# ANGULAR 17-BOOTSTRAP 5

# Introducción

A continuación analizaremos un proyecto Angular17 desde cero. En el abordaremos diversos temas fundamentales, para poder llevar a cabo este tutorial se debe tener instalado NodeJS con la versión 18 como mínimo, actualmente estamos por la versión 20.

Además se utilizará Boostrap5 para realizar algunos diseños e incorporar elementos predefinidos y concentrarnos en el FrameWork más que en los estilos.

Incluiremos también una pequeña demostración de como instalar íconos desde fontawesome.

# CLI instalación

Lo primero que haremos es instalar la terminal de comandos de Angular.

**Instalar Angular CLI**: Si aún no tienes instalado Angular CLI, necesitarás instalarlo globalmente en tu máquina. Puedes hacerlo ejecutando el siguiente comando en tu terminal:

npm install -g @angular/cli

ng version

Puedes verificar la versión de Angular y de la CLI con el comando:

# Crear un nuevo proyecto Angular

Una vez que Angular CLI esté instalado, puedes crear un nuevo proyecto Angular ejecutando el siguiente comando en tu terminal:

ng new grid-proyect

Este comando crea un nuevo proyecto llamado "grid-proyect", configura el enrutamiento y utiliza CSS como lenguaje de estilos

migración de sintaxis

**ng g @angular/core:control-flow**

( El comando anterior se incluye aquí para que se pueda usar posteriormente. Permite cambiar la sintaxis antigua de las directivas de control, por ahora ignorarlo).

# Instalar Bootstrap 5

Ahora, necesitamos instalar Bootstrap 5 y sus dependencias en nuestro proyecto. En la página oficial de Bootstrap se encuentra el comando para su instalación.

<https://getbootstrap.com/docs/5.3/getting-started/download/>

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

npm install bootstrap@5.3.2

(nota: esto puede cambiar en el futuro, cambiar a la actual)

Texto

Descripción generada automáticamente

Observa en el fichero angular.json la parte de styles, allí se debe agregar la librería de Bootstrap que ha sido instalada previamente y que se encuentra en el directorio de los node\_modules.

Texto

Descripción generada automáticamente

**Interfaz de usuario gráfica, Texto

Descripción generada automáticamente**

Lo mismo se hace con el script de Bootstrap

**Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente**

De forma similar instalaremos para tener acceso a ciertos íconos de forma gratuita, no es indispensable pero ayuda a practicar los comandos de la CLI y agregar al angular.json.

npm install @fortawesome/fontawesome-free

El comando para ejecutar es:

Texto

Descripción generada automáticamente

Ahora se incluye en el archivo angular.json el paquete instalado

Texto

Descripción generada automáticamente

Así se dispondrá de los íconos los cuales se invocan por medio de clases, se verá posteriormente.

**En este enlace se puede ver por ejemplo el icono de GitHub desde la documentación oficial:**

<i class="fa-brands fa-github"></i>

[**https://fontawesome.com/icons/github?f=brands&s=solid**](https://fontawesome.com/icons/github?f=brands&s=solid)

# Crear componentes para el encabezado, el pie de página y la barra lateral

Genera los componentes header, footer y sidebar utilizando el Angular CLI. Guardaremos cada componente en un directorio llamado components

ng generate component components/header

ng generate component components/footer

para el sidebar usaremos la forma corta pero equivalente del comando anterior:

ng g c components/sidebar

Agregar componentes al controlador app.component.ts para poder crear el layout.

Para ello se edita el fichero mencionado importando las rutas de los componentes:

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

import {RouterOutlet } from '@angular/router';

import { HeaderComponent } from './components/header/header.component';

import { FooterComponent } from './components/footer/footer.component';

import { SidebarComponent } from './components/sidebar/sidebar.component';

@Component({

  selector: 'app-root',

  standalone: true,

  imports:

  [

    HeaderComponent,

    SidebarComponent,

    RouterOutlet,

    FooterComponent,

  ],

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

  styleUrl: './app.component.css'

})

export class AppComponent {

  title = 'grid-proyect';

}

# Crear el Layout usando Grid

Ahora que tenemos nuestros componentes importados en el archivo principal app-root, vamos a crear la interfaz con la siguiente plantilla:

En el fichero app.component.html escribimos el siguiente código:

<app-header ></app-header>

<app-sidebar ></app-sidebar>

<div class="content">

  <router-outlet></router-outlet>

</div>

<app-footer ></app-footer>

Agregamos el siguiente estilo sobre el fichero app.component.css para crear la grilla responsive usando template áreas:

:host {

  margin: 0;

  gap: 1rem;

  display: grid;

  grid-template-areas:

    "header header"

    "sidebar content"

    "footer footer";

  grid-template-columns: 250px 1fr;

}

.content {

    display: block;

    grid-area: content;

}

*/\* Media query en CSS \*/*

@media (width <= 768px) {

*/\* Estilos para cuando el ancho de la ventana es máximo 768px \*/*

  :host {

    grid-template-columns: 1fr; */\* Cambia a una sola columna \*/*

*/\* grid-template-rows: .4fr 1fr .4r; \*/* */\* Comenta esta línea si no la necesitas \*/*

    grid-template-areas:

      "header"

      "sidebar"

      "content"

      "footer";

  }

}

De esta manera tendremos nuestros componentes básicos de diseño implementados en un Layout típico y responsive.

