

Fundamentos de Android

Indice

Módulo 1: Introducción a Android.....	2
Definición, evolución y composición de Android.....	2
Componentes del sistema operativo.....	2
Módulo 2: Hola Mundo y Android Studio.....	5
SDK Manager, Hola Mundo y Emuladores.....	5
Creación de proyecto android.....	5
Android Virtual Device Manager (AVD).....	5
Android Device Monitor (DDMS).....	10
Logcat.....	14
Módulo 3: Composición de un proyecto de Android.....	14
Archivos, carpetas y recursos en Android.....	14
Elementos de un proyecto Android y Nine Patch.....	14
Soportes múltiples.....	19
Soporte a múltiples pantallas.....	19
Drawable.....	21
Layouts y archivo strings.xml.....	24
Soporte a múltiples idiomas y archivo r.java.....	31
Diemens y styles.....	37
Diemens.....	37
Styles.....	42
Archivos Java.....	43
Android Manifest.....	46
Gradle.....	48
Módulo 4.....	51
Componentes de una aplicación movil.....	51
Activities.....	51
Servicios.....	51
Content Providers.....	52
Broadcast Receivers.....	53
Views.....	53
Clase Intent.....	54
Resumen.....	55
Módulo 5.....	55
Patrones de diseño para android.....	55
Android MCV y MPV.....	55
Material Design.....	57
¿Qué es y cómo funciona Material Design?.....	57
Lectura de Material Design.....	59
Maquetación.....	61
Cómo maquetar.....	61
Desarrollando mi maquetación.....	64

Módulo 1: Introducción a Android

Definición, evolución y composición de Android

Android es un sistema operativo para móviles que fue diseñado por Google, y que además está basado en el Kernel de Linux, esto quiere decir que Android es open source, es una de las cosas que hace increíble a Android.

Componentes del sistema operativo

Está definido por el siguiente diagrama de capas:

- **Capa Kernel:** La capa del Kernel de Linux es la capa más baja que tenemos en nuestro sistema operativo. Android como tal se dice que está embebido en Linux, así que podemos decir que Android es una versión de Linux.

Esta capa, el kernel, se encarga de manejar toda la compatibilidad entre el hardware. Es decir vamos a tener todos los drivers de wifi, de usb, etcétera, etc. Pantallas, todo lo que te imagines de hardware que puede tener tu teléfono.

La capa de Kernel de Linux tiene esos pequeños softwares que se encargan de la compatibilidad. Estos están contruidos con códigos C y C++.

O sea, android depende del kernel en cuanto a:

1. Sistema Central
2. Manejo de Memoria
3. Manejo de procesos
4. Network Stack
5. Controladores de hardware

- **Capa de librerías:** También están contruidas con C y C++, pero estas no son a nivel de hardware como la anterior capa, estas son a nivel de software. Estas librerías se van a encargar de manejar toda la compatibilidad con animaciones y gráficos en 2D y 3D, tipos de fuentes, administradores de datos, de contenidos, navegadores web, etcétera, etcétera. Toda la parte de compatibilidad a nivel de software lo incluye esta capa, librerías C y C++. El Android Runtime es la capa donde se maneja toda la magia de nuestras aplicaciones con el sistema operativo Android.
- **Android Runtime:** Dentro de la capa de librerías, es donde se encuentra la capa Android runtime que es la máquina virtual, la cual es similar en funcionamiento a la de java.

Al igual que el Java, Android tiene una máquina virtual, esta máquina virtual se llama Dalvik. y se llamaba así hasta la versión 21 de Android, con Android Lollipop. Ahora tenemos una nueva maquina virtual que se llama ART.

La máquina virtual lo que hace básicamente es generar que tus aplicaciones corran en ese sistema operativo. Si no estuviese la máquina virtual instalada en tu sistema operativo, nunca podrías correr tus apps. La máquina virtual, esta nueva máquina virtual ART hace que tus aplicaciones pesen un poco más pero consuman menos recursos. Es decir tus aplicaciones que corran en tu sistema operativo Android, pues ya no van a terminar con tu batería, y bueno además que nos van a ayudar a muchas otras cosas, entonces básicamente es reducir los recursos. Va a crecer un poco más tu aplicación, va a tener un poco más de peso pero se compensa bastante con consumir menos batería en tu sistema operativo.

Notas:

- En android todas las aplicaciones se programan en lenguaje Java y se ejecutan mediante la máquina virtual Dalvik o ART, las cuales fueron optimizadas y adaptadas a las particularidades de los dispositivos móviles (baja capacidad de procesamiento, baja memoria, alimentación por batería).
- Trabaja con ficheros de extensión .dex
- No trabaja con bytecode de Java sino que lo transforma en uno más eficiente
- **Application Framework:** Ésta capa es donde vamos a ir trabajando durante todo el curso, pues aquí están todas las clases de Java listas, que el SDK de Android ya tiene listas para para que generes tus nuevas clases y tus nuevas aplicaciones móviles.
- **Capa de aplicaciones:** Application Layer, esta es la última capa de la composición de nuestro sistema operativo y es la capa donde actúa el usuario final. El usuario final aquí ya tiene acceso a la aplicación que generaste y también a las aplicaciones que ya vienen instaladas en el sistema operativo. Así que Application Layer es la capa donde nuestros usuarios finales gozan de nuestras aplicaciones móviles.

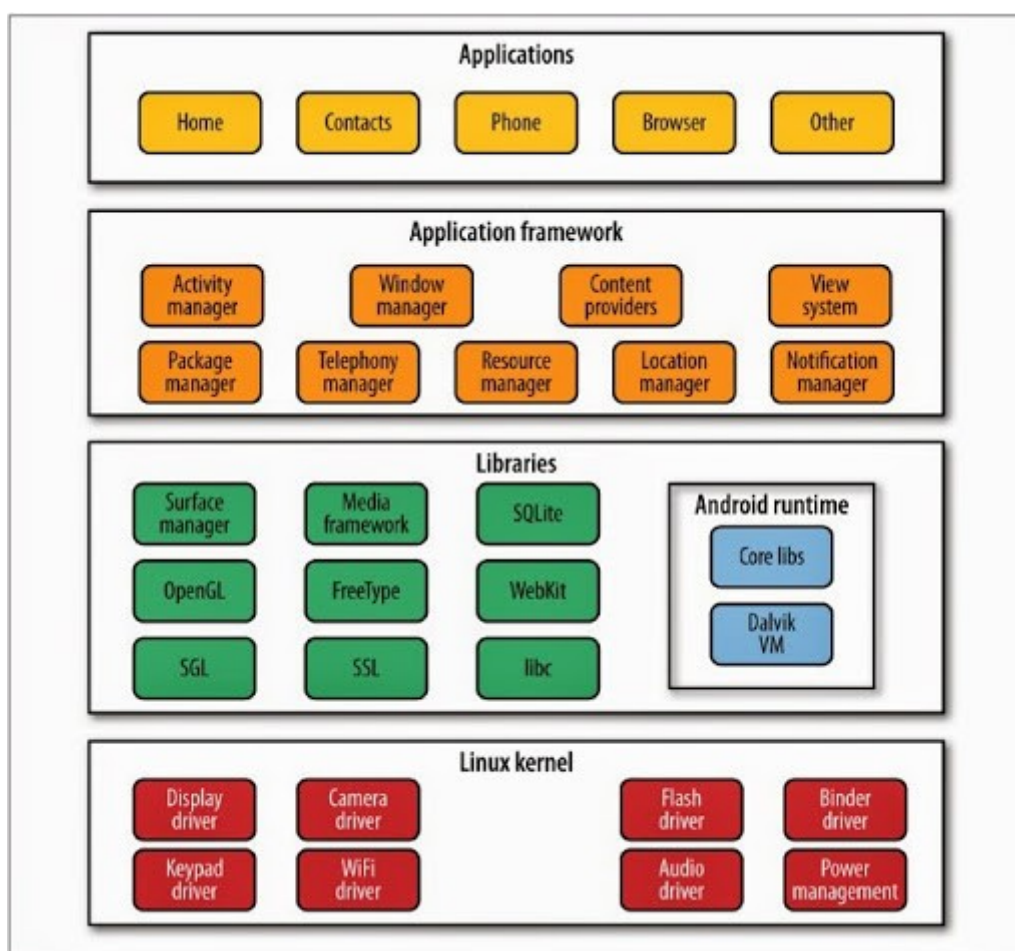


Figura 1: Diagrama de capas

Requisitos para desarrollar las apps:

1. **Un IDE:** El nuevo entorno de desarrollo oficial es Android Studio pero se puede usar otro.
2. **SDK Platform:** sin este no vas a poder desarrollar absolutamente nada. Aunque tengas el IDE y aunque tengas lo que sea si no tienes el SDK de Android, no puedes desarrollar tus apps. ¿Qué tiene el SDK de Android? Bueno pues todas la librerías de Java, la librería Activity, la librería Notification, la librería Geolocation, etc. Todas estas librerías nos vamos a basar en ellas para construir nuevas aplicaciones Android.
3. **SKD tools:** el Android SDK tools nos va a ayudar a que cuando tengamos algún problema en nuestra aplicación podamos realizar un proceso de debugging, es decir este proceso nos va a ayudar a que podamos localizar en dónde se encuentra nuestro error o cómo se va dando el flujo de nuestra aplicación.
4. **Android SDK build tools:** el Android SDK build tools nos va a ayudar a construir nuestra aplicación. Es decir todo el código que ya generaste, el Android build, como su nombre lo dice, lo va a construir y va a generar finalmente una aplicación, algo tangible, algo visible para tus usuarios.

Módulo 2: Hola Mundo y Android Studio

SDK Manager, Hola Mundo y Emuladores

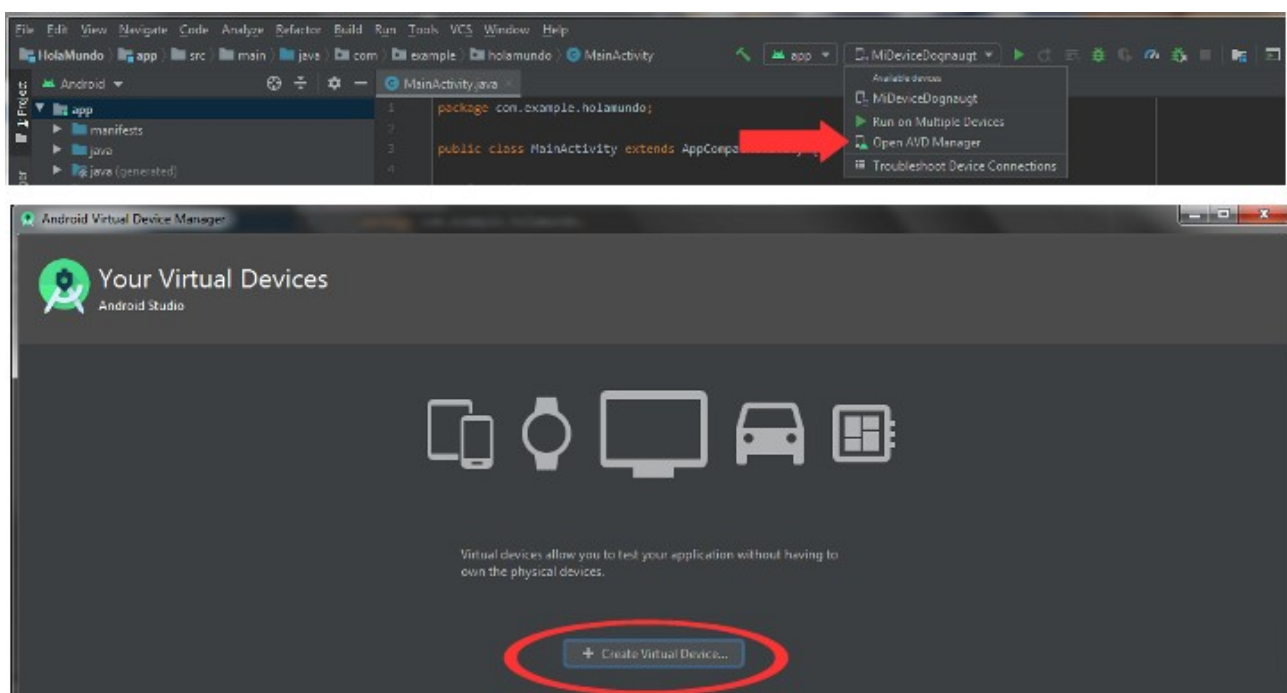
Acá ver cómo se hace para android studio

Creación de proyecto android

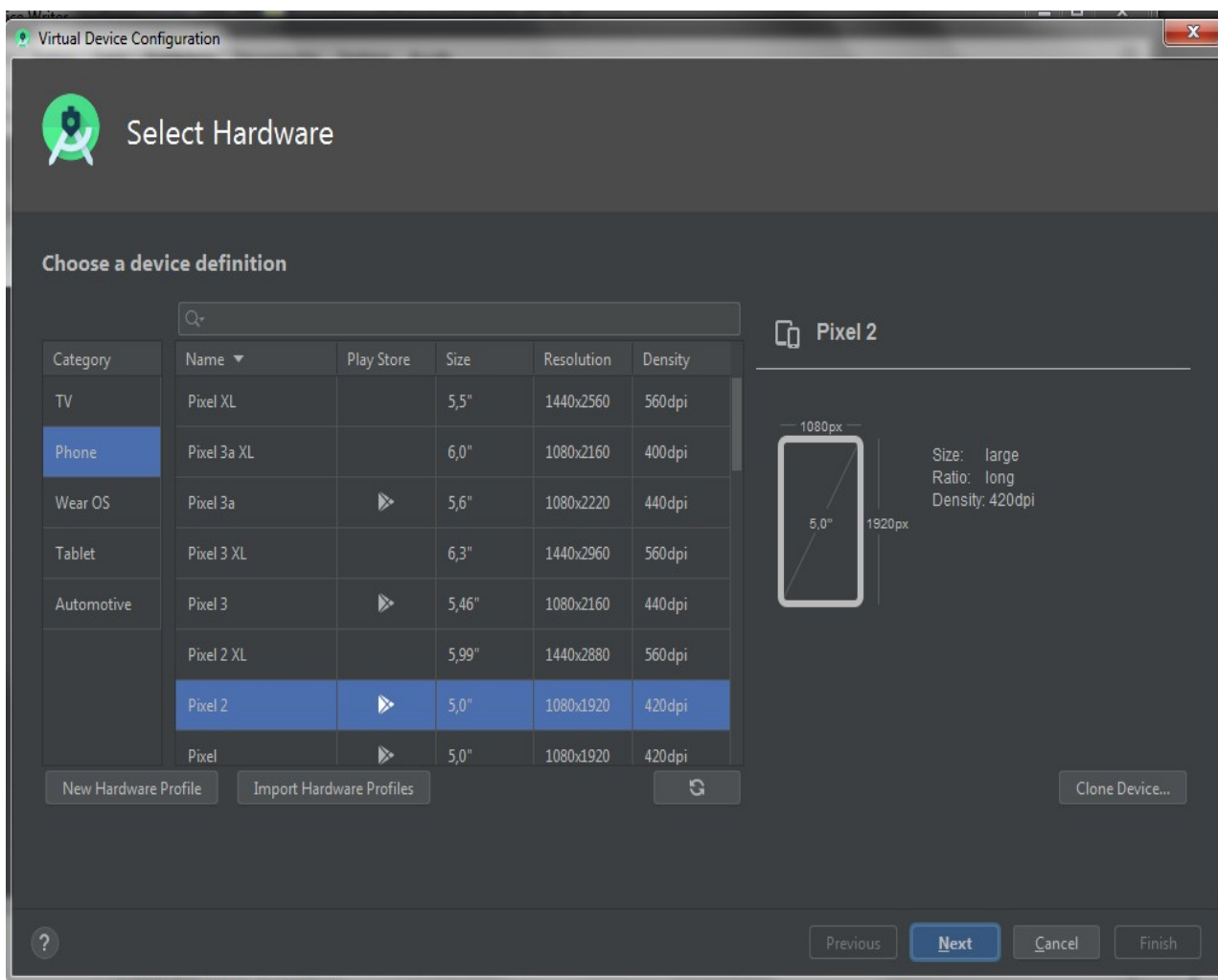
Android Virtual Device Manager (AVD)

Bien, ya se ha creado nuestro proyecto de Android y automáticamente se ha creado nuestro primer hola mundo. Como observas, por default nos tiene 2 archivos abiertos, uno que contiene todo nuestro código Java y el otro contiene todo nuestro código XML que va a definir nuestra interfaz de usuario.

Para poder correr nuestro proyecto de Android, nuestro hola mundo, es decir verlo en acción, puedo utilizar un emulador de Android. Entonces, para generar un emulador de Android voy a ir a esta opción, que es nuestro AVD manager, nuestro Android Virtual Device manager.



Una vez que se le da a “create virtual device”, lo primero que nos pide es el hardware. Recuerda que vamos a construir un dispositivo virtual de Android, es decir, un teléfono virtual, literal, un teléfono con el cual vamos a poder hacer casi todo lo que puedes hacer con tu teléfono físico.



Primero tengo que definir si el hardware que voy a manejar, voy a emular, va a ser un Android TV, un teléfono, un Android Wear o una tablet. Estas opciones van a estar visibles dependiendo de en mi SDK manager lo que haya seleccionado. Entonces para teléfono, puedo seleccionar por ejemplo un Nexus S, un Nexus 1, Nexus 6P, etcétera, etcétera. Puedo también seleccionar algún hardware ya por default, de esos que están acá que son un poco más genéricos, y literal, cuando seleccionamos por ejemplo un Nexus 6P, en cuanto a hardware literal, se va a construir como un Nexus 6P es en su hardware. Esto va a incluir si tiene cámara frontal, cámara trasera, memoria RAM, memoria interna del teléfono, etcétera, etcétera. Ahora, nosotros podríamos construir nuestro propio hardware.

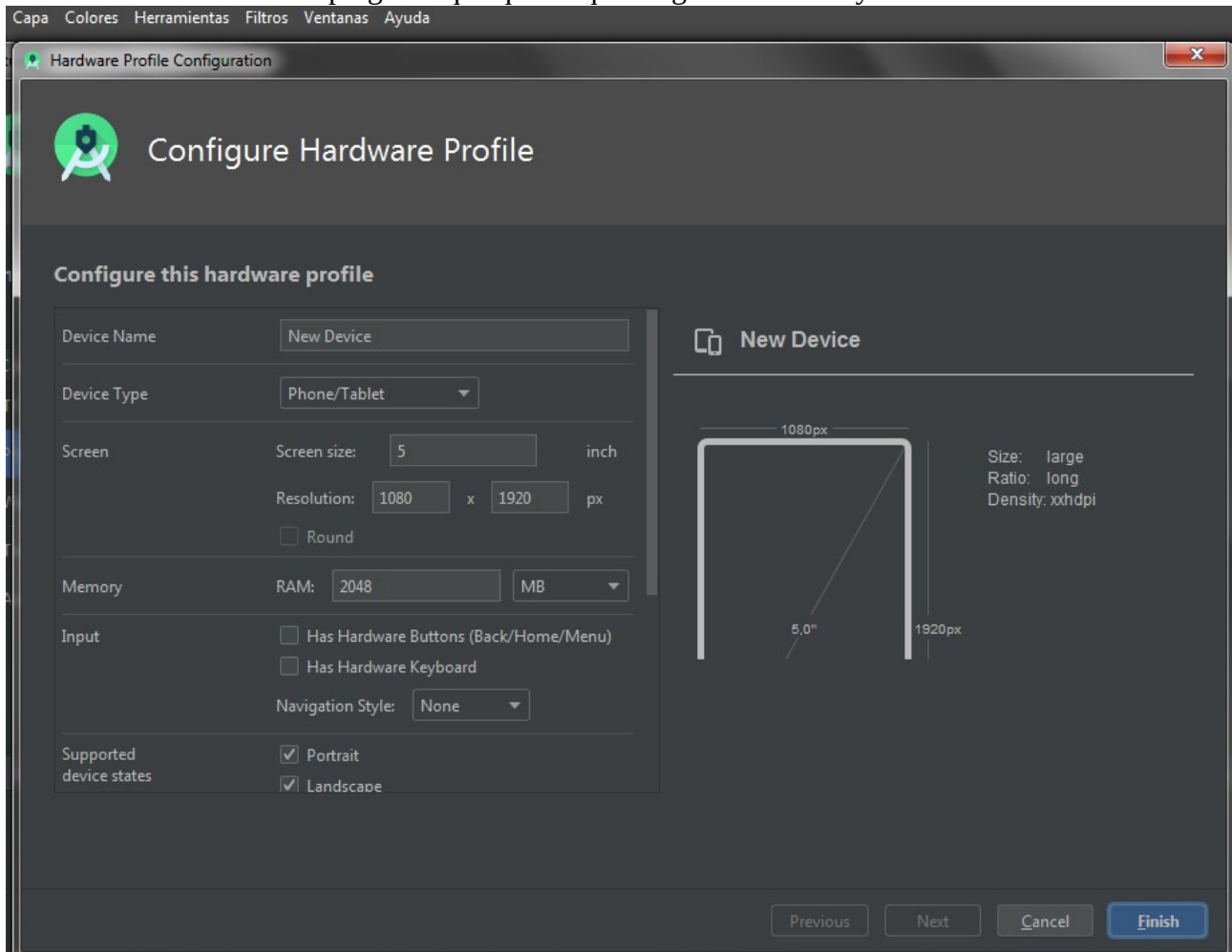
Para hacerlo podemos darle clic aquí en “new hardware profile”.

Bien, puedo definir que el teléfono tenga presente los botones de hardware, como por ejemplo el botón de back, el de home y el de menú. Hay teléfonos que tienen presentes esos botones, que no son touch, que no son táctiles, sino que son botones de hardware.

Puedo definir que también el teléfono por ejemplo tenga un keyboard, un teclado, también físico, un teclado de hardware puedo definirlo, eso sí lo quiero, y puedo definir también el tipo de navegación que quiero que tenga ese emulador. Por ejemplo, un trackball, un trackball es como lo que conocimos hace mucho con Blackberry. Esas bolitas que tenían como mouse, que fungía como mouse, eso es un trackball, yo puedo definir que mi teléfono, mi emulador tenga un tipo de tipo de

navegación como esa, o puedo decir ninguno, en nuestro caso será ninguno porque será un teléfono touch.

Le doy clic ahí y lo que vemos es primeramente el nombre del hardware, cómo quiero que se llame el hardware. Puedo ponerle Android, Dispositivo Android. Puedo seleccionar que esto sea un teléfono o una tableta, puedo seleccionar que sea un Android Wear o un Android TV, puedo seleccionar la cantidad de pulgadas que quiero que tenga mi teléfono y además la resolución. Lo

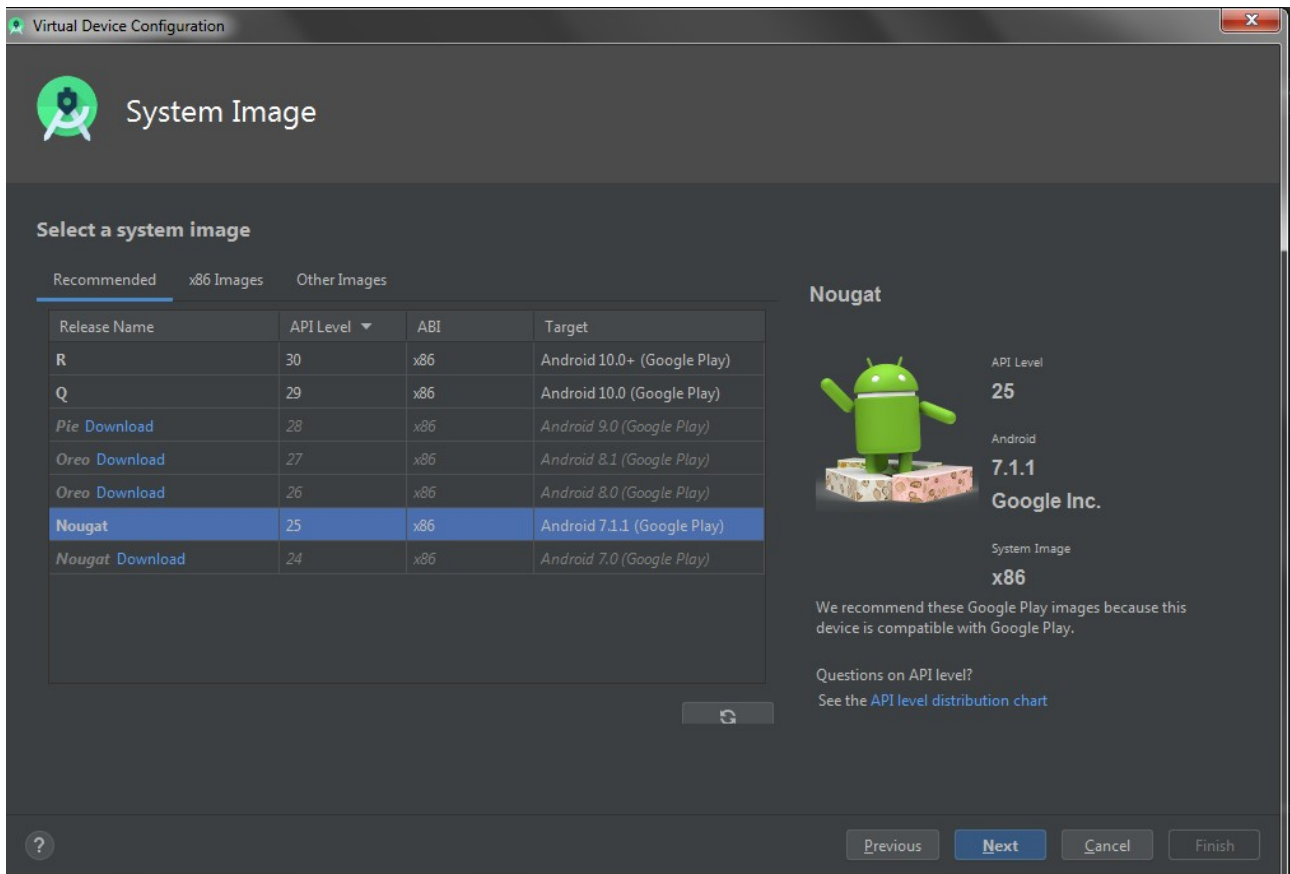


voy a dejar tal cual como se ve aquí.

Puedo manejar también cuántos gigas de RAM quiero que tenga este hardware. Aquí es un punto muy interesante, dado que cuando nosotros arrancamos el emulador, arrancamos este emulador, son 2 gigas de RAM que se tomarán físicamente de tu computadora, es decir si tu computadora tiene por ejemplo 3 gigas, y seleccionas 2 gigas de RAM, bueno, 2 gigas por lo pronto por parte del emulador ya van a estar ocupados. No sé donde está esto en la version nueva de Android Studio

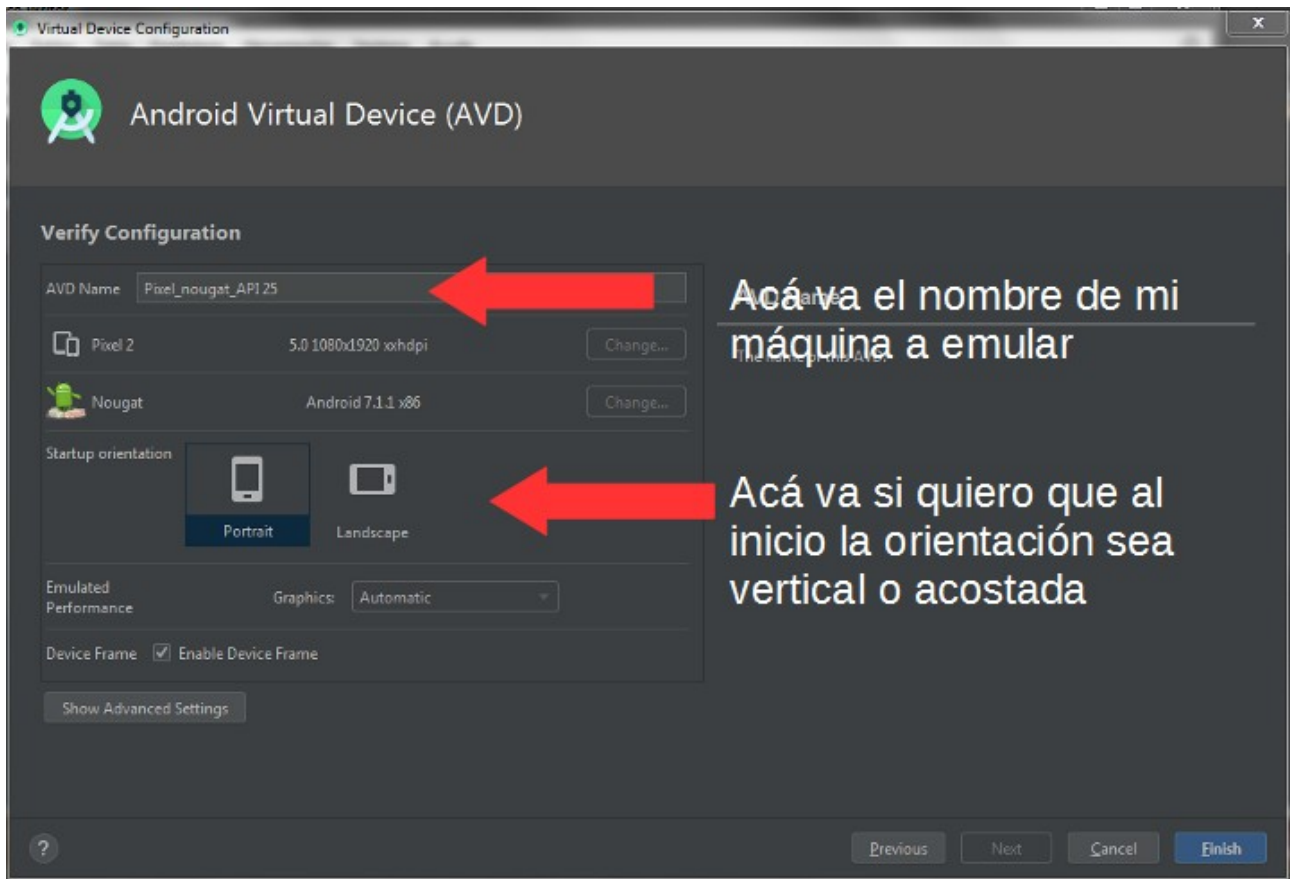
Si, y aparte también lo que consume Android Studio y entonces de pronto la computadora se empieza a hacer lenta, de pronto Android Studio se empieza a trabar, etcétera, etcétera, entonces es muy importante que este campo lo asignemos de acuerdo a los recursos que tenemos en nuestra computadora.

