Taller angular

Para instalar angular en node comenzaremos utilizando

npm install -g @angular/cli

Procederemos a ubicarnos en nuestra carpeta donde deseamos crear el proyecto y ejecutamos el comando y entramos a la carpeta creada

ng new workshop

Nos preguntara los siguiente

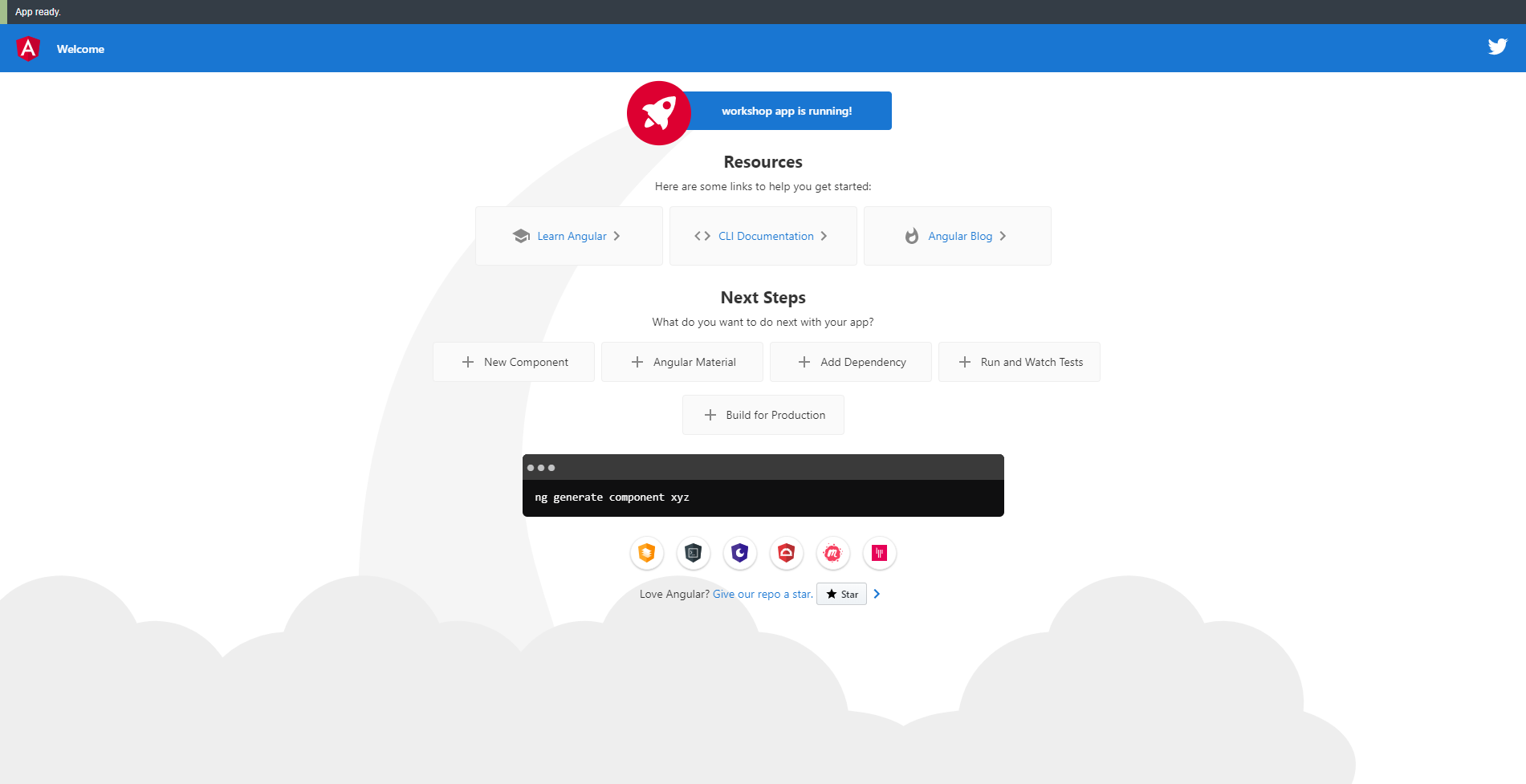
? Would you like to add Angular routing? Yes

? Which stylesheet format would you like to use? SCSS [ https://sass-lang.com/documentation/syntax#scss]

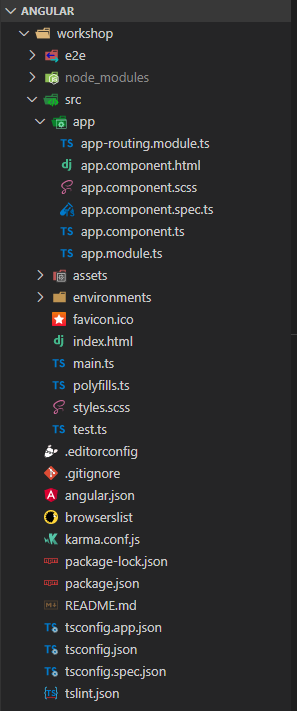
cd workshop

Procederemos a ejecutar el proyecto

ng serve



Estructura del proyecto



Src: es la carpeta principal del proyecto donde se encuentra nuestro código.

