


Módulo 6: Creación de una biblioteca de interfaz
de usuario de componentes con Angular CLI y
Angular CDK

Desarrollo de Aplicaciones Web II

- 
- Una aplicación angular consta de componentes que están organizados en módulos. Cuando los componentes necesitan compartir una apariencia o comportamiento similar entre módulos, extraemos su funcionalidad en componentes reutilizables y los agrupamos en un módulo compartido.
 - Los componentes reutilizables pueden variar desde estructuras complejas de interfaz de usuario (UI) con muchos controles (como formularios) hasta elementos únicos del lenguaje de marcado de hipertexto (HTML) (como botones).
 - Una biblioteca de IU de componentes es una colección de componentes reutilizables que se pueden usar fuera de un dominio de aplicación específico. Una aplicación empresarial construida con una arquitectura monorepo puede utilizar estos componentes en todas sus aplicaciones. Un proyecto fuera de una organización también puede usar la misma biblioteca de componentes como dependencia externa.

- La interfaz de línea de comandos (CLI) de Angular incluye todas las herramientas necesarias para crear bibliotecas con Angular. El kit de desarrollo de componentes angular (CDK) proporciona una amplia gama de funcionalidades para crear componentes de interfaz de usuario accesibles y de alto rendimiento. En este capítulo, los combinaremos con Bulma, un marco moderno de hojas de estilo en cascada (CSS), para crear una biblioteca de interfaz de usuario de componentes simple desde cero.
- En este capítulo, cubriremos los siguientes temas con más detalle:
 - Creando una biblioteca con Angular CLI
 - Crear una lista de tarjetas que se puede arrastrar
 - Interactuar con el portapapeles
 - Publicar una biblioteca Angular en npm
 - Usar componentes como elementos angulares

Teoría y contexto esenciales

- El CDK de angular contiene una colección de interacciones y comportamientos comunes que podemos aplicar a los componentes angular. Está en el corazón de la biblioteca Angular Material, pero se puede usar con cualquier framework CSS en una aplicación Angular. Angular CDK está disponible en el paquete `@angular/cdknpm`.
- Angular CLI admite la creación de bibliotecas angular listas para usar. La funcionalidad de una biblioteca Angular se puede usar solo en aplicaciones Angular y está desacoplada de la lógica empresarial específica. Si queremos usar una biblioteca angular en una aplicación no angular, necesitamos convertirla en elementos angular.

- Los elementos personalizados son un estándar web que permite la creación de elementos HTML, independientemente de cualquier framework de JavaScript. Funciona declarando una etiqueta HTML personalizada y asociándola con una clase de JavaScript. El navegador puede identificar la etiqueta HTML y ejecutar el código JavaScript que se define dentro de la clase.
- Los elementos angular son componentes angulares convertidos en elementos personalizados utilizando el paquete npm `@angular/elements`. El empaquetado de un componente Angular como un elemento personalizado conecta el framework Angular con el Modelo de objetos de documento (DOM), enriqueciendo el elemento HTML nativo con enlace de datos, ciclo de vida del componente y funciones de detección de cambios.

Proyecto

- En este proyecto, crearemos una biblioteca de interfaz de usuario de componentes para nuestros proyectos Angular. Inicialmente, usaremos la CLI de Angular para crear un nuevo espacio de trabajo de Angular para nuestra biblioteca. Luego usaremos Angular CDK y Bulma CSS framework para crear los siguientes componentes:
 - Una lista de tarjetas que podemos reorganizar mediante las funciones de arrastrar y soltar
 - Un botón que nos permitirá copiar contenido arbitrario al portapapeles
- Aprenderemos cómo implementar la biblioteca en un registro de paquetes como npm. Finalmente, convertiremos uno de nuestros componentes en un elemento angular para compartirlo con aplicaciones no angulares, usando la biblioteca ngx-build-plus.

Creando una biblioteca con Angular CLI



- Antes de que podamos comenzar a trabajar con bibliotecas Angular usando la CLI Angular, necesitamos crear un espacio de trabajo CLI Angular. El espacio de trabajo de Angular CLI contendrá nuestra biblioteca Angular y una aplicación Angular para probar la biblioteca.
- Utilice el siguiente comando para generar un nuevo espacio de trabajo CLI angular:

```
\webII> ng new my-components --defaults
```

- El comando anterior creará un nuevo espacio de trabajo CLI de Angular que contiene una aplicación Angular llamada my-components. Navegue a la carpeta my-components y ejecute el siguiente comando para generar una nueva biblioteca Angular:

```
my-components> ng generate library ui-controls
```


- El comando anterior creará una biblioteca de controles de interfaz de usuario dentro de la carpeta de proyectos del espacio de trabajo. Contendrá varios archivos similares a los de la creación de una aplicación Angular, incluidos los siguientes:
 - `src\lib`: contiene el código fuente de la biblioteca, como módulos, componentes y servicios.
 - `src\public-api.ts`: exporta artefactos de la biblioteca que queremos que estén disponibles públicamente en otras aplicaciones de Angular.
 - `ng-package.json`: contiene una configuración para la biblioteca `ng-packagr` que la CLI de Angular usa bajo el capó para construir bibliotecas.
 - `tsconfig.lib.json`: El archivo de configuración de TypeScript para nuestra biblioteca que también contiene varias opciones del compilador Angular.
 - `tsconfig.lib.prod.json`: El archivo de configuración de TypeScript que se usa al construir nuestra biblioteca en modo de producción.

