Generalitat de Catalunya
Consorci per a la Formació Contínua de Catalunya

pime Coneixement

MINISTERIO DE TRABAJO, MIGRACIONES Y SEGURIDAD SOCIAL

SERVICIO PÚBLICO DE EMPLEO ESTATAL

Angular
IFCD65

Formador: Borja Mulleras Vinzia

Enrutamiento

Creamos manualmente las rutas

PASO 1
Creamos un proyecto rutas y lo compilamos
ng new rutas
(Contestamos N en rutas)
cd rutas
ng serve --open

PASO 2
Vamos a crear las paginas de la aplicación.
Queremos una home y una página usuario.
Eso es crear 2 componentes:
ng g c componentes/home --inline-style (no creamos CSS)
ng g c componentes/usuario

Enrutamiento

Creamos manualmente las rutas

PASO 3
En app.component.html cambiamos el html por:

```
<h1>App Rutas</h1>
<div class="links">
  <a routerLink="/home">Home</a>
  <a routerLink="/usuario">Gestión de
usuario</a>
</div>
<div class="main-container">
  <!-- Cargar la página que debe
visualizarse -->
  <!--router-outlet></router-outlet-->
</div>
```

No existe el atributo href

Enrutamiento

Creamos manualmente las rutas

PASO 3
En app.component.html cambiamos el html por:

```
<h1>App Rutas</h1>
<div class="links">
  <a routerLink="/home">Home</a>
  <a routerLink="/usuario">Gestión de
usuario</a>
</div>
<div class="main-container">
  <!-- Cargar la página que debe
visualizarse -->
  <router-outlet></router-outlet>
</div>
```

En router-outlet se cargar el componente que se elija.

Enrutamiento

Creamos manualmente las rutas

PASO 4

Creamos app.routes.ts dentro de la carpeta app y ponemos el siguiente contenido

```
import {RouterModule,Routes} from '@angular/router';
import {HomeComponent} from './componentes/home/home.component';
import {UsuarioComponent} from './componentes/usuario/usuario.component';

const rutas: Routes = [
        {path: 'home',component: HomeComponent},
        {path: 'usuario',component: UsuarioComponent},
        {path: '**',pathMatch: 'full',redirectTo:'home'} //cualquier otra ruta se
redirige a home
];

export const GLOBAL_ROUTES = RouterModule.forRoot(rutas);
```

```
interface Persona {
        nif:string,
        nombre:string,
        edad:number | null,
//edad adminte un número como un null
        admin?:boolean
//El ? Indica que el campo es opcional
}
class TiposDatos {
        public nombre: string
        public edad: number
        public persona:Persona
        constructor() {
                this.nombre ='Borja'
                this.edad=45
                this.persona = {
                        nif: '6666666',
                        nombre: 'Borja',
                        edad: 45,
                        admin: true
                }
        }
}
```

(Interfaces)

Las interfaces se utilizan principalmente para definir la estructura y los tipos de datos de un objeto, mientras que las clases se utilizan para crear objetos concretos con propiedades y comportamientos

Es un tipo de dato complejo de typeScript

En este caso Routes es un interface definido como un array

Enrutamiento

Creamos manualmente las rutas

PASO 5

Creamos app.module.ts dentro de la carpeta app y añadimos el siguiente contenido

En las importaciones:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import {GLOBAL_ROUTES} from './app.routes';
import { HomeComponent } from './componentes/home/home.component';
import { UsuarioComponent } from './componentes/usuario/usuario.component';
```

Enrutamiento

Creamos manualmente las rutas

PASO 5

Creamos app.module.ts dentro de la carpeta app y añadimos el siguiente contenido

```
@NgModule({
  //declaramos componentes, pipes, etc... propias del módulo
  declarations: [
    AppComponent,
    HomeComponent,
    UsuarioComponent
  ],
  //declaramos elementos externos al módulo como constantes. GLOBAL_ROUTES lo es.
  imports: [
    BrowserModule,
    GLOBAL_ROUTES
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

