Elementos de la interfaz, navegación e interacción

Principales componentes de interfaz para aplicaciones nativas Android (Parte II)

{desafío} latam_

Utilizar elementos de interfaz de usuario básicos del entorno Android para la implementación de un prototipo de acuerdo a las especificaciones entregadas.

- Unidad 1: Ambiente de desarrollo y sus elementos de configuración.
- Unidad 2: Elementos de la interfaz, navegación e interacción.
- Unidad 3: Fundamentos de GIT y GitHub.

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Utilizar los principales componentes de interfaz distinguiendo su uso en un proyecto Android Studio.*

{desafío}
latam_

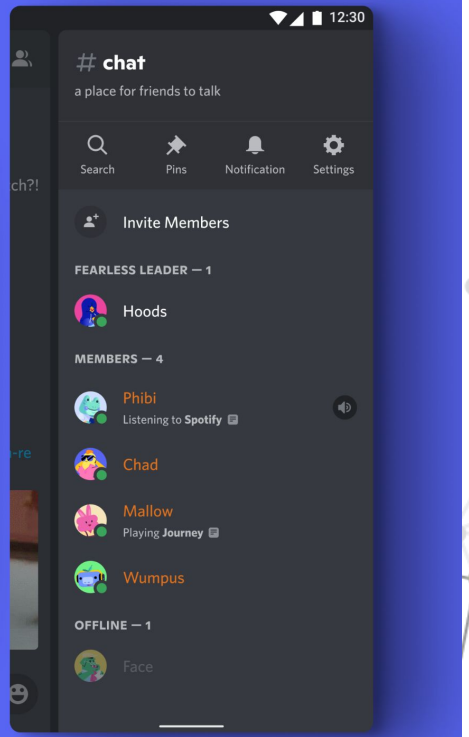
¡Identifica las vistas!

Descubre / identifica las vistas

Discord

Observando una imagen de Discord de Google Play, podemos inferir los componentes utilizados en el diseño de la app.

- ¿Cuáles componentes puedes identificar a partir de la imagen?



Fuente: [Discord en Google Play.]

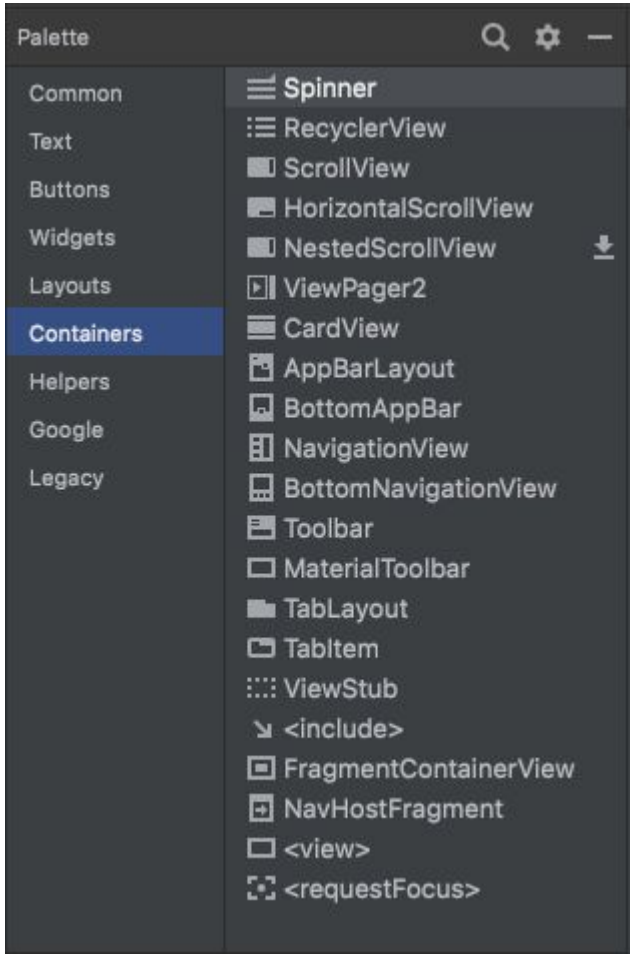
{desafío} latam_

/* Containers */

Containers

Como revisamos anteriormente, los containers sirven para agrupar vistas, mostrar elementos en listas, y también para agregar scroll a una porción de la pantalla.

¡No debes perder esta información de vista para lo que revisaremos en esta sesión!

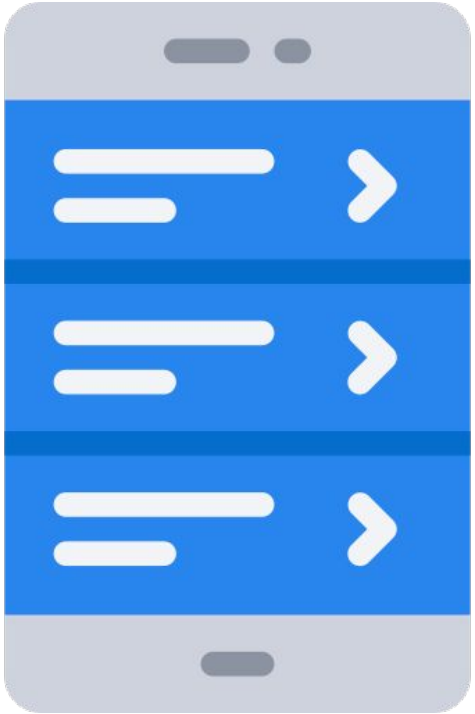


/* List */

Listados

Presentar listado de información

- Hay distintas formas de presentar listados de elementos o acciones en una app, y distintas formas de cómo enlazar la información con la vista.
- La responsabilidad de ListView o Spinner es mostrar y coordinar los elementos que se muestran.
- Por otro lado, la responsabilidad de qué hay que mostrar y cómo se ve cada elemento está en otra parte: Un archivo estático de elementos o un Adapter.



