Estructura general dentro de un proyecto de .NET MAUI

El único proyecto de interfaz de usuario de aplicaciones multiplataforma de .NET (.NET MAUI) toma las experiencias de desarrollo específicas de la plataforma que normalmente se encuentran al desarrollar aplicaciones y las abstrae en un único proyecto compartido que pueda tener como destino Android, iOS, macOS y Windows.

Un único proyecto de .NET MAUI proporciona una experiencia de desarrollo multiplataforma simplificada y coherente, independientemente de las plataformas destinadas. El proyecto único de .NET MAUI proporciona las siguientes características:

- Un solo proyecto compartido que pueda tener como destino Android, iOS, macOS y Windows.
- Selección simplificada de destino de depuración para ejecutar las aplicaciones .NET MAUI.
- Archivos de recursos compartidos dentro del único proyecto.
- Un único manifiesto de aplicación que especifica el título, el identificador y la versión de la aplicación.
- Acceso a las API y herramientas específicas de la plataforma cuando sea necesario.
- Un único punto de entrada de aplicación multiplataforma.

El proyecto único de .NET MAUI está habilitado con varios destinos y el uso de proyectos de estilo SDK en .NET 6.

Archivos de recursos

La administración de recursos para el desarrollo de aplicaciones multiplataforma ha sido tradicionalmente problemática. Cada plataforma tiene su propio enfoque para administrar recursos, que se deben implementar en cada plataforma. Por ejemplo, cada plataforma tiene requisitos de imagen diferentes que normalmente implican la creación de varias versiones de cada imagen en diferentes resoluciones. Por lo tanto, una sola imagen normalmente tiene que duplicarse varias veces por plataforma, en diferentes resoluciones, con las imágenes resultantes que tienen que usar diferentes convenciones de nombre de archivo y carpeta en cada plataforma.

El único proyecto MAUI de .NET permite almacenar archivos de recursos en una sola ubicación mientras se consumen en cada plataforma. Esto incluye fuentes, imágenes, el icono de la aplicación, la pantalla de presentación, los recursos sin procesar y los archivos CSS para aplicar estilos a aplicaciones .NET MAUI.

Cada archivo de recursos de imagen se usa como imagen de origen, a partir de la cual se generan imágenes de las resoluciones necesarias para cada plataforma en tiempo de compilación.

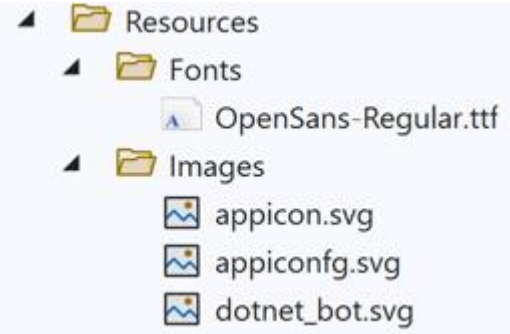
Normalmente, los archivos de recursos deben colocarse en la carpeta Resources del proyecto de aplicación MAUI de .NET o en las carpetas secundarias de la carpeta Resources y deben tener establecida correctamente su acción de compilación. En la tabla siguiente se muestran las acciones de compilación para cada tipo de archivo de recurso:

Recurso	**Acción de compilación**
Icono de la app	MauiIcon
Fuentes	MauiFont
Imágenes	MauiImage
Pantalla de presentación	MauiSplashScreen
Recursos sin procesar	MauiAsset
Archivos CSS	MauiCss

Los archivos XAML también se almacenan en el proyecto de aplicación MAUI de .NET y se asignan automáticamente la acción de compilación MauiXaml cuando se crean mediante plantillas de proyecto y elemento. Sin embargo, los archivos XAML no normalmente se ubicarán en la carpeta Resources del proyecto de aplicación.

Cuando se agrega un archivo de recursos a un proyecto de aplicación MAUI de .NET, se crea una entrada correspondiente para el recurso en el archivo del proyecto (.csproj), a excepción de los archivos CSS. Después de agregar un archivo de recursos, su acción de compilación se puede establecer en la ventana Propiedades.