Enrutamiento

Creamos manualmente las rutas

PASO 7 (plus)
Añadimos dentro del app.component.html

```
<h1>App Rutas</h1>
<div class="links">
  <a routerLink="/home" routerLinkActive="activo">Home</a>
  <a routerLink="/usuario" routerLinkActive="activo">Gestión de usuario</a>
</div>
<div class="main-container">
  <!-- Cargar la página que debe visualizarse -->
  <router-outlet></router-outlet>
</div>
```

Debemos añadir en app.component.css la clase activo

Enrutamiento

Creamos manualmente las rutas

PASO 8 (plus)
Añadir en app.module.ts

```
import {RouterLinkActive} from '@angular/router';
```

Enrutamiento

Creamos manualmente las rutas

Aclaración sobre el uso de [] o sin

<a routerLink="/usuario" routerLinkActive="activo">Gestión de usuario

En este caso, le indicamos el nombre de la clase que difimos en app.component.css

O bien

<a routerLink="/usuario" [routerLinkActive]="activo">Gestión de usuario

En este caso, le indicamos que activo es una propiedad que esta definido en el componente app.component.ts, es decir, en el objeto AppComponent.

Angular 9

Crear un componente de página 404 y que sea lo que sale cuando enrutamos cualquier ruta que no exista,

Angular 10

Crear una aplicación nueva de plantas donde tengamos la home y gestión de plantas

Entrutamiento - Rutas hijas

PASO 1
Vamos a crear 3 páginas hijas de usuario:

ng g c componentes/usuario/nuevoUsuario --inline-style --flat
ng g c componentes/usuario/editarUsuario --inline-style --flat
ng g c componentes/usuario/detalleUsuario --inline-style –flat

--flat evita la creación de la carpeta usuario

PASO 2
Tenemos que añadir en usuario.component.html y usuario.component.css el CSS y usuario correspondiente con los botones de las 3 opciones

Entrutamiento - Rutas hijas

PASO 2
Tenemos que añadir en usuario.component.html y usuario.component.css el CSS y usuario correspondiente con los botones de las 3 opciones

```
<div class="botones">
            <a><button>Nuevo</button></a>
            <a><button>Editar</button></a>
            <a><button>Detalle</button></a>
</div>

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

Entrutamiento - Rutas hijas

PASO 3
Añadir en app.routes.ts las rutas hijas

```
import {NuevoUsuarioComponent} from './componentes/usuario/nuevo-
usuario.component';
import {EditarUsuarioComponent} from './componentes/usuario/editar-
usuario.component';
import {DetalleUsuarioComponent} from './componentes/usuario/detalle-
usuario.component';
```

Entrutamiento - Rutas hijas

PASO 3
Añadir en app.routes.ts las rutas hijas

```
const rutas:Routes = [
        {path: 'home',component: HomeComponent},
        {path: 'usuario',
         component: UsuarioComponent,
         children: [
                        {path: 'nuevo',component: NuevoUsuarioComponent},
                        {path: 'editar',component: EditarUsuarioComponent},
                        {path: 'detalle',component: DetalleUsuarioComponent},
                        {path: '**',pathMatch: 'full',redirectTo:'home'}
         ]
        },
        {path: '**',pathMatch: 'full',redirectTo:'home'}];
```

Entrutamiento - Rutas hijas

PASO 4
Añadir en usuario.component.html las rutas hijas

```
<div class="botones">
        <a routerLink="nuevo"> <button>Nuevo</button></a>
        <a routerLink="editar"><button>Editar</button></a>
        <a routerLink="detalle"><button>Detalle</button></a>
</div>

