

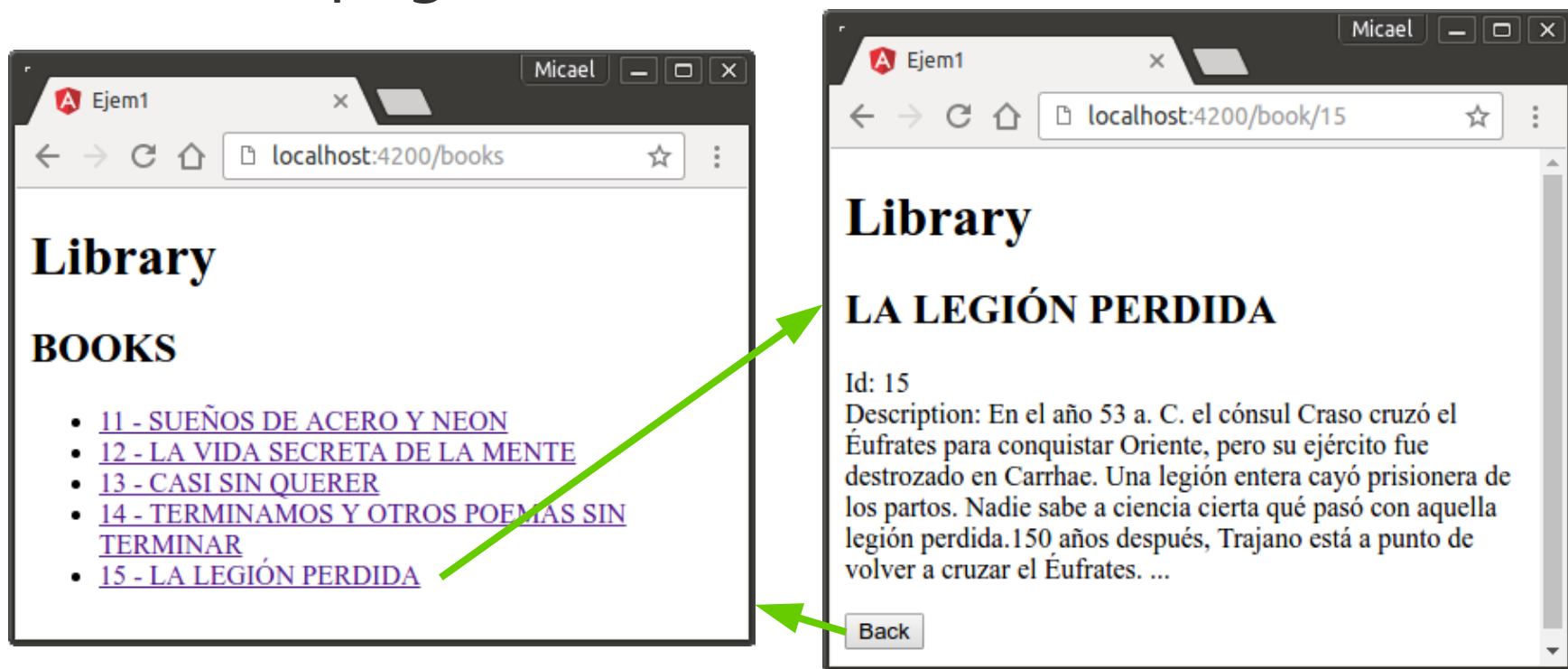
Desarrollo Web SPA con Angular

4. Aplicaciones Multipágina - Router



Aplicaciones multipágina: Router

- Las webs SPA (*single page application*) pueden tener **varias pantallas** simulando la **navegación** por diferentes páginas



<https://angular.io/docs/ts/latest/guide/router.html>

Aplicaciones multipágina: Router

- El **componente principal** de la aplicación (**app-root**) tiene una parte fija (**cabecera, footer**) y una parte cuyo **contenido** depende de la **URL** (**<router-outlet>**)
- En **app.routing.ts** se define qué **componente** se muestra en el **router-outlet** para cada **URL**
- Existen **links especiales** para navegar dentro de la aplicación web (**[routerLink]**)
- Desde el código se puede navegar de forma automática (**Router**)

<https://angular.io/docs/ts/latest/guide/router.html>

Componente principal

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1 class="title">Library</h1>
    <router-outlet></router-outlet>
  `
})
export class AppComponent { }
```

Componente principal

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: `
    <h1 class="title">Library</h1>
    <router-outlet></router-outlet>
  `
})
export class AppComponent { }
```

Zona que cambia en
función de la URL

Configuración de las rutas

app.routing.ts

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

import { BookListComponent } from './book-list.component';
import { BookDetailComponent } from './book-detail.component';

const appRoutes = [
  { path: 'book/:id', component: BookDetailComponent, },
  { path: 'books', component: BookListComponent },
  { path: '', redirectTo: 'books', pathMatch: 'full' }
]

export const routing = RouterModule.forRoot(appRoutes);
```

Configuración de las rutas

app.routing.ts

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

import { BookListComponent } from './book-list.component';
import { BookDetailComponent } from './book-detail.component';

const appRoutes = [
  { path: 'book/:id', component: BookDetailComponent, },
  { path: 'books', component: BookListComponent },
  { path: '', redirectTo: 'books', pathMatch: 'full' }
]

export const routerConfig = [RouterModule.forRoot(appRoutes)];
```

Para cada URL se indica el **componente** que será visualizado en el **router-outlet**

Configuración de las rutas

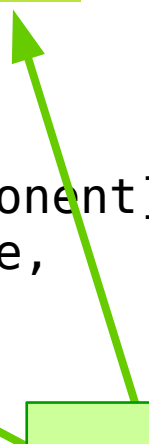
app.module.ts

```
...  
import { routing } from './app.routing';  
  
@NgModule({  
  declarations: [AppComponent,  
    BookDetailComponent, BookListComponent],  
  imports: [BrowserModule, FormsModule,  
    HttpClientModule, routing],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```


Configuración de las rutas

app.module.ts

```
...  
import { routing } from './app.routing';  
  
@NgModule({  
  declarations: [AppComponent,  
    BookDetailComponent, BookListComponent],  
  imports: [BrowserModule, FormsModule,  
    HttpClientModule, routing],  
  bootstrap: [AppComponent]  
})  
export class AppModule { }
```



Las rutas se consideran un módulo que debe importarse en la aplicación

Aplicaciones multipágina: Router

book-list.component.ts

BookListComponent

```
...
@Component({
  template: `
    <h2>BOOKS</h2>
    <ul>
      <li *ngFor="let book of books">
        <a [routerLink]="['/book',book.id]">
          {{book.id}}-{{book.title}}
        </a>
      </li>
    </ul>`
})
export class BookListComponent {

  books: Book[];

  constructor(service: BookService) {
    this.books = service.getBooks();
  }
}
```

Aplicaciones multipágina: Router

book-list.component.ts

BookListComponent

```
...
@Component({
  template: `
    <h2>BOOKS</h2>
    <ul>
      <li *ngFor="let book of books">
        <a [routerLink]="['/book',book.id]">
          {{book.id}}-{{book.title}}
        </a>
      </li>
    </ul>`
})
export class BookListComponent {

  books: Book[];

  constructor(service: BookService) {
    this.books = service.getBooks();
  }
}
```

En vez de **href**, los links usan **[routerLink]**. La URL se puede indicar como un string (**completa**) o como un **array** de strings si hay **parámetros**

Aplicaciones multipágina: Router

book-detail.component.ts

BookDetailComponent

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

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button></p>`
})
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {

    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Aplicaciones multipágina: Router

book-detail.component.ts

BookDetailComponent

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

@Component({
  template: `<h2>{{book.title}}</h2>
    <div><label>Id: </label>{{book.id}}</div>
    <div><label>Description: </label>{{book.description}}</div>
    <p><button (click)="gotoBooks()">Back</button></p>
  `)
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {

    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Para acceder a los parámetros desde el componente usamos el servicio **ActivatedRoute**

Aplicaciones multipágina: Router

book-detail.component.ts

BookDetailComponent

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

@Component({
  template: `<h2>{{book.title}}</h2>
    <div><label>Id: </label>{{book.id}}</div>
    <div><label>Description: </label>{{book.description}}</div>
    <p><button (click)="gotoBooks()">Back</button></p>
  `)
export class BookDetailComponent {
  book: Book;

  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {

    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Obtenemos un **snapshot** de los parámetros y accedemos al parámetro **id**

Aplicaciones multipágina: Router

book-detail.component.ts

BookDetailComponent

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

@Component({
  template: `<h2>{{book.title}}</h2>
<div><label>Id: </label>{{book.id}}</div>
<div><label>Description: </label>{{book.description}}</div>
<p><button (click)="gotoBooks()">Back</button></p>`
})
export class BookDetailComponent {
  book: Book;

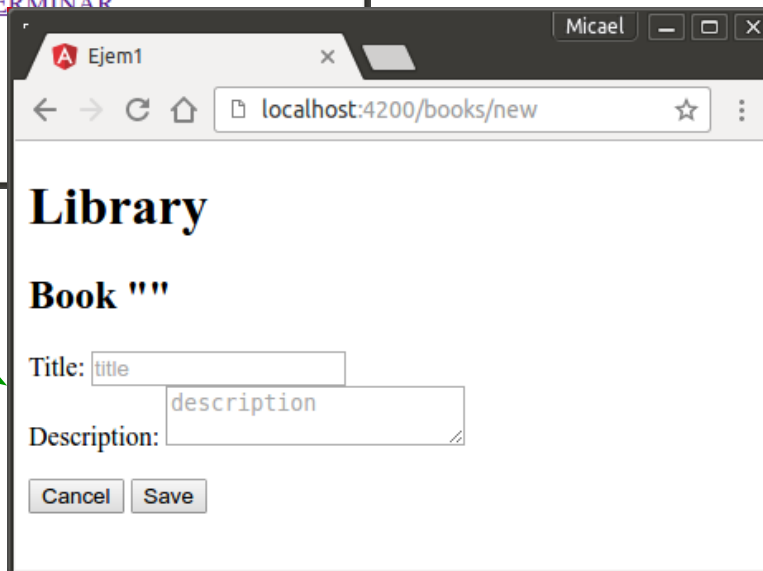
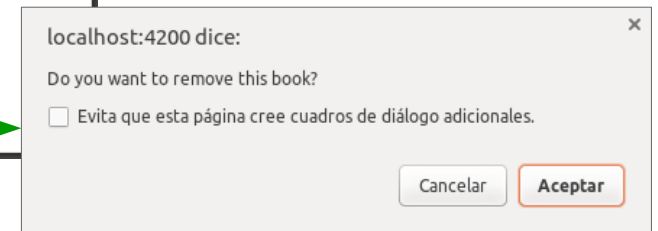
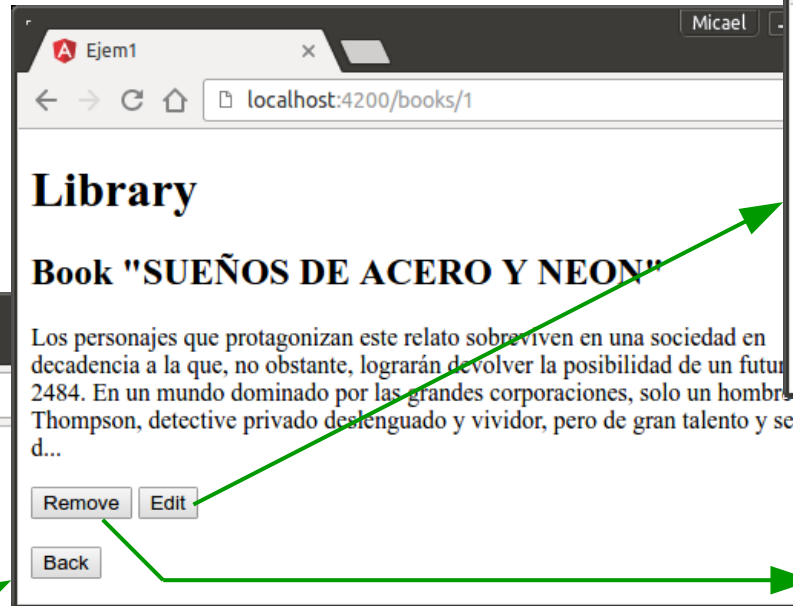
  constructor(private router: Router,
    activatedRoute: ActivatedRoute, service: BookService) {

    let id = activatedRoute.snapshot.params['id'];
    this.book = service.getBook(id);
  }
  gotoBooks() { this.router.navigate(['/books']); }
}
```

Para navegar desde código
usamos el servicio **Router** y el
método **navigate**

Ejercicio 6

- Implementa una **aplicación CRUD** de gestión de **libros**
- Funcionalidades (**varias pantallas**)
 - Listado de todos los libros (títulos)
 - Formulario de nuevo libro
 - Vista de detalle de un libro
 - Modificación de libro
 - Borrado de un libro
- Se proporciona una **API REST** (similar a la de los items pero para books).
- Cada libro tiene las propiedades: id, title, description



- **Navegar al mismo componente con otros datos**
 - En algunas apps se navega al mismo componente pero con otros datos en la URL
 - En ese caso, se reutiliza la misma instancia del componente, no se crea una nueva
 - La forma de acceder a los parámetros de la URL es diferente para que podamos actualizar el componente cuando cambia la URL

- Navegar al mismo componente con otros datos

```
constructor(private router: Router, activatedRoute:  
  ActivatedRoute, private service: BookService) {  
  
  const id = activatedRoute.snapshot.params['id'];  
  this.book = service.getBook(id);  
}
```



```
constructor(private router: Router, activatedRoute:  
  ActivatedRoute, private service: BookService) {  
  
  activatedRoute.params.subscribe(params => {  
    const id = params['id'];  
    this.book = this.service.getBook(id);  
  });  
}
```

- **Funcionalidades avanzadas**

- Rutas hijas (child routes) que se aplican a un componente concreto (en vez de las aplicadas a toda la app)
- Ejecutar código al salir de una pantalla
 - Si el usuario navega a otra página “sin guardar” se le puede preguntar si realmente desea descartar los cambios o abortar la navegación
- Verificar si se puede ir a una nueva pantalla
 - Generalmente se comprueba si el usuario tiene permisos para hacerlo
- Las pantallas sólo se cargan al acceder a ellas (*lazy loading*)
- Animaciones en las transiciones entre pantallas

<https://angular.io/docs/ts/latest/guide/router.html>