Luego lo que nos pide es la imagen de sistema android que quiero usar en ése dispositivo.



Esta parte me da 3 pestañas, las imágenes recomendadas, las x86 y otras, lo cual hace la versión del SO del dispositivo Android sea a gusto del consumidor. Yo voy a elegir porque puedo la Nougat con versión de API 25.

Lo siguiente es ponerle un nombre al dispositivo al emular:

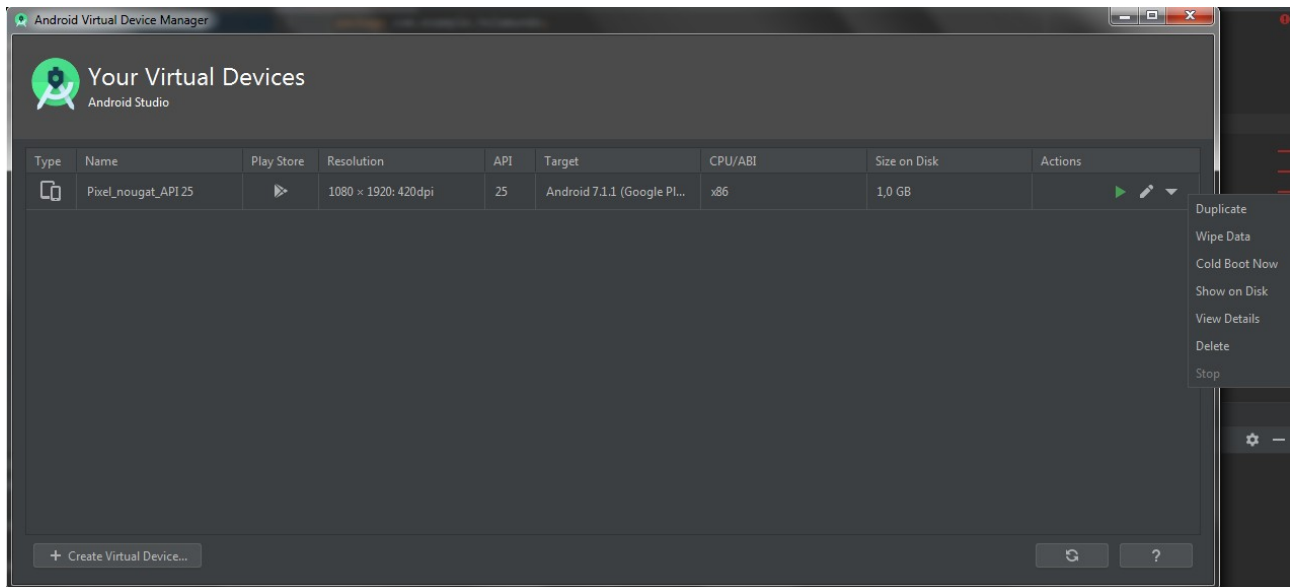


Puedes darle clic donde dice “show advanced settings”. Aquí puedes decir que tenga cámara frontal, pero aquí tú puedes decir de dónde quieres que tome la cámara. frontal, es decir, si quieres que la cámara frontal sea emulada por el emulador, o si quieres que la tome directamente desde tu webcam, Igualmente para la cámara trasera.

Podemos también definir una velocidad de red, en este caso está a full, y además podemos incluso emular una velocidad edge, si así lo quieres, o todas las que vienen ahí, podemos todavía cambiar un poco los ajustes de la memoria RAM, además el almacenamiento interno, hemos colocado 200 megas y también si quieres que tenga disponible una SD Card, la cantidad de almacenamiento que tiene disponible esa SD Card. 100 megas por ejemplo, puedes colocarle gigabytes, terabytes, etcétera, pero recuerda que eso también será memoria física que será tomada de tu computadora.

Entonces si seleccionas que tenga una SD Card de 1 terabyte, bueno debes tener el espacio suficiente en tu computadora. Viene el skin y el teclado habilitado también. Con esto ya está listo nuestro emulador. Voy a darle finish.

Luego, para ver los emuladores disponibles, se va de nuevo al AVD manager y ahí aparece la lista:

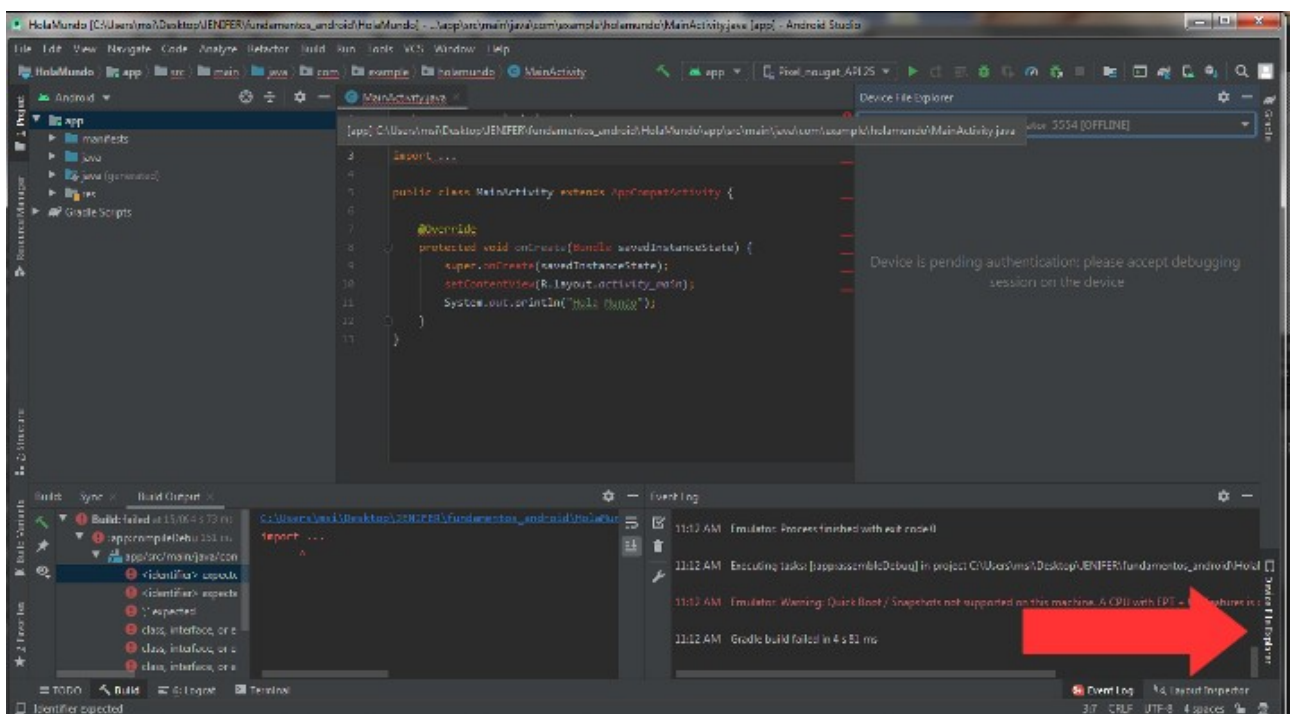


Con la flechita verde se le da a correr el dispositivo emulado que se quiera, con el lápiz te lleva a la posibilidad de editarlo y con la flechita blanca, podés duplicar el dispositivo, eliminarlo y otras opciones.

Si le das play, enciende el emulador y una vez que nuestro emulador esté encendido, vas a poder ver, pues todas las características que tiene el sistema operativo Android, pero ahora aquí en tu computadora.

Android Device Monitor (DDMS)

Lo siguiente es todo para Android Studio viejo porque el DDMS ya en la versión 3, no está. Lo más parecido es Device File Explorer

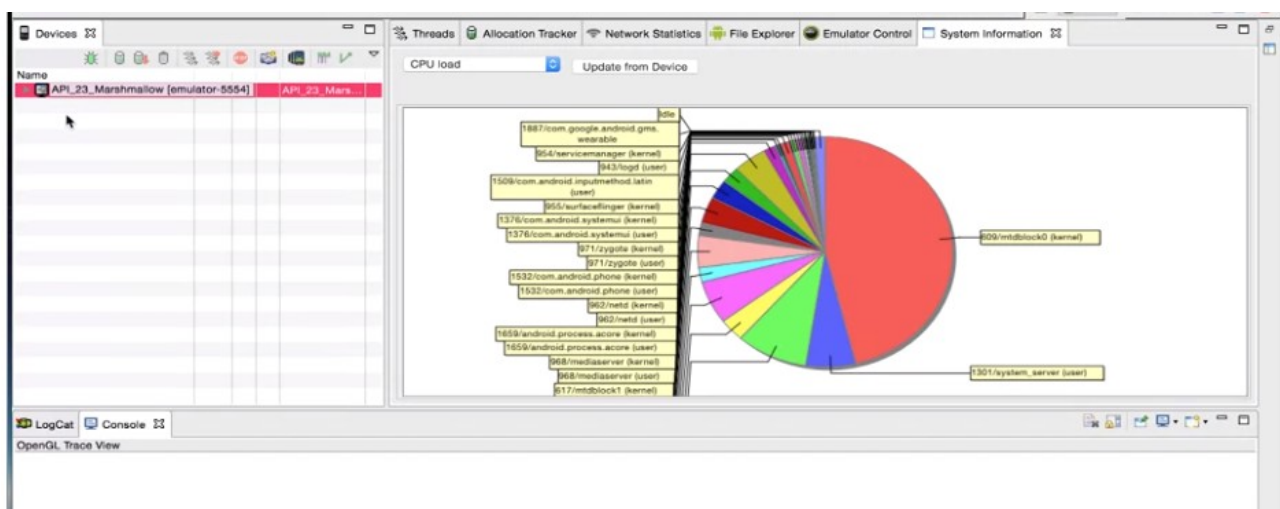


Muy bien. Una vez que nuestro emulador ha encendido, ya está listo para empezar a utilizarlo. Podemos desbloquearlo tal cual como si fuese un teléfono. Podemos entrar al menú, estar inspeccionando. Y como ves tenemos muchas apps. Si queremos podemos entrar, por ejemplo, al reloj, configurar una alarma, etcétera, etcétera. Esto funciona tal cual como un dispositivo físico.

Tenemos otra peculiaridad dentro de nuestro ID Android Studio y este es el Android Device Monitor.

El Android Device Monitor, como su nombre lo dice, es un monitor de dispositivos Android. Es decir, a través de esta herramienta vamos a poder estar monitoreando cómo funciona, cómo se está comportando nuestro emulador. Si tenemos algún dispositivo Android conectado a nuestro ID también vamos a poder estarlo manipulando y, bueno, en cuanto abren vemos algo así.

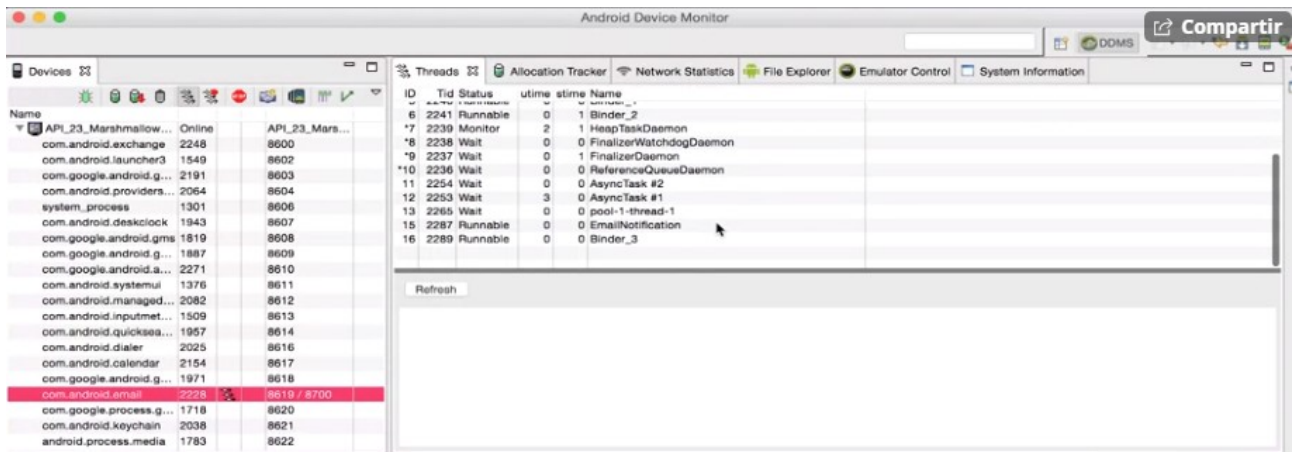
En el lado izquierdo veo la lista de dispositivos que tengo en este momento conectados. Hasta el momento sólo tengo mi emulador que es un API 23 con Marshmallow (yo usé otra versión).



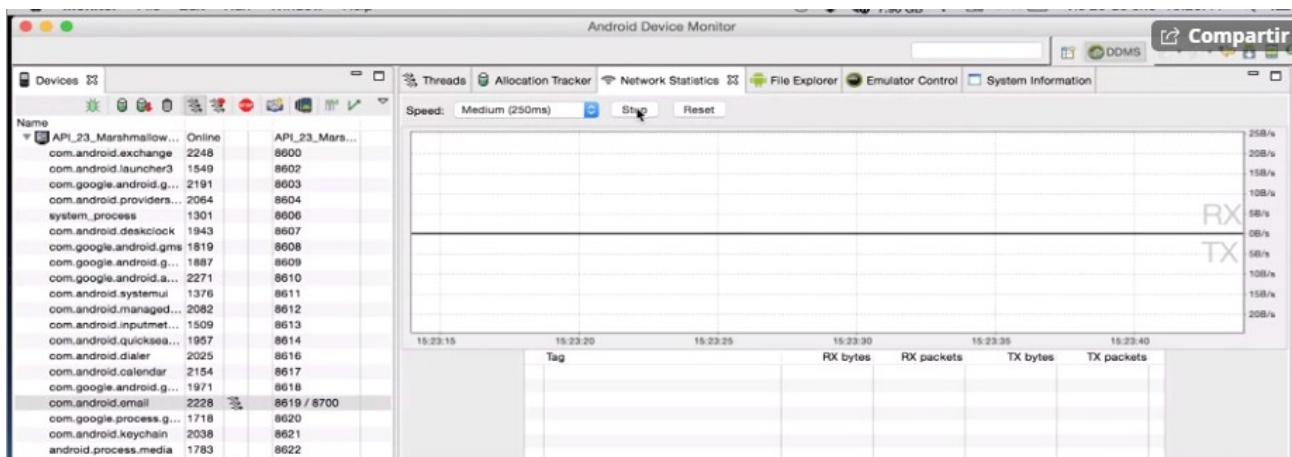
Viene un poco la primera pestaña que está abierta es un poco de la información del sistema. Donde se muestra un gráfico, un estadístico de más o menos la cantidad de recursos que está consumiendo cada aplicación en nuestro dispositivo.

Tenemos esta aplicación que es de threads donde al seleccionar nosotros una aplicación móvil, una aplicación que está instalada aquí.

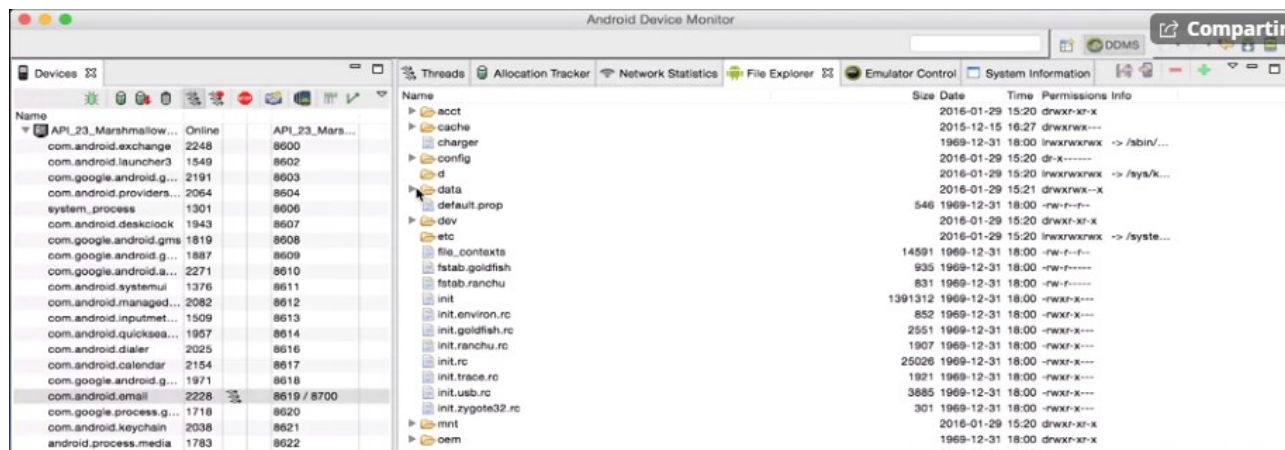
Recuerda que las podemos identificar a través de su package, como se ve aquí con .android.imei, esta es la aplicación de IMEI tal cual. Puedo seleccionar este botón que está aquí, threads, y automáticamente puedo estar monitoreando cómo es que se están dando el pull de hilos que maneja esta aplicación.



Puedo también estar trabajando un poco con la memoria. Puedo estar trabajando con la memoria, cómo se está comportando esta aplicación de memoria. Y además puedo estar testeando un poco la red, cómo se comporta esta aplicación en este tipo de red. Una red medium que son 250 microsegundos. Una red fast, o una red slow.



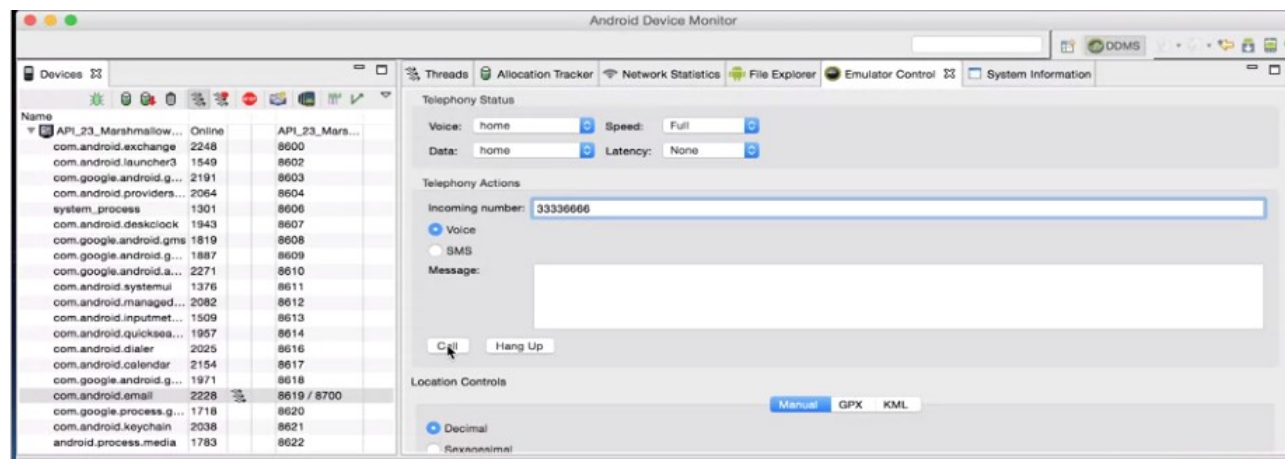
Entonces puedo detenerlo y básicamente aquí veríamos un gráfico de cómo está comportándose nuestra aplicación. Si es de una manera lenta en una red lenta, o de una manera rápida, incluso en una red lenta. Además tenemos el file explorer, que esto es una herramienta muy interesante, pues aquí tenemos, literalmente, el sistema de archivos de nuestro sistema operativo. Recuerda que Android está basado en Linux, entonces el sistema de archivos es muy similar al que maneja Linux como tal.



Vamos a tener acceso a todas las carpetas que componen ese sistema operativo. Y en particular existe la carpeta data, y dentro de sí está otra carpeta data. Y ahí vamos a encontrar todas las carpetas contenedoras de nuestras aplicaciones.

Al igual como en Windows por ejemplo, cuando tú instalas un software generalmente se va a una ubicación como program files. Y ahí encuentras una carpeta que contiene todos los archivos del programa, del software que estás utilizando. Algo así es la carpeta data data. Ahí vas a encontrar el sistema de archivos de esa aplicación en sí. Y va a ser ubicada a partir del identificador o del package name que le diste cuando la creaste. Por eso tiene que ser única, ninguna otra aplicación puede tener el mismo package name que originalmente has dado a una aplicación.

Posteriormente tenemos la siguiente pestaña que es emulador control, donde aquí podemos emular si yo quiero una llamada de entrada, por ejemplo. Yo quiero que mi emulador tenga una llamada por ejemplo del número 333, 6666, y puedo decirle call.



En ese momento mi emulador empieza a simular como si yo tuviera una llamada ejecutada por el número que acabo de escribir. Puedo contestar o puedo denegar la llamada si es que así lo quiero. Recuerda, esto es como un dispositivo físico. Puedo también por aquí detenerla, colgar la llamada.

Además también puedo decirle que le mande un mensaje, que le envíe un mensaje. Entonces puedo ver cómo me he llegado un mensaje.

Y bueno, esto son cosas que quisieras probar en algún momento en tu aplicación, puesto que cuando la aplicación está al frente hay que ver el estado de la aplicación cuando entre una llamada, por

ejemplo, qué va a pasar con la aplicación. O hay que ver el estado de la aplicación, cómo va a funcionar cuando entre un mensaje de texto. A veces quisieras manipular este tipo de cosas.

Adicional, yo puedo también emular que mi teléfono esté ubicado en cierto lugar. Y esto lo haré seleccionando la longitud y la latitud. Yo puedo decirle que mi teléfono esté ubicado en Timbuktú cuando a lo mejor estoy en la Montevideo. Entonces puedo decir, puedo hacer esto para poder predecir cómo va a funcionar una aplicación, a lo mejor cómo se vería en otra geolocalización, por ejemplo.

Entonces todas estas herramientas del Android Device Monitor me van a ayudar a manipular peculiaridades o cosas necesarias, cosas que tengo en mi teléfono, en mi aplicación móvil.

Logcat

En LogCat, lo que vas a observar es todo un log. Es decir, toda una serie de tareas de registros que se están mostrando aquí. Que son de todos los procesos que están ejecutando en tu emulador o en tu dispositivo móvil si es que tienes uno físico. Todo esto va a estarse mostrando aquí y es muy importante, porque cuando tu aplicación móvil por algún motivo truena o por algún motivo no corrió como tú querías, todo eso lo vas a ver registrado aquí en el LogCat.

Si tienes un error aquí se mostrará qué error está ocurriendo, qué está pasando. Y a veces hasta te puede dar tips de cómo solucionarlo. Te puede dar instrucciones que revises tal archivo o que revises tal línea de código donde ocurrió el error.

Es muy importante que también nosotros como programadores estemos acostumbrados a utilizar el log para cualquier serie de mensajes o notificaciones que necesitemos como información adicional para ti, programador, sólo para ti, programador.

No te acostumbres a utilizar estos mensajes en notificaciones que de pronto se vean en la interfaz de usuario. Eso puede provocar bugs o puede provocar malas experiencias de usuario con tu aplicación. Entonces acostúmbrate mucho a usar el LogCat. Hay librerías en Android muy útiles para poder estar debuggeando, para poder estar mandando mensaje desde nuestra consola LogCat. Esto es nuestro Android Device Monitor, espero que te sirva de mucho y que puedas estar jugando bastante con él.

Módulo 3: Composición de un proyecto de Android

Aquí aprenderemos además de cómo se compone un proyecto en Android, todos sus archivos y carpetas, etcétera. Vamos a aprender a dar soporte para múltiples pantallas en Android. Pantallas en portrait, pantallas en landscape. Además también soporte para múltiples idiomas y múltiples dispositivos

Archivos, carpetas y recursos en Android

Elementos de un proyecto Android y Nine Patch

Un recurso que mencionamos hace un momento son las imágenes. Recuerda que Android es un sistema que ha crecido muchísimo y que puedes encontrar en infinidad de dispositivos. Dispositivos

con pantallas pequeñas, medianas, grandes y cada uno va a tener incluso hasta su propia medida de pantalla. Bueno, muchas veces en las aplicaciones el manejo de imágenes en diferentes pantallas es un problema. Para esto tenemos un sistema de archivos llamado Nine-Patch.

Los archivos Nine-Patch es un sistema para ejecutar un reescalado automático de imágenes. Es decir, si tu imagen está siendo vista en un dispositivo pequeño, a lo mejor se va a ver muy bien, en cuanto a pixelaje, que no se vea borrosa, etc. Pero si de pronto la pantalla del dispositivo crece, pues entonces, la imagen si es de tipo Nine-Patch, la imagen se va a ajustar al tamaño, a la magnitud de la pantalla que tengas en ese momento. Este tipo de imágenes son comúnmente utilizadas para fondos de pantalla, fondos de botones. En general cualquier tipo de fondo, también conocidos como backgrounds. Y puedes ubicarlos por la extensión ".9.png".

Un Nine-Patch es un sistema que maneja Android para hacer el reescalado automático de imágenes. Esto quiere decir que una imagen puede aumentar o disminuir de acuerdo a las dimensiones de la pantalla donde estás viendo tu aplicación móvil. Los archivos Nine-Patch comúnmente son utilizados para backgrounds de cualquier cosa.

Podría ser un botón, el mismo background de la aplicación móvil, etcétera. Estos archivos van a tener la peculiaridad que tiene la extensión .9.png.

Creamos ya nuestro "Hola mundo", ya lo hemos corrido, pero vamos a ver toda la serie de archivos y carpetas que se han creado en nuestro proyecto. Te voy a explicar un poco cómo funciona Android Studio.

Del lado derecho es tu panel de actividades, del lado derecho siempre vas a tener los archivos en cuestión y aquí es donde vas a escribir todo tu código. Vas a tener tu archivo de Java, tus archivos XML, etc.

Pero del lado izquierdo tenemos un panel donde ahí tenemos jerárquicamente acomodados nuestros archivos. O la composición real de nuestro proyecto.

Nuestro proyecto, es esta vista que tenemos de Android. Tiene, comienza con la carpeta APP.

Si abrimos esta carpeta vamos a ver más carpetas, así que voy a comenzar explicándote qué son los recursos en Android. En Android básicamente todo lo que no es código JAVA, se va a considerar un recurso. Es decir, casi todos los recursos en Android van a ser compuestos por códigos.

Es decir, todos los recursos, todo lo que tu aplicación vaya a utilizar como recurso, va a estar contenido en nuestra carpeta. Como observas, la carpeta se llama res, dentro tiene el directorio drawable, después layout, mipmap y values. Así que, vamos a ver cada una detalladamente en nuestras siguientes secciones.

Por ejemplo, si se tiene una imagen muy pequeña la idea es que esta imagen pueda adaptarse. Si es un botón grande, un botón a lo mejor más pequeño o pueda crecer tanto horizontal como verticalmente. Para eso vamos a generar nuestro Nine-Patch.

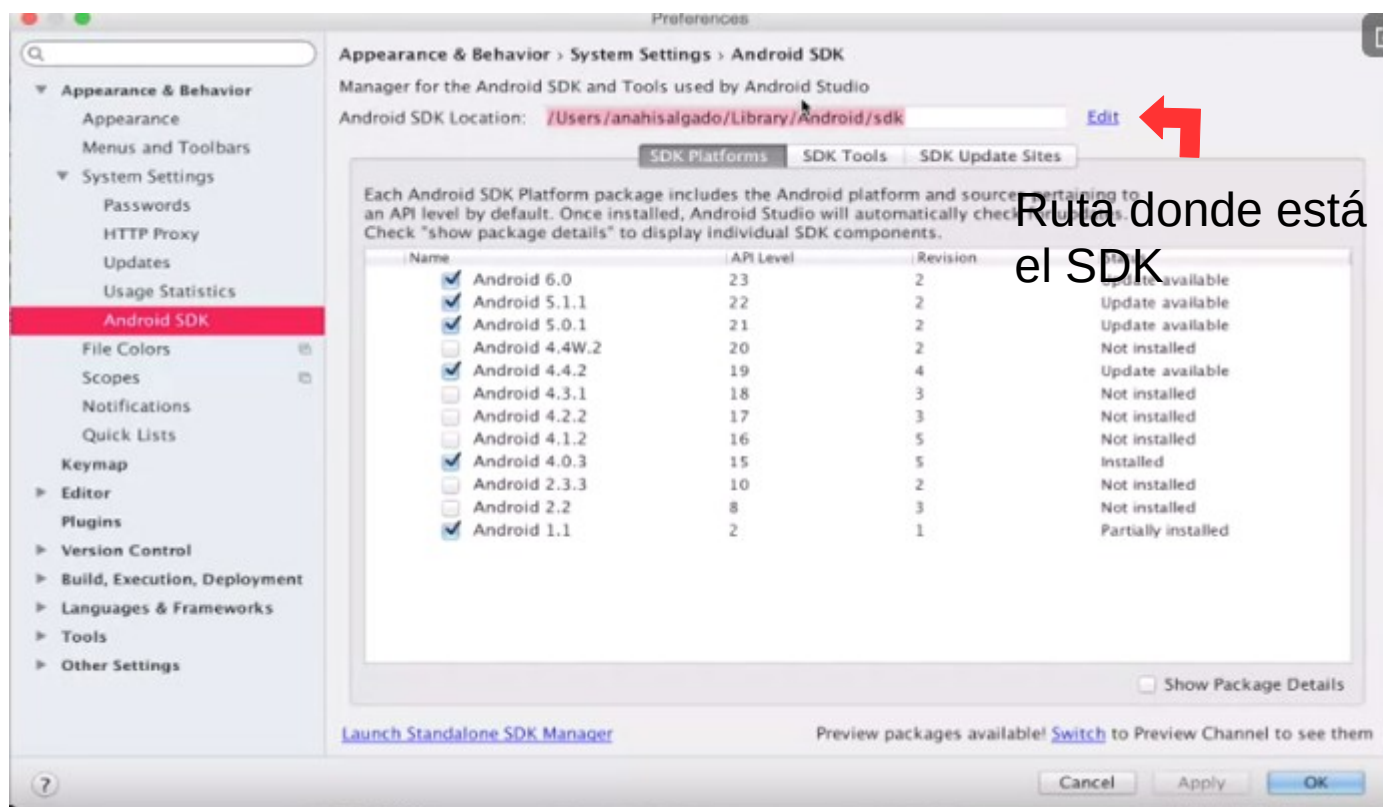
Utilizaremos una herramienta que viene incluida en el SDK de Android. Hay muchas herramientas que puedes consultar online y que lo único que reciben como parámetro es el archivo PNG y automáticamente te crean el archivo Nine-Patch. Esta herramienta ya viene incluida dentro del SDK

de Android. Para entrar a ella, lo debes hacer desde la terminal o desde la consola de comandos, ya sea que utilices Windows, Linux o Mac.

