Angular

Biblioteca Angular Material

CertiDevs

Índice de contenidos

1. Angular Material

Angular Material es una biblioteca de componentes de interfaz de usuario (UI) para Angular que implementa el diseño Material Design de Google.

Proporciona un conjunto de **componentes** y herramientas predefinidos y de alta calidad que facilitan la creación de aplicaciones Angular con una apariencia y experiencia de usuario consistentes y modernas.

Para instalar y utilizar **Angular Material**, sigue estos pasos:

1.1. 1. Instalar Angular Material y Angular CDK:

Ejecuta el siguiente comando para instalar Angular Material y Angular CDK (Component Dev Kit) en tu proyecto Angular:

```
ng add @angular/material
```

Este comando también configura automáticamente el archivo `angular.json` y agrega las fuentes y estilos necesarios a tu aplicación.

1.2. 2. Importar módulos de Angular Material:

Para utilizar los componentes de **Angular Material**, primero debes importar los módulos correspondientes en el módulo de tu aplicación o en un módulo de características.

Por ejemplo, si deseas utilizar el botón y la barra de herramientas de Angular Material, importa `MatButtonModule` y `MatToolbarModule` en tu módulo:

```
// src/app/app.module.ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatToolbarModule } from '@angular/material/toolbar';

import { AppComponent } from './app.component';

@NgModule({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        MatButtonModule,
        MatToolbarModule
```

```
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

1.3. 3. Utilizar componentes de Angular Material:

Después de importar los módulos necesarios, puedes utilizar los componentes de Angular Material en tu aplicación.

Por ejemplo, puedes agregar un botón y una barra de herramientas utilizando los componentes `mat-button` y `mat-toolbar`:

```
<!-- src/app/app.component.html -->
<mat-toolbar color="primary">
    <span>Mi aplicación Angular Material</span>
</mat-toolbar>

<div>
    <button mat-raised-button color="accent">Botón de Angular Material</button>
</div>
```

Aquí hay algunos ejemplos de componentes de Angular Material que puedes utilizar en tu aplicación:

- **Botones**: Utiliza los componentes `mat-button`, `mat-raised-button`, `mat-stroked-button`, `mat-flat-button` y `mat-icon-button` para agregar diferentes tipos de botones a tu aplicación.
- **Barra de herramientas**: Utiliza el componente mat-toolbar para agregar una barra de herramientas en la parte superior de tu aplicación.
- **Iconos**: Utiliza el componente mat-icon para agregar iconos a tu aplicación. Asegúrate de importar MatIconModule en tu módulo.
- **Lista**: Utiliza el componente mat-list para crear listas en tu aplicación. Asegúrate de importar MatListModule en tu módulo.
- **Tarjetas**: Utiliza el componente mat-card para crear tarjetas en tu aplicación. Asegúrate de importar MatCardModule en tu módulo.
- **Formularios**: Utiliza los componentes mat-form-field, mat-input, mat-checkbox, mat-radio-button, mat-select y otros para crear formularios en tu aplicación.

2. Ejemplo

Para crear una pantalla completa con Angular Material que incluya una barra de navegación, una tabla de datos centrada con botones y un pie de página, sigue estos pasos:

2.1. 1. Importar los módulos necesarios

Primero, importa los módulos necesarios de Angular Material en tu módulo:

```
// src/app/app.module.ts
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule } from '@angular/material/button';
import { MatToolbarModule } from '@angular/material/toolbar';
import { MatTableModule } from '@angular/material/table';
import { MatPaginatorModule } from '@angular/material/paginator';

import { AppComponent } from './app.component';

@NgModule({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule,
        BrowserAnimationsModule,
        MatButtonModule,
        MatToolbarModule,
        MatTableModule,
        MatPaginatorModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

2.2. 2. Crear el componente de la tabla

Genera un nuevo componente para la tabla de datos:

```
ng generate component data-table
```

2.3. 3. Agregar la tabla de datos

Agrega la siguiente estructura de tabla de datos en el archivo `data-table.component.html`:

```
<!-- src/app/data-table/data-table.component.html -->
<div class="mat-elevation-z8">
<table mat-table [dataSource]="dataSource">
<!-- Columna ID -->
```

```
<ng-container matColumnDef="id">
<th mat-header-cell *matHeaderCellDef>ID</th>
<td mat-cell *matCellDef="let element">{{element.id}}</td>
</ng-container>

    <!-- Columna Nombre -->
    <ng-container matColumnDef="name">
      <th mat-header-cell *matHeaderCellDef>Nombre</th>
      <td mat-cell *matCellDef="let element">{{element.name}}</td>
    </ng-container>

    <!-- Columna Acciones -->
    <ng-container matColumnDef="actions">
      <th mat-header-cell *matHeaderCellDef>Acciones</th>
      <td mat-cell *matCellDef="let element">
        <button mat-raised-button color="primary">Editar</button>
        <button mat-raised-button color="warn">Eliminar</button>
      </td>
    </ng-container>

    <tr mat-header-row *matHeaderRowDef="displayedColumns"></tr>
    <tr mat-row *matRowDef="let row; columns: displayedColumns;"></tr>
  </table>
  <mat-paginator [pageSizeOptions]="[5, 10, 20]" showFirstLastButtons></mat-paginator>
</div>
```

