



**Florida**  
Universitària

# Navegación multi- página. Routing.

Curso 2025/2026

Paco Segura

# Navegación multi-página

- SPA (single page applications) -> Ejecutan todo su contenido en una sola página.
  - ¿Cómo? -> Al pasar de una sección a otra, sólo se carga dinámicamente el contenido nuevo.
  - En una página podemos tener distintas vistas para cada apartado de la web.
  - Ventajas: navegación fluida, más facilidad para su creación y diseño.
-

# Navegación multi-página

- Veamos un ejemplo:

La Ciudad de las Artes y de las Ciencias

Edificios

[Hemisferic](#)  
[Museo de las Ciencias](#)



Museu de les Arts

Ubicada al final del viejo cauce del río Turia, la Ciudad de las Artes y las Ciencias es un complejo arquitectónico, cultural y de entretenimiento diseñado por Santiago Calatrava y Félix Candela, junto con los ingenieros autores del diseño estructural de las cubiertas del L'Oceanografic Alberto Domingo y Carlos Lázaro.

La Ciudad de las Artes y de las Ciencias

Edificios

[Hemisferic](#)  
[Museo de las Ciencias](#)



Hemisferic

Ubicada al final del viejo cauce del río Turia, la Ciudad de las Artes y las Ciencias es un complejo arquitectónico, cultural y de entretenimiento diseñado por Santiago Calatrava y Félix Candela, junto con los ingenieros autores del diseño estructural de las cubiertas del L'Oceanografic Alberto Domingo y Carlos Lázaro.

# Routing

- Para implementar navegación entre vistas en Angular:
    - Cada vista es el contenido que varía en la página. El resto se mantiene.
    - Con las vistas trabajamos como con los componentes. Las ubicamos en una carpeta **views** ->  
ng generate component views/home
-

# Routing

- En app.routes.ts:

```
import { Routes } from '@angular/router';  
import { View1 } from '../views/view1/view1';  
import { View2 } from '../views/view2/view2';
```

1º Importamos las vistas

```
export const routes: Routes = [  
  { path: '', redirectTo: 'view1', pathMatch: 'full' },  
  { path: 'view1', component: View1 },  
  { path: 'view2', component: View2 },  
];
```

2º Indicamos las rutas. Con path definimos la ruta de la url en que se cargará la vista. Si indicamos un path vacío, cuando se cargue el navegador en localhost, se redireccionará a la vista que indiquemos (view1 en el ejemplo).

# Routing

- Para indicar dónde irán las vistas -el contenido que varía en la página-, se utiliza **<router-outlet></router-outlet>**. en el archivo app.component.html:

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



Indica donde se ubicarán las Vistas de la aplicación.

# Routing

- Para insertar un link a una vista desde HTML, primero importamos **routerLink** y **routerLinkActive** en el archivo \*.ts:

```
import { NgClass } from '@angular/common';
import { Component } from '@angular/core';
import { RouterLink, RouterLinkActive } from '@angular/router';

@Component({
  selector: 'app-menu',
  imports: [NgClass, RouterLink, RouterLinkActive],
  templateUrl: './menu.html',
  styleUrls: ['./menu.css'],
})
export class Menu {
```

# Routing

- En el archivo component.html indicamos la ruta definida anteriormente en **path** con **routerLink**. También definimos **routerLinkActive**.

```
<nav [ngClass]="case">
  <ul>
    <li><a class="menulink" routerLink="/view1" routerLinkActive="active" >{{links[0]}}</a></li>
    <li><a class="menulink" routerLink="/view2" routerLinkActive="active" >{{links[1]}}</a></li>
  </ul>
</nav>
```