Elementos de la interfaz, navegación e interacción

Elementos de navegación

{desafío} latam_

Utilizar elementos de navegación e interacción de usuario disponibles en el entorno Android Studio para dar solución a un requerimiento.

- 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.

Te encuentras aquí

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Reconocer los elementos de navegación y de interacción distinguiendo su uso en un aplicativo Android nativo.*

{desafío} latam_

Al presionar el botón “compartir” de cualquier app, ¿cómo sabe Android cuales son mis app disponibles para la acción?

/* Intents */

Intents

¿Qué son?

- Un Intent es un objeto de mensajería que puedes usar para solicitar una acción de otro componente de una app.

Usos principales de intents

- Existen tres casos de uso principales:

Iniciar una actividad

Una **Activity** representa una única pantalla en una aplicación. Puedes iniciar una nueva instancia de una Activity pasando un Intent a **startActivity()**. El Intent describe la actividad que se debe iniciar y contiene los datos necesarios para ello.

Iniciar un servicio

Un **Service** es un componente que realiza operaciones en segundo plano sin una interfaz de usuario. Con Android 5.0 (nivel de API 21) y versiones posteriores, puedes iniciar un servicio con **JobScheduler**.

Transmitir una emisión

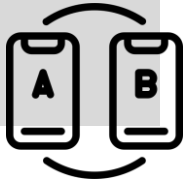
Una emisión es un aviso que cualquier aplicación puede recibir. El sistema transmite varias emisiones de eventos, por ejemplo cuando comienza a cargarse el dispositivo o cuando se inicia el sistema. Puedes transmitir una emisión a otras apps pasando un **Intent** a **sendBroadcast()** o **sendOrderedBroadcast()**.

Fuente: https://developer.android.com/guide/components/intents-filters

Tipos de intents

Intents explícitos

Especifican qué aplicación administrará el Intent, ya sea incluyendo el nombre del paquete de la app de destino o el nombre de clase del componente. Normalmente el usuario usa un Intent explícito para iniciar un componente en su propia aplicación porque conoce el nombre de clase de la actividad o el servicio que desea iniciar. Por ejemplo, puedes utilizarla para iniciar una actividad nueva en respuesta a una acción del usuario o iniciar un servicio para descargar un archivo en segundo plano.



Intents implícitos

No nombran el componente específico. En cambio declaran una acción general para realizar, lo cual permite que un componente de otra aplicación la maneje.

Por ejemplo, si deseas mostrar al usuario una ubicación en un mapa, puedes usar un Intent implícito para solicitar que otra aplicación compatible muestre una ubicación específica en un mapa (Google maps, Waze)

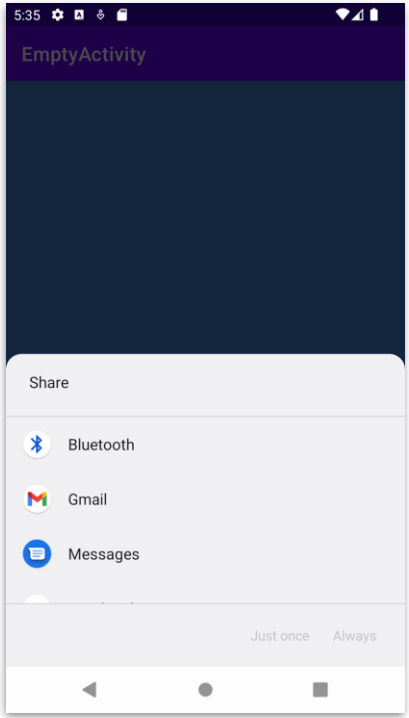
{desafío} latam_

Fuente: Developer Android

Intents implícitos

- Un ejemplo de Intents implícitos es la acción “Compartir” donde se muestra un listado de aplicaciones aptas para compartir la información.

```
public class MainActivity extends AppCompatActivity {

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

    private void shareIntent() {
        Intent sendIntent = new Intent();
        sendIntent.setAction(Intent.ACTION_SEND);
        sendIntent.putExtra(Intent.EXTRA_TEXT,
                "Mira mi nueva app:
https://play.google.com/store/apps/details?id=" +
BuildConfig.APPLICATION_ID);
        sendIntent.setType("text/plain");
        startActivity(sendIntent);
    }
}
```