Y en el archivo data-table.component.ts, agrega lo siguiente:

```
// src/app/data-table/data-table.component.ts
import { Component } from '@angular/core';
import { MatTableDataSource } from '@angular/material/table';

export interface UserData {
    id: string;
    name: string;
}

const EXAMPLE_DATA: UserData[] = [
    {id: '1', name: 'Nombre 1'},
    {id: '2', name: 'Nombre 2'},
    {id: '3', name: 'Nombre 3'},
    // ...
];

@Component({
    selector: 'app-data-table',
    templateUrl: './data-table.component.html',
    styleUrls: ['./data-table.component.css']
})
export class DataTableComponent {
```

```
    displayedColumns: string[] = ['id', 'name', 'actions'];
    dataSource = new MatTableDataSource(EXAMPLE_DATA);
}
```

2.4. 4. Crear y diseñar el componente de la barra de navegación

Genera un nuevo componente para la barra de navegación:

```
ng generate component nav-bar
```

A continuación, agrega el siguiente código en el archivo nav-bar.component.html:

```
<!-- src/app/nav-bar/nav-bar.component.html -->
<mat-toolbar color="primary">
  <span>Mi aplicación Angular Material</span>
</mat-toolbar>
```

2.5. 5. Crear y diseñar el componente del pie de página

Genera un nuevo componente para el pie de página:

```
ng generate component footer
```

A continuación, agrega el siguiente código en el archivo footer.component.html:

```
<!-- src/app/footer/footer.component.html -->
<footer class="footer">
    <mat-toolbar color="accent">
        <span class="footer-content">Derechos reservados &copy; 2023</span>
    </mat-toolbar>
</footer>
```

Y en el archivo footer.component.css, agrega los siguientes estilos:

```
/* src/app/footer/footer.component.css */
.footer {
    position: absolute;
    bottom: 0;
    width: 100%;
}

.footer-content {
```

```
    margin: 0 auto;
}
```

2.6. 6. Integrar los componentes en el componente principal

En el archivo app.component.html, agrega los componentes creados previamente y aplica estilos para centrar la tabla:

```
<!-- src/app/app.component.html -->
<app-nav-bar></app-nav-bar>

<div class="container">
    <app-data-table></app-data-table>
</div>

<app-footer></app-footer>
```

En el archivo app.component.css, agrega los siguientes estilos:

```
/* src/app/app.component.css */
.container {
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: calc(100vh - 64px - 48px); /* Altura de la ventana menos la barra de
navegación y el pie de página */
}
```

Con estos pasos, tendrás una pantalla completa en Angular Material con una barra de navegación, una tabla de datos centrada con botones y un pie de página.

3. Centrar elementos con CSS

Para centrar una tabla de Angular Material en una plantilla de componente Angular, primero necesitas instalar Angular Material y configurar los estilos y animaciones.

Asumiendo que ya has hecho eso, sigue estos pasos:

Importa el módulo `MatTableModule` en tu módulo de características:

```
import { MatTableModule } from '@angular/material/table';

@NgModule({
    imports: [
```

```
        // ...
        MatTableModule,
    ],
})
export class AppModule { }
```

Crea una tabla de Angular Material en la plantilla de tu componente:

```
<div class="table-container">
<table mat-table [dataSource]="dataSource">
<!-- Define las columnas de la tabla aquí -->
<ng-container matColumnDef="columna1">
<th mat-header-cell *matHeaderCellDef> Columna 1 </th>
<td mat-cell *matCellDef="let elemento"> {{elemento.columna1}} </td>
</ng-container>

    <!-- Encabezado y filas de la tabla -->
    <tr mat-header-row *matHeaderRowDef="columnasAMostrar"></tr>
    <tr mat-row *matRowDef="let row; columns: columnasAMostrar;"></tr>
  </table>
</div>
```

En este ejemplo, hemos creado una tabla simple de Angular Material dentro de un contenedor div con la clase table-container. Puedes reemplazar dataSource y columnasAMostrar con tus propios datos y columnas.

Añade estilos CSS para centrar la tabla en el archivo de estilos del componente (por ejemplo, `my-component.component.css`):

```
.table-container {
  display: flex;
  justify-content: center;
}
```

Aquí, utilizamos Flexbox para centrar la tabla horizontalmente. Si también deseas centrarla verticalmente, puedes agregar las siguientes reglas CSS:

```
.table-container {
  display: flex;
  justify-content: center;
  align-items: center;
  min-height: 100vh;
}
```

Con estos pasos, habrás centrado una tabla de Angular Material en una plantilla de componente Angular. Ajusta el código según sea necesario para adaptarse a tu proyecto y diseño específico.