El siguiente trabajo es escribir el código HTML y CSS de cada componente, para ello implementaré una mezcla de Bootstrap y CSS vanilla para personalizar algunos detalles.

En el fichero footer.component.html, escribo el código con las clases de Bootstrap

<footer class="nga-footer mt-auto">

  <div class="container py-5 text-center text-lg-start">

    <div class="row">

      <div class="col-12 col-lg-5 mb-3">

        <h2 class="h5"><img class="mb-1 me-1" src="./assets/kali-linux-logo.png"  width="25" height="25" alt="Logo Ganatan">CALIS7</h2>

        <hr class="text-white mb-4 mt-0 d-inline-block " style="width: 120px;">

        <p>Web Application : Angular 17, Bootstrap 5 </p>

        <p>Routing, Lazy Loading, SSR, PWA, SEO </p>

        <div>

          <a type="button" class="nga-btn-social nga-btn-linkedin" href="#"

            aria-label="Linkedin ganatan">

            <i class="fab fa-linkedin-in"></i>

          </a>

          <a type="button" class="nga-btn-social nga-btn-twitter" href="#"

            aria-label="Twitter ganatan">

            <i class="fab fa-twitter"></i>

          </a>

          <a type="button" class="nga-btn-social nga-btn-github" href="#"

            aria-label="Github ganatan">

            <i class="fab fa-github"></i></a>

          <a type="button" class="nga-btn-social nga-btn-gitlab" href="#"

            aria-label="Gitlab ganatan">

            <i class="fab fa-gitlab"></i></a>

        </div>

      </div>

      <div class="col-6 col-lg-3 mb-3">

        <h2 class="h5">Tools</h2>

        <hr class="text-white mt-0 d-inline-block" style="width: 70px;">

        <ul class="list-unstyled">

          <li class="mb-2"><a href="https://angular.io/">Angular</a></li>

          <li class="mb-2"><a href="https://getbootstrap.com/">Bootstrap</a></li>

          <li class="mb-2"><a href="https://fontawesome.com/">Font Awesome</a></li>

        </ul>

      </div>

      <div class="col-6 col-lg-3 mb-3">

        <h2 class="h5">Learn</h2>

        <hr class="text-white mt-0 d-inline-block" style="width: 70px;">

        <ul class="list-unstyled">

          <li class="mb-2"><a href="#">Tutorials</a></li>

          <li class="mb-2"><a href="#">About</a></li>

        </ul>

      </div>

    </div>

  </div>

</footer>

Y en el fichero footer.component.css algunos estilos personalizados, haciendo énfasis en las primeras líneas donde se incluye el componente al grid

:host{

  display: block;

  grid-area: footer;

}

.nga-footer {

  background-color: #212121;

  color: white;

}

.nga-footer {

  width: 100%;

  height: 100%;

}

.nga-footer a {

  color: white;

  text-decoration: none

}

.nga-footer a:hover,

.nga-footer a:focus {

  color: white;

  text-decoration: underline;

}

.nga-footer .nga-hint {

  background-color: #1976d2;

}

.nga-footer .nga-hint:hover {

  opacity: 0.8;

}

.nga-btn-social {

  position: relative;

  z-index: 1;

  display: inline-block;

  padding: 0;

  margin: 10px;

  overflow: hidden;

  vertical-align: middle;

  cursor: pointer;

  border-radius: 50%;

  -webkit-box-shadow: 0 5px 11px 0 rgba(0, 0, 0, 0.18), 0 4px 15px 0 rgba(0, 0, 0, 0.15);

  box-shadow: 0 5px 11px 0 rgba(0, 0, 0, 0.18), 0 4px 15px 0 rgba(0, 0, 0, 0.15);

  -webkit-transition: all 0.2s ease-in-out;

  transition: all 0.2s ease-in-out;

  width: 47px;

  height: 47px

}

.nga-btn-social i {

  font-size: 1.25rem;

  line-height: 47px

}

.nga-btn-social i {

  display: inline-block;

  width: inherit;

  color: white;

  text-align: center

}

.nga-btn-social:hover {

  -webkit-box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);

  box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19)

}

.nga-btn-social i:hover {

  color: black;

}

.nga-btn-github {

  background-color: #333;

}

.nga-btn-gitlab {

  background-color: #ff4500;

}

.nga-btn-linkedin {

  background-color: #0082ca;

}

.nga-btn-twitter {

  background-color: #55acee;

}

De forma análoga a los demás componentes se le escribe el código considerando que es personalizable y sus implementaciones pueden variar en función de los requisitos de diseño.