ListView

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="24dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Muestra una colección de vistas desplazables verticalmente, donde cada vista se coloca inmediatamente debajo de la vista anterior en la lista.

ListView

Atributos XML

android:divider	Drawable o color para dibujar entre las líneas
android:dividerHeight	Altura del divisor
android:entries	Referencia un recurso de listado con la información para la lista
android:footerDividersEnabled	Cuando está en falso, ListView no dibujará un divisor antes de cada footer
android:headerDividersEnabled	Cuando está en falso, ListView no dibujará un divisor antes de cada header

ListView + Arrays

Agregar información al ListView

- Crear el archivo de recursos res/values/arrays.xml
- En el archivo se define cada uno de los elementos.
- En la declaración de ListView se enlaza con el listado estático de elementos a mostrar usando **entries.**

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="items">
        <item>item 0</item>
        <item>item 1</item>
        <item>item 2</item>
        <item>item 3</item>
        <item>item 4</item>
        <item>item 5</item>
        <item>item 6</item>
        <item>item 7</item>
        <item>item 8</item>
        <item>item 9</item>
    </array>
</resources>
```

```xml
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="24dp"
        android:entries="@array/items"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

{desafío} latam_

ListView + Adapter

Agregar información al ListView

- Otra forma de agregar información a ListView es utilizando un adaptador.
- El adaptador tiene la información que se muestra en el listado
- Implementa la clase abstract BaseAdapter para poder obtener información de cada elemento en el listado
- getView(...) construye la vista de cada elemento del listado

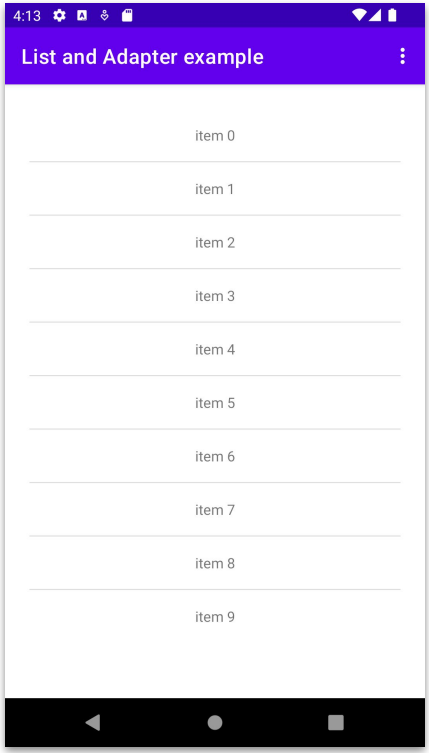
{desafío} latam_

```
private class MyAdapter extends BaseAdapter {
    private final List<String> values = new ArrayList<>();

    @Override
    public int getCount() {
        return values.size();
    }

    @Override
    public Object getItem(int i) {
        return values.get(i);
    }

    @Override
    public long getItemId(int i) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup
container) {
        if (convertView == null) {
            convertView =
getLayoutInflater().inflate(R.layout.list_item, container, false);
        }

        String textToShow = getItem(position).toString();
        ((TextView)
convertView.findViewById(R.id.textViewItem)).setText(textToShow);
        return convertView;
    }

    public void setValues(List<String> newValues) {
        values.clear();
        values.addAll(newValues);
    }
}
```

ListView + Adapter

Presentar listado de información

```
public void onViewCreated(@NonNull View view, Bundle
savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);

    // Se crea el adaptador y se asignan los valores
    MyAdapter adapter = new MyAdapter();
    adapter.setValues(getValues());

    // Se enlaza ListView con el adaptador
    binding.listView.setAdapter(adapter);
}

private List<String> getValues() {
    List<String> values = new ArrayList<>();
    for(int i=0; i<10; i++) {
        values.add("item " + i);
    }
    return values;
}
```



Fuente: ADL.

Usando un adaptador, la información no es estática como los arreglos

¿En qué caso usarías una adapter?

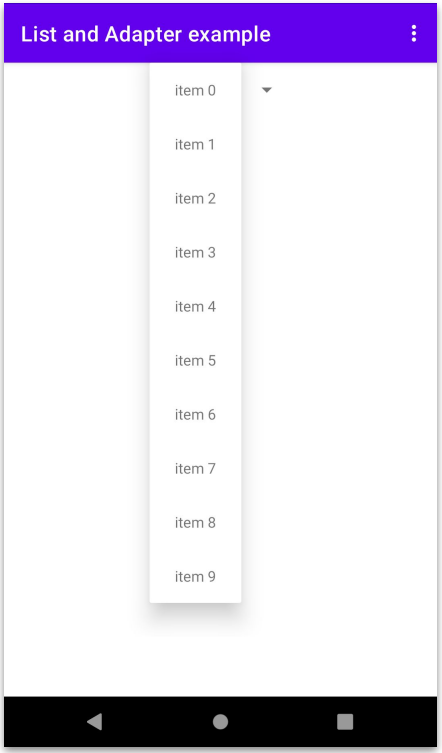
{desafío} latam_

Spinner + Adapter

Otra forma de presentar un listado de información

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".FirstFragment">

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Si cambiamos el tipo de ListView a Spinner se puede reutilizar el mismo adaptador.



