9 - Material : Material Design y CLI de Angular

El *ecosistema* de Angular está repleto de librerías para desarrolladores profesionales. Algunas hacen uso de los **schematics**, y entre ellas destaca Angular Material. Esta implementación de la casa de la guía de diseño *Material Design* de Google usa las capacidades de estas plantillas del CLI que permiten agregar librerías y generar código.

Un programador Angular debe **dominar el CLI** y debe conocer los beneficios que aporta un repositorio de multi-proyecto. Hay escenarios complejos muy adecuaos para estos *mono-repos*. Pero con el CLI es muy sencillo crear y usar nuevas aplicaciones dentro de un repositorio.

Partiendo de la aplicación tal cómo quedó en *Formularios reactivos con Angular*. Al finalizar tendrás, en el mismo repositorio, una nueva aplicación con la apariencia y usabilidad de *Material design*.

> Tienes una versión desplegada operativa para probar Angular Board

1. Repositorio multi-proyecto

El primer comando que se usa al empezar con angular es `ng new mi-aplicacion`. Desde ese momento tu mundo es la carpeta `/src` en la que se genera el código y en la que vas a desarrollar.

Pero con el tiempo, ciertos proyectos crecen y hay que dividirlos. O quizás surjan aplicaciones hermanas. Angular CLI permite disponer de más de un proyecto **compartiendo repositorio y configuración**.

1.1 Carpetas src y projects

Dado un repositorio inicial, para agregar una nueva aplicación usaremos el viejo comando *generate*. Por ejemplo voy a crear una aplicación en la que usar las capacidades de los **schematics** y de **material**; la llamaré *schemat*

```
ng g application schemat --routing
```

Esta aplicación comparte la configuración básica de `angular.json` y las dependencias y scripts de `package.json`. Su código específico va en la carpeta `projects` destinada a los nuevos proyectos generados tras haber creado el repo inicial.

1.2 Compilación multi - proyecto

A partir de ahora cada comando del CLI debería ir asociado a un proyecto concreto. Digo debería porque an `angular.json` puedes establecer un proyecto por defecto, que si no dices lo contrario será el inicial.

Pero, es buena práctica crear scripts específicos en el `package.json` para iniciar y compilar cada proyecto.

```
"start:schemat": "ng serve schemat --aot -o --port 4271",
"build:prod:schemat": "ng build schemat --prod",
```

2. Instalación y configuración de Material

Por muchas funcionalidades que aporte un *framework* como Angular, siempre necesitaremos echar mano de alguna **librería de terceros**. Normalmente eso implica instalarla con *npm*, importar sus módulos en Angular y en ocasiones alguna configuración extra.

Pero algunos proyectos ha adoptado la librería *schematics* para facilitar la adopción de sus librerías. Es el caso de **Angular Material**.

2.1 Agregar dependencias con schematics

Para agregar *Material* un proyecto basta con usar el comando `add` del CLI.

```
ng add @angular/material --project=schemat
```

Esto instalará y anotará la dependencia de `@angular/material` y otros paquetes necesarios. Pero además los registrará en `AppModule` y lo configurará.

2.2 Estilos, iconos y temas básicos

En el `index.html` se insertarán los enlaces a las hojas de estilos con fuentes e iconos. En el `styles.css` podremos importar el tema de *Material* que nos guste.

```
@import '~@angular/material/prebuilt-themes/indigo-pink.css';
```

3. Componentes básicos

Angular Material es mucho más que un *css*. El proyecto te ofrece más de una docena de componentes visuales para crear páginas web de aspecto y comportamiento profesional.

Por si fuera poco, ofrece sus propias plantillas de *schematics*, de forma que podemos crear pantallas con el CLI y luego trabajar sobre ellas.

3.1 Navegación y layout

Para empezar vamos a crear el armazón de la aplicación, con su menú y su espacio para cargar contenido. Después pondremos la primera piedra con un dashboard en la raíz de la web.

3.1.1 Navegación

Al control que hará de *shell* le llaman `nav` por sus capacidades de navegación. Lo usaremos para generar un componente. La novedad será que la plantilla a partir de la cual se crea no viene por defecto y tenemos que especificarla con más detalle.

```
ng g @angular/material:nav shell --project=schemat
```

EL resultado es un componente normal, muy adecuado para sustituir todo lo pre-generado por el cli en`app.component.html`

```
<app-shell></app-shell>
```

3.1.2 Dashboard

Para la ruta raíz elegí un cuadro de mando. De nuevo es una generación a partir de una plantilla específica. En este caso le he llamado `Home`.

```
ng g @angular/material:dashboard home --project=schemat
```

Asociamos el componente a la ruta en `app-routing`

```
const routes: Routes = [
  {
    path: '',
    component: HomeComponent
  }
];
```

Y le buscamos un lugar tanto al enlace como al `RouterOutlet` en el previamente generado `shell.component.html`.

```
<a mat-list-item [routerLink]="['/']">Home</a>
<!-- Add Content Here -->
<router-outlet></router-outlet>
```

3.2 Componentes básicos

Por supuesto que una librería de esta envergadura tienen soluciones para todo tipo de situaciones visuales. Para las más comunes incluso ofrecen una plantilla en sus *schematics* que permite generar prototipos funcionales de formularios, tablas y hasta árboles.

3.2.1 Formularios

Puedes generar un **formulario** para crear contactos y después modificarlo o usarlo como guía para crear cualquier otro.

```
ng g @angular/material:address-form contact --project=schemat
```

3.2.2 Tablas

También nos muestran cómo hacer **listados**, con datos de ejemplo incluidos.

```
ng g @angular/material:table elements --project=schemat
```

El ejemplo permite además ver los datos en páginas y configurar su comportamiento.

```
<mat-paginator #paginator
    [length]="dataSource.data.length"
    [pageIndex]="0"
    [pageSize]="5"
    [pageSizeOptions]="[5, 10, 15, 20]">
</mat-paginator>
```

3.2.3 Árboles

Termino con un ejemplo de algo reciente, **el árbol**. Este componente se resistió pero por petición popular acabaron integrándolo y ahora puedes mostrar datos jerárquicos como más les gusta a los usuarios: en forma de árboles.

```
ng g @angular/material:tree source --project=schemat
```

Por supuesto que hay más opciones, todas bien explicadas en la documentación de Angular Material. Es una solución muy recomendable para aplicaciones de gestión, o simplemente para que se vean muy bien en Android y recuerden al *look and feel* de todo lo que hace Google.

Con este conocimiento finalizas *tu introducción a Angular*. Espero que te haya sido útil y confío en que aprenderás más cosas para programar bien con Angular 7.