Entonces, primeramente debemos ubicarnos en donde está nuestro SDK de Android. Una forma donde puedes saber. Una forma por la cual puedes saber donde está.

Es yendo a las preferencias de Android Studio. En Mac pues simplemente es Android Studio, en preferences.

En Windows es en el menú file y en una opción que se llama settings. En settings te debe abrir una ventana como esta.



Entonces, inmediatamente pues, en el Android SDK vemos la ruta donde está ubicado nuestro SDK, que a partir de ahí, se están jalando todas las apis y todas las opciones que tenemos de Android. Otra manera también de abrir esto, pues ya sabes, es con el SDK manager, este que está aquí. Este SDK manager también nos va a abrir esta ventanita y entonces tenemos la dirección donde se encuentra ubicado nuestro SDK.

Para ejecutar en la consola

Mi ubicación es `Users/anhaisalgado/Library/Android/sdk`. Perfecto, con el comando `cd` es como nos podemos mover entre directorios. Ahora voy a visualizar lo que está adentro. Vamos a entrar a la carpeta `tools`, la carpeta `tools` que está dentro de la ubicación a donde estuvimos yendo. Listo ahora colocamos el comando `ls` (o dir en windows) y nos permite ver todo el listado de los ejecutables que contiene este directorio. Uno de ellos es este, `draw9patch`.

Es un archivo ejecutable, entonces tenemos que ejecutarlo como se ejecuta un archivo, en el caso de Windows un archivo .bat o en el caso de Unix, un archivo sh. Para el caso de Unix, colocaremos ./draw9patch. Cuando hablo de Unix me estoy refiriendo a Mac y a Linux. Para el caso de Windows, para ejecutar esto, simplemente quitamos el .diagonal y a continuación damos enter.

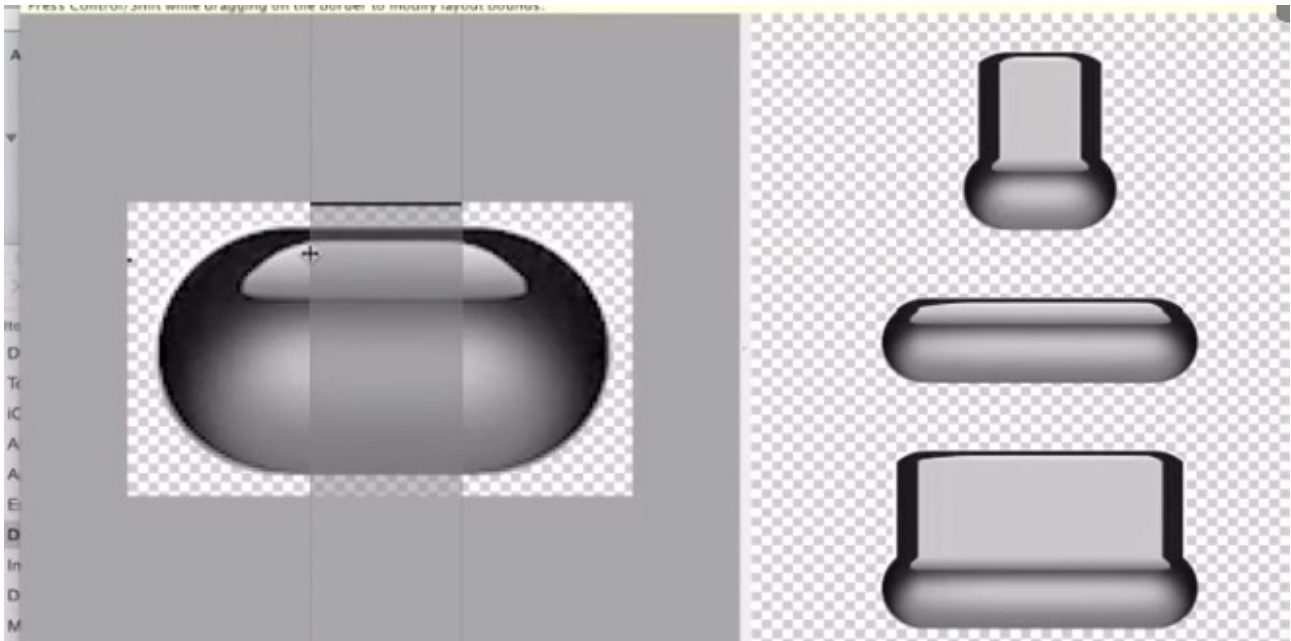
Resultado de correr ninepatch

Al dar enter nos abre esta ventana, que es nuestro inspector que va a recibir nuestra imagen para convertirla en un Nine-Patch. Ahora vamos a arrastrar la imagen que ya tenemos, se llama buttons.png, la voy a arrastrar aquí, y automáticamente me está abriendo esta vista, ok. Si yo acerco el cursor a la imagen puedo ver esta sombra que me está colocando los delimitadores de cuánto va a crecer mi botón, dependiendo si es horizontal o verticalmente. Como observas el botón, tal vez, verticalmente se ve mas o menos, aunque no es lo óptimo. Horizontalmente se está deformando demasiado.

Y este quiere decir que cuando lo crecemos en ambas direcciones, tanto vertical como horizontalmente. Bien, vamos a arreglar esto un poco, vamos a hacer primeramente que el botón, vamos a mover estas rayitas y vamos a hacer que en lo gordito, en el ancho solamente crezca en esta zona.



Eso es lo que hacemos con estas líneas, tú delimitas qué zona del botón quieres que crezca. Es como si de pronto el botón se convirtiera en un vector y entonces ya puedes delimitar las zonas. Ahora tenemos que marcar estas rayitas en todas las esquinas de nuestra imagen.

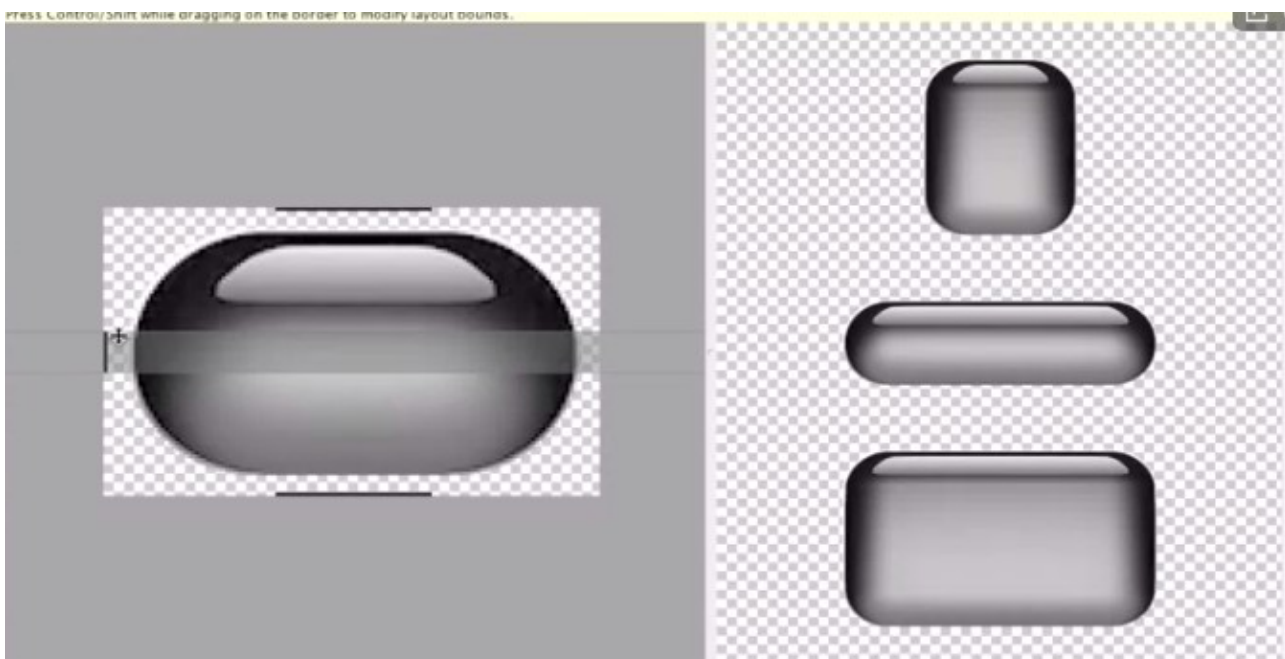


Es como si de pronto el botón se convirtiera en un vector y entonces ya puedes delimitar las zonas. Ahora tenemos que marcar estas rayitas en todas las esquinas de nuestra imagen.

También en la parte inferior tengo que delimitar qué zona quiero de la parte inferior que crezca, voy a seleccionar esta zona.

Como te comento esto es muy común utilizarlo en imágenes de fondo. ¿Por qué? Porque precisamente las imágenes de fondo se van a acomodar a la perfección tanto si crecen o decrecen.

Ahora vamos a acomodar este cachito, bien, lo tengo aquí. Este botón como tiene un gradiente voy a seleccionar una zona plana de esto para que no se note mucho lo que crece.



Me falta del otro lado y aquí también quiero que crezca poquito, así perfecto. Y ahora en este preview, como observas, si el botón crece o se estira horizontalmente, ya no se deforma, ya se ve

algo bien. Si el botón crece en la parte de la anchura, aquí mas bien es más bien horizontalmente y en el otro es verticalmente. Si el botón crece horizontalmente tampoco se deforma y entonces podemos incluir si quieres unos iconos aquí a dentro o un texto, etc.

Bien ya hemos terminado nuestro archivo 9-patch, ahora lo que sigue es que simplemente vamos a guardarlo. Voy a ir al menu file y le voy a decir save 9-patch.

Y le puedo poner botón, lo guardo y ahora lo que me ha generado es mi botón pero con la extensión .9.png. Y si observas este no es un botón como el que teníamos acá, es un botón donde tiene unas rayitas, en las partes, en las esquinas que delimitan las zonas que va a crecer el botón o que se va a reducir.

Simplemente este botón vamos y lo colocamos como un recurso en nuestra composición del proyecto.

Soportes múltiples

Soporte a múltiples pantallas

Vamos a ver ahora cómo generar un soporte en nuestra aplicación para múltiples pantallas. Esto lo vamos a lograr teniendo las imágenes adecuadas. Y además, recuerda que Android es un sistema operativo que, como ya dijimos, ha estado creciendo mucho y podemos encontrar Android en infinidad de dispositivos e infinidad de tamaños de pantallas. Por lo tanto, ha sido un poco complicado manejar nuestras interfaces en ciertas dimensiones o en ciertos tamaños de pantalla. Y seguramente alguna vez te has topado con una aplicación móvil en la cual ves las imágenes o ves los recursos que tiene la aplicación. Y de pronto se ven pixeladas o se ven borrosas o no tienen la resolución adecuada.

Android ha colocado un sistema para poder manejar esto a múltiples pantallas, que tu aplicación móvil se vea muy bien en cualquier pantalla que la veas. ya sea en un teléfono con una pantalla muy pequeña o en una tablet, una computadora, o en un Android TV, que tenga una pantalla muy, muy grande.

¿Cómo lo hizo? Bueno, fue complicado al principio definir que Android manejaba solamente una dimensión de pantalla, por lo tanto, Android empezó a controlar el tamaño de la pantalla y a demás las densidades. Para Android, Android manejó cuatro tamaños de pantalla principalmente. Para manejar todo esto de las densidades y de los tamaños de pantalla.

Si nosotros diseñamos una interfaz, entonces necesitaremos generar la misma interfaz para los cuatro diferentes densidades y diferentes tamaños que tenemos para Android.

Los 4 tamaños generales son:

- Pequeño
- Normal
- Grande (Large)
- Extragrande (Xlarge)

Los 4 tipos de densidades que android maneja son:

- Low (ldpi)
- Medium (mdpi)
- High (hdpi)
- ExtraHigh (xhdpi)

Y los directorios son:

```
MyProject/  
  res/  
    drawable-xhdpi/  
      awesomeimage.png  
    drawable-hdpi/  
      awesomeimage.png  
    drawable-mdpi/  
      awesomeimage.png  
    drawable-ldpi/  
      awesomeimage.png
```

Estas cuatro interfaces debemos ubicarlas en los directorios correspondientes, que más adelante veremos en un ejemplo. Nosotros al tener esto, al tener una imagen específica para cada directorio como se está mostrando aquí. De esta forma si lo categorizamos así, Android sabrá al momento de correr nuestra aplicación. Android sabe en qué tipo de pantalla estás corriendo la aplicación. Por lo tanto, sabe la resolución correcta que debes tomar para poder mostrar tu aplicación. Así tu aplicación nunca se va a pixelar, porque tienes todas las densidades, tienes el soporte para todas las densidades y todos los tamaños de pantalla.



¿Cómo logramos generar diferentes tamaños de interfaces para cada tamaño de pantalla?. Bueno, lo haremos de la siguiente forma.

Como se observa aquí, tomaremos como medida base el 100% que es una pantalla normal a medium density.

Posteriormente tenemos low density, de ese 100% que ya creamos nuestro lienzo, nuestro diseño, debemos tomar para low density el 75%. Después, si queremos manejar una high density, entonces debemos manejar del medium el 150 por ciento. Y para extra high density el 200 por ciento. Así sucesivamente deben ir creciendo cada interfaz que tú generes, para así poder ubicar cada elemento en su respectivo directorio.

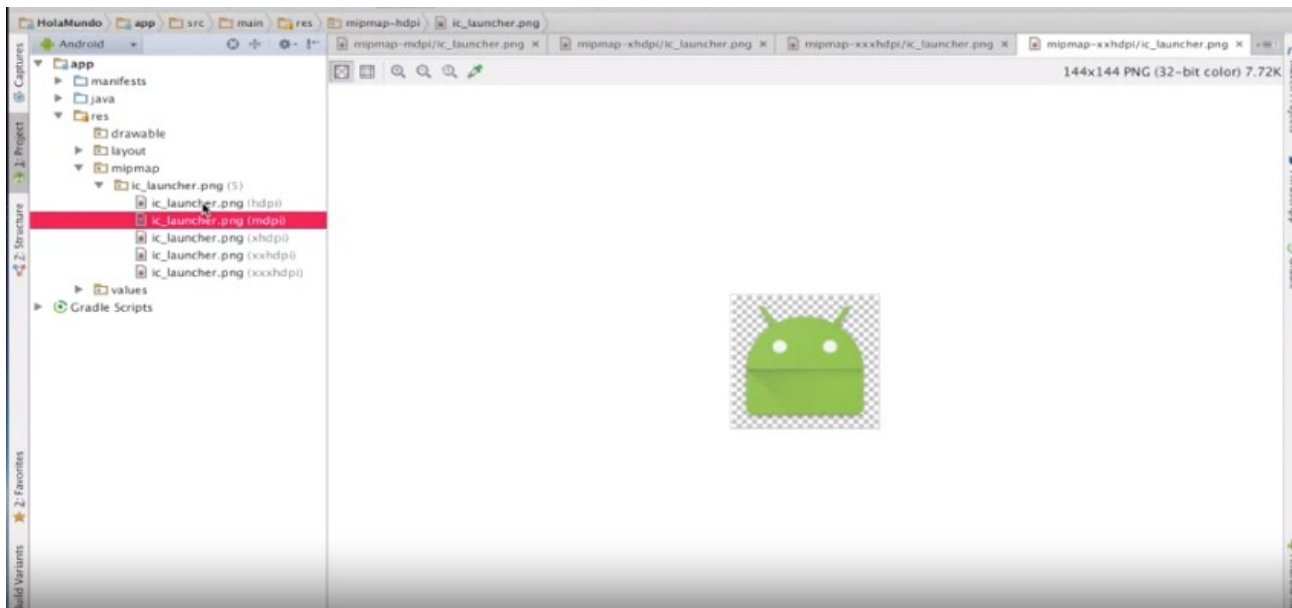
```
MyProject/  
  res/  
    layout/           # default (portrait)  
      main.xml  
    layout-land/      # landscape  
      main.xml  
    layout-large/     # large (portrait)  
      main.xml  
    layout-large-land/ # large landscape  
      main.xml
```

Drawable

Bien, en el segmento anterior hablamos que si querías tener una interfaz deberías generarla en diferentes tamaños y así mismo esa interfaz debes almacenarla en su respectivo directorio. El directorio que le corresponde a todas las imágenes que manejaremos en nuestra aplicación es el directorio drawable.

El directorio drawable, se va encargar de manejar todas las imágenes en sus respectivas densidades. Tenemos otro directorio que también almacena imágenes, pero este directorio es más exclusivo para iconos, el directorio se llama mipmap.

Como observas, en mip map tenemos ya también una imagen que se llama ic_launcher.png. Si abrimos ic_launcher.png alcanzamos a ver que tenemos una imagen y que a un lado dice hdpi o high density. Luego tenemos otra imagen medium dpi y como observas, es la misma imagen y además la imagen tiene el mismo nombre en todas sus versiones.

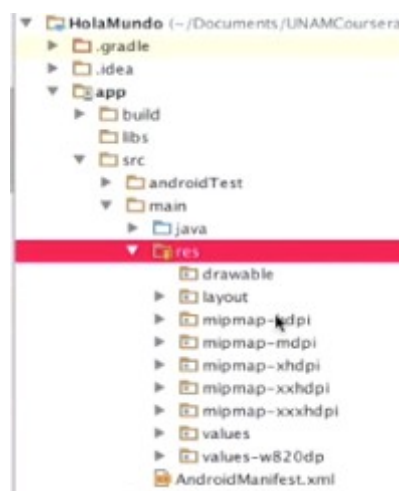


Ésto es precisamente lo que nosotros queremos lograr, lo que se observa aquí es un ícono generado en diferentes densidades. Al colocar nuestro icono en la carpeta mipmap, en diferentes densidades, cuando nuestra aplicación se instale, Android automáticamente vas a ver qué imagen debe tomar dependiendo del tamaño de la pantalla.

Nosotros haremos lo mismo para el directorio drawable. Bien, ahora, no se observa mucho en mipmap, la forma en que están separados los directorios en low density, medium density, etc.

Para eso voy a cambiar la forma en como estoy visualizando este sistema de archivos. Digamos que este sistema de archivos está siendo visualizado para un desarrollador Android, O la forma más directa de poder tener nuestros documentos organizados.

Tenemos otra vista, es esta vista de “proyecto”, vamos a ir ahí. Y esta vista de proyecto me va a permitir observar la composición de mis archivos como realmente existen en mi disco duro. Voy a abrir la carpeta app, posteriormeten src, y en la carpeta main, ahí está todo mi proyecto.



Tenemos la carpeta res, vamos a entrar ahí, y como observas está ahí el directorio drawable, solamente está ahí como único directorio. No se ha generado ningún derivado de ese directorio que tenga high density, medium density, etc.

De quien sí podemos ver que tiene esa composición es del directorio mip map, y como se puede observar, cada directorio mip map tiene la imagen específica que abrimos hace un momento. Para un medium density tiene una imagen, un icono de medium density, y lo que nosotros queremos hacer es lograr esta estructura como se ve aquí, también para el directorio drawable.

¿Esto para qué?, para que podamos almacenar aparte de iconos, como en mipmap, podemos almacenar también imágenes o recursos que vaya a ocupar en tu interfaz. Bien, esto podrías hacerlo en un archivo de photoshop o en illustrator, lo podrías dárselo a tu diseñador para que lo maqueten

Voy a tener como referencia medium density, entonces, si yo decido que mi imagen tenga 200 píxeles, bueno, en medium density recuerda que es el 100%. Va corresponder al 200dp. Entonces si yo tengo una imagen de 200 píxeles, y quiero generar una imagen extra high density, bueno esa imagen debe tener 400 píxeles.

Recuerda que es el 200%, si quiero en low density, esa imagen debe tener 150 píxeles, recuerda, es el 75%. Y así sucesivamente, este convertidor nos va a ayudar a poder estar manejando el pixelaje y las densidades adecuadamente. Bien, esto podría ser un poco tedioso, manejar esto, aunque hay algunos que les gusta mucho hacerlo así.

Hay otro recurso y es precisamente utilizando un plugin, un plugin que el mismo Android Studio nos proporciona. Cambiaré nuestra vista aquí de project a Android y para entrar a la ventana de plugins iré en el menú de Mac Android en preferences, o si estás en windows debes ir al menú → files → settings.

Si lo tienes instalado ya saldrá aquí, pero seguramente no lo tendrás instalado. Entonces debes dar clic en el botón que está aquí en medio, que dice browse repositories y aquí vamos a buscar Android Drawable Importer.

Bueno, te va pedir este botón verde, install plugin. Lo vamos a instalar, te va solicitar que reinicies tu id, y automáticamente ya vamos a tener nuestro plugin instado, y listo para usar y entonces generar nuestras imágenes en muchas densidades.

Una vez que hayas reiniciado tu id, vamos a comenzar a importar las imágenes. Voy a dar clic derecho, no importa, puede ser a cualquier nivel del proyecto, en cualquier carpeta. Clic derecho, New, y gracias a nuestro plugin ahora tenemos estas tres opciones. “Icon Pack Drawable Importer”, “Batch Drawable Import”, “Multisource-Drawable” y la opción que estaremos buscando es “Batch Drawable Import”.

Estando ahí en Bat Drawable Import, voy a buscar la imagen que quiero.

Esta imagen, tu debes indicar cuál es la resolución o cuál es la densidad que está ocupando esta imagen. Si decides que esa imagen que estás importando es la densidad media, entonces que es que a partir de esa, recuerda nuestro 100%, lo que hará este plugin, a partir de esta medium density, va generar las demás densidades automáticamente.

Sí, le voy a dar ok y nuevamente ok. Y lo que está haciendo en este momento import image es nos ha generado, en esta carpeta Drawable, tenemos todas las versiones todas las densidades de "mi imagen", que era nuestra imagen principal. Y como observas nos lo ha acomodado en una carpetita, que esto es una vista amigable para nosotros los desarrolladores. Podemos ir a la vista de project para ver realmente cómo quedó el proyecto, y observamos que ahí está drawable high dpi, drawable ldpi, mdpi, y en cada carpeta está la imagen que corresponde de la densidad correcta.

Bien, recuerda, todos nuestros recursos de imágenes deben estar en la carpeta Drawable y deben estar almacenados en su respectiva densidad. Con esto, podemos dar soporte a múltiples pantallas en cuestión de gráficos.

Además de los tamaños de pantalla, también debemos tomar en consideración en qué orientación se está viendo nuestra aplicación. Es decir si está horizontal o si está vertical, en portrait o en landscape.

Esto nos va a ayudar a generar una mejor experiencia de usuario si nosotros redibujamos la interfaz. Si de pronto la aplicación se ve de una forma así, en portrait y se ve muy bien. Pero si de pronto la estás viendo en landscape probablemente los elementos no se vean de una manera muy bonita si siguen, a lo mejor apilados como estaban en portrait. Sería muy bueno que pudieras redibujar la aplicación, si estuvieras ahora en formato landscape, esto va a generar una muy buena experiencia para tus usuarios cuando estén navegando en tus apps.

Nota: Para descargar Android Drawable y quede usablese recomienda seguir éste instructivo <https://www.youtube.com/watch?v=AoA37fOUKeU>

Layouts y archivo strings.xml

Layouts

Los lienzos en donde dibujamos nuestras pantallas se llaman layouts. Y al mismo tiempo si queremos controlar o manejar un poco la orientación de un lienzo o de un layout de la misma forma debemos manejar estos en diferentes carpetas, en diferentes directorios. Tal cual como se está observando en la imagen.

```
MyProject/
  res/
    layout/                # default (portrait)
      main.xml
    layout-land/           # landscape
      main.xml

    layout-large/          # large (portrait)
      main.xml
```



```
layout-large-land/    # large landscape
    main.xml
```

Podemos manejar un layout en landscape con la instrucción `-land`. Un layout, a lo mejor para una pantalla mucho más grande, que tienes más espacio con la instrucción `-large`. Y además otro layout, si supongamos que tienes una pantalla muy grande pero además también está en landscape con la instrucción `large-land`. De esta forma podemos ir manejando también la distribución de las pantallas y las orientaciones.

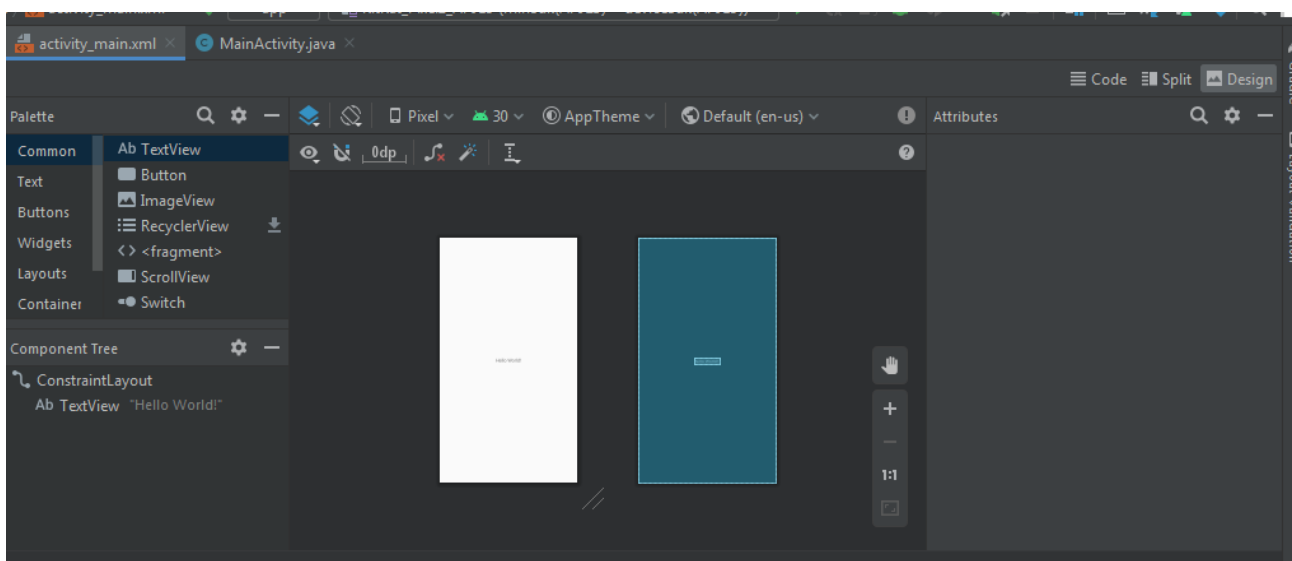
Bien, ahora vamos con uno de nuestros siguientes recursos, que es la carpeta `layout`. La carpeta `layout`, contiene nuestros archivos, nuestros llamados layouts que serán nuestras interfaces de usuario cuando tengamos las apps.

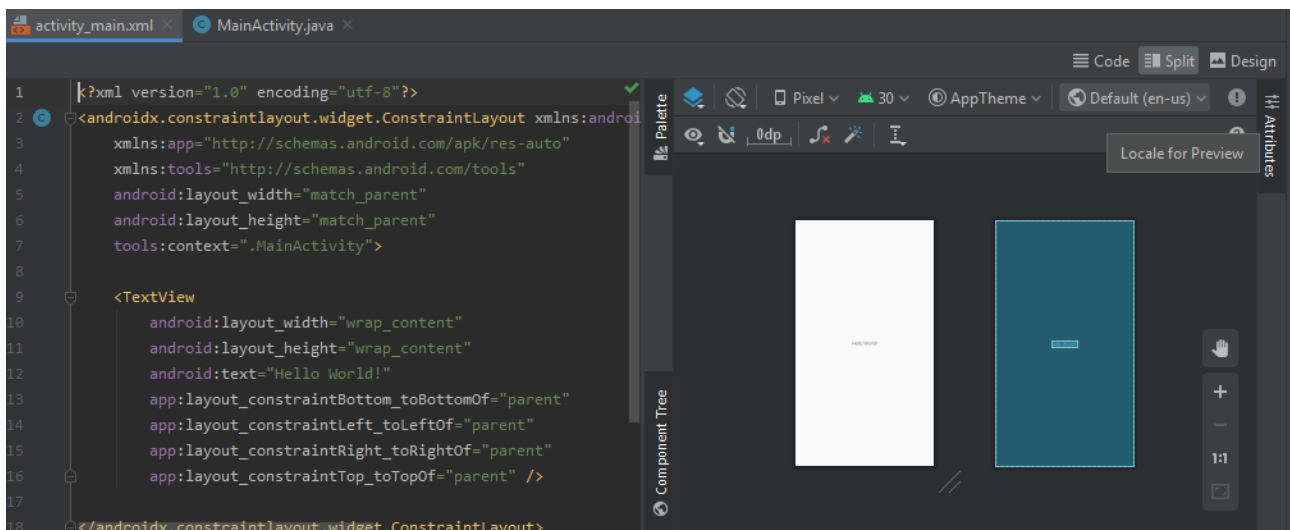
Estos archivos layouts son archivos escritos en código XML y tienen la particularidad que para ser nombrados deben ser siempre en minúsculas. Y además deben estar separados por guiones bajos. Cada palabra debe estar separada por guiones bajos, no se permite el uso de mayúsculas para estos layouts.

Tampoco se permite que un layout comience con un número, siempre debe comenzar con la letra y debe ser en minúsculas.

Bien, tenemos nuestro layout que se llama `activity_main` que es el layout que nos crea por default el ID cuando hacemos un `Hola Mundo`.

Y como observas pues en esta parte superior tenemos 3 vistas para ese layout. Uno que es la vista de diseño que es en la que ahorita me encuentro otra es la vista de texto (code) y la tercera es split que muestra a la izquierda el código xml y a la derecha el diseño.





Si tu cambias a la vista del texto vas a observar que tenemos nuestro código xml. El código xml como tal no es un lenguaje de programación. Debes saber que el lenguaje xml es un lenguaje de esquematización. Tan solo nos ayuda a presentar la información, a esquematizar la información, no es de programación.

