Angular - Aplicación de Una Sola Página

A partir de ahora trabajaremos con CLI COMMANDS

Creamos una aplicación angular con nombre **sitioangular**

- **ng new sitioangular**
- **ingresamos a la carpeta sitioangular**
- **ejecuto comando ng serve**

Creamos las carpetas

src->app->components

src->app->components->shared

Ejecutamos el comando para crear 4 nuevos componente

- **ng g c components/shared/navbar**
- **ng g c components/home**
- **ng g c components/about**
- **ng g c components/platos**

Modifico los HTML con las plantillas de Bootstrap

navBar

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
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 mr-auto">
        <li class="nav-item" routerLinkActive="active">
          <a class="nav-link" [routerLink]="['home']">Home <span class="sr-
only">(current)</span></a>
        </li>
        <li class="nav-item" routerLinkActive="active">
```

```
          <a class="nav-link" [routerLink]="['platos']">Platos</a>
        </li>
        <li class="nav-item" routerLinkActive="active">
            <a class="nav-link" [routerLink]="['about']">Acerca De</a>
        </li>
      </ul>
      <form class="form-inline my-2 my-lg-0">
        <input class="form-control mr-sm-
2" type="search" placeholder="Buscar Plato" aria-label="Buscar Plato">
        <button class="btn btn-outline-success my-2 my-sm-
0" type="submit">Search</button>
      </form>
    </div>
</nav>
```

Home

```
<div class="jumbotron jumbotron-fluid, animated fadeIn">
    <div class="container">
      <h1 class="display-4">El Buen Sabor Delivery</h1>
      <p class="lead">Arma tu pedido y te lo llevamos a tu casa</p>
    </div>
  </div>
```

app-component

```
<app-navbar></app-navbar>
```

Rutas en Angular

Nos permiten navegar a los diferentes componentes

Abrimos el archivo

app-routing.module.ts

la estructura base de un archivo ts para administrar las rutas es

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


const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Necesitamos asignar las rutas de nuestros componentes home, about y Platos por lo tanto nuestro archivo de rutas debe quedar:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './components/home/home.component';
import { AboutComponent } from './components/about/about.component';
import { PlatosComponent } from './components/platos/platos.component';


const routes: Routes = [
  { path: 'home', component: HomeComponent },
    { path: 'about', component: AboutComponent },
    { path: 'platos', component: PlatosComponent },
    { path: '**', pathMatch: 'full', redirectTo: 'home' }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Verifique que nuestro archivo de rutas está incluido en nuestra aplicación, en el archivo app.module.ts debe estar incluida la importación

```
import { AppRoutingModule } from './app-routing.module';
```

Ahora debemos asociar nuestro archivo de rutas con la pagina html donde están contenidos los vínculos del menú de opciones, para lograr esta asociación usamos la etiqueta

```
<router-outlet></router-outlet>
```

La cual agregaremos en este caso en nuestro archivo app.component.html

```
<app-navbar></app-navbar>
<router-outlet></router-outlet>
```

Para aplicar los hipervínculos haremos uso de **routerLink** y de **routerLinkActive** los cuales cumplen una función similar a la propiedad href de html

Nuestras opciones del menú deben quedar de la siguiente forma:

```
      <li class="nav-item" routerLinkActive="active">
          <a class="nav-link" [routerLink]="['home']">Home <span class="sr-
only">(current)</span></a>
      </li>
      <li class="nav-item" routerLinkActive="active">
          <a class="nav-link" [routerLink]="['platos']">Platos</a>
      </li>
      <li class="nav-item" routerLinkActive="active">
            <a class="nav-link" [routerLink]="['about']">Acerca De</a>
      </li>
```

VAMOS A MEJORAR UN POCO LA ESTETICA DE NUESTRO SITIO

- Modifiquemos el contenido de nuestras páginas home y about

  Home

```
<div class="jumbotron jumbotron-fluid, animated fadeIn">
    <div class="container">
      <h1 class="display-4">El Buen Sabor Delivery</h1>
```

```
      <p class="lead">Arma tu pedido y te lo llevamos a tu casa</p>
    </div>
  </div>
```

About

```
<h1 class="animated fadeIn fast">Ejemplo de Angular para TSP - UTN</h1>
<h3 class="animated fadeIn">Laboratorio IV</h3>
<h3 class="animated fadeIn">Gerardo Magni</h3>
```

- Agreguemos en nuestro archivo **css** de estilo las siguientes clases

```
/*ANIMACION*/
.animated {
    -webkit-animation-duration: 1s;
    animation-duration: 1s;
    -webkit-animation-fill-mode: both;
    animation-fill-mode: both;
  }

  .fast {
    -webkit-animation-duration: 0.4s;
    animation-duration: 0.4s;
    -webkit-animation-fill-mode: both;
    animation-fill-mode: both;
  }

  @keyframes fadeIn {
    from {
      opacity: 0;
    }

    to {
      opacity: 1;
    }
  }

  .fadeIn {
    animation-name: fadeIn;
  }
```

De esta forma animamos nuestras páginas.

Para finalizar vamos a modificar nuestra página de **platos,** haremos uso del contenedor **Cards-Decks** de **Bootstrap** y de las imágenes que poseemos como recursos.