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

Entrutamiento - Rutas hijas

PASO 4 Bis
Añadir en usuario.component.html las rutas hijas

```
<div class="botones">
        <a routerLink="nuevo"> <button>Nuevo</button></a>
        <a routerLink="editar"><button>Editar</button></a>
        <a [routerLink]=”ruta3"><button>Detalle</button></a>
</div>

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

En este caso, al poner [routerLink] entre corchetes,
ruta3 es una variable que se ha de definir en usuario.component.ts
Public ruta3

Entrutamiento - Rutas hijas

PASO 5

Como en el caso anterior podemos añadir a los links routerLinkActive="activo”

OJO!!! No sirve el class declarado en app.component.css.

Se ha de declarar la clase en usuario.component.css

Angular 11

Crear dentro de la página de gestión de plantas, una página de insertar, otra de listar y una tercera de modificar

Entrutamiento con parámetros

Vamos a poner en el apartado usuario un id, es decir, queremos acceder a la aplicación con
/usuario/56
/usuario/id

Entrutamiento con parámetros

PASO 1

Vamos a modificar la constante rutas del app.routes.ts

```
const rutas:Routes = [
        {path: 'home',component: HomeComponent},
        {path: 'usuario/:id', //si se necesitan más parámetros 'usuario/:par1/:par2'
         component: UsuarioComponent,
        /* children: [
                        {path: 'nuevo',component: NuevoUsuarioComponent},
                        {path: 'editar',component: EditarUsuarioComponent},
                        {path: 'detalle',component: DetalleUsuarioComponent}
                        {path: '**',pathMatch: 'full',redirectTo:'nuevo'}
         ]*/
        },
        {path: '**',pathMatch: 'full',redirectTo:'home'} //cualquier otra ruta se redirige a
home
];
```

Entrutamiento con parámetros

PASO 2
Hay que importar en el componente que recibamos
el parámetro el módulo correspondiente

En este caso, es en usuario.component.ts

Y hay que añadir:
import { ActivatedRoute} from '@angular/router';

Y hay que añadir el siguiente contructor:

```
//INYECCIÓN DE DEPENDENCIA
constructor(private rutaActiva:ActivatedRoute) {  }
```

El objeto rutaActiva que se ha creado por inyección de dependencia
tiene una serie de propiedades que recibiremos.

Entrutamiento con parámetros

PASO 3
Localizamos el parámetro en el objeto rutaActiva

Podemos imprimir el objeto rutaActiva en el console.log

```
constructor(private rutaActiva:ActivatedRoute) {
    console.log(rutaActiva);
    let id = rutaActiva.snapshot.params['id’];
    console.log(id);
 }
```

Hay que modificar el app.component.html

```
<a routerLink="/usuario/20" routerLinkActive="activo">Gestión de usuario</a>
```

Entrutamiento con parámetros

Observamos que recuperamos el id

Pero OJO!!! ¿Qué pasa si ponemos varios links?

Modificamos el app.component.html

```
<a routerLink="/usuario/20" routerLinkActive="activo">Usuario 1</a>
<a routerLink="/usuario/30" routerLinkActive="activo">Usuario 2</a>
<a routerLink="/usuario/40" routerLinkActive="activo">Usuario 3</a>
```

¡¡PROBLEMA GORDO!!

Entrutamiento con parámetros

Para solucionarlo utilizamos los observables

Los observables nos permiten suscribirnos a eventos que se recibirán de manera asíncrona manteniendo un alto rendimiento

En el objeto ActivatedRouted tenemos una propiedad llamada params que es un observable y que nos permite suscribirnos a cambios en los parámetros enviados al componente.

Las suscripciones a los cambios en los observables se hace mediante el método suscribe(). Este método recibe varios parámetros, pero solo nos interesa, de momento, el primero. Se trata de una función callback que se ejecuta en cada cambio de aquello que se está observando.

TODO LO QUE CAMBIA ES UN OBSERVABLE

Entrutamiento con parámetros

PASO 4

En usuario.components.ts importar Params:
import { ActivatedRoute,Params} from '@angular/router';

Y cambiar el constructor():

```
constructor(private rutaActiva:ActivatedRoute) {
      //console.log(rutaActiva);
      //let id = rutaActiva.snapshot.params['id'];
      this.rutaActiva.params.subscribe((parametros:Params) =>
                                  {
                                    let id=parametros['id'];
                                    console.log(id);
                                  });
  }
```

Entrutamiento con parámetros

¿Podemos acceder al id des de los componentes hijos?

Queremos que al acceder a nuevo recuperar el id de la ruta padre

PASO 1
Modificamos nuevo-usuario.component.ts

Entrutamiento con parámetros