Fuente: ADL.

{desafío} latam_

¿Qué tipo de vista de lista usarías para mostrar un listado de películas?

{desafío} latam_

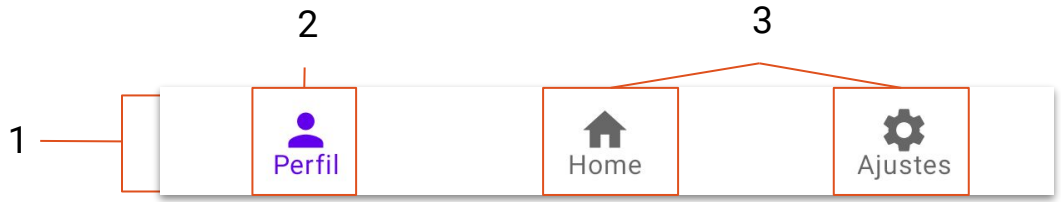
¿Qué tipo de vista de lista usarías para mostrar un listado para que el usuario pueda elegir su edad? ¿Usarías un adapter o un archivo de recursos?

{desafío} latam_

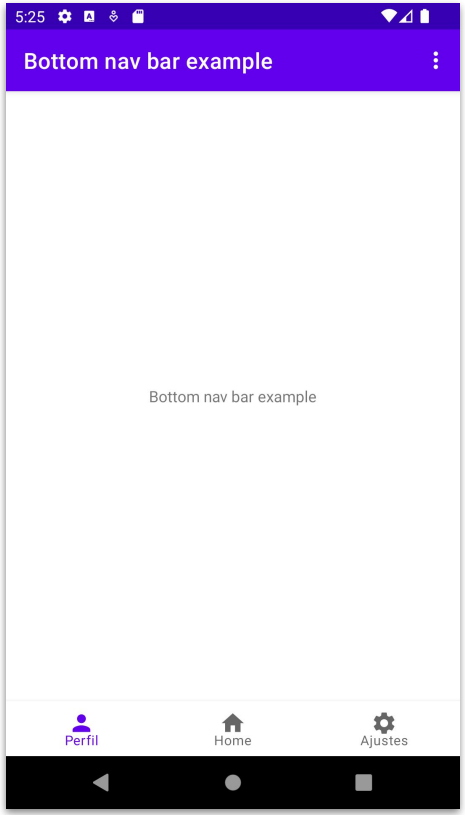
/* Bottom Navigation */

Bottom Navigation Bar

Las barras de navegación inferiores permiten el movimiento entre los destinos principales de una aplicación.



1. Contenedor
2. Icono activo con texto
3. Icono inactivo con texto



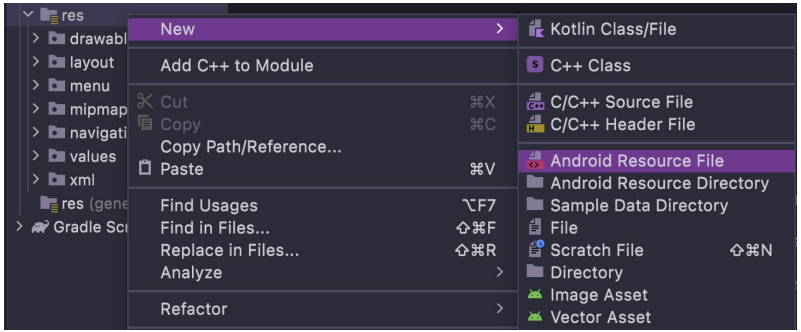
Fuente: ADL.

{desafío} latam_

Recurso de menú

Define los elementos del menú

Los elementos que se muestran en la barra se definen en un archivo de recursos de menú




```xml
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@id/profile nav item id"
        android:icon="@drawable/ic profile"
        android:title="@string/profile" />

    <item
        android:id="@id/home nav item id"
        android:icon="@drawable/ic home"
        android:title="@string/home"
        android:visible="true" />

    <item
        android:id="@id/settings nav item id"
        android:icon="@drawable/ic settings"
        android:title="@string/settings" />
</menu>
```

{desafío} latam_

Archivo de recursos

ID

- Es un ID de recurso único definido en XML.
- Usando el nombre que proporcionas en el elemento <item>, automáticamente se crea un número entero único en la clase R.java de tu proyecto, que puedes usar como identificador para los recursos de una aplicación

