Tipo : Guía de Enunciado
Capítulo : ASP NET Core
Duración : 60 minutos

I. OBJETIVO

Publicación de una Web Api de ASP NET Core en IIS y acceder desde nuestra aplicación React.

II. REQUISITOS

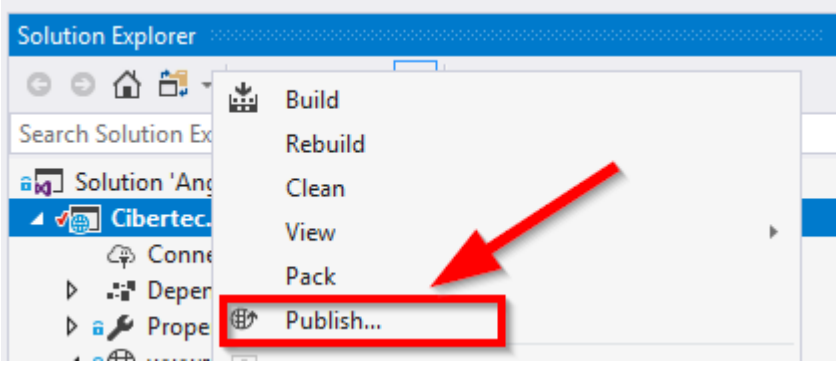
Los siguientes elementos de software son necesarios para la realización del laboratorio:

- Windows 10 (como mínimo Windows 8)
- Visual Studio 2017 como mínimo.
- Visual Studio Code

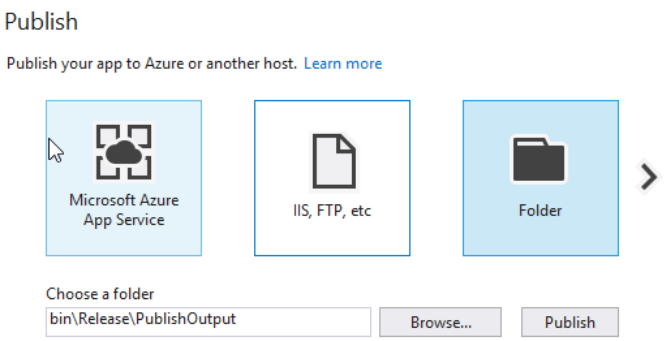
III. EJECUCIÓN DEL LABORATORIO

1. Abrir la solución "**Cibertec.NetCore.sln**" y ubicarse en el proyecto Cibertec.WebApi.

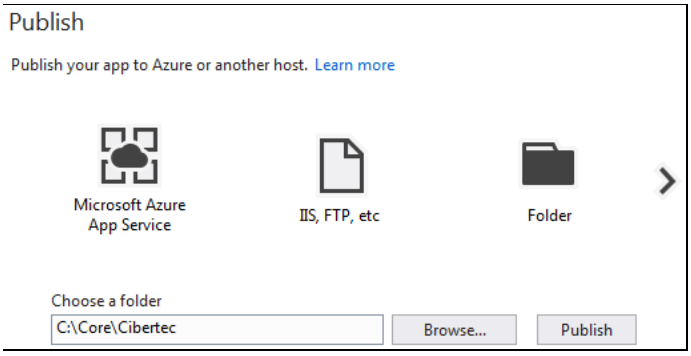
2. Hacer clic derecho en el proyecto "**Cibertec.Angular**" y hacer clic en "**Publish**"