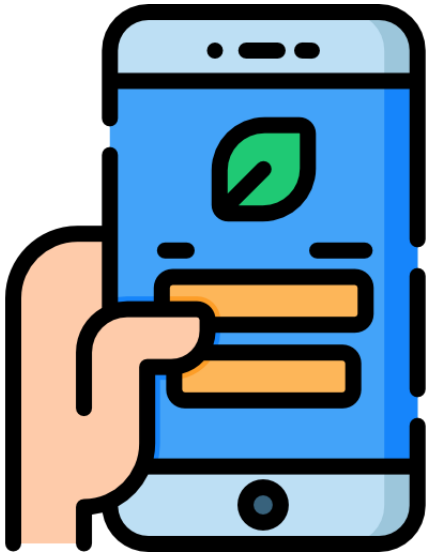


Fuente: ADL.

/* Fragmentos */

Fragments

- **Un Fragment representa una parte de la interfaz de usuario**.
- Puedes pensar en un fragmento como una sección modular de una actividad que tiene un ciclo de vida propio, que recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando
- **Un fragmento siempre debe estar alojado en una actividad** y el ciclo de vida del fragmento se ve afectado directamente por el ciclo de vida de la actividad anfitriona.

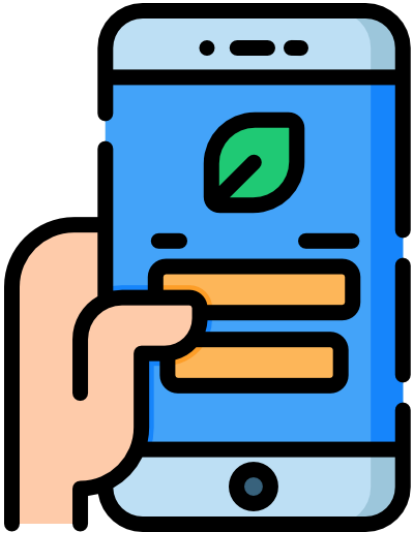


{desafío} latam_

Fuente: Developer Android

Fragments

- Cuando realizas una transacción de fragmentos, también puedes agregarlos a una **pila de actividades** administrada por la actividad.
- La pila de actividades le permite al usuario invertir una transacción de fragmentos (**navegar hacia atrás**) al presionar el botón *Atrás*.
- Las actividades se organizan en una pila (pila de actividades) cada actividad y fragmentos en el orden en que se abre/cierra cada uno.



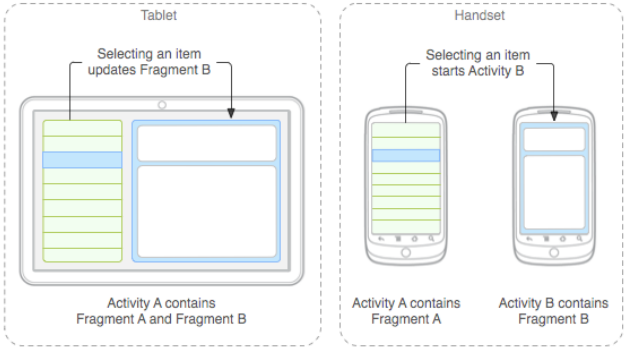
{desafío} latam_

Fuente: Developer Android

Filosofía de diseño

Los fragmentos se introducen en Android 3.0 principalmente para admitir diseños de IU más dinámicos y flexibles para distintos tamaños de pantallas. Por ejemplo, la pantalla de un teléfono es más pequeña que la de una tablet y el tener una pantalla más grande permite combinar e intercambiar componentes de la interfaz.