Y el código XML se rige a través de etiquetas padre y etiquetas hijo. Las etiquetas hijo son las que están contenidas dentro de las etiquetas padre. Y entonces cada etiqueta XML va a tener propiedades. O si pudiéramos verlo en términos de HTML y CSS pues podríamos decir que cada etiqueta va a tener estilos. Entonces cada etiqueta XML tiene propiedades, esas propiedades le ayudan a dar un estilo específico a esa etiqueta en particular. Y al mismo tiempo como en HTML también existen etiquetas hijo. Por ejemplo, aquí podemos ver claramente que tenemos nuestra etiqueta `androidx.constraintlayout.widget.ConstraintLayout`, que contienen todas estas propiedades:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
```

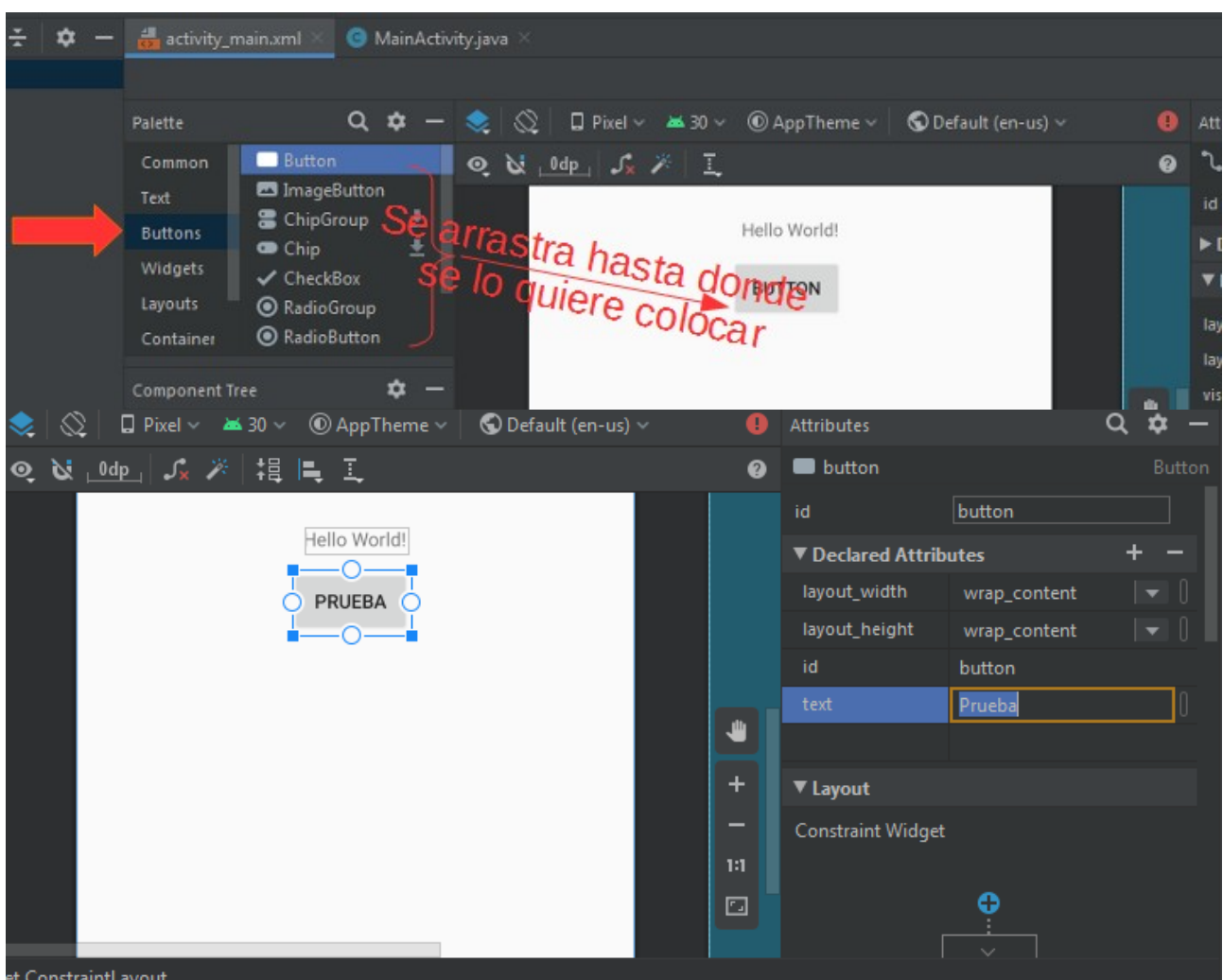
```
app:layout_constraintLeft_toLeftOf="parent"  
app:layout_constraintRight_toRightOf="parent"  
app:layout_constraintTop_toTopOf="parent" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

La etiqueta abre aquí y la etiqueta cierra aquí. Nuestra etiqueta hijo es esta etiqueta que se llama TextView. La etiqueta abre aquí y cierra aquí. La etiqueta TextView tiene estas propiedades y pudiste observar que ambas etiquetas, tanto esta como esta. Cierran de diferente modo a pesar de que son etiquetas xml.

Bien, esta etiqueta `androidx.constraintlayout.widget.ConstraintLayout` cierra así porque nos indica que es una etiqueta padre. Entonces se queda a la espera de colocar más hijos dentro de nuestro archivo. Esta etiqueta cierra de esta forma porque nos indica que es una única etiqueta, es decir, no va a contener etiquetas hijo, por lo tanto se cierra y punto.

Bien, si regresamos a la vista de diseño, tenemos del lado derecho todos nuestros elementos que podemos colocar en nuestro layout. Por ejemplo, voy a colocar ahora un botón y lo voy a colocar aquí. Para eso se arrastra el botón a donde quiero colocarlo.



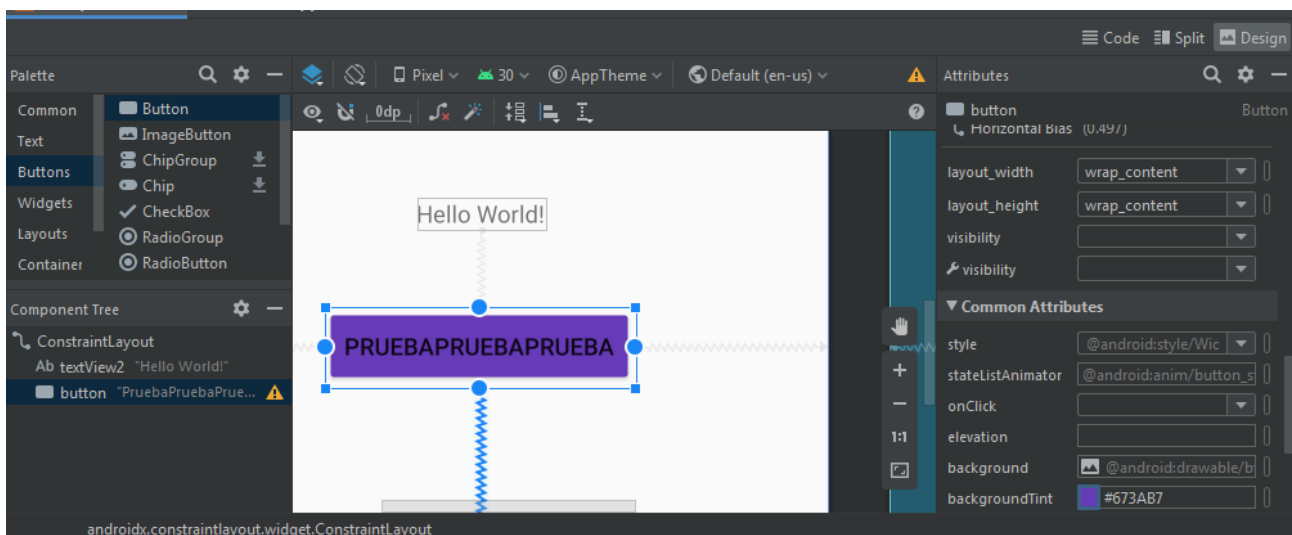
Y le cambio el texto que dice el boton en “Declare Atributes”

Voy a tratar de centrarlo un poco. Listo, ahí está mi botón, ahora voy a colocar un elemento, ImageView, como observas es sumamente sencillo insertar elementos en la vista de diseño. Si yo selecciono mi botón, en la parte derecha voy a tener una ventanita de propiedades, y en la parte superior aquí me va diciendo cómo va.

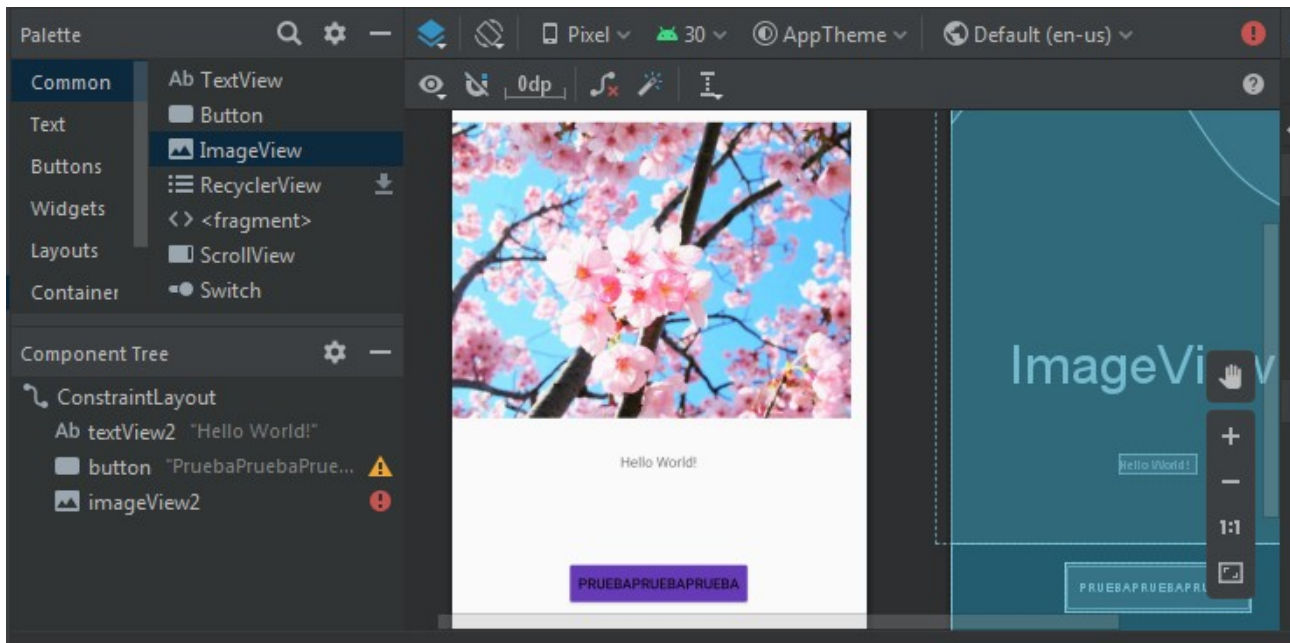
Nota: Para alinear cosas y que quede centrado tanto en portrait como en landscape, te parás sobre el elemento, clic derecho → organize y ahí darle la orientación que uno quiera.

Tenemos también el control de nuestras etiquetas, este control de etiquetas nos está diciendo la jerarquía que ocupa cada etiqueta. ¿Recuerdas nuestras etiquetas padre e hijo? Aquí puedo observar que la etiqueta padre es `androidx.constraintlayout.widget.ConstraintLayout` y que contiene una etiqueta `TextView` y ahora hemos agregado una etiqueta `Botón` y además una etiqueta `ImageButton`. Voy a quitar este `ImageButton` y voy a buscar la etiqueta `ImageView`, que esa es la que realmente quiero. Coloco aquí mi `ImageView` y ahí está, ahora sí tenemos un `ImageView`.

Para seleccionarle un color al botón, lo modifico en atributos.



Un recurso de color o una imagen con la que quiero mostrar. Vamos a seleccionar nuestra imagen que habíamos puesto.

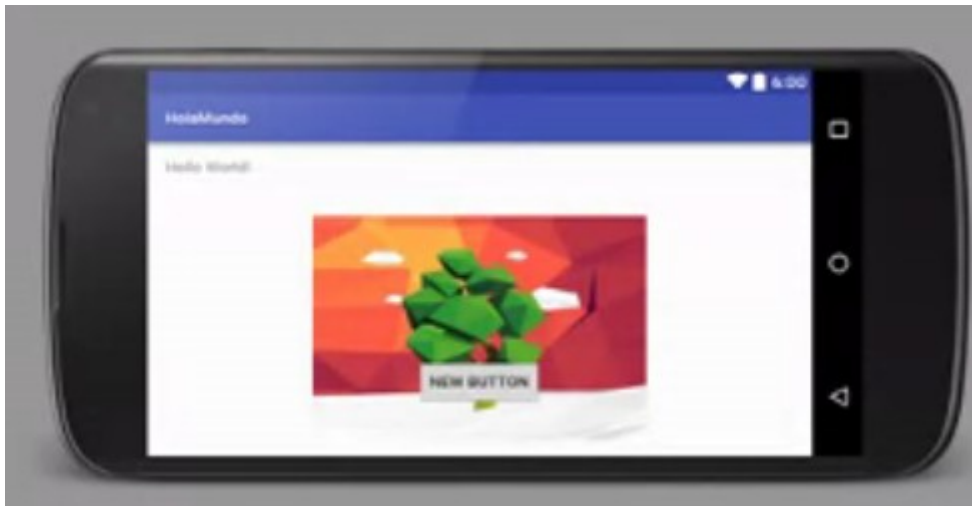


Me dirás "solo aparece una imagen". Bueno, esto es lo que hace la magia. Android lo ve como si fuese una sola imagen, a pesar de que la imagen está ubicada en diferentes directorios en sus respectivas densidades. Entonces no te preocupes, todo este proceso lo va a hacer Android internamente. Selecciono mi árbol y listo, ya tengo la imagen ahí incrustada. Muy bien, voy a cambiar de lugar y voy a poner ahora el botón por acá.

Una de las desventajas de estar trabajando con el editor gráfico. Muchas veces, de pronto ya no nos permite hacer cosas porque las propiedades de los elementos se quedaron como enjauladas.

Ahora, vamos a generar una vista para landscape. Esta vista que tenemos es portrait. Voy a salvar mi archivo, mi layout, Activity main. Y ahora, en esta parte superior donde está aquí esta hojita con Android.

Es una de las ventajas de utilizar Android Studio, nos va a hacer todo el trabajo. Yo puedo colocar aquí, un nuevo layout con una variación en landscape. Entonces selecciono eso y tan solo con un click me ha generado una nueva vista, pero ahora en landscape. Si observamos de este lado, vemos que como que se ha unificado el layout, como en cuestión de las imágenes. Pero si vemos la vista de proyecto, podemos ver que ahora tenemos nuestro layout-land y ahí adentro está definido el layout que va a responder cuando nosotros coloquemos nuestro teléfono en landscape. Ahora, ¿observas que nuestro botón se está encimando?.



Esto es algo muy común que ocurre, cuando manejamos interfaces primero en portrait y luego lo ponemos en landscape. Vamos a redibujar esto.

Vamos a hacer que esta imagen este por aquí, entonces nuestro botón mejor se reacomode de este lado. Voy a colocar, iré a la vista del texto para manejar mejor esto. Y vamos a quitarle todas estas propiedades que tiene por aquí. Y ahora sí, voy a meter mi botón por acá.

Ahora sí ya se está acomodando mejor. Voy a hacer la imagen un poco más pequeña. A lo mejor para el fin de este proyecto. Incluso yo podría quitar elementos. Observamos que el TextView ya no está, ya no se ve. Lo podría quitar, entonces esto me facilitaría mi interfaz. Si vemos ahora en portrait, nuestro layout se ve así y en landscape se ve así. Cuando corramos nuestro proyecto vamos a ver que esto ocurre realmente. Los elementos están acomodados de diferente manera y además uno de los elementos ya no está presente, nuestro TextView.



Esta es una forma de manejar el landscape, ahora, si quiero, puedo manejar para una extra large. Una ventana extra large. Puedo darle click ahí y ahora nos ha generado un extra large. ¿Observas que nuestro botón se fue hasta el fondo?

Vamos a acomodarlo para un extra large, que más o menos será para una tablet. Entonces estamos rediseñando nuestra interfaz. Esto, como te recuerdo, causa una buena experiencia de usuario. Puedo también generar a partir de aquí, un layout extralarge, pero en landscape.

Acá está el de landscape, vamos a ponerle aquí en create other, orientation, le vamos a poner landscape.

Soporte multi idioma

Ahora si queremos hablar sobre darle soporte multi idioma a nuestra aplicación móvil. Vamos a hacer algo similar a lo que hemos venido haciendo y tenemos que partir de que, bueno, primeramente tengo que hablarte de un archivo muy especial que es el archivo Strings que vas a encontrar en tu carpeta Values.

```
MyProject/  
  res/  
    values/  
      strings.xml  
    values-es/  
      strings.xml  
    values-fr/  
      strings.xml
```

Y la idea de esto es que en tu archivo Strings concentres todos los textos que contiene tu aplicación, todos, absolutamente todos. Desde títulos, textos, indicadores, incluso hasta los textos que tienen tus botones, todos vamos a concentrarlos en nuestro archivo Strings.

Ahora en el ejemplo vamos a ver cómo vamos a estar manejando este archivo Strings. Y posteriormente lo que tenemos que hacer es generar, y con ayuda de una herramienta que ahí mismo Android estudió nos proporciona. Podemos generar diferentes versiones de archivo strings dependiendo del idioma y del código del país que queramos manejar. Como observas en la imagen, esta es la manera en cómo debemos manejar nuestra distribución de archivos Strings en diferentes idiomas.

Soporte a múltiples idiomas y archivo r.java

Hemos manejado ya las imágenes, hemos manejado ya diferentes orientaciones de pantalla, incluso hasta diferentes tamaños de pantalla y ahora vamos a manejar idiomas.

El archivo strings es el que se encarga de concentrar todos los textos de nuestra aplicación, es el archivo strings.

El archivo strings lo vas a encontrar en la carpeta values, que está también contenida en la carpeta res.

Bien, cuando nosotros abrimos nuestro archivo strings, como observas y como todos los recursos de Android, es un archivo XML. Un archivo XML está también ubicado a través de etiquetas, como lo hemos venido manejando. Estas etiquetas también son etiquetas que tienen un identificador y además también un valor.

Yo comúnmente les llamo, nombre de la variable, valor de la variable.

```
<resources>  
  <string name="app_name">HolaMundo</string>  
</resources>
```


Como se observa aquí, tenemos nombre de la variable `app_name`, y valor de la variable `HolaMundo`. Bien, si nosotros queremos almacenar o generar una nueva variable, simplemente colocamos string.

Y ponemos en el nombre de la variable, podemos poner título app. Y el título de la aplicación le puedo poner Mi Hola Mundo. El título app dice Mi Hola Mundo.

```
<resources>
  <string name="app_name">HolaMundo</string>
  <string name="titulo_app">Mi Hola Mundo</string>
</resources>
```

Muy bien, entonces, esta variable, automáticamente se ha creado y lo que sucede es que Android utiliza un archivo muy especial que es el archivo R.

Archivo R

El archivo R, maneja todas las referencias que nosotros dejamos en nuestros recursos. Es decir, nuestro archivo R es un archivo de código java, es una clase de java. Que cuando nosotros creamos un recurso en XML y también un recurso que también puede ser una imagen. Un layout, una variable string o un archivo, etc.

Todos esos recursos, automáticamente se van mapeando en la memoria, se van creando en memoria. Entonces lo que hace el archivo R, es que toma la dirección en memoria de ese recurso que se ha creado y lo mapea en su archivo `r.java`.

Entonces, todos los recursos de nuestra aplicación se estarán mapeando en el archivo `r.java`.

Ese archivo es muy importante, y es muy importante que si lo llegas a encontrar dentro de tu proyecto, porque créeme está un poco escondido.

Si lo llegas a encontrar no lo toques. `R.java` no se toca, no se borra nada, porque es muy peligroso pues puedes dañar tu proyecto. Y entonces si llegas a dañar el proyecto, llegas a dañar el archivo R tendrás que generarlo nuevamente.

Bien, en este momento que he creado mi variable string, que dice Mi Hola Mundo, se ha creado una variable que se llama `titulo_app`. Y se ha referenciado también en el archivo R.

Acceso al los XMLs correspondientes

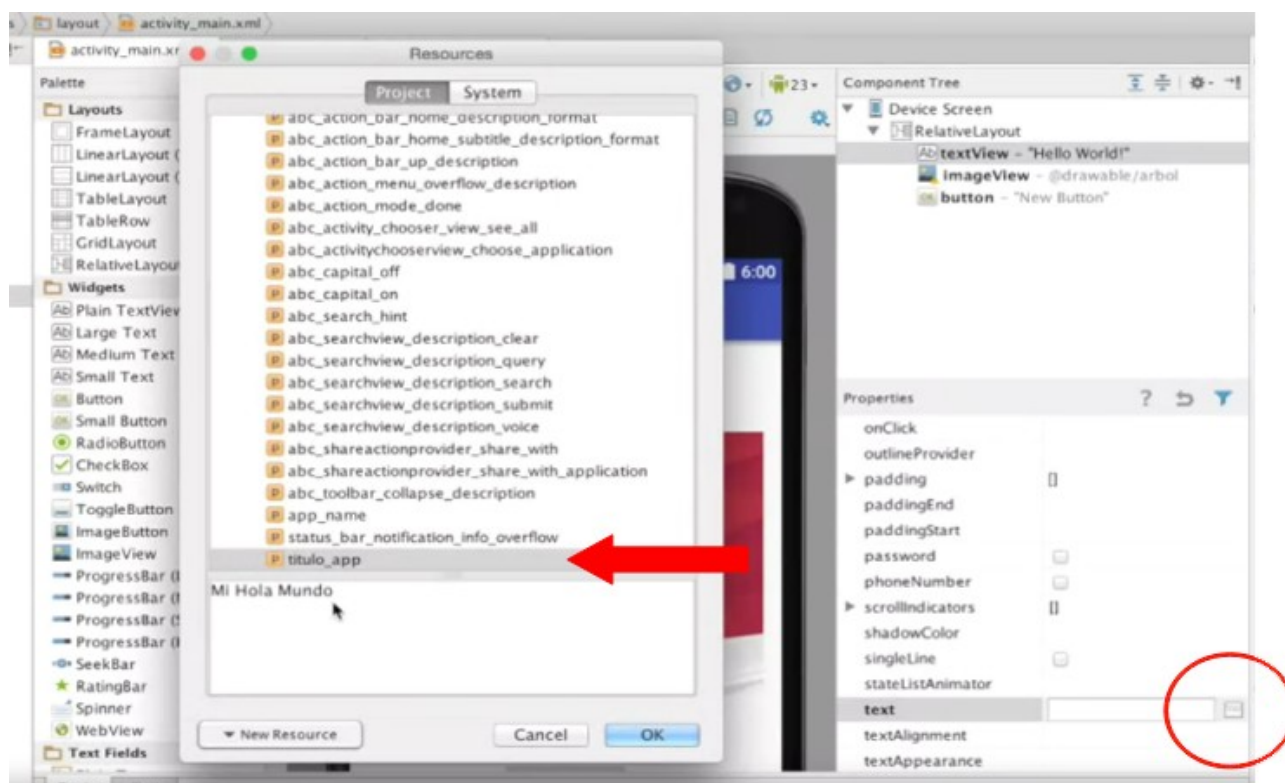
Bien, ¿cómo puedo acceder a este texto, desde un archivo XML a otro archivo XML? Bueno, voy a ir a mi layout. Y en mi layout, te acuerdas que tengo un text view, ese text view dice hello world.

Si voy a la propiedad text, de este text view. Si voy a propiedad text, que están ordenadas alfabéticamente, observas que tiene el texto tal cual escrito ahí.

No está haciendo referencia a ninguna variable, eso está muy mal. Porque si nosotros queremos generar un archivo en diferentes idiomas, necesitamos tener referenciado este texto en una variable.

Entonces lo que vamos hacer es que lo voy a quitar, y voy a referenciarlo a mi variable que acabo de crear.

Si observas aquí está string, que es uno de los recursos, y aquí dice mi titulo_app. Mi titulo_app y aquí contiene un preview de el texto.



Al mismo tiempo puedo generar aquí un nuevo recurso si es que lo quiero.

Perfecto, entonces, siempre que yo quiera acceder a un recurso, lo voy hacer a través de arroba, seguido del nombre del archivo.

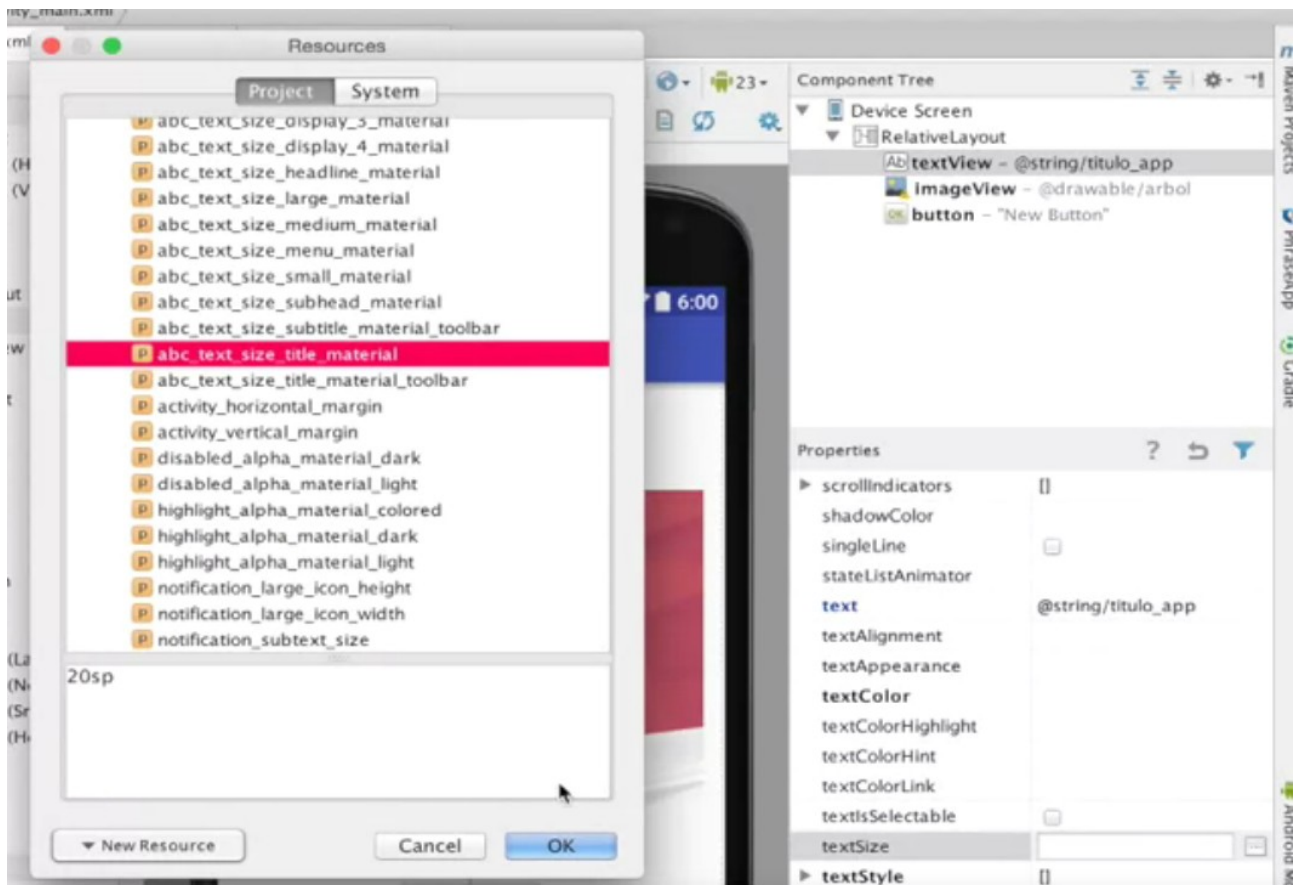


En este caso @string, en singular. Recuerda que nuestro archivos se llama strings.

.xml. Si yo acceso @string, estoy accesorando a sólo un recurso contenido en el archivo strings. Entonces @string/ nombre de la variable, tal cual como se está observando aquí. En este caso el nombre de mi variable es titulo_app.

Entonces esto de esta forma, me estoy trayendo el valor del texto y como observas ahora aquí dice Mi Hola Mundo.

Vamos a darle un poquito de formato, vamos a centrar esto, a lo mejor vamos a poner el texto un poco más grande y vamos a decirle con un textSize. A lo mejor por aquí tenemos alguna dimensión, que ahorita vamos a ir para allá hablando del archivo de dimensiones.



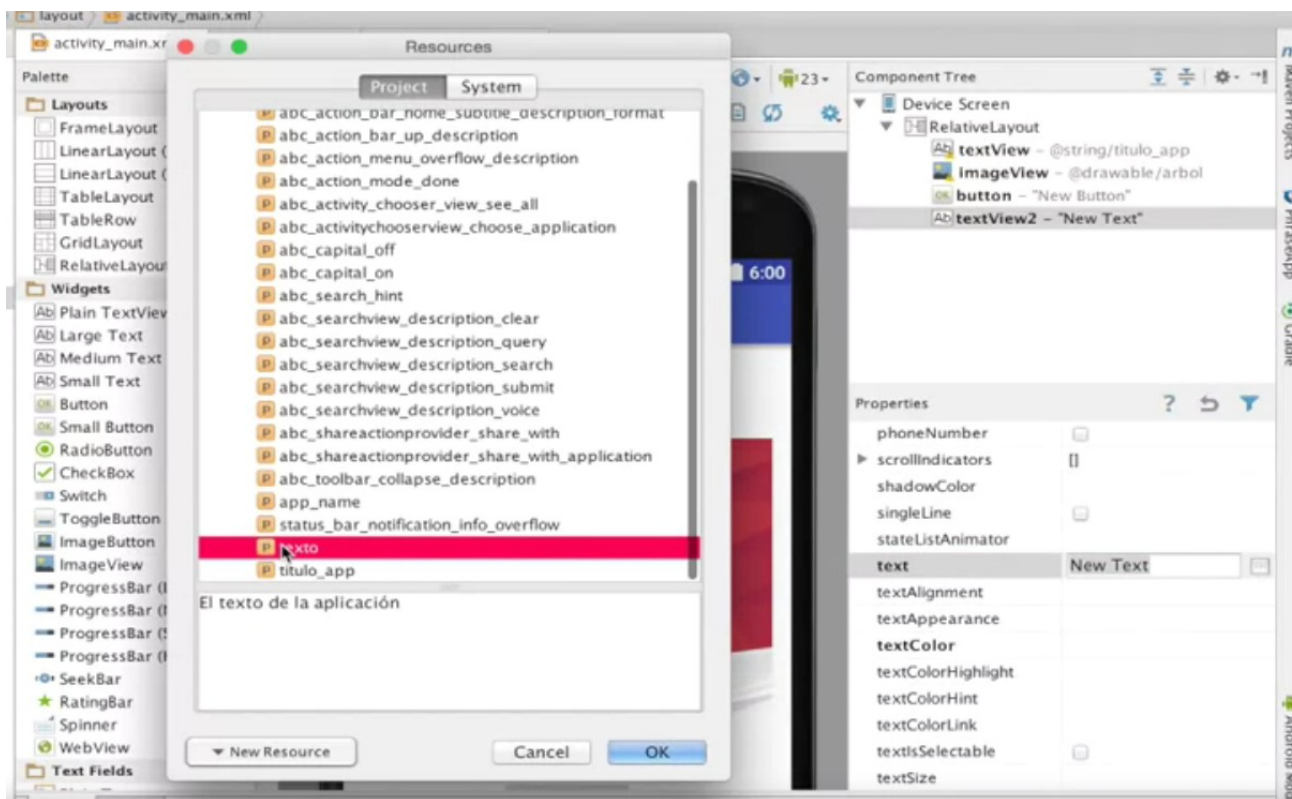
Entonces textSize también está accedendo a un recurso, un recurso que se llama @dimen. Entonces @dimen es otro recurso que se encarga de almacenar las dimensiones.



