">
<inputtype="submit"value="Guardar"class="btnbtn-primary"/>
</div>
</div>
</div>
    }
</div>
@Html.ActionLink("Regresar a la lista", "Index")
</div>
</body>
</html>
```



4. Se realizarán algunas modificaciones en el código generado, para que se muestren los mensajes en español. Comentaremos la generación del **input** para el ingreso del código de cliente, ya que al ser en la tabla un campo de tipo **Identity**, será el servidor de base de datos el que generará su valor.
5. Si se ejecuta, se puede observar el formulario HTML que nos permite ingresar los datos de un cliente y luego, mediante la ejecución de la acción **Create**, se guardan los datos en la tabla de la base de datos. Veamos la imagen:

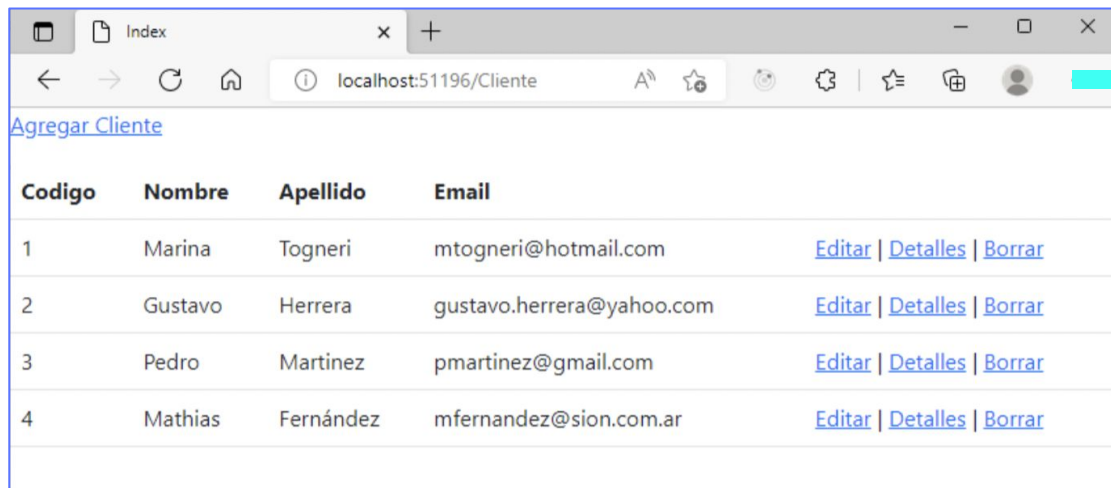


The screenshot shows a web browser window with the address bar displaying 'localhost:51196/Cliente/Create'. The page title is 'Cliente'. The form contains the following fields:

- Nombre**: A text input field containing the value 'Mathias'.
- Apellido**: A text input field containing the value 'Fernández'.
- Email**: A text input field containing the value 'mfernandez@sion.com.ar'.

Below the email field, there is a blue button labeled 'Guardar' and a blue link labeled 'Regresar a la lista'.





The screenshot shows a web browser window with the address bar displaying 'localhost:51196/Cliente'. Below the address bar, there is a link 'Agregar Cliente'. The main content area contains a table with four columns: 'Codigo', 'Nombre', 'Apellido', and 'Email'. Each row represents a client and includes three action links: 'Editar', 'Detalles', and 'Borrar'.

Codigo	Nombre	Apellido	Email	
1	Marina	Togneri	mtogneri@hotmail.com	Editar Detalles Borrar
2	Gustavo	Herrera	gustavo.herrera@yahoo.com	Editar Detalles Borrar
3	Pedro	Martinez	pmartinez@gmail.com	Editar Detalles Borrar
4	Mathias	Fernández	mfernandez@sion.com.ar	Editar Detalles Borrar

Nota: Es importante notar que la acción cuyo nombre es **Create(FormCollectioncollection)** no tiene una vista asociada; si analizamos el código, mediante la llamada a **RedirectToAction(Index)** se ejecuta entonces la acción que muestra el listado de clientes.

Baja de cliente

Se implementará el algoritmo para que se borre un cliente, cuando se haga clic en el vínculo **Borrar**.

1. Deberemos codificar las dos acciones **Delete**:

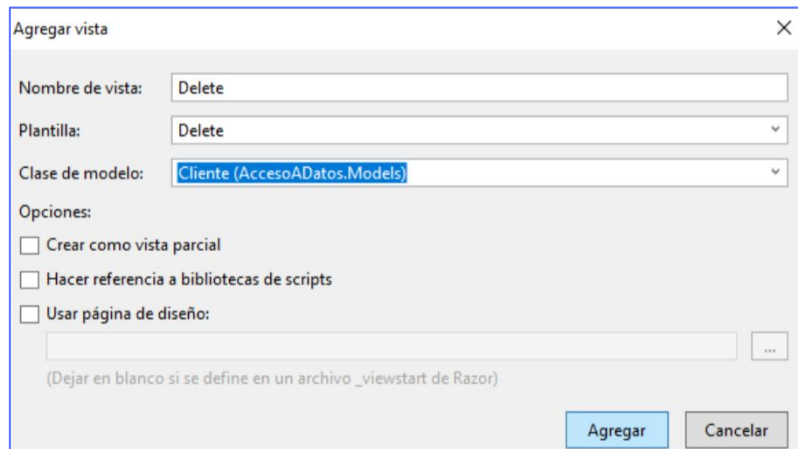
```
// GET: Cliente/Delete/5
public ActionResult Delete(int id)
{
    AdmCliente objAdmCli = new AdmCliente();
    Cliente cliente = objAdmCli.TraerCliente(id);
    return View(cliente);
}
```



...