Cada fragmento debe ser diseñado como un componente modular y reutilizable, definiendo su propio diseño y comportamiento sin que dependa de la actividad que lo aloja para poder ser reutilizado.

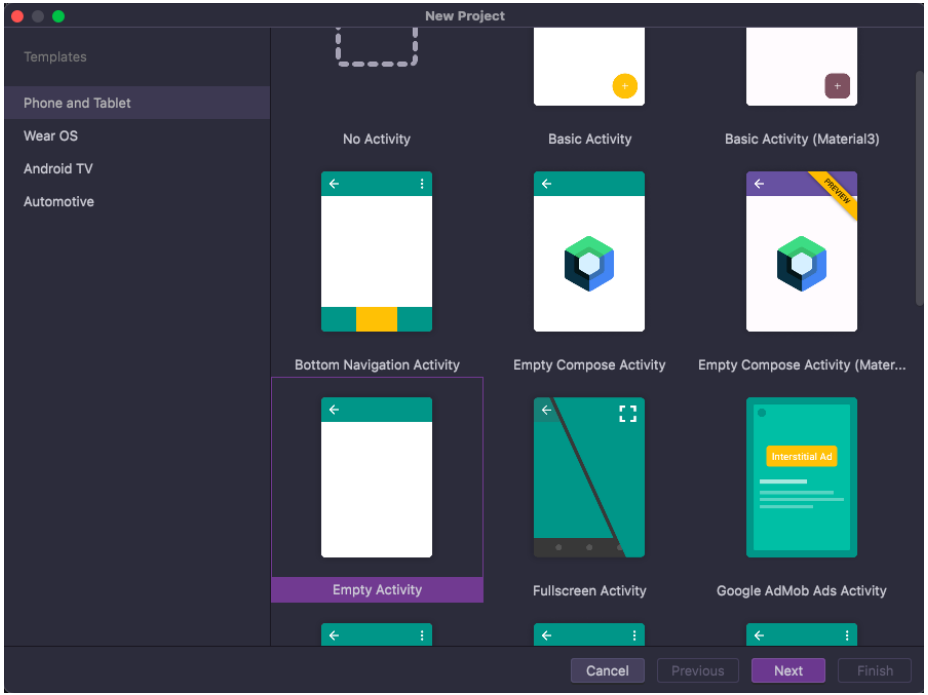


Fuente: developer.android.com.

Ejercicio práctico

Crear proyecto

Usar el template de Empty Activity para crear el proyecto que usaremos para agregar fragmentos y conocer formas de cómo interactuar con ellos.

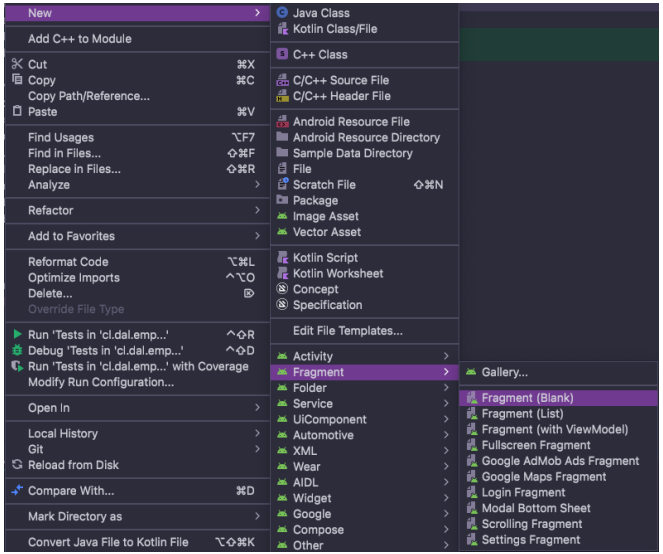


Fuente: ADL.

