

TALLER 3 - FrontEnd
Diplomado de actualización en nuevas tecnologías para desarrollo de software
DESARROLLO DE SOFTWARE DE ÚLTIMA GENERACIÓN
UNIVERSIDAD DE NARIÑO
INGENIERÍA DE SISTEMAS
2023

HÉCTOR ARMANDO ACOSTA ORTIZ

1. Creación de componentes, uso de ngModel, RouterLink, Servicios.

Se debe contar con Noder.js y Angular. Para este ejercicio, Node.js ya está instalado.

Se instala angular:

```
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE> npm install -g @angular/cli
```

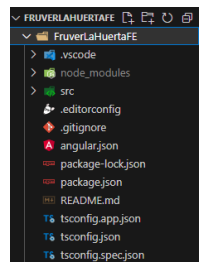
Se crea el proyecto:

```
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE> ng new FruverLaHuertaFE
```

Se responde yes al Angular routing y se selecciona Hojas de estilo **CSS**:

```
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE> ng new FruverLaHuertaFE
Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use?
  CSS
  Sass [ https://sass-lang.com/documentation/syntax#scss ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less  [ http://lesscss.org ]
```

Una vez instalado, en el panel izquierdo se desplegará el entorno de Angular:



En la carpeta **/src/app** estarán todos los archivos de la aplicación.

Para entrar a la aplicación se entra a la carpeta del proyecto y se ejecuta la siguiente línea:

```
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE> cd .\FruverLaHuertaFE\
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE\FruverLaHuertaFE> ng serve --open
```

Se instala y se configura el framework Bootstrap:

Desde la página oficial de Bootstrap, se copia y se pega en el index.html del proyecto las líneas:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-9ndCyUalbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhlNuZ2cdeRhO02iuK6FUUVM" crossorigin="anonymous">
```

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j/68Sly3Te4Bkz" crossorigin="anonymous"></script>
```

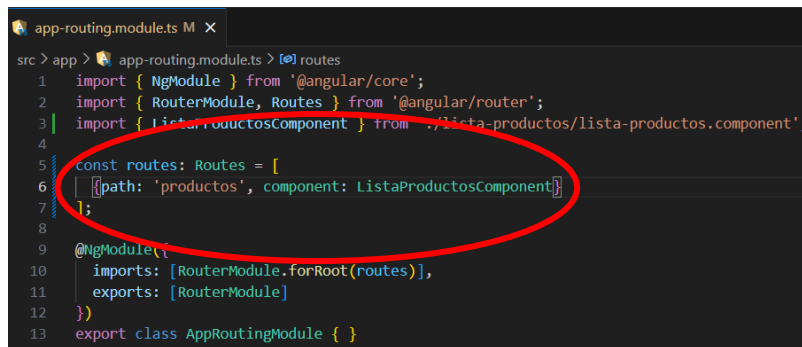
Generación de componentes.

Componente para la visualización de productos.

Dentro de la carpeta del proyecto se ejecuta la siguiente línea:

```
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE> cd .\FruverLaHuertaFE\
PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE\FruverLaHuertaFE> ng generate component lista-productos
CREATE src/app/lista-productos/lista-productos.component.html (30 bytes)
CREATE src/app/lista-productos/lista-productos.component.spec.ts (616 bytes)
CREATE src/app/lista-productos/lista-productos.component.ts (237 bytes)
CREATE src/app/lista-productos/lista-productos.component.css (0 bytes)
UPDATE src/app/app.module.ts (509 bytes)
```

En el archivo **app-routing.module.ts** se adiciona la ruta las rutas de los componentes:

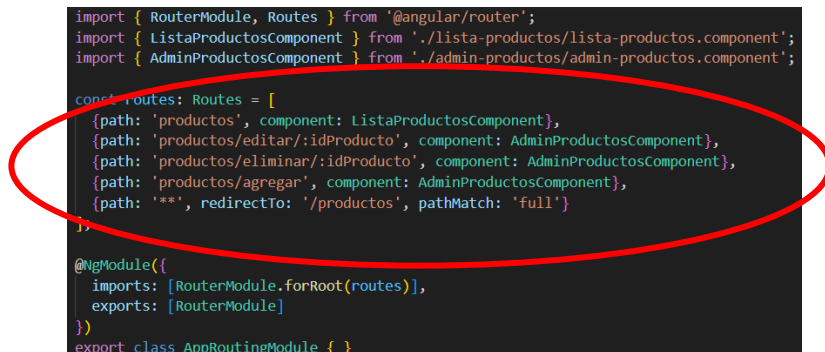


```
app-routing.module.ts M x
src > app > app-routing.module.ts > routes
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { ListaProductosComponent } from '../lista-productos/lista-productos.component';
4
5 const routes: Routes = [
6   {path: 'productos', component: ListaProductosComponent},
7 ];
8
9 @NgModule({
10   imports: [RouterModule.forRoot(routes)],
11   exports: [RouterModule]
12 })
13 export class AppRoutingModule { }
```