<https://getbootstrap.com/docs/5.3/components/navbar/>

En el enlace anterior se puede encontrar el código fuente tomado de la documentación de Bootstrap.

header.component.html

  <nav class="navbar navbar-expand-lg bg-body-tertiary" data-bs-theme="dark">

    <div class="container">

      <a class="navbar-brand" routerLink= "">Navbar</a>

      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"

        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

        <span class="navbar-toggler-icon"></span>

      </button>

      <div class="collapse navbar-collapse" id="navbarSupportedContent">

        <ul class="navbar-nav me-auto mb-2 mb-lg-0">

          <li class="nav-item">

            <a class="nav-link active" aria-current="page" routerLink="">Home</a>

          </li>

          <li class="nav-item">

            <a class="nav-link" routerLink="">Link</a>

          </li>

          <li class="nav-item dropdown">

            <a class="nav-link dropdown-toggle" routerLink="" role="button" data-bs-toggle="dropdown" aria-expanded="false">

              Dropdown

            </a>

            <ul class="dropdown-menu">

              <li><a class="dropdown-item" href="#">Action</a></li>

              <li><a class="dropdown-item" href="#">Another action</a></li>

              <li>

                <hr class="dropdown-divider">

              </li>

              <li><a class="dropdown-item" href="#">Something else here</a></li>

            </ul>

          </li>

          <li class="nav-item">

            <a class="nav-link disabled" aria-disabled="true">Disabled</a>

          </li>

        </ul>

        <form class="d-flex" role="search">

          <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

          <button class="btn btn-outline-success" type="submit">Search</button>

        </form>

      </div>

    </div>

  </nav>

header.component.css

:host {

  display: block;

  grid-area: header;

  width: auto;

  position: sticky;

  top: 0;

  z-index: 1000; */\* Establece un valor alto para asegurarte de que esté por encima \*/*

}

sidebar.component.html

    <div class="nga-container-sidebar">

      <ul class="list-unstyled p-2">

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#home-collapse" aria-expanded="false">

            API proyects

          </button>

          <div class="collapse" id="home-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

                <a routerLink="/clima" class="link-dark rounded"><i

                    class="fa-cloud fa-solid text-primary mt-1 me-2"></i>Clima App</a>

              </li>

              <li>

                <a routerLink="/bootstrap/badges" class="link-dark rounded"><i

                    class="fas fa-user-friends text-primary mt-1 me-2"></i>Badges</a>

              </li>

              <li>

                <a routerLink="/bootstrap/blockquotes" class="link-dark rounded"><i

                    class="fas fa-house-user text-primary mt-1 me-2"></i>Blockquotes</a>

              </li>

              <li>

                <a routerLink="/bootstrap/cards" class="link-dark rounded"><i

                    class="fab fa-asymmetrik text-primary mt-1 me-2"></i>Cards</a>

              </li>

              <li>

                <a routerLink="/bootstrap/collapse" class="link-dark rounded"><i

                    class="fas fa-balance-scale-right text-primary mt-1 me-2"></i>Collapse</a>

              </li>

              <li>

                <a routerLink="/bootstrap/dropdowns" class="link-dark rounded"><i

                    class="fas fa-bahai text-primary mt-1 me-2"></i>Dropdowns</a>

              </li>

              <li>

                <a routerLink="/bootstrap/list-group" class="link-dark rounded"><i

                    class="fab fa-battle-net text-primary mt-1 me-2"></i>List-group</a>

              </li>

              <li>

                <a routerLink="/bootstrap/modal" class="link-dark rounded"><i

                    class="fab fa-canadian-maple-leaf text-primary mt-1 me-2"></i>Modal</a>

              </li>

              <li>

                <a routerLink="/bootstrap/toasts" class="link-dark rounded"><i

                    class="fas fa-cogs text-primary mt-1 me-2"></i>Toasts</a>

              </li>

              <li>

                <a routerLink="/bootstrap/tooltips" class="link-dark rounded"><i

                    class="fab fa-edge text-primary mt-1 me-2"></i>Tooltips</a>

              </li>

            </ul>

          </div>

        </li>

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#orders-collapse" aria-expanded="false">

            Forms

          </button>

          <div class="collapse" id="orders-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