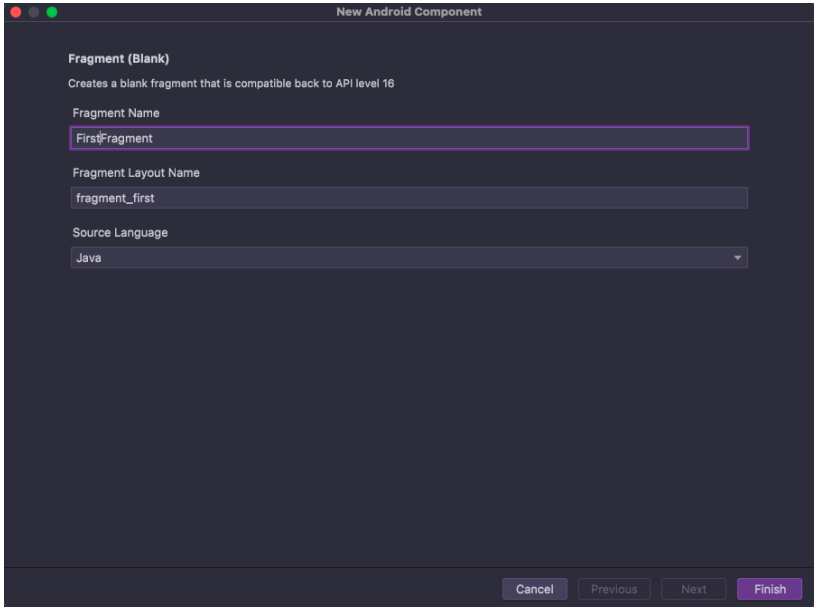
Agregar fragmento

Primer fragmento

Usando el botón derecho podemos agregar un nuevo fragmento. Vamos a agregar uno vacío para nuestra app



Fuente: ADL.



Fuente: ADL.

Nombramos al fragmento como FirstFragment. En forma conjunta, se crea el layout correspondiente al fragmento.

{desafío} latam_

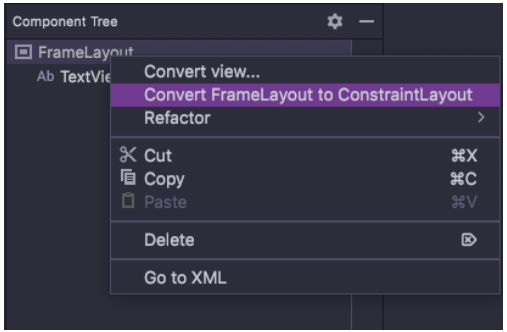
Modificar diseño del fragmento

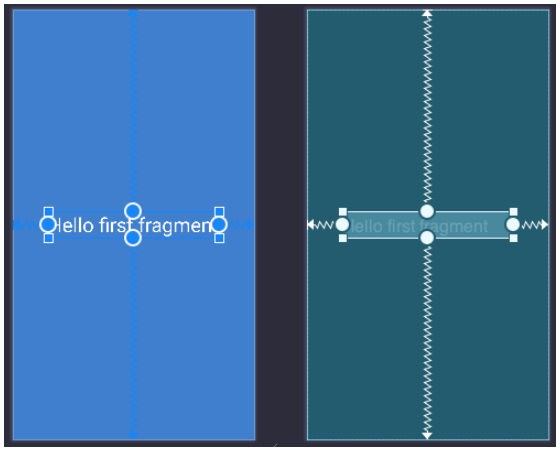
Layout del primer fragmento

1. Convertir FrameLayout a Constraint layout.
2. Ubicar el TextView al centro de la pantalla en forma relativa.
3. Crear recurso de texto en strings.xml

```
<string name="hello_first_fragment">Hello first fragment</string>
```

1. Cambiar el color de fondo modificando fragment_first.xml asignando `android:background="#3F7FCD".`
2. Cambiar la apariencia del TextView:
    i. android:layout_width=”wrap_content”
    ii. android:layout_height=”wrap_content”
    iii. android:textColor=”@color/white”
    iv. android:textSize=”34sp”
    v. android:text=”@string/hello_first_fragment”