De igual manera se agregan los componentes necesarios:

ng generate component admin-productos (Para editar, eliminar, agregar)

Las rutas quedarían de la siguiente manera. La última línea de las rutas redirige la página a productos cuando la url es diferente a las otras rutas:



```
import { RouterModule, Routes } from '@angular/router';
import { ListaProductosComponent } from '../lista-productos/lista-productos.component';
import { AdminProductosComponent } from '../admin-productos/admin-productos.component';

const routes: Routes = [
  {path: 'productos', component: ListaProductosComponent},
  {path: 'productos/editar/:idProducto', component: AdminProductosComponent},
  {path: 'productos/eliminar/:idProducto', component: AdminProductosComponent},
  {path: 'productos/agregar', component: AdminProductosComponent},
  {path: '**', redirectTo: '/productos', pathMatch: 'full'}
];

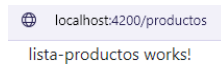
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Para editar la página inicial, se edita el archivo **app.component.html**:

Se crea el nuevo código con la siguiente estructura esencial obligatoria:

```
src > app > app.component.html > ...  
Go to component  
1 <div class="container">  
2   <router-outlet></router-outlet>  
3 </div>
```

Como resultado, en el navegador se ve lo siguiente porque llama al archivo **lista-productos-component.html** del componente:



Ahora se crean los modelos (Clases). Al mismo nivel de las carpetas de los componentes se crea la capeta que contendrá los modelos y dentro de esta carpeta que son archivos **ts** (TypeScript) que contendrá las clases:

Modelo ProductoModel:

```
app > modelos > TS productoModel.ts > ...  
export class ProductoModel {  
  constructor(  
    public ide:string,  
    public nombre:string,  
    public tipo:string,  
    public descripcion:string,  
    public precio_compra:string,  
    public precio_venta:string,  
    public cant_disponible:string,  
    public uni_medida:string) {  
  }  
}
```

En el archivo **app.component.html** se insertan todos los elementos HTML que estarán estáticos en la página, por ejemplo, el logo:

```
<div class="container">  
    
  <br /><br />  
  <router-outlet></router-outlet>  
</div>
```

Se edita el archivo **lista-productos-component.html** que será el archivo que se carga en el index:

Se crea el menú principal:

```

<nav class="navbar navbar-expand-lg bg-body-tertiary">
  <div class="container-fluid">
    <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" href="#">Inicio</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Nosotros</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Pedidos</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Mi carrito</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" [routerLink]="['/login']">Sesión</a>
        </li>
      </ul>
    </div>
  </div>
</nav>

```

Para filtrar productos, se crea un pipe llamado filtroProducto:

PS C:\Users\Hector\Desktop\UDENAR\Diplomado\taller3\fruverlahuertaFE\FruverLaHuertaFE> ng generate pipe filtroTipoProducto

Se crea en el archivo lista-productos-component.html un input text para el filtrado de productos:

```

<b>Buscar</b>
<div class="form-group">
  <input type="text" class="form-control" name="filtrarProducto" placeholder="Buscar producto..." [(ngModel)]="filtrarProducto">
</div>

```

El input text tiene un name="filtrarProducto" y un [(ngModel)]="filtrarProducto" para ejecutar el evento que filtra la lista de productos. La variable **filtrarProductos** se la define en archivo lista-productos-component.ts:

```
filtrarProducto = "";
```

En la carga de los registros en el html, se ejecuta la directiva *ngFor de la siguiente manera para ejecutar el filtro:

```
<tr *ngFor="let producto of productos | async | filtroProducto:filtrarProducto" class="row">
```

El archivo **filtro-producto.pipe.ts** queda de la siguiente manera:

```

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'filtroProducto'
})
export class FiltroProductoPipe implements PipeTransform {
  transform(value: any, arg: any): any {
    const resultadoFiltro = [];
    for (const producto of value) {
      const pro = producto.nombre;
      if (pro.toLowerCase().indexOf(arg.toLowerCase()) > -1) {
        resultadoFiltro.push(producto);
      }
    }
    return resultadoFiltro;
  }
}

```