Módulo 1 - Lectura: Proyectos con Code Sharing

Desde el comienzo de Angular (e incluso desde Angular 2), puedes usar Nativescript con Angular para crear aplicaciones móviles. Sin embargo, cuando necesitabas crear una aplicación web y una aplicación móvil nativa, tenías que crear dos proyectos separados. Esto ha cambiado gracias a la creación de los Schematics y ng add.

Los equipos de Angular y Nativescript se unieron para crear esquemas nativos, una colección de esquemas que te permite construir aplicaciones web y móviles desde un solo proyecto. El paquete @nativescript/schematics solo funciona con @angular/cli: 6.1.0 o posterior, tal como lo usamos en este curso.

Compartir código entre proyectos
Un proyecto de código compartido es uno donde guardamos el código para la web y las aplicaciones móviles en un solo lugar. Analicemos como sería a un alto nivel. El objetivo es compartir la mayor cantidad de código posible y dividir el código específico de la plataforma en archivos separados.

Esto generalmente significa que podemos compartir el código para:

1. Rutas para la navegación.
2. Servicios para la lógica común de negocio.
3. Definición de clase de componente para el comportamiento común de un componente, es decir, el código Typescript de nuestro componente.

Mientras que deberemos dividir el código para:

1. La capa gráfica (por un lado, CSS y HTML de web y, por otro lado, el HTML - XML - y CSS para Mobile nativo). Esto es así ya que se necesita usar diferentes componentes de interfaz de usuario en aplicaciones nativas web y compuestas por Nativescript.
2. NgModules: para que puedas importar módulos específicos de la plataforma, sin crear conflictos entre la web y el móvil (por ejemplo, Angular Material Design, que es solo web, o bien, NativeScriptModule que es solo mobile).

La convención de nombres de archivo, para que Angular sepa cuál usar, consiste en que, si tenemos por ejemplo un archivo estilos.css, se use para la versión web y, si queremos crear su contraparte mobile, creemos un archivo llamado estilos.tns.css, para indicar que dicho archivo se use solo para la versión de Nativescript.

Esquemas - Una Introducción: Schematics
Schematics es una herramienta de flujo de trabajo para la web moderna, ya que te permite aplicar transformaciones a tu proyecto, tales como crear un nuevo componente o actualizar el código para

corregir cambios en una dependencia. Y también para agregar una nueva opción de configuración o framework a un proyecto existente.

La misión de la CLI Angular es mejorar la productividad de tu desarrollo y por eso llegamos a un punto en el que se requería una instalación más potente y genérica para admitir los andamios (scaffolding) de CLI, y se basa en los siguientes objetivos principales:

1. Facilidad de uso y desarrollo.
2. Extensibilidad y reutilización. Los esquemas se pueden agregar como entrada, o la salida de otros esquemas. Por ejemplo, una aplicación se puede crear utilizando componentes y esquemas de módulos.
3. Atomicidad. Todos los cambios se registran en la memoria y solo se aplican una vez que se confirma que son válidos. Por ejemplo, crear un archivo que ya existe es un error y descartaría todos los demás cambios aplicados hasta el momento.
4. Asincronicidad. Muchos flujos de trabajo son de naturaleza asíncrona (por ejemplo, para acceder a servidores web), por lo que Schematics tuvo que soportar esos casos de uso. De esta manera, los desarrolladores pueden reutilizar sin siquiera saber que un esquema es asíncrono.

Todas las decisiones de diseño de Schematics se desarrollaron en torno a estos 4 objetivos principales. Los esquemas son los esfuerzos combinados para construir una mejor herramienta de flujo de trabajo.

En un esquema, no se realiza ninguna acción directa en el sistema de archivos. Más bien, se describe qué transformación le gustaría aplicar a un árbol de archivos. También hace que los esquemas sean herméticos, lo que garantiza la reutilización y la seguridad.

El Árbol es una estructura de datos que contiene una base (un conjunto de archivos que ya existe) y un área de preparación (una lista de cambios que se aplicarán a la base). Al realizar modificaciones, realmente no cambia la base, sino que agrega esas modificaciones al área de preparación. Esto es realmente poderoso, pero puede ser complicado.

La cli angular toma el árbol original de proyecto, el cual es tu punto to de partida, y luego aplica los esquemas iterativamente uno tras otro en una copia del proyecto, y los mismos alteran dicho árbol auxiliar, sin alterar el árbol de archivos original.

En el caso concreto de Nativescript y Angular Web, en el Árbol inicial existen todos los archivos de ambas plataformas, por ejemplo:

1. Estilos.css y Estilos.tns.css
2. MiComponente.html y MiComponente.tns.html
3. un-modulo.module.ts y un-module.module.tns.ts

Y al aplicarse el schematics (que Angular web y Nativescript ya tienen configurados por defecto), lo que sucede concretamente es que el Esquema solo pasa al área o Árbol de preparación de los

archivos que correspondan, por ejemplo, si estamos usando el comando tns run android, sucederá lo siguiente:

1. El esquema selecciona los archivos *.tns.* si es que existen y si no, selecciona la implementación por defecto.
    a. Por ejemplo, si tenemos estilos.css y estilos.tns.css, elegirá estilos.tns.css y lo pasará al área de preparación como estilos.css.
    b. Por otro lado, si tenemos otro archivo con una sola implementación, por ejemplo, otro-estilo.css, entonces copiará ese archivo conservando su nombre.
2. Luego, una vez que el área de preparación ha sido llenada con todos los archivos, se prosigue a realizar la compilación estándar de Angular.

Para más información puede remitirse a la documentación oficial en:
https://docs.nativescript.org/code-sharing/intro#introduction