Nombre: Rodrigo Mena Serna

Asignatura: Diseño de Interfaces Web

Segundo curso de Desarrollo de Aplicaciones Web

Nombre de la práctica: Material para Angular

Contenido

Primeros pasos con material angular:

Esta guía explica cómo configurar un proyecto Angular para comenzar a usar Angular Material. Incluye información sobre los requisitos previos, la instalación de Angular Material y, opcionalmente, la visualización de un componente Material de muestra en su aplacación para verificar su configuración.

Instalar material angular:

Use el esquema de instalación de Angular CLI para configurar su proyecto de Angular Material ejecutando el siguiente comando:

ng add @angular/material

El ng add comando instalará Angular Material, Component Dev Kit (CDK), Angular Animations y le hará las siguientes preguntas para determinar qué características incluir:

1. Elija un nombre de tema prediseñado “personalizado” para un tema personalizado: puede elegir entre temas de diseño de materiales preconstruidos o configurar un tema personalizado extensible.
2. Configure estilos tipográficos globales de material angular: ya sea para aplicar los estilos de tipografía global a su aplicación.
3. Configure las animaciones del navegador para material angular: importar a su aplicación habilita el sistema de animación BrowserAnimationsModule de Angular. Rechazar esto deshabilitará la mayoría de las animaciones de Angular Material.

El ng add comando realizará además las siguientes acciones:

- Agregar dependencias del proyecto a package.json.
- Agregue la fuente Roboto a su index.html.
- Agregue la fuente del icono Material Desing a su index.html.
- Agregue algunos estilos CSS globales a:
    - Eliminar márgenes de body.
    - Establecer height: 100%_html body.
    - Establecer Roboto como fuente de aplicación predeterminada.

El material angular ahora está configurado para usarse en su aplicación.

Mostrar un componente:

Mostraremos un componente de alternancia de diapositivas en su aplicación y verifiquemos que todo funcione.

Debe importar el MatSlideToggleModule que desea mostrar agregando las siguiente líneas a su app.module.ts archivo.

```
import { MatSlideToggleModule } from '@angular/material/slide-
toggle';

@NgModule ({
  imports: [
    MatSlideToggleModule,
  ]
})
class AppModule {}
```

Agregue la <mat-slide-toggle> etiqueta al app.component.html:

```
<mat-slide-toggle>Toggle me!</mat-slide-toggle>
```

Ejecute su servidor de desarrollo local:

```
ng serve
```

Luego apunte su navegador a http://localhost:4200.

Debería ver el componente "slide toggle" de Material en la página.

Generar componentes:

Angular material es un conjunto de componentes visuales que nos permite desarrollar interfaces de usuario consistentes.

Disponemos de un gran conjunto de componentes:

- Autocomplete.
- Badge.
- Botton Sheet.
- Button.
- Button toggle.
- Card.
- Checkbox.
- Chips.
- Core.
- Datepicker.
- Dialog.
- Divider.
- Expansión Panel.
- Form field.
- Grid list.
- Icon.

- Input.
- List.
- Menu.
- Paginator.
- Progress bar.
- Progress spinner.
- Radio button.
- Ripples.
- Select.
- Slider toggle.
- Slider.
- Snackbar.
- Sort header.
- Stepper.
- Table.
- Tabs.
- Toolbar.
- Tooltip.
- Tree.

Instalación y Generación de Código

Angular Material viene empaquetado con esquemas Angular CLI para facilitar la creación de aplicaciones de materiales.

Esquemas de instalación:

Los esquemas se incluyen con ambos @angular/cdk y @angular/material. Una vez que instale los paquetes npm, estarán disponibles a través de Angular CLI.

Con el siguiente comando, se instalará Angular Material, Component Dev Kit (CDK) y Angular Animations en su proyecto. Luego ejecutará el esquema de instalación.

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

En caso de que solo desee instalar el @angular/cdk, también hay esquemas para el hit de desarrollo de componentes.

```
ng add @angular/cdk
```

El esquema de Angular Material ng add lo ayuda a configurar un priyecto de Angular CLI que usa Material. Correr ng add:

- Asegúrese de que las dependencias del proyecto se coloquen en package.json.
- Habilite el módulo BrowserAnimationsModule de su aplicación.
- Agregue un tema preconstruido o un tema personalizado.
- Agregue fuentes Roboto a su index.html.

- Agregue la fuente Material Icon a su index.html.
- Añadir estilos globales a:
    - Eliminar márgenes de body.
    - Establecer height: 100%_html body
    - Haz que Roboto sea la fuente predeterminada de tu aplicación.

Esquema de componentes

Además del esquema de instalación, Angular Material viene con múltiples esquemas que se pueden usar para generar fácilmente componentes de Material Desing:

Nombre	Descripción
`address-form`	Componente con un grupo de formularios que utiliza controles de formulario de Material Design para solicitar una dirección de envío
`navigation`	Crea un componente con un sidenav de Material Design sensible y una barra de herramientas para mostrar el nombre de la aplicación
`dashboard`	Componente con varias tarjetas y menús de Material Design alineados en un diseño de cuadrícula
`table`	Genera un componente con una tabla de datos de Material Design que admite clasificación y paginación
`tree`	Componente que visualiza de forma interactiva una estructura de carpetas anidadas utilizando el `<mat-tree>` componente

Además, Angular CDK también viene con una colección de esquemas de componentes:

Nombre	Descripción
`drag-drop`	Componente que utiliza las `@angular/cdk/drag-drop` directivas para crear una lista de tareas interactiva

Esquema de formulario de dirección

Ejecutar el address-form esquema genera un nuevo componente Angular que se puede usar para comenzar con un grupo de formularios de Material Desing que consta de:

- Campos de formulario de diseño de Material.
- Controles de radio de diseño de Material.
- Botones de diseño de Material.

```
ng generate @angular/material:address-form <component-name>
```

Esquema de navegación

El navigation esquema creará un nuevo componente que incluye una barra de herramientas con el nombre de la aplicación y una navegación lateral receptiva basada en puntos de interrupción de material.

```
ng generate @angular/material:navigation <component-name>
```

Esquema de tabla

El esquema de la tabla creará un componente que representa un Material Angular <table> que ha sido preconfigurado con una fuente de datos para ordenar y paginar.

```
ng generate @angular/material:table <component-name>
```

