Tema 5 – Routing y navegación

- ¿Qué son las rutas en Angular?
- Crear y usar rutas
- Navegación mediante enlaces
- Rutas a componentes anidados
- *Lazy loading* de módulos mediante rutas
- Gestionar rutas no existentes
- Usar parámetros en las rutas
- Navegación programática

Rutas en SPA vs Rutas tradicionales

- **Rutas en navegación tradicional:**
  - Basada en URL,s. que enlazan a páginas estáticas HTML, o a páginas dinámicas generadas en servidor.
  - El DOM se sustituye completamente en cada navegación.
  - Todos los enlaces generan llamadas al servidor, quien determina el contenido a mostrar.

Rutas en SPA vs Rutas tradicionales

- **Rutas en Single Page Applications (SPA):**
  - Basada en URL,s. que enlazan a componentes, no a páginas
  - El DOM no se recarga totalmente, solo se sustituye la parte afectada.
  - Los enlaces no generan llamadas al servidor, se gestionan dentro de la aplicación en cliente. Solo si es necesario se solicita la descarga de módulos o componentes.

Rutas en Angular

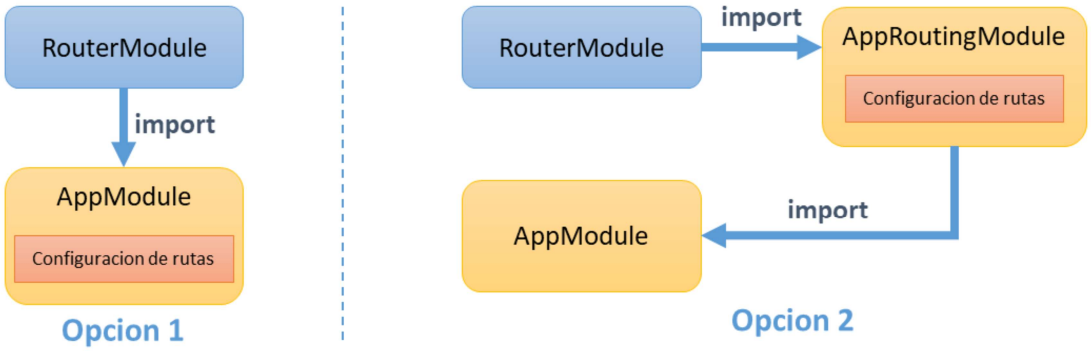
- Las aplicaciones Angular son SPA, por tanto se aplica todo lo anterior.
- Soporta el botón “Atrás” del navegador.
- Permite la carga “perezosa” de los módulos (lazy load), para descargarlos solo cuando se navega a alguno de sus componentes.

Crear y usar rutas

- Para usar rutas en Angular, 3 pasos:
  1. Configurar la ruta con el módulo **RouterModule**
  2. Indicar la ubicación del componente con un **Router – Outlet**
  3. Crear **enlaces** a la ruta en la plantilla HTML

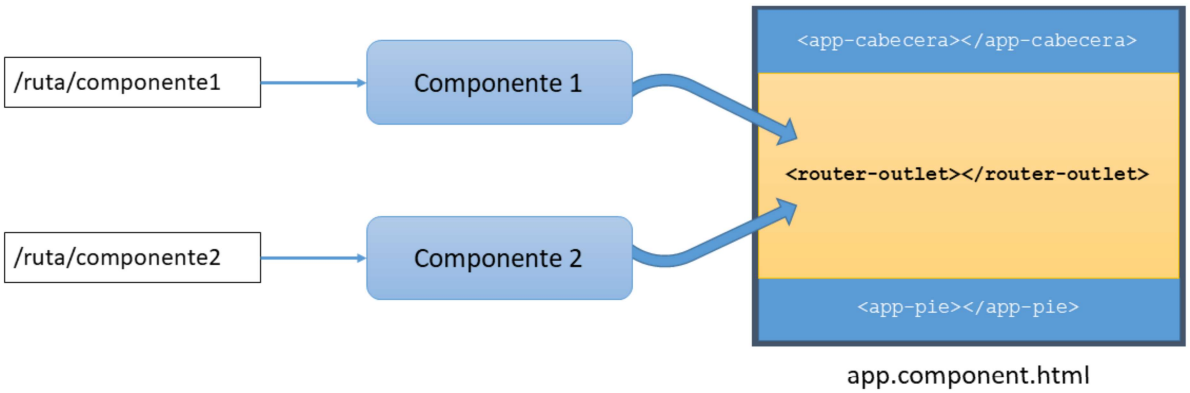
El módulo RouterModule

- El módulo RouterModule (@angular/router) permite asociar rutas URL a componentes.
- Para usarlo hay que importarlo en el módulo App o en un módulo intermedio:



