Reto 2.6 - Lista de guitarras

Yoël Gómez Benítez
DAH - 2º DAW
17/10/2025

Índice

1. **Introducción**
   - 1.1 Objetivo general del proyecto
   - 1.2 Descripción de la estructura funcional de componentes

2. **Creación de la Interfaz de Datos**
   - 2.1 Uso del comando ionic generate interface
   - 2.2 Definición de la estructura del elemento principal
   - 2.3 Ejemplos de posibles interfaces (Producto, Noticia, Receta, etc.)

3. **Creación de un Componente Reutilizable**
   - 3.1 Uso del comando ionic generate component
   - 3.2 Estructura y propósito del componente hijo
   - 3.3 Ejemplo: producto-item o noticia-card

4. **Definición de la Estructura de la Interfaz**

5. **Uso de la Interfaz en los Componentes**
   - 5.1 Implementación en la lógica de los componentes
   - 5.2 Declaración de variables como propiedades de clase

6. **Configuración del Componente como Standalone**
   - 6.1 Añadir manualmente standalone: true
   - 6.2 Código inicial del archivo list-item.component.ts

7. **Data Binding: Conectando Lógica y Vista**
   - 7.1 La vista: task-item.component.html
   - 7.2 La lógica: home.page.ts
   - 7.3 Uso de directivas *ngIf y *ngFor
   - 7.4 Importación de módulos y componentes necesarios

8. **La Vista en home.page.html**

9. **Integración Completa del Proyecto**

10. **Resultado Final**

11. **Mejoras del Proyecto**
    - 11.1 Mejora 2 – Crear un Componente de Cabecera
    - 11.1.1 Crear el componente app-header.component.ts
    - 11.1.2 Marcarlo como standalone
    - 11.1.3 Vista app-header.component.html

11.1.4 Usar el nuevo componente en home.page.ts
11.1.5 Reemplazar <ion-header> por <app-header> en home.page.html
11.2 Mejora 3 – Usar Pipes de Angular
11.2.1 Aplicar Pipe en la lista de guitarras (home.page.html)
11.2.2 Ejemplo: mostrar nombre en mayúsculas
11.3 Mejora 4 – Componente Hijo con Manejo de Datos Opcionales
11.3.1 Generar el componente hijo guitarra-item.component.ts
11.3.2 Vista guitarra-item.component.html
11.3.3 Añadir el componente a home.page.ts
11.3.4 Usar el componente hijo en home.page.html (reemplazar bucle)

1. **Introducción:**

   El objetivo es construir una estructura de componentes funcional que muestre una lista de datos estáticos relacionados con la temática de tu proyecto. Para ello, debes:

2. **Crear tu Interfaz de Datos:**

   Usando el comando ionic generate interface, crea una interfaz que defina la estructura del elemento principal de tu app (ej: Producto, Noticia, Receta, etc.).

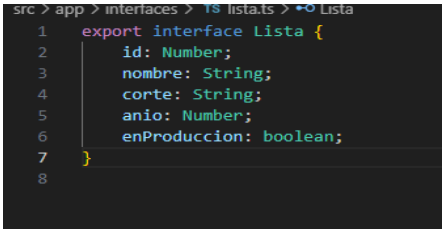
```
C:\Users\9_2DAW\mi-app-guitar>ionic generate interface interfaces/lista
> ng.cmd generate interface interfaces/lista --project=app

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

   No
Global setting: not set
Local setting: disabled
Effective status: disabled
CREATE src/app/interfaces/lista.ts (29 bytes)
[OK] Generated interface!

C:\Users\9_2DAW\mi-app-guitar>
```

3. **Crear un Componente Reutilizable:**

   Usando ionic generate component, crea un componente "hijo" que sirva para mostrar un único elemento de tu lista (ej: producto-item, noticia-card).

```
C:\Users\9_2DAW\mi-app-guitar>ionic generate component components/list-item
> ng.cmd generate component components/list-item --project=app
CREATE src/app/components/list-item/list-item.component.html (31 bytes)
CREATE src/app/components/list-item/list-item.component.spec.ts (735 bytes)
CREATE src/app/components/list-item/list-item.component.ts (294 bytes)
CREATE src/app/components/list-item/list-item.component.scss (0 bytes)
[OK] Generated component!
```

