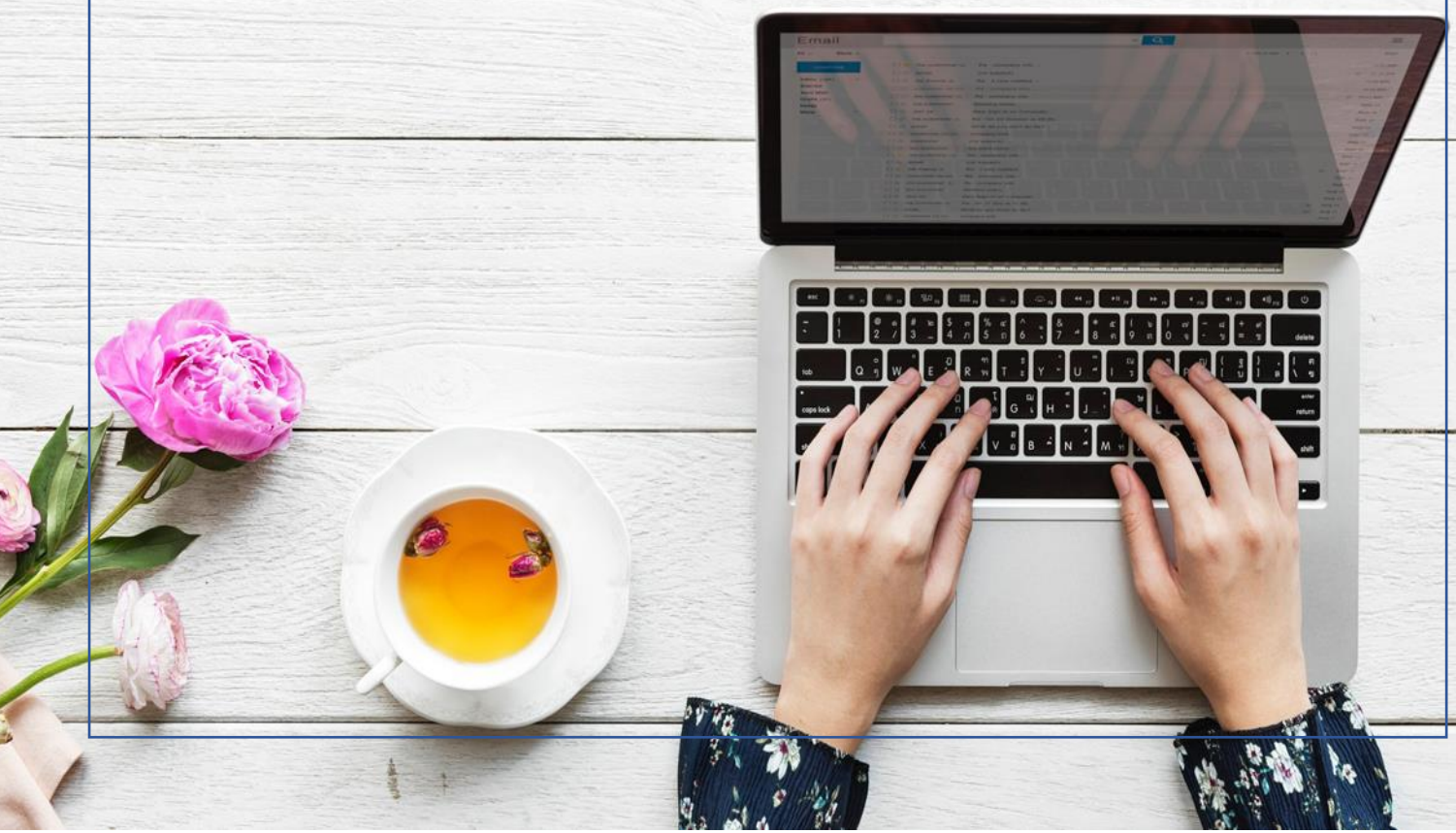


Laura Feü

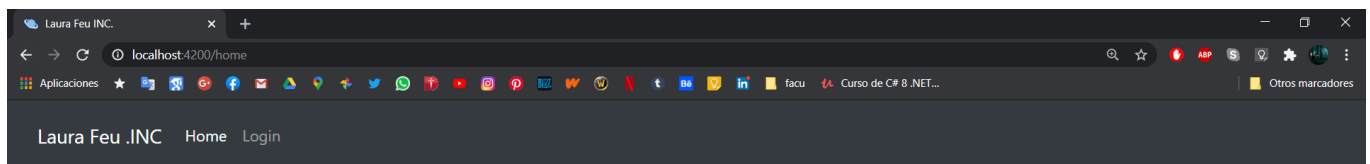


Implementación de una WebAPI con .NET y Angular



INTRODUCCIÓN

A continuación se presentan los pasos que se llevaron a cabo para crear una ABM-WebApi Utilizando las