Ubicar el componente con un Router – Outlet

- Router Outlet: Indica la ubicación dentro de la plantilla HTML donde se deben cargar los componentes según la ruta configurada en el Router:



Crear enlaces a la ruta en la plantilla HTML

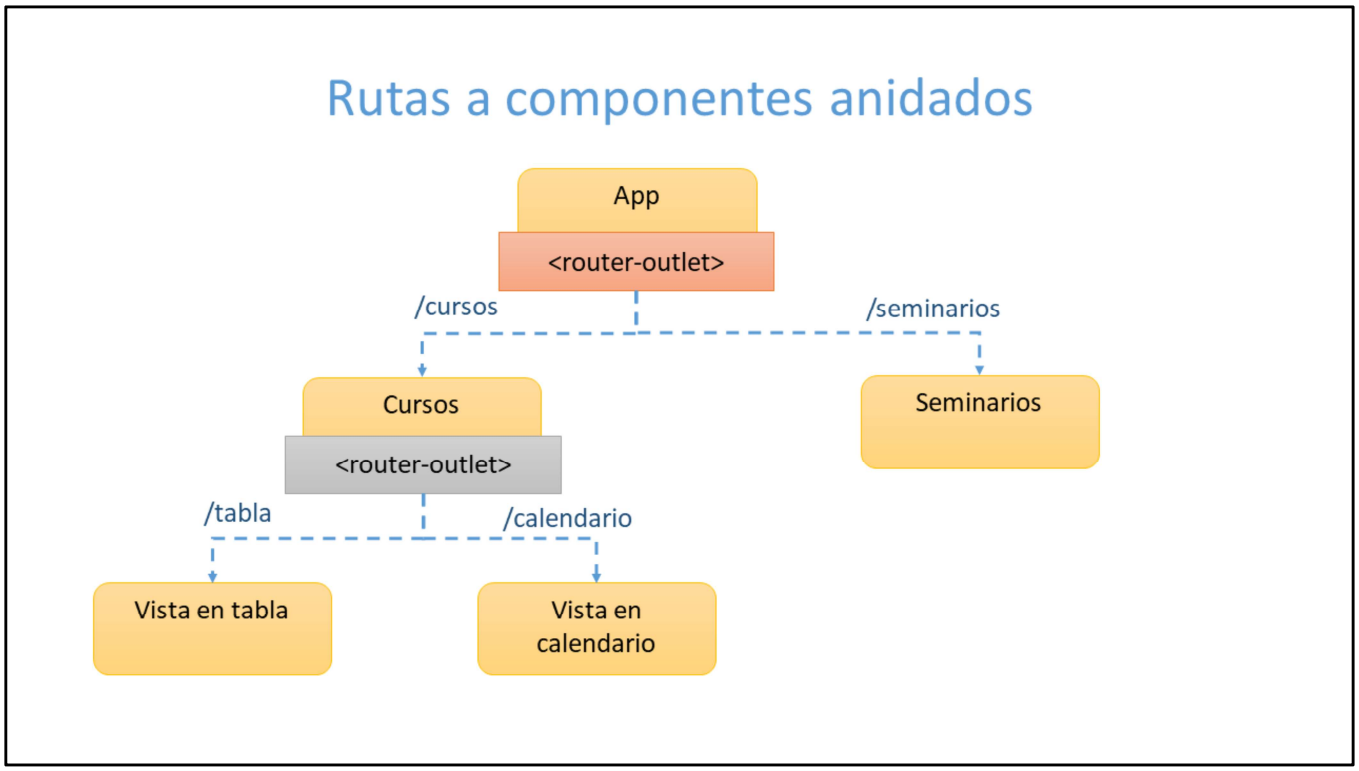
```
<a href="/cursos"> Cursos </a>
```

```
<a routerLink="/cursos"> Cursos </a>
```

En Angular no se debe usar el atributo "href" del elemento <a>. Aunque funciona, se vuelve a recargar toda la aplicación Angular, con todos los recursos asociados.