```
// POST: Cliente/Delete/5
[HttpPost]
public ActionResult Delete(int id, FormCollection collection)
{
    try
    {
        // TODO: Add delete logic here
        AdmCliente objAdmCliente = new AdmCliente();
        objAdmCliente.Borrar(id);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```

2. Generamos la vista para la acción llamada **Delete(int id)**: hacer clic en el botón derecho del mouse y cargar los valores en la ventana, según la imagen de la derecha:



Agregar vista

Nombre de vista: Delete

Plantilla: Delete

Clase de modelo: Cliente (AccesoADatos.Models)

Opciones:

- ☒ Crear como vista parcial
- ☐ Hacer referencia a bibliotecas de scripts
- ☐ Usar página de diseño:

(Dejar en blanco si se define en un archivo _viewstart de Razor)

Agregar Cancelar

3. Se generará automáticamente el archivo **Delete.cshtml** que con algunas modificaciones del idioma en los mensajes, quedará de la siguiente manera:

```
@model AccesoADatos.Models.Cliente
@{
    Layout = null;
}

<!DOCTYPEhtml>

<html>
<head>
<metaname="viewport"content="width=device-width"/>
<title>Delete</title>
<!-- CSS only -->
<linkhref="https://cdn.jsdelivrivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"rel="stylesheet"integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFB�6D0itfPri4tjfhXaWutUpFmBp4vmVor"crossorigin="anonymous">
</head>
```

```
<body>
<h3>Desea borrar este Cliente ?</h3>
<div>
<h4>Cliente</h4>
<hr/>
<dlclass="dl-horizontal">
<dt>
@Html.DisplayNameFor(model =>model.Codigo)
</dt>

<dd>
@Html.DisplayFor(model =>model.Codigo)
</dd>

<dt>
@Html.DisplayNameFor(model =>model.Nombre)
</dt>

<dd>
@Html.DisplayFor(model =>model.Nombre)
</dd>
```