tecnologías .NET y Angular. Se describe el proceso, los programas utilizados y los problemas surgidos a medida que se iba desarrollando el proyecto así mismo como sus soluciones.



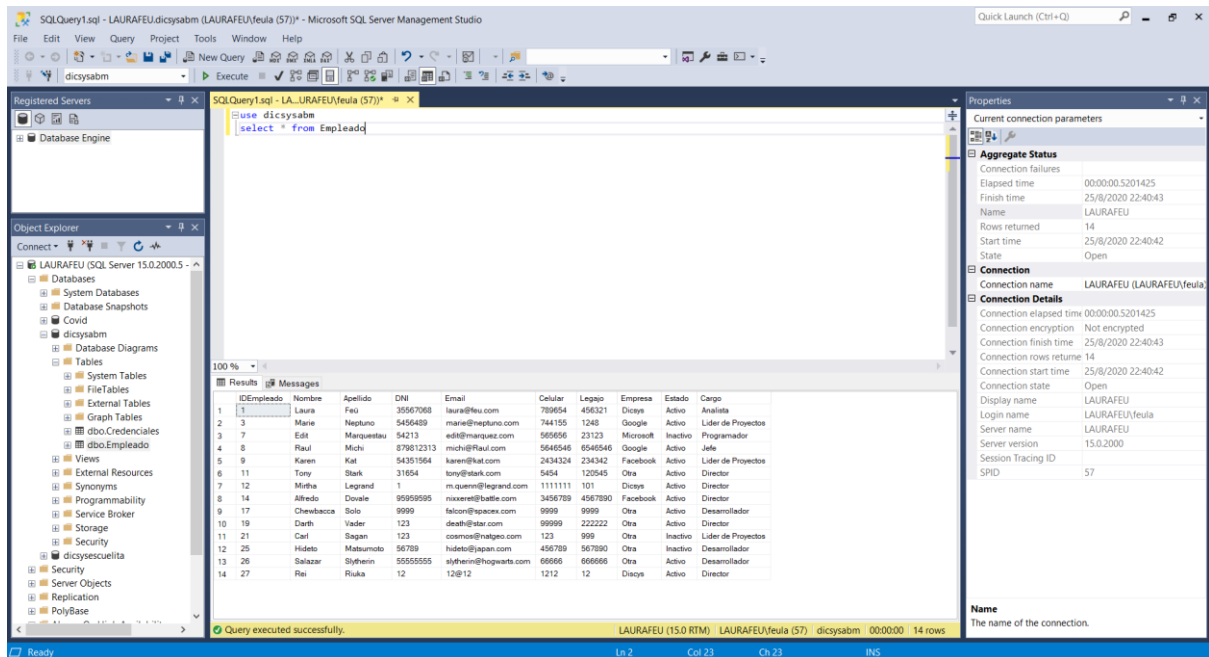
Bienvenido a Laura Feu .INC ¡Regístrate!

