Módulo 1 - Lectura: Introducción al uso de componentes

Un componente controla una porción de pantalla llamada vista. Por ejemplo, los componentes individuales definen y controlan cada una de las siguientes vistas del Tutorial que realizamos con la plantilla Side Drawer:

- La raíz de la aplicación con los enlaces de navegación.
- El panel de navegación lateral.
- Las pantallas de detalle de cada sección.

Define la lógica de la aplicación de un componente (o sea, lo que hace para admitir la vista) dentro de una clase. La clase interactúa con la vista a través de una API de propiedades y métodos. De esta manera se comunican la plantilla .html (que en el caso de Nativescript, el código que contiene no es HTML sino XML) y el código Typescript.

Angular crea, actualiza y destruye componentes a medida que el usuario se mueve a través de la aplicación. Su aplicación puede actuar en cada momento de este ciclo de vida a través de eventos o "hooks" de ciclo de vida opcionales, como ngOnInit ().

Metadata de Componentes:
El decorador @Component identifica la clase inmediatamente debajo de ella como una clase que será un componente de la aplicación y especifica sus metadatos. Es decir que una clase, sin ninguna anotación o sintaxis de Angular especial, no será un componente hasta que lo marques como tal con el decorador @Component.

Los metadatos de un componente le indican a Angular dónde obtener los principales bloques de construcción que necesita para crear y presentar el componente y su vista. En particular, asocia una plantilla con el componente, ya sea directamente con el código en línea o por referencia. Juntos, el componente y su plantilla, describen una vista.

Además de contener o apuntar a la plantilla, los metadatos de @Component configuran, por ejemplo, cómo se puede hacer referencia al componente en HTML y qué servicios requiere.

Por ejemplo, algunas de las opciones de configuración de @Component más útiles son:

1. **selector**: un selector CSS que le dice a Angular que cree e inserte una instancia de este componente donde encuentre la etiqueta correspondiente en la plantilla HTML. Por ejemplo, si el HTML de una aplicación contiene <Listado> </Listado>, Angular inserta una instancia de la vista ListadoComponent entre esas etiquetas.

2. **templateUrl**: la dirección relativa al módulo de la plantilla HTML de este componente. Alternativamente, puede proporcionar tanto la plantilla HTML en línea como el valor de la propiedad de la plantilla. Esta plantilla define la vista de host del componente.

3. **providers**: una variedad de proveedores de servicios que requiere el componente. Esto permite una integración directa a la inyección de dependencias de Angular que más adelante estudiaremos en detalle.

Plantillas y Vistas

Se define la vista de un componente con su plantilla complementaria. Una plantilla es una forma de HTML que le dice a Angular cómo representar el componente.

Las vistas suelen estar ordenadas jerárquicamente, lo que le permite modificar o mostrar y ocultar secciones o páginas completas de la interfaz como una unidad. La plantilla inmediatamente asociada con un componente define la vista de host de ese componente. El componente también puede definir una jerarquía de vistas, que contiene vistas incrustadas, alojadas por otros componentes.

Una jerarquía de vistas puede incluir vistas de componentes en el mismo NgModule, pero también puede (y suele) incluir vistas de componentes que se definen en diferentes NgModules.
Esto es particularmente útil, por ejemplo, cuando usamos librerías de terceros, como de hecho lo son todos los widgets definidos por Nativescript o, por ejemplo, si queremos usar mapas de Google o MapBox.