App: es la ubicación donde crearemos nuestros componentes.

Assets: lugar donde ubicamos todo aquello que no es código.

Envioments: lugar donde se encuentra la configuración del proyecto.

Index.html: es el archivo principal donde se ejecuta todo el proyecto.

Styles.scss: es la hoja de estilos general del proyecto.

Procederemos a instalar material angular al proyecto ejecutando el comando

ng add @angular/material

Instalaremos Pink/Blue Grey

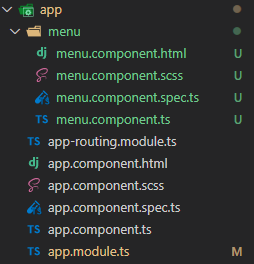
? Set up HammerJS for gesture recognition? Yes

? Set up browser animations for Angular Material? Yes

Esto realizado procedemos a crear un nuevo componente utilizando el comando

ng generate component menu

Donde menú vendría siendo el nombre de nuestro componente



Podemos observar que al momento de generar un nuevo componente se generar 4 archivos

El archivo html es donde van las etiquetas

El archivo scss es donde van los estilos del componente

El archivo typscript donde uno mantiene la lógica del componente y el otro es un archivo para pruebas.

El componente cuenta con la siguiente estructura

menu.component.ts

import { Component, OnInit } from '@angular/core'; //bibliotecas para que nuestro componente funcione

@Component({

  selector: 'app-menu',

  templateUrl: './menu.component.html',

  styleUrls: ['./menu.component.scss']

}) //cargamos el template y los estilos

export class MenuComponent implements OnInit {

  constructor() { } //constructor del componente

  ngOnInit() {

  } //Actividades que se ejecutaran cuando se inicialice el componente

}

Procederemos a agregar nuestro nuevo componente al proyecto basta con agregar una etiqueta

app.component.html

<router-outlet></router-outlet>

<app-menu></app-menu>

Donde app indicamos que es un componente y menú es el nombre del componente

Procederemos a agregar un toolbar añadiendo lo siguiente

app.module.ts

import {MatToolbarModule} from '@angular/material/toolbar';

…

@NgModule ({....

  imports: [...,

  MatToolbarModule,

…]

})

Y añadimos

menu/menu.component.html

<mat-toolbar color="primary">

  <mat-toolbar-row>

    <span>workshop Toolbar</span>

  </mat-toolbar-row>

</mat-toolbar>

Ahora crearemos un nuevo componente el cual llamaremos content

ng generate component content

Y lo añadimos a la vista

app.component.html

<router-outlet></router-outlet>

<app-menu></app-menu>

<app-content></app-content>

Operaciones

Declaramos la variable title en el archivo content.component.ts

export class ContentComponent implements OnInit {

  title = 'my content';

Y sustituimos

content.component.html

<p>content works!</p>

Por

<p>{{title}}</p>

Podremos observar como cambia el contenido al valor de la variable title

Ahora añadimos el modulo de forms

app.module.ts

import { FormsModule } from '@angular/forms;

…

@NgModule ({....

  imports: [...,

  FormsModule,

…]

})

content.component.html

<input type="text" [(ngModel)]="title">

Esto nos permite modificar el valor de title por medio de un input

Condicionales content.component.html

<div \*ngIf=" title === 'my content'">

    Esto solo se ve cuando es my content

</div>

<div \*ngIf=" title !== 'my content'">

    Esto solo se ve cuando no es my content

</div>

Un switch corresponde de la siguiente manera

<div [ngSwitch]="title">

    <p \*ngSwitchCase="'enrique'">este es enrique</p>

    <p \*ngSwitchCase="'julian'">este es julian</p>

    <p \*ngSwitchCase="'camilo'">este es camilo</p>

    <p \*ngSwitchDefault > No hay match </p>

  </div>

Agregamos la variable count y la inicializamos en 0 y creamos la función incrementMyCount

content.component.ts

  title = 'my content';

  count = 0;

  ...

  incrementMyCount(){

    this.count++;

  }

Posteriormente ingresamos lo siguiente para añadir a count

Condicionales content.component.html

{{count}}

<button (click)="incrementMyCount()" >Add count</button>

La sentencia For nos permite iterar sobre un arreglo

content.component.ts

 items = ['hola', 'mi', 'nombre', 'es'];

Condicionales content.component.html

<ul>

    <li \*ngFor="let item of items">

      {{item}}

    </li>

</ul>

Otra forma de realizar iteraciones es de la siguiente manera

Condicionales content.component.html

<ul>

  <li \*ngIf="items.length === 0"> La lista esta vacia</li>

  <li \*ngFor="let item of items; index as i">

    {{i}}.-{{item}}

  </li>

</ul>

También podemos agregar y eliminar valores a un arreglo de datos ademas de pasar el indice como parámetro.

