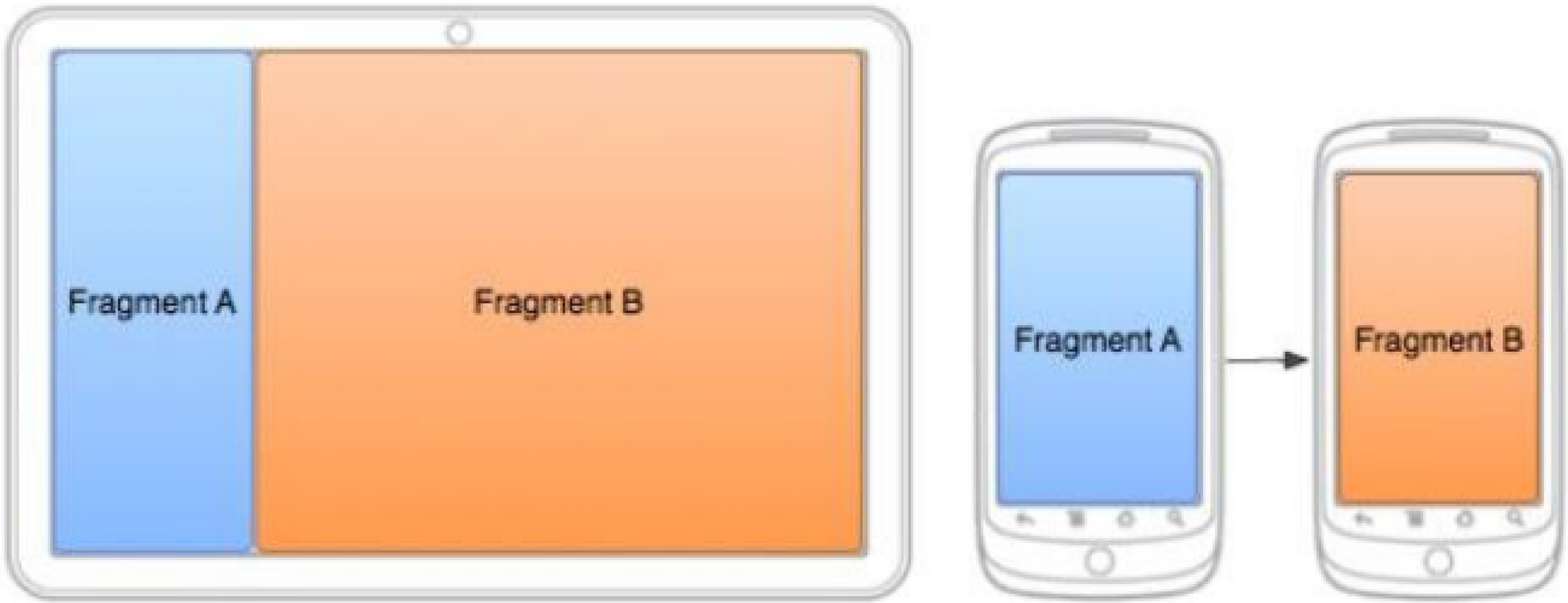
COMPONENTES DE UNA APLICACIÓN

Los componentes de una aplicación en Android Studio son las partes o elementos que componen la aplicación.