Tenemos Mi Hola Mundo, voy a colocar otro textView debajo de este, por aquí, otro texto, vamos a ponerlo ahí.

Bien, yo tengo otro texto y ese texto vamos a colocarle que diga, vamos a dar de alta otra variable.





Lo selecciono y ya dice el texto de la aplicación.

Bien, esos son, hasta el momento las dos variables o los dos recursos que voy a traducir.

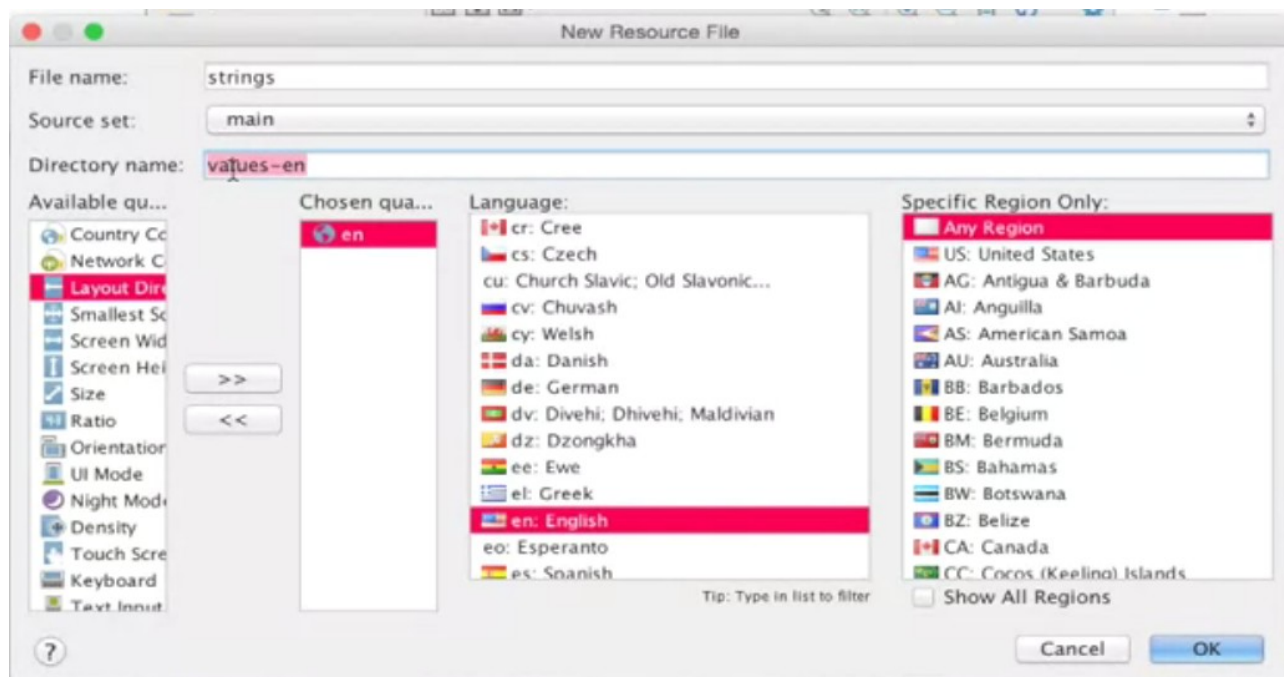
Bien, para generar esto, voy a darle click en valores, click derecho, New y vamos a decirle valores resource file. Y voy a seleccionar el nombre de mi recurso, en este caso va a ser un recurso que se va a llamar strings. Y le voy a poner esta opción Locale.

Lo voy a pasar para acá y lo que va a hacer Locale es que va a comenzar a definir en qué idioma se está manejando estos strings.

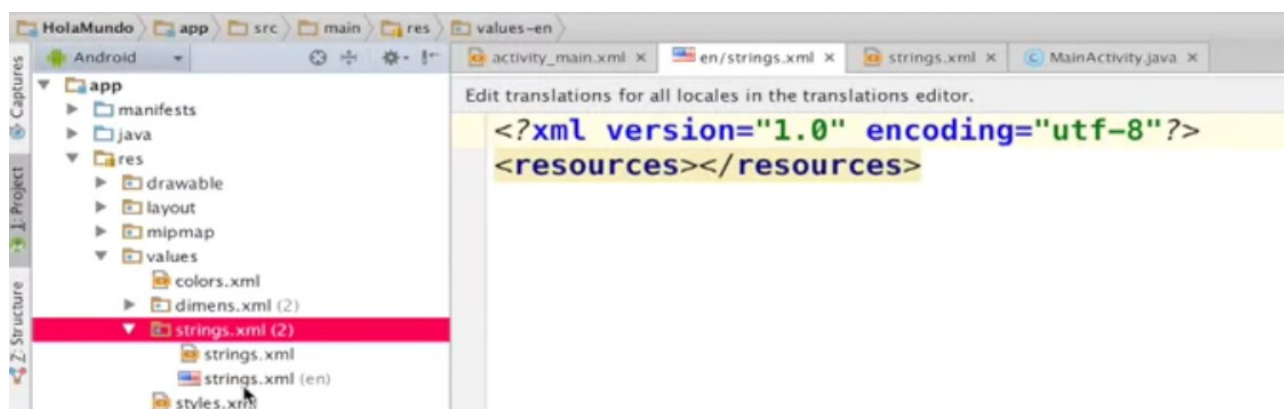
Vamos a decirle que sea en inglés.

Vamos a buscar inglés y debe ser con en English y podemos definir incluso una región de la cual es ese inglés. Puede ser Australia, Bermudas, etc, Canada, etc. O puedes dejarlo así tal cual.

Como observas nos ha colocado un directorio que dice values-en Entonces vamos a darle ok.



Y ahora en values nos ha generado, igualmente, un string unificado.



Un string unificado que nos indica que ésta es la versión de nuestros strings en inglés y esta es la versión de nuestros strings en español.

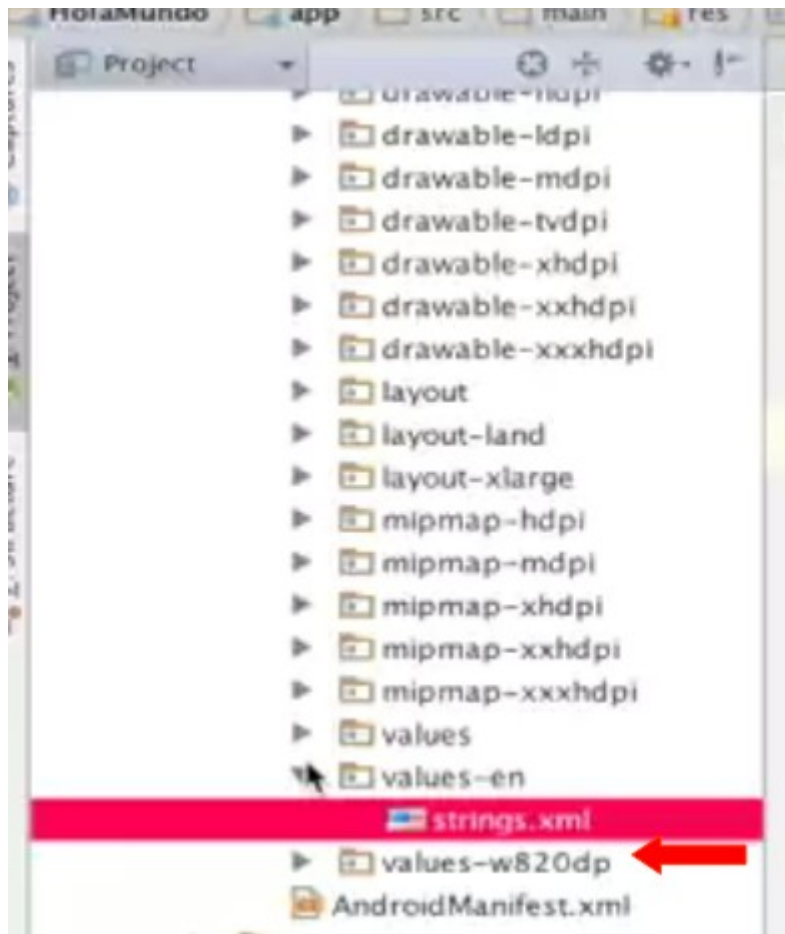
Este que está aquí (el que no está entre paréntesis en la figura), es el texto por default, si tu quieres que tu texto por default sea portugués, pues este texto tendrá que estar en portugués. Si quieres que sea en alemán o si quieres que sea el default en inglés, lo puedes colocar aquí. Y posteriormente hacer su traducción a español.

Bien voy a copiar estas variables y las voy a copiar en las que no están por default y se le pone el valor en el idioma correspondiente. Entonces, ya he dado soporte a múltiples idiomas, cuando, si observas por aquí, tenemos el texto por default.

Cuando mi aplicación sea abierta en un dispositivo cuyo lenguaje sea, por ejemplo inglés. Si en ese dispositivo se ha definido por default el lenguaje inglés y mi aplicación se abre ahí.

Automáticamente se va a tomar el archivo strings en inglés. Si mi dispositivo está en alemán y yo tengo un strings disponible en alemán, igualmente, mi aplicación abrirá el archivo strings alemán.

Entonces observese aquí está unificado el archivo strings, pero si lo vemos en la vista de proyecto, tenemos un archivo values. Un archivo values que contiene un archivo strings por default. Y un archivo values-en que significa english y ahí van a estar todos los strings en inglés.



Un archivo values que contiene un archivo strings por default. Y un archivo values-en que significa english y ahí van a estar todos los strings en inglés.

Igualmente si vas a hacer uno en francés o en alemán, cada uno debe de tener seguido el indicador del idioma que quieres manejar.

Diemens y styles

Bien, nos quedan solamente dos archivos de recursos que no hemos hablado aún. Tenemos, primeramente el archivo, diemens que está contenido también en la carpeta values y el archivo styles.

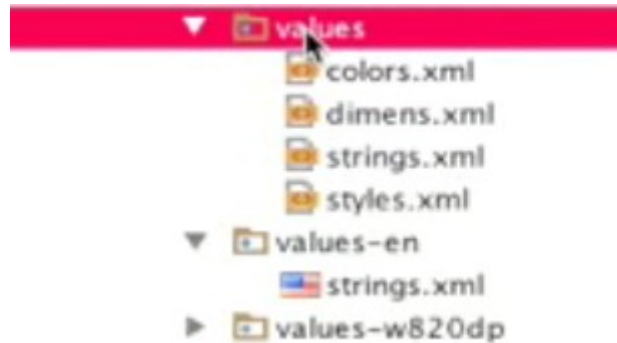
Diemens

Como observas tenemos un diemens aquí por default que está unificado.

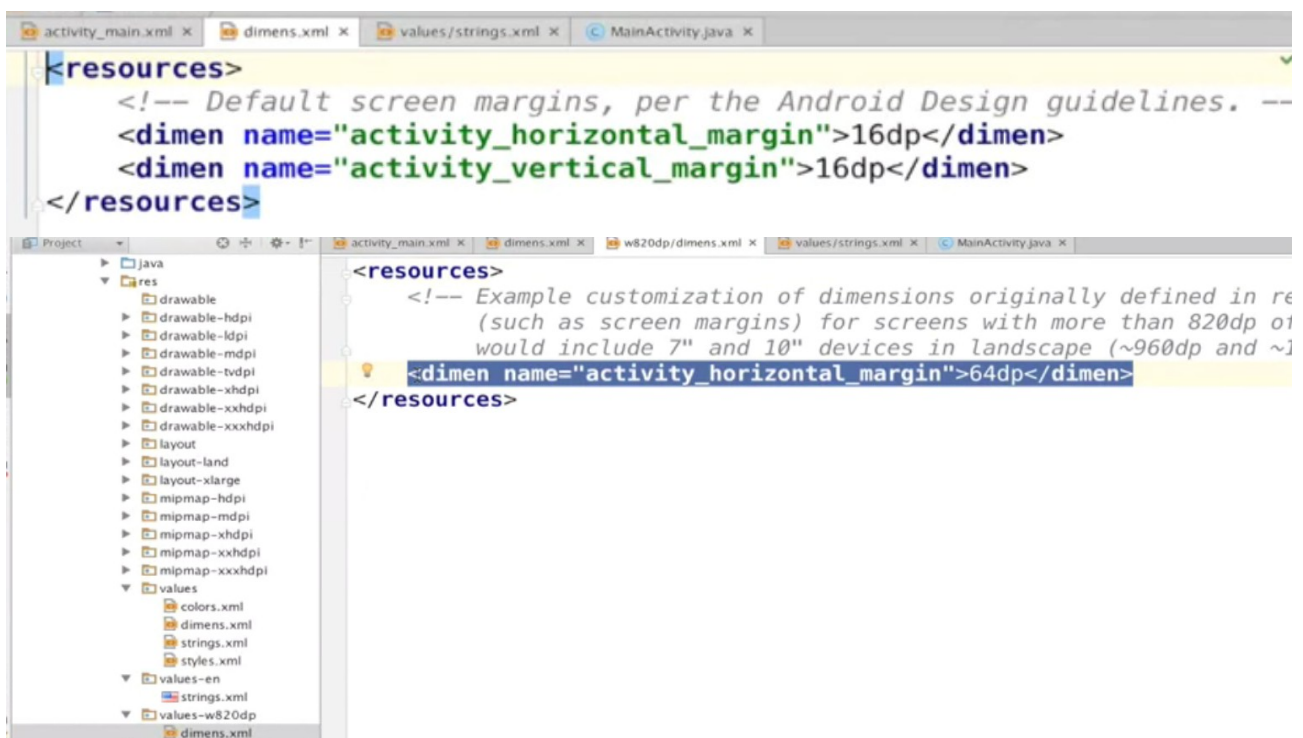
Que si ahora lo vemos en la vista de proyecto, vamos a tener un values, una carpeta values, normal, que contiene su respectivo archivo diemens.

Vamos a abrirlo, ahí está, el archivo diemens también es un archivo xml que contiene etiquetas.

Después tenemos, por acá abajo, otro values que dice values-w820dp. Y bueno esto significa que también son valores para pantallas, w provienen de widescreen. Para pantallas que están en una resolución de 820dps, es decir, una resolución bastante grande.



Si abrimos ese dimens, podemos ver que también tenemos un par, una etiqueta en xml. Vamos a comparar cuál es la diferencia entre uno y otro.



Bien, ambos son archivos de recursos y ambos están definidos a partir del nombre de una variable y el valor de una variable.

El nombre de esta variable se llama activity_horizontal_margin. Su valor es de 16dp.

Posteriormente tenemos otra variable, activity_vertical_margin y su valor es de, también, 16 dp.

Ahora, observemos el otro. El otro contiene la misma variable, activity_horizontal_margin. Pero ésta tiene una variación en la unidad de medida, que es, ahora aquí mide, 64dp. Es decir, es la misma variable pero con diferente valor.

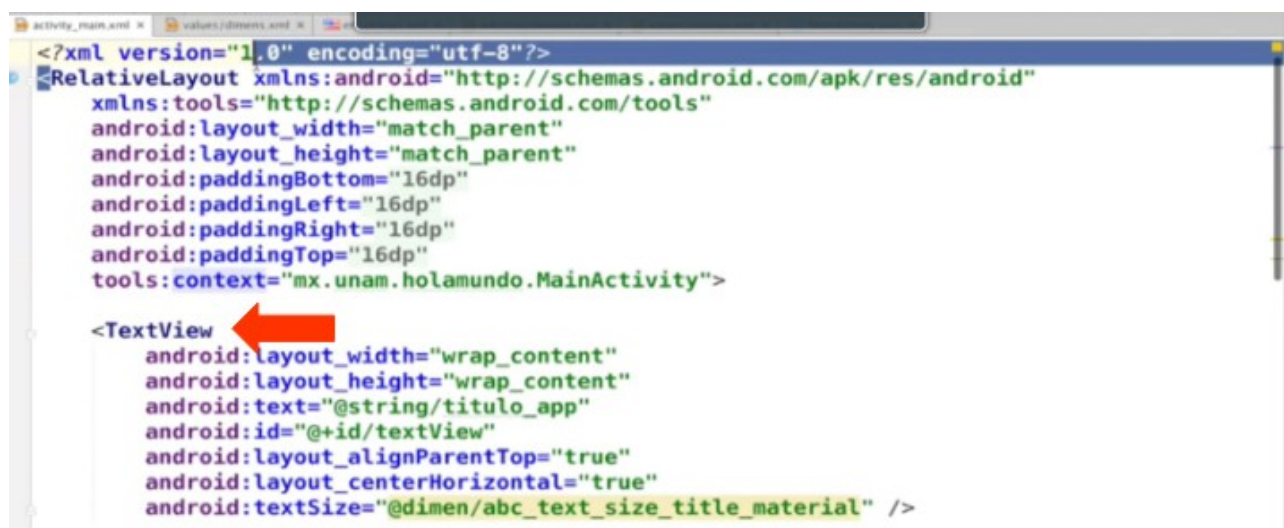
De esta forma es como vamos a manejar todo el soporte, tanto para idiomas, como para imágenes, como para rotaciones de pantallas, etc.

Todos los recursos deben conservar su nombre de origen, su nombre natural. Y si vas a duplicarlo, si vas a generar una nueva versión de ese archivo del cual estás partiendo o de ese recurso del cual estás partiendo.

Bueno, lo único que tiene que cambiar va ser su valor. Pero no puedes cambiar el nombre del recurso como tal.

Como observamos en el archivo anterior de nuestros idiomas, en ambos archivos strings, los nombres de las variables eran exactamente los mismos. Pero obviamente los valores pues eran las traducciones, eran valores diferentes. Para este archivo dimen tenemos una unidad de 64dp y para el otro tenemos 16dps.

Vamos a ver cómo se ve aplicado esto y en qué momento se ve aplicado esto. Iré a mi layout, aquí tengo mi layout y ahora voy a ir a la vista de texto.



En la vista de texto tengo de primera instancia en mi etiqueta relativelayout, y esta etiqueta relativelayout, si observas por aquí, tiene dos pares de paddings, tiene un padding bottom, un paddingleft, right y top.

Ahora, cada uno significa Está teniendo como valor un @dimen_activity_horizontal_margin. En este caso es un paddingleft.

O sea el padding significa es el espacio que hay entre la pantalla, es como la sangría, un espacio que dejas a partir de la pantalla y en ese momento comienzas a escribir lo demás.

Eso es el padding. Entonces estamos diciendo que esta vista tiene un paddingleft, de lo que vale esta variable. Según el archivo dimen, que es de donde lo estamos extrayendo activity_horizontal. Si vemos aquí, se ve un padding pequeño, de 16 dp.

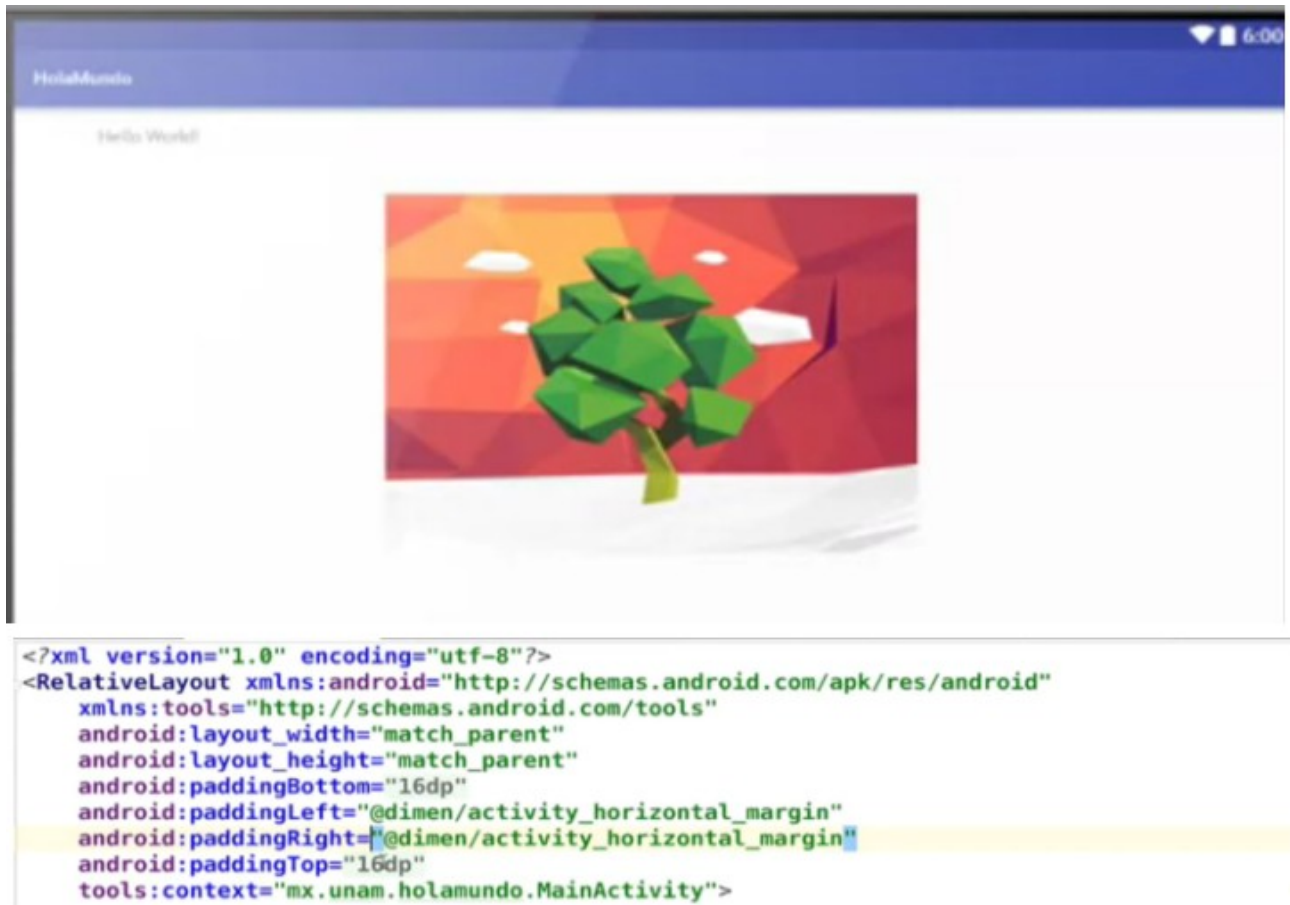
Está tomando el padding por default que está aquí en nuestra carpeta dimen, pero ahora si cambiamos a nuestra vista súper grande, nuestra extra large, vamos a observar en la vista de texto que ahora el padding, me voy a regresar, ¿sí?

Pero ahora si observamos, este mismo layout, pero con otra con otra vista. Es decir, que no tenga este teléfono sino que podamos ver este mismo layout en un dispositivo a lo mejor más grande.

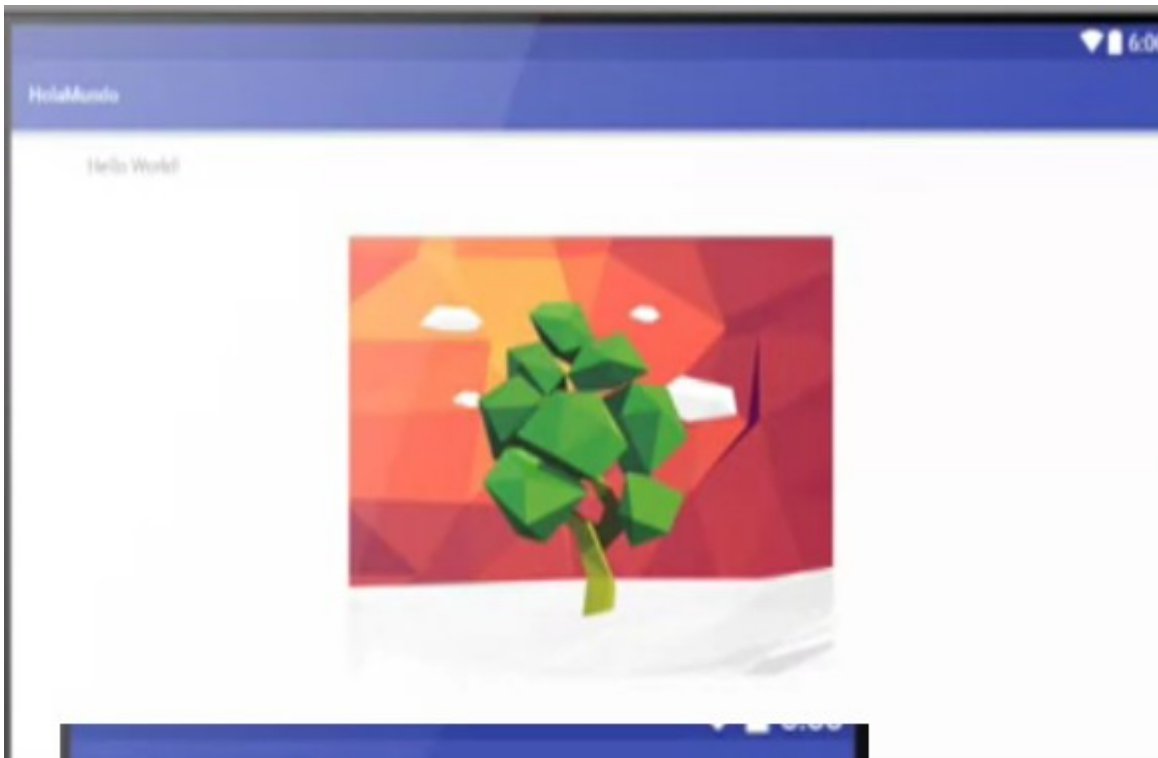
¿Recuerdas que el archivo `dimen` está colocado para `w820 dps`?

Vamos a buscar una resolución que sobrepase esa dimensión, para ver cómo se altera la variable.

Puedo probarlo, por ejemplo, para que tenga un Android TV. Tengo un Android TV, puedo colocarlo por ahí y ¿qué estamos viendo?. El padding ha cambiado, si vamos a nuestra vista de texto, aparentemente estamos tomando la misma variable.



Pero puedes observar que este padding, desde el texto que viste `hello world` y la pantalla, pues es mayor en comparación del que tenemos por aquí.



Porque el de aquí sólo tiene 16 y para una pantalla muy grande está tomando el d64, es una pantalla que sobrepasa los 820dps, densidades. De esta forma es como podemos generar todas nuestras dimensiones y para múltiples pantallas también.

El archivo `dimens` se creó para manejar todas las unidades de medida, todas las medidas que manejemos en nuestra aplicación. Esto va a incluir `padding`s, va a incluir `margins`. Va a incluir, a lo mejor, tamaños de texto, tamaños de imágenes, todos los tamaños son las unidades que te puedas imaginar que tu aplicación móvil pueda ocupar. Las vamos a manejar y las vamos a concentrar en este archivo `dimen`.

Si en algún momento tuvieras la necesidad de manejar dimensiones exclusivas para pantallas más grandes; entonces, creamos también nuestra variante del archivo `dimen` como se ve aquí.

Pero manejar las variables de esta forma, las unidades en variables, nos va ayudar a que si en algún momento nos solicitan un cambio, pues sea más fácil realizarlo.

Imagínate que de pronto te solicitan que todos los botones que manejas en tu aplicación sean más grandes.

O las imágenes o el logotipo de la empresa que estás manejando en tu aplicación sean más grande. No vas a ir a cada pantalla a cambiar el tamaño de la imagen. Si tu manejas ese tamaño en una sola variable solamente alteras esa variable y automáticamente se aplica a todas tus demás pantallas.

Ésta es una utilidad del archivo `dimen`.

Styles

Como su nombre lo dice va manejar todo el estilo de tu aplicación.

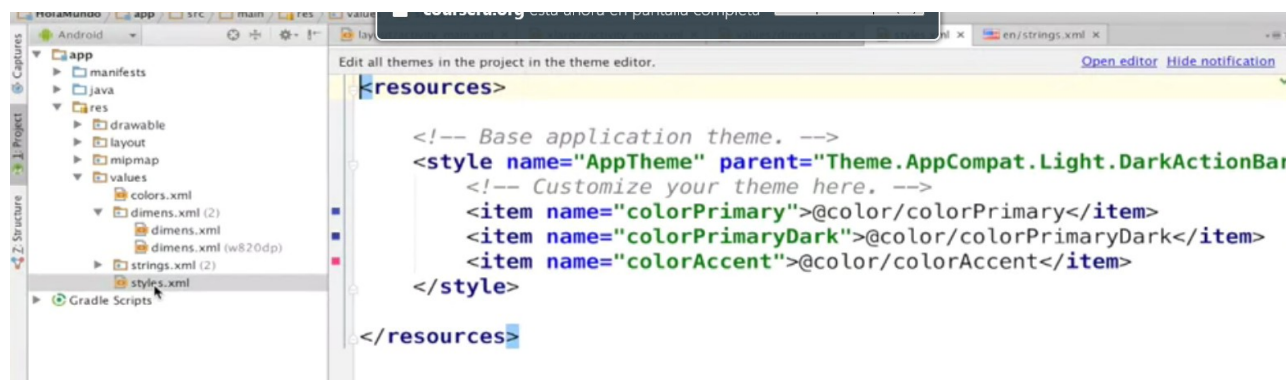
Hoy en día tenemos un nuevo estilo que es muy favorable para todos y es con la apertura de `material design`.

