Introducción al Desarrollo de Páginas Web

Taller de Angular

Gestión de clientes

Realizado por:

201123291 - Arlem Gabriel Brenes Aguilar

201235853 - Alejandro José Schmidt Ramírez

2016009280- Jean Anthony Vega Díaz

Contenido

Introducción

El presente taller tiene como finalidad conocer con un ejemplo básico las ventajas que nos ofrece este framework al momento de realizar aplicaciones web.

Dentro de este taller haremos uso de funcionalidades que caracterizan a angular tales como uso de módulos, componentes, rutas y manejo de variables directamente del código en typescript.

Problema a resolver

Se debe crear un pequeño formulario para el registro de clientes, y un lugar donde visualizarlos a través del uso de componentes y servicios para el manejo de datos.

Requisitos previos

Para poder realizar este taller es necesario tener instalado el ambiente de desarrollo de angular.

En el siguiente enlace se puede encontrar un instructivo de instalación https://jeanvegad.github.io/portafolio/investigacion/InstructivoInstalacionAngular.pdf

Taller

Generar un nuevo proyecto

1. Para iniciar necesitamos crear un nuevo proyecto en angular, para ello es necesario que creemos una carpeta para almacenar los archivos.
2. Posterior a esto abrimos Visual Studio Code (editor recomendado) y arrastramos la carpeta al editor.
3. Damos clic derecho en el proyecto y abrimos la terminal del proyecto. (ver ilustración 2)

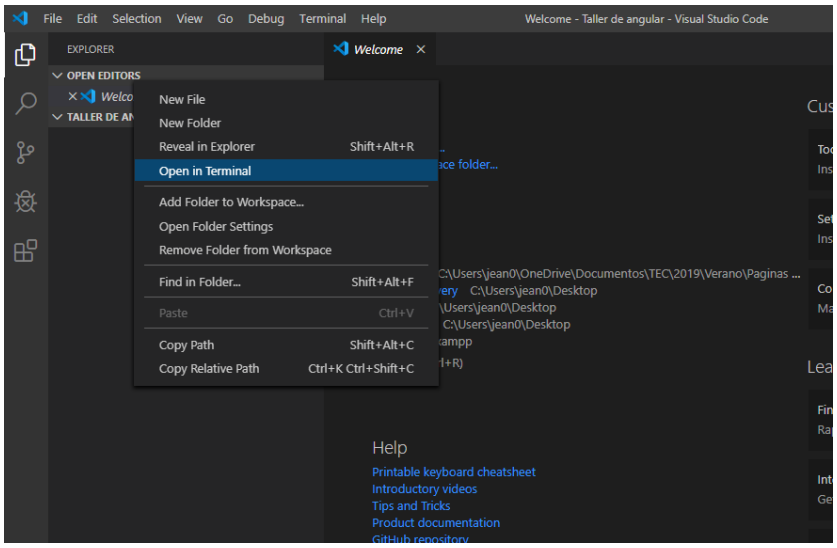

Ilustración 1

4. En la consola ejecutamos el siguiente comando. (ver ilustración 2)

> “ng new gestion_usuarios”

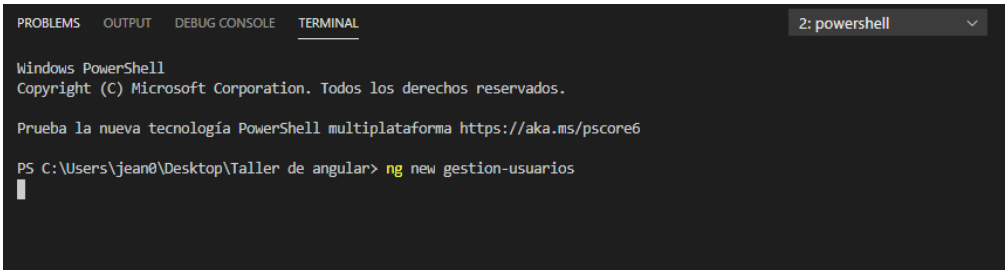

Ilustración 2

5. Angular nos preguntara si deseamos que se incluya la configuración de rutas en nuestro proyecto. (ver ilustración 3)
6. Para cuestiones de nuestro taller seleccionamos la opción de “y” (si)

```
PS C:\Users\jean0\Desktop\Taller de angular> ng new gestion-usuarios
? Would you like to add Angular routing? (y/N)
```