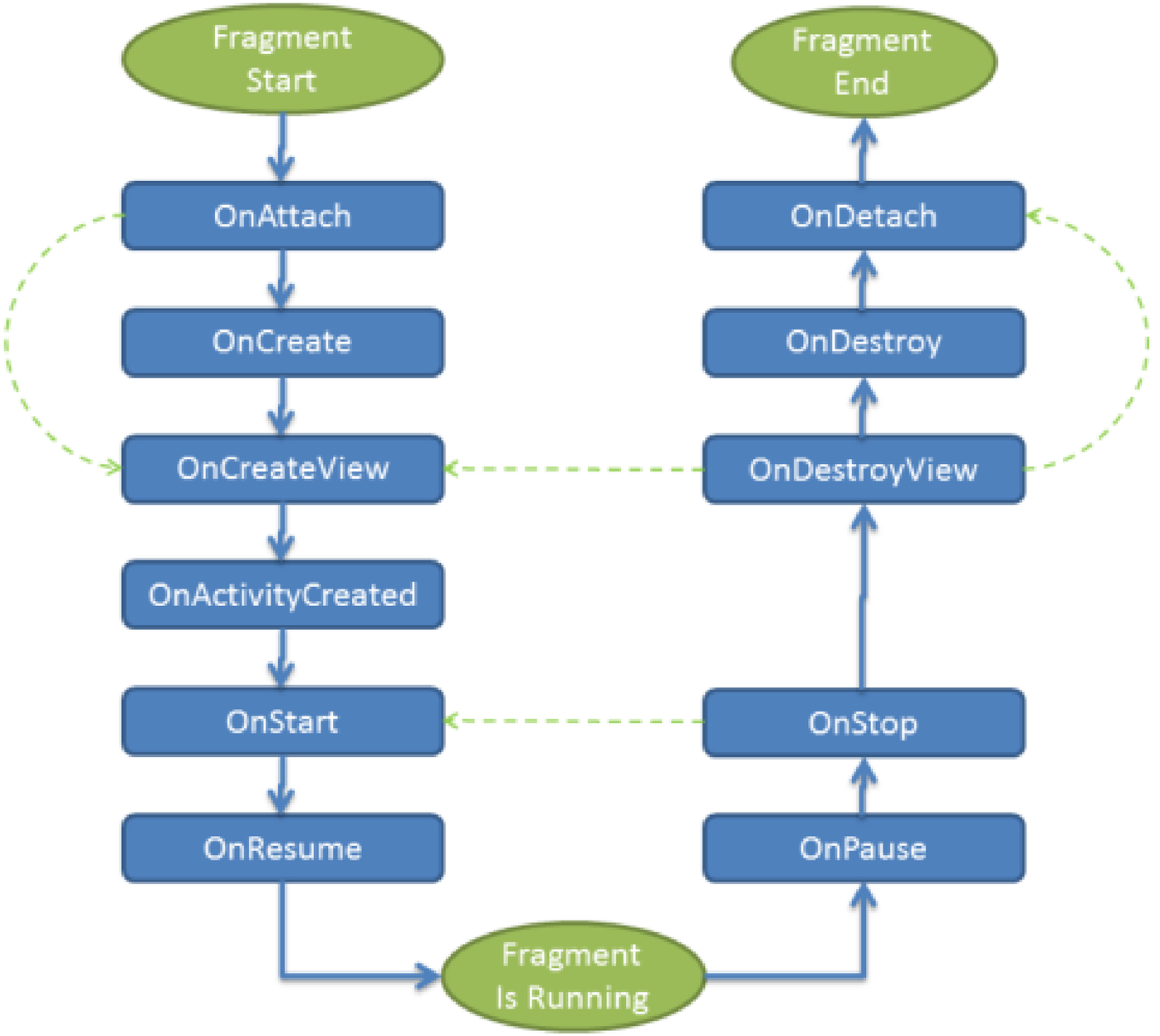
- Activities: las actividades son las pantallas o ventanas de la aplicación.
- Views: los objetos vista son los componentes visuales que representan elementos de la interfaz de usuario con los que los usuarios interactúan. Son utilizados para mostrar información, permitir la entrada de datos y realizar acciones en la aplicación. Text View, Edit Text, Button...
- Fragments: los framentos son componentes más pequeños que pueden ser utilizados en una actividad para crear interfaces de usuario más flexibles .
- Services: los servicios son componentes que se ejecutan en segundo plano para realizar tareas de larga duración sin inferir con la interfaz de usuario.
- Broadcast Receivers: los receptores de emisión permiten que una aplicación responda a eventos o notificaciones del sistema o de otras aplicaciones
- Content Providers: los proveedores de contenido ofrecen una forma de compartir datos, como la información de contactos o la base de datos de una aplicación.
- Intents: las intenciones se utilizan para iniciar componentes y realizar acciones en una aplicación como abrir una actividad o enviar datos entre componentes.
- Resources: los recursos incluyen elementos como diseños XML, imágenes, archivos en cadena y otros recursos que se utilizan en la interfaz de usuario.
- AndroidManifest.xml: manifiesto de la aplicación es un archivo XML que describe la estructura y configuración de la aplicación, incluyendo la lista de componentes, permisos y otras configuraciones importantes de la aplicación.
- Context: contexto, se considera un componente esencial en Android, ya que facilita la interacción entre la aplicación y el sistema operativo, permitiendo así que la aplicación funcione correctamente y acceda a los recursos y servicios necesarios.
- ...

LOS FRAGMENTOS

Los Fragmentos (fragments). Son muy parecidos a las actividades. Son elementos que pueden asumir la apariencia de una actividad o simplemente de un elemento de la interfaz.



Los Fragmentos (fragments). El ciclo de vida e un fragmento es muy similar al de un activity



LOS SERVICIOS

En Android Studio, puedes crear diferentes tipos de servicios según las necesidades de tu aplicación.

- Foreground Service: servicio en primer plano, se utiliza cuando deseas que el servicio tenga alta prioridad y esté visible para el usuario.
  - Media Player Service: reproductores de música, una aplicación de reproductor de música utiliza un servicio en primer plano para reproducir música y mostrar controles de reproducción en la barra de notificaciones
  - Navigation Service: servicios de navegación, las aplicaciones de navegación utilizan servicios en primer plano para proporcionar instrucciones de navegación paso a paso y mostrar mapas en tiempo real mientras se conduce.
  - Audio recorder Service: grabadora de audio, las aplicaciones de grabación de audio utilizan servicios en primer plano para permitir a los usuarios grabar audio mientras muestran una notificación que indica que la grabación está en curso.