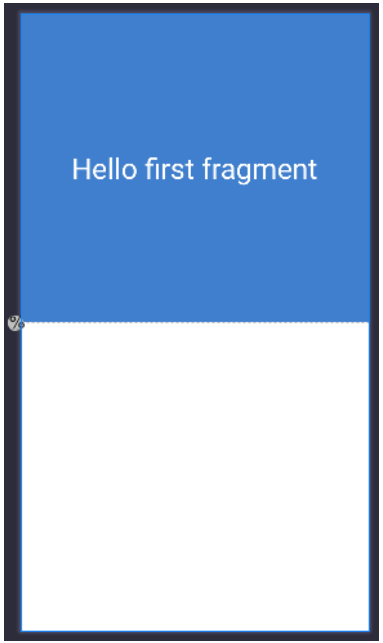


Fuente: ADL.

{desafío} latam_

Mostrar el fragmento en la pantalla

Modificar activity_main.xml



Fuente: ADL.

```xml
<?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">

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guidelineHorizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <fragment
        android:id="@+id/firstFragment"
        class="cl.dal.emptyactivity.FirstFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/guidelineHorizontal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout="@layout/fragment_first" />

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

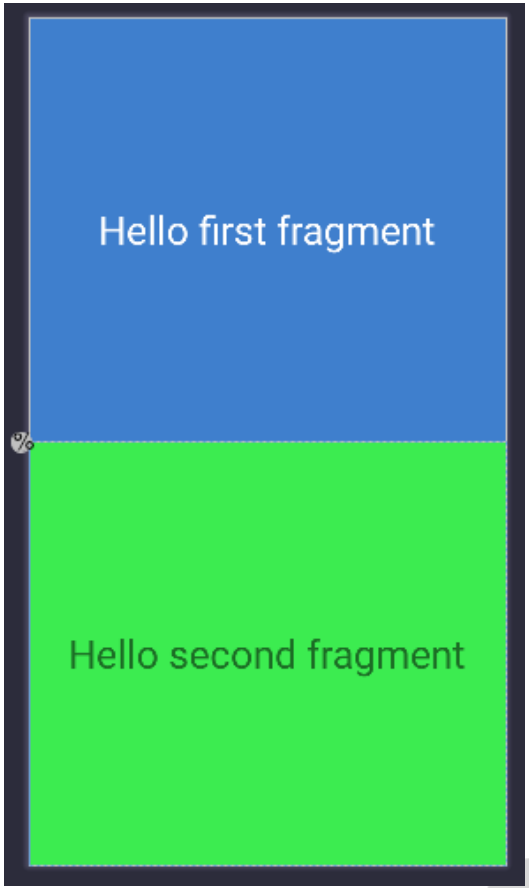
Ejercicio - Agregar un segundo fragmento

Ejercicio

Agregar un segundo fragmento

Crea y agregar un segundo fragmento a la parte inferior de activity_main.xml

- SecondFragment
- Background color: `#3CEC50`
- Text size: 34sp
- Text: Hello second fragment



Fuente: ADL.

Fragmentos

Fragmentos estáticos

Estos fragmentos son estáticos y no se puede modificar su ubicación durante la ejecución, perdiendo flexibilidad y adaptabilidad para los distintos tipos de pantallas.

- ¿Qué pasa si queremos que el fragmento 1 ocupe toda la pantalla, y que con un botón se pueda mover al fragmento 2?
- ¿Cómo se muestran / ocultan fragmentos?

A diferencia de las Activity, los fragmentos no utilizan Intent.

/* Fragment Manager */

Administrador Fragments

- FragmentManager es la clase responsable de realizar acciones en los fragmentos.
- El FragmentManager administra la pila de actividades del fragmento.
- Cada conjunto de cambios se confirma como una sola unidad llamada FragmentTransaction.
- Cuando el usuario presiona el botón *Atrás* en el dispositivo, se quita la transacción que se encuentra en la parte superior de la pila (LIFO). Si no hay más transacciones en la pila, el evento Atrás es entregado a la actividad.
- Se puede acceder al Fragment Manager a través del método getSupportFragmentManager()

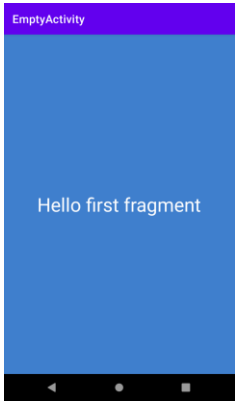
Fuente: Developer Android

FragmentContainerView

main_activity.xml

FragmentContainerView es un layout especial para fragmentos. Puede manejar transacciones de fragmentos de manera confiable.

Es el contenedor que permite manipular fragmentos, tanto para agregar y quitar, como para reemplazarlos.



Fuente: ADL.

