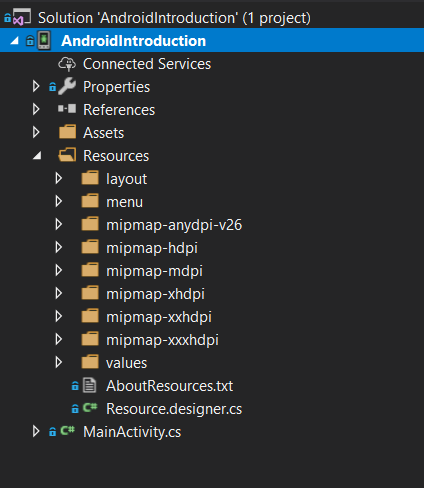
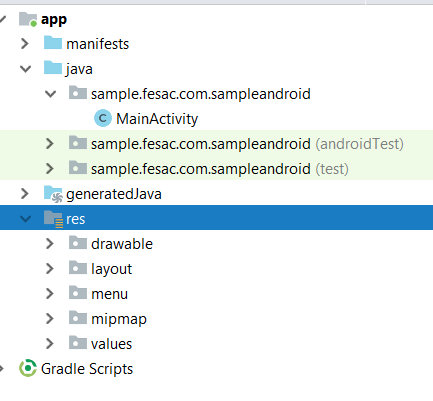
# Introducción

Durante estás páginas vamos a ver las diferentes particularidades de Android y a su vez de Xamarin Android, para empezar a desarrollar. Teniendo en cuenta estos apartados podrás comenzar a desarrollar aplicaciones en Android o Xamarin Android. Ya sea con C#, Kotlin o Java.

# Estructura de proyecto



Estructura de proyecto en Visual Studio (Xamarin Android)



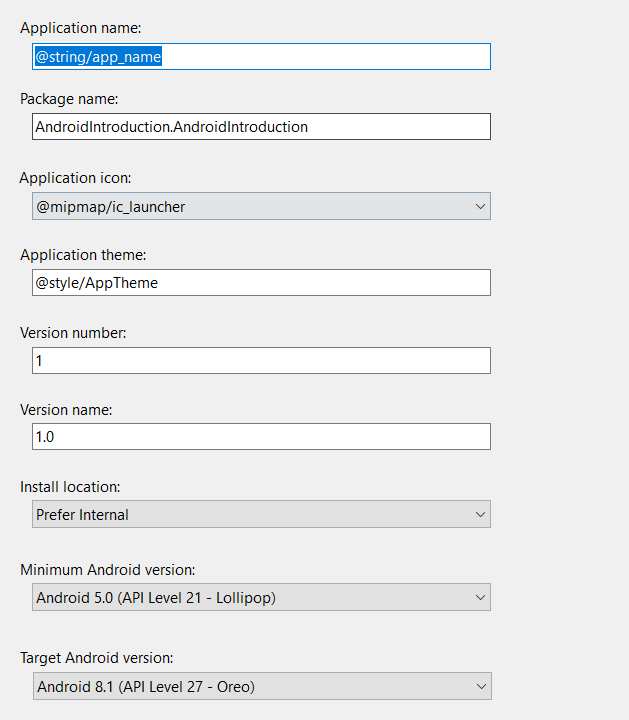
Estructura de proyecto Android Studio (Android)

Como podemos comprobar las imágenes, la estructura de los proyectos son exactamente iguales.

El apartado más importante aquí es la carpeta Resources, común para Android en ella vamos a almacenar las diferentes imágenes de recursos. (mipmap / drawable) Las diferentes vistas que tenemos asociadas (Layout) Además de ficheros de estilos y recursos de String dentro de la carpeta Values.

# Android Manifest

Este archivo es un XML que incluye toda la información del binario, permisos, las actividades implicadas dentro del proyecto, el nombre del paquete, nombre de versión… etc.