                <a routerLink="/bootstrap/buttons" class="link-dark rounded">

                  <i class="fab fa-artstation text-primary mt-1 me-2"></i>Buttons</a>

              </li>

              <li>

                <a routerLink="/bootstrap/checkbox" class="link-dark rounded">

                  <i class="fas fa-atom text-primary mt-1 me-2"></i>Checkbox</a>

              </li>

              <li>

                <a routerLink="/bootstrap/forms" class="link-dark rounded">

                  <i class="fas fa-basketball-ball text-primary mt-1 me-2"></i>Forms</a>

              </li>

              <li>

                <a routerLink="/bootstrap/pagination" class="link-dark rounded">

                  <i class="far fa-address-card text-primary mt-1 me-2"></i>Pagination</a>

              </li>

              <li>

                <a routerLink="/bootstrap/tables" class="link-dark rounded">

                  <i class="fas fa-chart-line text-primary mt-1 me-2"></i>Tables</a>

              </li>

            </ul>

          </div>

        </li>

        <li class="border-top my-3"></li>

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#account-collapse" aria-expanded="false">

            Extra

          </button>

          <div class="collapse" id="account-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

                <a routerLink="/bootstrap/accordions" class="link-dark rounded">

                  <i class="far fa-address-card text-primary mt-1 me-2"></i>Accordions</a>

              </li>

              <li>

                <a routerLink="/bootstrap/breadcrumb" class="link-dark rounded">

                  <i class="fab fa-app-store text-primary mt-1 me-2"></i>Breadcrumb</a>

              </li>

              <li>

                <a routerLink="/bootstrap/popovers" class="link-dark rounded">

                  <i class="fab fa-jedi-order text-primary mt-1 me-2"></i>Popovers</a>

              </li>

              <li>

                <a routerLink="/bootstrap/progress" class="link-dark rounded">

                  <i class="fab fa-galactic-republic text-primary mt-1 me-2"></i>Progress</a>

              </li>

              <li>

                <a routerLink="/bootstrap/spinners" class="link-dark rounded">

                  <i class="fab fa-empire text-primary mt-1 me-2"></i>Spinners</a>

              </li>

              <li>

                <a routerLink="/bootstrap/typography" class="link-dark rounded">

                  <i class="fas fa-dove text-primary mt-1 me-2"></i>Typography</a>

              </li>

            </ul>

          </div>

        </li>

      </ul>

    </div>

sidebar.component.css

:host{

  display: block;

  grid-area: sidebar;

  padding-top: auto;

}

.nga-btn-toggle {

    display: inline-flex;

    align-items: center;

    padding: .25rem .5rem;

    font-weight: 600;

    color: rgba(0, 0, 0, .65);

    background-color: transparent;

    border: 0;

  }

  .nga-btn-toggle:hover,

  .nga-btn-toggle:focus {

    color: rgba(0, 0, 0, .85);

    background-color: #cfe2ff;

  }

  .nga-btn-toggle::before {

    width: 1.25em;

    line-height: 0;

    content: url("data:image/svg+xml,%3csvg xmlns='http://www.w3.org/2000/svg' width='16' height='16' viewBox='0 0 16 16'%3e%3cpath fill='none' stroke='rgba%280,0,0,.5%29' stroke-linecap='round' stroke-linejoin='round' stroke-width='2' d='M5 14l6-6-6-6'/%3e%3c/svg%3e");

    transition: transform .35s ease;

    transform-origin: .5em 50%;

  }

  .nga-btn-toggle[aria-expanded="true"] {

    color: #0d6efd;

  }

  .nga-btn-toggle[aria-expanded="true"]::before {

    transform: rotate(90deg);

  }

  .nga-btn-toggle-nav a {

    display: inline-flex;

    padding: .1875rem .5rem;

    margin-top: .125rem;

    margin-left: 1.25rem;

    text-decoration: none;

  }

  .nga-btn-toggle-nav a:hover,

  .nga-btn-toggle-nav a:focus {

    color: #0d6efd;

    background-color: #cfe2ff;

    font-weight: bold;

  }

  @media (min-width: 768px) {

    .nga-container-layout {

      display: grid;

      gap: 1.5rem;

      grid-template-columns: 1fr 3fr

    }

  }

  @media (min-width: 992px) {

    .nga-container-layout {

      grid-template-columns: 1fr 5fr

    }

  }

  .nga-container-sidebar {

    overflow: auto;

    font-weight: 600

  }

  @media (min-width: 768px) {

    .nga-container-sidebar {

      position: -webkit-sticky;

      position: sticky;

      top: 5rem;

      display: block;

      height: calc(100vh - 7rem);

      padding-left: .25rem;

      margin-left: -.25rem;

      overflow-y: auto

    }

  }

Con lo realizado anteriormente se ha creado la estructura de la aplicación web , lo que sigue a continuación es poder crear diversos componentes de páginas, rutas, consumo de servicios por medio de alguna API etc.

Además se puede personalizar cada elemento de los componentes según sea necesario

# Enlazar componentes desde el sidebar para renderizarlos en el main content

Para este ejemplo reutilizaré código de un proyecto que ya se encuentra programado en Angular. Lo que haremos es reutilizar ese módulo que se maneja como una SPA, de esta manera lo renderizaremos sobre nuestro propio main content por medio del RouterOutlet.