```
<h1>Nuestros Platos</h1>

<div class="card-deck">
    <div class="card" style="max-width: 25%; padding: 10px;">
      <img class="card-img-top" src="assets/img/pizza-
especial.jpg" alt="Card image cap">
      <div class="card-body">
        <h5 class="card-title">Pizza Especial</h5>
        <p class="card-text">$ 120</p>
      </div>
      <div class="card-footer" style="text-align: center;">
        <a href="#" class="btn btn-primary" style="width: 90%;">Detalle</a>
      </div>

    </div>
  </div>
```

La vista Actual de nuestra página debería ser:

Introducción a los Servicios

Los servicios cumplen las siguientes tareas:

- Brindan información a quien lo necesite
- Realizan peticiones CRUD (create, read, update, delete)
- Mantener la data de forma persistente (para el uso de multiples componentes)
- Servir como recurso reutilizable para nuestra aplicación de forma centralizada.

Haremos uso de los servicios de angular para de forma dinámica crear y cargar N tarjetas de platos.

Crearemos un servicio para realizar esta tarea.

Creamos una carpeta en src->app->servicios

Dentro de la carpeta creamos el archivo platos.service.ts y codificamos el servicio

Podemos crearlo mediante el comando:

ng g service <name-service> [options]
ng g service servicios/Delivery

Automáticamente me crea la clase DeliveryService
Verificar que se haya incluido en el app.module, caso contrario incluirlo

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class DeliveryService {

  constructor() { }
}
```

En este momento estamos en condiciones de usar el servicio, para ello importamos el servicio en nuestro componente Platos.component.ts

```
import { Component, OnInit } from '@angular/core';
import { DeliveryService } from 'src/app/servicios/delivery.service';

@Component({
  selector: 'app-platos',
  templateUrl: './platos.component.html',
```

```
  styleUrls: ['./platos.component.css']
})
export class PlatosComponent implements OnInit {

  constructor(private servicioDelivery:DeliveryService) { }

  ngOnInit(): void {
  }

}
```

Como vemos en el constructor

```
constructor(private servicioDelivery:DeliveryService) { }
```

Cuando instanciamos nuestro componente Platos también disparamos la llamada a nuestro servicio.

Podemos verificar esto visualizando la consola del navegador, debemos ver el mensaje **"servicio cargado!!!"**

Ahora nos queda cargar los datos a nuestra página de Platos por medio de nuestro servicio

Detengo la ejecución de la aplicación para realizar los cambios.

Los datos a cargar están contenidos en el archivo json platos.json (assets/datos/platos.json)

Para incluir un archivo JSON modifico el archivo
tsconfig.json
Agregó la propiedad

```
"resolveJsonModule": true,
```

Debe quedar asi:

```
{
  "compileOnSave": false,
  "compilerOptions": {
    "baseUrl": "./",
    "outDir": "./dist/out-tsc",
    "sourceMap": true,
    "declaration": false,
```

```
    "downlevelIteration": true,
    "experimentalDecorators": true,
    "resolveJsonModule": true,
    "module": "esnext",
    "moduleResolution": "node",
    "importHelpers": true,
    "target": "es2015",
    "lib": [
      "es2018",
      "dom"
    ]
  },
  "angularCompilerOptions": {
    "fullTemplateTypeCheck": true,
    "strictInjectionParameters": true
  }
}
```

Vuelvo a iniciar la aplicación

ng serve –o

Cargamos los datos JSON en nuestro servicio mediante la definición de una variable para tal fin, en este caso en la variable platos:

```
import { Injectable } from '@angular/core';
import * as data from 'src/assets/datos/platos.json'

@Injectable({
  providedIn: 'root'
})
export class DeliveryService {

  platosFile:any = (data as any).default;
```

```
  constructor() {
    console.log("Servicio Cargado!!!");
    console.log(this.platosFile);
   }

  public getPlatos():any[]{
    return this.platosFile.platos;
    console.log(this.platosFile);
  }

  public getPlatoXId(idx:string):any{
      for(let plato of this.platosFile){
          if(plato.id == idx){
            return plato;
          }
      }
  }

  public buscarPlatos(termino:string):any[]{
      let platosArr:any[] = [];
      termino = termino.toLowerCase();

      for(let plato of this.platosFile){
        let nombre = plato.nombre.toLowerCase();
        if(nombre.indexOf(termino) >= 0){
          platosArr.push(plato);
        }
      }
      return platosArr;
    }
}
```