`Material design`, una tecnología creada por google. Y básicamente, lo que ha hecho `Material Design` es darle un nuevo estilo a nuestras aplicaciones.

¿A qué se refiere esto? Bueno, toda la concentración de colores, formas, diseños e incluso hasta cómo se ve un botón.

Si el botón se ve con bordes, con sombras, todo eso lo vamos a manejar lo vamos a concentrar en nuestros estilos, aquí, este archivo de estilos.

Como observas, también es un estilo XML, tiene esta etiqueta `estilo`, la variable se llama `AppTheme`. Y tiene asignado el tema, `AppCompat.Light.DarkActionBar`, y además tiene incrustados los tres colores de `material design`, que los veremos más adelante.



Tu color primario, tu color oscuro primario, y el color de acentuación. Tú podrías manejar más estilos aquí si así lo quieres, pues yo podría manejar otro estilo por acá y asignarle una nueva variable.

Le puedo poner "Mi Tema" y puedo alterar un poco aquí, que esto sea un "NoActionBar". Entonces, aquí estoy manejando dos estilos o varios estilos concentrados en este archivo.

```
<style name="MiTema" parent="Theme.AppCompat.Light.NoActionBar">
    <!-- Customize your theme here. -->
    <item name="colorPrimary">@color/colorPrimary</item>
    <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
    <item name="colorAccent">@color/colorAccent</item>
</style>
```

Pero ¿realmente en dónde se define el estilo que va a ocupar mi aplicación? Eso lo hacemos en otro archivo, que más adelante platicaremos que es nuestro archivo manifest.

En tu archivo manifest, vas a definir el estilo que estarás ocupando. Puedes tener varios ahí viviendo, pero sólo puedes ocupar uno y ese es el que va alterar toda tu aplicación.

Todas tus pantallas, todos tus elementos, la combinación de colores, de formas, todo va estar concentrado en estilos.

Archivos Java

Los archivos java los vas a poder encontrar en esta carpeta, la carpeta se llama java, tal cual. Y además está contenido en el package que definimos al principio del proyecto.

Como observas tenemos aquí dos packages, uno que dice Android test y el otro simplemente pues no lo dice.

Bien, donde vamos a estar trabajando nosotros es en el package donde **no** contiene Android test.

Android test es para generar pruebas unitarias o pruebas de otros estilos.

Nosotros vamos a estar manejando todos nuestros archivos java aquí, en este package.

Bien, Ahora, como observas ya tenemos una clase de java, que se llama main activity. Main activity que cuando creamos nuestro proyecto, también es una pestaña que está ahí abierta por default.

Es una clase de java, que contiene un método o un create y que además esta clase está heredando de AppCompatActivity.

```
package mx.unam.holamundo;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

¿Qué nos quiere decir esto? En primera instancia, si está heredando de AppCompatActivity quiere decir que esta clase es una actividad.

Cuando aplicamos la herencia en java, automáticamente, haz de cuenta que nos convertimos en esa clase, de la clase de la que estamos heredando y ya obtenemos todos los atributos, todos los métodos, ya son nuestros. Entonces esta clase, podemos saber que es una actividad.

Una actividad que ese concepto todavía no lo hemos visto como tal, pero te puedo decir que este archivo de java, va a controlar o va a manejar todo lo que tengas en tu interfaz gráfica.

Recuerdas hace un momento, en las secciones anteriores estuvimos hablando mucho sobre interfaces gráficas. Cómo generar tu interfaz, tus botones, tus imágenes, textos, etc. Bueno, precisamente, este archivo de java es el que se va a encargar de que cuando le des click a ese botón, pues ese botón haga algo. O que si estas escribiendo un texto y entonces quieres mandarlo a algún lado.

Bueno, ese archivo de java se debe encargar de manejar toda esa lógica, a dónde mandar ese texto, cómo manejar ese texto y dónde almacenarlo finalmente. Bien, tus archivos de java, o sea tus clases de java, deben estar ubicadas siempre en nuestro package.

Y podemos identificar que este archivo está controlando o este archivo le pertenece cierta vista o cierto layout, a través de la instrucción que está aquí.

```
package mx.unam.holamundo;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

Dice SetContentView, que traducido podríamos decir que tiene el contenido de cierta vista, que vista nuestra interfaz gráfica, o sea nuestro layout. SetContentView, le estamos diciendo a ese archivo obtén el contenido de ese layout y enseguida viene R.layout.activity_main.

Bueno, si primero le estamos diciendo que maneje el contenido de esta vista, necesitamos indicarle pues, ¿de qué vista?

Porque ya podemos tener muchas interfaces gráficas, infinidad de archivos xml, etc.

Entonces yo necesito decirle a ese archivo de java, qué archivo va a controlar.

Bien, ¿recuerdas nuestro archivo R?, nuestro archivo R que estuvimos hablando, el cual guarda todas las referencias de todos los recursos que creas en tu proyecto de Android.

Este archivo R, aquí estamos accediendo a él, a través de R.layout. Con R.layout, le estoy diciendo que del archivo R obtenga el layout.activity_main. Es decir, le digo ve y busca el archivo R, el layout que se llama activity_main.

Y activity_main, puedes observar que es el mismo nombre de nuestro layout activity_main.xml. Y se llama así, sin la extensión aquí en nuestro código java, sin la extensión .xml.

Porque en el momento en el que fue creado, se mapeo en el archivo R pero como una variable.

Es decir, en el archivo R, no existe como un xml, sino existe simplemente como una variable que contiene la dirección de memoria donde está habitando ese recurso de Android. Recuerda es una variable.