Ilustración 3

7. Ahora seleccionamos nuestro estilo de diseño favorito. (ver ilustración 4)
8. En nuestro caso seleccionamos CSS

```
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
  Stylus [ http://stylus-lang.com                                         ]
```

Ilustración 4

9. Esperamos a que se cree el proyecto. (puede tardar un poco)
10. Una vez finalizado el proceso nos creara la siguiente estructura. (ver ilustración 5)

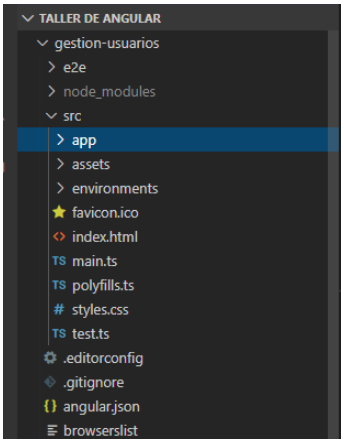

Ilustración 5

Ejecutar en el proyecto

Con angular podemos generar un servidor local para poder correr probar los cambios que realizamos en nuestros proyectos .

1. En la terminal accedemos a la carpeta app
2. Para ello en la consola del proyecto ejecutamos el siguiente comando. (ver ilustración 6)

> ng serve”

```
PS C:\Users\jean0\Desktop\Taller de angular\gestion-usuarios\src\app> ng serve
Browserslist: caniuse-lite is outdated. Please run next command `npm update`
10% building 3/3 modules 0 activei [wds]: Project is running at http://localhost:4200/webpack-dev-server/
i [wds]: webpack output is served from /
i [wds]: 404s will fallback to //index.html

chunk {main} main.js, main.js.map (main) 49.5 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 264 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.73 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 4.08 MB [initial] [rendered]
Date: 2020-01-06T03:15:55.909Z - Hash: d646f2bc37f49712bd09 - Time: 10683ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
i [wdm]: Compiled successfully.
```

Ilustración 6

3. Accedemos en el navegador a la dirección http://localhost:4200/ (ver ilustración 7)

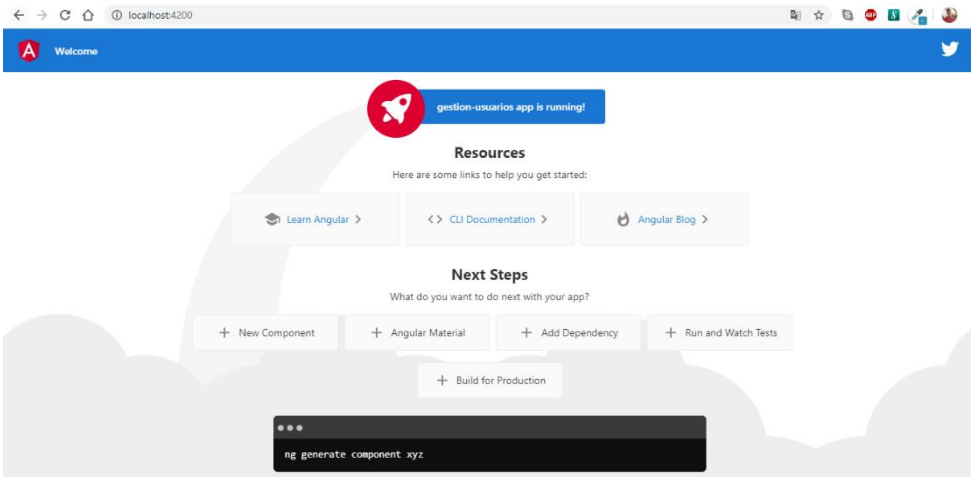

Ilustración 7

Librerías necesarias

En caso de que el proyecto requiera la importación de librerías, estas deben ser llamadas en un archivo denominado app.module.ts.

En nuestro caso, para poder hacer binding de la información desde los formularios es necesario importar la librería correspondiente.

1. Abrimos el archivo app.module.ts
2. Incluimos la siguiente librería.

```
import {FormsModule}   from '@angular/forms';
```