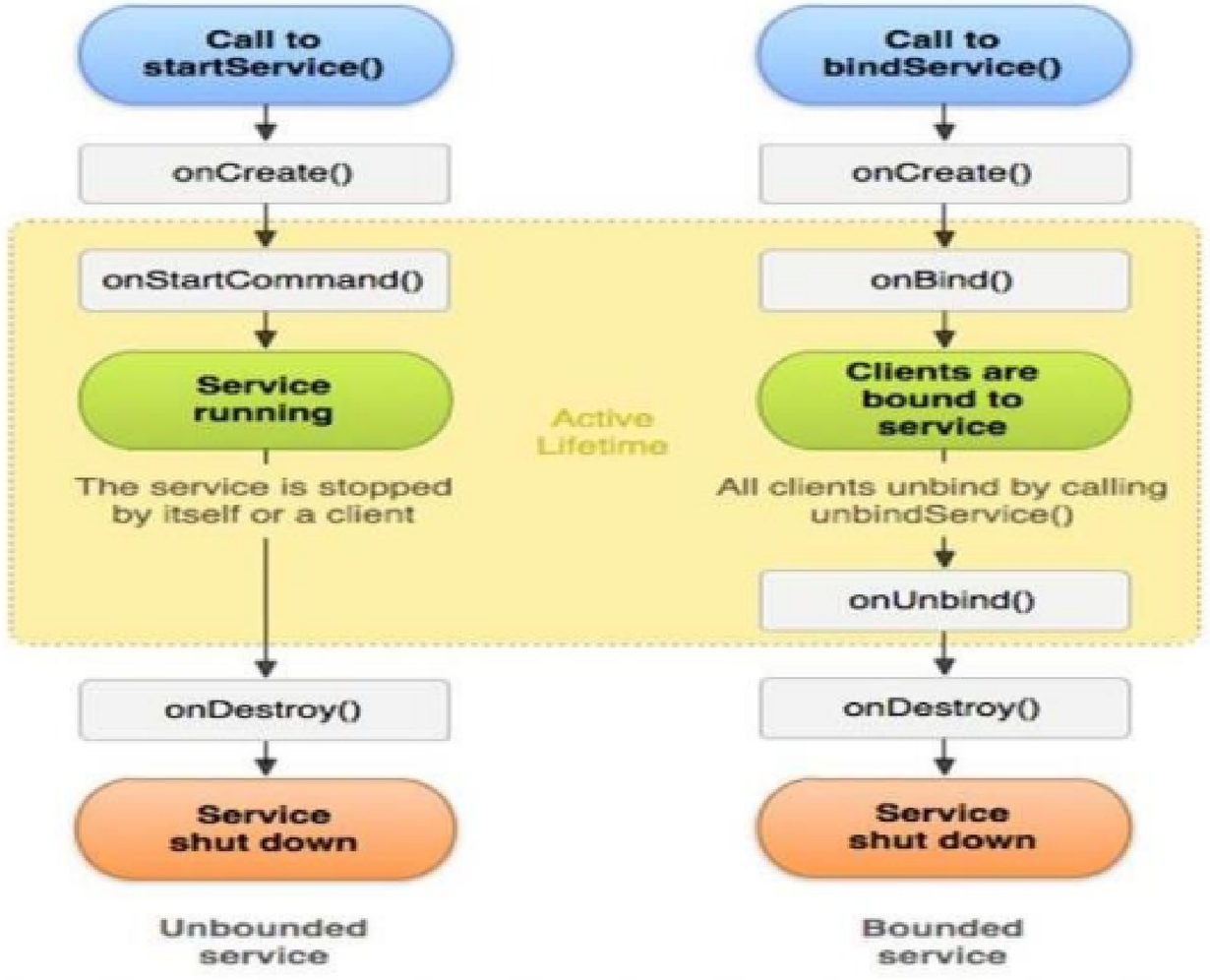
- VoiIP Calling Apps: aplicaciones de llamadas VoIP, las aplicaciones de llamadas de voz por internet (VoIP) utilizan servicios en primer plano para gestionar llamadas y notificar al usuario sobre llamadas entrantes y activas
- Live Chat Service: servicio de chat en vivo, las aplicaciones de chat en vivo utilizan servicios en primer plano para mantener una conexión en tiempo real con un servidor de chat y mostrar notificaciones de nuevos mensajes.
- Fitness Tracking: monitoreo de actividad física, las aplicaciones de seguimiento de actividad física utilizan servicios en primer plano para rastrear, contar los pasos y mostrar información en tiempo real sobre la actividad física.
- Food Delivery Service: servicio de entrega de alimentos, las aplicaciones de entrega de alimentos utilizan servicios en primer plano para actualizar en tiempo real el estado de los pedidos y mostrar notificaciones de estado de la entrega.
- ...