4. **Definimos la estructura de la interfaz**



5. **Paso 3: Usar nuestra nueva interfaz**

   Una vez definida la interfaz, podemos usarla en nuestros componentes para definir los datos que manejará la vista. Las variables que contienen los datos de un componente se declaran como propiedades de su clase (y por tanto, sin `let`).

```ts
// Dentro de src/app/home/home.page.ts
import { Component } from '@angular/core';
import { Lista } from 'src/app/interfaces/lista';

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss']
})
export class HomePage {

  listaDeGuitarras: Lista[] = [
    { id: 1, nombre: "Fender Stratocaster", corte: "Strat", anio: 1954, enProduccion: true },
    { id: 2, nombre: "Gibson Les Paul Standard", corte: "Single Cut", anio: 1952, enProduccion: true },
    { id: 3, nombre: "Ibanez RG550", corte: "Superstrat", anio: 1987, enProduccion: true },
    { id: 4, nombre: "Gibson SG", corte: "Double Cut", anio: 1961, enProduccion: true },
    { id: 5, nombre: "Fender Telecaster", corte: "Single Cut", anio: 1950, enProduccion: true },
    { id: 6, nombre: "Jackson Soloist", corte: "Superstrat", anio: 1984, enProduccion: true },
    { id: 7, nombre: "Gretsch White Falcon", corte: "Hollow Body", anio: 1955, enProduccion: false },
    { id: 8, nombre: "PRS Custom 24", corte: "Double Cut", anio: 1985, enProduccion: true }
  ];

  constructor() {}

}
```

6. **Configurar nuestro Componente como Standalone**

Cuando generamos un componente, la CLI no añade `standalone: true` por defecto. Debemos hacerlo nosotros manualmente. Este es el código inicial que define la lógica de nuestro list-item.component.ts.

```ts
// Contenido de: src/app/components/task-item/task-item.component.ts

import { Component, OnInit } from '@angular/core';
// Añadimos las importaciones adicionales que nuestro componente necesita
import { Input } from '@angular/core';
import { Lista } from 'src/app/interfaces/lista';
import { IonicModule } from '@ionic/angular';

@Component({
  selector: 'app-list-item',
  templateUrl: './list-item.component.html',
  styleUrls: ['./list-item.component.scss'],
  // Añadimos esta línea para que el componente pueda usar las etiquetas de Ionic en su HTML
  imports: [IonicModule]
})
export class ListItemComponent implements OnInit {

  // Propiedad de entrada para recibir los datos de la lista
  @Input() lista!: Lista;

  constructor() { }

  ngOnInit() {}

  // Método para mostrar los detalles de la lista en la consola
  mostrarDetalles() {
    console.log('Datos de la lista:', this.lista);
  }

}
```

7. Data Binding: Conectando Lógica y Vista

La Vista: task-item.component.html

```html
<!-- Contenido de: src/app/components/task-item/task-item.component.html -->

<ion-card *ngFor="let guitarra of listaDeGuitarras" [color]="guitarra.enProduccion ? 'primary' : 'light'">
  <ion-item>
    <ion-label>
      <h2>{{ guitarra.nombre }}</h2>
      <p>Corte: {{ guitarra.corte }}</p>
      <p>Año: {{ guitarra.anio }}</p>
      <p *ngIf="!guitarra.enProduccion" style="color: red; font-weight: bold;">
        Fuera de producción
      </p>
    </ion-label>

    <ion-button (click)="mostrarDetalles()" slot="end">
      Ver
    </ion-button>
  </ion-item>
</ion-card>
```

8. La Lógica en home.page.ts

Ahora, en la plantilla, usamos la estructura de cabecera con efecto de colapso, *ngIf para la condición de lista vacía y *ngFor para repetir nuestro componente.