Para ello crearemos un nuevo componente y lo ubicaremos en un directorio llamado pages, así:

ng g c pages/clima

una vez creado el componente debemos agregarlo al fichero de rutas app.routes.ts.

Por ahora pondremos por defecto que el home de nuestra app sea el RouterOutlet, más adelante modificaremos esto creando un componente home:

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

import { ClimaComponent } from './pages/clima/clima.component';

export const routes: Routes = [

{path: '', component:RouterOutlet},

{path: 'clima', component:ClimaComponent},

{path: '\*\*',redirectTo: '',pathMatch:'full' },

];

Obsérvese que al final del fichero se pone como ruta por defecto el RouterOutlet si esta es una cadena vacía. Esto se hace para que se redirija a dicha página cualquier enlace vacío.

Al crear el componente clima, Angular debió crear una estructura de directorios similar a la siguiente:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Vamos a pegar el código reutilizado del proyecto fuente pegando los códigos en cada fichero correspondiente:

clima.component.css

  .contenedor {

    max-width: 500px;

    margin: 0 auto;

    padding: 20px;

    text-align: center;

    background-color: #a0d3ff;

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

  }

  h1 {

    color: #333333;

  }

  input[type="text"] {

    padding: 10px;

    font-size: 16px;

    width: 70%;

  }

  button {

    padding: 10px 20px;

    font-size: 16px;

    background-color: #007bff;

    color: #ffffff;

    border: none;

    cursor: pointer;

  }

  #datosClima {

    margin-top: 20px;

    font-size: 18px;

  }

clima.component.html

    <div class="contenedor">

        <h1>Aplicación de Clima</h1>

        <input type="text" [(ngModel)]="ciudad" placeholder="Ingrese una ciudad">

        <button (click)="buscarClima(ciudad)" id="botonBusqueda">Buscar</button>

        <div \*ngIf="datosClima.ciudadNombre" id="datosClima">

            <h2>{{ datosClima.ciudadNombre }}, {{ datosClima.paisNombre }}</h2>

            <p>La temperatura es: {{ datosClima.temperatura }}ºC</p>

            <p>La humedad es: {{ datosClima.humedad }}%</p>

            <img [src]="datosClima.icono" alt="Icono del clima">

            <p>La descripción meteorológica es: {{ datosClima.descripcion }}</p>

        </div>

    </div>

clima.component.ts

import { CommonModule } from '@angular/common';

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

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

@Component({

  selector: 'app-clima',

  standalone: true,

  imports: [CommonModule, FormsModule,], *// Agregar CommonModule aquí*

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

  styleUrl: './clima.component.css'

})

export class ClimaComponent {

  urlBase = 'https://api.openweathermap.org/data/2.5/weather';

  api\_key = '605507acf87117e111e54a3ab5238541';

  difKelvin = 273.15;

  datosClima: any = {};

  ciudad: string = ''; *// Definición de la propiedad ciudad*

  constructor() { }

  buscarClima(ciudad: string) {

    if (ciudad) {

      this.fetchDatosClima(ciudad);

    }

  }

  fetchDatosClima(ciudad: string) {

    fetch(`${this.urlBase}?q=${ciudad}&appid=${this.api\_key}`)

      .then(data => data.json())

      .then(data => this.mostrarDatosClima(data));

  }

  mostrarDatosClima(data: any) {

    this.datosClima = {

      ciudadNombre: data.name,

      paisNombre: data.sys.country,

      temperatura: Math.floor(data.main.temp - this.difKelvin),

      humedad: data.main.humidity,

      descripcion: data.weather[0].description,

      icono: `https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`

    };

  }

}

De esta manera podremos incorporar todos los componentes que tengamos de nuestros proyectos anteriores y reutilizar el código.

Es importante mencionar que el fichero clima.component.spec.ts se utiliza para realizar pruebas pero por ahora no lo necesitamos.

# Manejo de rutas en el sidebar

Ahora debemos editar nuestro archivo sidebar.component.html para agregar la ruta de nuestro componente, para que no haya confusión lo reescribiré y mostraré donde se edita:

<div class="nga-container-sidebar">

      <ul class="list-unstyled p-2">

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#home-collapse" aria-expanded="false">

            API proyects

          </button>

          <div class="collapse" id="home-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