3. En la sección de “imports” incluimos el “FormsModule”. (ver ilustración 8)

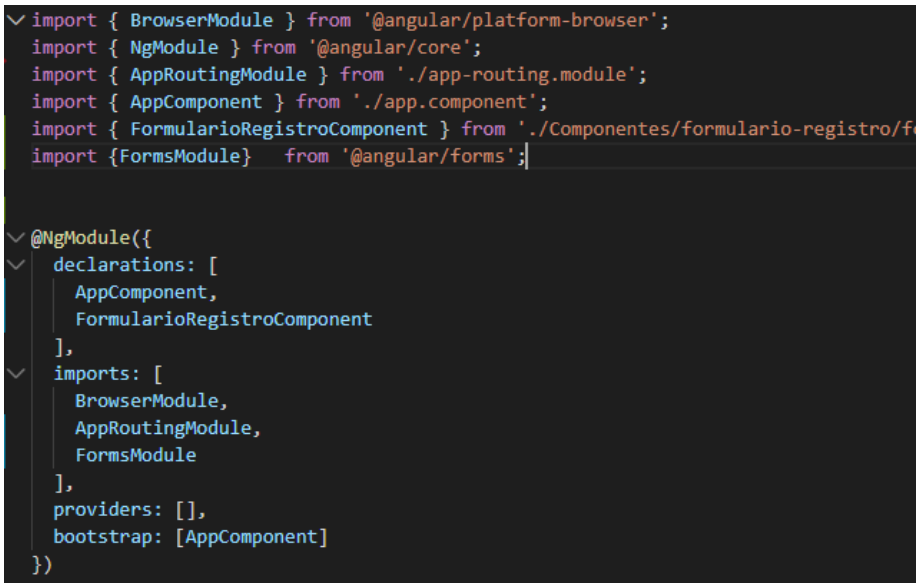

Ilustración 8

Crear un modelo de datos

Para administrar la información de los datos ingresados, lo que podemos hacer es crear una interfaz.

1. Dentro del módulo “app” generado por el proyecto creamos un archivo de tipo TypeScript llamado “usuarios.model.ts” donde generaremos el modelo de los datos a utilizar. (ver ilustración 9)

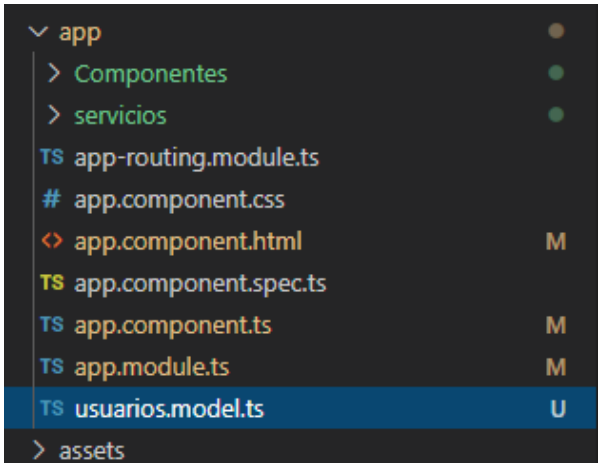

Ilustración 9

2. Incluimos el siguiente código en el archivo

```
export interface Usuario {
    nombre:string;
    apellidos:string;
    cedula:string;
    correo:string;
    telefono:string;
    edad:string;
}
```

Crear un servicio

Los servicios en angular, nos permite concentrar todas las tareas en un solo lugar.

1. Creamos una carpeta de “servicios” y en ella abrimos la terminal y ejecutamos el siguiente comando. (ver ilustración 10)

> “ng g s userService”

```
PS C:\Users\jean0\Desktop\Taller de angular\gestion-usuarios\src\app\servicios> ng g s userService
CREATE src/app/servicios/user-service.service.spec.ts (359 bytes)
CREATE src/app/servicios/user-service.service.ts (140 bytes)
```

Ilustración 10

2. Importamos el modelo de datos en el servicio

```
import {Usuario} from '../usuarios.model';
```

3. Definimos los métodos y variables necesarias para el servicio.

```
export class UserServiceService {

  private usuarios: Usuario[];

  constructor() {
    this.usuarios = [];
  }
  getUsuarios() {
    return this.usuarios;
  }
  agregarUsuarios(user: Usuario) {
    this.usuarios.push(user);
  }

  nuevoUsuario(): Usuario {
    return {
      nombre:'',
      apellidos:'',
      cedula:'',
      correo:'',
      telefono:'',
      edad:''
    };
  }
}
```

4. Abrimos el archivo app.module.ts
5. importamos el servicio creado.

```
import {UserServiceService} from './servicios/user-service.service';
```

6. En el array de providers incluimos también el servicio

```
providers: [UserServiceService]
```

Crear un componente

Formulario

Los componentes en angular son trozos de código en los cuales se pueden reutilizar mediante una etiqueta de tipo HTML.