En su lugar se debe usar la directiva “routerLink”.

Probar ambos métodos y ver la diferencia en las llamadas a la red.



Supongamos que el componente Cursos, a su vez tiene un menú y una zona en la que carga o bien la lista de cursos o un calendario. Necesitaríamos configurar otro router-outlet, y asignar una ruta a cada uno de los componentes hijos.

Si configuramos las rutas /cursos/tabla y /cursos/calendario, igual que hemos configurado /cursos" y "seminario", todos los componentes usarán el router-outlet de AppComponent.

Rutas a componentes anidados

app.module.ts

```
{
    path: 'cursos',
    component: CursosComponent,
    children: [
      {
        path: 'tabla',
        component: CursosTablaComponent
      },
      {
        path: 'calendario',
        component: CursosCalendarioComponent
      }
    ]
  }
```

Se carga en el router-outlet de **AppComponent**

Se cargan en el router-outlet de **CursosComponent**

Para que los componentes hijos de “Cursos” se inserten en el router-outlet de Cursos, se deben incluir como rutas “hijas”, con el atributo “children”, que contiene un array de rutas hijas.

O dicho de otro modo, los componentes asociados a las rutas hijas se cargan en el router outlet del componente asociado a la ruta principal.

Lazy loading de módulos mediante rutas

- *Carga normal*: Se descargan todos los módulos al arrancar la aplicación.
- ***Lazy loading***: Los módulos se descargan solo cuando van a ser usados.
  - Los módulos que usan la carga "perezosa" se configuran en el RouterModule

Por defecto, en una aplicación Angular, aunque se haya estructurado en diferentes modulos, al cargar cualquiera de las rutas se van a descargar al cliente todos los módulos, independientemente de que se vayan o no a utilizar.

Esto no es bueno para el tiempo de arranque de la aplicación, sobre todo si la aplicación tiene muchos componentes y modulos.

Se denomina "Lazy loading" o "Carga perezosa" de un módulo de Angular, al mecanismo por el cual un modulo solo se decarga al cliente cuando se va a utilizar alguno de sus componentes

Lazy loading de módulos mediante rutas

RouterModule de **app.module.ts**

```
RouterModule.forRoot( [
{
    path: 'administracion',
    loadChildren: './admin/admin.module#AdminModule'
}
] )
```

RouterModule de **admin.module.ts**

```
RouterModule.forChild( [
 {
   path: 'usuarios',
   component: GestionUsuariosComponent
 },
 {
    path: 'tablas_maestras',
    component: TablasMaestrasComponent
 }
] )
```

El módulo AdminModule **no se descargará** hasta que no se active la ruta /administración/usuarios o /administración/tablas_maestras

La forma de indicar que un modulo debe ser cargado de forma “lazy”, es diciéndole a Angular que cargue la configuración para una ruta a partir de dicho módulo.

Observar que en el módulo App se usa el método Router.forRoot(), mientras que en el resto de los módulos se usa el método RouterModule.forChild().

Gestionar rutas no existentes

- Las rutas no existentes en nuestra aplicación, deberían mostrar un componente específico, o bien redirigir a una página conocida (por ejemplo la de inicio).

- Si se solicita una ruta no configurada en el RouterModule, se produce un error en la consola.

- Para personalizar la información mostrada:
  1. Crear un componente para “página o recurso no encontrado”
  2. Asociar la ruta “**” a dicho componente, o redirigir a otra ruta
  3. Ubicar la configuración de la ruta “**” en el último lugar

Usar parámetros en las rutas

- Parámetros = paso de información a través de la ruta
- Tipos:
  - **Parámetros en el "path"** /cursos/45
    - Forman parte de la ruta
  - **Parámetros en la "query"** /cursos/detalle?idCurso=45
    - No forman parte de la ruta
    - Son opcionales

Los parámetros en las URL,s. son una forma muy habitual de pasar información a una página web.