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <item name="profile nav item id" type="id" />
   <item name="home nav item id" type="id" />
   <item name="settings_nav_item_id" type="id" />
</resources>
```

Recuerda que los recursos de un ID no hacen referencia a un elemento de recurso real; solo representan una identificación única que puedes adjuntar a otros recursos o usar como un número entero único en tu aplicación.

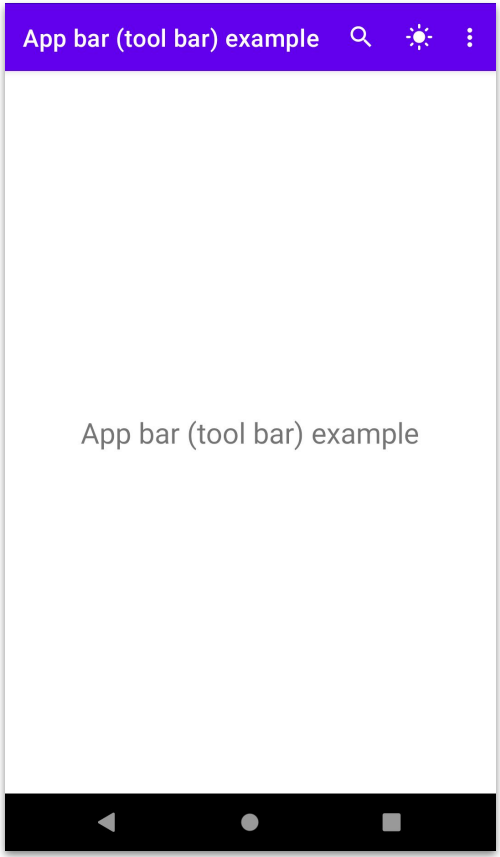
/* App bar */

Action bar (app bar)

Barra de acciones

- Entrega una estructura visual y elementos interactivos que son familiares para los usuarios.

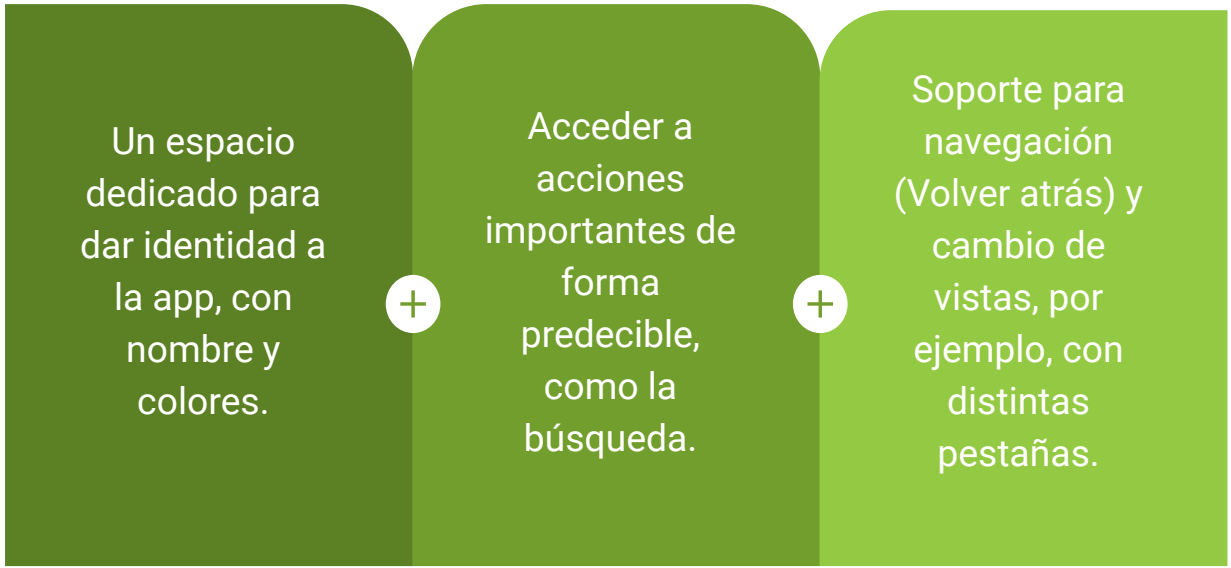
- Hace que nuestra app sea consistente con otras aplicaciones Android, permitiendo al usuario entender rápidamente cómo operar nuestra app.



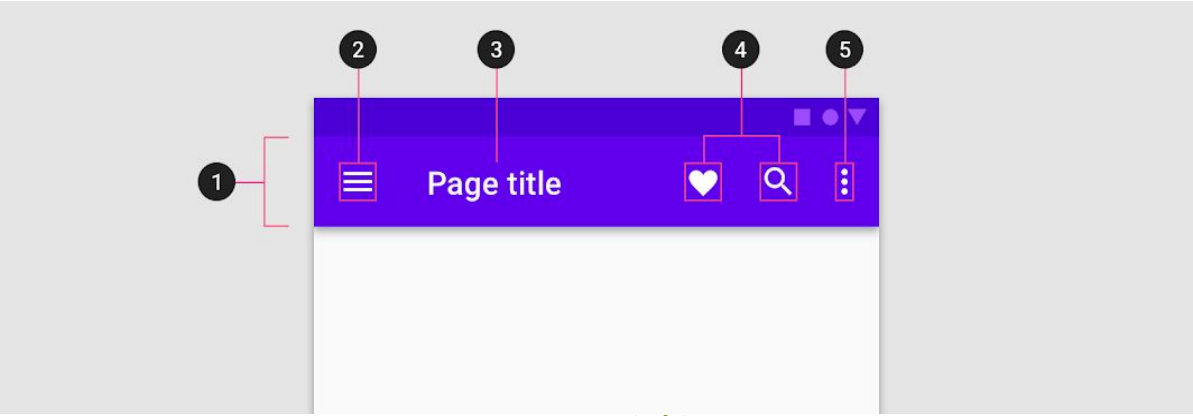
Fuente: ADL.

Action bar (app bar)

Tres funciones clave de app bar



Toolbar



Fuente: material.io

1. Contenedor
2. Icono de navegación (menú o flecha de navegación) **
3. Título **
4. Elementos de acción **
5. Menú de desbordamiento (elementos que no caben en la barra) **

** opcionales

{desafío} latam_

Recurso de menú

Define los elementos a mostrar

Los elementos que se muestran en la barra se definen en un archivo de recursos de menú.