1. Creamos una carpeta donde queremos ubicar los componentes de la aplicación. (ver ilustración 11)

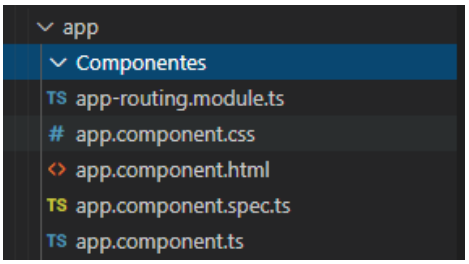

Ilustración 11

2. Abrimos la consola de comandos dentro de la carpeta creada y ejecutamos el siguiente comando. (ver ilustración 12)

> “ng g c formulario_registro”

```
PS C:\Users\jean0\Desktop\Taller de angular\gestion-usuarios\src\app\Componentes> ng g c formulario_registro
CREATE src/app/Componentes/formulario-registro/formulario-registro.component.html (34 bytes)
CREATE src/app/Componentes/formulario-registro/formulario-registro.component.spec.ts (713 bytes)
CREATE src/app/Componentes/formulario-registro/formulario-registro.component.ts (320 bytes)
CREATE src/app/Componentes/formulario-registro/formulario-registro.component.css (0 bytes)
UPDATE src/app/app.module.ts (537 bytes)
```

Ilustración 12

3. Una vez creado el componente se genera la siguiente estructura. (ver ilustración 13)

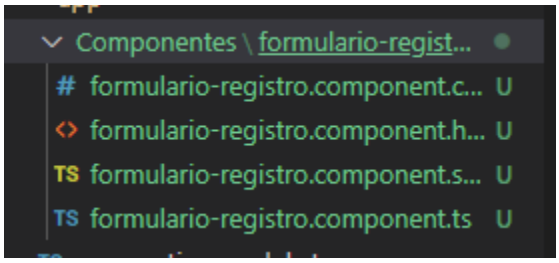

Ilustración 13

4. Abrimos el archivo formulario-registro.component.ts
5. En el importamos el servicio y el modelo de datos.

```
import {UserServiceService} from '../../servicios/user-service.service';
import {Usuario} from '../../usuarios.model';
```

6. En el constructor inyectamos el servicio creado.

```
constructor(private userService: UserServiceService) { }
```

7. Creamos las variables y funciones necesarias, el archivo quedaría de la siguiente manera.

```
export class FormularioRegistroComponent implements OnInit {


  usuario: Usuario;
  constructor(private userService: UserServiceService) { }

  ngOnInit() {
    this.usuario = this.userService.nuevoUsuario();

  }

  nuevoUsuario(): void {
    this.userService.agregarUsuarios(this.usuario);
    this.usuario = this.userService.nuevoUsuario();
  }

}
```

8. En el formulario-registro.component.html, escribimos las líneas de HTML del componente, en nuestro caso es el formulario de registro.

```
<form id="form_usuarios">
    <h1>Registro de usuarios</h1>
    <h2>Nombre: </h2>
    <input type="text" class="textfield" name="nombre" [(ngModel)]="usuario.nombre"/>
    <h2>Apellidos: </h2>
    <input type="text" class="textfield" name="var" [(ngModel)]="usuario.apellidos"/>
    <h2>Cedula: </h2>
    <input type="text" class="textfield" name="var" [(ngModel)]="usuario.cedula"/>
    <h2>Correo: </h2>
    <input type="text" class="textfield" name="var" [(ngModel)]="usuario.correo"/>
    <h2>Teléfono: </h2>
    <input type="text" class="textfield" name="telefono" [(ngModel)]="usuario.telefono"/>
    <h2>Edad: </h2>
```

```
    <input type="text" class="textfield" name="edad" [(ngModel)]="usuario.edad"/>
    <br>
    <br>
    <button type="submit" form="form_usuarios" value="Submit" (click)="this.nuevoUsuario()">Agregar usuario</button>
</form>
```

9. Aplicamos el CSS en el documento formulario-registro.component.css

```
input {
    display: inline-block;
    width: 100%;
    padding-top: 3px;
    padding-bottom: 3px;
    border: 0.5px;
    border-style: solid;
    border-radius: 3px;
    opacity: 0.85;
    font-family: 'poppins_regular', 'arial', sans-serif;
    font-size: 12px;
    line-height: 20px;
    text-decoration: none;
    /*colores*/
    background-color:white;
    color:#646464;
    border-color: #646464;
}



h1{
    color: #646464;
    font-family: 'arial', sans-serif;
    font-size: 20px;
    text-align: center;
}



h2{
    color: #646464;
    font-family: 'arial', sans-serif;
    font-size: 15px;
}
```

```
button{
    background-color: #646464;
    color: white;
    cursor: pointer;
    width: 100%;
    padding: 12px;
    border: none;
    border-radius: 4px;
    opacity: 0.85;
    display: inline-block;
    font-size: 17px;
    line-height: 15px;
    text-decoration: none;
    font-family: 'arial', sans-serif;
}
```

Listado

Para poder visualizar los datos generados, deberemos crear un nuevo componente.