El módulo de enrutado de Angular soporta configurar rutas con parámetros.

Los parámetros pueden ser de 2 tipos:

- Parámetros en el path
- Parámetros en la query

Configurar rutas con parámetros

app.module.ts

```
{
    path: 'cursos',
    component: CursosComponent
},
{
    path: 'cursos/detalle',
    component: CursosDetalleComponent
},
{
    path: 'cursos/:idCurso',
    component: CursosDetalleComponent
},
```

El parámetro de tipo QUERY NO forma parte de la ruta

El parámetro de tipo PATH forma parte de la ruta

Recoger el valor de los parámetros en el componente

1. Inyectar en el componente un objeto de tipo ActivatedRoute
   - ➢ Contiene información de la ruta activa en el Router
2. Para los parámetros de tipo “path”:
   - Atributo “paramMap”: mapa de parámetros (nombre – valor)
3. Para los parámetros tipo “query”:
   - Atributo “queryParamMap”: mapa de parámetros (nombre – valor)

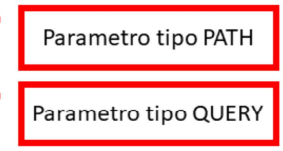
Para acceder desde el componente a los parámetros de la ruta, es necesario inyectar un objeto de la clase "ActivatedRoute“, como si fuera un servicio.

Este objeto tiene una propiedad "paramMap", que es un observable al que podemos suscribirnos para acceder a los parámetros en forma de mapa

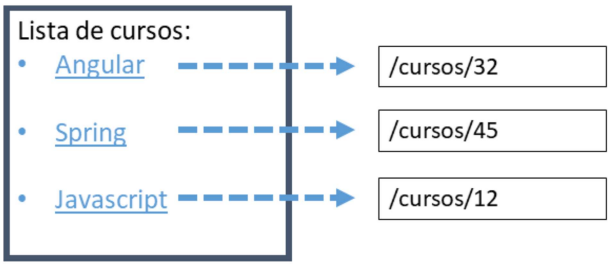
Recoger el valor de los parámetros en el componente

cursos-detalle.component.ts

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

...

constructor(private ruta: ActivatedRoute) { }
  ...

ngOnInit() {
   let idCurso =
       this.ruta.snapshot.paramMap.get("idCurso");

   let idCurso =
       this.ruta.snapshot.queryParamMap.get("idCurso");
}
```



Crear enlaces con parámetros dinámicos




```
<h1> Lista de cursos </h1>
<ul>
  <li *ngFor="let curso of cursos">
      <a [routerLink]="['/cursos', curso.id ]"</a>
  </li>
</ul>
```

[routerLink]= "[ruta_base, parametro1, parametro2..., { queryParam1: valor, queryParam2: valor...}]"

Hemos visto anteriormente que se usa la directiva "routerLink" para establecer la ruta de un enlace o de un botón HTML.

Imaginemos que en la página de la lista de cursos, queremos que cada curso tenga un enlace al detalle del curso, y queremos pasar el id como parámetro de la ruta.

En ese caso, el valor del parámetro es dinámico, y no podemos asignarlo de forma estática.

Para asignarlo de forma dinámica, usamos el mismo atributo routerLink, pero haciendo property binding, y asignándole una expresión especial.

OJO a los corchetes en "routerLink"

Navegación programática

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

...

constructor(private router: Router) { }

navegarPorPrograma() {
    this.router.navigate(
       ['/cursos/tabla', {page: 1, orden: "ascendente" }] );
}
```

En ocasiones necesitaremos llamar a una ruta directamente desde el código del componente, es decir, desde la capa de controlador, en lugar desde la plantilla o vista.

Por ejemplo, cuando queremos navegar a una u otra ruta en base a ciertas validaciones, o cuando queremos hacer una redirección a la página de login si el usuario no está autenticado.

Para navegar desde el código del componente, en lugar desde los links de la plantilla HTML, es necesario inyectar en el constructor un objeto de tipo "Router", y usar el método "navigate()"