3. Seleccionar la opción Folder:



4. Y hacer clic en Browse y como ruta indicar "**C:\Core\Cibertec**".

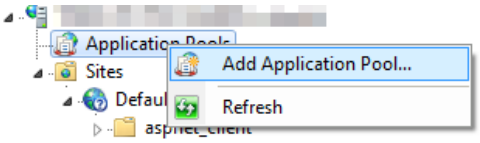


5. Hacer clic en Publish.

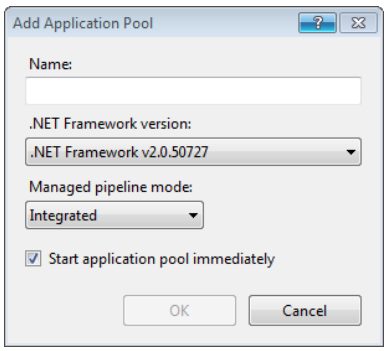
```
Cibertec.Angular -> C:\Juvega\github\CibertecWeb100\CibertecWeb100\Angular\Ci
Web App was published successfully file:///C:/Web/Angular

========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Publish: 1 succeeded, 0 failed, 0 skipped ==========
```

6. Abrimos la consola de IIS:

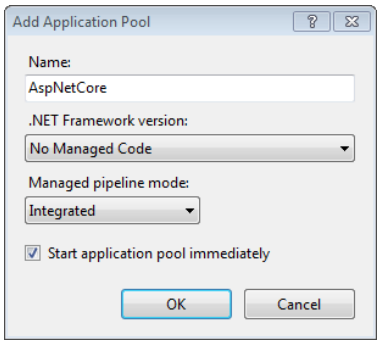
   (1) En la sección Applicacton Pool hacemos clic derecho para agregar un nuevo pool:



   (2) Visualizaremos la siguiente ventana:

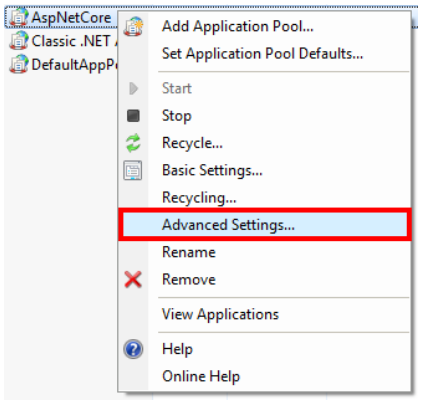


   (3) Como nombre le ponemos "**AspNetCore**"

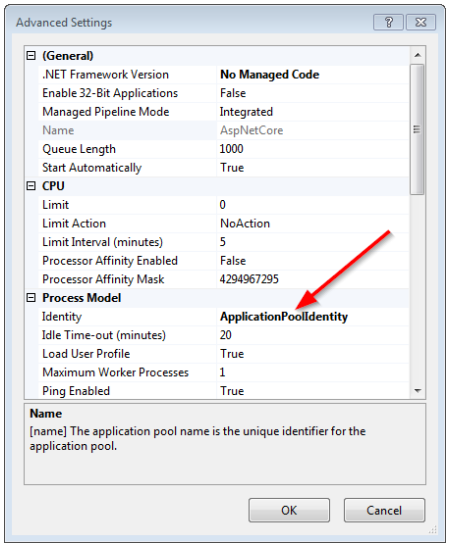


   (4) Y clic en ***Ok***

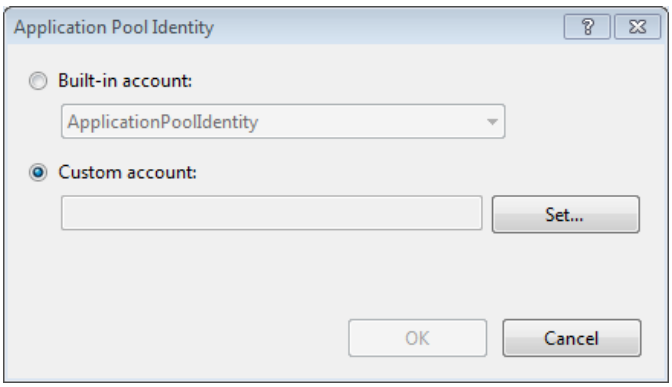
(5) Procedemos a editar las configuraciones avanzadas del pool.