<a routerLink="/clima" class="link-dark rounded"><i

                    class="fas fa-user text-primary mt-1 me-2"></i>Clima App</a>

              </li>

              <li>

                <a routerLink="/bootstrap/badges" class="link-dark rounded"><i

                    class="fas fa-user-friends text-primary mt-1 me-2"></i>Badges</a>

              </li>

              <li>

                <a routerLink="/bootstrap/blockquotes" class="link-dark rounded"><i

                    class="fas fa-house-user text-primary mt-1 me-2"></i>Blockquotes</a>

              </li>

              <li>

                <a routerLink="/bootstrap/cards" class="link-dark rounded"><i

                    class="fab fa-asymmetrik text-primary mt-1 me-2"></i>Cards</a>

              </li>

              <li>

                <a routerLink="/bootstrap/collapse" class="link-dark rounded"><i

                    class="fas fa-balance-scale-right text-primary mt-1 me-2"></i>Collapse</a>

              </li>

              <li>

                <a routerLink="/bootstrap/dropdowns" class="link-dark rounded"><i

                    class="fas fa-bahai text-primary mt-1 me-2"></i>Dropdowns</a>

              </li>

              <li>

                <a routerLink="/bootstrap/list-group" class="link-dark rounded"><i

                    class="fab fa-battle-net text-primary mt-1 me-2"></i>List-group</a>

              </li>

              <li>

                <a routerLink="/bootstrap/modal" class="link-dark rounded"><i

                    class="fab fa-canadian-maple-leaf text-primary mt-1 me-2"></i>Modal</a>

              </li>

              <li>

                <a routerLink="/bootstrap/toasts" class="link-dark rounded"><i

                    class="fas fa-cogs text-primary mt-1 me-2"></i>Toasts</a>

              </li>

              <li>

                <a routerLink="/bootstrap/tooltips" class="link-dark rounded"><i

                    class="fab fa-edge text-primary mt-1 me-2"></i>Tooltips</a>

              </li>

            </ul>

          </div>

        </li>

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#orders-collapse" aria-expanded="false">

            Forms

          </button>

          <div class="collapse" id="orders-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

                <a routerLink="/bootstrap/buttons" class="link-dark rounded">

                  <i class="fab fa-artstation text-primary mt-1 me-2"></i>Buttons</a>

              </li>

              <li>

                <a routerLink="/bootstrap/checkbox" class="link-dark rounded">

                  <i class="fas fa-atom text-primary mt-1 me-2"></i>Checkbox</a>

              </li>

              <li>

                <a routerLink="/bootstrap/forms" class="link-dark rounded">

                  <i class="fas fa-basketball-ball text-primary mt-1 me-2"></i>Forms</a>

              </li>

              <li>

                <a routerLink="/bootstrap/pagination" class="link-dark rounded">

                  <i class="far fa-address-card text-primary mt-1 me-2"></i>Pagination</a>

              </li>

              <li>

                <a routerLink="/bootstrap/tables" class="link-dark rounded">

                  <i class="fas fa-chart-line text-primary mt-1 me-2"></i>Tables</a>

              </li>

            </ul>

          </div>

        </li>

        <li class="border-top my-3"></li>

        <li class="mb-1">

          <button class="btn nga-btn-toggle align-items-center rounded collapsed" data-bs-toggle="collapse"

            data-bs-target="#account-collapse" aria-expanded="false">

            Extra

          </button>

          <div class="collapse" id="account-collapse">

            <ul class="nga-btn-toggle-nav list-unstyled fw-normal pb-1 small">

              <li>

                <a routerLink="/bootstrap/accordions" class="link-dark rounded">

                  <i class="far fa-address-card text-primary mt-1 me-2"></i>Accordions</a>

              </li>

              <li>

                <a routerLink="/bootstrap/breadcrumb" class="link-dark rounded">

                  <i class="fab fa-app-store text-primary mt-1 me-2"></i>Breadcrumb</a>

              </li>

              <li>

                <a routerLink="/bootstrap/popovers" class="link-dark rounded">

                  <i class="fab fa-jedi-order text-primary mt-1 me-2"></i>Popovers</a>

              </li>

              <li>

                <a routerLink="/bootstrap/progress" class="link-dark rounded">

                  <i class="fab fa-galactic-republic text-primary mt-1 me-2"></i>Progress</a>

              </li>

              <li>

                <a routerLink="/bootstrap/spinners" class="link-dark rounded">

                  <i class="fab fa-empire text-primary mt-1 me-2"></i>Spinners</a>

              </li>

              <li>

                <a routerLink="/bootstrap/typography" class="link-dark rounded">

                  <i class="fas fa-dove text-primary mt-1 me-2"></i>Typography</a>

              </li>

            </ul>

          </div>

        </li>

      </ul>

    </div>

En esa porción enmarcada en color verde está la clave, se debe escribir la ruta con la sintaxis de Angular y no como un enlace tradicional :

<a routerLink="/clima"

De esta manera cada enlace del sidebar puede llevar a todos los componentes que tengamos incorporados en el proyecto.

Un último detalle importante es agregar el módulo RouterLink de Angular para que nos dé el efecto pointer en los enlaces de forma automática:  
  
Escribimos en el fichero sidebar.component.ts el siguiente código:

import { CommonModule } from '@angular/common';

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

import { RouterLink} from '@angular/router';

@Component({

  selector: 'app-sidebar',

  standalone: true,

  imports: [CommonModule,RouterLink],

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

  styleUrl: './sidebar.component.css'

})

export class SidebarComponent {

}

Puede observarse que al añadir del módulo RouterLink si se escribe directamente en imports generará la respectiva importación en la cabecera del fichero.