Usuario	Password
<input type="text"/>	<input type="password"/>
<input type="button" value="Regístrate"/>	

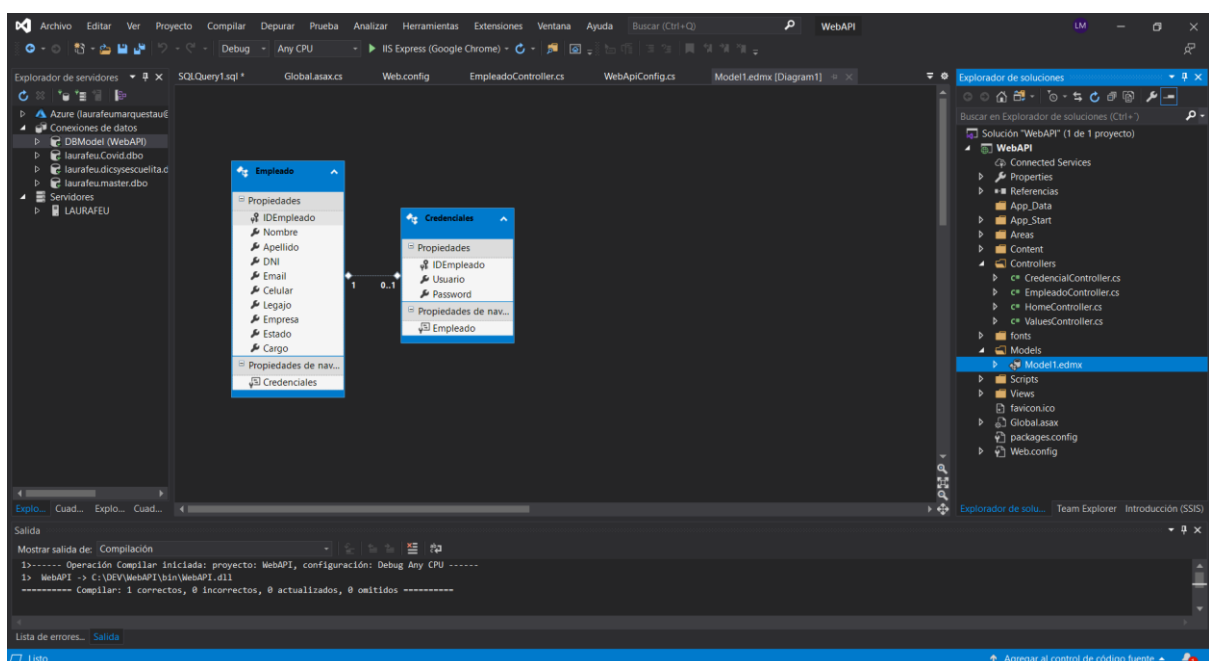
DESCRIPCION DEL MODELO

Utilicé DatabaseFirst como modo de trabajo con el objetivo de que sea el Entity Framework el que se encargue de crear las clases necesarias para trabajar con los datos.

La base de datos fue creada desde en Microsoft SQL Server, donde se crearon dos tablas, una donde contiene todos los datos de los empleados y otra donde contiene los Usuarios y Contraseñas de los mismos.