- Background Service: servicio en segundo plano, estos servicios se ejecutan en segundo plano y no son visibles para el usuario. Pueden realizar tareas en segundo plano, como cargar datos o realizar procesamientos, sin mostrar notificaciones permanentes. No requieren la atención constante del usuario y no bloquean la interfaz de usuario.
    - Location Updates: actualización de geolocalización, aplicaciones de seguimiento en tiempo real, utilizan servicios en segundo plano para actualizar la ubicación del usuario.
    - Notification Updates: actualización de notificaciones, las aplicaciones utilizan servicios en segundo plano para actualizar notificaciones de clima, noticias o recordatorios, sin necesidad de que la aplicación esté abierta.
    - File Downloading: descarga de archivos, puedes utilizar un servidor en segundo plano para descargar archivos grandes, sin interrumpir la experiencia del usuario.
    - ...

- Bound Service: servicio de vinculación, se utiliza para permitir la comunicación entre componentes de la aplicación como actividades, fragmentos u otros servicios. Es útil cuando deseas que varias partes de tu aplicación se comuniquen entre sí.
    - Servicio de Impresión de Documentos: puedes implementar un servicio de vinculación para enviar documentos a una impresora y permitir que diferentes partes de la aplicación (como actividaes o fragmentos) compartan el acceso al servicio de impresión.
    - Aplicación de navegación GPS: puedes utilizar un servicio de vinculación para proporcionar información de navegación desde un servicio en segundo plano a una actividad principal que muestra el mapa y las direcciones.
    - Gestión de Sesiones de Usuario: puedes usar un servicio de vinculación para administrar la información de la sesión del usuario (como la autenticación y la gestión de permisos) y permitir que varias partes de la aplicación accedan a esa información.
    - ...

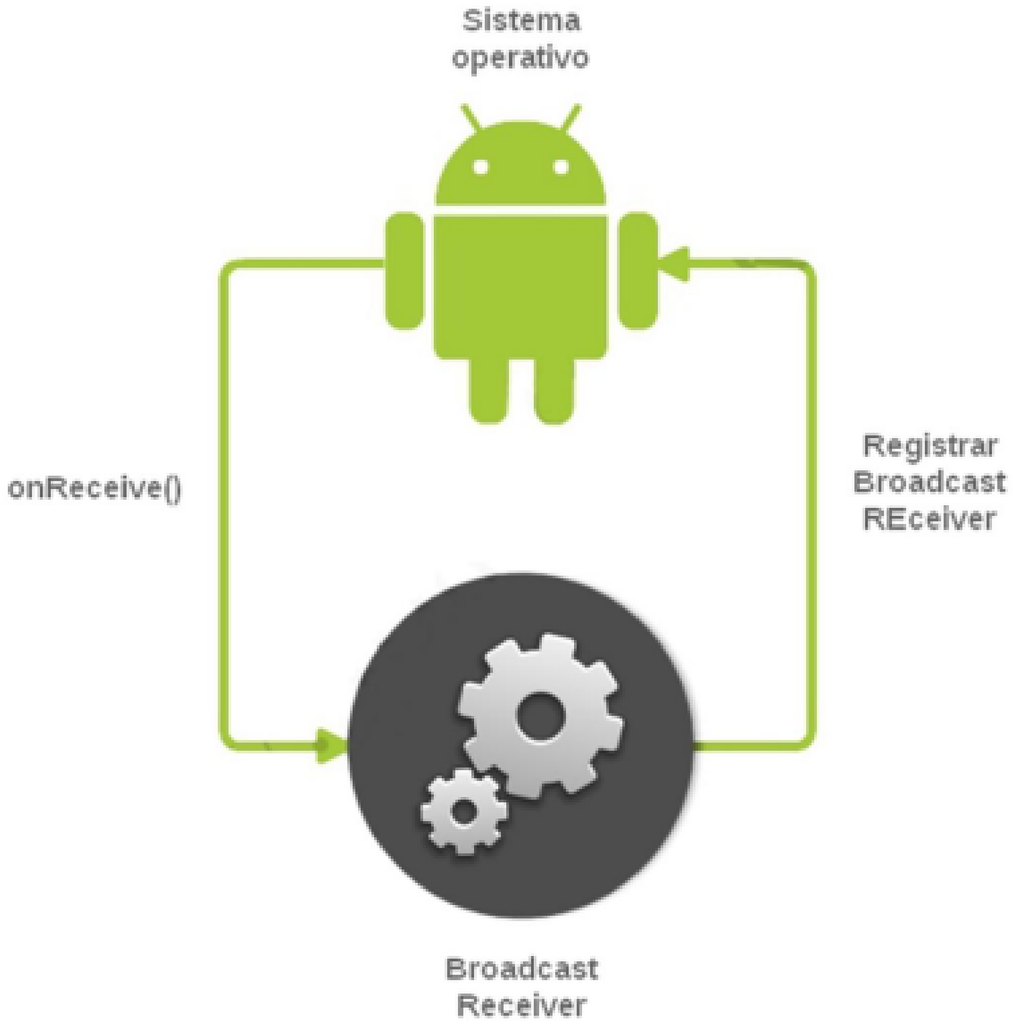
- Remote Service: servicio remoto, estos servicios permiten la comunicación entre diferentes procesos de aplicaciones. Son útiles cuando deseas que aplicaciones separadas se comuniquen entre sí.
- ...