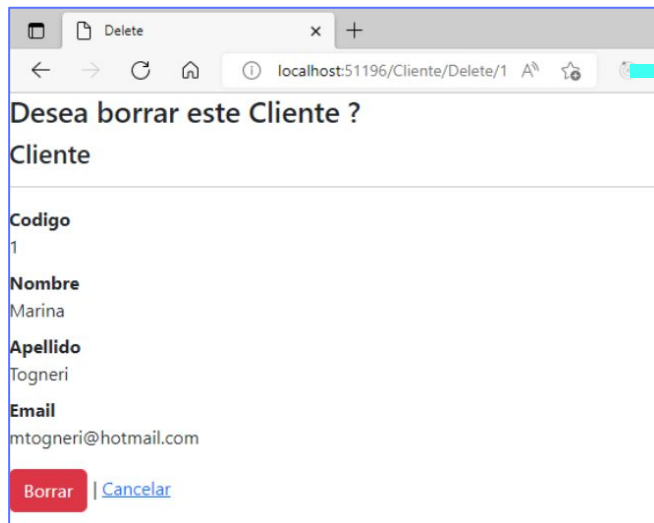
```
...  
<dt>  
@Html.DisplayNameFor(model =>model.Apellido)  
</dt>  
  
<dd>  
@Html.DisplayFor(model =>model.Apellido)  
</dd>  
  
<dt>  
@Html.DisplayNameFor(model =>model.Email)  
</dt>  
  
<dd>  
@Html.DisplayFor(model =>model.Email)  
</dd>  
  
</dl>
```




```
@using (Html.BeginForm()) {  
    @Html.AntiForgeryToken()  
  
    <div class="form-actions no-color">  
        <input type="submit" value="Borrar" class="btn btn-danger"/> |  
        @Html.ActionLink("Cancelar", "Index")  
    </div>  
    }  
</div>  
</body>  
</html>
```



4. Al ejecutar la aplicación y hacer clic en el vínculo, cuyo nombre es **Borrar**, se cargará la siguiente página:



Modificación de cliente

1. Para la modificación o edición de un cliente deberemos implementar el código correspondiente en las acciones del controlador **Edit**.

El código a implementar será:

```
// GET: Cliente/Edit/5
public ActionResult Edit(int id)
{
    AdmCliente objAdmCli = new AdmCliente();
    Cliente cliente = objAdmCli.TraerCliente(id);
    return View(cliente);
}
```

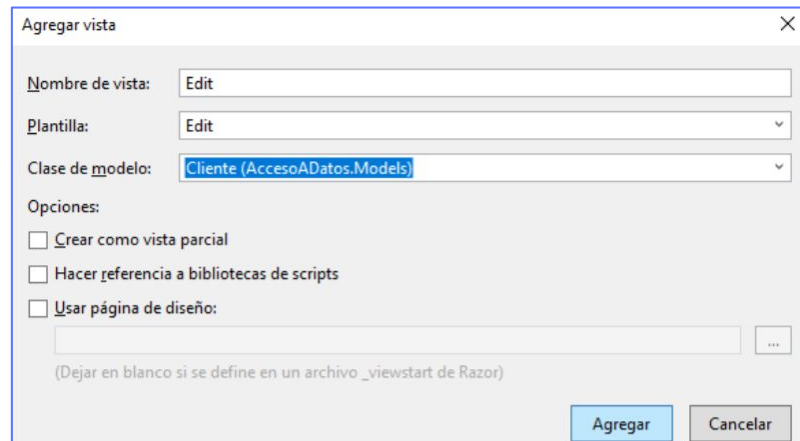


...

```
// POST: Cliente/Edit/5
[HttpPost]
public ActionResult Edit(int id, FormCollection collection)
{
    try
    {
        // TODO: Add update logic here
        AdmCliente objAdmCli = new AdmCliente();
        Cliente cliente = new Cliente
        {
            Codigo = id,
            Nombre = collection["nombre"].ToString(),
            Apellido = collection["apellido"].ToString(),
            Email = collection["email"].ToString(),
        };
        objAdmCli.Modificar(cliente);
        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}
```



- Además, se deberá generar la vista para la acción **Edit(int id)**, como podemos ver en la imagen de la derecha.
- Se generará automáticamente el archivo **Edit.cshtml** que con algunas modificaciones del idioma en los mensajes quedará como veremos en la siguiente pantalla.