```xml
<?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">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/main_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="cl.dal.emptyactivity.FirstFragment"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

    </androidx.fragment.app.FragmentContainerView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

{desafío} latam_

Hacer una transacción

Para mostrar un fragmento en la app usamos FragmentManager para crear una FragmentTransaction.

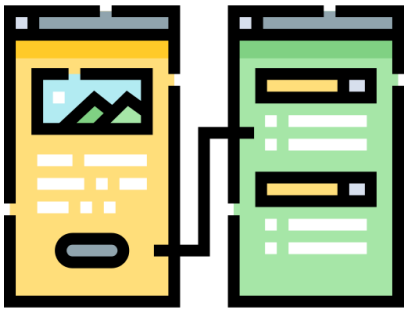
La transacción puede ser de tipo:

- add()
- replace()
- remove()


```
public class MainActivity extends AppCompatActivity {

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

        FragmentManager fragmentManager = getSupportFragmentManager();
        fragmentManager.beginTransaction()
                .replace(R.id.main_container, SecondFragment.class, null)
                .setReorderingAllowed(true)
                .addToBackStack("Second fragment")
                // La transacción no se ejecuta sin commit
                .commit();
    }
}
```

{desafío} latam_

Jetpack Navigation



“Es posible que nunca interactúes con FragmentManager directamente si usas la biblioteca de Jetpack Navigation, ya que funciona como una capa por sobre FragmentManager. Sin embargo, cualquier app que utilice fragmentos usa FragmentManager en algún punto, por lo que es importante comprender qué es y cómo funciona.

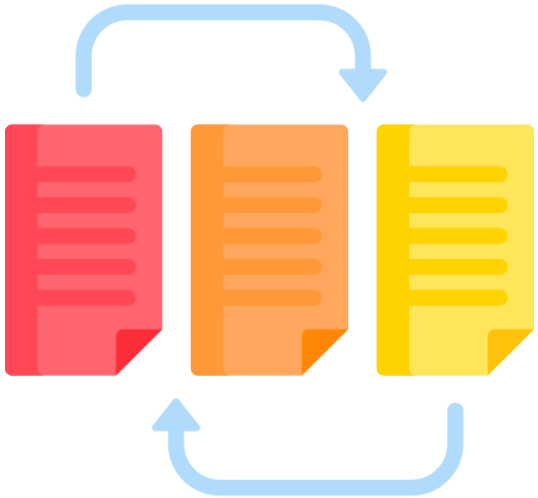
Jetpack Navigation es el componente que permite diseñar la navegación entre distintos fragmentos de forma visual”

Fuente: developer.android.com

/* Biblioteca de navegación de Jetpack (Navigation component) */

Componente de Navigation

- La navegación se refiere a las interacciones que permiten a los usuarios navegar a través, entrar y salir de diferentes contenidos de la app.
- El componente Navigation de Android Jetpack permite implementar la navegación desde simples clics de botones hasta patrones más complejos, como las barras de apps (App or top bar, Bottom bar) y los paneles laterales de navegación (Navigation Drawer).