Entonces con esto, estamos diciendo que esta clase de java va a manejar y va a contener toda la vista de nuestro archivo, de nuestro layout activity_main.xml. Si tu presionas la tecla control y observas que activity_main se coloca como un link, le damos clic ahí y aquí me aparece que escoja a cuál de los layouts quiero ir.

```
package mx.unam.holamundo;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```



Y con ésto me está indicando que este archivo de java, MainActivity, está controlando o está manejando todos los views de esta interfaz.

De esa forma es como vamos a estar manejando nuestros archivos de java. Generalmente un archivo de java siempre va a manejar o siempre va a contener todo lo de una vista, lo de un layout, en términos de view model.

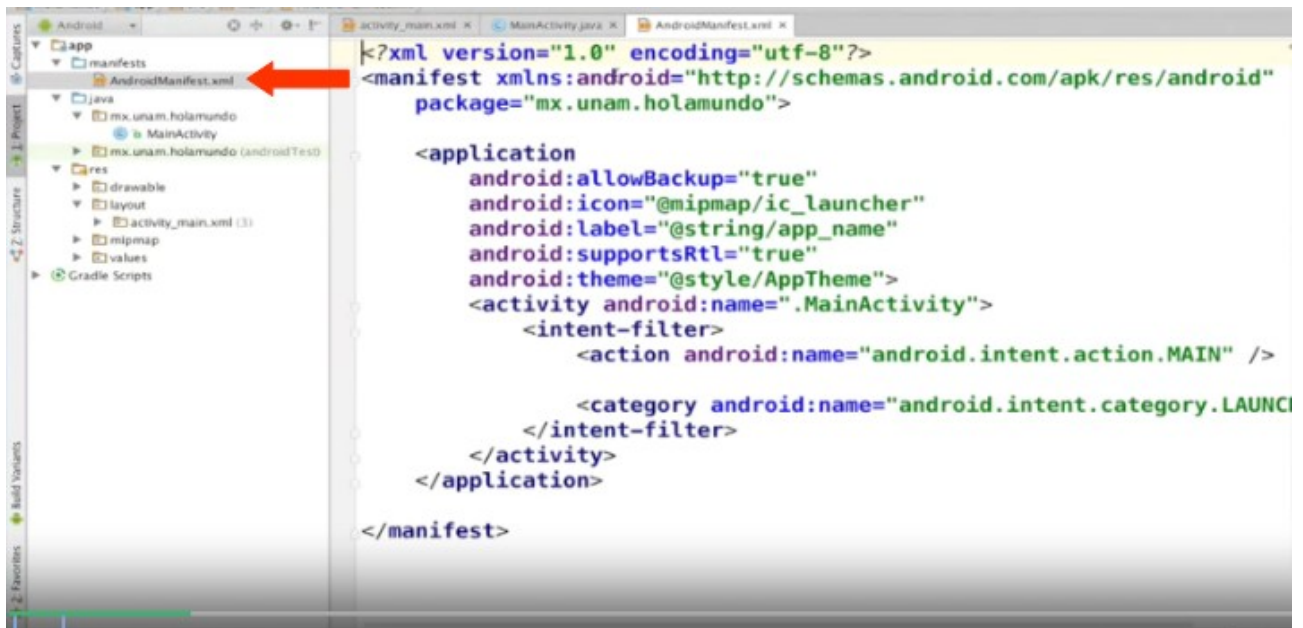
Pero también los archivos de java pueden controlar bases de datos, pueden controlar conexiones a un servidor, pueden controlar sensores o infinidad de cosas que tú le digas a tu archivo de java.

Todo el control, toda la lógica del negocio se va a concentrar en tus archivos de java y deben estar ubicados en ese package.

Android Manifest

Estamos ya con nuestro archivo Android Manifest. Y lo vas a ubicar en la carpeta manifest que está contenida en la carpeta app. Este es nuestro archivo androidmanifest.xml.

Vamos a abrirlo, y lo que encontramos en ese archivo, este archivo androidmanifest se conoce también como el corazón de nuestra aplicación. El corazón de nuestra aplicación porque es el archivo de configuración de toda nuestra aplicación.



En él vas a encontrar el paquete de la aplicación, vas a encontrar incluso el tema que está ocupando la aplicación en este momento.

¿Te acuerdas cuando vimos nuestro archivo styles? Bueno todo eso lo vamos a encontrar aquí. Fíjate bien, igualmente es un archivo XML, aquí podemos ver en la etiqueta application, contiene la propiedad icon. Y está haciendo referencia al ícono de la aplicación.

Es decir, cuando abra, cuando se instale esta app, la imagen que le estemos diciendo aquí es la imagen que se va a quedar ahí como vista. Como acceso directo de nuestra aplicación desde nuestro menú de aplicaciones.

También viene el label que está tomado de un recurso @string. Recuerda que accedamos a recursos de XML a XML con @ nombre del recurso en singular.

Y desde Java accedamos a un recurso a través del archivo R.

Bien, entonces aquí voy a estar accedando a un recurso a través de @. Por ejemplo, mipmap el ícono, @string/ app_name, es el nombre de nuestra aplicación. Este también va ser el nombre que aparece cuando nuestra aplicación se instala.

Después tenemos el theme, que como te platiqué hace un momento, en el theme tenemos @style. Y está accedando al recurso styles, y está tomando la variable AppTheme.

Así vamos para allá, recordamos que tenemos la variable AppTheme. Y de igual forma puedes añadir un nuevo estilo, asignarle una nueva variable.

Y aquí es donde configuramos cuál de los dos estilos que están dados de alta es el que va a tener tu aplicación móvil.

Muy bien, otra cosa que también va a tener nuestro archivo Android Manifest, es siempre que declaremos un activity, osea una pantalla.

Siempre que declaremos un servicio, siempre que declaremos permisos, que ocupemos permisos para lo que sea. Para abrir la cámara, para habilitar el wifi, para utilizar el bluetooth, cualquier cosa de hardware que nosotros ocupemos del teléfono. Se considera algo privado y algo privado del usuario.

Y es algo que nosotros como desarrolladores o nuestra aplicación, debe siempre solicitar permiso para usarlo.

Estos permisos también se declaran aquí, en nuestro archivo Android Manifest.

Y más adelante, en ejemplos posteriores vamos a ver detalladamente como dar de alta a nuestros permisos. Entonces el archivo Android Manifest, el corazón.

El corazón de nuestra aplicación, un archivo clave de configuración para nuestras aplicaciones móviles.

Gradle

Gradle es una de las herramientas que se introdujeron en Android Studio. Como uno de los plus que tiene Android Studio en comparación con Eclipse.

Gradle básicamente es una herramienta que nos va a ayudar a construir nuestro proyecto y va a ser de una forma súper automática.

Va a tomar todas las librerías, todos los recursos, todo lo que tenemos en nuestro proyecto. Lo va a computar y va a construir finalmente, una aplicación móvil lista para correr.

Una de las cosas que tiene Gradle muy peculiares es que tiene un sistema interno donde Gradle, al momento de correr nuestra aplicación, verifica si hubo un cambio en el código.

Si hubo un cambio en el código, re compila todo y genera nuestra aplicación. Pero si no hubo ningún cambio, simplemente se ahorra el trabajo, por lo tanto hace mucho más eficiente y mucho más rápido correr nuestras aplicaciones móviles.

Vamos a ver como funciona Gradle y cómo es que está presente en nuestro proyecto.

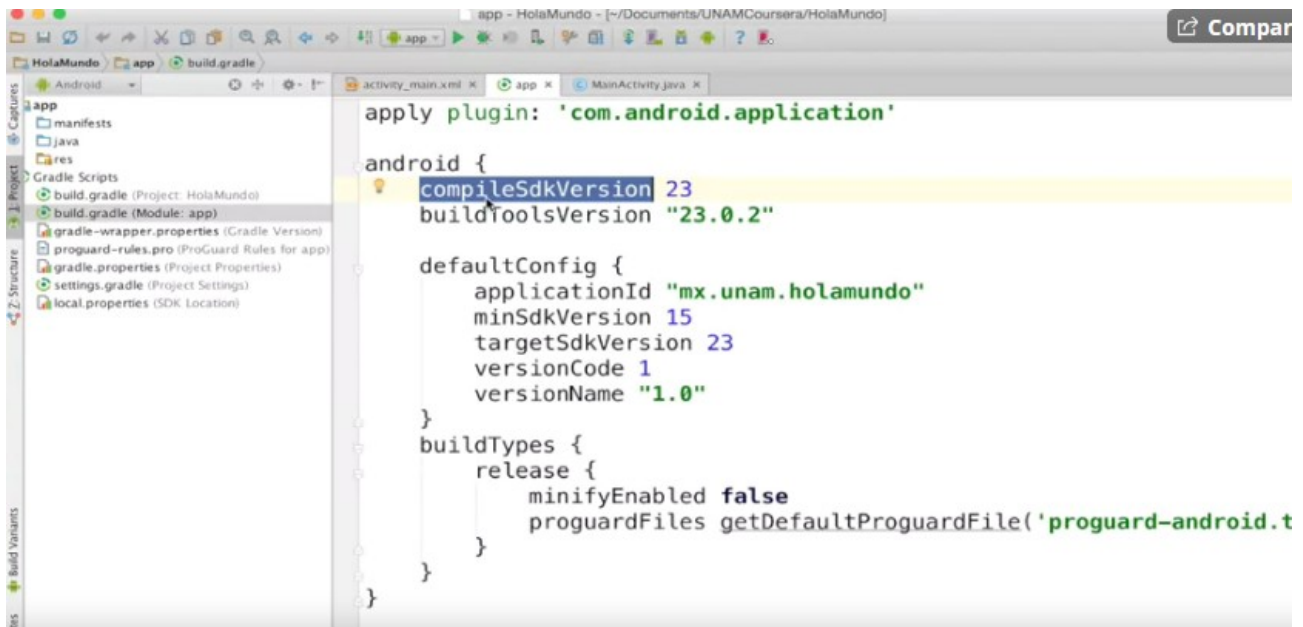
Gradle lo vas a ver presente en esta parte de nuestro proyecto. Está al mismo nivel de la carpeta APP y allí tenemos Gradle scripts.

Gradle scripts tiene todo esto. Todos estos son archivos de configuración importantes para Gradle.

Y un archivo que a nosotros nos importa bastante es este build.gradle (Module: app).

Si nosotros abrimos build.gradle (Module: app) vamos a encontrar primeramente una serie de configuraciones que también son importantes para construir nuestro proyecto.

Por ejemplo tenemos el Sdk de compilación. Y compatibilidad de versión de nuestra aplicación móvil. Por ejemplo, nuestra aplicación móvil hasta que versión de Android va a ser compatible.



Aquí hemos colocado que hasta la 23. Hemos definido también por acá más abajo en defaultConfig el applicationId. que es el package que configuramos también al principio de nuestro proyecto.

El mínimo SDK version, colocamos el API 15. Si tú en algún momento quisieras alterar alguno de estos parámetros.

Por ejemplo algo que es muy común es que de pronto cuando yo creo un proyecto, a veces sin saber defino un mínimo SDK por ejemplo coloco un API 21, el cual corresponde a Lollipop.

Entonces si quiero correr mi aplicación en una versión anterior al Lollipop como Kitkat, simplemente no es compatible.

Y entonces me surge la curiosidad de que quiero que sea compatible, bueno pues simplemente en este renglón es donde yo modifico la versión del API. Si es que yo tenía 21 entonces ahora lo voy a cambiar a 15 o 14 o la que quiera seleccionar.

Como observas, en la parte superior ya nos apareció una notificación que dice que ahora es necesario volver a sincronizar todo el proceso interno de Gradle en Android para que esto nuevamente se genere. Simplemente damos sincronizar ahora y automáticamente Gradle hace todo el cambio, todo el proceso. Si teníamos 21 de compatibilidad, ahora va a ser, 15.

Tenemos también el targetVersion que es también, efectivamente. Hasta que versión de API va a ser compatible nuestra aplicación. Desde la 15 hasta la 23.

parámetro versionCode

El versionCode es algo muy importante, sobre todo cuando nosotros subimos nuestra aplicación al Play Store. El versionCode indica la versión de tu aplicación.

Es muy común que posterior a haber subido nuestra aplicación al play store pues surgen actualizaciones, o mejoras, o simplemente cambios de módulos, etc.

Play Store nos va a pedir o nos va a solicitar un versionCode distinto. Es importante que si tú estuviste manejando en la versión preliminar el versionCode 1 haces una actualización. Es importante modificar este versionCode.

El versionCode siempre debe ser un número entero.

Es decir si estábamos en la uno, pues ahora lo que sería consecutivo sería la dos. No podemos colocar uno punto cinco por ejemplo aquí.

No podemos, debe ser un número entero y siempre debe ser mayor al anterior. Puedes colocar el dos que sería mayor a uno o si tú quieres puedes colocar el cinco o el seis o lo que tú quieras en ese aspecto. Pero siempre tiene que ser mayor al anterior.

Parámetro versionName

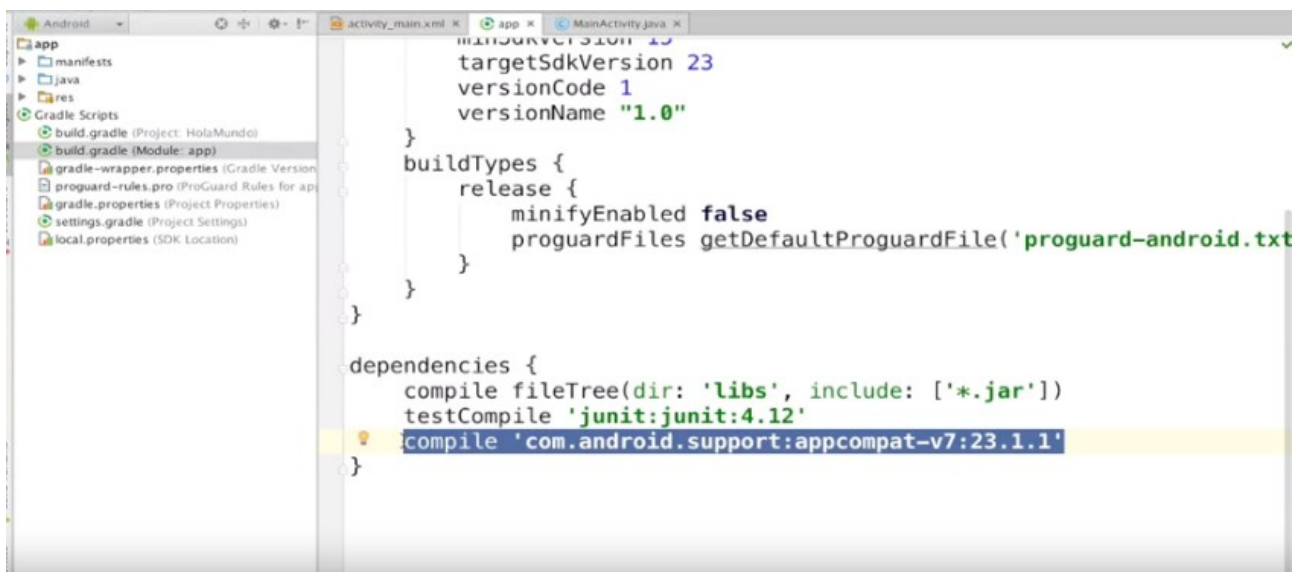
El versionName nos va a ayudar, precisamente, para decirle o notificar a nuestros usuarios de nuestras aplicaciones, que tenemos otra versión de aplicación. Si observas el versionName ya es un parámetro un poco más flexible, es un string que almacena un número en términos de puntos y entonces aquí podemos modificar 1.1 si a lo mejor la actualización es mínima. 1.2, 1.2.3, 1.5, etcétera.

Este versionName es la versión de nuestra aplicación que nuestros usuarios ven cuando van a descargar nuestra aplicación.

Dependencias

Es donde nosotros si queremos agregar una librería adicional, que queramos que nuestro proyecto maneje.

Podría ser, por ejemplo, cuando implementamos algunas herramientas del material design, que no están disponibles en versiones al API en versiones anteriores al API 21. Nosotros, esas dependencias, aquí justamente las agregamos. Y con la instrucción compile, espacio, seguido de las comillas, justo como se ve aquí.



Entonces, lo que va a hacer Gradle es que va a tomar toda esta configuración que hemos definido. Y va a construir nuestro proyecto, finalmente nos va a generar un archivo ejecutable de Android.

El archivo ejecutable de Android es un archivo APK, Application Package File.

Esto es lo que hace el sistema Gradle en Android.

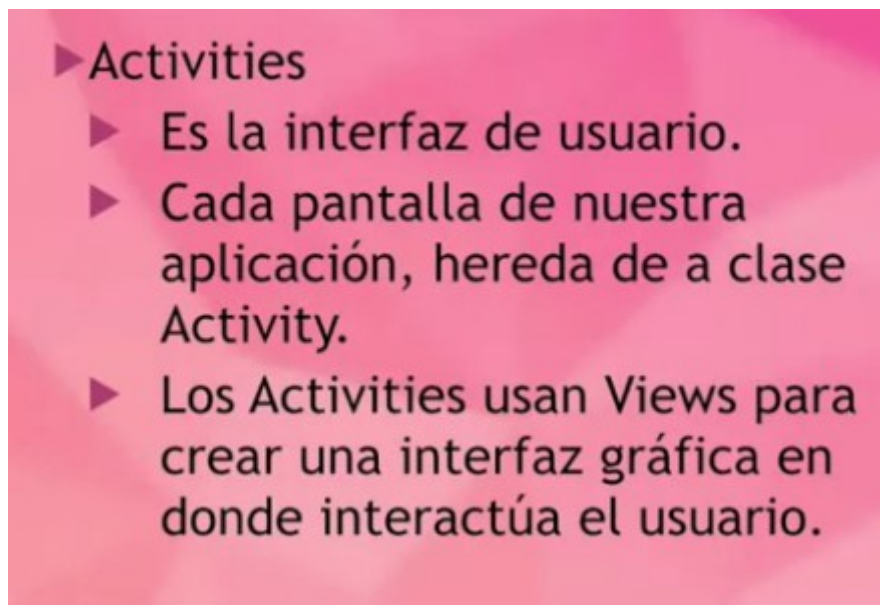
Módulo 4

Componentes de una aplicación móvil

Activities

Las activities son cada pantalla que compone tu aplicación, o sea, una Activity es representada por cada pantalla que compone nuestra app. Cada una de estas pantallas se va a definir a partir de un archivo de xml que va a definir nuestra interfaz gráfica y un archivo de Java que se va a encargar de estar manipulando la interfaz gráfica.

Estos en conjunto van a ser nuestro activity. Los activities, o sea nuestras clases de Java utilizan views para generar interfaces gráficas. Lo que tú ves finalmente en tu aplicación móvil, cada pantalla que ves eso, cada uno de ellos son actividades.



Servicios

Los servicios no son activities, sino los servicios es un fragmento de código que se comporta invisiblemente.

Puede ser activado a partir de una actividad, puede ser a través de un botón y de pronto el botón arranca un servicio.

El servicio está ejecutándose, ese fragmento de código está ejecutándose en background, en el sistema operativo pero en background.

Es muy común encontrar servicios como por ejemplo, servicios de notificaciones. Por ejemplo Whatsapp utiliza servicios todo el tiempo, no necesitas apretar un botón para que de pronto te lleguen todas tus notificaciones, sino que existe un fragmento de código que se está ejecutando todo el tiempo en el background y que está al pendiente de cuando te llega una notificación.

Así funcionan los servicios.

► Services

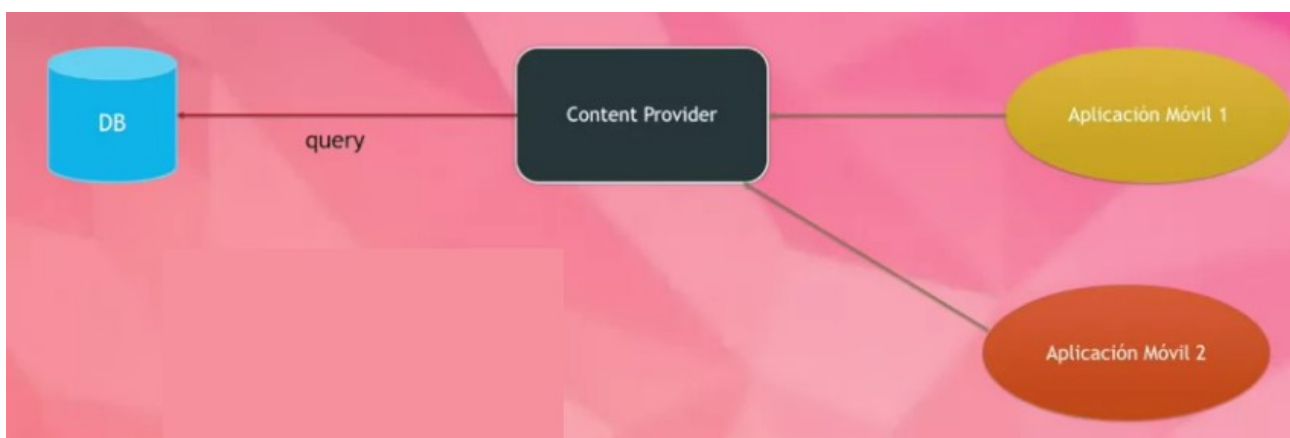
- Es una parte de la aplicación que se comporta “invisiblemente” (background)
- Pueden ejecutarse en segundo plano sin necesidad de que el usuario interactúe con él.

Content Providers

Lo común de una base de datos es que pues que cada aplicación tiene su propia base de datos y la base de datos es privada para cada aplicación. Pero los content providers van mucho más allá de ello. Básicamente tendremos nuestra base de datos y tendremos también un content provider.

El content provider puede acceder a la base de datos, la base de datos puede estar en cualquier lugar del sistema.

El content provider hace la magia para que cualquier aplicación pueda acceder a esos datos, es decir, los datos no están privados para nadie. Si están definidos como un content provider, todas las aplicaciones a través de él pueden accederlos.



Broadcast Receivers

Los Broadcast Receivers básicamente todo el tiempo están alertas al sistema operativo, si hay algún cambio en el sistema operativo o algún estatus, un aviso especial por parte del sistema operativo.

Por ejemplo, si tu batería está baja, lo que sucederá es que el broadcast estará pendiente de cuando exista este estatus en la batería.

El broadcast va a procesar esa información y entonces lanzará una notificación, lanzará una notificación a ti diciéndote que ahora la batería está baja.


También por ejemplo si hay una llamada entrante, el broadcast está pendiente. Si existe una llamada entrante entonces la procesa y en ese momento levanta una aplicación, en este caso la aplicación que levanta es nuestra aplicación de llamadas, mostrándonos no se, la foto de la persona, el número telefónico de quién está llamando, una serie de botones donde puedes colgar o desviar la llamada, etc.

Los broadcast receivers todo el tiempo están alerta a notificaciones y avisos del sistema operativo.

- ▶ **Broadcast Receivers**
 - ▶ Son componentes que responden a avisos y anuncios de difusión.
 - ▶ Estos avisos provienen del sistema
 - ▶ Batería baja
 - ▶ Llamada entrante
 - ▶ Etc.
 - ▶ Se pueden pasar de una aplicación a otra
 - ▶ Se activan a través de los intents

Views

Los views son cada componente de tu interfaz de usuario. Es decir, cada botón, cada ready button, cada spinner o cada etiqueta de texto que ves ahí en tus interfaces móviles, todos estos componen tu interfaz de usuario y además están agrupados en layouts. Estos son los views.



Componentes: Views

- ▶ Son los componentes de la interfaz de usuario.
- ▶ Estos pueden agruparse en un layout.
- ▶ View es la clase padre de Activity.

Clase Intent

La clase intent, es un componente de nuestras aplicaciones que básicamente nos va a ayudar, va a funcionar como un pegamento, un pegamento entre una actividad y otra. Es decir la clase intent se va a encargar de levantar procesos.

- ▶ Son mensajes que provocan notificaciones, cambios de estatus.
- ▶ Al ser recibidos por actividades pueden levantar procesos.
- ▶ Unen componentes dentro de la misma aplicación o de diferentes aplicaciones.

Seguramente has visto cuando de pronto tecleas un botón y nos pasa a otra actividad. Bueno, ese flujo entre una actividad, una pantalla y otra lo produce la clase intent.

También cuando por ejemplo seleccionas el botón de compartir puedes ver que de pronto se abre una ventana, y nos dice, ¿con qué aplicación quisieras compartir esta información? Bueno, también este proceso está siendo levantado por la clase intent.

El intent puede servir como un pegamento entre pantallas de la misma aplicación o entre componentes de una aplicación y otra.

Es decir, la clase intent puede levantar otra aplicación estando en donde sea.

Resumen

Hemos visto todos los ingredientes que necesitas para crear tus aplicaciones móviles.

Por ejemplo, tu aplicación se puede componer de muchas pantallas, eso seguro lo vas a tener.

Estas pantallas se llaman activities que a su vez están unidas por intents.

Estas pantallas seguramente van a tener una interfaz gráfica, que están compuestas por views. y también si lo deseas tu aplicación puede levantar un servicio que va a estar escuchando en background o puede estar alerta para cualquier notificación del sistema operativo.

Módulo 5

Patrones de diseño para android

Android MCV y MPV

Veremos en especial dos patrones de diseño, el modelo vista controlador y el modelo vista presentador.

Modelo Vista Controlador

El modelo, el modelo básicamente van a ser nuestros datos, todas nuestras bases de datos en Android. Para ello, el manejador de base de datos que utiliza Android es SQLite. El modelo.

Luego tenemos la vista. La vista, como hemos estado hablando, son nuestras interfaces gráficas, todos nuestros layouts, nuestros archivos xml.

Y el controlador es precisamente aquél que se va a encargar de estar manipulando la vista pero también manipulando los datos.

¿Y te acuerdas quién dijimos que es? Nuestros archivos de Java.

Todas nuestras clases de Java van a ser nuestros controladores.

Es decir, si lo vemos en términos de un activity, un activity tiene un layout que sería la vista, y también tiene una clase de Java.

Este sería nuestro controlador.

Si nosotros en algún momento ejecutamos una consulta a una base de datos desde el controlador, estaríamos completando el esquema, modelo, vista y controlador.



Modelo Presentador Vista

¿Cuál es la diferencia entre el controlador? Bueno, se ha dejado de usar un poco el anterior modelo vista controlador porque, pues, cuando el controlador quiere ejecutar una consulta en la base de datos, y al mismo tiempo el controlador tiene que mantener toda la relación, todo el control entre la vista, se dice que es demasiada carga para un solo archivo de Java.

Entonces, de pronto llegan a ser archivos de Java enormes, muy complicados de leer y muy complicados de entender, puesto que maneja todo el control de las vistas, de los views, y además todo el control de la base de datos.

Para ello se creó este modelo, este esquema modelo vista presentador.

Básicamente la vista se ha configurado de tal forma que ahora la vista va a ser el archivo xml y nuestro archivo de Java de nuestro activity.

Ambos archivos van a componer la vista, tanto el controlador que teníamos anteriormente, la clase activity, como también nuestro layout, nuestro archivo xml.

Eso será nuestra vista.

El modelo se conserva tal cual, el modelo serán todos nuestros datos, nuestras bases de datos escritas con SQLite.

Ahora, para poder acceder desde nuestra vista, que es nuestra activity completa, hasta el modelo, se ha creado un intermediario que es el presentador.

El presentador se va a encargar de manejar toda la lógica, y además que todas las transacciones se hagan correctamente. El presentador va a interactuar con el controlador que dejamos por allá con nuestra actividad, y además ese controlador va a estar por ahí jugando con toda la vista.

Entonces el presentador va a estar interactuando con la vista completa, y además también con nuestro modelo de datos. Dando como resultado un modelo, presentador y vista. O modelo, vista, presentador.

Material Design

¿Qué es y cómo funciona Material Design?

Material Design es una guía integral para el diseño de movimientos y de interacción entre el usuario y distintos dispositivos. Fue introducido por Google a partir de la versión Android Lollipop en el API 21.

Debe cumplir con estos 3 principios:

1. atractivo visual,
2. movimiento
3. diseño interactivo

Es llamado Material Design porque todos sus elementos deben estar formados por materiales.

En especial materiales de impresión como hojas, papel, sombras, luces, etc., todo lo que provoque en una hoja de papel, o sea lo flat de una hoja de papel.

A todo esto se le llama Material Design.

Se le ha dado vida a este papel, a estos materiales de papel a partir del efecto que producen cuando proyectas una sombra o cuando doblas, cuando haces un pliegue sobre el papel, etc.

Además en cuestión de los botones y todo lo que tocas en la pantalla de tu dispositivo, produce un efecto, un efecto como de agua, como de ondas. Eso ha sido inspirado al lanzar una gota de tinta en el agua, como se puede observar cuando lanzas una gota de tinta en el agua esta produce un efecto en el agua, como si de pronto la tinta se disolviera y el agua produjera su efecto natural.