En la siguiente captura de pantalla se muestra una carpeta **Resources** que contiene recursos de imagen y fuente en carpetas secundarias:



Las carpetas secundarias de la carpeta *Resources* se pueden designar para cada tipo de recurso editando el archivo de proyecto de la aplicación:

```
<ItemGroup>
    <!-- Images -->
    <MauiImage Include="Resources\Images\*" />

    <!-- Fonts -->
    <MauiFont Include="Resources\Fonts\*" />

    <!-- Assets -->
    <MauiAsset Include="Resources\Assets\*" />
</ItemGroup>
```

El carácter comodín (*****) indica que todos los archivos de la carpeta se tratarán como del tipo de recurso especificado. Además, es posible incluir todos los archivos de carpetas secundarias:

```
<ItemGroup>
    <!-- Images -->
    <MauiImage Include="Resources\Images\**\*" />
</ItemGroup>
```

En este ejemplo, el carácter comodín doble ('**') especifica que la carpeta Images puede contener carpetas secundarias. Por lo tanto, <MauiImage Include="Resources\Images***" /> especifica que los archivos de la carpeta Resources\Images , o cualquier carpeta secundaria de la carpeta Images , se usarán como imágenes de origen de las que se generan imágenes de la resolución necesaria para cada plataforma.

Los recursos específicos de la plataforma invalidarán sus homólogos de recursos compartidos. Por ejemplo, si tiene una imagen específica de Android ubicada en Platforms\Android\Resources\drawable-xhdpi\logo.pngy también proporciona una imagen compartida Resources\Images\logo.svg , el archivo De gráficos vectoriales escalables (SVG) se usará para generar las imágenes de Android necesarias, excepto para la imagen XHDPI que ya existe como una imagen específica de la plataforma.

Iconos de aplicación

Se puede agregar un icono de aplicación al proyecto de aplicación arrastrando una imagen a la carpeta Resources\Images del proyecto y estableciendo la acción de compilación del icono en MauiIcon en la ventana Propiedades. Esto crea una entrada correspondiente en el archivo del proyecto:

```
<MauiIcon Include="Resources\Images\appicon.png" />
```

En tiempo de compilación, el icono de la aplicación se cambia de tamaño a los tamaños correctos de la plataforma y el dispositivo de destino. Los iconos de la aplicación cuyo tamaño se ha cambiado se agregan al paquete de la aplicación. Los iconos de la aplicación se cambian de tamaño a varias resoluciones porque tienen varios usos, incluidos los que se usan para representar la aplicación en el dispositivo y en la tienda de aplicaciones.

Imágenes

Las imágenes se pueden agregar al proyecto de aplicación arrastrándolas a la carpeta Resources\Images del proyecto y estableciendo su acción de compilación en MauiImage en la ventana Propiedades . Esto crea una entrada correspondiente por imagen en el archivo del proyecto:

```
<MauiImage Include="Resources\Images\logo.jpg" />
```

En tiempo de compilación, las imágenes se cambian de tamaño a las resoluciones correctas para la plataforma y el dispositivo de destino. A continuación, las imágenes con tamaño se agregan al paquete de la aplicación.

Fuentes

Las fuentes de formato de tipo true (TTF) y fuente de tipo abierto (OTF) se pueden agregar al proyecto de aplicación arrastrándolas a la carpeta Resources\Fonts del proyecto y estableciendo su acción de compilación en MauiFont en la ventana Propiedades . Esto crea una entrada correspondiente por fuente en el archivo del proyecto:

```
<MauiFont Include="Resources\Fonts\OpenSans-Regular.ttf" />
```

En tiempo de compilación, las fuentes se copian en el paquete de la aplicación.

Pantalla de presentación

Se puede agregar una pantalla de barra diagonal al proyecto de aplicación arrastrando una imagen a la carpeta Resources\Images del proyecto y estableciendo la acción de compilación de la imagen en MauiSplashScreen en la ventana Propiedades . Esto crea una entrada correspondiente en el archivo del proyecto:

```
<MauiSplashScreen Include="Resources\Images\splashscreen.svg" />
```

En tiempo de compilación, la imagen de la pantalla de presentación se cambia de tamaño al tamaño correcto para la plataforma y el dispositivo de destino. La pantalla de presentación cuyo tamaño se ha cambiado se agrega al paquete de la aplicación.

Recursos sin procesar

Los archivos de recursos sin procesar, como HTML, JSON y vídeos, se pueden agregar al proyecto de aplicación arrastrándolos a la carpeta Resources del proyecto (o una subcarpeta, como Resources\Raw) y estableciendo su acción MauiAsset de compilación en en la ventana Propiedades . Esto crea una entrada correspondiente por recurso en el archivo del proyecto:

```
<MauiAsset Include="Resources\Raw\index.html" />
```

Los controles pueden consumir activos sin procesar, según sea necesario:

```
<WebView Source="index.html" />
```

En tiempo de compilación, los recursos sin procesar se copian en el paquete de la aplicación.