Servicios (services): El ciclo de vida de un servicio es mucho más sencillo que el de una actividad.



RECEPTORES DE DIFUSIÓN

Receptores de difusión (broadcast receivers): Son componentes que permanecen latentes a la espera de que el sistema operativo active una notificación de difusión a la que están suscritos.



Ejemplos

PROVEEDORES DE CONTENIDO

Son componentes esenciales que permiten compartir y acceder a datos estructurados en una aplicación con otras aplicaciones. Ofrecen una interfaz estandarizada para iteractuar con los datos, lo que permite a otras aplicaciones leer, escribir o realizar consultas en esos datos de manera segura

- Agenda de contactos: el proveedor de contenidos de la agenda de contactos almacena información de contactos y permite a otras aplicaciones acceder a los datos de contactos para mostrarlos en su interfaz o realizar acciones como llamadas,mensajes...
- Galería de imágenes: los proveedores de contenido de la galería de imágenes permiten a otras aplicaciones acceder a fotos y vídeos almacenados en el dispositivo para mostrarlos en una aplicación de visualización de imágenes o realizar ediciones.
- Calendario: los datos del calendario como eventos y recordatorios, se almacenan en un proveedor de contenido que permite a otras aplicaciones acceder y mostrar información de calendario.

- Historial de llamadas: el proveedor de contenido de llamadas almacena información sobre las llamadas entrantes y salientes, lo que permite a las aplicaciones de llamadas o marcación acceder a estos datos.
- Almacenamiento de documentos: los proveedores de contenido se utilizan para acceder a archivos o documentos almacenados en el dispositivo, lo que permite a las aplicaciones de visor de documentos y edición interactuar con ellos.
- Configuración del sistema: los proveedores de contenido también almacenan configuraciones del sistema, como preferencias y ajustes de usuario, que pueden ser accedidos y modifiadospor aplicaciones de configuración
- ...

Los proceedores de contenidos suelen utilizar bases de datos (SQLite) para estructurar y administrar datos de manera organizada.

INTENTS

Componente fundamental en el desarrollo de aplicaciones. Permite a las aplicaciones interactuar entre sí.

Permite comunicar dos componentes de una aplicación. Una forma sencilla de que una aplicación le pase a otra una pequeña información es a través de una intención. Sin embargo cuando se necesite compartir mayores cantidades de información habrá que recurrir a la creación de un proveedores de contenido.

Dos tipos:

- Impícito: no especifica un componente específici y permite que el sistema elija la mejor actividad o servicio para manejar la acción. Puede invocar a cualquier aplicación del dispositivo que sea capaz de realizarla. (abrir una URL)
- Explícito: Se utiliza para iniciar un componente específico de la aplicación identificándolo por su nombre de clase.

LOS RECURSOS

Se refiere a archivos, valores y otros elementos que se utilizan para definir y personalizar la interfaz de usuario y otros aspectos de una aplicación. Son fundamentales para mantener una estructura organizada en el desarrollo de aplicaciones

- Layout Resources: recursos de diseño, estos archivos XML definen la estructura y organización de las interfaces de usuario de la aplicación. Los diseños pueden ser utilizados por actividades, fragmentos y otros componentes para defirnir cómo se muestran por pantalla.
- String Resources: recursos de cadena, almacenan cadenas de texto utilizadas en la aplicación, lo que facilita la internacionalidad y la gestión de texto en diferentes idioma. En lugar de codificar cadenas directamente en el código, se almacenan en archivos XML de recursos de cadena. (Ampliaremos esta información en las siguientes diapositivas).