Esquema de tablero

El dashboard esquema creará un nuevo componente que contiene una lista de cuadrícula dinámica de tarjetas de Material Desing.

```
ng generate @angular/material:dashboard <component-
name>
```

Esquema de árbol

El tree esquema se puede usar para generar rápidamente un componente Angular que usa el <mat-tree> componente Material Angular para visualizar una estructura de carpetas anidadas.

```
ng generate @angular/material:tree <component-name>
```

Esquema de arrastrar y soltar

El drap-drop esquema lo proporciona @angular/cdk y se puede usar para generar un componente que use las directivas de arrastrar y soltar de CDK.

```
ng generate @angular/cdk:drag-drop <component-name>
```

Creación del esqueleto básico de un proyecto con Angular Material

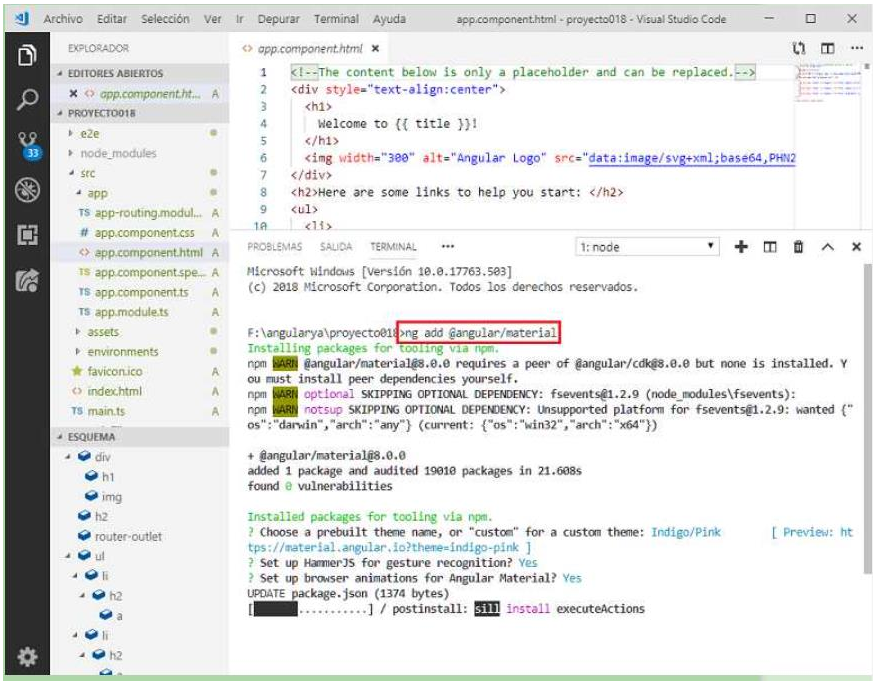
Crearemos una aplicación que disponga de un menú lateral a la izquierda con los nombres de tres países.

1. Crearemos primero el proyecto (es importante la opción “routing” ya que la aplicación la requiere, si no disponemos dicho valor elegir la que queremos utilizar routing cuando nos consulte Angular CLI).

```
ng new proyecto018 --routing
```

2. Procedemos a instalar todas las dependencias de Angular Material ayudados por Angular CLI mediante el comando ‘add’ (se nos pide elegir el tema y algunos otros datos, podemos dejar los valores por defecto presionando la tecla ‘Enter’):

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



Luego de esto ya tenemos en la carpeta node_module/angular/material todos los componentes de Angular Material para ser utilizados en nuestro proyecto.

3. Crearemos la barra lateral que dispondrá los enlaces:

```
ng generate @angular/material:material-nav --name barraLateral
```

Se crea un componente llamado “BarraLateralComponent” con los 4 archivos correspondientes y la modificación del archivo “app.module.ts”.

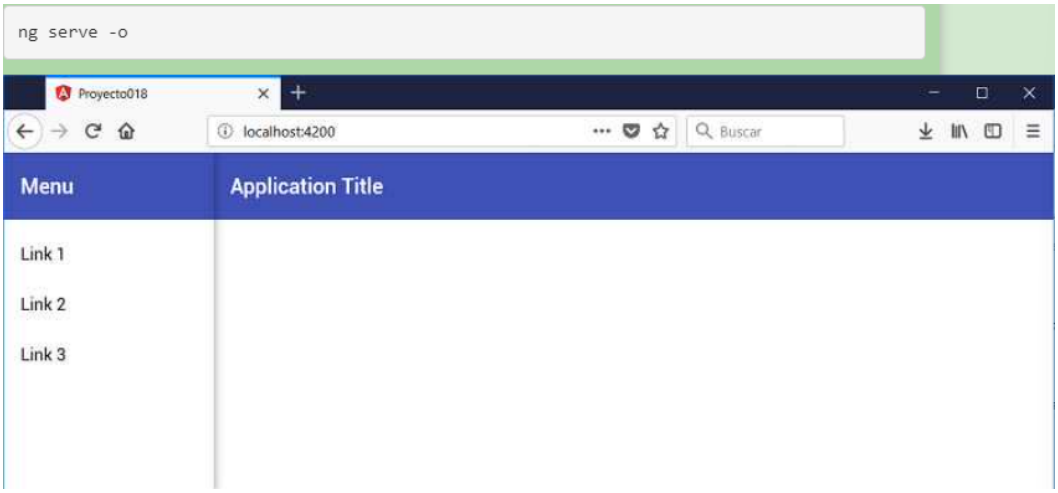
```
F:\angularya\proyecto018>ng generate @angular/material:material-nav --name barraLateral
CREATE src/app/barra-lateral/barra-lateral.component.html (954 bytes)
CREATE src/app/barra-lateral/barra-lateral.component.spec.ts (748 bytes)
CREATE src/app/barra-lateral/barra-lateral.component.ts (608 bytes)
CREATE src/app/barra-lateral/barra-lateral.component.css (129 bytes)
UPDATE src/app/app.module.ts (902 bytes)
```

4. Modificamos ahora el archivo ‘app.component.html’ donde creamos una etiqueta del componente que acabamos de crear en el paso anterior.

app.component.html

```
<app-barra-lateral></app-barra-lateral>
```

Si en este momento ejecutamos la aplicación ya podemos ver la barra de navegación lateral:



Hasta ahora solo hemos empleado Angular CLI para llegar a este ligar, es decir, crear el proyecto y el componente del menú lateral.