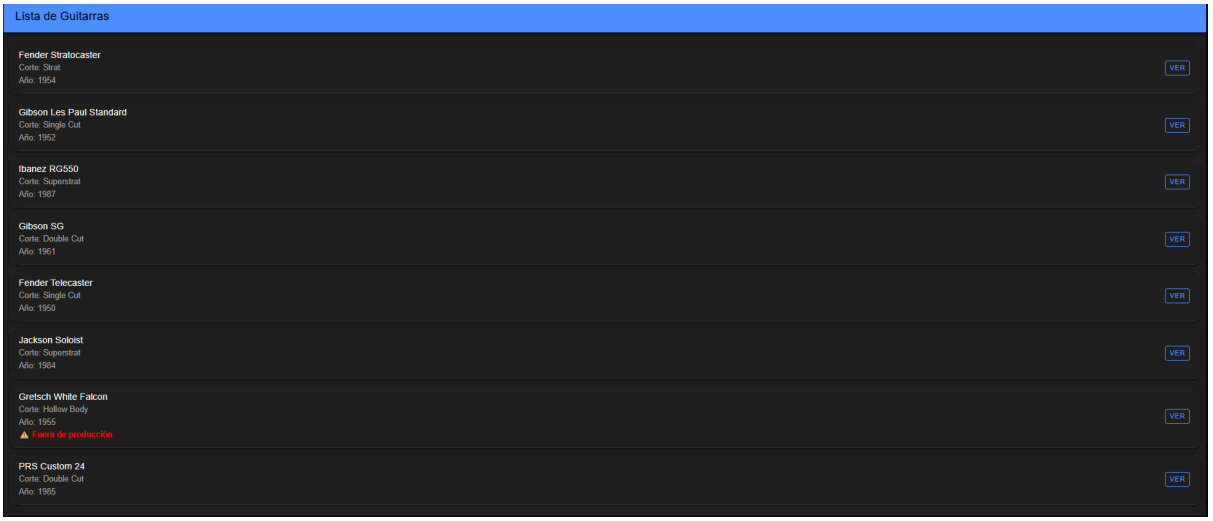
Nos aseguramos de que el componente HomePage importe todo lo que su plantilla va a necesitar: los componentes de Ionic, nuestro TaskItemComponent y el CommonModule (que contiene las directivas *ngIf y *ngFor).

```ts
// Imports básicos de Angular y Ionic
import { Component } from '@angular/core';
import { IonHeader, IonToolbar, IonTitle, IonContent } from '@ionic/angular/standalone';

// Imports de nuestra aplicación
import { Lista } from '../interfaces/lista'; // Nuestra interfaz de datos
import { ListItemComponent } from '../components/list-item/list-item.component'; // El componente hijo
import { CommonModule } from '@angular/common'; // Módulo necesario para usar *ngIf y *ngFor

@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
  imports: [
    IonHeader, IonToolbar, IonTitle, IonContent, // Componentes de Ionic
    ListItemComponent, // Nuestro componente hijo
    CommonModule // El módulo para las directivas
  ],
})
export class HomePage {
  listaDeGuitarras: Lista[] = [
    { id: 1, nombre: "Fender Stratocaster", corte: "Strat", anio: 1954, enProduccion: true },
    { id: 2, nombre: "Gibson Les Paul Standard", corte: "Single Cut", anio: 1952, enProduccion: true },
    { id: 3, nombre: "Ibanez RG550", corte: "Superstrat", anio: 1987, enProduccion: true },
    { id: 4, nombre: "Gibson SG", corte: "Double Cut", anio: 1961, enProduccion: true },
    { id: 5, nombre: "Fender Telecaster", corte: "Single Cut", anio: 1950, enProduccion: true },
    { id: 6, nombre: "Jackson Soloist", corte: "Superstrat", anio: 1984, enProduccion: true },
    { id: 7, nombre: "Gretsch White Falcon", corte: "Hollow Body", anio: 1955, enProduccion: false },
    { id: 8, nombre: "PRS Custom 24", corte: "Double Cut", anio: 1985, enProduccion: true }
  ];

  constructor() {}

}
```

9. La Vista en home.page.html

```html
<ion-header>
  <ion-toolbar color="primary">
    <ion-title>Lista de Guitarras</ion-title>
  </ion-toolbar>
</ion-header>

<ion-content>
  <ion-list>
    <ion-card *ngFor="let guitarra of listaDeGuitarras" [color]="guitarra.enProduccion ? 'primary' : 'light'">
      <ion-item>
        <ion-label>
          <h2>{{ guitarra.nombre }}</h2>
          <p>Corte: {{ guitarra.corte }}</p>
          <p>Año: {{ guitarra.anio }}</p>
          <p *ngIf="!guitarra.enProduccion" class="fuera-produccion">
            ⚠️ Fuera de producción
          </p>
        </ion-label>

        <ion-button slot="end" fill="outline" (click)="mostrarDetalles(guitarra)">
          Ver
        </ion-button>
      </ion-item>
    </ion-card>
  </ion-list>
</ion-content>
```

10. Resultado final



Retos para la Máxima Calificación (Bien / Excelente)

Mejora	Objetivo	Resultado final
1Estilos Dinámicos	Ya lo tengo hecho (con enProduccion).	
Componente de Cabecera	Crear app-header y usarlo en home.page.html.	Mostrar título/subtítulo o logo
Pipes	Aplicar uppercase, date, u otros para formatear datos.	Mostrar datos con formato
Datos Opcionales	Crear componente hijo guitarra-item que reciba un @Input.	Reutilizar el componente y manejar datos opcionales

Mejora 2 – Crear un Componente de Cabecera

Crear el componente

```
C:\Users\9_2DAW\mi-app-guitar>ionic generate component components/app-header
> ng.cmd generate component components/app-header --project=app
CREATE src/app/components/app-header/app-header.component.html (32 bytes)
CREATE src/app/components/app-header/app-header.component.spec.ts (742 bytes)
CREATE src/app/components/app-header/app-header.component.ts (298 bytes)
CREATE src/app/components/app-header/app-header.component.scss (0 bytes)
[OK] Generated component!
```

app-header.component.ts

Asegúrate de marcarlo como *standalone* (si tu proyecto usa ese sistema):

app-header.component.ts U

src > app > components > app-header > app-header.component.ts > AppHeaderComponent

```ts
import { Component } from '@angular/core';
import { IonHeader, IonToolbar, IonTitle } from '@ionic/angular/standalone';

@Component({
  selector: 'app-header',
  standalone: true,
  imports: [IonHeader, IonToolbar, IonTitle],
  templateUrl: './app-header.component.html',
  styleUrls: ['./app-header.component.scss']
})
export class AppHeaderComponent {}
```

Vista app-header.component.html

app-header.component.html U

src > app > components > app-header > app-header.component.html > ion-header

Go to component

```html
<ion-header>
  <ion-toolbar color="dark">
    <ion-title>Galería de Guitarras</ion-title>
  </ion-toolbar>
</ion-header>

```

Usar nuevo componente en home.page.ts

Lo importamos y añadimos al array de imports.

```ts
src > app > home > TS home.page.ts > HomePage
import { Component } from '@angular/core';
import { IonHeader, IonToolbar, IonTitle, IonContent, IonList, IonCard, IonItem, IonLabel, IonButton } from '@ionic/angular/standalone';

// Imports de nuestra aplicación
import { Lista } from '../interfaces/lista'; // Nuestra interfaz de datos
import { ListItemComponent } from '../components/list-item/list-item.component'; // El componente hijo
import { CommonModule } from '@angular/common'; // Módulo necesario para usar *ngIf y *ngFor

// Import de nuevo componente app-header
import { AppHeaderComponent } from 'src/app/components/app-header/app-header.component';


@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
  imports: [IonButton, IonLabel, IonItem, IonCard, IonList,
    IonHeader, IonToolbar, IonTitle, IonContent, // Componentes de Ionic
    ListItemComponent, // Nuestro componente hijo
    CommonModule, // El módulo para las directivas estructurales
    AppHeaderComponent // Nuestro nuevo componente de encabezado
  ],
})
```