Archivos CSS

Las aplicaciones .NET MAUI pueden tener un estilo parcial con archivos de hoja de estilos en cascada (CSS). Los archivos CSS se pueden agregar al proyecto de aplicación arrastrándolos a cualquier carpeta del proyecto y estableciendo su acción de compilación MauiCss en en la ventana Propiedades .

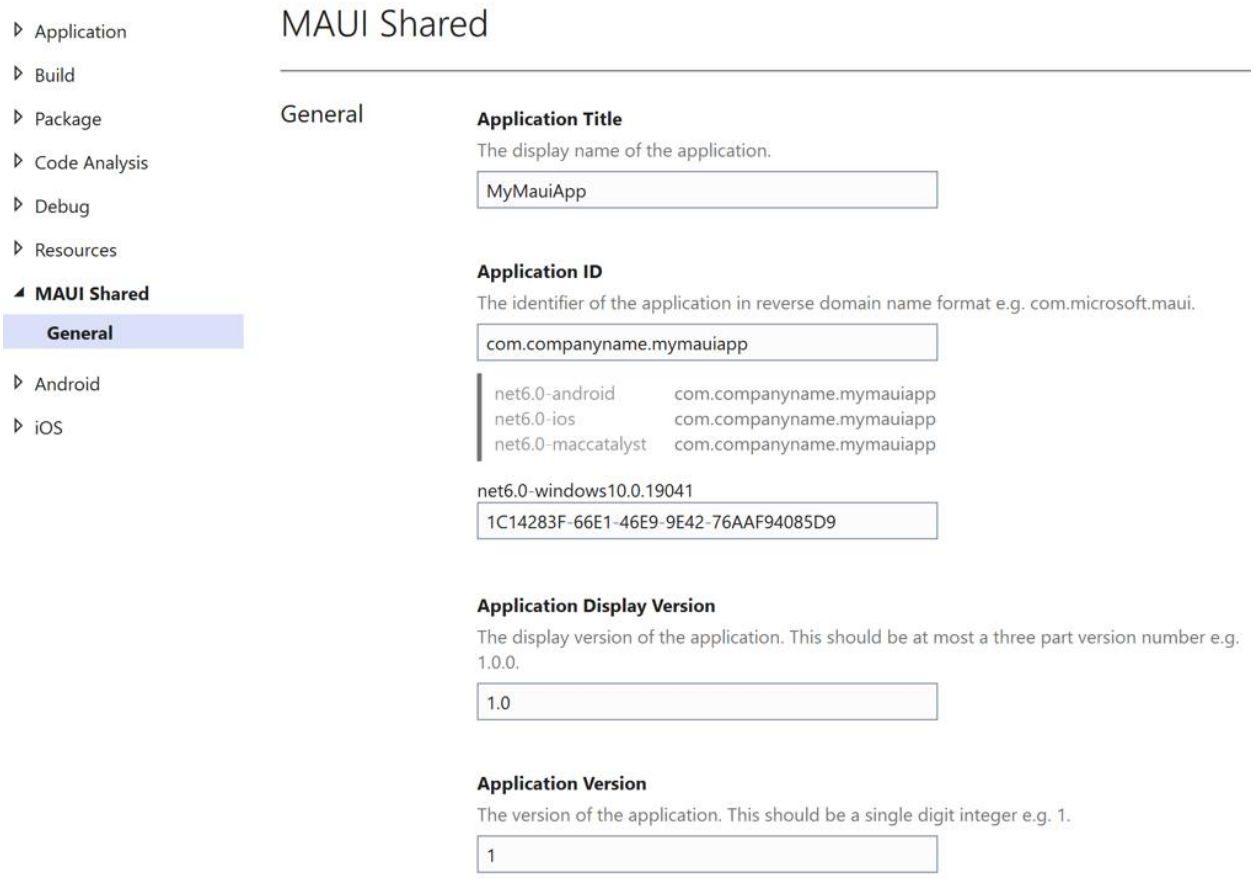
La clase debe cargar StyleSheet los archivos CSS antes de agregarlos a :ResourceDictionary

```
<Application ...>
    <Application.Resources>
        <StyleSheet Source="/Resources/styles.css" />
    </Application.Resources>
</Application>
```

Manifiesto de aplicación

Cada plataforma usa su propio archivo de manifiesto de aplicación nativa para especificar información como el título de la aplicación, el identificador, la versión, etc. El proyecto único de .NET MAUI permite especificar estos datos comunes de la aplicación en una sola ubicación en el archivo del proyecto (.csproj).