5. Creamos los tres componentes pais1, pais2, pais3 que tienen por objetivo mostrar datos de países:

```
ng generate component pais1
```

```
ng generate component pais2
```

```
ng generate component pais3
```

6. Podemos abrir el archivo 'app.module.ts' y ver que se han importado los tres componentes que acabamos de crear.

```
import { BrowserModule } from '@angular/platform-brows
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/plat
import { BarraLateralComponent } from './barra-lateral
import { LayoutModule } from '@angular/cdk/layout';
import { MatToolbarModule, MatButtonModule, MatSidenav
import { Pais1Component } from './pais1/pais1.componen
import { Pais2Component } from './pais2/pais2.componen
import { Pais3Component } from './pais3/pais3.componen

@NgModule({
  declarations: [
    AppComponent,
    BarraLateralComponent,
    Pais1Component,
    Pais2Component,
    Pais3Component
```

7. Debemos ahora modificar el archivo ‘app-routing.module.ts’ con las tres rutas:

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router'
import { Pais1Component } from './pais1/pais1.componen
import { Pais2Component } from './pais2/pais2.componen
import { Pais3Component } from './pais3/pais3.componen

const routes: Routes = [
  {
    path:'pais1',
    component:Pais1Component
  },
  {
    path:'pais2',
    component:Pais2Component
  },
  {
    path:'pais3',
    component:Pais3Component
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

8. Mediante la etiqueta ‘router-outlet’ indicamos el lugar que debe mostrar el componente especificado por la ruta configurada en el archivo ‘app-routing.module.ts’, para esto abrimos el archivo “barra-lateral.component.html” y agregaremos la etiqueta ‘router-outlet’:

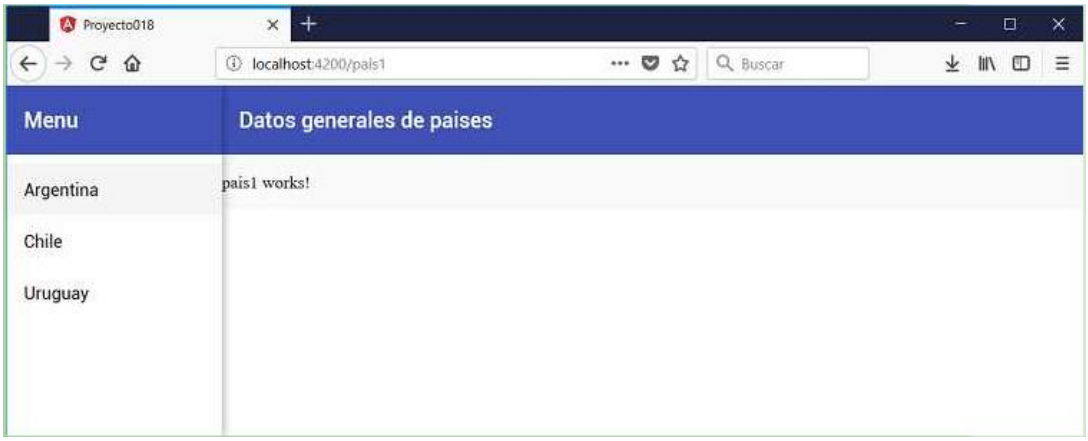
```
<mat-sidenav-container class="sidenav-container">
  <mat-sidenav #drawer class="sidenav" fixedInViewport
      [attr.role]="(isHandset$ | async) ? 'dialog' : '
      [mode]="(isHandset$ | async) ? 'over' : 'side'"
      [opened]="(isHandset$ | async) === false">
    <mat-toolbar>Menu</mat-toolbar>
    <mat-nav-list>
      <a mat-list-item routerLink="/pais1">Argentina</
      <a mat-list-item routerLink="/pais2">Chile</a>
      <a mat-list-item routerLink="/pais3">Uruguay</a>
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content>
    <mat-toolbar color="primary">
      <button
        type="button"
        aria-label="Toggle sidenav"
        mat-icon-button
        (click)="drawer.toggle()"
        *ngIf="isHandset$ | async">
        <mat-icon aria-label="Side nav toggle icon">me
      </button>
      <span>Datos generales de paises</span>
    </mat-toolbar>
    <!-- Add Content Here -->
    <router-outlet></router-outlet>
  </mat-sidenav-content>
</mat-sidenav-container>
```

Hemos modificado del código generado los hipervínculos a las distintas rutas:

```
      <a mat-list-item routerLink="/pais1">Argentina</a>
      <a mat-list-item routerLink="/pais2">Chile</a>
      <a mat-list-item routerLink="/pais3">Uruguay</a>
```

Y también es fundamental agregar la etiqueta ‘router-outlet’ para indicar donde se muestran los componentes.

Si ejecutamos ahora la aplicación los tres hipervínculos activan el componente respectivo donde se mostrará información referente a cada país:



9. Solo nos queda componer los contenidos de las componentes ‘Pais1Component’, ‘Pais2Component’ y ‘Pais3Component’.

pais1.component.html

```
<div class="datos">
  <p><strong>Nombre del país:</strong>Argentina</p>
  <p>
  <img src="assets/imagenes/argentina.png">
  </p>
  <h3>Datos generales.</h3>
  <p>Argentina, llamada oficialmente República Argenti
    ubicado en el extremo sur y sudeste de dicho subco
    democrática, representativa y federal.</p>
  <p>La Argentina está organizada como un Estado feder
    Estado nacional y 24 estados autogobernados,11?12?
    autónoma de Buenos Aires designada como Capital Fe
    Cada estado tiene autonomía política, constitución
    Las 23 provincias mantienen todos los poderes no d
    garantizan la autonomía de sus municipios.13?14?</
</div>
```

pais2.component.html

```
<div class="datos">
    <p><strong>Nombre del país:</strong>Chile</p>
    <p>
    <img src="assets/imagenes/chile.png">
    </p>
    <h3>Datos generales.</h3>
    <p>Chile es un país de América ubicado en el extre
      Su nombre oficial es República de Chile26? y su
      Primer país sudamericano en ingresar a la Organi
      Desarrollo Económicos, Chile es una de las econo
      crecido desde mediados de la década de 1980.</p>
    <p>Antes del descubrimiento de América, las tierra
      se llamaban Chili en la tradición indígena.46? U
      Nueva Toledo, los conquistadores españoles sigui
      región del sur, a veces también conocida como «v
      posteriormente a todo el actual país.47?</p>
</div>
```