# Creando un componente home

Ahora crearemos un componente home dentro de Pages que servirá para introducir al usuario a nuestra app y explicarle de que va…

Usamos el siguiente comando para crear el componente:

ng g c pages/home

Ahora editamos sus ficheros con los siguientes códigos:

home.component.html

<div class="container">

    <div class="row">

      <div class="col-md-6 offset-md-3 text-center mt-0 container-fluid "> *<!-- Ajuste en el margen superior -->*

        <h1 class="display-4 mt-5">¡Bienvenido a nuestra App de Enseñanza!</h1>

        <p class="lead mb-4 small">Aquí aprenderás sobre Angular, Bootstrap y FontAwesome de manera fácil y divertida.</p> *<!-- Cambio en la clase para reducir el tamaño de la letra -->*

        <p class="mb-4">Navega a través del menú lateral izquierdo para explorar todas las aplicaciones disponibles. Puedes probar aplicaciones que utilizan APIs o reciclan componentes de otros proyectos.</p> *<!-- Nueva información adicional -->*

        <img src="https://angular.io/assets/images/logos/angular/angular.svg" alt="Angular Logo" width="100">

        <img src="https://getbootstrap.com/docs/5.1/assets/brand/bootstrap-logo.svg" alt="Bootstrap Logo" width="100">

        <i class="fab fa-font-awesome-flag fa-3x mt-3"></i>

        <div class="visit-counter mt-4">

          <p>¡Has visitado esta página {{ visitCount }} veces!</p>

          <button (click)="incrementVisitCount()" class="btn btn-primary">Aumenta</button>

        </div>

      </div>

    </div>

  </div>

home.component.css

:host {

    padding-top: auto;

    margin: 0;

  }

  h1 {

    color: #007bff; */\* Azul Bootstrap \*/*

  }

  .lead {

    color: #28a745; */\* Verde Bootstrap \*/*

  }

  .visit-counter {

    background-color: #ffc107; */\* Amarillo Bootstrap \*/*

    border: 2px solid #007bff; */\* Borde azul \*/*

    border-radius: 10px; */\* Bordes redondeados \*/*

    padding: 10px; */\* Espaciado interno \*/*

    font-size: 18px; */\* Tamaño de fuente \*/*

    font-weight: bold; */\* Negrita \*/*

  }

home.component.ts

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

@Component({

  selector: 'app-home',

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

  styleUrls: ['./home.component.css']

})

export class HomeComponent implements OnInit {

  visitCount: number = 0;

  appName: string = 'MiApp';

  constructor() { }

  ngOnInit(): void {

*// No incrementamos el contador al inicializar el componente*

  }

  incrementVisitCount() {

*// Incrementamos el contador de visitas al hacer clic en el botón*

    this.visitCount += 1;

}

}

Por ahora cambiamos el RouterOutlet por el HomeComponent

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

import { ClimaComponent } from './pages/clima/clima.component';

import { HomeComponent } from './pages/home/home.component';

export const routes: Routes = [

    {path: '', component:HomeComponent},

    {path:'clima', component:ClimaComponent},

    {path: '\*\*',redirectTo: '',pathMatch:'full' },

];

# Creación de servicios para consumo de API

Aunque hemos programado en nuestro componente clima.component.ts el consumo de la API para obtener los datos de cada ciudad, es recomendable utilizar otra arquitectura para dicho propósito.

Si quisiéramos reutilizar en otro componente dicho consumo deberíamos de nuevo declarar los atributos de clase como son las rutas , las llaves etc.

Para evitar dicha redundancia se recomienda crear servicios en los cuales se programa una única vez en un componente que se encargue de la lógica de negocio del consumo de API.

Es lo que en Angular se llama un “service”, para ello ejecutamos el comando:

(Antes de ejecutar este comando se recomienda crear una nueva rama de desarrollo con Git , por ejemplo “servicios” para tener ambas versiones y poder comparar posteriormente).

ng generate service services/clima

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

He eliminado el fichero de pruebas y solo dejaré el clima.service.ts

Ahora escribimos el siguiente código en dicho archivo.

import { Injectable, inject } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class ClimaService {

  private \_http = inject(HttpClient);

  private urlBase = 'https://api.openweathermap.org/data/2.5/weather';

  private apiKey = '605507acf87117e111e54a3ab5238541';

  private difKelvin = 273.15;

  buscarClima(ciudad: string): Observable<any> {

    return this.\_http.get(`${this.urlBase}?q=${ciudad}&appid=${this.apiKey}`);

  }

  procesarDatosClima(data: any): any {

    return {

      ciudadNombre: data.name,

      paisNombre: data.sys.country,

      temperatura: Math.floor(data.main.temp - this.difKelvin),

      humedad: data.main.humidity,

      descripcion: data.weather[0].description,

      icono: `https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`

    };