Principios de la navegación

1. **El destino de inicio fijo.**

Es la primera pantalla que ve el usuario y es la última pantalla que ve el usuario antes de salir de la app.

2. **El estado de navegación se representa como una pila de destinos.**

La primera pantalla se convierte en la base de la pila de actividades, y los cambios en la pila ocurren en la parte superior.

3. **Los botones Atrás de la barra y Atrás del teléfono son idénticos.**

Cuando se presiona el botón Atrás, el destino actual se saca de la pila de actividades y se navega al destino previo. Esto permite navegar en orden cronológico inverso por el historial de pantallas. El botón Atrás de la barra no se debe mostrar si es la primera pantalla.

{desafío} latam_

Fuente: Android developer

Componentes claves

Navigation graph	NavHost	NavController
Archivo de recursos (XML) con toda la información relacionada con la navegación, incluyendo destinos y todos los posibles caminos para llegar a ellos.	Contenedor vacío que muestra los destinos del gráfico de navegación. El componente Navigation contiene un NavHost implementado por defecto llamado NavHostFragment. setPrimaryNavigationFragment(finalHost) es equivalente a app:defaultNavHost="true": permiten que el NavHost intercepte las activaciones del botón Atrás del sistema.	Un objeto que maneja la navegación del NavHost. Orquesta los cambios de destino dentro del NavHost mostrando las distintas pantallas de la app.

Al navegar por la app, se le dice al **NavController** que se quiere navegar siguiendo un camino del gráfico de navegación o directamente hasta una destino específico que fueron definidos en el **gráfico de navegación**. El NavController muestra el contenido del destino elegido en el **NavHost**.

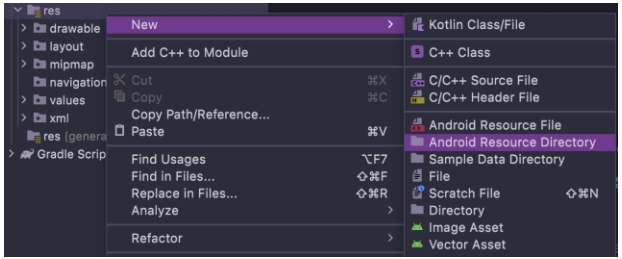
Navigation
Dependencias

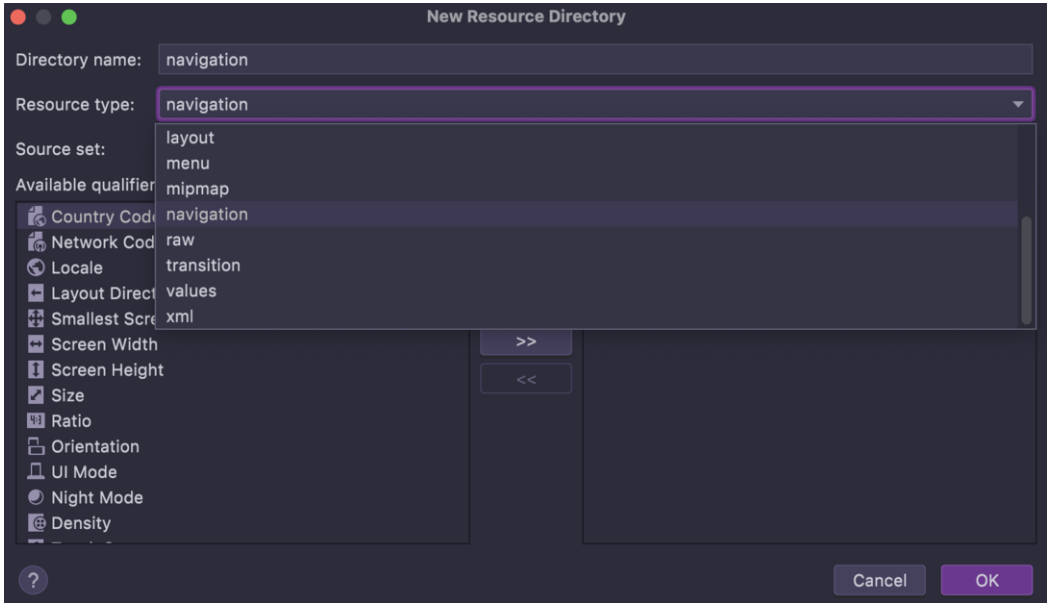
- ¿En cuál archivo se deben agregar las dependencias de *navigation*?
- ¿Qué significa que se marquen como *implementation*?
- ¿En qué lenguaje están escritas las dependencias de *navigation*?

```
dependencies {
    . . .

    def navigation_version = "2.5.2"
    implementation "androidx.navigation:navigation-fragment:$navigation_version"
    implementation "androidx.navigation:navigation-ui:$navigation_version"
}
```