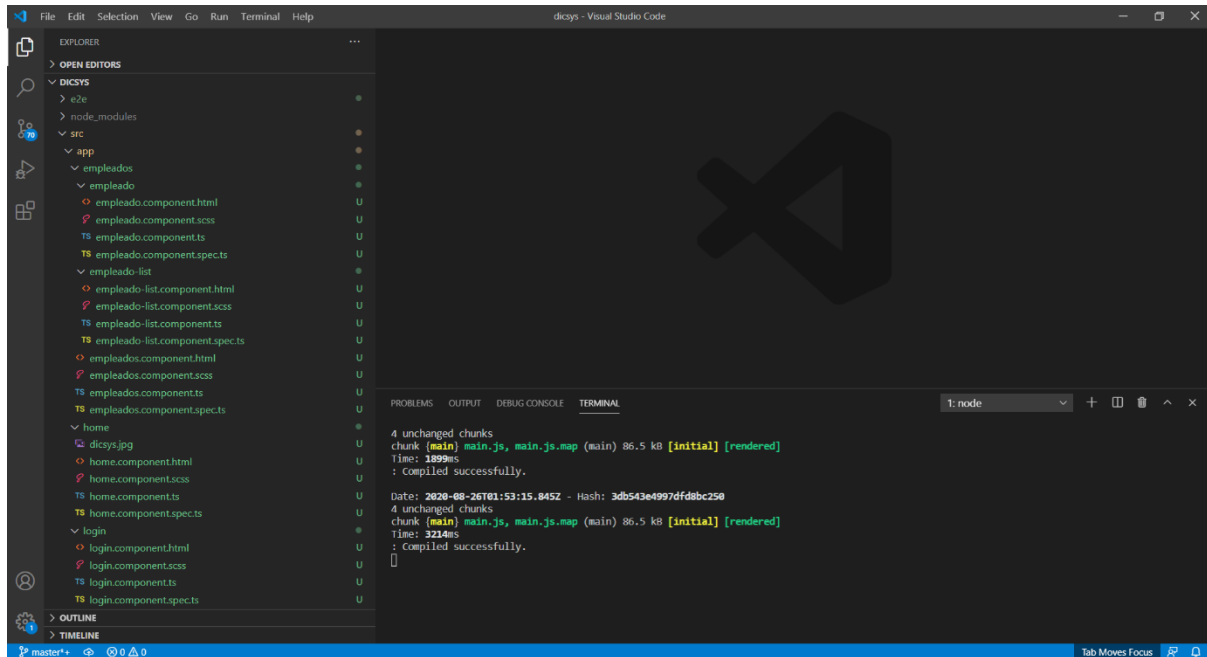


Una vez realizada la base de datos, en Visual Studio creamos un proyecto nuevo WebApi en el Entity Framework. Una vez generado el proyecto desde la carpeta Model conectamos a nuestra base de datos, como se puede ver a continuación:



Para proceder a realizar la parte del FrontEnd decidí utilizar Angular. Para esto realicé los siguientes pasos:

1. Se instaló Visual Studio Code
2. Se instaló Node.js.
3. Se instaló Angular 10.0.11 a través de la Terminal del VS Code.
4. Se crea el proyecto en Angular con el comando “ng new dicsys”



Algunos de los comandos que me sirvieron para la creación de mi proyecto fueron:

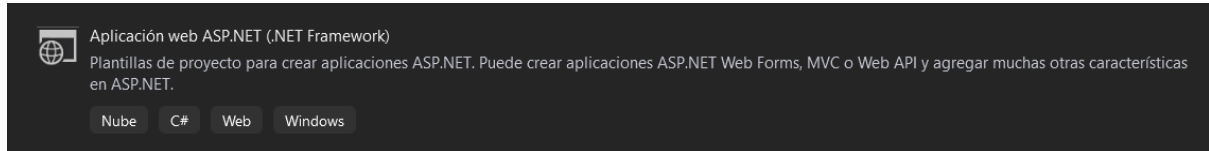
ng g cl empleado --type=model → Para crear modelos

ng g s empleado → Para crear servicios

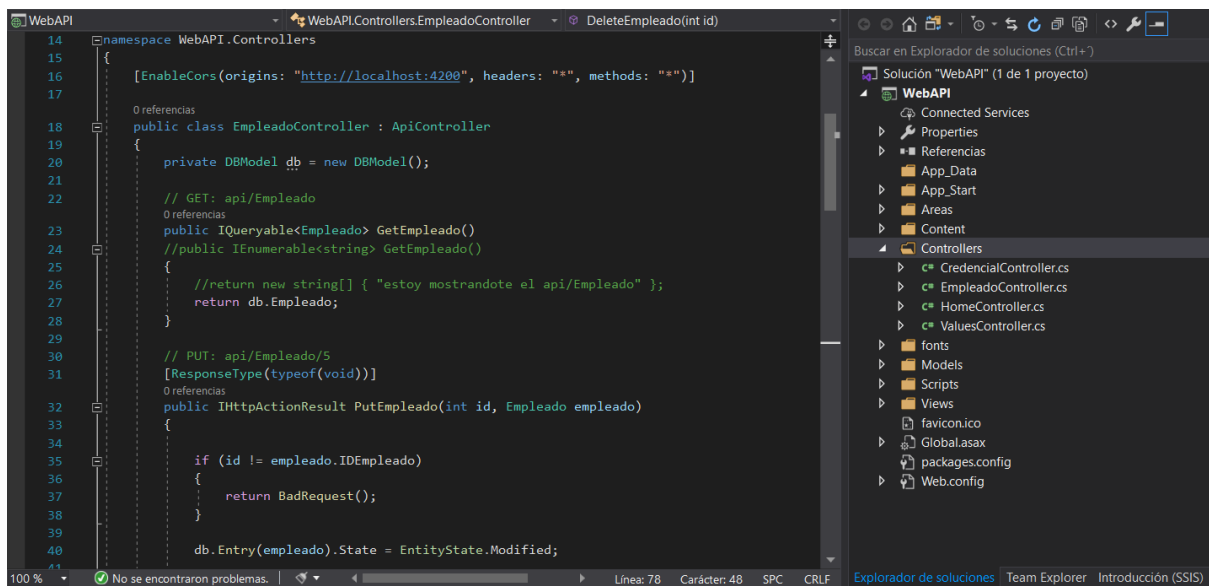
ng g c empleado-list → Para crear componentes