Aunque podemos modificarlo únicamente cambiando el xml, también existen diferentes editores para poder hacerlo de manera amigable

Los aspectos más conocidos que podemos modificar dentro del manifiesto son:

* Application Name
  + El nombre de aplicación que lo usuarios verán
* Package name
  + El nombre del paquete con el que se identificará dentro de nuestro dispositivo y la PlayStore. La estructura común es invertir la url relacionada con el paquete. Por lo que es común empezar con .com y .es
* Application icon
  + La ubicación por recursos del icono de aplicaciones.
* Application theme
  + El tema es el estilo general que tendrá toda la aplicación también ubicada en los recursos
* Version number
  + El número de versión identificable internamente por la Play store. Para poder actualizar dentro de la store, debe ser superior a la versión anterior.
* Version name
  + El código de versión identificable por los usuarios
* Install location
  + Permite especificar las ubicaciones donde la aplicación puede ser instalada.

## Android versión

Dentro del manifiesto tenemos tres valores de versiones: Minimum, Target, Compile. Mediante el valor de la versión mínima será la versión mínima de API de Android que permitirá ser instalado. <https://developer.android.com/about/dashboards/>

## Permissions

Dentro de nuestra aplicación no podemos usar distintas funcionalidades del dispositivo al menos que integremos los diferentes permisos que requiramos. Entre los más comunes son:

* Access Fine Location y Access coarse location
  + Permiten dar el posicionamiento dentro de nuestra aplicación.
* Camera
  + Nos da acceso a la cámara
* Internet
  + Nos permite dar la conexión de internet.
* Read external storage y Write external storage
  + Tenemos acceso al espacio externo del dispositivo y las carpetas compartidas, documentos, fotos, videos, etc.

Aunque existen muchísimos permisos más para controlar los diferentes aspectos del dispositivo. Es cierto que siempre que probemos una funcionalidad del dispositivo, si no tenemos el permiso recibiremos una excepción que nos declarará que permiso debemos adaptar.

Además desde la versión 6.0 de la API hay ciertos permisos que se deben pedir implícitamente al usuario, que veremos en un ejemplo de código.

# API de Android

La manera que funciona Xamarin Android es realizando un wrapper hacia .NET de la API de Android desarrollada en Java, por lo que absolutamente todo el espacio de nombres de Android tiene cada una de las APIs que podríamos hacer uso en un proyecto convencional de Android nativo con Java o Kotlin.

Es por ello que en ciertas partes del código nos podemos encontrar que recuperamos o pasamos objetos provenientes de la API de Android e inclusive de Java o que aunque tengamos llamadas más reconocidas para el desarrollo en .NET, sigamos manteniendo las llamadas propias de Java. Como puede tratarse de los eventos.

# 

# Activity

Es la pieza mínima que implementa la interfaz de usuario visual de la aplicación, para una aplicación es necesario al menos tener una actividad en cada aplicación que además tendrá la propiedad *MainLauncher* para ser esta la actividad principal de la aplicación.

Para asociar visualmente una actividad necesitamos asociarlo a un Layout localizado dentro de la carpeta de recursos, usando el método SetContentView. Para hacer el uso de los elementos dentro del Layout deberemos usar el atributo @+id/nombreIdentificador para poder consumirlo desde la actividad, tenemos un ejemplo de código.

# Intent

Mediante este tipo de clase el mensaje que nos permite comunicar diferentes actividades, no solo diferentes actividades dentro de nuestra propia aplicación, si no entre diferentes actividades de otras aplicaciones. Mediante el IntentFilter podremos indicar que modalidades de comunicación permite la aplicación, algunos ejemplos son: poder recibir imágenes, texto, recibir notificaciones push, etc.

Además el Intent, nos permite pasar información entre diferentes actividades de nuestra propia aplicación. Para la navegación, dentro del código de ejemplo veremos un cambio de navegación.

# Fragment

Los fragmentos son introducidos en la API 3.0, estos pueden ser considerados “subactividades” un fragmento se encuentra integrado dentro de una de las actividades que tengamos dentro de nuestra aplicación.

Las actividades nos permiten una fácil reutilización de las mismas, nos permiten tener dentro de una actividad un multipanel de funcionalidad, como ocurre en la navegación por pestañas o menú de vista/detalle (menú hamburguesa) además de poder realizar apilaciones de las mismas.