pais3.component.html

```
<div class="datos">
    <p><strong>Nombre del país:</strong>Uruguay</p>
    <p>
    <img src="assets/imagenes/uruguay.png">
    </p>
    <h3>Datos generales.</h3>
    <p>Uruguay, oficialmente República Oriental del Ur
      situado en la parte oriental del Cono Sur americ
      de Río Grande del Sur—, al oeste con Argentina —
      tiene costas en el océano Atlántico al sureste y
      Abarca 176?215 km² y es el segundo país más pequ
      Según los datos del último censo del INE en 2011
      habitantes, por lo que figura en la décima posic
      oficialmente República Oriental del Uruguay, es
      parte oriental del Cono Sur americano. Limita al
      —estado de Río Grande del Sur—, al oeste con Arg
      tiene costas en el océano Atlántico al sureste y
      Abarca 176?215 km² y es el segundo país más pequ
      Según los datos del último censo del INE en 2011
      es de 3.290.454 habitantes, por lo que figura en
      sudamericanos.</p>
</div>
```

Lo nuevo es que hemos agregado las tres imágenes de las banderas en una carpeta llamada imágenes que se debe crear dentro de la carpeta 'assets':

```
    <img src="assets/imagenes/uruguay.png">
```

Si ejecutamos nuevamente la aplicación tenemos como resultado:

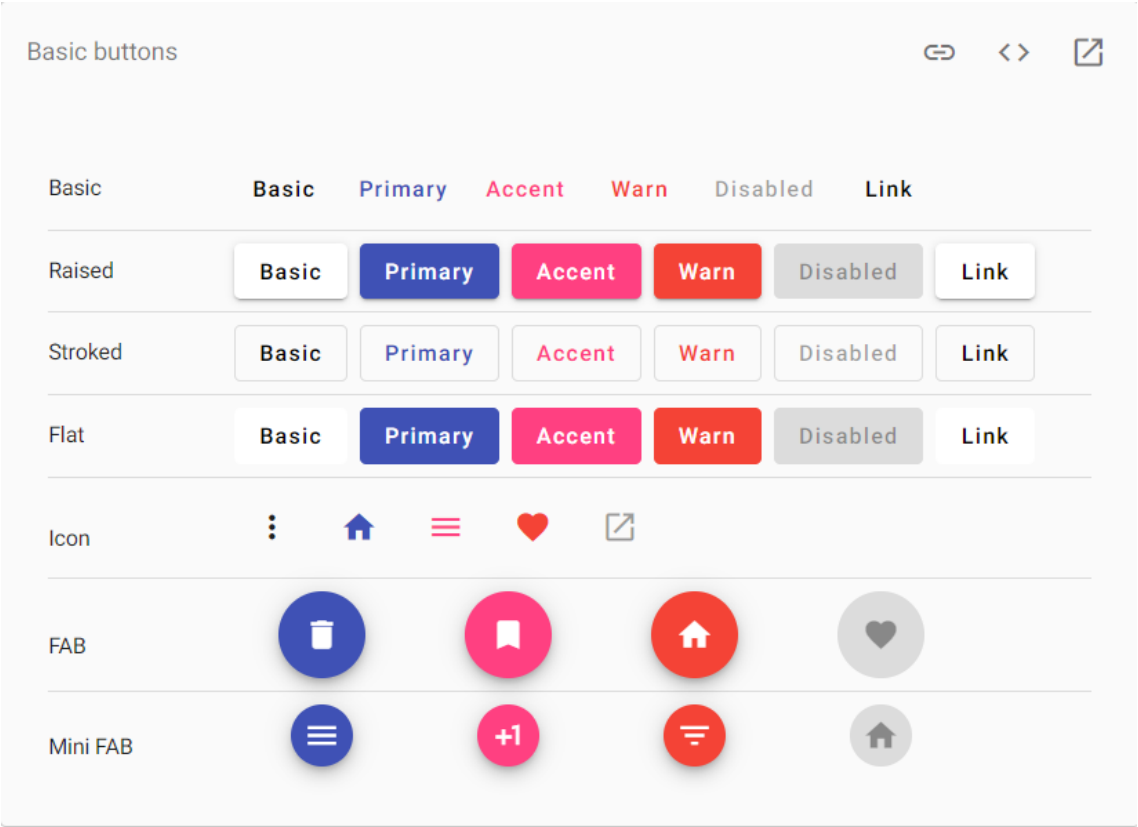
Componentes:

Angular Material ofrece una amplia variedad de componentes de interfaz de usuario basados en la especificación Material Desing.

A continuación, se mostrarán varios componentes a utilizar:

Botón:

Los botones de material angular son elementos nativos <button> o <a> mejorados con estilo de Material Desing.



Los elementos nativos <button> y <a> siempre se utilizan para proporcionar la experiencia más sencilla y accesible para los usuarios. Un <button> elemento debe usarse cada vez que se realiza alguna acción. Se debe usar <a> siempre que el usuario navegue a otra vista.

Hay varias variantes de botones, cada una aplicada como un atributo:

Atributo	Descripción
`mat-button`	Botón de texto rectangular sin elevación
`mat-raised-button`	Botón contenido rectangular con elevación
`mat-flat-button`	Botón contenido rectangular sin elevación
`mat-stroked-button`	Botón con contorno rectangular sin elevación
`mat-icon-button`	Botón circular con fondo transparente, destinado a contener un icono
`mat-fab`	Botón circular con elevación, el valor predeterminado es el color de acento del tema
`mat-mini-fab`	Igual que `mat-fab` pero más pequeño

Tematización:

Los botones se pueden colorear en términos del tema actual utilizando la propiedad "color" para establecer el color de fondo en primary, accent o warn.

Botones fabulosos extendidos:

Los botones tradicionales de FAB son circulares y solo tienen espacio para un solo icono. Sin embargo, puede agregar el atributo "extended" para permitir que se expanda en forma de rectángulo redondeado con espacio para una etiqueta de texto además del icono. Solo las FAB admiten el atributo "extended", las mini FAB no.

```
<button mat-fab extended>
  <mat-icon>home</mat-icon>
  Home
</button>
```

Accesibilidad:

Angular Material utiliza elementos nativos <button> y <a> para garantizar una experiencia accesible de forma predeterminada. Un elemento <button> debe usarse para cualquier interacción que realice una acción en la página actual. Se debe usar <a> para cualquier interacción que navegue a otra URL. Todas las mejores prácticas de accesibilidad estándas para botones y anclas se aplican a "MatButton".