Agregar vista

Nombre de vista: Edit

Plantilla: Edit

Clase de modelo: Cliente (AccesoADatos.Models)

Opciones:

☐ Crear como vista parcial

☐ Hacer referencia a bibliotecas de scripts

☐ Usar página de diseño:

(Dejar en blanco si se define en un archivo _viewstart de Razor)

Agregar Cancelar

```
@model AccesoADatos.Models.Cliente
```

```
@{  
Layout = null;  
}
```

```
<!DOCTYPEhtml>
```

```
<html>
```

```
<head>
```

```
<metaname="viewport"content="width=device-width"/>
```

```
<title>Edit</title>
```

```
<!-- CSS only -->
```

```
<linkhref="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0-beta1/dist/css/bootstrap.min.css"rel="stylesheet"integrity="sha384-0evHe/X+R7YkIZDRvuzKMRqM+OrBnVFBL6DOitfPri4tjfHxaWutUpFmBp4vmVor"crossorigin="anonymous">
```

```
</head>
```



```
<body>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <divclass="form-horizontal">
    <h4>Cliente</h4>
    <hr/>
    @Html.ValidationSummary(true, "", new{ @class = "text-danger" })
    <divclass="form-group">
    @Html.LabelFor(model =>model.Codigo, htmlAttributes: new { @class = "control-label
col-md-2" })
    <divclass="col-md-10">
    @Html.EditorFor(model =>model.Codigo, new { htmlAttributes = new { @class =
"form-control", disabled = "disabled", @readonly = "readonly" } })
    @Html.ValidationMessageFor(model =>model.Codigo, "", new { @class = "text-danger"
})
    </div>
    </div>
    </div>
```

...

```
<divclass="form-group">
@Html.LabelFor(model =>model.Nombre, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Nombre, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.Nombre, "", new { @class = "text-danger"
})
</div>
</div>

<divclass="form-group">
@Html.LabelFor(model =>model.Apellido, htmlAttributes: new { @class =
"control-label col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Apellido, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.Apellido, "", new { @class = "text-danger"
})
</div>
</div>
```

...

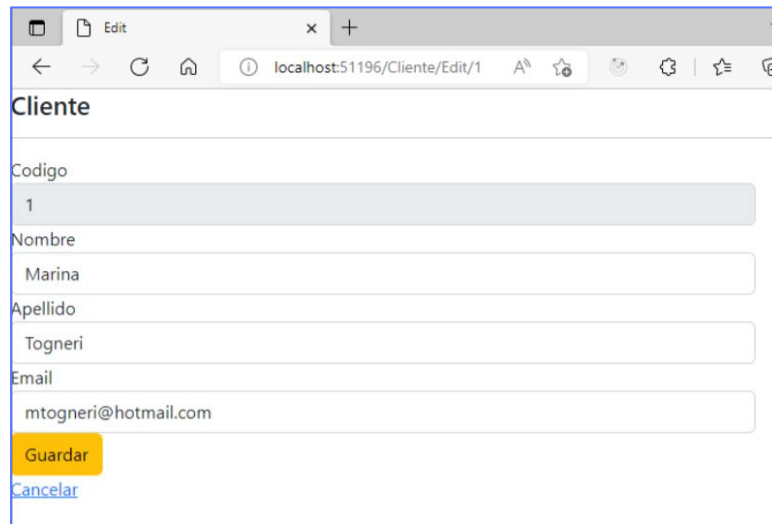


```
<divclass="form-group">
@Html.LabelFor(model =>model.Email, htmlAttributes: new { @class = "control-label
col-md-2" })
<divclass="col-md-10">
@Html.EditorFor(model =>model.Email, new { htmlAttributes = new { @class =
"form-control" } })
@Html.ValidationMessageFor(model =>model.Email, "", new { @class = "text-danger" })
</div>
</div>

<divclass="form-group">
<divclass="col-md-offset-2 col-md-10">
<inputtype="submit"value="Guardar"class="btnbtn-warning"/>
</div>
</div>
</div>
    }
<div>
@Html.ActionLink("Cancelar", "Index")
</div>
</body>
</html>
```



4. Al ejecutar la aplicación y hacer clic en el vínculo **Editar** se mostrará la siguiente página:



The screenshot shows a web browser window with the address bar displaying 'localhost:51196/Cliente/Edit/1'. The page title is 'Cliente'. The form contains the following fields:

- Codigo**: A text input field containing the value '1'.
- Nombre**: A text input field containing the value 'Marina'.
- Apellido**: A text input field containing the value 'Togneri'.
- Email**: A text input field containing the value 'mtogneri@hotmail.com'.