  }

}

*//otra forma de crear el servicio, sin el constructor*

*/\* import { Injectable } from '@angular/core';*

*import { HttpClient } from '@angular/common/http';*

*import { Observable } from 'rxjs';*

*@Injectable({*

*providedIn: 'root'*

*})*

*export class ClimaService {*

*private urlBase = 'https://api.openweathermap.org/data/2.5/weather';*

*private apiKey = '605507acf87117e111e54a3ab5238541';*

*private difKelvin = 273.15;*

*constructor(private http: HttpClient) { }*

*buscarClima(ciudad: string): Observable<any> {*

*return this.http.get(`${this.urlBase}?q=${ciudad}&appid=${this.apiKey}`);*

*}*

*procesarDatosClima(data: any): any {*

*return {*

*ciudadNombre: data.name,*

*paisNombre: data.sys.country,*

*temperatura: Math.floor(data.main.temp - this.difKelvin),*

*humedad: data.main.humidity,*

*descripcion: data.weather[0].description,*

*icono: `https://openweathermap.org/img/wn/${data.weather[0].icon}@2x.png`*

*};*

*}*

*}*

*\*/*

Como se puede ver he escrito dos alternativas para dicho código, en una se incluye una variable de clase directamente para injectar el HttpClient y en la otra se instancia con el constructor. Ambas son equivalentes y se deja al gusto del programador y/o los requisitos del software.

Es importante que este componente HTTP….. sea importado en la configuración del archivo app.config.ts de la siguiente manera:

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

import { provideRouter } from '@angular/router';

import { routes } from './app.routes';

import { provideHttpClient } from '@angular/common/http';

export const appConfig: ApplicationConfig = {

  providers: [provideRouter(routes),provideHttpClient()]

};

De esta manera el servicio podrá consultar el tipo de rutas con el protocolo Http.

Ahora actualizamos el componente que consume dicho servicio:

clima.component.ts

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

import { ClimaService } from '../../services/clima.service';

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

import { CommonModule } from '@angular/common';

@Component({

  selector: 'app-clima',

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

  styleUrls: ['./clima.component.css'],

  standalone: true,

  imports: [FormsModule,CommonModule],

})

export class ClimaComponent  {

  ciudad: string = '';

  private \_climaService = inject(ClimaService);

  datosClima:any;

  buscarCiudad() {

    this.\_climaService.buscarClima(this.ciudad).subscribe(

      (data) => {

        this.datosClima = this.\_climaService.procesarDatosClima(data);

      });

  }

}

*/\* import { Component, inject } from '@angular/core';*

*import { ClimaService } from '../../services/clima.service';*

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

*import { CommonModule } from '@angular/common';*

*@Component({*

*selector: 'app-clima',*

*templateUrl: './clima.component.html',*

*styleUrls: ['./clima.component.css'],*

*imports: [FormsModule,CommonModule],*

*standalone: true,*

*})*

*export class ClimaComponent {*

*ciudad: string = '';*

*datosClima: any;*

*constructor(private climaService: ClimaService) { }*

*buscarCiudad() {*

*this.climaService.buscarClima(this.ciudad)*

*.subscribe((data: any) => {*

*this.datosClima = this.climaService.procesarDatosClima(data);*

*});*

*}*

*}*

*\*/*

De nuevo se dan las dos alternativas de codificación.

Puede observarse en el código, que la clase llama los métodos que se han programado en el servicio y reescribe uno propio para poder invocarlo en el template y darle funcionalidad al botón de búsqueda.

clima.component.html

<div class="contenedor">

  <h1>Aplicación de Clima</h1>

  <input type="text" [(ngModel)]="ciudad" placeholder="Ingrese una ciudad" aria-label="Ingrese una ciudad">

  <button (click)="buscarCiudad()" id="botonBusqueda">Buscar</button>

  @if (datosClima.ciudadNombre) {

    <div id="datosClima">

      <h2>{{ datosClima.ciudadNombre }}, {{ datosClima.paisNombre }}</h2>

      <p>La temperatura es: {{ datosClima.temperatura }}ºC</p>

      <p>La humedad es: {{ datosClima.humedad }}%</p>

      <img [src]="datosClima.icono" alt="Icono del clima">

      <p>La descripción meteorológica es: {{ datosClima.descripcion }}</p>

    </div>

  }

</div>

Obsérvese que se introduce la nueva sintaxis de la directiva de control @if actualizando lo que teníamos previamente.

Importante he de mencionar que el método buscarCiudad() no tiene parámetros, ya que con el Binding que se hace usando la directiva [(ngModel)] se accede al atributo de clase.

Esto es de suma importancia ya que el método que se programa en el servicio llamado :

buscarClima(ciudad: string): Observable<any> {

    return this.\_http.get(`${this.urlBase}?q=${ciudad}&appid=${this.apiKey}`);

  }

Es el encargado de pasar por parámetro el texto del input del formulario.