```
<menu 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"
    tools:context="cl.dal.containersexample.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />

    <item
        android:id="@+id/action_search"
        android:title="@string/action_search"
        android:icon="@drawable/ic_search"
        app:showAsAction="ifRoom"/>

    <item
        android:id="@+id/action_like"
        android:title="@string/action_like"
        android:icon="@drawable/ic_like"
        app:showAsAction="ifRoom"/>
</menu>
```

{desafío} latam_

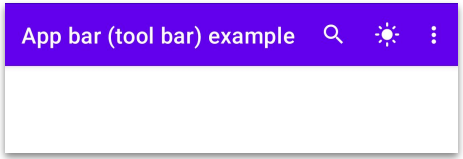
Recuerda que los recursos de un ID no hacen referencia a un elemento de recurso real; solo representan una identificación única que puedes adjuntar a otros recursos o usar como un número entero único en tu aplicación.

Asociar con la actividad

- Se agrega la barra personalizada al layout de la actividad.
- En onCreate se asigna la barra como la barra a utilizar.
- En onCreateOptionsMenu se infla el menú y se agregan los elementos a la barra si es que está presente.

```
<com.google.android.material.appbar.AppBarLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:theme="@style/Theme.ContainersExample.AppBarOverlay">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        app:popupTheme="@style/Theme.ContainersExample.PopupOverlay"
/>