4. Centrar elementos con FlexLayout

Angular Material utiliza el paquete @angular/flex-layout para proporcionar utilidades de diseño flexibles.

Primero, necesitas instalar `@angular/flex-layout`:

```
npm install @angular/flex-layout
```

Luego, importa `FlexLayoutModule` en tu módulo de características:

```
import { FlexLayoutModule } from '@angular/flex-layout';

@NgModule({
    imports: [
        // ...
        FlexLayoutModule,
    ],
})
export class AppModule { }
```

Una vez hecho esto, puedes utilizar las directivas de Angular Material para crear un diseño flex y centrar la tabla en la plantilla de tu componente:

```
<div fxLayout="column" fxLayoutAlign="center center" style="height: 100vh;">
    <table mat-table [dataSource]="dataSource">
    <!-- Define las columnas de la tabla aquí -->
    <ng-container matColumnDef="columna1">
    <th mat-header-cell *matHeaderCellDef> Columna 1 </th>
    <td mat-cell *matCellDef="let elemento"> {{elemento.columna1}} </td>
    </ng-container>

    <!-- Encabezado y filas de la tabla -->
    <tr mat-header-row *matHeaderRowDef="columnasAMostrar"></tr>
    <tr mat-row *matRowDef="let row; columns: columnasAMostrar;"></tr>
    </table>
</div>
```

En este ejemplo, hemos utilizado las directivas `fxLayout` y `fxLayoutAlign` para crear un diseño flex y centrar la tabla tanto horizontal como verticalmente.

La propiedad `style="height: 100vh;"` se utiliza para asegurar que el contenedor ocupe todo el espacio vertical disponible.

4.1. Otro ejemplo de centrado con formulario

Para hacer que el diseño sea responsive, puedes utilizar las directivas `fxLayout.lt-md`, `fxLayout.md` y `fxLayout.gt-md` (donde lt significa "menor que", md es "medio" y gt significa "mayor que").

Estas directivas permiten aplicar diferentes configuraciones de diseño en función del tamaño de la pantalla.

Por ejemplo, si quieres aplicar diferentes alineaciones según el tamaño de la pantalla, puedes hacer lo siguiente:

```
<div
fxLayout="column"
fxLayout.lt-md="row"
fxLayoutAlign="center center"
fxLayoutAlign.lt-md="start center"
style="height: 100vh;"
>
    <form [formGroup]="miFormulario" (ngSubmit)="onSubmit()" class="formulario">
    <!-- Contenido del formulario -->
    </form>
</div>
```

En este ejemplo, se utiliza `fxLayout="column"` para dispositivos más grandes y `fxLayout.lt-md="row"` para dispositivos más pequeños. La alineación también se ajusta en función del tamaño de la pantalla.

`Flex Layout` ofrece una gran cantidad de directivas y opciones para crear diseños adaptables y personalizados. Si bien es similar al sistema de grid de Bootstrap en algunos aspectos, se basa en Flexbox y proporciona sus propias funcionalidades y características únicas.

5. Otras bibliotecas

- **NG Bootstrap** (https://ng-bootstrap.github.io/): Implementación de Bootstrap para Angular, que ofrece componentes y estilos de Bootstrap adaptados para su uso en aplicaciones Angular sin depender de jQuery.
- **PrimeNG** (https://www.primefaces.org/primeng/): Colección de componentes de UI ricos para Angular, ofreciendo una amplia gama de componentes de alta calidad y personalizables.
- **Angular CDK** (https://material.angular.io/cdk/categories): Kit de desarrollo de componentes para Angular que proporciona herramientas y utilidades para construir componentes de UI personalizados sin depender de Angular Material.
- **Akita** (https://datorama.github.io/akita/): Biblioteca de gestión de estado para Angular basada en la simplicidad y la escalabilidad, proporcionando una solución fácil de aprender y de alto rendimiento.
- **Apollo Angular** (https://www.apollographql.com/docs/angular/): Integración de GraphQL para Angular, que facilita la consulta y manipulación de datos de API utilizando GraphQL en lugar de

REST.

- **AngularFire** (https://github.com/angular/angularfire): Integración oficial de Firebase para Angular, que simplifica el desarrollo de aplicaciones en tiempo real, la autenticación y el almacenamiento de datos utilizando los servicios de Firebase.
- **ngx-translate** (https://github.com/ngx-translate/core): Biblioteca de internacionalización (i18n) para Angular, que facilita la traducción de textos y la localización de aplicaciones.
- **NgRx** (https://ngrx.io/): Biblioteca para la gestión del estado de la aplicación basada en Redux, facilitando el desarrollo de aplicaciones Angular con arquitectura de flujo de datos unidireccional.