1. Dentro de la carpeta creada anteriormente de componentes, abrimos la terminal y ejecutamos el comando.

> “ng g c listado_usuarios”

2. Abrimos el archivo listado-usuarios.component.ts
3. En el importamos el servicio y el modelo de datos.

```
import {UserServiceService} from '../../servicios/user-service.service';
import {Usuario} from '../../usuarios.model';
```

4. En el constructor inyectamos el servicio creado.

```
constructor(private userService: UserServiceService) { }
```

5. Creamos las variables y funciones necesarias, el archivo quedaría de la siguiente manera.

```
export class ListadoUsuariosComponent implements OnInit {

  usuarios: Usuario[];

  constructor(private userService: UserServiceService) { }

  ngOnInit() {
    this.usuarios = this.userService.getUsuarios();
  }

}
```

6. Abrimos el archivo “listado-usuarios.component.html” y pegamos el siguiente código

```
<h1>Listado de usuarios</h1>
    <div *ngIf="! usuarios.length">
      <h2>No hay usuarios por el momento</h2></div>
    <div>
    <div *ngIf=" usuarios.length">
        <article>
            <span><b>Nombre</b></span>
            <span><b>Apellidos</b></span>
            <span><b>Cédula</b></span>
            <span><b>Correo</b></span>
            <span><b>Teléfono</b></span>
            <span><b>Edad</b></span>
        </article>
    </div>
    <article *ngFor="let usuario of usuarios">
      <span>{{usuario.nombre}}</span>
      <span>{{usuario.apellidos}}</span>
      <span>{{usuario.cedula}}</span>
      <span>{{usuario.correo}}</span>
      <span>{{usuario.telefono}}</span>
      <span>{{usuario.edad}}</span>
    </article>
  </div>
```

7. Abrimos el archivo “listadousuarios.component.css” y pegamos el siguiente código

```
input {
    display: inline-block;
    width: 100%;
    padding-top: 3px;
    padding-bottom: 3px;
    border: 0.5px;
    border-style: solid;
    border-radius: 3px;
    opacity: 0.85;
    font-family: 'poppins_regular', 'arial', sans-serif;
    font-size: 12px;
    line-height: 20px;
    text-decoration: none;
    /*colores*/
    background-color:white;
    color:#646464;
    border-color: #646464;
}

h1{
    color: #646464;
    font-family: 'arial', sans-serif;
    font-size: 20px;
    text-align: center;
}

h2{
    color: #646464;
    font-family: 'arial', sans-serif;
    font-size: 15px;
}

button{
    background-color: #646464;
    color: white;
    cursor: pointer;
    width: 100%;
    padding: 12px;
    border: none;
    border-radius: 4px;
```

```
    opacity: 0.85;
    display: inline-block;
    font-size: 17px;
    line-height: 15px;
    text-decoration: none;
    font-family: 'arial', sans-serif;
}

article {
    display: flex;
    border-bottom: 1px solid #ddd;
    padding: 10px;
    font-size: 0.9em;
  }
  span {
    display: inline-block;
    width: 22%;
    margin-right: 2%;
  }
```

Mostrar componentes

Para finalizar se deben mostrar los componentes creado anteriormente mediante el selector que se encuentra en el archivo de extensión .ts (ver ilustración 14)

```
@Component({
  selector: 'app-formulario-registro',
  templateUrl: './formulario-registro.compone
```

Ilustración 14

1. Abrimos el archivo app.component.html y pegamos el siguiente código

```
<body>
  <div class="row">
    <div class="col-lg-4">
      <div style="width: 70%; margin: auto;">
        <app-formulario-registro></app-formulario-registro>
      </div>
    </div>
    <div class="col-lg-8">
      <app-listado-usuarios></app-listado-usuarios>
    </div>
  </div>
</body>
```

Nota: este ejemplo esta haciendo uso de columnas de Bootstrap, por lo cual se debe importar la librería correspondiente. Para ello, basta con abrir el archivo “index.html” y en el <head> pegar el siguiente código.

```
 <!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```