</com.google.android.material.appbar.AppBarLayout>
```

```
public class MainActivity extends AppCompatActivity {

    private ActivityMainBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        …
        // se asocia la toolbar declarada como la action bar
        setSupportActionBar(binding.toolbar);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // infla el menú y agrega los elementos a la action bar
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }
}
```



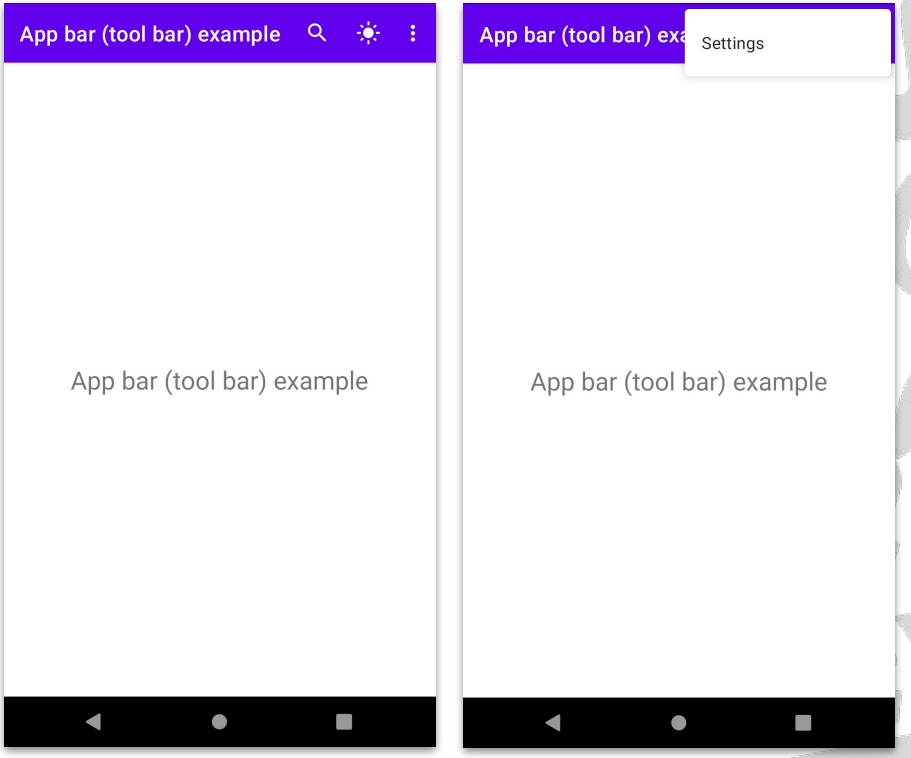
Fuente: ADL.

Ejercicio

Actividad

- ¿Por qué algunos elementos del menú están presentes en la barra y otro dentro del menú de desbordamiento?
- ¿Qué configuración deben tener todos los elementos para que siempre estén presentes en la barra?

`android:showAsAction`



Fuente: ADL.

{desafío} latam_

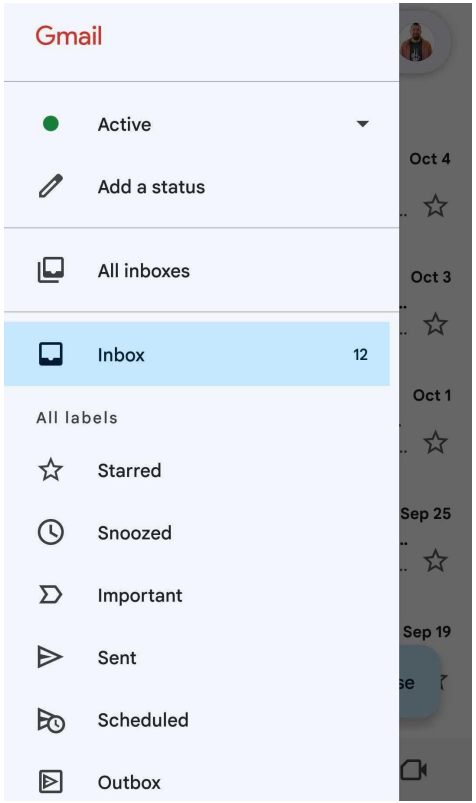
/* Navigation Drawer */

Navigation Drawer

¿Qué es?

- Permiten acceder a destinos y funcionalidades de las apps, como por ejemplo, cambiar de cuentas.

- Pueden estar en forma permanente en la pantalla o mostrarse/ocultarse con el icono del menú de navegación.



Fuente: ADL.

Navigation Drawer

Principios

- **Identificable:** Su contenido y su forma de listado lo identifica cómo acciones de navegación.
- **Organizada:** Los elementos están ordenados de acuerdo a la importancia, con los destinos frecuentes primero y los destinos parecidos agrupados.
- **Contextual:** Se puede mostrar y ocultar para adecuarse a distintos layouts.

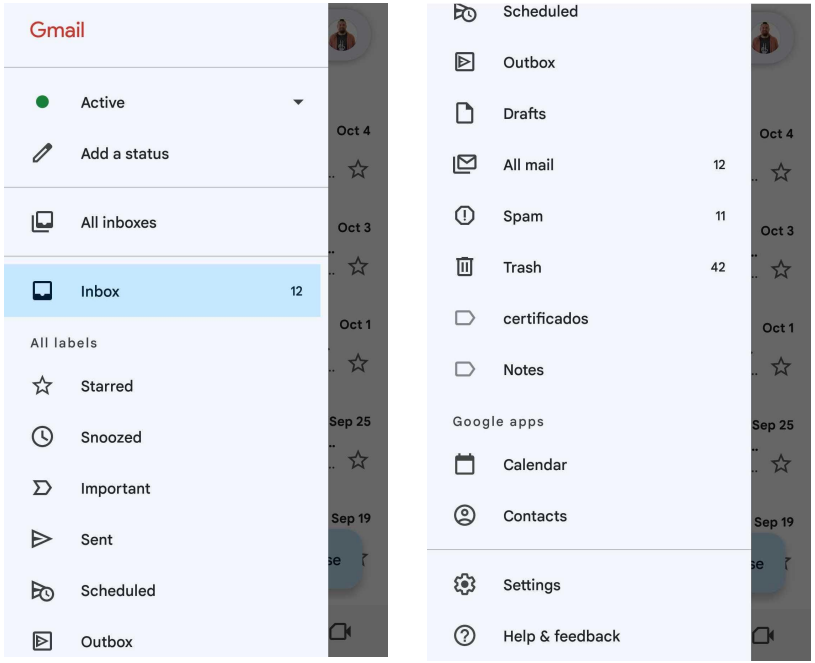
Gmail App

Navigation Drawer

Se recomienda su uso para apps:

- Con más de 5 destinos de alto nivel.
- Con 2 o más niveles de jerarquía de navegación.
- Navegación rápida entre destinos no relacionados.

¿Cuáles de las características mencionadas están presentes en el navigation drawer de la app Gmail?

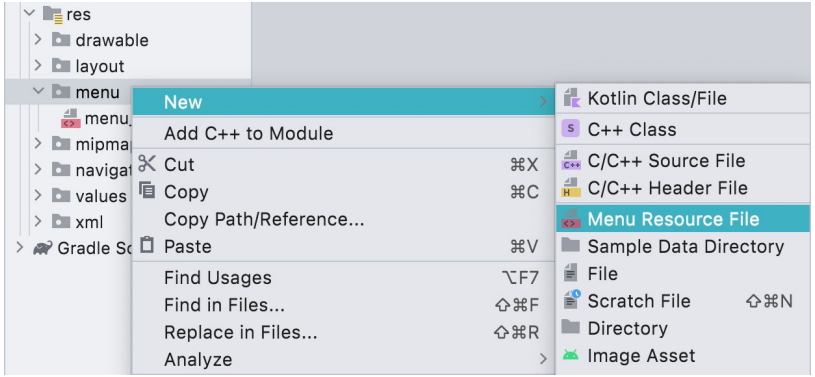


Fuente: ADL.

{desafío} latam_

Navigation Drawer

Menú



Fuente: ADL.

- Agregar un archivo de recursos para menú con las opciones que se van a mostrar.

DrawerLayout & DrawerView

- DrawerLayout actúa como un contenedor de alto nivel para el contenido de la ventana interactiva (drawer) que aparece uno o ambos bordes verticales de la ventana.

- NavigationView representa el menú de navegación estándar de la app. El contenido del menú puede ser llenado usando un archivo de recursos de menú.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
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:id="@+id/drawer layout id"
   android:layout width="match parent"
   android:layout_height="match_parent"
   tools:openDrawer="end">
   <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=".FirstFragment">
...
</androidx.constraintlayout.widget.ConstraintLayout>
   <com.google.android.material.navigation.NavigationView
       android:id="@+id/navigation_view_id"
       android:layout_width="wrap_content"
       android:layout height="match parent"
       android:layout gravity="start"
       tools:menu="@menu/menu_drawer"
       app:menu="@menu/menu_drawer"/>
</androidx.drawerlayout.widget.DrawerLayout>
```