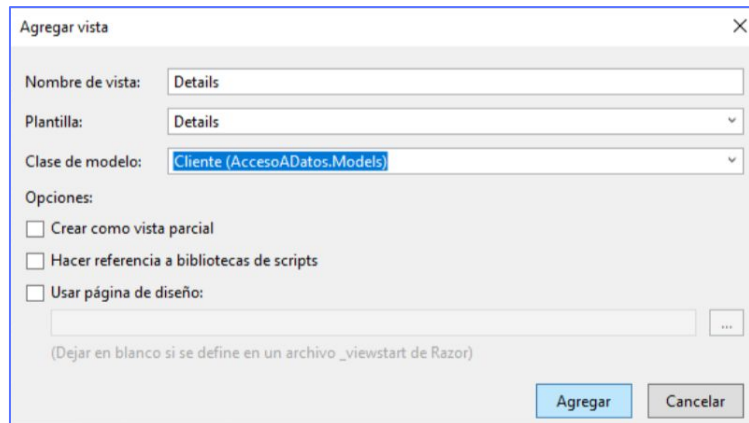
Below the form fields, there are two buttons: a yellow 'Guardar' button and a blue 'Cancelar' button.

Por último, implementaremos el código para el momento en que se haga clic en el vínculo de **Detalle** donde deberán mostrarse los datos de un cliente en particular.

1. Codificaremos la acción **Details(int id)** del controlador, de la siguiente manera:

```
// GET: Cliente/Details/5
public ActionResult Details(int id)
{
    AdmCliente objAdmCli = new AdmCliente();
    Cliente cliente = objAdmCli.TraerCliente(id);
    return View(cliente);
}
```

2. Además, generaremos la vista para la acción **Details(int id)**, como se ve a continuación:



Agregar vista

Nombre de vista:

Plantilla:

Clase de modelo:

Opciones:

☐ Crear como vista parcial

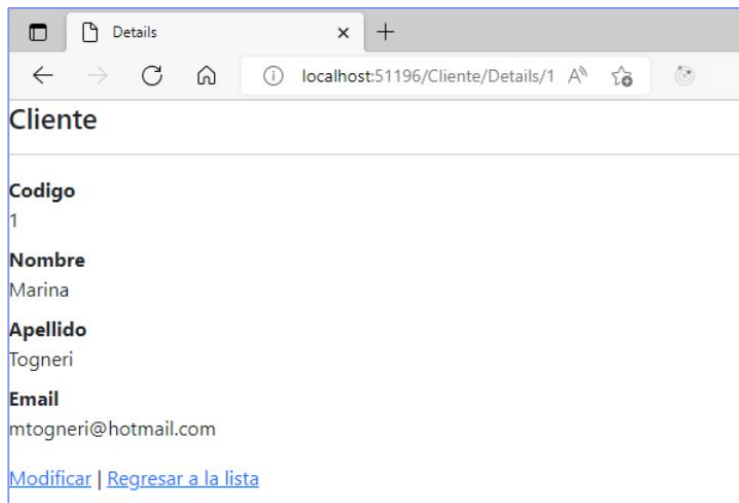
☐ Hacer referencia a bibliotecas de scripts

☐ Usar página de diseño:

...

(Dejar en blanco si se define en un archivo _viewstart de Razor)

3. Se generará automáticamente el archivo llamado ***Details.cshtml***.
4. Al ejecutar la aplicación y seleccionar el hipervínculo ***Detalle*** se obtiene la siguiente página:



Nota: La solución completa del código desarrollado en el módulo, está en la sección de **Descargas**.



**¡Sigamos
trabajando!**