- Angular CLI generará un módulo, un componente y un servicio en la carpeta src\lib para nosotros de forma predeterminada. También los exportará para que puedan ser utilizados por cualquier aplicación Angular que utilice la biblioteca. Puedes ver un ejemplo de esto aquí:

```
projects > ui-controls > src > TS public-api.ts
1  /*
2  * Public API Surface of ui-controls
3  */
4
5  export * from './lib/ui-controls.service';
6  export * from './lib/ui-controls.component';
7  export * from './lib/ui-controls.module';
8
```

- Ahora que hemos configurado nuestro espacio de trabajo de Angular CLI, podemos continuar e instalar las bibliotecas Bulma y Angular CDK, de la siguiente manera:
- Ejecute el siguiente comando npm para instalar Angular CDK:

```
ts> npm install @angular/cdk
```

- Ejecute el siguiente comando para instalar el marco CSS de bulma:


```
s> npm install bulma
```

- Abra el archivo de configuración angular.json y agregue el archivo de hoja de estilo CSS de la biblioteca bulma en la sección de estilos de la entrada del arquitecto de compilación, de la siguiente manera:

```
29 "styles": [  
30   "src/styles.css",  
31   "./node_modules/bulma/css/bulma.css"  
32 ],  
33 "scripts": {  
34   "start": "react-scripts start",  
35   "build": "react-scripts build",  
36   "test": "react-scripts test",  
37   "eject": "react-scripts eject",  
38   "predeploy": "npm run build",  
39   "deploy": "gh-pages -d build" }  
36 }
```


- Abra el archivo package.json de la carpeta projects\ui-controls y agregue los siguientes paquetes a la sección peerDependencies:

```
"peerDependencies": {  
  "@angular/common": "^12.2.0",  
  "@angular/core": "^12.2.0",  
  "@angular/cdk": "^12.0.2",  
  "bulma": "^0.9.2"  
}
```

- 
- Agregamos Angular CDK y la biblioteca Bulma a la sección `peerDependencies` para asegurarnos de que cualquier aplicación consumidora tenga una versión específica de los paquetes para ejecutar nuestra biblioteca.
 - Los números de versión de cada paquete pueden variar si sigue este proyecto. Para asegurarse de que tiene las versiones correctas, cópielas del archivo `package.json` de la carpeta raíz del espacio de trabajo.
 - Ahora hemos completado la configuración básica de nuestra biblioteca de componentes de la interfaz de usuario. También hemos configurado la aplicación Angular que viene con el espacio de trabajo CLI de Angular para obtener una vista previa y probar la biblioteca. En la siguiente sección, crearemos el primer componente de nuestra biblioteca: una lista de tarjetas que se puede reordenar.


Construyendo una lista de tarjetas que se puede arrastrar

- El primer componente de nuestra biblioteca de UI será una lista de elementos de la tarjeta Bulma. Cada tarjeta mostrará un título, una descripción y un elemento de enlace de anclaje. También podremos arrastrar una tarjeta y cambiar el orden de la lista de tarjetas usando el Angular CDK. La construcción de nuestro componente constará de las siguientes tareas:
 - Visualización de datos de la tarjeta
 - Añadiendo funcionalidad de arrastrar y soltar
- En la siguiente sección, primero veremos cómo mostrar datos en la lista de tarjetas.

Visualización de datos de la tarjeta

- Nuestra aplicación Angular debería pasar una lista de tarjetas como una propiedad de entrada al componente para mostrarlas. Veamos cómo podemos crear un componente de tarjeta arrastrable, de la siguiente manera:
- Ejecute el siguiente comando CLI de Angular para crear un componente Angular:

```
> ng generate component card-list --project=ui-controls --export
```


- 
- El comando anterior creará un componente de lista de tarjetas en el proyecto ui-controls de nuestro espacio de trabajo Angular CLI. La opción --export también exportará el componente desde UiControlsModule.
 - La clase UiControlsModule ya se exportó desde el archivo public-api.ts. Entonces, cuando nuestra aplicación Angular importe UiControlsModule, también tendrá nuestro componente disponible para usar.

- Use el comando generate de Angular CLI para crear una interfaz para la estructura de los datos de la tarjeta, de la siguiente manera:

```
ng generate interface card --project=ui-controls
```

- El comando anterior creará un archivo card.ts en el proyecto ui-controls de nuestro espacio de trabajo.

- Abra el archivo card.ts y agregue las siguientes propiedades a la interfaz de la tarjeta:


```
projects > ui-controls > src > lib > TS card.ts > ...  
1  export interface Card {  
2      title: string;  
3      description: string;  
4      link: string;  
5  }
```

- Abra el archivo public-api.ts y agregue las siguientes declaraciones de exportación para que el componente y la interfaz estén disponibles para los consumidores de la biblioteca:

```
export * from './lib/ui-controls.service';  
export * from './lib/ui-controls.component';  
export * from './lib/ui-controls.module';  
export * from './lib/card-list/card-list.component';  
export * from './lib/card';
```


- Abra el archivo card-list.component.ts y use el decorador @Input para definir una propiedad de entrada, de la siguiente manera:

```
projects > ui-controls > src > lib > card-list > TS card-list.component.ts > CardListCom
1  import { Component, Input, OnInit } from '@angular/core';
2  import { Card } from '../card';
3
4  @Component({
5    selector: 'lib-card-list',
6    templateUrl: './card-list.component.html',
7    styleUrls: ['./card-list.component.css']
8  })
9  export class CardListComponent implements OnInit {
10    @Input() cards: Card[] = [];
11    constructor() { }
12
13    ngOnInit(): void {
```


- 
- La propiedad de las tarjetas se configurará posteriormente desde la aplicación Angular con los datos de la tarjeta que queremos mostrar.
 - Abra el archivo `card-list.component.html` y reemplace su contenido con la siguiente plantilla HTML:

projects > ui-controls > src > lib > card-list > <> card-list.component.html > ...

```
1  <div>
2    <div class="card m-4" *ngFor="let card of cards">
3      <header class="card-header">
4        <p class="card-header-title">{{card.title}}</p>
5      </header>
6      <div class="card-content">
7        <div class="content">{{card.description}}</div>
8      </div>
9      <footer class="card-footer">
10       <a [href]="card.link" class=
11         | "card-footer-item">View on Wikipedia</a>
12      </footer>
13    </div>
14  </div>
```


- La plantilla anterior usa el componente de tarjeta de la biblioteca Bulma e itera sobre la propiedad del componente de tarjetas para mostrar cada uno, usando la directiva *ngFor.
- Abra el archivo card-list.component.css y agregue los siguientes estilos CSS:

```
projects > ui-controls > src > lib > card-list > #  
1  :host > div {  
2      display: grid;  
3      grid-auto-flow: column;  
4      overflow: auto;  
5  }  
6  .card {  
7      width: 200px;  
8  }
```


- 
- En los estilos anteriores, usamos el selector: host para apuntar al elemento div en el elemento host de nuestro componente y aplicamos un estilo CSS Grid para mostrar todas las tarjetas en una sola fila.
 - Abra el archivo ui-controls.module.ts y agregue CommonModule a la matriz de importaciones del decorador @NgModule, de la siguiente manera:

- La clase CommonModule es necesaria para la directiva * ngFor que usamos en la plantilla de componentes de la lista de tarjetas.

```
projects > ui-controls > src > lib > TS ui-controls.module.ts > ...  
1  import { NgModule } from '@angular/core';  
2  import { UiControlsComponent } from './ui-co  
3  import { CardListComponent } from './card-li  
4  import { CommonModule } from '@angular/commo  
5  @NgModule({  
6    declarations: [  
7      UiControlsComponent,  
8      CardListComponent  
9    ],  
10   imports: [CommonModule  
11   ],  
12   exports: [  
13   ]
```

- 
- Nuestro componente ahora está listo para aceptar datos y mostrarlos en una representación de lista de tarjetas.
 - Veamos cómo consumirlo desde la aplicación Angular, de la siguiente manera:
 - Primero, ejecute el siguiente comando para crear la biblioteca de la interfaz de usuario del componente:

```
ng build ui-controls
```

Building entry point 'ui-controls'


✓ Compiling with Angular sources in Ivy partial compilation mode.
✓ Bundling to FESM2015
✓ Bundling to UMD
✓ Writing package metadata
✓ Built ui-controls

Built Angular Package

- from: C:\Users\Geovany\Desktop\webII\my-components\projects\ui-controls
- to: C:\Users\Geovany\Desktop\webII\my-components\dist\ui-controls

- Abra el archivo app.module.ts y agregue la clase UiControlsModule a la matriz de importaciones del decorador @NgModule, de la siguiente manera:

```
4 | import { UiControlsModule } from 'ui-controls';  
5 | @NgModule({  
6 |   declarations: [  
7 |     AppComponent  
8 |   ],  
9 |   imports: [  
10 |     BrowserModule,  
11 |     UiControlsModule  
12 |   ],  
13 |   providers: []
```


- 
- Importamos UiControlsModule desde el espacio de nombres ui-controls, que es el nombre de la biblioteca, y no desde la ruta absoluta completa en nuestro espacio de trabajo.
 - Abra el archivo app.component.ts y declare una propiedad de componente del tipo Card [], de la siguiente manera (pidan el archivo assassins.ts):

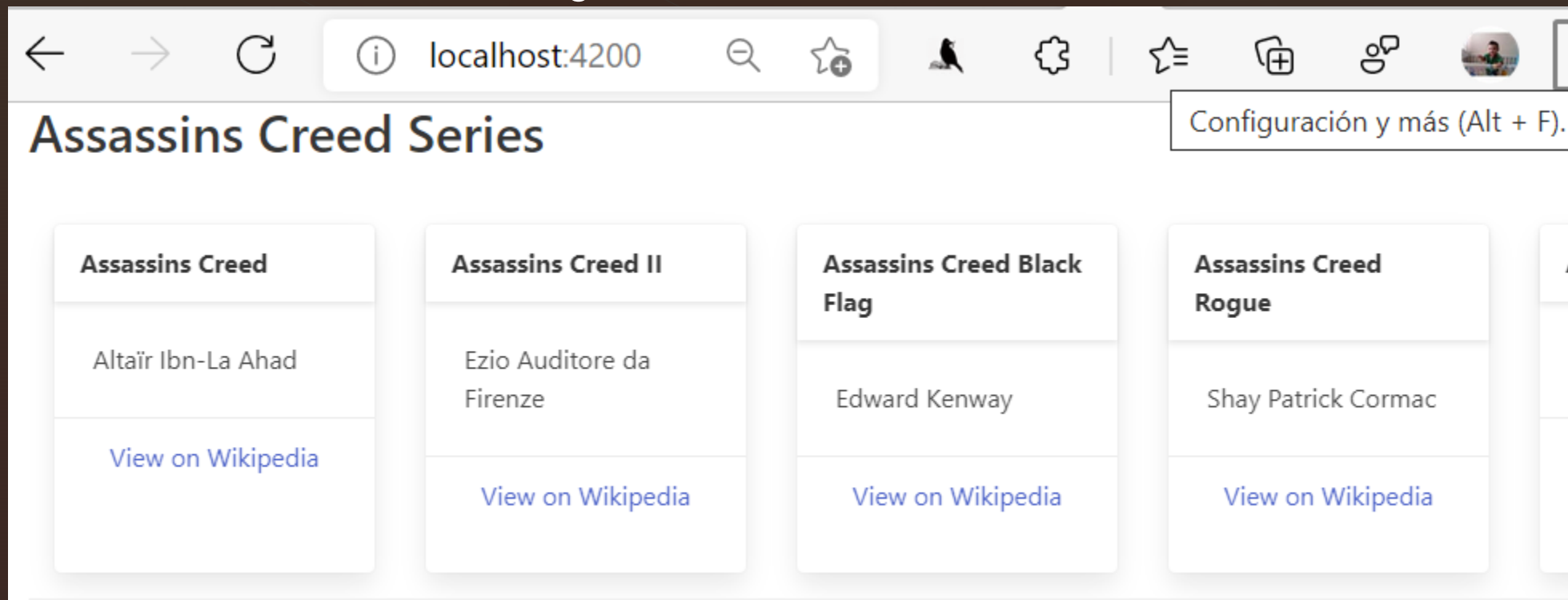
src > app > TS app.component.ts > ...


```
1  import { Component } from '@angular/core';
2  import { Card } from 'ui-controls';
3  import { assassins } from './assassins';
4
5  @Component({
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.css']
9  })
10 export class AppComponent {
11   title = 'my-components';
12   cards: Card[] = assassins;
13 }
```

- Inicializamos la propiedad del componente de tarjetas usando datos de demostración del archivo `assassins.ts`
- Abra el archivo `app.component.html` y reemplace su contenido con la siguiente plantilla HTML:

```
src > app > <> app.component.html > ...  
1 | <div class="container is-fluid">  
2 |   <h1 class="title">Assassins Creed Series</h1>  
3 |   <lib-card-list [cards]="cards"></lib-card-list>  
4 | </div>  
- |
```

- Para obtener una vista previa de la aplicación, ejecute `ng serve` y abra su navegador en `http://localhost:4200`. Entonces debería ver algo como esto:




- 
- El componente de lista de tarjetas muestra los datos que una aplicación de consumidor pasó utilizando la propiedad de entrada de tarjetas.
 - En la siguiente sección, llevaremos nuestro componente un paso más allá y haremos que nuestras tarjetas puedan cambiar su ubicación en la lista.

Añadiendo funcionalidad de arrastrar y soltar


- Una característica del componente de lista de tarjetas es que podremos cambiar la ubicación de una tarjeta arrastrándola y soltándola dentro de la lista. El orden de la lista de tarjetas debe enviarse de nuevo a la aplicación del consumidor mediante un enlace de propiedad de salida.
- El Angular CDK contiene un módulo de arrastrar y soltar que podemos usar para este propósito.
- Para hacerlo, siga estos pasos:
- Abra el archivo `ui-controls.module.ts` e importe `DragDropModule` desde el espacio de nombres `@ angular / cdk / drag-drop`, así:

```
4 import { CommonModule } from '@angular/common';
5 import { DragDropModule } from '@angular/cdk/drag-drop';
6 @NgModule({
7   declarations: [
8     UiControlsComponent,
9     CardListComponent
10  ],
11  imports: [CommonModule,
12    DragDropModule
13  ],
14  exports: [
15
```



- Abra el archivo card-list.component.html y modifique la plantilla de la siguiente manera:

```
projects > ui-controls > src > lib > card-list > <> card-list.component.html > div > div.card.m-4
1  <div cdkDropListOrientation="horizontal" cdkDropList
2  (cdkDropListDropped)="sortCards($event)">
3    <div cdkDrag class="card m-4" *ngFor="let card of cards">
4      <header class="card-header">
5        <p class="card-header-title">{{card.title}}</p>
6      </header>
7      <div class="card-content">
```

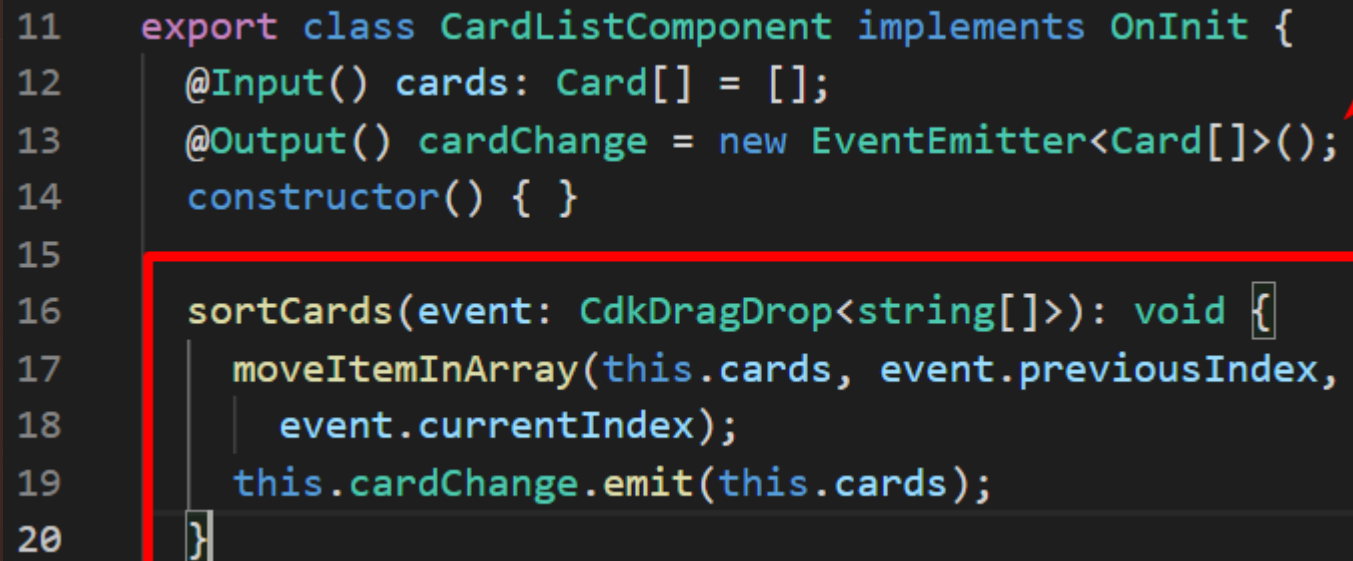
- 
- Primero, agregamos la directiva `cdkDrag` a cada elemento de la tarjeta para poder moverlo arrastrándolo. Luego, agregamos la directiva `cdkDropList` al elemento contenedor para marcarlo como una lista desplegable.
 - Una lista desplegable en Angular CDK indica que su contenido se puede reordenar mediante acciones de arrastrar y soltar. Establecemos la orientación de arrastrar y soltar en horizontal porque nuestra lista de tarjetas se representa en una sola fila, y también vinculamos un método de componente `sortCards` al evento `cdkDropListDropped` de la lista desplegable.


- Abra el archivo `card-list.component.ts` y agregue las siguientes declaraciones de importación:


```
projects > ui-controls > src > lib > card-list > TS card-list.component.ts > ...  
1  import { Component, Input, Output, OnInit, EventEmitter }  
2  from '@angular/core';  
3  import { Card } from '../card';  
4  import { CdkDragDrop, moveItemInArray } from '@angular/cdk/drag-drop';  
5
```


- Cree una propiedad de salida utilizando el decorador @Output y utilícela en el método del componente sortCards para emitir la lista reordenada al consumidor del componente, de la siguiente manera:

```
11 export class CardListComponent implements OnInit {  
12     @Input() cards: Card[] = [];  
13     @Output() cardChange = new EventEmitter<Card[]>();  
14     constructor() { }  
15  
16     sortCards(event: CdkDragDrop<string[]>): void {  
17         moveItemInArray(this.cards, event.previousIndex,  
18             event.currentIndex);  
19         this.cardChange.emit(this.cards);  
20     }
```



- 
- En el fragmento de código anterior, usamos el método integrado `moveItemInArray` de `DragDropModule` para cambiar el orden de la propiedad de las tarjetas.
 - Pasamos el parámetro de evento al método `moveItemInArray` que contiene el índice anterior y actual de la tarjeta movida. También usamos el método de emisión de la propiedad `cardChange` para propagar el cambio a la aplicación Angular.

- 
- El componente de lista de cartas ahora ha adquirido superpoderes de arrastrar y soltar. Intentémoslo, de la siguiente manera:
 - Abra el archivo `app.component.html` y agregue un enlace de evento al evento `cardChange` del componente `lib-card-list`, de la siguiente manera:

src > app > <> app.component.html > ...

```
1  <div class="container is-fluid">
2    <h1 class="title">Assassins Creed Series</h1>
3    <lib-card-list [cards]="cards"
4      | (cardChange)="onCardChange($event)">
5    </lib-card-list>
6  </div>
7
```

- Abra el archivo `app.component.ts` y cree un método `onCardChange` para registrar la nueva lista de tarjetas, de la siguiente manera:


```
onCardChange(cards: Card[]) {  
  console.log(cards);  
}
```


- Ejecute el siguiente comando para construir la biblioteca:

```
ts> ng build ui-controls
```

- Ejecute el comando de servicio de Angular CLI para iniciar su aplicación, así:

```
nts> ng serve
```

- 
- Intente arrastrar y soltar algunas de las tarjetas de la lista y observe el resultado en la ventana de la consola de su navegador.
 - El primer componente de nuestra biblioteca de interfaz de usuario ahora incluye todas las funciones para convertirlo en una lista de arrastrar y soltar. Puede mostrar una lista pasada desde nuestra aplicación Angular en un formato de tarjeta Bulma. También puede cambiar el orden de cada elemento en la lista usando el módulo de arrastrar y soltar de Angular CDK, y propagar el cambio a nuestra aplicación.
 - En la siguiente sección, crearemos un segundo componente de nuestra biblioteca para copiar datos al portapapeles.

Interactuar con el portapapeles

- La biblioteca Angular CDK contiene una colección de artefactos angulares que podemos usar para interactuar con el portapapeles del sistema. Específicamente, incluye una directiva para copiar datos en el portapapeles y un enlace de eventos para realizar acciones adicionales cuando se ha copiado el contenido. Veamos cómo podemos integrar ambos en nuestra biblioteca de componentes, de la siguiente manera:
- Ejecute el siguiente comando de Angular CLI para crear un nuevo componente Angular en la biblioteca:

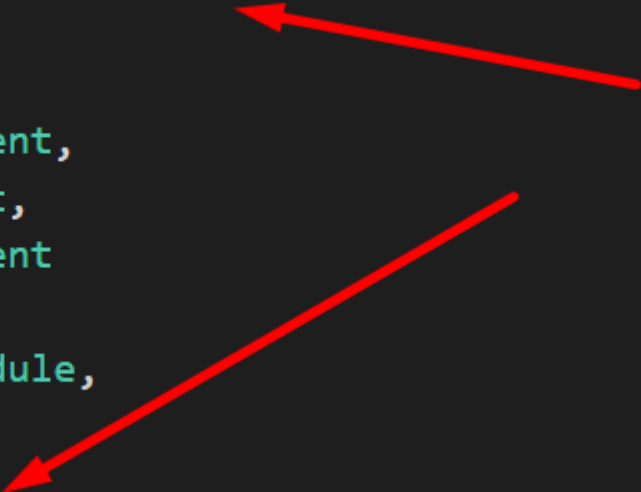
```
PS C:\Users\Geovany\Desktop\webII\my-components> ng generate component copy-button --project=ui-controls --export
```

- Exporte el componente recién generado desde el archivo public-api.ts, de la siguiente manera:


```
export * from './lib/ui-controls.service';  
export * from './lib/ui-controls.component';  
export * from './lib/ui-controls.module';  
export * from './lib/card-list/card-list.component';  
export * from './lib/card';  
export * from './lib/copy-button/copy-button.component';
```

- Abra el archivo ui-controls.module.ts e importe ClipboardModule desde el espacio de nombres @angular cdk/clipboard, así:

```
import { ClipboardModule } from '@angular/cdk/clipboard';
@NgModule({
  declarations: [
    UiControlsComponent,
    CardListComponent,
    CopyButtonComponent
  ],
  imports: [CommonModule,
    DragDropModule,
    ClipboardModule
```



- Abra el archivo copy-button.component.ts y declare las siguientes propiedades del componente:


```
projects > ui-controls > src > lib > copy-button > TS copy-button.component.ts >   
1  import { Component, EventEmitter, Input, OnInit, Output }  
2  from '@angular/core';  
3  @Component({  
4    selector: 'lib-copy-button',  
5    templateUrl: './copy-button.component.html',  
6    styleUrls: ['./copy-button.component.css']  
7  })  
8  export class CopyButtonComponent implements OnInit {  
9    @Input() data = '';  
10   @Output() copied = new EventEmitter();  
11   constructor() { }  
12
```


- La propiedad de datos se utilizará para configurar los datos del portapapeles y el evento copiado se activará cuando los datos se copien correctamente en el portapapeles.
- Cree un método de componente para desencadenar un evento de salida copiado, de la siguiente manera:


```
onCopy() {  
  |   this.copied.next();  
}
```

- Abra el archivo `copy-button.component.html` y reemplace su contenido con la siguiente plantilla HTML:

```
projects > ui-controls > src > lib > copy-button > <> copy-but  
1  <button  
2    class="button is-light is-primary"  
3    [cdkCopyToClipboard]="data"  
4    (cdkCopyToClipboardCopied)="onCopy()">  
5    Copy  
6  </button>
```

- 
- En la plantilla anterior, usamos un elemento de botón Bulma y le adjuntamos dos enlaces Angular CDK. El enlace de propiedad `cdkCopyToClipboard` indica que la propiedad del componente de datos se copiará en el portapapeles cuando se haga clic en el botón.
 - El enlace de eventos `cdkCopyToClipboardCopied` llamará al método del componente `onCopy` tan pronto como los datos se hayan copiado correctamente en el portapapeles.

- 
- Ahora que hemos configurado nuestro componente, descubramos cómo usarlo en nuestra aplicación Angular, de la siguiente manera:
 - Abra el archivo `app.component.html` y agregue un elemento `div` que consta de un elemento HTML de entrada y un componente `lib-copy-button`, de la siguiente manera:

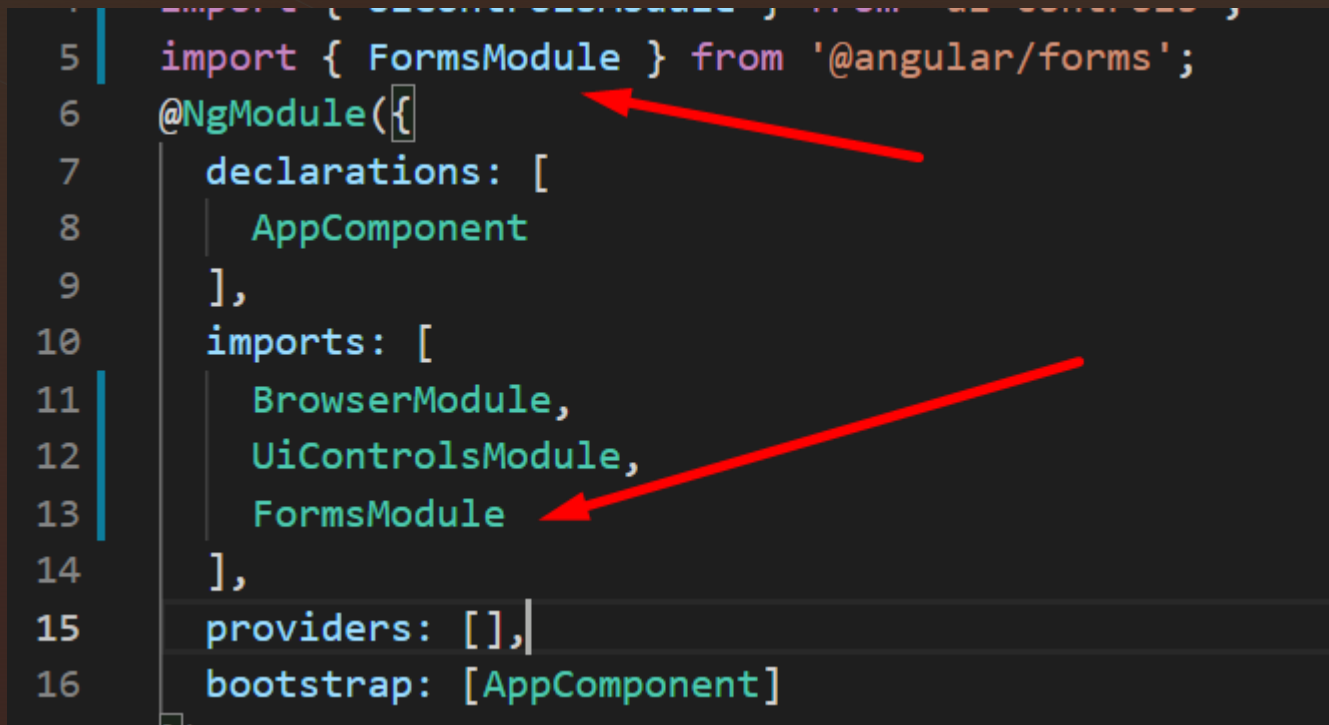

```
src > app > <> app.component.html >  div.container.is-fluid
2 | <h1 class="title">Assassins Creed Series</h1>
3 | <lib-card-list [cards]="cards"
4 | | (cardChange)="onCardChange($event)">
5 | </lib-card-list>
6 |
7 | <h1 class="title mt-5">Clipboard interaction</h1>
8 | <div class="field has-addons">
9 | | <div class="control">
10 | | | <input class="input" type="text"
11 | | | [(ngModel)]="title">
12 | | </div>
13 | | <div class="control">
14 | | | <lib-copy-button [data]="title"
15 | | | (copied)="log()"></lib-copy-button>
16 | | </div>
17 | </div>
18 |
```

- En la plantilla anterior, vinculamos la propiedad title del componente al elemento de entrada usando la directiva ngModel. También lo vinculamos a la propiedad de datos del componente lib-copy-button para copiar el contenido del elemento de entrada al portapapeles. También vinculamos el evento copiado al método del componente de registro.
- Abra el archivo app.component.ts y cree un método de registro para mostrar un mensaje de información cuando los datos se copian en el portapapeles, de la siguiente manera:

```
log() {  
  alert(this.title + ' copied to the clipboard');  
}
```


- Abra el archivo app.module.ts e importe FormsModule, así:

```
5 import { FormsModule } from '@angular/forms';
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10  imports: [
11    BrowserModule,
12    UiControlsModule,
13    FormsModule
14  ],
15  providers: [],
16  bootstrap: [AppComponent]
```

A screenshot of a code editor showing the app.module.ts file. The code is written in TypeScript and defines an @NgModule. Two red arrows are drawn on the image: one points to the @NgModule decorator on line 6, and the other points to the FormsModule import in the imports array on line 13.

- La clase `FormsModule` es parte del paquete `@angular/forms` npm y es necesaria cuando queremos usar `ngModel` en nuestra aplicación.
- Ejecute el siguiente comando para construir la biblioteca para que nuestra aplicación pueda reconocer el nuevo componente:

```
ts> ng build ui-controls
```

- 
- Ejecute la aplicación usando el comando serve de Angular CLI, y debería obtener el siguiente resultado:

←

→

↺

localhost:4200

🔗

★

🔥

🌀

♻️

Assassins Creed

Assassins Creed

Altair Ibn-La Ahad

[View on Wikipedia](#)

Ezio Auditore da Firenze

[View on Wikipedia](#)

Edward Kenway

[View on Wikipedia](#)

Ya reprobaron


Copy

localhost:4200 dice

Ya reprobaron copied to the clipboard

Aceptar

Clipboard interaction

- 
- ¡Hemos creado con éxito un botón que podemos adjuntar a una aplicación Angular e interactuar con el portapapeles directamente!
 - Angular CDK contiene varios otros componentes y comportamientos que podemos usar en nuestras aplicaciones Angular. Cuando se combina con un framework CSS altamente personalizable como Bulma, puede crear interfaces atractivas y únicas.
 - Pruébelos en sus proyectos de Angular y cree una biblioteca con un rico conjunto de componentes. En la siguiente sección, aprenderemos cómo publicar una biblioteca en el registro de paquetes npm.

Publicar una biblioteca angular en npm

- Ya hemos visto cómo construir una biblioteca Angular y consumirla en una aplicación Angular cuando ambas existen en el mismo repositorio u organización. Sin embargo, hay casos en los que es posible que desee que su biblioteca esté disponible para proyectos de Angular fuera de su infraestructura a través de un registro de paquetes público como npm.
- Un caso habitual es cuando desea que su biblioteca sea de código abierto para que otros miembros de la comunidad de desarrollo puedan beneficiarse de esto. Veamos cómo publicar nuestra biblioteca de controles de interfaz de usuario en npm, de la siguiente manera:
- Si no tiene una cuenta npm, vaya a <https://www.npmjs.com/signup> para crear una.

- Abra el archivo package.json que existe en la carpeta projects\ui-controls del espacio de trabajo de Angular CLI y cambie la versión de su biblioteca en consecuencia, de la siguiente manera:

```
projects > ui-controls > {} package.json > ...  
1  {  
2    "name": "ui-controls",  
3    "version": "1.0.0",  
4    "peerDependencies": {  
5      "@angular/common": "^12.2.0",  
6      "@angular/core": "^12.2.0",  
7      "@angular/cdk": "^12.0.2",  
8      "bulma": "^0.9.2"
```

- Abra una ventana de terminal y ejecute el siguiente comando Angular CLI para construir su biblioteca:

```
ponents> ng build ui-controls
```

- Navegue a la carpeta dist donde Angular CLI ha generado el paquete final de nuestra biblioteca, como se ilustra en el siguiente fragmento de código:

```
ponents> cd dist\ui-controls
```


- Ejecute el siguiente comando npm para iniciar sesión en el registro npm desde la terminal:

```
ui-controls> npm login
```





- El comando anterior le pedirá que complete sus valores de Nombre de usuario, Contraseña y Correo electrónico.
- Una vez que se haya autenticado correctamente con npm, ejecute el siguiente comando para publicar su biblioteca:

```
ui-controls> npm publish
```

- La ejecución del comando anterior arrojará un error porque el registro del paquete npm ya contiene un paquete ui-controls. Si desea obtener una vista previa del resultado del comando anterior, asegúrese de cambiar el campo de nombre en el archivo package.json de la biblioteca.

```
+ ui-controls-geo@1.0.0
```

```
PS C:\Users\Geovany\Desktop\webII\my-components\dist\ui-controls> █
```

- 
- ¡Bien hecho! Su biblioteca ahora está en el registro público de npm y otros desarrolladores pueden usarla en sus aplicaciones Angular.
 - Recuerde siempre cambiar el número de versión en el archivo package.json de su biblioteca antes de publicarlo. De lo contrario, el registro npm arrojará un error, indicando que la versión que está intentando publicar ya existe. También hay algunos paquetes npm, como la versión estándar, que pueden automatizar esto por usted.
- 
- 
- 

Resumen

- En este proyecto, creamos una biblioteca de interfaz de usuario de componentes que podemos usar en nuestras aplicaciones Angular. Inicialmente, aprendimos cómo usar Angular CLI para crear una biblioteca Angular.
- Creamos un nuevo espacio de trabajo CLI de Angular que contenía nuestra biblioteca Angular, junto con una aplicación Angular para probarla.
- Luego usamos Angular CDK con el marco CSS de Bulma para construir los componentes de la interfaz de usuario de nuestra biblioteca. Creamos una lista de tarjetas que se puede reordenar usando funciones de arrastrar y soltar y un botón para copiar contenido al portapapeles.
- También vimos cómo publicar nuestra biblioteca en el registro npm para usarla en otros proyectos de Angular.

Preguntas de práctica

- 1. ¿Cómo generamos una nueva biblioteca Angular usando la CLI de Angular?
- 2. ¿Cómo hacemos público un artefacto angular de nuestra biblioteca?
- 3. ¿Qué selector de CSS usamos para apuntar al elemento host de un componente angular?
- 4. ¿Cómo marcamos un elemento como arrastrable en Angular CDK?
- 5. ¿Qué método utilizamos para reordenar una lista de elementos que se puede arrastrar?
- 6. ¿Qué directiva Angular CDK es responsable de pasar datos al portapapeles?
- 7. ¿En qué modo construimos una biblioteca Angular para publicarla?
- 8. ¿Cómo creamos un solo paquete usando la biblioteca ngx-build-plus?
- 9. ¿Cómo pasamos datos hacia y desde un elemento angular?
- 10. ¿Por qué necesitamos polyfills cuando trabajamos con elementos angulares?