Archivo de recursos de menú

- El tag <menu> es la raíz indicando que existen 2 grupos. Estos grupos son opcionales
- Cada item se identifica con un id que debe ser declarado en un archivo de recursos.
- El icono es recomendable, aunque opcional

Recuerda que los recursos de un ID no hacen referencia a un elemento de recurso real; solo representan una identificación única que puedes adjuntar a otros recursos o usar como un número entero único en tu aplicación.

```xml
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group
        android:id="@+id/group1"
        android:checkableBehavior="single">
        <item
            android:id="@id/profile_nav_item_id"
            android:icon="@drawable/ic_profile"
            android:title="@string/profile" />
        <item
            android:id="@id/item1_nav_item_id"
            android:icon="@drawable/ic_item1"
            android:title="@string/item1"
            android:visible="true" />
    </group>
    <group
        android:id="@+id/group2"
        android:checkableBehavior="single">
        <item
            android:id="@id/settings_nav_item_id"
            android:icon="@drawable/ic_settings"
            android:title="@string/settings" />
        <item
            android:id="@id/logout_nav_item_id"
            android:icon="@drawable/ic_logout"
            android:title="@string/logout" />
    </group>
</menu>
```

Navigation Drawer

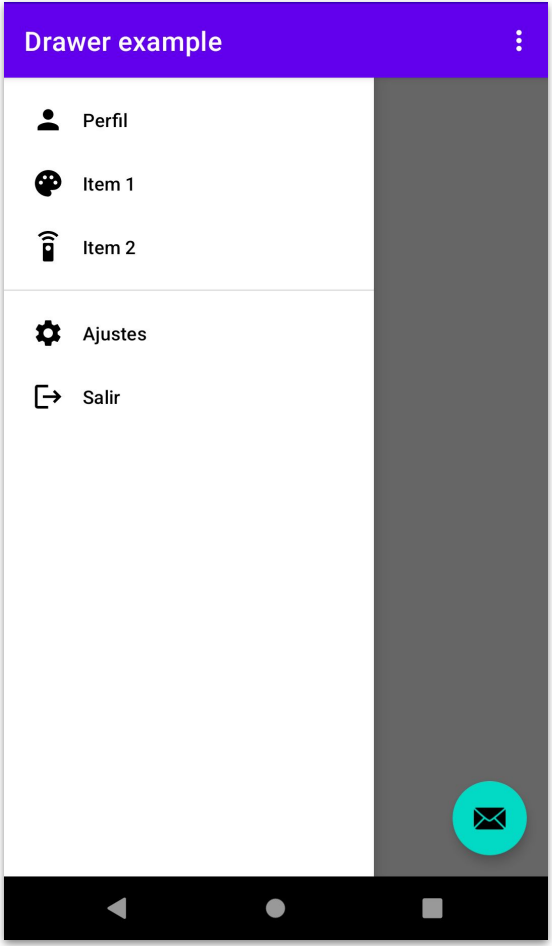
Uniendo todos los componentes

Se necesita:

- El menú con las opciones en un archivo de menú XML.
- ids en un recurso para los ids de los elementos.
- Agregar DrawerLayout y un DrawerView al layout para dar soporte al Navigation Drawer.

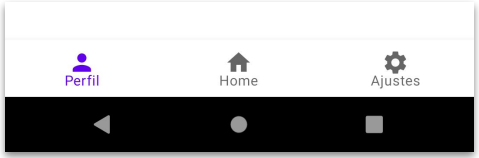
Se puede:

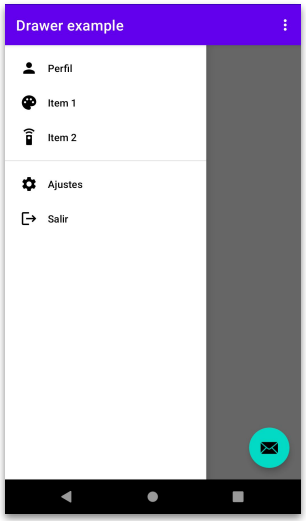
- Agrupar elementos por funcionalidad usando <group>.
- Agregar iconos a cada elemento.
- Modificar estilo y color a los textos de los elementos.



Fuente: ADL.

Tipos de menús





Los distintos menús ofrecen al usuario distintas opciones y pantallas de la app.

Se debe evitar usar Drawer con otro componente de navegación primario, por ejemplo, la Bottom navigation bar.