BACK-END

Como había mencionado previamente, mi backend lo generé con .NET desde Microsoft Visual Studio, para esto generé un nuevo proyecto del tipo “Aplicación WEB ASP.NET (NET Framework)” y lo ubiqué en la ruta C:\DEV



Luego procedí a crear el modelo y finalmente los controladores que me permiten conectarme con el front-end.



FRONT-END

El front-end se realizó en Angular, siguiendo la siguiente estructura:

```
■ src
+----- ■ app
|
|   +----- ■ home
|   |   |-- home.component.ts|.html|.scss
|   |   ■ login
|   |   |-- login.component.ts|.html|.scss
|   |   ■ empleados
|   |   |-- empleados.component.ts|.html|.scss
|   |   + -- ■ empleado
|   |   |   |-- empleado.component.ts|.html|.scss
|   |   |   ■ empleado-list
|   |   |   |-- empleado-list.component.ts|.html|.scss
|   |   ■ shared
|   |   |-- empleado.service.ts
|   |   |-- empleado.model.ts
|   |   |-- login.service.ts
|   |   |-- login.model.ts
|   |--app.module.ts
|-- index.html
```

PROBLEMAS DETECTADOS Y SUS SOLUCIONES

1. Problema: CORS

Solución: Agregar en el Web.Config lo siguiente después de <system.webServer>

```
<httpProtocol>
  <customHeaders>
    <add name="Access-Control-Allow-Origin" value="*" />
    <add name="Access-Control-Allow-Headers" value="Content-Type" />
    <add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE, OPTIONS" />
  </customHeaders>
</httpProtocol>
```

Luego seguí obteniendo errores de CORS y me fijé en varios foros (stackoverflow) donde algunos aconsejaban sacar las siguientes líneas:

```
<add name="Access-Control-Allow-Origin" value="*" />
<add name="Access-Control-Allow-Headers" value="Content-Type" />
```

El Web.Config me quedó así entonces después del <system.webServer>

```
<httpProtocol>
  <customHeaders>
    <add name="Access-Control-Allow-Methods" value="GET, POST, PUT, DELETE, OPTIONS" />
  </customHeaders>
</httpProtocol>
```

2. Problema: "el-tipo-objectcontent1-no-pudo-serializar-el-cuerpo-de-respuesta-para-el-tipo-de-contenido"

Solución: En el Global.asax para evitar referencias circulares en la base

```
//Evito las referencias circulares al trabajar con Entity Framework
GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings.ReferenceLoopHandling =
Newtonsoft.Json.ReferenceLoopHandling.Serialize;
GlobalConfiguration.Configuration.Formatters.JsonFormatter.SerializerSettings.PreserveReferencesHandling =
Newtonsoft.Json.PreserveReferencesHandling.Objects;
```

```
//Elimino que el sistema devuelva en XML, sólo trabajaremos con JSON
GlobalConfiguration.Configuration.Formatters.Remove(GlobalConfiguration.Configuration.Formatters.XmlFormatter);
```

Referencia: <http://www.resuelvetusproblemas.com/el-tipo-objectcontent1-no-pudo-serializar-el-cuerpo-de-respuesta-para-el-tipo-de-contenido/>

3. Problema: Error: The 'Access-Control-Allow-Origin' header contains multiple values 'http://localhost:4200, *', but only one is allowed.