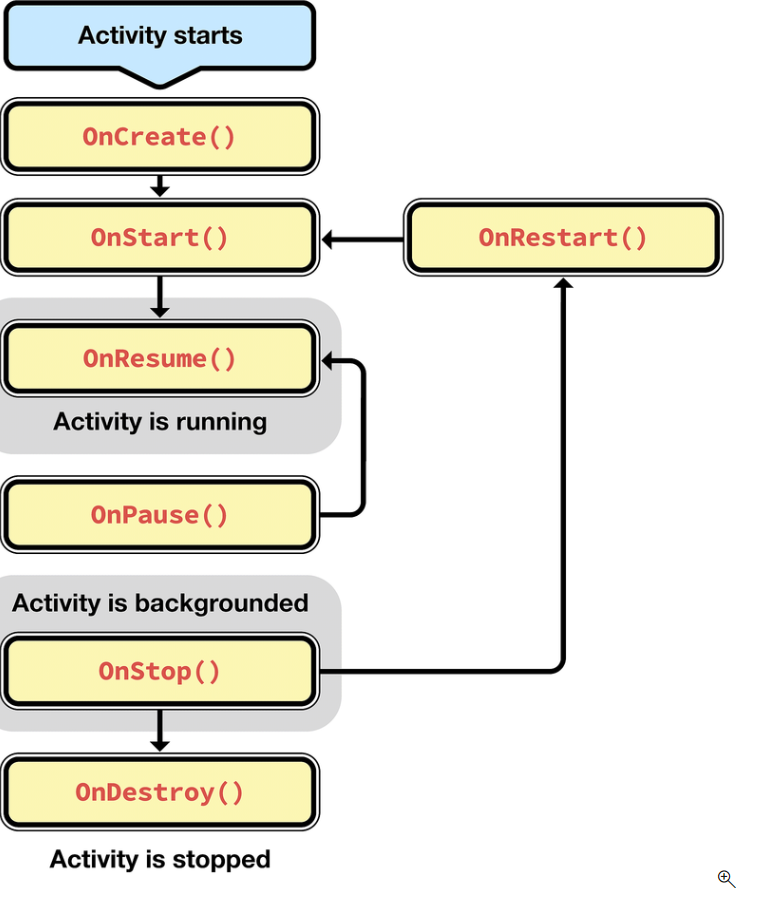
Dentro del código de ejemplo hay una rama para la prueba de los fragmentos.

# Navegación

Al igual que ocurre con otros sistemas moviles, Android posee por defecto una navegación por pila. Es decir, siempre que naveguemos a una nueva pantalla, la anterior se quedará apilada por defecto y el usuario al pulsar atrás podrá volver a esta.

Podemos eliminar la navegación a estas actividades, siempre que usemos el atributo NoHistory propio de la actividad, si habilitamos tal atributo tras navegar hacia adelante, esa actividad será destruida, siendo eliminada de la pila de navegación.

# Ciclo de vida



Como se indica en la ilustración, la actividad tiene un ciclo de vida como el indicado en la gráfica, de esa manera tendremos los diferentes métodos detallados en la imagen para ser sobrecargados.

Muy importante remarcar los eventos del ciclo de vida OnRestart y OnStop que se activan también cuando el usuario manda nuestra aplicación a background (el usuario cambia de aplicación o el botón Home)

Es por ello que debemos tener una programación defensiva al respecto y si estuviésemos realizando alguna operación con el usuario y no hubiese finalizado guardemos toda la información para posteriormente recuperarla.

También debemos señalar que un cambio de orientación del dispositivo hará que la actividad vuelva a iniciarse, haciendo que todo el ciclo de vida se inicie como si hubiésemos creado la actividad desde 0.

# Controles

Dentro de Android tenemos muchos controles que son similares a los que ya conocemos dentro de Windows Forms o WPF. Vamos a tener diferentes controles como los que conocemos, como pueden ser Button, TextView (Label), EditText (TextBox), etc.

Todos los controles heredan de la super clase View o si se trata de un contenedor, también será ViewGroup.

Aunque hay diferentes contenedores los más comunes son: LinearLayout (Similar a StackPanel de WPF) donde nos permite mostrar los elementos apilados. RelativeLayout (que nos permite agrupar los elementos con posiciones relativas de los diferentes controles)

# ListView

Gran parte de las aplicaciones están conformadas en su mayor medida en Listas de datos, es una manera eficiente de mostrarle al usuario la información, además de que también suele ser más eficiente por nuestra parte también.