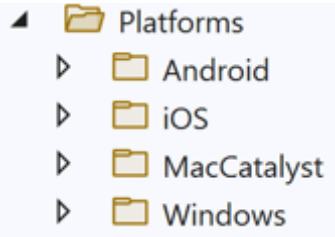
Para especificar los datos del manifiesto de la aplicación compartida para un proyecto, abra el menú contextual del proyecto en Explorador de soluciones y, a continuación, elija Propiedades. El título, el identificador y la versión de la aplicación se pueden especificar en MAUI Shared > General:



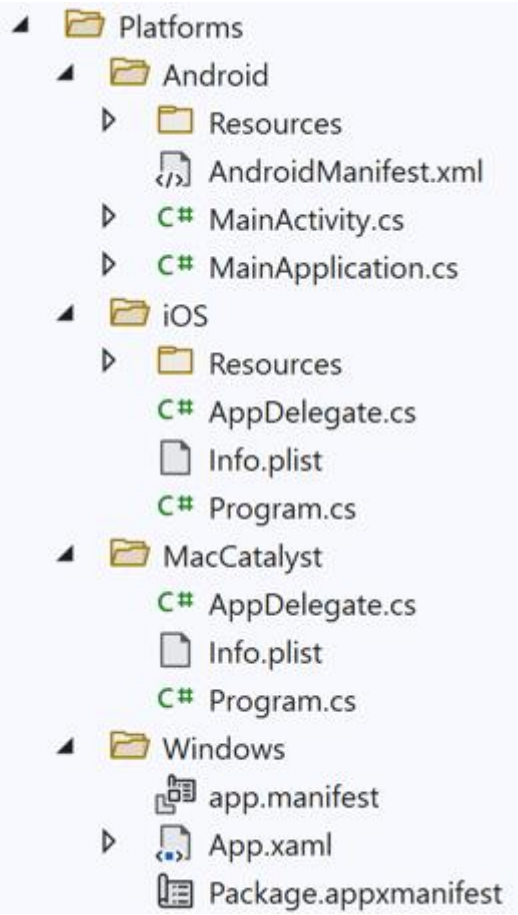
En tiempo de compilación, los datos del manifiesto de la aplicación compartida se combinan con datos específicos de la plataforma en el archivo de manifiesto de aplicación nativa, para generar el archivo de manifiesto que se incluye con la aplicación.

Código específico de la plataforma

Un proyecto de aplicación MAUI de .NET contiene una carpeta Platforms , con cada carpeta secundaria que representa una plataforma que .NET MAUI puede tener como destino:



Las carpetas de cada plataforma contienen recursos específicos de la plataforma y código y que inician la aplicación en cada plataforma:



En tiempo de compilación, el sistema de compilación solo incluye el código de cada carpeta al compilar para esa plataforma específica. Por ejemplo, al compilar para Android, los archivos de la carpeta Plataformas\Android se integrarán en el paquete de la aplicación, pero los archivos de las demás carpetas Plataformas no serán.

Este enfoque usa varios destinos para tener como destino varias plataformas desde un único proyecto. El destino múltiple se puede combinar con clases y métodos parciales para invocar la funcionalidad de plataforma nativa desde código multiplataforma.

Además de este enfoque predeterminado de múltiples destinos, las aplicaciones de .NET MAUI también pueden tener varios destinos en función de sus propios criterios de nombre de archivo y carpeta. Esto le permite estructurar el proyecto de aplicación MAUI de .NET para que no tenga que colocar el código de la plataforma en subcarpetas de la carpeta *Plataformas*.

Los múltiples destinos también se pueden combinar con la compilación condicional para que el código esté destinado a plataformas específicas:

```
#if ANDROID
handler.NativeView.SetBackgroundColor(Colors.Red.ToNative());
#elif IOS
                    handler.NativeView.BackgroundColor = Colors.Red.ToNative();
                    handler.NativeView.BorderStyle = UIKit.UITextBorderStyle.Line;
#elif WINDOWS
                    handler.NativeView.Background = Colors.Red.ToNative();
#endif
```

Punto de entrada de la aplicación

Aunque las carpetas Plataformas contienen código específico de la plataforma que inicia la aplicación en cada plataforma, las aplicaciones MAUI de .NET tienen un único punto de entrada de aplicación multiplataforma. Cada punto de entrada de la plataforma llama a un CreateMauiApp método en la clase estática MauiProgram del proyecto de aplicación y devuelve un MauiApp, que es el punto de entrada de la aplicación. El CreateMauiApp método arranca la aplicación mediante el host genérico de .NET. Esto proporciona la capacidad de configurar la aplicación, los servicios y las bibliotecas de terceros desde una sola ubicación.