{desafío} latam_

/* Dialogs */

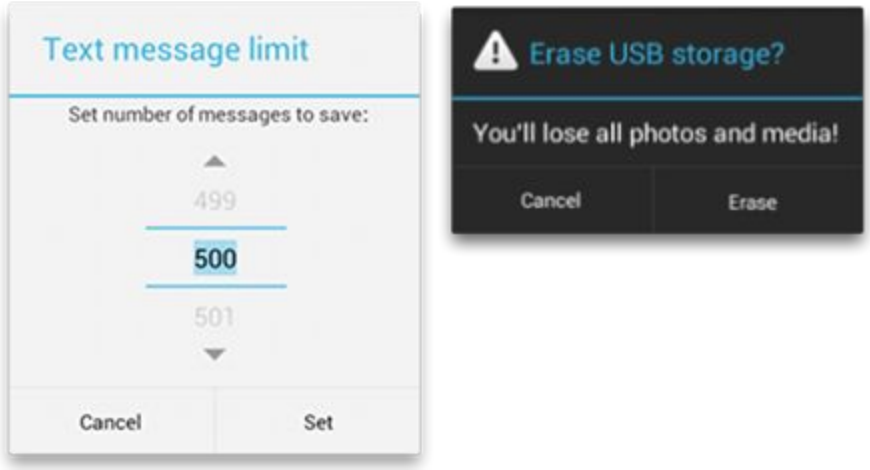
Cuadros de diálogos

Dialogs

- Un diálogo es una ventana pequeña que le indica al usuario que debe tomar una decisión o ingresar información adicional.
- Un diálogo no ocupa toda la pantalla y, generalmente, se usa para eventos modales que requieren que los usuarios realicen alguna acción para poder continuar.

Cuadros de diálogos
Dialogs

- Estos cuadros de diálogo aparecen por sobre la pantalla actual en forma bloqueante, concentrando el foco en una acción y pidiendo al usuario realizar una acción.



Fuente: developer.android.com.

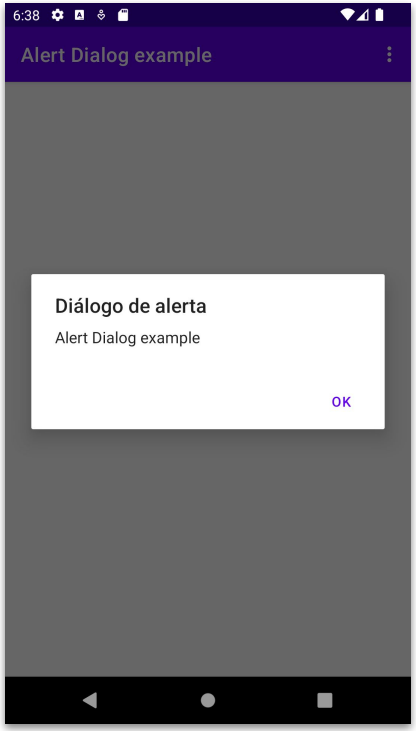
Diálogo de alerta

La clase AlertDialog permite crear una variedad de diseños de diálogo y generalmente es la única clase de diálogo que necesitarás.

1. **Título:** Es opcional y solo se debe usar cuando el área de contenido está ocupada por un mensaje detallado, una lista o un diseño personalizado. Si necesitas indicar un mensaje o una pregunta simple, no necesita un título.
2. **Área de contenido:** Esto puede mostrar un mensaje, una lista u otro diseño personalizado.
3. **Botones de acción:** No debe haber más de tres botones de acción en un diálogo.

Fuente: developer.android.com.

Diálogo de alerta



Fuente: ADL.

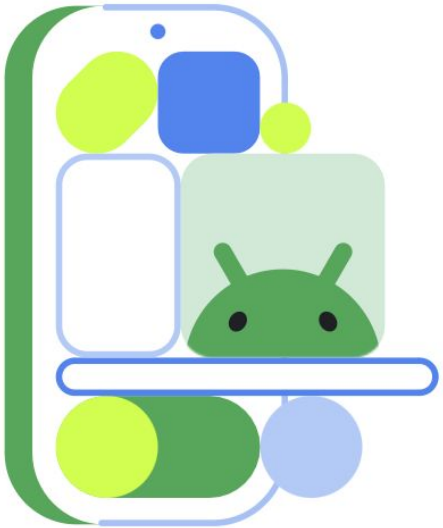
```java
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    showDialog();
}


private void showDialog() {
    // 1. Instanciar un AlertDialog.Builder con su constructor
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

    // 2. Se personalizan las características del dialog llamando a setters sucesivos
    builder.setMessage(R.string.dialog_message)
           .setTitle(R.string.dialog_title);

    // 3. Se agrega las acciones
    builder.setPositiveButton(R.string.ok, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int id) {
         // Acción a ejecutar cuando el usuario presionar el botón
        }
    });

    // 4. Obtener el AlertDialog del create()
    AlertDialog dialog = builder.create();

    // 5. Mostrar el diálogo
    dialog.show();
}
```

{desafío} latam_

Guía de IU - Android

Para conocer más sobre la interfaz gráfica de Android, visita este link.

En este link encontrarás información detallada y recursos útiles para comprender y trabajar con la interfaz de usuario en Android.



Fuente: developer.android

¿Por qué al usar correctamente los componentes de Android, nos aseguramos que nuestra app tenga un comportamiento predecible para el usuario?

Próxima sesión...

- *Guía de ejercicios.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales