Se trata de seguir el siguiente ejemplo guiado para crear una aplicación básica de altas de registros (en memoria mediante array) a partir de un formulario, así practicaremos con módulos, componentes y servicios.

Creamos nuestra aplicación

El primer paso es, usando el CLI, crear nuestra nueva aplicación.

Ejecutamos:

```
ng new clientes --no-standalone
```

Luego entramos en la carpeta de la aplicación y lanzamos el servidor web de desarrollo, para ver lo que se ha construido hasta el momento.

```
cd clientes
```

```
ng serve -o
```

Creamos nuestro módulo de clientes

La aplicación recién generada ya contiene un módulo principal, sin embargo, queremos crear un módulo, aparte, para que realice sólo las funciones que le correspondan, de esta forma estamos dividiendo nuestra aplicación en módulos, que interactúan a través de sus componentes y los cuales son responsable de sólo una parte de la aplicación.

Creamos un nuevo módulo llamado: "**clientes**".

Encargamos a Angular CLI la creación del esqueleto de nuestro módulo con el siguiente comando (lo creará dentro de un directorio con el mismo nombre):

```
ng g module clientes
```

Definir el modelo de datos

Vamos a comenzar nuestro código definiendo los tipos de datos que vamos a usar en esta aplicación. Vamos a trabajar con clientes y grupos.

Crearemos el modelo de datos dentro del directorio del módulo "**clientes**". Lo nombramos "**cliente.model.ts**".

En este archivo colocamos los interfaces de TypeScript que definen los datos de nuestra aplicación.

```
export interface Cliente {
    id: number;
    nombre: string;
    cif: string;
    direccion: string;
    grupo: number;
}
export interface Grupo {
    id: number;
    nombre: string;
}
```

Hemos creado el tipo de datos Cliente y, por complicarlo un poquito más, el tipo de datos Grupo. Así, cada cliente generado pertenecerá a un grupo.

> **Nota:** Esta parte de la creación de interfaces es perfectamente opcional. Solo la hacemos para usar esos tipos en la declaración de variables. El compilador de TypeScript nos avisará si en algún momento no respetamos estos tipos de datos, ayudando en tiempo de desarrollo y previniendo posibles errores.

Crear un servicio para los clientes

Lo ideal es crear un servicio (service de Angular) donde concentremos las tareas de trabajo con los datos de los clientes (**modelo** en MVVM) ,

descargando de código a los componentes de la aplicación y centralizando en un solo archivo la lógica de la aplicación.

El servicio lo vamos a crear dentro del directorio del módulo **clientes**, por lo que especificamos la ruta completa.

```
ng generate service ./clientes/clientesService
```

Código para el servicio creado:

```
import { Injectable } from '@angular/core';
import { Cliente, Grupo } from './cliente.model';
@Injectable({
  providedIn: 'root'
})
export class ClientesService {
  private clientes: Cliente[];
  private grupos: Grupo[];
  constructor() {
    this.grupos = [
      {
        id: 0,
        nombre: 'Sin definir'
      },
      {
        id: 1,
        nombre: 'Activos'
      },
      {
        id: 2,
        nombre: 'Inactivos'
```

```
      },
      {
        id: 3,
        nombre: 'Deudores'
      },
    ];
    this.clientes = [];
  }
  getGrupos() {
    return this.grupos;
  }
  getClientes() {
    return this.clientes;
  }
  agregarCliente(cliente: Cliente) {
    this.clientes.push(cliente);
  }
  nuevoCliente(): Cliente {
    return {
      id: this.clientes.length,
      nombre: '',
      cif: '',
      direccion: '',
      grupo: 0
    };
  }
}
```

1. Los dos atributos (**clientes** y **grupos**)del servicio contienen los **datos que se van a mantener**. Sin embargo, los hemos definido como privados, de modo que no se puedan tocar directamente y tengamos que usar los métodos del servicio creados para su acceso.
2. Los grupos los construyes con un literal en el constructor. Generalmente los traerías de algún servicio REST/BBDD, pero de momento está bien para empezar.
3. Agregar un cliente es un simple "**push**" al array de clientes, de un cliente recibido por parámetro.
4. Crear un nuevo cliente es simplemente devolver un nuevo objeto vacío, que tiene que respetar la interfaz, ya que en el método **nuevoCliente( )** se está especificando que el valor retornado será un objeto del tipo **Cliente**.

Declarar el servicio para poder usarlo en los componentes

Una tarea fundamental para poder usar los servicios es declararlos en el "module" donde se vayan a usar, ya veremos más adelante que por convención debería de estar en el módulo **'core'**, de momento lo añadimos en nuestro nuevo módulo creado.

Para añadir el servicio en el module "**clientes-module.ts**", el primer paso es importarlo.

```
import { ClientesService } from './clientes-service';
```

Luego habría que declararlo en el array "**providers**" (en este caso podemos omitirlo, el servicio está disponible a nivel global como única instancia habiendo usado `providedIn: 'root'` en su propia clase).

```
providers: [
    ClientesService
```

```
  ],
```

Crear componente que da de alta a los clientes

Vamos a continuar nuestra práctica creando un primer componente en el directorio que se ha creado junto al módulo **clientes-module.ts**. Es el que se encargará de dar de alta a los clientes.

Generamos el esqueleto usando el Angular CLI.

```
ng generate component clientes/altaCliente
```

Comenzaremos editando el archivo del componente y luego iremos a trabajar con el template. Por tanto,vamos a abrir el fichero "**alta-cliente.ts**"

Agregar el servicio al componente

Muy importante. Para poder usar el servicio anterior, tengo que agregarlo al componente recién creado, realizando el correspondiente **import**.

```
import { ClientesService } from './../clientes-service';
```

Y posteriormente ya podré inyectar el servicio en el constructor del componente.

```
constructor(private clientesService: ClientesService) { }
```

Agregar el modelo de datos

Para poder seguir usando los tipos de datos de mi modelo, vamos a agregar el archivo donde se generaron los interfaces.

```
import { Cliente, Grupo } from './../cliente.model';
```

Código TypeScript completo del componente

El código completo de "**alta-cliente.ts**", para la definición de mi componente, quedaría así :

```
import { Component } from '@angular/core';
import { ClientesService } from './../clientes-service';
import { Cliente, Grupo } from './../cliente.model';


@Component({
  selector: 'app-alta-cliente',
  standalone: false,
  templateUrl: './alta-cliente.html',
  styleUrl: './alta-cliente.css'
})
export class AltaCliente {
  cliente: Cliente;
  grupos: Grupo[];
  constructor(private clientesService: ClientesService) {
    this.cliente = this.clientesService.nuevoCliente();
    this.grupos = this.clientesService.getGrupos();


  }
  nuevoCliente(): void {
    this.clientesService.agregarCliente(this.cliente);
```

```
    this.cliente = this.clientesService.nuevoCliente();
  }
  ngOnInit():void {
  }
}
```

Es importante aclarar los siguientes puntos:

1. El componente declara un par de atributos, el cliente y el array de grupos.
2. En el constructor, que se ejecuta lo primero, conseguimos una instancia del servicio de clientes, mediante la inyección de dependencias (ID).En este punto ya se ha recibido el servicio de clientes, por lo que lo puedo usar para generar los valores que necesito en los atributos del componente. Si queremos estar seguro de que el componente se ha inicializado totalmente, se utiliza el método **ngOnInit()** que se carga después del constructor (en este caso no es necesario).
3. El método **nuevoCliente()** es el que se ejecutará cuando, desde el formulario de alta, se produzca el envío de datos. En este código usamos el servicio **ClientesService**, para agregar el cliente y generar un cliente nuevo, para que el usuario pueda seguir dando de alta clientes *sin reemplazar los clientes anteriormente creados*.

Template del componente, con el formulario de alta de cliente

Vamos a ver ahora cuál es el HTML del componente de alta de clientes , que básicamente contiene un formulario.

Pero antes de ponernos con el HTML, vamos a hacer una importante tarea. Consiste en declarar en el módulo de clientes que se va a usar la directiva "**ngModel**". Para ello tenemos que hacer dos pasos:

En el archivo "**clientes-module.ts**" comenzamos por importar "**FormsModule**".

```
import { FormsModule } from '@angular/forms';
```

En el decorador, indicamos el imports del **FormsModule**.

```
imports: [
    CommonModule,
    FormsModule
  ]
```

Ahora veamos el código del template, en el que se reconoce el uso de la propiedad o atributo "**cliente**" declarada en el constructor, así como el **array** de **grupos**.

alta-cliente.html

```
<h2>Alta cliente</h2>
<p>
  <span>Nombre:</span>
  <input type="text" [(ngModel)]="cliente.nombre">
</p>
<p>
  <span>CIF:</span>
  <input type="text" [(ngModel)]="cliente.cif">
</p>
<p>
  <span>Dirección:</span>
```

```
  <input type="text" [(ngModel)]="cliente.direccion">
</p>
<p>
  <span>Grupo:</span>
  <select [(ngModel)]="cliente.grupo">
     @for(grupo of grupos;track grupo.id)
         {<option value="{{grupo.id}}"> {{grupo.nombre}}</option>}
  </select>
</p>
<p>
  <button (click)="this.nuevoCliente()">Guardar</button>
</p>
```

Usar el componente Alta cliente

Este componente, de alta de clientes, lo quiero usar desde el componente raíz de mi aplicación. Como el componente raíz está declarado en otro módulo (el raíz), necesitamos hacer que lo reconozca. Esto lo conseguimos en dos pasos:

1.- Agregar a **declarations[ ]** y a **exports [ ]** el componente **AltaCliente**.

En el módulo de clientes "**clientes-module.ts**" ,si no está ya, agregamos a **declarations[ ]** el propio componente (con su correspondiente **import)** para que lo reconozca su módulo, y lo exportamos en el array **exports[ ]** para poderlo usar en otros módulos, **clientes-module.ts** quedaría :

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ClientesService } from './clientes.service'; //Opcional
```

```
import { FormsModule } from '@angular/forms';
import { AltaCliente } from './alta-cliente/alta-cliente';
@NgModule({
  declarations: [
    AltaCliente
  ],
  imports: [
    CommonModule,
    FormsModule
  ],
  providers:[ClientesService], // Opcional
  exports:[ AltaCliente]
})
export class ClientesModule { }
```

2.- Importar en el módulo raíz

Ahora, en el módulo raíz, "**app-module.ts**", debemos declarar que se van a usar componentes que vienen del módulo **clientes-module.ts**. Para ello hacemos el correspondiente import:

```
import { ClientesModule } from './clientes/clientes-module';
```

Y luego declaramos el módulo en el array de imports:

```
 imports: [
    BrowserModule,
    AppRoutingModule,
    ClientesModule
```

```
  ],
```

Hechos los dos pasos anteriores, ya podemos usar el componente en el template. Para ello simplemente tenemos que escribir su tag, en el archivo "**app.html**" , sustituyendo el contenido que tenga por defecto.

```
<app-alta-cliente></app-alta-cliente>
```

Llegado a este punto, si todo ha ido bien, deberíamos de ver el componente de alta de clientes funcionando en nuestra página.

Crear el componente listado-clientes

Para acabar nuestra práctica vamos a crear un segundo componente de aplicación. Será el componente que nos muestre un listado de los clientes que se van generando.

Comenzamos con el comando:

```
ng generate component clientes/listadoClientes
```

Ahora el flujo de trabajo es similar al realizado para el componente anterior. Vamos detallando por pasos…

Creamos los import del servicio y de los tipos de datos del modelo.

```
import { Cliente, Grupo } from './../cliente.model';

import { ClientesService } from './../clientes-service';
```

Inyectamos el servicio en el constructor.

```
constructor(private clientesService: ClientesService) { }
```

En este componente tendremos como propiedad/atributo el array de clientes que el servicio vaya creando. Así pues, declaramos dicho array de clientes:

```
clientes: Cliente[];
```

Cuando se inicialice el componente tenemos que solicitar los clientes al servicio. Esto lo hacemos con el constructor.

```
constructor(private clientesService: ClientesService) {
    this.clientes = this.clientesService.getClientes();
  }
```

Código completo del componente ListadoClientes

Podemos ver a continuación el código TypeScript completo de cómo nos quedaría este segundo componente.

