Guía de Presentación: Tema 07 - Enrutamiento (Router)

Objetivo de la charla: Que la clase entienda cómo crear una web con varias "páginas" sin que el navegador se refresque nunca.

1. Introducción: ¿La magia de la SPA?

Explicación sencilla: "Imaginad que vuestra web es un marco de fotos digital. El marco (el navegador) siempre es el mismo, no cambia. Lo único que cambia es la foto que hay dentro (el Componente). El **Router** es el mecanismo que decide qué foto poner en el marco según la dirección que escribas arriba."

- **SPA (Single Page Application):** No pedimos un HTML nuevo al servidor cada vez que hacemos clic. Solo cambiamos trozos de la pantalla. Eso lo hace instantáneo.

---

2. El Mapa: Definiendo las Rutas (`app.routes.ts`)

Para que Angular sepa qué mostrar, necesita un mapa. En las versiones modernas (Standalone), esto se hace en el archivo `app.routes.ts`.

Código para mostrar en pantalla:

```
import { Routes } from '@angular/router';
import { InicioComponent } from './paginas/inicio/inicio.component';
import { ContactoComponent } from './paginas/contacto/contacto.component';
import { Error404Component } from './paginas/error404/error404.component';

export const routes: Routes = [
  // 1. Ruta básica: Si la URL es /inicio, carga InicioComponent
  { path: 'inicio', component: InicioComponent },

  // 2. Ruta básica: Si la URL es /contacto, carga ContactoComponent
  { path: 'contacto', component: ContactoComponent },

  // 3. Redirección: Si no ponen nada (ruta vacía), llévalos a inicio
  { path: '', redirectTo: 'inicio', pathMatch: 'full' },

  // 4. Ruta comodín (**): Si escriben algo raro, muestra error
  // IMPORTANTE: Esta siempre debe ir al final
  { path: '**', component: Error404Component }
];
```

Puntos clave a explicar:

- `path`: Es lo que sale en la barra del navegador.
- `component`: Es el componente que queremos pegar en la pantalla.
- `**`: Es el "cajón de sastre". Si el usuario escribe `/loquesea` y no existe, va aquí.

---

3. El Hueco: `<router-outlet>`

Ya tenemos el mapa, pero ¿dónde se "pintan" esos componentes? Normalmente en el `app.component.html`.

Código (`app.component.html`):

```
<header>
  <h1>Mi Súper Aplicación</h1>
  </header>

<main>
  <router-outlet />
</main>

<footer>
  <p>Pie de página fijo (siempre visible)</p>
</footer>
```

Nota para la clase: "Fijaos que el *Header* y el *Footer* **no** se recargan. Solo cambia lo que hay donde está el `<router-outlet />`. Para que esto funcione, tenéis que importar `RouterOutlet` en los `imports` del `AppComponent`".

---

4. La Navegación: `routerLink` vs `href`

Este es el error número 1 de los novatos.

- ❌ `<a href="/inicio">`: **MAL.** Esto fuerza al navegador a recargar toda la página (pantallazo blanco). Perdemos la velocidad de Angular.
- ☑ `<a routerLink="/inicio">`: **BIEN.** Angular intercepta el clic y cambia el componente instantáneamente.

Código (`app.component.html` - Menú):

```
<nav>
  <a routerLink="/inicio" routerLinkActive="activo">Ir a Inicio</a>

  <a routerLink="/contacto" routerLinkActive="activo">Ir a Contacto</a>
</nav>
```

Importante: Recordad importar `RouterLink` y `RouterLinkActive` en el array de `imports` del componente donde esté el menú.

---

5. Rutas con Parámetros (Input Binding) - *Nivel Pro Simplificado*

A veces necesitamos pasar datos, como ver el perfil de un usuario concreto: `/usuario/5` o `/producto/zapatillas`.

En Angular 17+ esto es facilísimo si activamos una opción mágica llamada `withComponentInputBinding`.

Paso A: En `app.config.ts` (Configuración global)

```
import { provideRouter, withComponentInputBinding } from '@angular/router';
import { routes } from './app.routes';

export const appConfig: ApplicationConfig = {
  providers: [
    // Activamos withComponentInputBinding para recibir datos como Inputs
    provideRouter(routes, withComponentInputBinding())
  ]
};
```

Paso B: En las rutas (`app.routes.ts`)

```
// Los dos puntos (:) indican que 'idUsuario' es dinámico (puede cambiar)
{ path: 'usuario/:idUsuario', component: PerfilUsuarioComponent }
```

Paso C: En el componente (`perfil-usuario.component.ts`) "Simplemente usamos un `@Input`. Angular coge el dato de la URL y lo mete en la variable automáticamente. ¡Magia!"

```
@Component({ ... })
export class PerfilUsuarioComponent {
  // El nombre de la variable debe coincidir con el del path (:idUsuario)
  @Input() idUsuario!: string;

  // Opcional: Podéis usarlo en el HTML así:
  // <h2>Perfil del usuario {{ idUsuario }}</h2>
}
```

---

6. Navegar desde el Código (TypeScript)

A veces no navegamos haciendo clic en un enlace, sino después de que pase algo (ej: el usuario rellena un formulario de Login y, si es correcto, lo mandamos al Inicio).

Aquí usamos **Inyección de Dependencias** (Tema 05) con `inject()`.

Código ejemplo:

```typescript
import { Component, inject } from '@angular/core';
import { Router } from '@angular/router';

@Component({ ... })
export class LoginComponent {
  // Inyectamos el servicio Router (nuestro GPS)
  private _enrutador = inject(Router);

  iniciarSesion() {
    // Aquí iría la lógica de comprobar usuario/contraseña...
    console.log("Login correcto");

    // Redirigimos al usuario a la página de inicio
    this._enrutador.navigate(['/inicio']);
  }
}
```

Resumen para vuestros compañeros (La "Chuleta")

Para cerrar la presentación, ponedles esta lista de verificación. Es lo que necesitan para aprobar la práctica:

1. **Definir rutas** en `app.routes.ts` (`path` y `component`).
2. **Poner el hueco** `<router-outlet />` en el `AppComponent`.
3. **Crear enlaces** con `routerLink="/ruta"` (nunca `href`).
4. **Importar** `RouterOutlet` y `RouterLink` en los componentes donde los uséis.
5. **Para redirigir** desde código: `inject(Router)` y `.navigate(['/ruta'])`.