Solución: <https://stackoverflow.com/questions/22343384/the-access-control-allow-origin-header-contains-multiple-values>

Elimino línea de Web.Config

```
<add name="Access-Control-Allow-Origin" value="*" />
```

4. Problema: "Error converting value {null} to type 'System.Int32'. Path 'IDEmpleado', line 1, position 18."

Solución/es: Cuando quiero agregar un nuevo registro "empleado" luego de haber agregado funciones de modificación y borrado, obtengo el error indicado arriba

Si saco del empleado.component.html la línea: `<input type="hidden" name="IDEmpleado" #IDEmpleado="ngModel" [(ngModel)]="service.formData.IDEmpleado">`

Si funciona el agregado, pero no modifica (si borra) por la lógica que tengo en el typescript

Notas: en la mayoría de casos de error los debuggeaba fijándome en la consola del explorador y con ayuda de Fiddler Everywhere que me ayudaba a ubicar en qué parte estaba el error y podía ir a buscar en google algunas soluciones.

The screenshot shows a web browser displaying a login form for 'Laura Feu .INC' with fields for 'Usuario' and 'Password'. The browser's developer console shows multiple 'Failed to load resource' errors with status 502 (Fiddler - Connection Failed). An orange arrow points from these errors to the Fiddler Everywhere interface. In Fiddler, the 'Live Traffic' tab is active, showing a list of requests. Request 47 is highlighted, showing a GET request to `/sockjs-node/828/01rbq1xx/websocket` with a status of 101. The 'Inspectors' pane on the right shows the 'Headers' tab for this request, displaying a 'Sec-WebSocket-Extensions' header with the value 'permessage-deflate; client_max_v'.

#	Result	Protocol	Host	URL	Method	Body
35	304	HTTP	localhost:4200	/runtime.js	GET	0
36	304	HTTP	localhost:4200	/polyfills.js	GET	0
37	304	HTTP	localhost:4200	/styles.js	GET	0
38	304	HTTP	localhost:4200	/vendor.js	GET	0
39	200	HTTP	localhost:4200	/main.js	GET	80,046
40	304	HTTP	localhost:4200	/runtime.js.map	GET	0
41	304	HTTP	localhost:4200	/polyfills.js.map	GET	0
42	304	HTTP	localhost:4200	/styles.js.map	GET	0
43	304	HTTP	localhost:4200	/vendor.js.map	GET	0
44	200	HTTP	localhost:4200	/main.js.map	GET	31,388
45	200	HTTP	localhost:4200	/sockjs-node/info?t=1598398401072	GET	90
46	200	HTTP	localhost:4200	/sockjs-node/828/01rbq1xx/websocket	CONNECT	0
47	101	HTTP	localhost:4200	/sockjs-node/828/01rbq1xx/websocket	GET	0
48	200	HTTP	localhost:44316	localhost:44316	CONNECT	0
49	200	HTTP	settings-win.data.microsoft.com:443	settings-win.data.microsoft.com:443	CONNECT	0
50	200	HTTP	clients4.google.com:443	clients4.google.com:443	CONNECT	0
51	200	HTTP	localhost:44316	localhost:44316	CONNECT	0
52	200	HTTP	oauthaccountmanager.googleapis.com:443	oauthaccountmanager.googleapis.com:443	CONNECT	0
53	304	HTTP	ctdl.windowsupdate.com:443	ctdl.windowsupdate.com:443	GET	0
54	200	HTTP	localhost:44316	localhost:44316	CONNECT	0



CONCLUSIÓN

Fue una gran experiencia crear esta WEBApi, aunque aún tiene cosas por mejorar, por ejemplo, me gustaría agregarle en el front-end una plantilla para hacerla visualmente más atractiva probar varias opciones extras que ofrece Bootstrap y además de arreglar los errores que aún quedaron por resolver.

Me llevo un gran aprendizaje de esta experiencia.

Gracias.