Mostramos los datos en platos.component.html haciendo uso de ***ngFor**

```
<h1>Nuestros Platos</h1>

<div class="card-deck">
    <div class="card" *ngFor="let platoAux of platosArr; let i = index" style="ma
x-width: 33%; padding: 10px;">
```

```
      <img class="card-img-
top" src="assets/img/{{platoAux.imagenPath}}" alt="Card image cap" style="max-
height: 150px;">
      <div class="card-body">
        <h5 class="card-title">{{platoAux.nombre}}</h5>
        <p class="card-text">$ {{platoAux.precio}}</p>
      </div>
      <div class="card-footer" style="text-align: center;">
        <a href="#" class="btn btn-primary" style="width: 90%;">Detalle</a>
      </div>

    </div>
  </div>
```

Nuestro Componente Platos toma la forma

```
import { Component, OnInit } from '@angular/core';
import { DeliveryService } from 'src/app/servicios/delivery.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-platos',
  templateUrl: './platos.component.html',
  styleUrls: ['./platos.component.css']
})
export class PlatosComponent implements OnInit {

  platosArr:any[] = [];

  constructor(private servicioDelivery:DeliveryService, private router:Router) {

  }

  ngOnInit(): void {
    this.platosArr = this.servicioDelivery.getPlatos();
    console.log(this.platosArr);
  }

  public verPlato(idx:string){
```

```
    this.router.navigate(['/Plato', idx])
  }


}
```

Creamos un nuevo componente para ver el detalle de cada Plato.
ng g c components/DetallePlato

Donde vamos a mostrar el detalle del Plato al seleccionarlo

La ruta del botón la ejecutamos mediante la función verPlato(i) aplicando la función de angular **this.router.navigate**
Previa modificación de nuestro archivo de rutas, agregamos la nueva ruta

```
  { path: 'detallePlato/:id', component: DetallePlatoComponent },
```

Nuestra página DetallePlato.component.html contendrá el detalle del Plato:

```
<h1>{{plato.nombre | uppercase}}</h1>
<hr>
<div class="row">
    <div class="col-md-4" style="text-align: center">
        <img src="assets/img/{{plato.imagenPath}}" style="max-
width: 300px;" class="img-fluid" [alt]="plato.nombre">
        <br><br>
        <a [routerLink]="['/platos']" class="btn btn-outline-danger btn-
block">Regresar</a>
    </div>
    <div class="col-md-8">
        <h3 style="color: crimson">{{plato.nombre}}</h3>
        <hr>
        <p>{{plato.titulos}}</p>
        <p><b>Precio:</b> ${{plato.precio}}</p>
        <p><b>Rubro:</b> {{plato.rubro}}</p>
```

```
        <p><b>Ingredientes:</b><li *ngFor="let ingredienteAux of plato.ingredient
es; let i = index">{{ingredienteAux}}</li> </p>
        <div>
            <span *ngIf="plato.rubro == 'Bebidas'" style="color:red; font-
weight: bold;">
                Es Bebida
            </span>
        </div>
    </div>
</div>
```

Uso ***ngIf** para mostrar o no la leyenda es bebida

Para cerrar nuestra aplicación debemos recibir el id y cargar el Plato correspondiente
Según lo indicado en la ruta **'Plato/:id',** en el constructor de la clase **PlatoComponent** indico que al recibir el parámetro id ejecuto el método del servicio **getPlatoXId** y asigno el Plato encontrado en la variable Plato.

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { DeliveryService } from 'src/app/servicios/delivery.service';

@Component({
  selector: 'app-detalle-plato',
  templateUrl: './detalle-plato.component.html',
  styleUrls: ['./detalle-plato.component.css']
})
export class DetallePlatoComponent implements OnInit {

  plato:any;

  constructor(private activatedRoute:ActivatedRoute, private servicioDelibery:Del
iveryService) {

    this.activatedRoute.params.subscribe(params =>{
      console.log(params['id'])
      this.plato = this.servicioDelibery.getPlatoXId(params['id'])
```