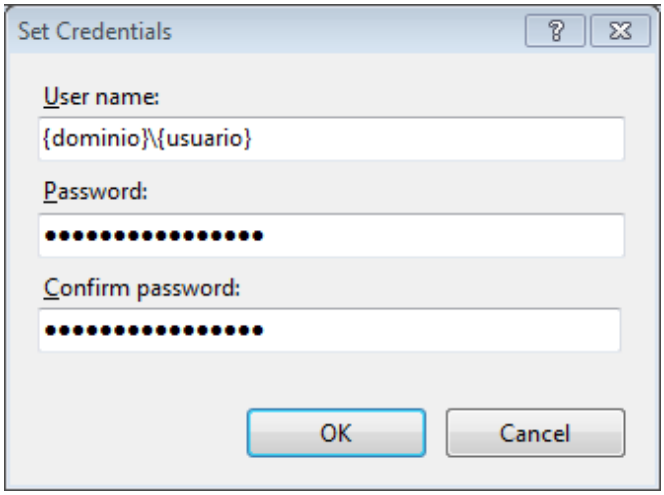
(6) Seleccionamos la propiedad Identity



(7) Seleccionamos **Custom account**



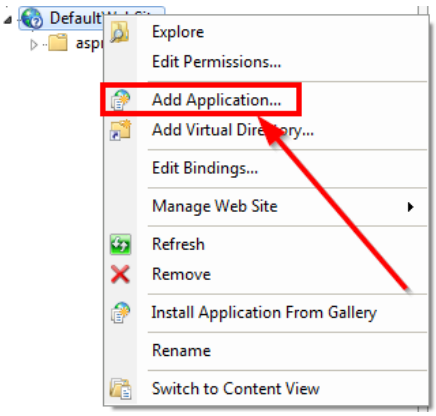
(8) Clic en Set y agregamos un usuario que tenga acceso a la base de datos.



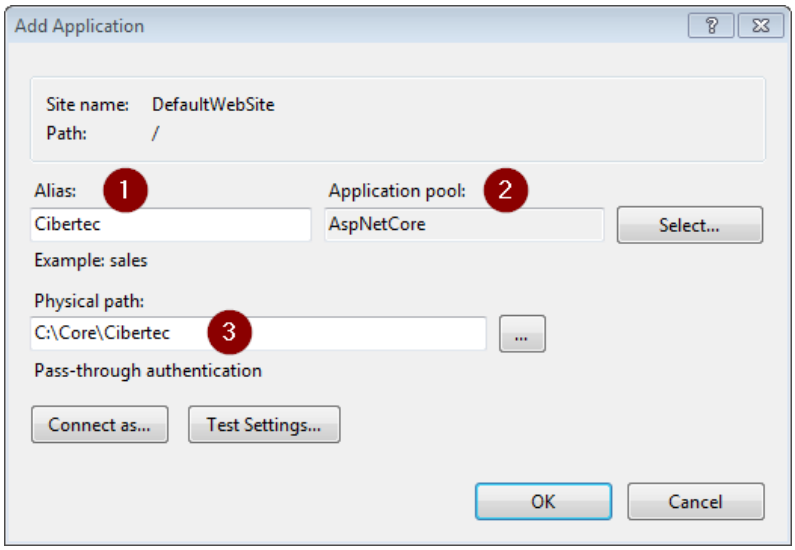
(9) Recuerda que tiene que ser un usuario valido para nuestro motor de base de datos.

(10) Hacemos clic en el botón Ok de las tres ventanas.