- Drawable Resources: recursos de dibujo, estos recursos contienen inágenes, iconos y otros elementos gráficos utilizados en la interfaz de usuario. Puede ser en formato raster (como PNG) o vectorial (como SVG) (ampliaremos esta información en las siguientes diapositivas).
- Color Resources: recursos de color, define paletas de colores que se utilizan en la aplicación, lo que facilita la coherencia de la apariencia visual. (ampliaremos esta información en las siguientes diapositivas)
- Style and Theme Resources: recursos de estilo y tema. Estos recursos permiten definir estilos personalizados para elementos de la interfaz de usuario y aplicar temas a la aplicación para lograr una apariencia visual específica. (ampliaremos esta información en las siguientes diapositivas).
- Dimension Resources: recursos de dimensión, alamcena valores dimensionales, como márgenes, rellenos y tamaños, que se utilizan en la definición de la interfaz de usuario. Usar recursos de dimensión en lugar de valores codificados directamente en el código es una buena práctica ya que facilita la gestión y la coherencia de la aprariencia de tu aplicación. (ampliaremos esta información es las siguientes diapositivas)

- Animation Resources: recursos de diseño de animación, se utilizan para definir animaciones que se aplican a elementos de la interfaz de usuario, como transiciones y efectos visuales.
- Raw Resources: recursos de sonido y multimedia, almacenan archivos de audio, video u otros medios que se utilizan en la aplicación.
- Layout Resources for Fragments: recursos de diseño de fragmentos, específicos para fragmentos. Definen la estructura de los fragmentos que se utilizan en las actividades.
- ...

El uso de recursos en lugar de codificar datos y elementos directamente en el código, facilita la personalización, el mantenimiento y la internacionalización de una aplicación,

ANDROIDMANIFEST.XML

Es un archivo esencial en el desarrollo de aplicaciones para la plataforma Android. Alguno de los elementos clave que se encuentran en el archivo de manifiesto son:

- Declaración de componetes: en el archivo de manifiesto, declaras los componentes de tu aplicación, como actividades, servicios, receptores de emisión, proveedores de contenido...Esto permite al sistema Android saber qué componentes están presentes en tu aplicación
- Permisos: especificas los permisos que tu aplicación necesita para acceder a recursos o funcionalidades del dispositivo. Estos permisos deben ser declarados en el archivo de manifiesto y los usuarios deben otorgar su consentimiento antes de instalar la aplicación
    - Un ejemplo de permiso sería el permiso para acceder a la cámara del dispositivo

```
<uses-permission
android:name="android.permission.CAMERA"
 />
```

cuando el usuario instala la apliación, se le mostrará una lista de permisos que la aplicación solicita. El usuario debe otorgar o denegar el permiso individualmante

- Configuración de Actividades: define las actividades de tu aplicación, incluyendo su nombre, icono, etiqueta de la aplicación y la actividad principal que se inicia cuando se abre la aplicación.
- Filtros de Intents: Puedes configurar filtros de los intents para que tu aplicación responda a eventos específicos, como la recepción de una llamada entrante, una notificación de alarma o una activación de un widget.
- Versión de la aplicación: especificas la versión de tu aplicación, así como el nombre del paquete de la aplicación, que ha de ser único para cada aplicación.
- Compatibilidad con dispositivos: define la versión mínima de Android que tu aplicación requiere y las características del hardware que son necesarias para su funcionamiento.
- Iconos y Etiquetas: especificas los iconos y las etiquetas que se muestran en el lanzador de la aplicación y en la lista de aplicaciones.

- Recursos de Proveedor de Contenido: si tu aplicación proporciona datos que deben ser compartidos con otras aplicaciones, puedes declarar proveedores de contenido en el archivo de manifiesto.
- Actividades de Servicios en Primer Plano: si tu aplicación utiliza servicios en primer plano, debes declararlos en el manifiesto y proporcionar información sobre su notificación permanente

```
<service

android:name="com.ejemplo.MiServicioEnPr
imerPlano"
    android:exported="false">
    <intent-filter>
        <action
android:name="android.intent.action.MAIN
" />
    </intent-filter>
</service>
```

En el ejemplo anterior, se declara un servicio en primer plano llamado “MiServicioEnPrimerPlano”. Este servicio puede mostrar notificaciones permanentes y realizar tareas visibles para el usuario.

- Actividades de Acceso a Recursos: especificas las configuraciones necesarias para acceder a recursos específicos, como el acceso a la red.

```
<uses-permission
android:name="android.permission.INTERNE
T" />
```

declararías el permiso de internet en el archivo de manifiesto de la siguiente manera:

- ...

El archivo de manifiesto es esencial para la administración, seguridad y comportamiento de una aplicación de Android. Cada aplicación debe tener uno y este se encuentra en el directorio raiz del proyecto.

Otros permisos y servicios que puedes declarar en el archivo de manifiesto de tu aplicación según las funcionalidades específicas que requieras

Es importante recordar que, antes de declarar permisos en el archivo de manifiesto, debes asegurarte de que realmente necesitas esos permisos y que los solicitas de manera apropiada en tu aplicación. Además, ten en cuenta que algunos permisos pueden requerir la aprobación del usuario cuando instala la aplicación o cuando la aplicación intenta usar esos permisos. La solicitud de permisos debe ser transparente y justificada para el usuario.