```
import { Component } from '@angular/core';
import { ActivatedRoute,Params} from '@angular/router';

@Component({
  selector: 'app-nuevo-usuario',
  templateUrl: './nuevo-usuario.component.html',
  styles: [
  ]
})
export class NuevoUsuarioComponent {
constructor(private rutaActivaHija:ActivatedRoute) {
            //parent? Significa que puede ser opcional
     this.rutaActivaHija.parent?.params.subscribe((parametros:Params) =>
                          {
                            let id=parametros['id'];
                            console.log('Parametro Ruta hija Nuevo',id);
                          });
  }
}
```

Entrutamiento con parámetros

En caso de tener rutas hijas puede ser necesario que las rutas a las hijas que encontramos en usuario.component.html se indiquen:

```
<div class="botones">
            <a routerLink="nuevo" routerLinkActive="activo"><button>Nuevo</button></a>
            <a routerLink="editar" routerLinkActive="activo"><button>Editar</button></a>
            <a routerLink="detalle" routerLinkActive="activo"><button>Detalle</button></a>

</div>
```

O bien

```
<div class="botones">
            <a [routerLink]="['nuevo']" routerLinkActive="activo"><button>Nuevo</button></a>
            <a [routerLink]="['editar']" routerLinkActive="activo"><button>Editar</button></a>
            <a [routerLink]="['detalle']" routerLinkActive="activo"><button>Detalle</button></a>

</div>
```

ANGULAR 12

Crear 3 url’s que respondan a:

http://localhost:4200/planta/20/nuevo
http://localhost:4200/planta/20/editar
http://localhost:4200/planta/20/detalle

MÓDULOS

Creamos un nuevo proyecto, pero ahora le diremos que si cree el enrutado

ng new modulos

Vamos a crear una aplicación con 4 módulos:

Módulo común: consta de los elementos comunes, header, footer, error404, navbar, etc..
Módulos clientes, proveedores y almacén

MÓDULOS

Componentes:

Módulo común:
HeaderComponent / FooterComponent / NavbarComponent
Error404Component
HomeComponent

Clientes
ListaClientesComponent
DetalleClienteComponent

Proveedores
ListaProveedoresComponent
DetalleProveedorComponent

Alamcén
ListaProductosComponent
SalidaProductoComponent
LlegadaProductoComponent

MÓDULOS

Borramos el contenido de app.component.html
Pero dejaremos la etiqueta <router-outlet>

Creamos los módulos
ng g m comun --routing
ng g m clientes --routing
ng g m proveedores --routing
ng g m almacen –routing

MÓDULOS

Creamos los componentes del módulo común
ng g c comun/header --export
ng g c comun/footer --export
ng g c comun/navbar --export
ng g c comun/error404 --export
ng g c comun/home --export

MÓDULOS

ATENCIÓN: hay que ir al app.modulo.ts y añadir el módulo

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ComunModule } from './comun/comun.module';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ComunModule //Módulo con los componentes comunes de la aplicación
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

MÓDULOS

Añadir en app.component.html:

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

Añadir en app.component.html:

```
<app-header></app-header>
<app-home></app-home>
<router-outlet></router-outlet>
```

Realmente queremos que app-home se cargue en <router-outlet>

Esto se hace con el enrutado.

MÓDULOS

Vamos a app-routing.module.ts
Es un poco distinto al que creamos ayer, pero hace lo mismo.
Añadimos la ruta vacia y que cargue el HomeComponent

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './comun/home/home.component';

const routes: Routes = [
    {path:'',component:HomeComponent}
];

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

MÓDULOS

Continuamos añadiendo componentes a Clientes
ng g c clientes/listaClientes --export
ng g c clientes/detalleCliente --export

Continuamos añadiendo componentes a Proveedores
ng g c proveedores/listaProveedor --export
ng g c proveedores/detalleProveedor --export

Y, finalmente, añadimos componentes de alamcen.
ng g c almacen/listaProductos --export
ng g c almacen/salidaProducto --export
ng g c almacen/llegadaProducto --export

MÓDULOS

Trabajaremos la barra de navegación.

Añadimos en app.component.html

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

Y en navbar.component.html
El menú que ejecutaremos