Botones con iconos:

Los botones o enlaces que contienen solo iconos (como mat-fab, mat-mini-fab y mat-icon-button) deben recibir una etiqueta significativa a través aria-label o aria-labelledby.

Card:

<mat-card> es un contenedor de contenido para texto, fotos y acciones en el contexto de un solo tema.

Secciones básicas de cartas:

La tarjeta más básica solo necesita un elemento <mat-card> con algún contenido. Sin embargo, Angular Material proporciona una serie de secciones preestablecidas que puede usar dentro de un <mat-card>:

Elemento	Descripción
`<mat-card-header>`	Sección anclada a la parte superior de la tarjeta (agrega relleno)
`<mat-card-content>`	Contenido de la tarjeta principal (agrega relleno)
`<img mat-card-image>`	imagen de la tarjeta. Estira la imagen al ancho del contenedor.
`<mat-card-actions>`	Contenedor para botones en la parte inferior de la tarjeta (agrega relleno)
`<mat-card-footer>`	Sección anclada a la parte inferior de la tarjeta

Estos elementos sirven principalmente como contendores de contenido prediseñados sin ninguna API adicional. Sin embargo, la propiedad align en <mat-card-actions> puede usarse para posicionar las acciones en 'start' o 'end' del contenedor.

Relleno de tarjeta:

El elemento <mat-card> en sí no agrega ningún relleno alrededor de su contenido. Esto permite a los desarrolladores personalizar el relleno a su gusto aplicando relleno a los elementos que colocan en la tarjeta.

En muchos casos, es posible que los desarrolladores solo deseen el relleno estándar especificado en las especificaciones de Material Desing. En este caso, se pueden utilizar las secciones <mat-card-header>, <mat-card-content> y <mat-card-footer>.

- <mat-card-content> agrega relleno estándar a lo largo de sus lados, así como a lo largo de la parte superior si es el primer elemento del <mat-card>, y en la parte inferior si es el último elemento del <mat-card>.
- <mat-card-header> agregar relleno estándar a lo largo de los lados y la parte superior.
- <mat-card-actions> agrega relleno apropiado para los botones de acción en la parte inferior de una tarjeta.

Encabezado de tarjetas:

Una <mat-card-header> puede contener cualquier contenido, pero hay varios elementos predefinidos que se pueden usar para crear un encabezado enriquecido para una tarjeta. Éstos incluyen:

Elemento	Descripción
`<mat-card-title>`	Un título dentro del encabezado.
`<mat-card-subtitle>`	Un subtítulo dentro del encabezado.
`<img mat-card-avatar>`	Una imagen utilizada como avatar dentro del encabezado.

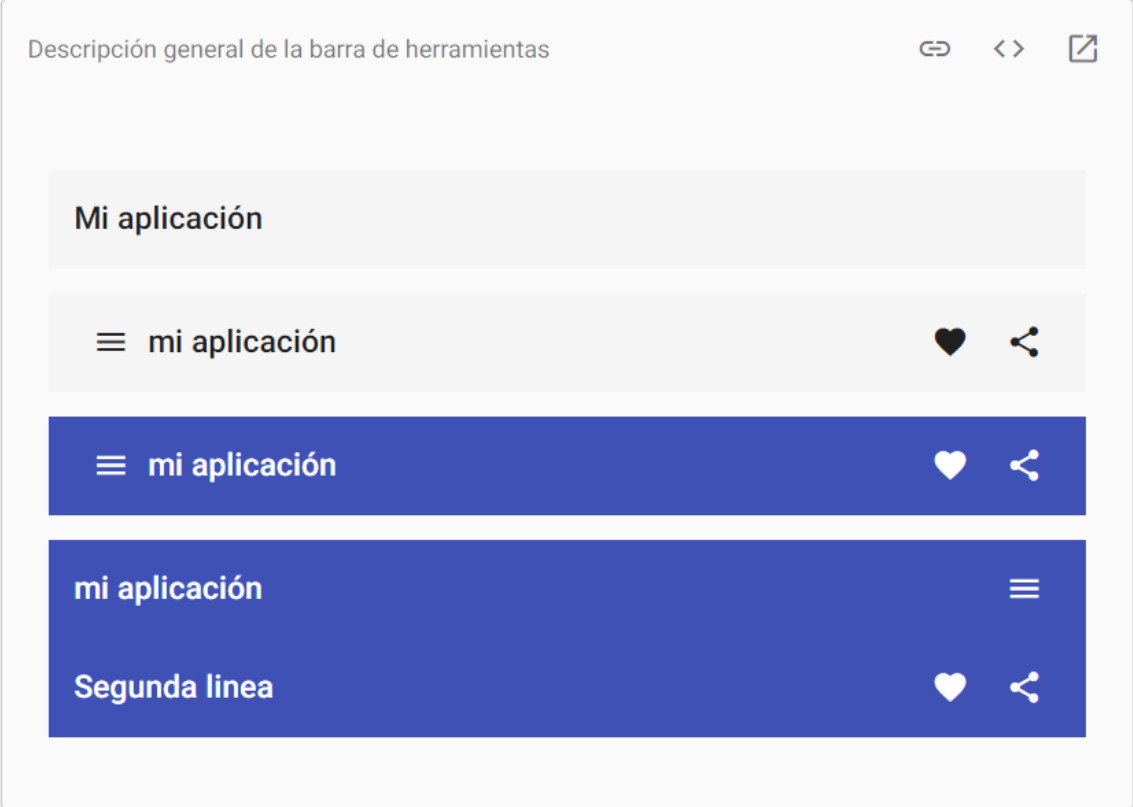
Grupos de títulos:

<mat-card-title-group> se puede utilizar para combinar un título, un subtítulo y una imagen en una sola sección. Este elemento puede contener:

- <mat-card-title>
- <mat-card-subtitle>
- Uno de:
    - <img mat-card-sm-image>
    - <img mat-card-md-image>
    - <img mat-card-lg-image>

Toolbar:

<mat-toolbar> es un contenedor para encabezados, títulos o acciones.