7. Desde el **DefaultWebSite** agregamos una nueva aplicación web:



8. Una vez seleccionada, configurar la ventana como se muestra a continuación:



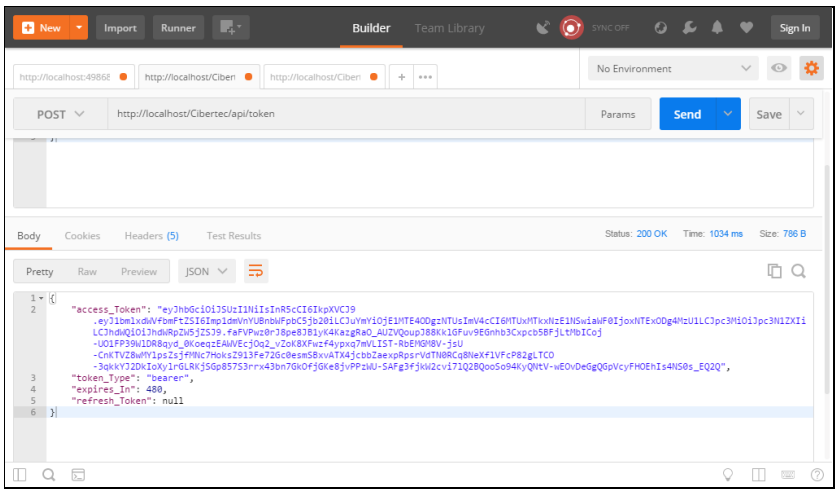
(1) Alias “**Cibertec**”
(2) Application Pool: “**AspNetCore**”. El pool que acabamos de crear.
(3) La ruta: “**C:\Core\Cibertec**”, que viene a ser la ruta donde publicamos nuestra WebApi.

9. Ahora procedemos a testear desde “**Postman**” la utenticación:

   - URL: http://localhost/Cibertec/api/token
   - Json:

```
{
    "grant_type":"password",
    "email":"{tu_usuario}",
    "password":"{tu_password}"
}
```

   Resultado esperado:



10. Validad la autenticación, procedemos a abrir nuestra aplicación React con Visual Studio Code.

11. Nos ubicamos en la carpeta “**src\actions**” y procedemos a editar el fichero “**creators.tsx**” y editamos la variable “**apiUrl**” con el siguiente valor:

    - http://localhost/Cibertec/api (Ver Imagen)

```
You, a few seconds ago | 2 authors (Julio Velarde and others)
1  import * as types from './types';
2  import axios from 'axios';
3  import { Dispatch } from "redux";
4  import * as URLSearchParams from 'url-search-params'
5
6  let apiUrl = 'http://localhost/Cibertec/api';
7
```

12. En el mismo archivo modificamos la data que se envía a nuestra Web Api como se muestra en la imagen:

    Remplazamos:

```
let params = new URLSearchParams();
params.append('grant_type', 'password');
params.append('username', email);
params.append('password', password);
```

Con:

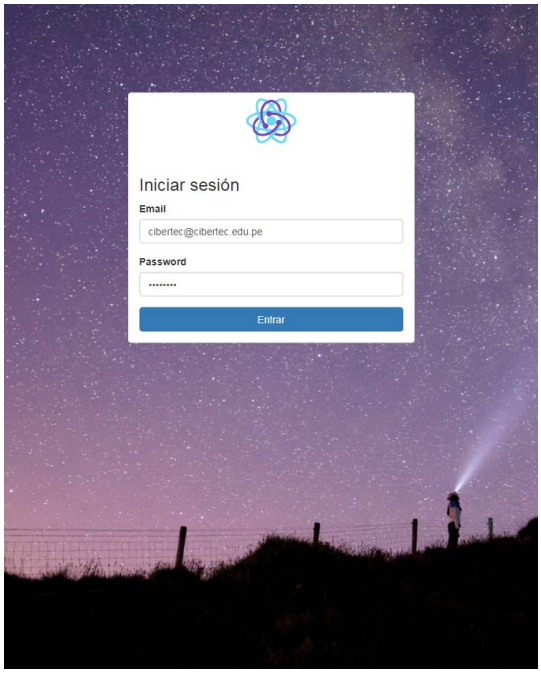
```
let params = {
      "grant_type":"password",
      "email":email,
      "password":password
};
```

```
let apiUrl = 'http://localhost/Cibertec/api';

export function loginUser(email: string, password: string) {
    return function (dispatch: any) {
        let params = {
            "grant_type":"password",
            "email":email,
            "password":password
        };
        axios.post(`${apiUrl}/token` , params)
            .then(response => {
                dispatch({
                    type: types.GOT_TOKEN,
                    token: response.data.access_Token
                });
            })
            .catch((error) => { console.log(error) });
    }
}
```