```
<div class="links">
  <a routerLink="/" routerLinkActive="activo">Inicio</a>
  <a routerLink="/clientes" routerLinkActive="activo">Clientes</a>
  <a routerLink="/proveedores" routerLinkActive="activo">Proveedores</a>
  <a routerLink="/almacen" routerLinkActive="activo">Almacen</a>
</div>
```

MÓDULOS

Y añadimos en app-routing.module.ts las rutas

```
…
import { HomeComponent } from './comun/home/home.component';
import { ListaProductosComponent } from './almacen/lista-productos/lista-
productos.component';
import { ListaClientesComponent } from './clientes/lista-clientes/lista-
clientes.component';
import { ListaProveedorComponent } from './proveedores/lista-proveedor/lista-
proveedor.component';
const routes: Routes = [
    {path:'',component:HomeComponent},
    {path:'clientes',component:ListaClientesComponent},
    {path:'proveedores',component:ListaProveedorComponent},
    {path:'almacen',component:ListaProductosComponent}
];
…
```

ANGULAR 12

Crear el componente de error 404 que se ejecuta cuando la url es errónea.

ANGULAR 13

Cuando se apriete en Clientes aparezcan 2 botones con Lista de Clientes y Detalle

MÓDULOS

PASO 1
Creamos componentes nuevos en clientes
ng g c clientes/clientes --export

MÓDULOS

PASO 2
Editamos clientes.component.html

```
<h2>Módulo de clientes</h2>
<div class="links">
        <a routerLink="/listaClientes" routerLinkActive="activo">
                <button>Lista de clientes</button>
        </a>
         <a routerLink="/detalleCliente" routerLinkActive="activo">
                <button>Detalle de clientes</button>
        </a>
</div>
<router-outlet></router-outlet>
```

MÓDULOS

PASO 3
Editamos app-routing.module.ts y modificamos

```
import { ClientesComponent } from './clientes/clientes/clientes.component';
import { ListaProveedorComponent } from './proveedores/lista-proveedor/lista-proveedor.component';
import { Error404Component } from './comun/error404/error404.component';

const routes: Routes = [
    {path:'',component:HomeComponent},
    {path:'clientes',component:ClientesComponent},
    {path:'proveedores',component:ListaProveedorComponent},
    {path:'almacen',component:ListaProductosComponent},
    {path: '**',pathMatch: 'full',component:Error404Component}
];
```

Eliminamos el import de ListaClientesComponent y añadimos el de clientes

MÓDULOS

PASO 4

Editamos app-routing.module.ts y modificamos

```
…
import { ClientesComponent } from './clientes/clientes/clientes.component';
import { ListaClientesComponent } from './clientes/lista-clientes/lista-clientes.component';
import { DetalleClienteComponent } from './clientes/detalle-cliente/detalle-cliente.component';
…
const routes: Routes = [
    {path:'',component:HomeComponent},
    {path:'clientes',component:ClientesComponent,children: [
        {path:'listaClientes',component:ListaClientesComponent},
        {path:'detalleCliente',component:DetalleClienteComponent},
    ]},
    {path:'proveedores',component:ListaProveedorComponent},
    {path:'almacen',component:ListaProductosComponent},
    {path: '**',pathMatch: 'full',component:Error404Component}
];
```

MÓDULOS

PASO 5
Editamos app.module.ts

```
…
import { ClientesModule } from './clientes/clientes.module';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    ComunModule, //Módulo con los componentes comunes de la aplicación
    ClientesModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

ANGULAR 14

Repetir los mismos pasos para hacer proveedores y almacen

Generalitat de Catalunya
Consorci per a la Formació Contínua de Catalunya

pime Coneixement

MINISTERIO DE TRABAJO, MIGRACIONES Y SEGURIDAD SOCIAL

SERVICIO PÚBLICO DE EMPLEO ESTATAL

Angular
IFCD65

Formador: Borja Mulleras Vinzia