Ahora creamos las siguientes dos funciones

content.component.ts

  addItem() {

    this.items.push(this.title);

  }

  deleteItem(index: number) {

    this.items.splice(index, 1);

  }

Condicionales content.component.html

<p>Añadir item</p>

<button (click)="addItem()" >Add item</button>

<ul>

    <li \*ngIf="items.length === 0"> La lista esta vacia</li>

    <li \*ngFor="let item of items; index as i">

        {{i}}.-{{item}} <button (click)="deleteItem(i)">delete</button>

    </li>

</ul>

Ciclo de vida de un modulo



content.component.ts

import { Component, OnInit, SimpleChanges, ChangeDetectionStrategy, DoCheck, OnChanges, OnDestroy } from '@angular/core';

...

export class ContentComponent implements OnInit, DoCheck, OnDestroy {

  ...

  ngOnInit() {

    console.log('ON INIT');

  }

  ngDoCheck(){

    console.log('DO CHECK')

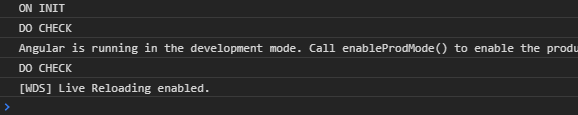
  }

  ngOnDestroy() {

    console.log('ON DESTROY');

  }

A continuación en la con solo podremos observar lo siguiente



Crear el layout

ng generate component header

ng generate component footer

ng generate component layout

Ahora que tenemos esto nos vamos al archivo

app-routing.module.ts

import { NgModule } from '@angular/core';

import { Routes, RouterModule } from '@angular/router';

import { LayoutComponent } from './layout/layout.component';

const routes: Routes = [

  {

    path: '',

    component: LayoutComponent

  },

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

Despues de ello añadimos lo siguiente

layout.component.html

<app-header></app-header>

<div class="container">

    <app-content></app-content>

</div>

<app-footer></app-footer>

Y en el archivo

header.component.html

<app-menu></app-menu>

app.component.html

<router-outlet></router-outlet>

Como el header y el footer es algo que se estará utilizando constantemente en nuestro proyecto esto lo usaremos como un componente recurrente por lo cual a dicho componente le crearemos hijos

ng generate component form

app-routing.module.ts

import { ContentComponent } from './content/content.component';

import { FormComponent } from './form/form.component';

const routes: Routes = [

  {

    path: '',

    component: LayoutComponent,

    children: [

      {

        path: '',

        component: ContentComponent,

      },

      {

        path: 'form',

        component: FormComponent,

      },

    ],

  },

];

layout.component.html

<app-header></app-header>

<div class="container">

    <router-outlet> </router-outlet>

</div>

<app-footer></app-footer>

menu.component.html

Ahora realizaremos un mini formulario en el cual agregaremos nuestra comida favorita

form.component.ts

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-form',

  templateUrl: './form.component.html',

  styleUrls: ['./form.component.scss']

})

export class FormComponent implements OnInit {

  food = 'pizza';

  reason = 'delicious';

  favoriteFood = [

    {

      food: 'lasagna',

      reason: 'is easy to prepare',

    },

  ];

  constructor() { }

  ngOnInit() {

  }

  addFavoriteFood() {

    const object: any = {

      food: this.food,

      reason: this.reason

    };

    this.favoriteFood.push(object);

  }

}

form.component.html

<h1>What is your favorite food</h1>

<p>Food name</p>

<input placeholder="Ex. Pizza" [(ngModel)]="food">

<br>

<p>Reason</p>

<input placeholder="Ex. Pizza" [(ngModel)]="reason">

<br>

<br>

<button (click)="addFavoriteFood()">Add</button>

<ul>

    <li \*ngFor="let myfood of favoriteFood">

      {{myfood.food}}: {{myfood.reason}}

    </li>

</ul>

Ahora como parte final el enrutamiento por medio del menu

menu.component.html

<mat-toolbar color="primary">

  <mat-toolbar-row>

    <span>workshop Toolbar</span>

    <a href="" mat-button routerLink="/" routerLinkActive="active"> Home </a>

    <a href="" mat-button routerLink="/form" routerLinkActive="active"> Form</a>

  </mat-toolbar-row>

</mat-toolbar>

app.module.ts

import {MatButtonModule} from '@angular/material/button';

@NgModule({

  declarations: [

    ...

  ],

  imports: [

...

    MatButtonModule,

Temas faltantes:

Crear módulos

Agregar componentes a los módulos

Enrutarse a un módulo

Crear módulosgenerales

Enviar/recibir parámentros dentro de los módulos

Trabajo con formularios

Consumo de apis (get/post/pull/delete)

Tablas

Entre otros