Esto mismo se ha querido plasmar en Material Design, las luces y sombras por un lado para dar profundidad al papel, efectos de animación como ondas en el agua, etcétera, y además todos estos efectos y animaciones deben cumplir estrictamente con las leyes de la física.

Es decir, en ningún momento un papel debe traspasar a otro papel sino que esto debe darse de acuerdo a las leyes de la física.

Por lo tanto todo efecto o toda animación, todo lo que produce Material Design es tan intuitivo como nuestra realidad misma.

Nuestra realidad, nuestros efectos, nuestros movimientos que tenemos son producidos a través de las leyes de la física, lo mismo tiene Material Design.

La composición de colores es un tema muy importante en Material Design, de tal forma que Google ha diseñado toda una paleta de colores que está diseñada a prueba de ingenieros, literal cualquier color que elijas aunque sea al azar todo combinará a la perfección.

Es muy importante que manejes el colorido en tus aplicaciones, que lo denotas bastante.

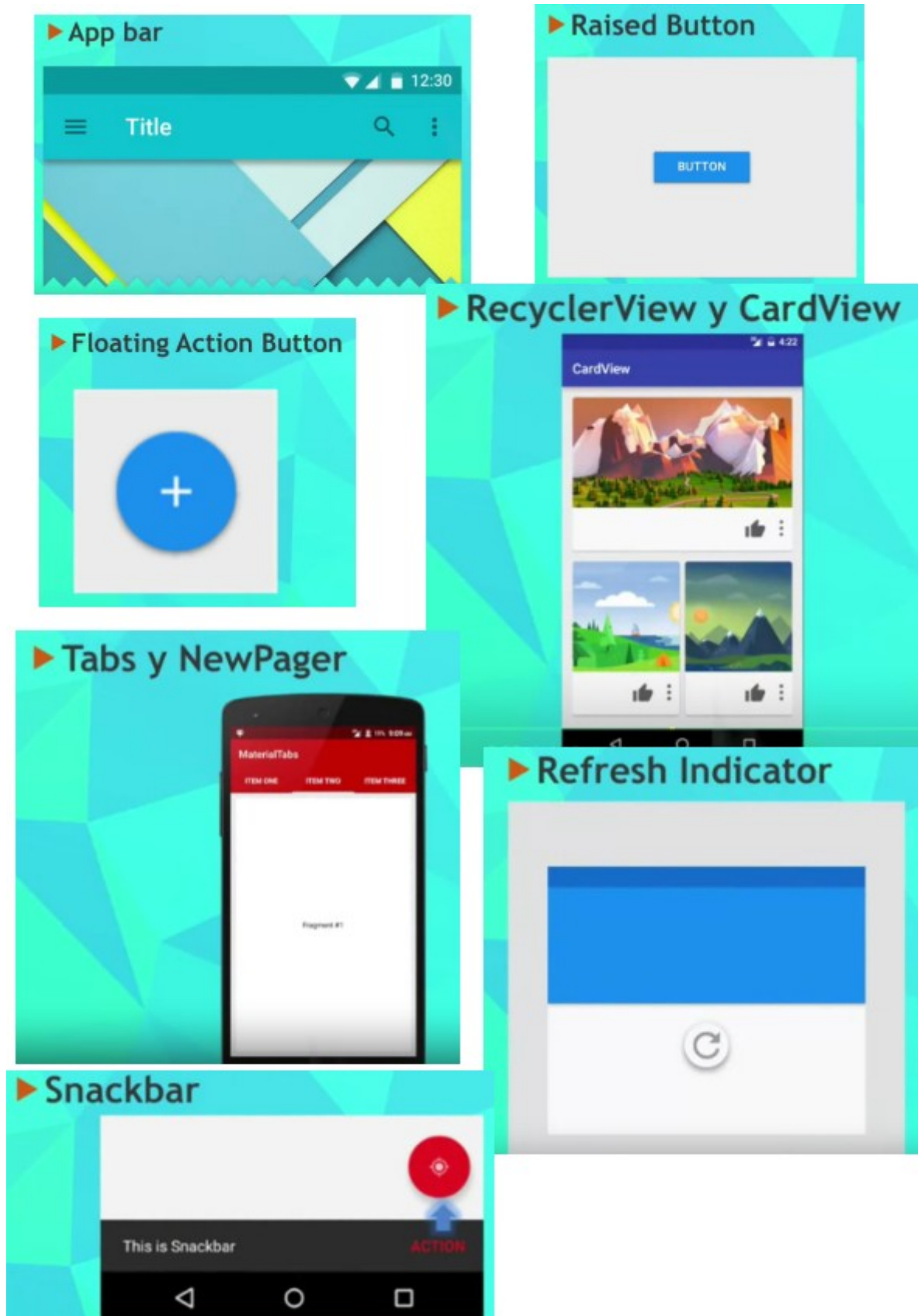
Pero Google ha colocado, por lo menos para Android, 3 colores principales.

1. Primer color, primary color
2. nuestro segundo color, dark primary color
3. accent color.

Estos colores deben estar presentes en tus aplicaciones Android si es que vas a trabajar con Material Design.

Algunos views que se han modificado o que se han integrado como nuevos a Material Design son los siguientes:

1. App Bar
2. Raised Button,
3. Floating Action Button
4. CardView
5. RecyclerView,
6. Tabs y NewPager
7. Refresh Indicator
8. Snackbar.



Lectura de Material Design

Material design es una guía completa para el diseño visual, interactivo y de movimiento en plataformas y dispositivos. A fin de usar material design en tus apps para Android, sigue las pautas

definidas en su especificación y usa los nuevos componentes y estilos disponibles en la biblioteca de compatibilidad de material design. En esta página, se ofrece una descripción general de los patrones y las API que debes usar.

Android ofrece las siguientes funciones para ayudarte a crear apps de material design:

- Un tema de app de material design para diseñar todos tus widgets de IU
- Widgets para vistas complejas, como listas y tarjetas
- Nuevas API para sombras y animaciones personalizadas

Tema de Material y widgets

Para aprovechar las características de los materiales, como el estilo de los widgets de IU estándar, y a fin de optimizar la definición de estilo de tu app, aplica un tema basado en material a tu app.

Para obtener más información, consulta cómo [aplicar el tema de material](#).

Para ofrecer a tus usuarios una experiencia familiar, usa los patrones de UX más comunes del material:

- Promueve la acción principal de tu IU con un [botón de acción flotante](#) (FAB).
- Muestra tu marca, navegación, búsqueda y otras acciones con la [barra de la app](#).
- Muestra y oculta la navegación de tu app con el [panel lateral de navegación](#).
- Usa uno de los muchos otros componentes materiales para el diseño y la navegación de tu app, como contracción de las barras de herramientas, pestañas, una barra de navegación inferior y mucho más. Para verlos todos, consulta el [catálogo de componentes de material design para Android](#).

Además, siempre que sea posible, usa íconos de materiales predefinidos. Por ejemplo, el botón de "menú" de navegación para tu panel lateral de navegación debe usar el ícono estándar de "hamburguesa". Consulta [Íconos de Material Design](#) para obtener una lista de los íconos disponibles. También puedes importar íconos SVG de la biblioteca de íconos de material con [Vector Asset Studio](#) de Android Studio.

Sombras y tarjetas de elevación

Además de las propiedades X e Y, las vistas en Android tienen una propiedad Z. Esta nueva propiedad representa la elevación de una vista, que determina lo siguiente:

- El tamaño de la sombra. Las vistas con valores Z más altos proyectan sombras más grandes.
- El orden de dibujo. Las vistas con valores Z más altos aparecen encima de otras vistas.

Con frecuencia, la elevación se aplica cuando tu diseño incluye un diseño basado en tarjetas, que te ayuda a mostrar información importante dentro de tarjetas que proporcionan un estilo de material. Puedes usar el widget [CardView](#) para crear tarjetas con una elevación predeterminada. Para obtener más información, consulta [Cómo crear un diseño basado en tarjetas](#).

Para obtener información sobre cómo agregar elevación a otras vistas, consulta [Cómo crear vistas de recorte y sombras](#).

Animaciones

Las nuevas API de animación te permiten crear animaciones personalizadas para respuestas táctiles en controles de IU, cambios en el estado de las vistas y transiciones de actividades.

Estas API te permiten:

- Responder a los eventos táctiles en tus vistas con animaciones de **respuestas táctiles**.
- Ocultar y mostrar vistas con animaciones con **efecto revelar circular**.
- Alternar entre las actividades con animaciones personalizadas de **transición de actividades**.
- Crear animaciones más naturales con **movimiento curvo**.
- Animar los cambios en una o más propiedades de vista con animaciones de **cambio de estado de las vistas**.
- Mostrar animaciones en los **elementos de diseño de listas de estado** entre los cambios de estado de las vistas.

Las animaciones de respuesta táctil se integran en varias vistas estándar, como botones. Las nuevas API te permiten personalizar estas animaciones y agregarlas a tus vistas personalizadas.

Para obtener más información, consulta [Descripción general de las animaciones](#).

Elementos de diseño

Estas nuevas funciones de los elementos de diseño te ayudan a implementar apps de material design:

- Los **elementos de diseño vectoriales** son escalables sin perder definición y perfectos para los íconos de la app de un solo color. Obtén más información sobre los [elementos de diseño vectoriales](#).
- El **ajuste de tono de los elementos de diseño** te permite definir mapas de bits como una máscara alfa y puedes ajustar el tono del color en el tiempo de ejecución. Consulta cómo [agregar ajustes de tonos a los elementos de diseño](#).
- La **extracción de color** te permite extraer automáticamente colores destacados de una imagen de mapa de bits. Descubre cómo [seleccionar colores con la API de Palette](#).

Maquetación

Cómo maquetar

El último fundamento, que es muy importante es respecto a la maquetación de tus apps.

Aquí vamos a aprender desde cómo maquetar una app a papel y lápiz y posteriormente pasar esto a un sistema, a un software en la computadora.

Lo primero para el login pues, tendremos una pantalla, únicamente tendremos dos pantallas, solo maquetaremos dos pantallas.

Tendremos una pantalla donde el fondo pues puede ser una imagen, de hecho vamos a meter una imagen de fondo y por aquí vamos a colocar otra imagen que esta imagen será nuestro logotipo .

Entonces esta será la pantalla 1, la pantalla 1, la cual inmediatamente que se abre la aplicación, esto es lo que ve el usuario, ve la pantalla de la app y ve el logotipo y además verá un botón que diga acceso.

Acceso es lo que nos gustaría que vieran. Bien, ese botón de acceso lo que hará será activar precisamente la pantalla de login.

Tú podrías tener pues varios colores si es que así lo quieres, de hecho es muy recomendable trabajar con varios colores en la maqueta y bueno pasando de la pantalla 1 a la pantalla 2.

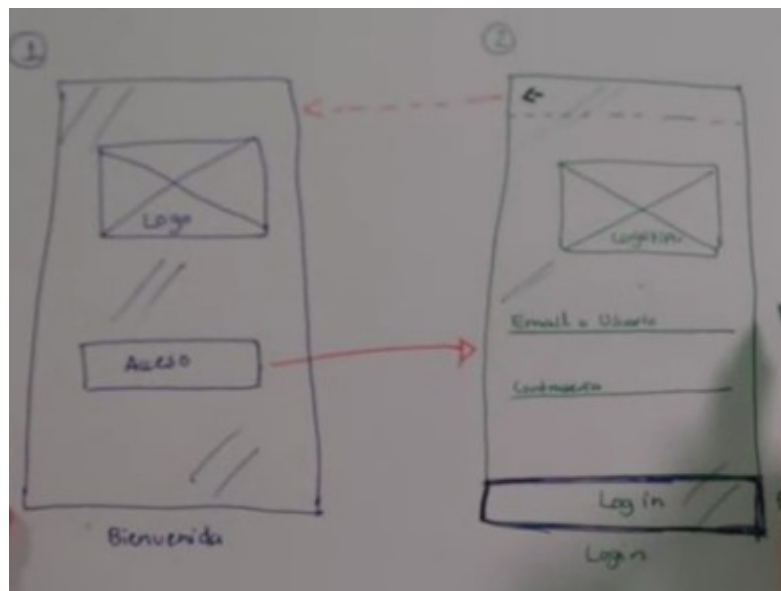
En esta pantalla 2 es en donde precisamente se requerirá el acceso.

Pues, lo primero que vamos a visualizar es que tendrá una pequeña, que será un poquito transparente, un app bar que es muy común hoy en día en material design, un app bar que es como transparente, coloco esta línea punteada únicamente para indicar que eso es un app bar,

Y bueno si estoy yendo del acceso a esta pantalla que es de login, esto sería como la pantalla de bienvenida, de bienvenida a la aplicación.

Si estoy yendo a login pues tendría, debería tener una flechita, como una flecha que me permita regresar a la pantalla de bienvenida.

Esta flecha hará el flujo hacia acá.



Lo que vamos a mostrar aquí, pues continuando con este flujo es que vamos a mostrar nuestro logotipo, una imagen, logotipo.

Por supuesto en el fondo, en este fondo que tenemos por acá, tanto en azul todo esto que tenemos acá, esto será una imagen y también el fondo que vamos a tener por acá que involucra incluso el app bar, también todo esto será una imagen. Entonces después del logotipo en nuestra pantalla vamos a solicitar el usuario y la contraseña.

Un input en material design simplemente se va a denotar por una línea, una rayita y podemos colocar, podemos solicitarle el email, email o usuario. Y también podemos solicitar más abajo la contraseña.

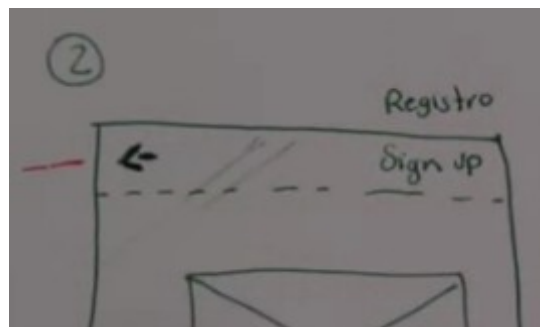
Sabemos que en material design cuando damos touch en cualquiera de estos elementos, el texto aparece flotando y entonces nos deja el espacio para escribir. Entonces eso es muy bueno.

En la parte inferior vamos a colocar un botón, toda esta zona que va a estar aquí, toda esta zona que podría denotarse como el footer de la app.

Todo esto va a ser un botón que va a tener la indicación de logearnos, de ejecutar el login. Este botón pues va a tener otro color, vamos a colocarlo con un color distinto a lo que estamos manejando para que se note que estamos, que esto es un botón de acción, es un botón de acción.

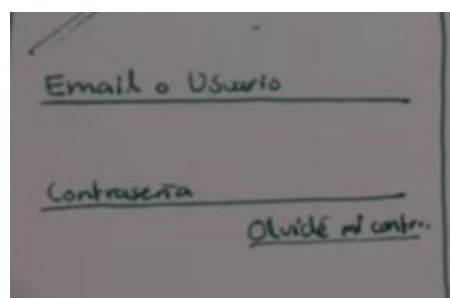
Adicional, sería bueno que de acuerdo al flujo de la aplicación, si no tienes un usuario y si tampoco tienes una contraseña pues sería bueno que pudieras registrarte.

Vamos a dejar nada más hasta aquí y vamos a colocar un enlace que se llame sign up o también podríamos llamarlo en español registro, registro y aquí a lo mejor ponemos lo vamos a dejar así, registro o sign up dependiendo de cómo estés manejando tu aplicación.



Aquí, ¿qué pasaría si no, podremos tener un caso, qué pasaría si no tengo mi contraseña o si no la recuerdo? Al mismo tiempo puedo definir un enlace que no sería un botón en este caso, para tener como seguir en el flujo y que el único botón de acción sea el login, podría decir que esto de aquí sea un enlace que se llame olvidé mi contraseña.

Entonces esto va a ser un enlace que lo voy a subrayar, esto será un enlace que me dirá olvidé mi contraseña.



Y listo, hasta el momento tengo 2 pantallas que me van a dar como resultado una ventana de login, una aplicación simplemente para logearme.

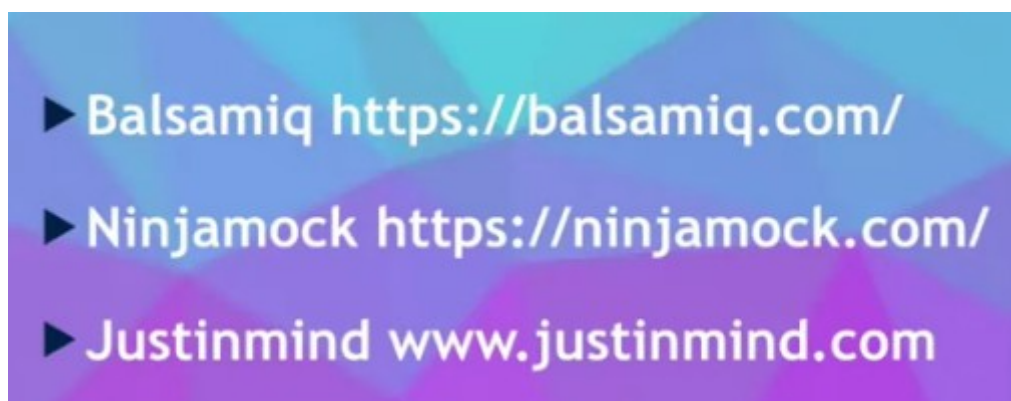
Vamos a pasar esto que tenemos en papel a un software, para que esto quede mejor documentado.

Bien, ya hemos construido nuestra interfaz gráfica en una hoja de papel con un par de plumones, ahora lo siguiente es llevar esto a la computadora para tenerlo pues en una mejor presentación.

Hay infinidad de herramientas online que puedes utilizar, de hecho, pues yo te voy a recomendar 3.

En internet puedes encontrar muchísimas que son gratuitas y otras que son parcialmente gratuitas.

- La primera es Balsamic, Balsamic es parcialmente gratuita y es un sistema que pues es compatible con cualquier sistema operativo, Windows, Linux o Mac, el que tú quieras. Y además es muy sencilla de utilizar, simplemente arrastras y sueltas elementos y voilá. Tienes ya tu interfaz construida, tu maqueta construida.
- La segunda es una plataforma online, una plataforma que además es completamente gratuita, la dirección es ninjamock.com. Ninjamock.com es un sitio donde puedes construir literal tus, puedes maquetar cualquier tipo de aplicación, puede ser web, puede ser móvil, puede ser para Android, para iOS, para lo que quieras y lo mejor, es gratuita.
- Y la última la cual es la que vamos a utilizar a continuación, es una plataforma que para mí es mi favorita, su nombre es just in mind, y es mi favorita porque just in mind ha adaptado o ha incluido recientemente todos los elementos de material design y de la misma forma que Balsamic, es muy sencillo de utilizar, simplemente arrastras y sueltas y además pues los elementos ya vienen contruidos, muchos vienen ya coloreados, vienen incluso puedes generar simulaciones, puedes generar animaciones y realmente just in mind te acerca mucho a la realidad a cómo se puede ver tu aplicación móvil.

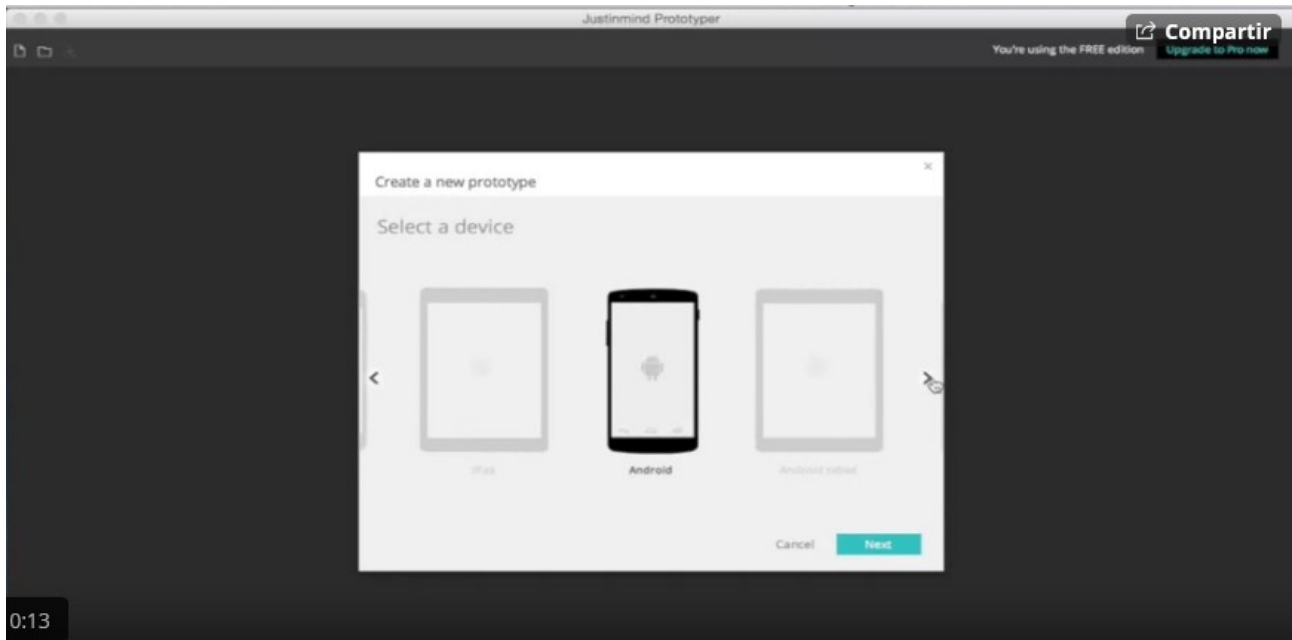


Desarrollando mi maquetación

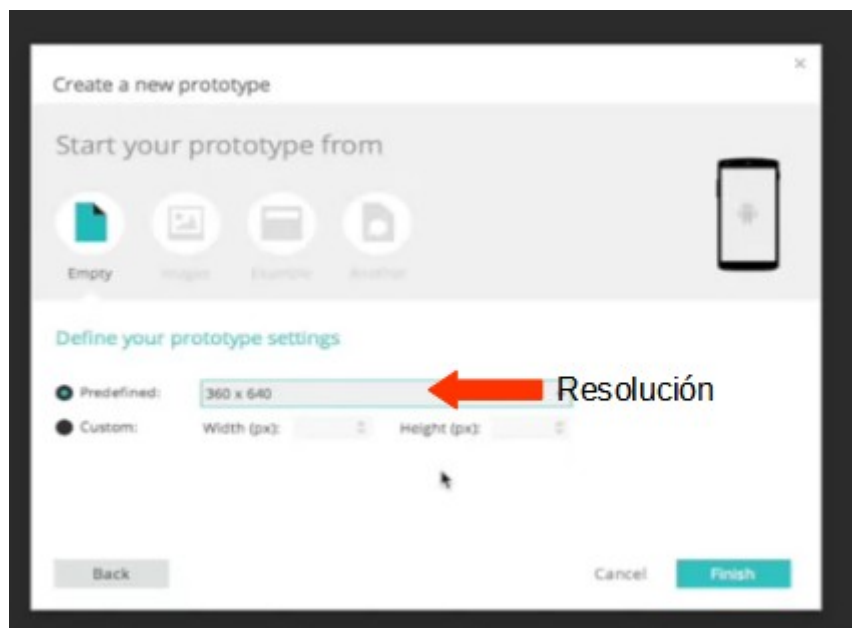
Ya tengo abierto just in mind en mi computadora. Lo primero que voy a hacer es crear un nuevo prototipo, le voy a dar aquí en create a new prototype, e inmediatamente ahora me dice qué tipo de prototipo o qué tipo de maquetación quiero crear.

Bien, puedo seleccionar uno web, o puedo seleccionar uno para iPhone, para iPhone plus, para iPad y la que nosotros estamos buscando es para Android, al mismo tiempo puedo seleccionar para una tablet, también una tablet para Android o algo que sea customizado o personalizado para ti.

Vamos a seleccionar Android, un dispositivo Android un smartphone, le voy a dar next.

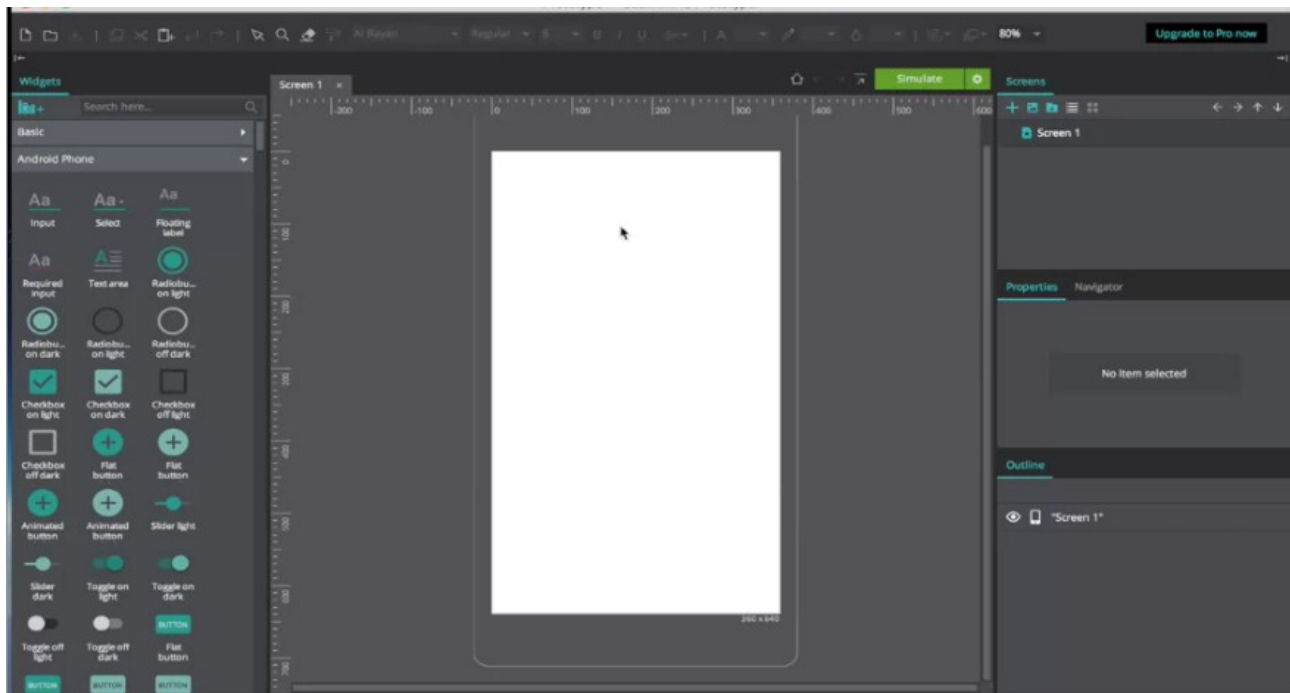


Lo siguiente que nos pide es la resolución predeterminada. La resolución predeterminada o lo que considera Android como normal, una resolución normal o medium es 360 pixeles por 640, así que vamos a basarnos en ella para construir esto.



Bien, vamos a darle finish.

Ya tenemos listo aquí nuestro prototipo.



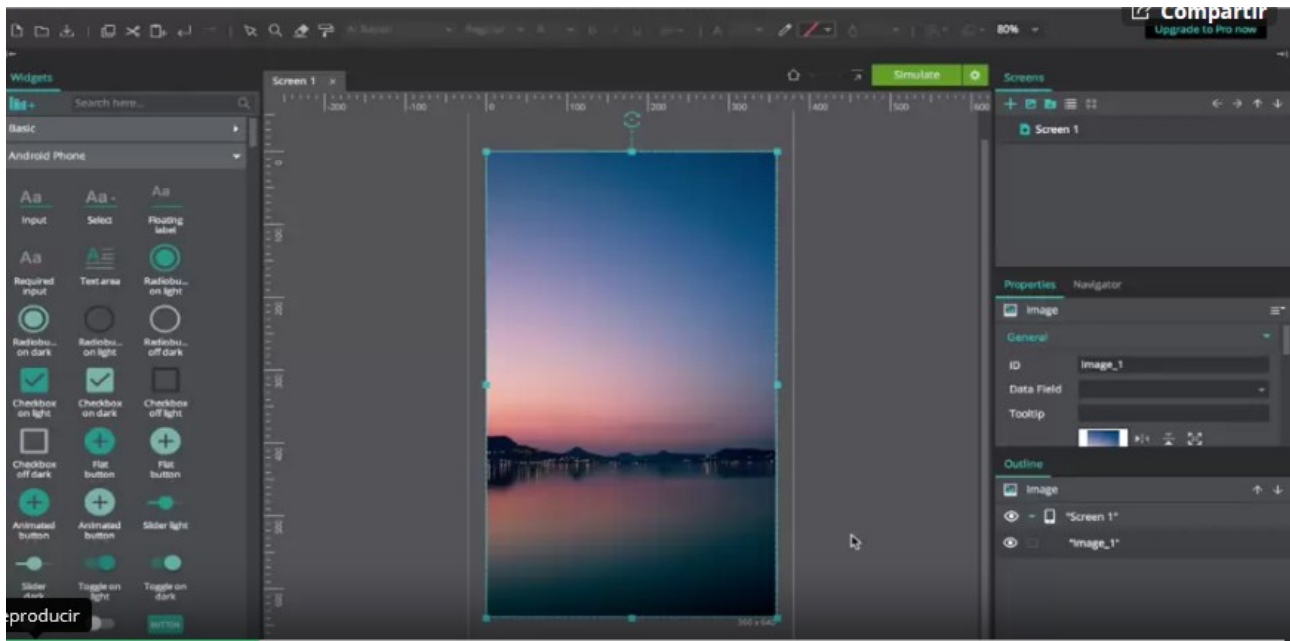
Como observamos nos ha colocado un pequeño marco donde nos indica que, pues es el dispositivo, es el smartphone y toda esta área blanca podemos dibujar. En la parte izquierda tenemos todos los elementos que podemos incluir en una interfaz Android, y como ves ya vienen incluidos los elementos de material design.

Vienen barras, vienen botones, barras de progreso, calendarios, diálogos, etc.

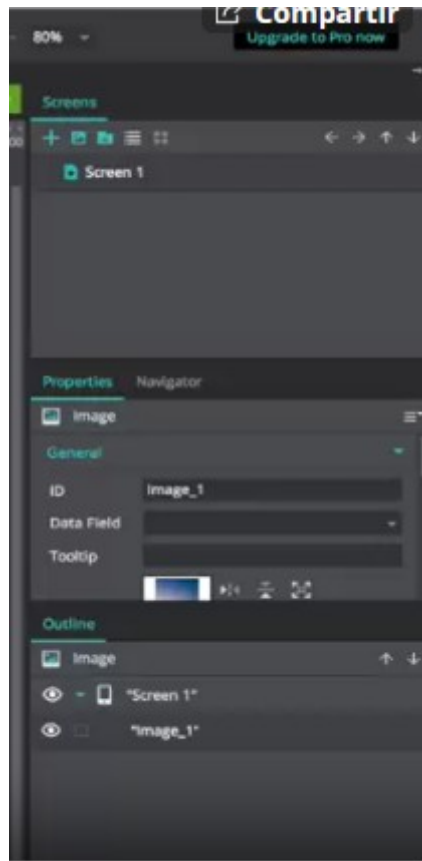
Lo primero que tenemos en nuestra maquetación que acabamos de hacer es una ventana de login y tenemos la bienvenida, la pantalla de bienvenida.

Bien, lo primero que observamos en nuestra maquetación es que tenemos como fondo una imagen, esa imagen yo ya he seleccionado una y le he llamado back city. Es simplemente una imagen que ya está recortada en la resolución que necesito y tiene un paisaje, un fondo muy bonito sobre una ciudad que está en el océano.

Bien, voy a acomodar esto para que encuadre perfectamente bien aquí.

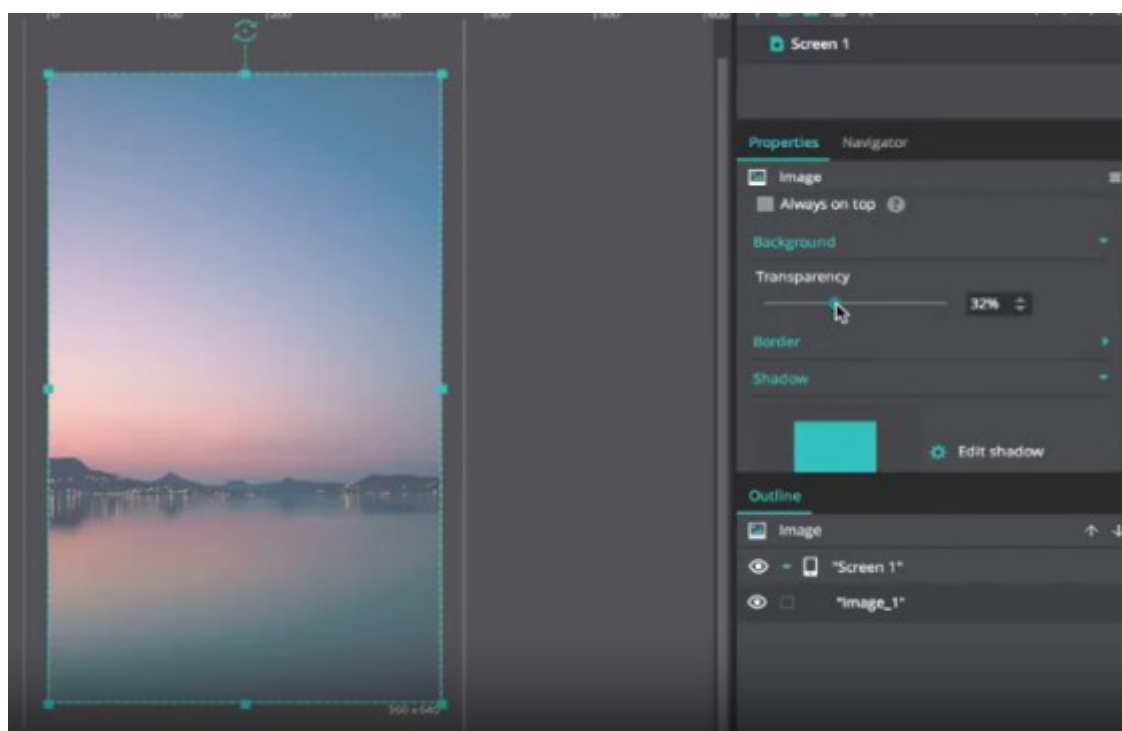


Así. Listo. Ahora si yo coloco esto a full a lo mejor mis elementos van a perderse un poco. Voy a colocarle un poco, voy a quitarle más bien un poco de opacidad y del lado derecho vamos a tener nuestra ventana de propiedades. En la parte superior estarán todas nuestras pantallas, todas nuestras screens que creemos. Y en la parte de hasta abajo donde dice outline ahí estarán, si pudiéramos asimilar esto con Photoshop pues sería como tu diagrama o sería más bien como tus capas, las capas que vas manejando cuando diseñas imágenes. Aquí se van a estar registrando todos los elementos que vamos sobreponiendo, y en ese orden de capas.



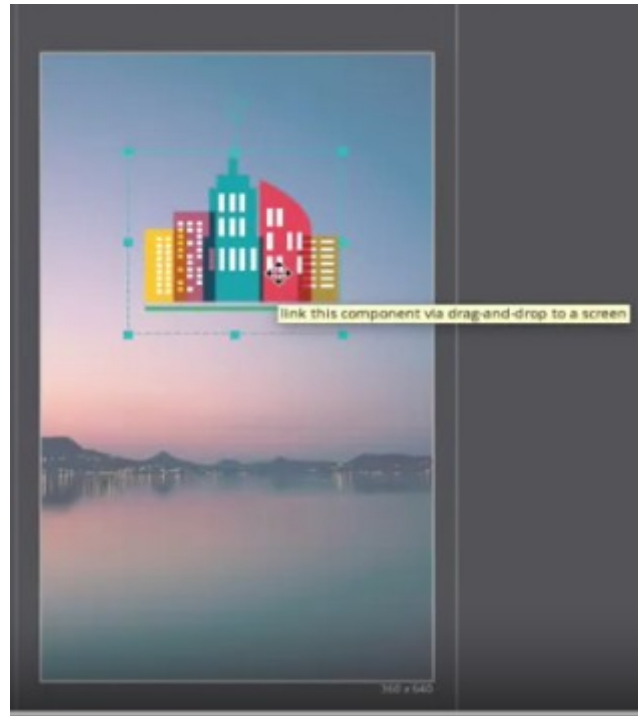
Entonces selecciono la imagen y vamos a las propiedades en background y la transparencia vamos a agregarle un poquito de transparencia para que esto se vea como con este efecto.

Este efecto que la verdad es muy común, ya encontrarnos con él y la idea es como quitar la intensidad a la imagen, bajárselo para que se vea así.



Lo siguiente es colocar nuestro logotipo.

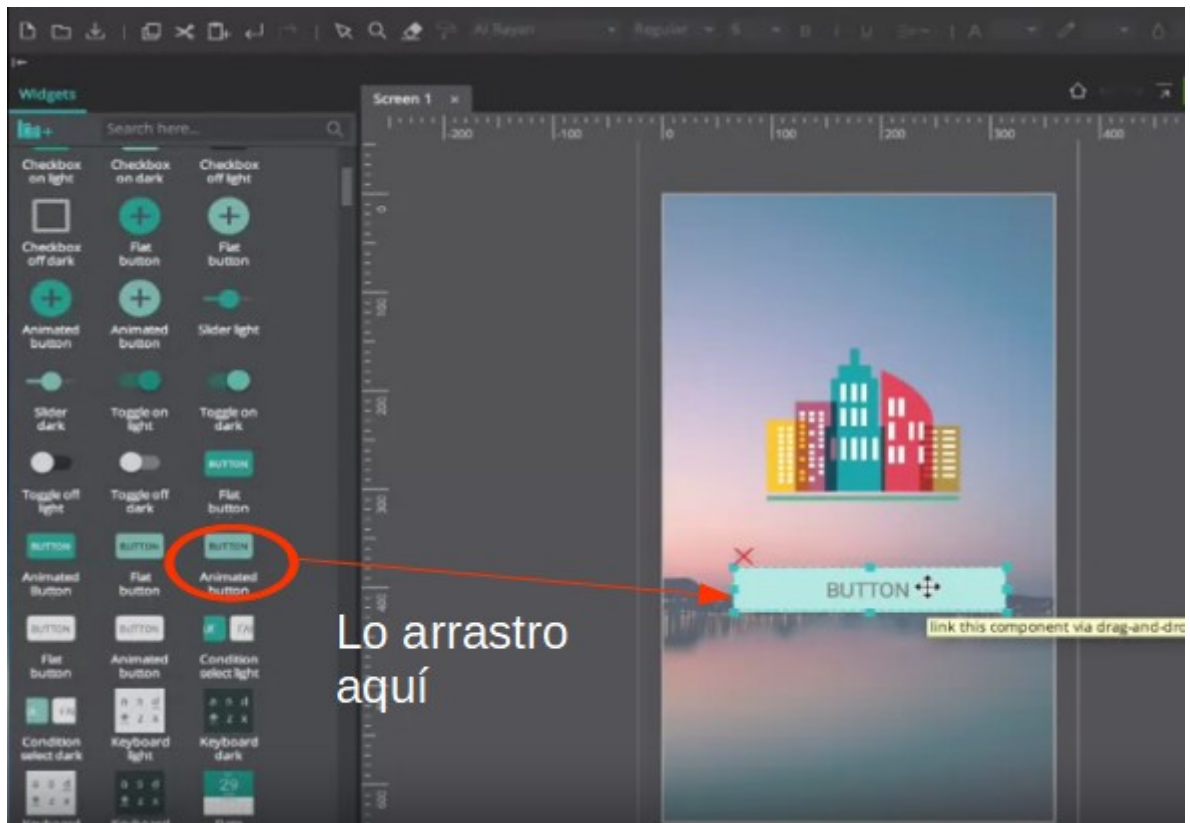
Yo ya tengo un logotipo por aquí creado control C, control V. Bien. Voy a acomodar nada más mi logotipo por aquí, de acuerdo al mock up debe estar centrado y como por aquí.



Lo siguiente que tenemos en nuestro mock up que hicimos en papel es un botón de acceso.

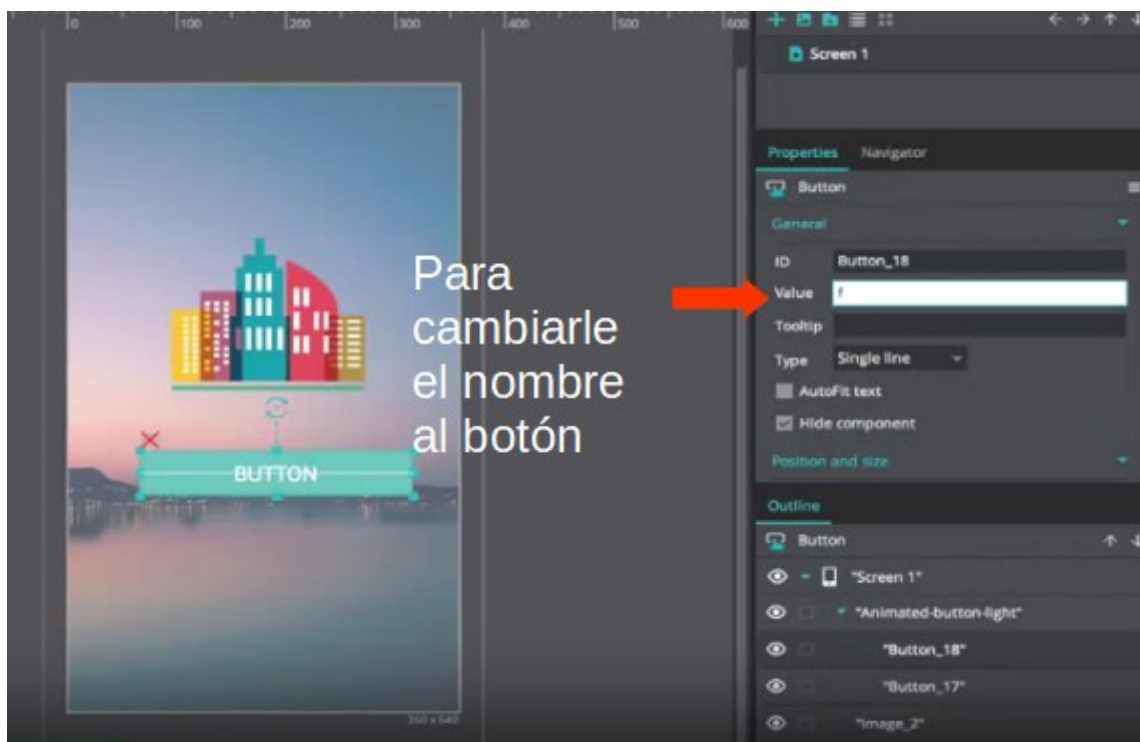
Aquí podríamos tomar incluso un botón animado. Yo aquí tengo uno simplemente lo arrastro y lo suelto.

Este botón es un botón, un botón dark animado, voy a escoger otro, este que está aquí que también es animado, es un botón light, este color me gusta más.



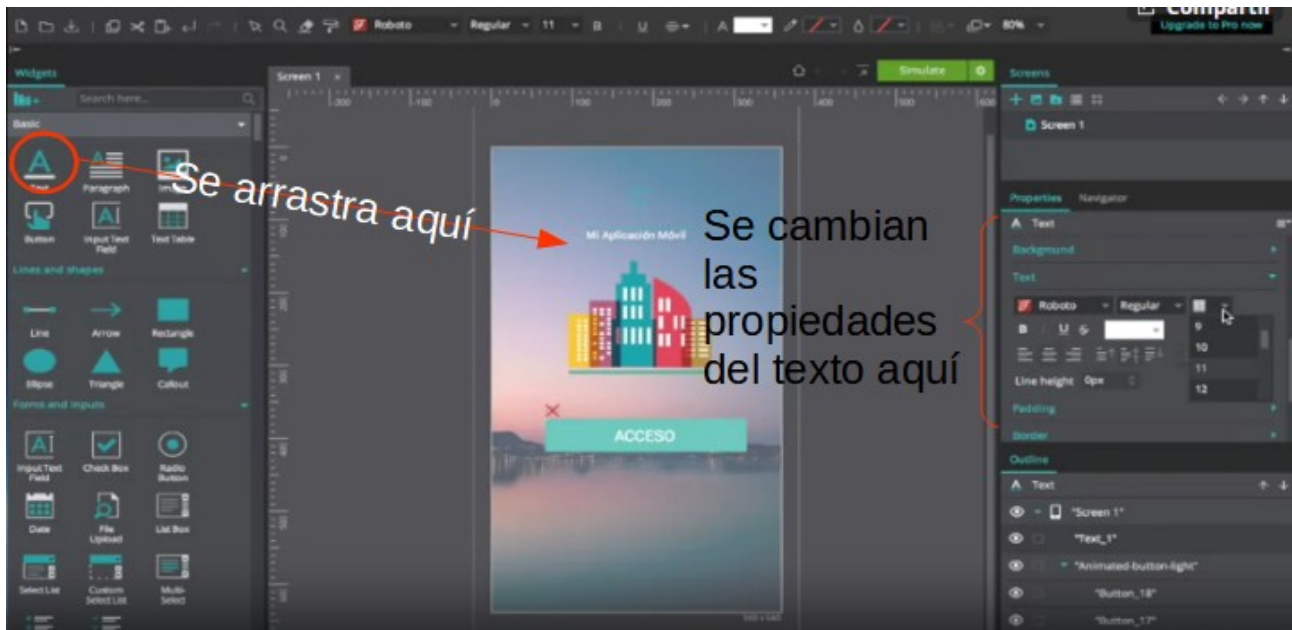
Lo acomodamos, lo alineamos bien para que esto se vea bien, y podemos colocar acá está el botón.

Vamos a cambiarle el nombre a ese botón o el contenido, damos doble clic en el botón, y vamos a acomodar y podemos colocar que diga acceso.



Seguramente esto te vas a topar en algún momento, listo, y es que es un poco complicado determinar exactamente en dónde es el texto, entonces vamos a quitar eso que dice botón y vamos a poner acceso. Acceso, listo, acceso.

Voy a colocarle nada más el título de aplicación, puedo aquí en basic buscar un texto, un texto que diga mi aplicación, mi aplicación móvil.



Mi aplicación móvil, lo centramos por aquí, lo acomodamos, y también a ese texto le puedo cambiar la transparencia, el background, puedo definir cómo está el texto, que esté centrado, además puedo decir que sea un texto regular por ejemplo.

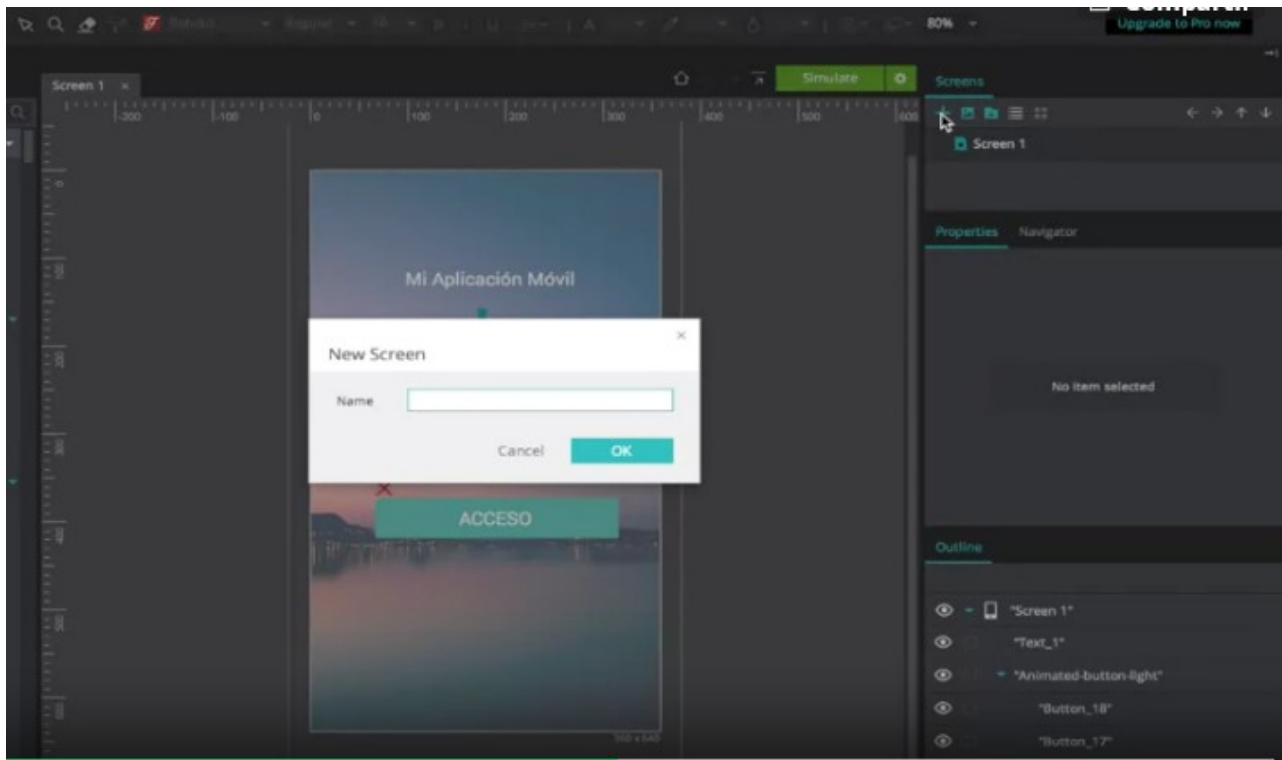
Vamos a ponerle un color blanco, y además vamos a aumentar el tamaño de ese texto. Que como observas puedo manipular todas las propiedades de estos, de los textos, los vamos a dejar así.

Ya se va viendo mejor esto. Mi aplicación móvil.

Bien, esta será nuestra pantalla de bienvenida. Vamos a guardar nuestro proyecto.

Aquí en maquetación podemos guardar maqueta android y listo.

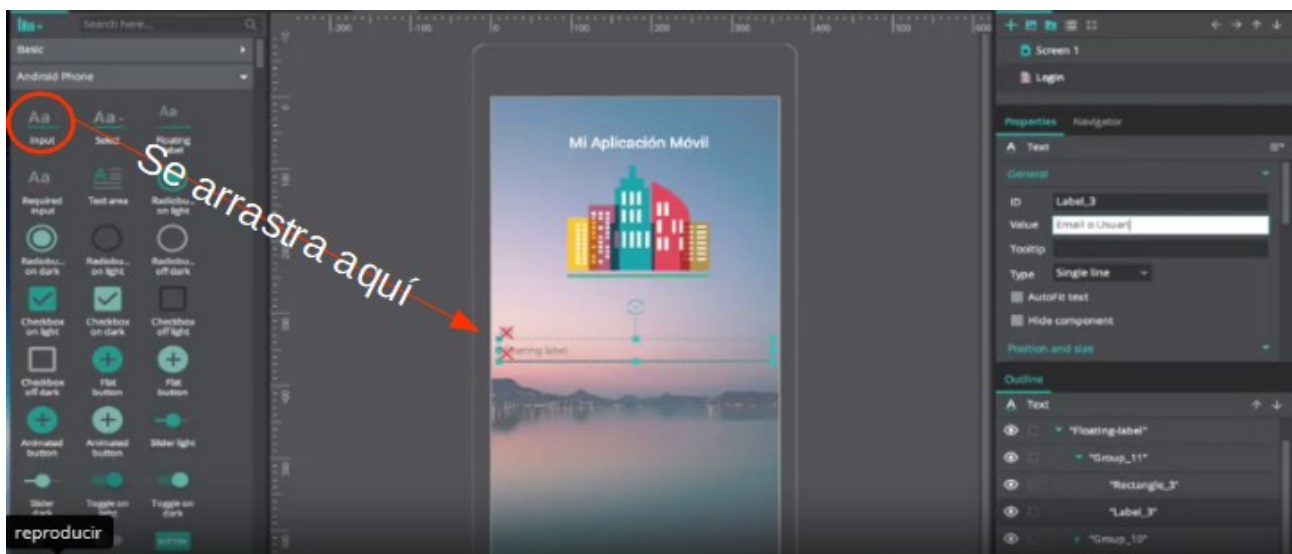
Bien, si queremos añadir una nueva pantalla, vamos a darle al botón de más y vamos a añadir el nombre de la pantalla. Esta pantalla se va a llamar login.



La idea es que el botón de acceso vaya inmediatamente al login. Entonces lo que haremos será linkear precisamente eso, entonces estamos diseñando nuestro login, nuestro login debe permanecer también con estos elementos, mi aplicación.

Y ahora vamos a colocar nuestros 2 inputs, nuestros 2 inputs serán unos inputs de material design. Entonces voy a colocar este por aquí, que también es un poco complicado poder seleccionar el texto que queremos editar.

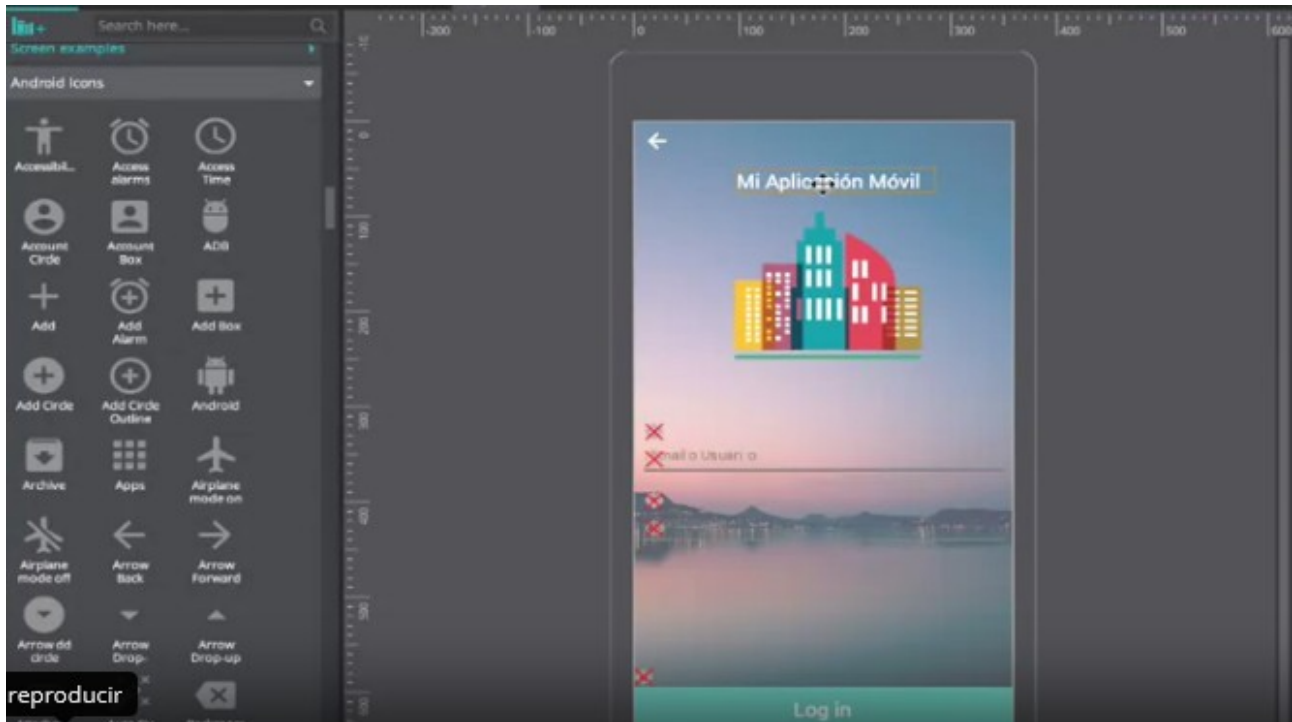
La idea es que esa floating label no aparezca, aquí está. Y vamos a ponerle email o usuario.



Bien y ahora vamos a colocar otro que este va a ser la contraseña.

Y por último, bueno, aún no hemos terminado, vamos a colocar nuestro botón animado en la parte inferior que es el footer, así, y este botón debe decir login.

Nos falta olvidé mi contraseña, nos falta la flechita de atrás y por último el texto de sign up. Vamos a colocar la flechita de atrás. Casi en la última parte tenemos android icons.



Vamos a ponerlo por aquí. Y también podemos cambiarle el color a esas letras.

Vamos a poner el texto de registro, lo vamos a poner por aquí, que recuerda esto es un link, no será un botón como tal, se comportará como un botón pero no es un botón es un enlace.

Y por último en el mismo tono por acá, y con esto hemos terminado ya nuestra maqueta.

Podemos simular esto. Estoy en la pantalla de bienvenida y puedo darle clic en el botón de simulate.