Única fila:

En la mayoría de las situaciones, se colocará una barra de herramientas en la parte superior de su aplicación y solo tendrá una fila que incluya el título de su aplicación.

```
< mat-toolbar >
  < span > Mi aplicación </ span >
</ mat-toolbar >
```

Múltiples filas:

Las especificaciones de Material Desing describen que las barras de herramientas también pueden tener varias filas. La creación de barras de herramientas con varias filas en Angular Material se puede realizar colocando <mat-toolbar-row> elementos dentro de un archivo <mat-toolbar>.

```
< mat-toolbar-row >
  < span > Barra de herramientas personalizada </ span >
</ mat-toolbar-row >
```

Temas Angular Material

¿Qué es tematizar?

El sistema de temas de Angular Material le permite personalizar el color, la tipografía y los estilos de densidad para los componentes de su aplicación. El sistema de tematización se basas en la especificación Material Desing de Google.

SASS

Las APIs de temas de Angular Material se construyen con Sass. Este documento asume la familiaridad con los conceptos básicos de CSS y SASS, incluidas variables, funciones y mixins.

Puede usar Angular Material sin SASS usando un tema preconstruido. Sin embargo, el uso directo de la API SASS de la biblioteca le brinda el mayor control sobre los estilos en su aplicación.

Paletas

Una paleta es una colección de colores que representa una parte del espacio de color. Cada valor de esta colección se denomina matiz. En Material Desing, cada tono de una paleta tiene un número de identificación. Estos números de identificación incluyen 50 y luego cada 100 valores entre 100 y 900. Los números ordenan los tonos dentro de una paleta de más claro a más oscuro.

Angular Material representa una paleta como un mapa Sass. Este mapa contiene los tonos de la paleta y otro mapa anidado de colores de contraste para cada uno de los tonos. Los colores de contraste sirven como color de texto cuando se utiliza una matriz como color de fondo. El siguiente ejemplo demuestra la estructura de una paleta.

```
$indigo-palette: (
 50: #e8eaf6,
 100: #c5cae9,
 200: #9fa8da,
 300: #7986cb,
 // ... continues to 900
 contrast: (
   50: rgba(black, 0.87),
   100: rgba(black, 0.87),
   200: rgba(black, 0.87),
   300: white,
   // ... continues to 900
 )
);
```

Crea tu propia paleta

Puedes crear su propia paleta definiendo un mapa Sass que coincida con la estructura descrita en la sección Paletas anterior. El mapa debe definir tonalidades para 50 y cada centena entre 100 y 900. El mapa también debe definir un ‘contrast’ mapa con colores de contraste para cada tonalidad.

Paletas predefinidas

Angular Material ofrece paletas predefinidas basadas en la versión 2014 de la especificación Material Desing.

Además de los tonos numerados del cero al 900, las paletas de Material Desing de 2014 incluyen tonos distintivos numerados como A100, A200, A400 y A700. Angular Material no requiere de estos tonos, pero puede usarlos al definir un tema como se describe en a continuación:

```
@use '@angular/material' as mat;

$my-palette: mat.$indigo-palette;
```

Temas

Un tema es una colección de opciones de color, tipografía y densidad. Cada tema incluye tres paletas que determinan los colores de los componentes:

- Una paleta primaria para el color que aparece con mayor frecuencia en toda su aplicación.
- Una paleta de acento, o secundaria, que se utiliza para resaltar de forma selectiva partes claves de la interfaz de usuario.
- Una paleta de advertencia o error utilizada para advertencias y estado de error.

Puede incluir los estilos CSS para un tema en su aplicación de dos maneras: definiendo un tema personalizado con Sass o importante un archivo CSS de tema preconstruido.

Temas personalizados con SASS

Un archivo de tema es un archivo Sass que llama a los mixins de Angular Material Sass para generar estilos CSS de color, tipografía y densidad.

The core mixins

Material Angular define un nombre de combinación core que incluye estilos de requisitos previos para características comunes utilizadas por múltiples componentes, como ondas. El core mixin debe incluirse exactamente una vez para su aplicación, incluso si define varios temas. Incluir el core mixin varias veces dará como resultado un CSS duplicado en su aplicación.

```
@use '@angular/material' as mat;

@include mat.core();
```

Definición de un tema

Angular Material representa un tema como un mapa Sass que contiene sus opciones de color, tipografía y densidad.

La construcción del tema primero requiere definir sus paletas principales y de acento, con una paleta de advertencia opcional. La define-palettefunción Sass acepta una paleta de colores, descrita en la sección Paletas anterior, así como cuatro números de tono opcionales. Estos cuatro tonos representan, en orden: el tono "predeterminado", un tono "más claro", un tono "más oscuro" y un tono de "texto". Los componentes usan estos tonos para elegir el color más apropiado para las diferentes partes de ellos mismos.

```
@use '@angular/material' as mat;

$my-primary: mat.define-palette(mat.$indigo-palette,
500);
$my-accent: mat.define-palette(mat.$pink-palette, A200,
A100, A400);

// The "warn" palette is optional and defaults to red
if not specified.
$my-warn: mat.define-palette(mat.$red-palette);
```

Puede construir un tema llamando define-light-themeo define-dark-themecon el resultado de define-palette. La elección de un tema claro frente a uno oscuro determina los colores de fondo y de primer plano utilizados en todos los componentes.

```
@use '@angular/material' as mat;

$my-primary: mat.define-palette(mat.$indigo-palette,
500);
$my-accent: mat.define-palette(mat.$pink-palette, A200,
A100, A400);

// The "warn" palette is optional and defaults to red
if not specified.
$my-warn: mat.define-palette(mat.$red-palette);

$my-theme: mat.define-light-theme((
 color: (
   primary: $my-primary,
   accent: $my-accent,
   warn: $my-warn,
 ),
 typography: mat.define-typography-config(),
 density: 0,
));
```

Aplicar un tema a los componentes

Sass mixin core-themeemite estilos de requisitos previos para características comunes utilizadas por múltiples componentes, como ondas. Este mixin debe incluirse una vez por tema.

Cada componente de material angular tiene una mezcla para cada color, tipografía y densidad. Por ejemplo, MatButtondeclara button-color, button-typographyy button-density. Cada mixin emite solo los estilos correspondientes a esa zona de personalización.