```
    })
  }

  ngOnInit(): void {
  }
}
```

Finalmente siguiendo la misma línea de todo lo visto anteriormente codificamos la lógica para que funcione el buscador del sitio
Creo un componente buscador

ng g c components/buscador

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { DeliveryService } from 'src/app/servicios/delivery.service';

@Component({
  selector: 'app-buscador',
  templateUrl: './buscador.component.html',
  styleUrls: ['./buscador.component.css']
})
export class BuscadorComponent implements OnInit {

  platosBusqueda:any = [];
  termino:string;

  constructor(private activatedRoute:ActivatedRoute, private servicioDelivery:Del
iveryService, private router:Router) { }

  ngOnInit(): void {

    this.activatedRoute.params.subscribe(params=>{
      this.termino = params['termino'];
      this.platosBusqueda = this.servicioDelivery.buscarPlatos(params['termino'])
;
```

```
    });
  }

  public verPlato(idx:string){
    this.router.navigate(['/detallePlato', idx])
  }
}
```

HTML

```
<h1>Buscando: {{termino}}  </h1>

<div class="card-deck">
    <div class="card" *ngFor="let platoAux of platosBusqueda; let i = index" styl
e="max-width: 33%; padding: 10px;">
      <img class="card-img-
top" src="assets/img/{{platoAux.imagenPath}}" alt="Card image cap" style="max-
height: 150px;">
      <div class="card-body">
        <h5 class="card-title">{{platoAux.nombre}}</h5>
        <p class="card-text">$ {{platoAux.precio}}</p>
      </div>
      <div class="card-footer" style="text-align: center;">
        <button (click)="verPlato(platoAux.id)" class="btn btn-
primary" style="width: 90%;">Detalle</button>
      </div>
    </div>
  </div>
```

Agrego la ruta

```
    { path: 'buscar/:termino', component: BuscadorComponent },
```

El método de búsqueda (Clase Servicio) por término es el siguiente:

```
 public buscarPlatos(termino:string):any[]{
      let platosArr:any[] = [];
      termino = termino.toLowerCase();
```

```
      for(let plato of this.platosFile.platos){
        let nombre = plato.nombre.toLowerCase();
        if(nombre.indexOf(termino) >= 0){
          platosArr.push(plato);
        }

      }
      return platosArr;
    }
}
```

Modifico el componente navbar

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

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {

  constructor(private router:Router) { }

  ngOnInit(): void {
  }

  buscarPlatos(textoBusqueda:string){
    // console.log(textoBusqueda);
     this.router.navigate(['/buscar', textoBusqueda]);
   }


}
```

HTML

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="#">Navbar</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
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 mr-auto">
        <li class="nav-item" routerLinkActive="active">
          <a class="nav-link" [routerLink]="['home']">Home <span class="sr-
only">(current)</span></a>
        </li>
        <li class="nav-item" routerLinkActive="active">
          <a class="nav-link" [routerLink]="['platos']">Platos</a>
        </li>
        <li class="nav-item" routerLinkActive="active">
            <a class="nav-link" [routerLink]="['about']">Acerca De</a>
        </li>
      </ul>
      <form class="form-inline my-2 my-lg-0">
        <input class="form-control mr-sm-
2" type="search" placeholder="Buscar Plato" aria-
label="Buscar Plato" #buscarTexto>
        <button (click)="buscarPlatos(buscarTexto.value)" class="btn btn-outline-
success my-2 my-sm-0" type="submit">Search</button>
      </form>
    </div>
</nav>
```

Y de esta forma concluimos nuestra aplicación, podemos probarla y ver los resultados conseguidos.

En varios puntos de la aplicación podrá observar el uso de filtros o formateadores de datos conocidos como PIPES. Por ejemplo en el Detalle del Plato puede ver:

```
{{plato.nombre | uppercase}}
```

El cual pasa todo a MAYUSCULAS
Los pipes nos permiten transformar y/o formatear el dato original

Ejemplo de PIPES |

Permiten modificar propiedades visuales a los datos
Por ejemplo:
Dato | uppercase = pone todo en mayusculas
Dato | date : 'y' = obtiene solo el año de una fecha

Usando Parámetros en RUTAS

Declaro la Ruta en mi archivo de Ruteo.

```
export const routes: Routes = [
  { path: 'detalle-producto/:id', component: DetalleProducto }
];
```

La Ruta anterior sería equivalente a por ejemplo:

```
localhost:3000/detalle-producto/5
```

Vinculando a una ruta con parámetros en HTML

```
<a *ngFor="let producto of productos"
  [routerLink]="['/detalle-producto', producto.id]">
  {{ producto.nombre }}
</a>
```

Vinculando a una ruta con parámetros programáticamente

```
goToProductoDetalle(id) {
  this.router.navigate(['/detalle-producto', id]);
}
```