```
import { Component } from '@angular/core';
import { Cliente, Grupo } from './../cliente.model';
import { ClientesService } from './../clientes-service';


@Component({
  selector: 'app-listado-clientes',
  standalone: false,
  templateUrl: './listado-clientes.html',
  styleUrl: './listado-clientes.css'
})
export class ListadoClientes {
  clientes: Cliente[];
  constructor(private clientesService: ClientesService) {
```

```
    this.clientes = this.clientesService.getClientes();
  }
}
```

Código de la vista

Ahora podemos ver cómo sería la vista, código HTML, del listado de clientes.

```
<h2>
    Listado clientes
  </h2>
  @if (clientes.length === 0) {
    <div>No hay clientes por el momento</div>
  }
  <div>
    @for (cliente of clientes; track cliente.cif) {
      <article>
        <span>{{cliente.nombre}}</span>
        <span>{{cliente.cif}}</span>
        <span>{{cliente.direccion}}</span>
        <span>{{cliente.grupo}}</span>
      </article>
    }
  </div>
```

Damos un poco de estilos a los componentes editando el archivo de CSS. Por ejemplo, este sería un poco de CSS que podríamos colocar en el fichero "**listado-clientes.css**".

```
article {
    display: flex;
    border-bottom: 1px solid #ddd;
    padding: 10px;
    font-size: 0.9em;
  }
  span {
    display: inline-block;
    width: 22%;
    margin-right: 2%;
  }
```

Usar el componente del listado

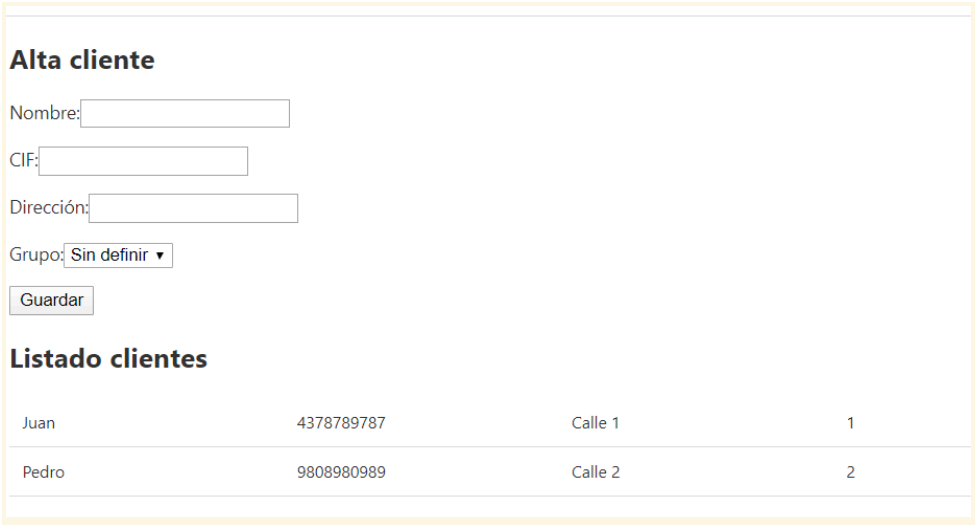
Para usar este componente de listado de clientes, ya que lo queremos invocar desde el módulo raíz, tienes que ampliar el exports del module "**clientes-module.ts**".

```
  exports: [
    AltaCliente,
    ListadoClientes
  ]
```

Como para el anterior componente, de alta de clientes, ya habíamos importado el módulo de clientes, no necesitamos hacer nada más. Ahora ya podemos usar el componente directamente en el template del componente raíz "**app.html**".

```
<app-alta-cliente></app-alta-cliente>
<app-listado-clientes></app-listado-clientes>
```

El aspecto de la aplicación que hemos realizado debería ser el siguiente:



Compilar y desplegar.

Finalmente, vamos a repetir este ejemplo guiado , pero sin usar **NgModules** , es decir, partiremos de `ng new clientessinmodulos` , sólo será necesario crear los componentes y servicios, sin módulos ni operaciones relacionadas con éstos, intentaremos resolver los errores que nos vayan surgiendo y con la ayuda del profe…