Además, cada componente tiene un mixin de "tema" que emite todos los estilos que dependen de la configuración del tema. Esta combinación de temas solo emite estilos de color, tipografía o densidad si proporcionó una configuración correspondiente a define-light-themeo define-dark-theme.

Aplique los estilos para cada uno de los componentes utilizados en su aplicación incluyendo cada uno de sus temas Sass mixins.

```
@use '@angular/material' as mat;

@include mat.core();

$my-primary: mat.define-palette(mat.$indigo-palette,
500);
$my-accent: mat.define-palette(mat.$pink-palette, A200,
A100, A400);

$my-theme: mat.define-light-theme((
 color: (
   primary: $my-primary,
   accent: $my-accent,
 ),
 density: 0,
));

// Emit theme-dependent styles for common features used
across multiple components.
@include mat.core-theme($my-theme);

// Emit styles for MatButton based on `$my-theme`.
Because the configuration
// passed to `define-light-theme` omits typography,
`button-theme` will not
// emit any typography styles.
@include mat.button-theme($my-theme);

// Include the theme mixins for other components you
use here.
```

Como alternativa a enumerar todos los componentes que usa su aplicación, Angular Material ofrece mixins de Sass que incluyen estilos para todos los componentes de la biblioteca: all-component-colors, all-component-typographies, all-component-densitiesy all-component-themes. Estos mixins se comportan igual que los mixins de componentes individuales, excepto que emiten estilos para loscore-theme más de 35 componentes en Angular Material. A menos que su aplicación use todos los componentes, esto producirá CSS innecesario.

```
@use '@angular/material' as mat;

@include mat.core();

$my-primary: mat.define-palette(mat.$indigo-palette,
500);
$my-accent: mat.define-palette(mat.$pink-palette, A200,
A100, A400);

$my-theme: mat.define-light-theme((
 color: (
   primary: $my-primary,
   accent: $my-accent,
 ),
 typography: mat.define-typography-config(),
 density: 0,
));

@include mat.all-component-themes($my-theme);
```

Para incluir los estilos emitidos en su aplicación, agregue su archivo de tema a la stylesmatriz del angular.jsonarchivo de su proyecto .

Usar un tema preconstruido

Angular Material incluye cuatro archivos CSS de temas preconstruidos, cada uno con diferentes paletas seleccionadas. Puede usar uno de estos temas preconstruidos si no desea definir un tema personalizado con Sass.

Temática	¿Claro u oscuro?	Paletas (primario, acento, advertir)
`deeppurple-amber.css`	Ligero	morado oscuro, ámbar, rojo
`indigo-pink.css`	Ligero	índigo, rosa, rojo
`pink-bluegrey.css`	Oscuro	rosa, gris azulado, rojo
`purple-green.css`	Oscuro	morado, verde, rojo

Estos archivos incluyen el CSS para cada componente de la biblioteca. Para incluir solo el CSS para un subconjunto de componentes, debe usar la API de Sass que se detalla en Definición de un tema más arriba. Puede hacer referencia al código fuente de estos temas preconstruidos para ver ejemplos de definiciones de temas completas.

Puede encontrar los archivos de temas preconstruidos en el directorio "temas preconstruidos" del paquete npm de Angular Material (@angular/material/prebuilt-themes). Para incluir el tema preconstruido en su aplicación, agregue el archivo CSS elegido a la stylesmatriz del angular.json archivo de su proyecto .

Definición de varios temas

Con la API de Sass descrita en Definición de un tema, también puede definir varios temas repitiendo las llamadas a la API varias veces. Puede hacerlo en el mismo archivo de tema o en archivos de tema separados.

Múltiples temas en un archivo

La definición de varios temas en un solo archivo le permite admitir varios temas sin tener que administrar la carga de varios activos CSS. Sin embargo, la desventaja es que su CSS incluirá más estilos de los necesarios.

Para controlar qué tema se aplica cuando, @includelos mixins solo dentro de un contexto especificado a través de la declaración de la regla CSS.

```
@use '@angular/material' as mat;

@include mat.core();

// Define a dark theme
$dark-theme: mat.define-dark-theme((
 color: (
   primary: mat.define-palette(mat.$pink-palette),
   accent: mat.define-palette(mat.$blue-grey-palette),
 ),
  // Only include `typography` and `density` in the default
dark theme.
  typography: mat.define-typography-config(),
  density: 0,
));

// Define a light theme
$light-theme: mat.define-light-theme((
 color: (
   primary: mat.define-palette(mat.$indigo-palette),
   accent: mat.define-palette(mat.$pink-palette),
 ),
));

// Apply the dark theme by default
@include mat.core-theme($dark-theme);
@include mat.button-theme($dark-theme);

// Apply the light theme only when the user prefers light
themes.
@media (prefers-color-scheme: light) {
 // Use the `-color` mixins to only apply color styles without
reapplying the same
 // typography and density styles.
 @include mat.core-color($light-theme);
 @include mat.button-color($light-theme);
}
```

Múltiples temas en archivos separados

Puede definir varios temas en archivos separados creando varios archivos de tema por Definición de un tema, agregando cada uno de los archivos a stylessu archivo angular.json. Sin embargo, también debe configurar la inject opción para cada uno de estos archivos para false evitar que todos los archivos de temas se carguen al mismo tiempo. Al establecer esta propiedad en false, su aplicación se vuelve responsable de cargar manualmente el archivo deseado. El enfoque para esta carga depende de su aplicación.

Color de fondo de la aplicación

De forma predeterminada, Angular Material no aplica ningún estilo a su DOM fuera de sus propios componentes. Si desea configurar el color de fondo de su aplicación para que coincida con el tema de los componentes, puede:

1. Coloque el contenido principal de su aplicación dentro mat-sidenav-containerde , asumiendo que está usando MatSidenav, o
2. Aplique la mat-app-backgroundclase CSS a su elemento raíz de contenido principal (típicamente body).