- Permisos de Almacenamiento Externo: para acceder y escribir en el almacenamiento externo, puedes declarar el permiso: `**android.permission.WRITE_EXTERNAL_STORAGE**` para escribir. `**android.permission.READ_EXTERNAL_STORAGE**` para leer.
    - Se alienta a los desarrolladores a utilizar el almacenamiento interno de la aplicación o el almacenamiento específico de la aplicación.
    - Para acceder a un directorio específico en el almacenamiento externo en versiones más recientes de Android (10 y posteriores), puedes utilizar el método getExternalFilesDir() para obtener un directorio en el almacenamiento externo que está asociado con tu aplicación.
    - El uso de getExternalFilesDir() garantiza que tu aplicación almacene datos en un directorio específico de la aplicación en el almacenamiento externo, lo que es más seguro y cumple con las restricciones de privacidad de Android.
    - Recuerda que debes solicitar los permisos apropiados en el archivo AndroidManifest.xml y, si es necesario, solicitar permisos de tiempo de ejecución para acceder al almacenamiento externo, incluso cuando utilices métodos como getExternalFilesDir().

```
File externalFilesDir = getExternalFilesDir(null); // null indica el directorio raíz de la aplicación
if (externalFilesDir != null) {
    File myFile = new File(externalFilesDir, "miarchivo.txt");
try { FileWriter writer = new FileWriter(myFile);
    writer.write("Contenido del archivo");
    writer.close();
}
 catch (IOException e) {
    e.printStackTrace(); }
}
```

```
File externalFilesDir = getExternalFilesDir(null);
// null indica el directorio raíz de la aplicación
if (externalFilesDir != null) {
    File myFile = new File(externalFilesDir,
"miarchivo.txt");
try { FileWriter writer = new FileWriter(myFile);
    writer.write("Contenido del archivo");
    writer.close();
}
 catch (IOException e) {
    e.printStackTrace(); }
}
```

`File externalFilesDir = getExternalFilesDir(null);`: Esta línea obtiene una referencia al directorio específico de archivos de tu aplicación en el almacenamiento externo. `getExternalFilesDir(null)` es un método proporcionado por la clase `Context` en Android que devuelve una ruta al directorio de archivos específico de la aplicación en el almacenamiento externo. El argumento `null` indica que se está solicitando el directorio raíz de la aplicación.

`if (externalFilesDir != null) {`: Esta línea verifica si `externalFilesDir` se ha inicializado correctamente y no es nulo. Si es nulo, significa que no se pudo obtener el directorio de archivos externo, lo que podría ocurrir si no se otorgan los permisos adecuados o si el almacenamiento externo no está disponible.

`File myFile = new File(externalFilesDir, "miarchivo.txt");`: Aquí se crea un objeto `File` llamado `myFile` que representa un archivo en el directorio de archivos externo. Se especifica el nombre del archivo como "miarchivo.txt". Esto no crea el archivo físicamente en el almacenamiento, solo crea una representación del archivo en la memoria.

`FileWriter writer = new FileWriter(myFile);`: Esta linea crea un objeto `FileWriter` que se utilizará para escribir en el archivo `myFile`. Esto abre el archivo para escritura.

`writer.write("Contenido del archivo");`: Aquí se escribe la cadena "Contenido del archivo" en el archivo `myFile`.

`writer.close();`: Se cierra el `FileWriter` para finalizar la escritura en el archivo.

`} catch (IOException e) { e.printStackTrace(); }`: Este bloque `catch` maneja cualquier excepción de tipo `IOException` que pueda ocurrir al intentar escribir en el archivo. En caso de que ocurra un error, se imprimirá la traza de error en la consola.

- Permisos de Ubicación: si tu aplicación necesita acceder a la ubicación del dispositivo, puedes declarar el permiso
  `**android.permission.ACCESS_FINE_LOCATION**` para una ubicación precisa
  `**android.permission.ACCESS_COARSE_LOCATION**` para una ubicación aproximada.
- Permisos de Cámara y Micrófono:
  `**android.permission.CAMERA**`para acceder a la cámara
  `**android.permission.RECORD_AUDIO**`para acceder al micrófono
- Permisos de Contactos: si tu aplicación necesita acceder a los contactos del dispositivo
  `**android.permission.READ_CONTACTS**` para acceso de solo lectura
  `**android.permission.WRITE_CONTACTS**` para acceso de escritura
- Permisos de SMS: Si tu aplicación necesita enviar o recibir mensajes de texto,
  `**android.permission.SEND_SMS**` para enviar mensajes
  `**android.permission.RECEIVE_SMS**` para recibir mensajes.