Crear listas dentro de Android es un proceso que tiene diferentes pasos que debemos contemplar para una correcto funcionamiento. Las partes implicadas en el desarrollo de una lista son:

* Un adapter que nos provee de información
* El control listview
* El layout que conforma cada una de las celdas del listado
* Los datos usados en el listado

Se incluye un ejemplo en el código para indicar como crear un listview. Además, tenemos que tener en cuenta como funciona a grandes rasgos un listado, no se crean todos los elementos, si no que se crean X celdas y estas van reciclándose conforme aparecen y desaparecen.

Debido a que el uso de listados es una de los aspectos más comunes en el desarrollo de aplicaciones, desde la API 5.0 existe una clase para gestionar listados más eficientes que el ListView que es el RecyclerView, en el ejemplo de listados veremos como utilizarlo. Este es un poco más complejo en su composición, ya que se compone de las siguientes partes:

* RecyclerView
* LayoutManager
* Adapter
* Los datos usados en el listado

# Permisos

Como ya indicamos en el apartado del AndroidManifest, a partir de la API 6.0 hay ciertos permisos que deben ser solicitados implícitamente dentro de la aplicación. Estos pueden ser pedidos en cualquier momento de la aplicación, aunque por defecto se suelen solicitar cuando son requeridos o al inicio de la aplicación.

Además, debemos actuar consecuentemente dentro de nuestra aplicación, ya que puede darse el caso de que un usuario no nos haya concedido un permiso para realizar una cierta acción y si no nos defendemos para ello, provocar una excepción.

Dentro del código de ejemplo, se incluye un ejemplo para la concesión de permisos y la programación defensiva pertinente para protegernos de ello.

# Almacenamiento

Dentro de una aplicación tenemos varias maneras de almacenar información dentro de nuestra aplicación. Dentro del sistema Android, tenemos dos espacios de almacenamiento en nuestra aplicación el Interno y el Externo. El interno conforma a todos los datos de nuestro sistema y aplicaciones, donde tendremos una pequeña carpeta incluida dentro de la ruta data/data siendo el sitio de nuestra aplicación el nombre del paquete que hayamos indicado en nuestro AndroidManifest.

Para el almacenamiento externo, el usuario podrá acceder sin problemas, que conforma todas la información incluida en una posible tarjeta SD y en el almacenamiento que el usuario tiene acceso (siempre y cuando no tenga root en el teléfono). En este segundo, nuestra aplicación solo podrá acceder si le hemos concedido los permisos pertinentes.

Dentro de la carpeta interna de nuestra aplicación, podemos almacenar cualquier tipo de fichero y crear todas las carpetas que deseemos. Además tenemos varias maneras de poder persistir dentro de nuestra aplicación, estas serán:

* Settings
* Ficheros
* BBDD

## Settings

Android nos provee de unas clases que nos permiten guardar pequeños valores, para los ajustes de nuestra aplicación, está puede guardarse de manera estándar o de manera segura, somos nosotros los que dependiendo de la naturaleza de la información nos decantaremos de una manera u otra.

Este es el lugar adecuado para almacenar pequeña información de la aplicación relacionada con su funcionamiento y ajustes.

## Ficheros

Como ya hemos indicado, podemos crear todo tipo de ficheros dentro de nuestra carpeta de aplicación, para posteriormente poder consultarlos. Siempre que creemos un archivo sin indicar la ruta este se creará en la carpeta files de nuestra aplicación.

## BBDD

Por defecto, Android tiene los drivers de SQLite integrados dentro del dispositivo, por lo que podemos crear y usar BBDD de SQLite, aunque existen otras BBDD no nativas del sistema, como puede ser: Realm, LiteDB.

Debemos tener por cuenta que SQLite es una base de datos muchísimo más sencilla de lo que estamos acostumbrados a usar por ejemplo en escritorio (SQLServer, MySQL, Oracle, etc) y tiene particularidades como: que no posee relaciones entre tablas y que solo se puede acceder mediante una conexión.

Para todo esto, tenemos diferentes ejemplos

# Información de Android

Nosotros como desarrolladores, tenemos acceso a toda la información del sistema, siempre y cuando tengamos los permisos concedidos por parte del usuario. La manera de acceder a estos datos es mediante consultas SQL, ya que toda la información del dispositivo, contactos, imágenes en la galería, alarmas activas, etc. Se encuentra almacenada en una base de datos SQLite interna.

En el ejemplo veremos como acceder a esta información