Personalización de estilo de ámbito

Puede usar los mixins Sass de Angular Material para personalizar estilos de componentes dentro de un ámbito específico en su aplicación. La declaración de la regla CSS en la que incluye un Sass mixin determina su alcance. El siguiente ejemplo muestra cómo personalizar el color de todos los botones dentro de los elementos marcados con la .my-special-sectionclase CSS.

```
@use '@angular/material' as mat;

.my-special-section {
 $special-primary: mat.define-palette(mat.$orange-palette);
 $special-accent: mat.define-palette(mat.$brown-palette);
 $special-theme: mat.define-dark-theme((
   color: (primary: $special-primary, accent: $special-accent),
 ));

 @include mat.button-color($special-theme);
}
```

Lectura de tonos de paletas

Puede usar la get-color-from-palettefunción para obtener tonos específicos de una paleta por su identificador de número. También puede acceder al color de contraste para un tono en particular agregando el sufijo del identificador del número del tono con -contrast.

```
@use '@angular/material' as mat;

$my-palette: mat.define-palette(mat.$indigo-palette);

.my-custom-style {
 background: mat.get-color-from-palette($my-palette, 500);
 color: mat.get-color-from-palette($my-palette, '500-
contrast');
}
```

También puede hacer referencia a los colores utilizando los colores "default", "lighter", "darker"y "text"pasados a define-palette.

```
@use '@angular/material' as mat;

$my-palette: mat.define-palette(mat.$indigo-palette);

.my-custom-darker-style {
 background: mat.get-color-from-palette($my-palette, 'darker');
 color: mat.get-color-from-palette($my-palette, 'darker-
contrast');
}
```

Personalización de estilo fuera del sistema de tematización

El material angular admite la personalización del color, la tipografía y la densidad como se describe en este documento. Angular desaconseja encarecidamente, y no admite directamente, la anulación del CSS del componente fuera de las API de creación de temas descritas anteriormente. La estructura DOM del componente y las clases CSS se consideran detalles de implementación privados que pueden cambiar en cualquier momento.

Cómo integrar Bootstrap en un proyecto Angular:

Paso 1: Crear proyecto angular.

Primero que nada, crearemos nuestro proyecto y colocaremos “y” para crear el archivo que se encarga de rutear las páginas.

Este archivo es innecesario para esta guía ya que solo veremos el funcionamiento de los estilos ya integrados con Bootstrap.

```
PS C:\Users\franc\Desktop\Proyectos> ng new proyecto-bootstrap
? Would you like to add Angular routing? (y/N)
```

Luego elegimos la hoja de estilos que queremos ocupar y damos “ENTER”.

```
  CSS
> SCSS   [ https://sass-lang.com/documentation/syntax#scss                ]
  Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less   [ http://lesscss.org                                             ]
  Stylus [ http://stylus-lang.com                                         ]
```

Damos “ng serve” y deberíamos poder ver la nueva pantalla de inicio de angular.

Paso 2: Instalar Bootstrap.

Instalamos dependencias necesarias desde el “NPM”, lo primero que tenemos que hacer es ir a nuestra consola y tipear lo siguiente.

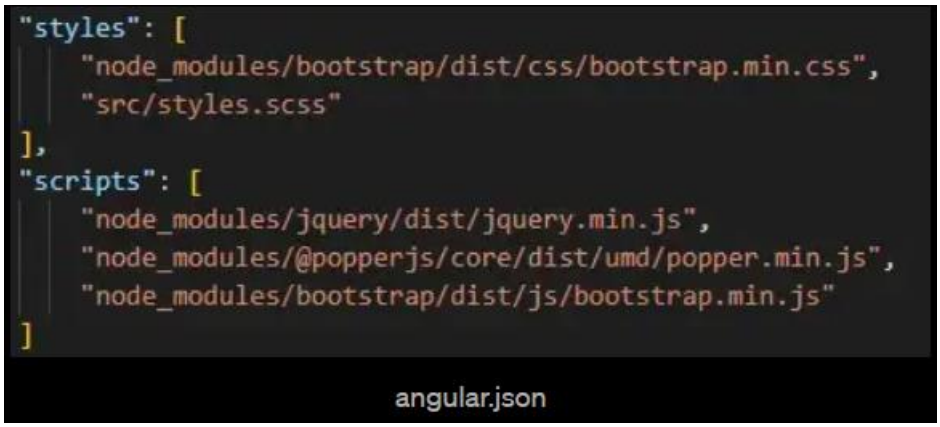
```
npm install bootstrap jquery @popperjs/core
```

Estos comandos instalarán las dependencias del Bootstrap, jquery y el @popperjs/core que son necesarios para darle mejor potencial al funcionamiento del framework.

Luego de eso nos dirigimos al archivo “angular.json” y colocamos las siguientes instrucciones en los objetos “styles” y “scripts” en donde llamaremos a las propiedades css del Bootstrap y las dependencias scripts correspondientes a las interacciones de cada uno.

```
"styles": [
  "node_modules/bootstrap/dist/css/bootstrap.min.css",
  "src/styles.scss"
],
"scripts": [
  "node_modules/jquery/dist/jquery.min.js",
  "node_modules/@popperjs/core/dist/umd/popper.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
]
```

Debería quedar de la siguiente manera:

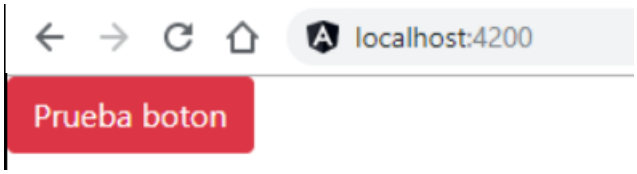


Para hacer la prueba básica y verificar que el Bootstrap este instalado correctamente, nos vamos al archivo “app.componente.html” y eliminaremos todo el contenido que tiene esta hoja y agregaremos el tag button con la clase: “btn btn-danger” lo cual agregará un botón de color rojo con el nombre “Prueba botón”, esto debes hacerlo debajo del <router-outlet></router-outlet>, debería quedar de la siguiente forma:

```
<router-outlet></router-outlet>

<button class="btn btn-danger">Prueba botón</button>
```

Finalmente detenemos nuestro servidor y volveremos a iniciarlo con un “ng serve”, el resultado que deberíamos ver sería el siguiente:



Con esto tenemos creado nuestro proyecto angular con Bootstrap integrado.

Bibliografía:

https://material.angular.io/guide/getting-started

https://fbellod.medium.com/como-integrar-el-framework-bootstrap-en-un-proyecto-angular-a5d53fa79e03

https://material.angular.io/components/categories

https://www.tutorialesprogramacionya.com/angularya/

https://material.angular.io/guide/schematics