- Permisos de Llamadas:
  `**android.permission.CALL_PHONE**`, para realizar llamadas
  `**android.permission.READ_PHONE_STATE**`para leer el estado del teléfono

  Ejemplos de la información que se puede obtener al leer el estado del teléfono son:
  - Número de teléfono: permite obtener el número de teléfono asociado al dispositivo. Esto puede ser útil para aplicaciones que necesitan verificar o mostrar el número de teléfono del usuario.
  - Estado de la red: proporciona información sobre si el dispositivo está conectado a una red móvil o a una red Wi-Fi, así como el tipo de red (2G, 3G, 4G, 5G, etc.).
  - ID del dispositivo: permite obtener un identificador único del dispositivo, que puede ser útil para rastrear dispositivos específicos o para fines de autenticación.
  - Estado de la llamada: permite determinar si el teléfono está en una llamada, en espera, fuera de servicio o en otros estados de llamada.
  - Información de la tarjeta SIM: permite acceder a información relacionada con la tarjeta SIM insertada en el dispositivo, como el número IMSI (International Mobile Subscriber Identity) y otros detalles.

    Es importante destacar que el acceso a esta información debe utilizarse con responsabilidad y solo cuando sea necesario para el funcionamiento de la aplicación, ya que se trata de datos sensibles y privados.
    Los usuarios deben otorgar su permiso para que la aplicación acceda a estos datos, y las aplicaciones deben solicitar este permiso en su manifiesto. Desde Android 10 en adelante, hay restricciones adicionales en el acceso a esta información para proteger la privacidad del usuario.

- Servicios de Notificaciones: si tu aplicación utiliza notificaciones push, puedes declarar servicios específicos de notificación en el manifiesto. (lo veremos cuando declaremos servicios en nuestra aplicación)

- Permisos de Bluetooth: para interactuar con dispositivos Bluetooth, `**android.permission.BLUETOOTH**` `**android.permission.BLUETOOTH_ADMIN**`
  - android.permission.BLUETOOTH:
    - Este permiso permite a la aplicación activar y desactivar el Bluetooth del dispositivo.
    - También permite a la aplicación buscar dispositivos Bluetooth cercanos y acceder a información básica de esos dispositivos, como su dirección MAC y nombre.
    - Es un permiso de nivel más bajo y se utiliza para realizar operaciones que no requieren control total sobre las capacidades Bluetooth del dispositivo.
  - android.permission.BLUETOOTH_ADMIN:
    - Este permiso proporciona un acceso más amplio y control sobre las capacidades Bluetooth del dispositivo.
    - Permite a la aplicación no solo activar y desactivar el Bluetooth, sino también establecer conexiones Bluetooth, administrar perfiles de conexión, configurar y desconectar dispositivos emparejados, y realizar otras operaciones más avanzadas relacionadas con Bluetooth.
    - Es un permiso de nivel más alto y se utiliza para aplicaciones que necesitan un mayor control sobre la funcionalidad Bluetooth del dispositivo.

  En resumen, android.permission.BLUETOOTH se utiliza para tareas más simples relacionadas con Bluetooth, como encender o apagar el Bluetooth y buscar dispositivos cercanos, mientras que android.permission.BLUETOOTH_ADMIN se utiliza para tareas más avanzadas que implican un mayor control sobre las operaciones Bluetooth, como el emparejamiento y la gestión de conexiones. El uso de estos permisos debe ser adecuado y justificado en función de las necesidades específicas de tu aplicación.

- ...

El contexto de la aplicación

Es una clase que proporciona información y acceso a recursos y servicios que permiten que una aplicación interactúe con el sistema operativo y realice tareas específicas.
El "Context" se utiliza en muchos lugares dentro de una aplicación para acceder a recursos, iniciar actividades, mostrar diálogos...
Es importante distinguir entre los dos tipos de contexto en una aplicación de Android:

- Context de la Aplicación: este es el contexto global de la aplicación y se obtiene típicamente llamando a getApplicationContext(). Representa el entorno de toda la aplicación y proporciona acceso a recursos y servicios que son comunes a todas las actividades y componentes de la aplicación. Este Context es más duradero y debe utilizarse con precaución para evitar pérdidas de memoria.
- Context de la Activity: cada Activity en la aplicación tiene su propio Context que se refiere específicamente a esa actividad. Se puede acceder a este contexto utilizando **this** dentro del código de la actividad. Este Context es más limitado en alcance y está destinado a tareas relacionadas con la interfaz de usuario y las operaciones específicas de la actividad en la que se encuentra.
    - Context context = this;

Es importante utilizar el contexto adecuado según el contexto de la tarea que deseas realizar en tu aplicación. Usar el contexto incorrecto puede llevar a problemas, como pérdida de memoria o referencias incorrectas. Por lo general, se recomienda utilizar el Context de la Activity cuando necesites acceder a recursos o servicios específicos de esa actividad y el Context de la aplicación cuando necesites recursos o servicios compartidos en toda la aplicación.