DESARROLLO DE APLICACIONES CON ANGULAR

Angular

ANGULAR

- Routing
  - Mecanismo para la creación de rutas "virtuales" para navegar a las diferentes partes de la aplicación SPA creada con Angular.
  - Permite acceder, además a la página principal, a las rutas internas directamente.
  - Permite utilizar el uso del histórico del navegador.
  - Participantes:
    - Módulos Routes y RouterModule.
    - Listado con las rutas (objetos de tipo Routes).
    - Contenedor.

ANGULAR

- Routing
  - La definición se realiza en el fichero:
    - **/src/app/app-routing.module.ts**

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

const routes: Routes = [];

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

Módulos importados

Rutas

Módulo contenedor

Importado en app.module.ts

ANGULAR

- Routing
  - La definición se realiza en el fichero src/app/app-routing.module.ts:
  - Sintaxis básica:

    ```
    {path:'path',component:componente}
    ```
  - Ejemplo:

    ```
    {path:'c2',component:Componente2Component,pathMatch:'prefix'}
    ```
  - Parámetro redirectTo, permite redireccionar un path a otro.

    ```
    {path:'componente1',redirectTo:'c2'}
    ```
  - Path raíz:

    ```
    {
        path:'',
        component:PrincipalComponent,
        pathMatch:'full'//Siempre para el path raíz
      }
    ```
  - Path por defecto (debe ir el último):

    ```
    {
        path:'**',
        component:Componente404Component
    }
    ```

ANGULAR

- Routing
  - **Ejemplo:**

```
const routes: Routes = [
  {path:'c1',component:Componente1Component,pathMatch:'full'},
  {path:'c2',component:Componente2Component,pathMatch:'prefix'},
  {path:'c3',component:Componente3Component},//Por defecto es prefix
  {path:'componente1',redirectTo:'c1'},
  {
    path:'',
    component:PrincipalComponent,
    pathMatch:'full'//Siempre para el path ''
  },
  {
    path:'**',//El último
    component:Componente404Component
  }
];
```

ANGULAR

- Routing
  - Enlaces:
    - Externamente (desde la barra de direcciones del navegador):
      - servidor/ruta
      - Ejemplo:
        - http://localhost:4200/c2
    - Internamente (desde el template HTML):
      - Utilizando la directiva [routerLink]
      - Ejemplo:
        - `<a [routerLink]="['/c1']">C1</a>`
        - `<a routerLink="/c1">C1</a>`
      - Admite parámetros:
        - `routerLink="/padre/hijo1/{{nombre}}"`
        - `[routerLink]="['/padre','hijo1',nombre]"`

ANGULAR

- Routing
  - Se puede resaltar el elemento activo para facilitar el diseño de las interfaces:
    - Utilizando la directiva [routerLinkActive] se puede asignar una clase css
      - Ejemplo:

```
<ul>
    <li><a [routerLink]="['/c1']" [routerLinkActive]="['enlace-activo']">Géneros</a></li>
    <li><a [routerLink]="['/c2']" [routerLinkActive]="['enlace-activo']">Plataformas</a></li>
    <li><a [routerLink]="['/c3']" routerLinkActive="enlace-activo">Formulario</a></li>
</ul>
```

      -

```
.enlace_activo {
    color:red;
    font-weight: bolder;
}
```

ANGULAR

- Routing
  - Navegación desde código:
    - Desde el TS origen de la navegación se debe utilizar el método **navigate** de la clase **Router**.
    - La instancia de Router se reciben en el constructor y debe ser un atributo.

```
constructor(private router:Router) { }
```

    - La llamada a **navigate** recibe un parámetro any[].

```
this.router.navigate(['c2’]);
this.router.navigateByUrl(‘/c2');
```

ANGULAR

- Routing
  - Rutas dinámicas, variables o con parámetros:
    - Permite construir rutas con partes dinámicas que funcionarán como parámetros.
    - En el array de routas (app-routing.module.ts) se indican los parámetros dentro del path precedidos por dos puntos (:):

```
{path:'c4/:idProducto',component:Componente4Component},
```

    - En el constructor del componente se indica que éste recibe un objeto ActivatedRoute que será el encargado de recoger los parámetros.

```
constructor(private ar:ActivatedRoute) {
```

    - Un el método ngOnInit se realiza la suscripción a la recepción de parámetros y su proceso.

```
ngOnInit(): void {
    this.ar.params.subscribe(params=>{
      this.identificador=params['idProducto'];
    })
  }
```

ANGULAR

- Routing
  - Rutas hijas:
    - Permite agregar otros componentes a rutas, mediante rutas 'children'.
    - Declaración, mediante el atributo **children**:

```
{path:'c4/:idProducto',component:Componente4Component, children: [
    {path:'hijo1',component:Componentechildren1Component},
    {path:'hijo2',component:Componentechildren2Component}
  ]
},
```

    - En el elemento al que se van a incrustar los hijos (Componente4Compontent en el ejemplo) hay que introducir el nodo <router-outlet></router-outlet> en la posición en la que se desee que aparezca el elemento 'hijo'.

```
<p>{{identificador}}</p>
<router-outlet></router-outlet>
```

    - El path podrá incluir parámetros e los hijos. Ejemplo: llamada al path c4, con un parámetro y un hijo:

```
http://localhost:4200/c4/100/hijo1
```

ANGULAR

- Routing
  - Router guards:
    - Permite poner límites a las rutas.
    - Se debe crear una clase que proporcione true o false para permitir acceder a una ruta o no.
    - ng generate guard *nombre_guard.*
      - Se debe elegir el tipo de guard:
        - CanActivate →Bloquea navegación (más habitual)
        - CanActivateChild →B loquea el acceso a una ruta con hijos
        - CanDeactivate →Bloquea la salida de una ruta.
        - CanLoad →Bloquea la carga
    - En función de la elección en la creación (pueden ser múltiples) se crean los métodos canActivate, canActivateChild, canDeactivate, canLoad, que devolverán true o false para indicar al path si es viable o no.
    - En la lista de rutas (Routes) se harán referencia a los guards en función de su tipo:

```
{path:'c1',component:Componente1Component,pathMatch:'full',canActivate:[GuardComponente1Guard],canLoad:[GuardComponente2Guard]},…
```

ANGULAR

- Routing
  - Router guards:
    - Dentro del guard se pueden devolver objetos UrlTree para redirigir la navegación hacia un recurso concreto.

```
constructor(private router:Router){}

canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (localStorage.getItem("nombre")!=null) {
      return true;
    } else {
      //Alternativas:
      //return this.router.parseUrl("/noautorizado");
      return this.router.createUrlTree(["/noautorizado"]);
    }
  }
```

ANGULAR

- Routing
  - Router guards:
    - Dentro del guard se puede hacer uso del routing para redirigir a una página de error.

```
constructor(private router:Router){}

canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (localStorage.getItem("nombre")!=null) {
      return true;
    } else {
      this.router.navigateByUrl("/noautorizado");
      return false;
    }
  }
```

ANGULAR

- Dar nombre a las páginas:
  - Inyectando el servicio Title y mediante el método setTitle:

```
constructor(private titleService:Title) {
    this.titleService.setTitle("Título");
}
```

- Ruta global:
  - Modificando el valor de `<base href="/">` del fichero index.html todas las rutas tendrán como base el valor indicado.