Enhome.page.html, reemplazamos el <ion-header> por <app-header>

```
home.page.html M
src > app > home > home.page.html > app-header
      Go to component
<app-header>
  <ion-toolbar color="primary">
    <ion-title>Lista de Guitarras</ion-title>
  </ion-toolbar>
</app-header>
```

Mejora 3 – Usar Pipes de Angular

Usa un Pipe en la lista de guitarras (home.page.html)

Ponemos, por ejemplo, el nombre en mayúsculas.

```
<ion-label>
  <h2>{{ guitarra.nombre | uppercase }}</h2>
```

Mejora 4 – Componente Hijo con Manejo de Datos Opcionales

Genera el componente hijo

```
C:\Users\9_2DAW\mi-app-guitar>ionic generate component components/guitarra-item
> ng.cmd generate component components/guitarra-item --project=app
CREATE src/app/components/guitarra-item/guitarra-item.component.html (35 bytes)
CREATE src/app/components/guitarra-item/guitarra-item.component.spec.ts (763 bytes)
CREATE src/app/components/guitarra-item/guitarra-item.component.ts (310 bytes)
CREATE src/app/components/guitarra-item/guitarra-item.component.scss (0 bytes)
[OK] Generated component!
```

En guitarra-item.component.ts

TS guitarra-item.component.ts 4, U

src > app > components > guitarra-item > TS guitarra-item.component.ts > ...

```ts
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { IonCard, IonItem, IonLabel, IonButton } from '@ionic/angular/standalone';
import { Lista } from 'src/app/interfaces/lista';

@Component({
  selector: 'app-guitarra-item',
  standalone: true,
  imports: [CommonModule, IonCard, IonItem, IonLabel, IonButton],
  templateUrl: './guitarra-item.component.html',
  styleUrls: ['./guitarra-item.component.scss']
})
export class GuitarraItemComponent {
  @Input() guitarra?: Lista;

  mostrarDetalles() {
    console.log('Detalles de guitarra:', this.guitarra);
  }
}
```

En guitarra-item.component.html

guitarra-item.component.html U

src > app > components > guitarra-item > guitarra-item.component.html > ion-card

Go to component

```
<ion-card *ngIf="guitarra" [color]="guitarra.enProduccion ? 'primary' : 'light'">
  <ion-item>
    <ion-label>
      <h2>{{ guitarra.nombre | uppercase }}</h2>
      <p>Corte: {{ guitarra.corte }}</p>
      <p>Año: {{ guitarra.anio }}</p>
      <p *ngIf="!guitarra.enProduccion" class="fuera-produccion">
        Fuera de producción
      </p>
    </ion-label>

    <ion-button slot="end" fill="outline" (click)="mostrarDetalles()">
      Ver
    </ion-button>
  </ion-item>
</ion-card>
```

Añadimos el nuevo componente a home.page.ts

```
@Component({
  selector: 'app-home',
  templateUrl: './home.page.html',
  styleUrls: ['./home.page.scss'],
  imports: [IonButton, IonLabel, IonItem, IonCard, IonList,
    IonHeader, IonToolbar, IonTitle, IonContent, // Componentes de Ionic
    ListItemComponent, // Nuestro componente hijo
    CommonModule, // El módulo para las directivas estructurales
    AppHeaderComponent // Nuestro nuevo componente de encabezado
    , GuitarraItemComponent // Componente para mostrar cada guitarra
  ],
})
```

Usamos el componente hijo en home.page.html

Reemplazamos el bucle por:

guitarra-item.component.html U | home.page.html M | home.page.ts 6, M

src > app > home > home.page.html > ion-content > ion-list > app-guitarra-item

Go to component

```
<app-header>
  <ion-toolbar color="primary">
    <ion-title>Lista de Guitarras</ion-title>
  </ion-toolbar>
</app-header>

<ion-content>
  <ion-list>
    <app-guitarra-item
      *ngFor="let guitarra of listaDeGuitarras"
      [guitarra]="guitarra">
    </app-guitarra-item>
  </ion-list>
</ion-content>
```

Resultado: