3-Template

Componentes dinámicos, directivas y pipes

```
# 1. Plantillas de contenido dinámico

# 2. Atributos custom con Directivas

# 3. Funciones de transformación con Pipes
```

1 Plantillas de contenido dinámico

```
As a: customer,
  I want: to see a product card with price in euros
  so that: i can decide to purchase it or not

As a: seller,
  I want: to see a product card with stock
  so that: I can ask for more or not
```

```
ng g @nrwl/angular:library products --prefix=ab-products
# "@a-boss/products": ["libs/products/src/index.ts"]
```

1.1 Un componente común

```
ng g c product-template --project=products --module=products.module.ts --export
```

> La visión del comprador y del vendedor es parecida. Mantenemos estructura, inyectamos contenido.

```
<article style="margin: 5px; padding: 5px; border: 2px; border-style: solid;">
  <header>
    <h2>
      {{ product.description }}
    </h2>
  </header>
  <main>
    <ng-content select="main"></ng-content>
```

```
    </main>
    <footer style="margin-top: 5px">
      <ng-content select=".actions"></ng-content>
    </footer>
  </article>
```

La directiva `ngContent` permite crear *slots* para incrustar contenido a voluntad del consumidor. Cada *slot* se identifica mediante un `select="css-selector"`.

1.2 Implementaciones distintas

```
ng g m catalog --project=shop --module=app.module.ts --routing --route=catalog
ng g c catalog/product --project=shop

# imports: [ProductsModule]
# <a [routerLink]="['/catalog']">Catalog</a>
```

apps\shop\src\app\catalog\catalog.component.ts

```
export class CatalogComponent implements OnInit {
  public catalog$: Observable<Product[]>;
  constructor(private http: HttpClient) {}

  ngOnInit() {
    this.catalog$ = this.http.get<Product[]>('./assets/data/products.json');
  }
}
```

apps\shop\src\app\catalog\catalog.component.html

```
<section *ngIf="catalog$ | async as catalog">
  <ab-shop-product *ngFor="let product of catalog"
                   [product]="product"></ab-shop-product>
</section>
```

apps\shop\src\app\catalog\catalog.component.html

```
<ab-products-product-template [product]="product">
  <main>
    <div>
      {{ product.brand }} - {{ product.category }}
    </div>
    <div>
      Price: {{ product.price }}
```

```
    </div>
  </main>
  <nav class="actions">
    <button (clic)="buy.next()"
            style="background-color:coral; padding: 5px"><strong>Buy me!</strong>
</button>
  </nav>
</ab-products-product-template>
```

2 Atributos custom con Directivas

```
As a: seller,
  I want: to see a green mark on products with stock
  so that: I know I don't do need to refill

As a: seller,
  I want: to see a red mark on products with out stock
  so that: I know I need to refill
```

2.1 Generación de directivas

```
ng g directive out-of-stock --project=products --export
```

```
@Directive({
  selector: '[abProductsOutOfStock]'
})
export class OutOfStockDirective {
  private minimalStock = 10;

  @Input()
  set abProductsOutOfStock(stock: number) {
    const color = stock <= this.minimalStock ? 'MistyRose' : 'Aquamarine';
    this.el.nativeElement.style.backgroundColor = color;
  }

  constructor(private el: ElementRef) {}
}
```

2.2 Consumo de directivas

```
@NgModule({
  imports: [CommonModule],
  declarations: [ProductTemplateComponent, OutOfStockDirective],
  exports: [ProductTemplateComponent, OutOfStockDirective]
})
export class ProductsModule {}
```

apps\shop\src\app\catalog\product\product.component.html

```
  <div [abProductsOutOfStock]="product.stock">
    Stock: {{ product.stock }}
  </div>
```

3 Funciones de transformación con Pipes

```
As a: customer,
  I want: to see a product price also in dollars
  so that: I can compare prices
As a: customer,
  I want: to see a product price also in pounds
  so that: I can compare prices
```

3.1 Generación de pipes

```
ng g pipe exRate --project=products --export
```

```
@Pipe({
  name: 'exRate'
})
export class ExRatePipe implements PipeTransform {
  private readonly euroDollars = 1.13;
  private readonly ratesApi = 'https://api.exchangeratesapi.io/latest?symbols=';

  constructor(private httpClient: HttpClient) {}

  public transform(euros: number, symbol: string): number | Observable<number> {
    if (!symbol) {
      return euros * this.euroDollars;
    } else {
      return this.getOnlineRates$(symbol).pipe(map(rate => euros * rate));
```

```
    }
  }

  private getOnlineRates$(symbol: string) {
    const ratesUrl = this.ratesApi + symbol;
    return this.httpClient.get<any>(ratesUrl).pipe(
      shareReplay(1),
      refCount(),
      map(resp => resp.rates[symbol])
    );
  }
}
```

3.2 Consumo de pipes

apps\shop\src\app\catalog\product\product.component.html

```
<div>
  ${{ product.price | exRate | number:'1.0-0'}}
  {{ product.price | exRate:'GBP' | async | number:'1.0-0'}} £
</div>
```

> **Blog de apoyo:** Componentes dinámicos, directivas y pipes con Angular
>
> > By Alberto Basalo

> Next:

Redux con observables RxJs

Arquitectura del patrón Redux

Implementación de un Store con RxJs