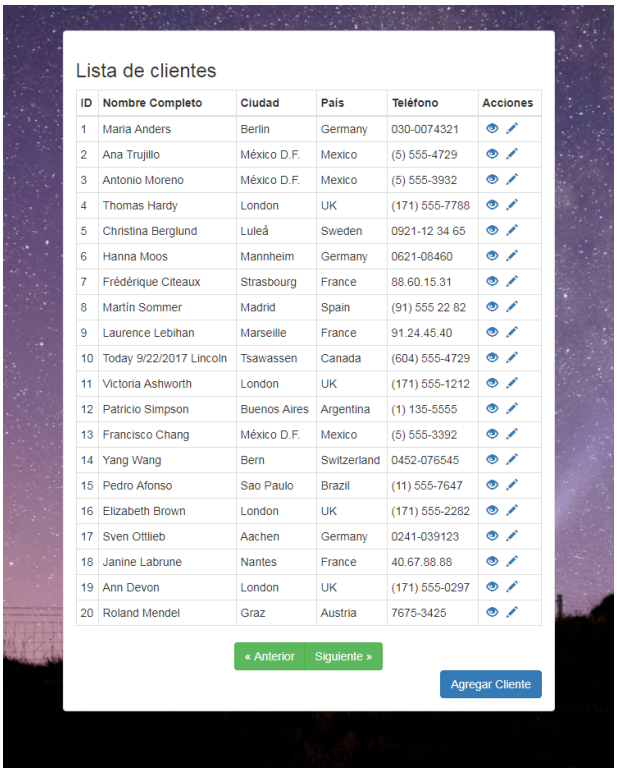
13. Ejecutar el comando webpack en el terminal integrado de Visual Studio Code.

```
$ webpack

[at-loader] Using typescript@2.1.6 from typescript and "tsconfig.json" from C:\Repos\WebDeveloper\Cibertec-React/tsconfig.json.


[at-loader] Checking started in a separate process...

[at-loader] Ok, 0.184 sec.
Hash: 2611dc8c12d65e5b2803
Version: webpack 2.2.1
Time: 4455ms
       Asset     Size  Chunks                    Chunk Names
    bundle.js  355 kB       0  [emitted]  [big]  main
bundle.js.map  398 kB       0  [emitted]         main
   [6] ./~/redux/es/index.js 1.08 kB {0} [built]
   [8] ./~/react-router/es/index.js 1.46 kB {0} [built]
  [11] ./~/react-redux/es/index.js 194 bytes {0} [built]
  [17] (webpack)/buildin/global.js 509 bytes {0} [built]
  [55] ./src/routes.tsx 2.34 kB {0} [built]
  [56] ./~/url-search-params-polyfill/index.js 8.18 kB {0} [built]
  [64] ./src/defaultParams.tsx 474 bytes {0} [built]
  [67] ./src/reducers/rootReducer.tsx 390 bytes {0} [built]
 [111] ./~/react-router/es/IndexLink.js 597 bytes {0} [built]
 [112] ./~/react-router/es/IndexRedirect.js 1.36 kB {0} [built]
 [113] ./~/react-router/es/IndexRoute.js 1.38 kB {0} [built]
 [114] ./~/react-router/es/Route.js 1.38 kB {0} [built]
 [115] ./~/react-router/es/Router.js 5.3 kB {0} [built]
 [127] ./~/redux-thunk/lib/index.js 529 bytes {0} [built]
 [137] ./src/index.tsx 270 bytes {0} [built]
    + 123 hidden modules
```

14. Abrir el index.html en el browser y validar su normal funcionamiento.



15. Ingreso a la aplicación:



IV. EVALUACIÓN

1. ¿Qué diferencia a .NET Core del .NET Framework?
2. ¿Por qué usar Asp Net Core?
3. ¿Cuáles son los beneficios de una Web Api con ASP NET Core?
4. ¿Dónde está el web.config?