Navigation

Crear directorio de recurso de navegación

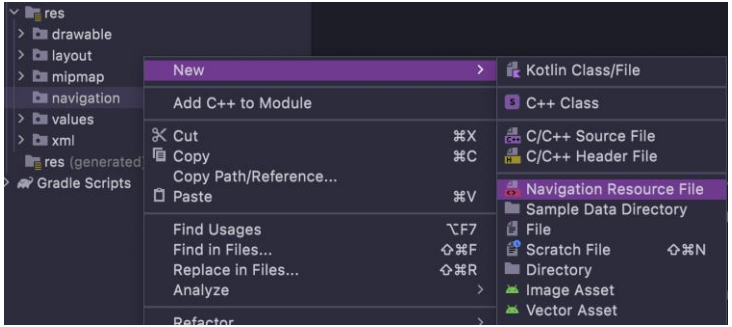


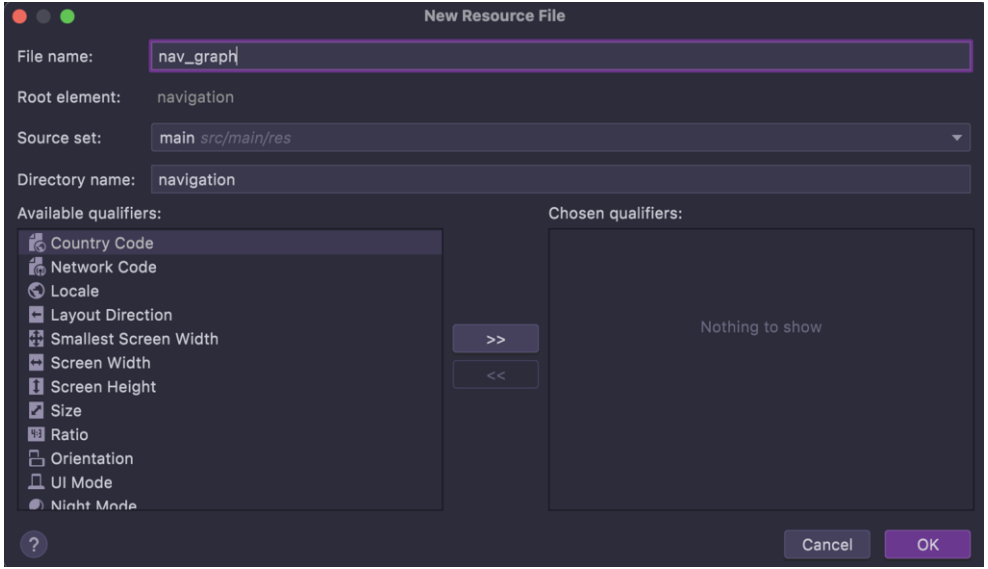


{desafío} latam_

Navigation

Crear archivo de recurso de navegación





Por defecto, se denomina **nav_graph** al gráfico de navegación que usará la app

{desafío} latam_

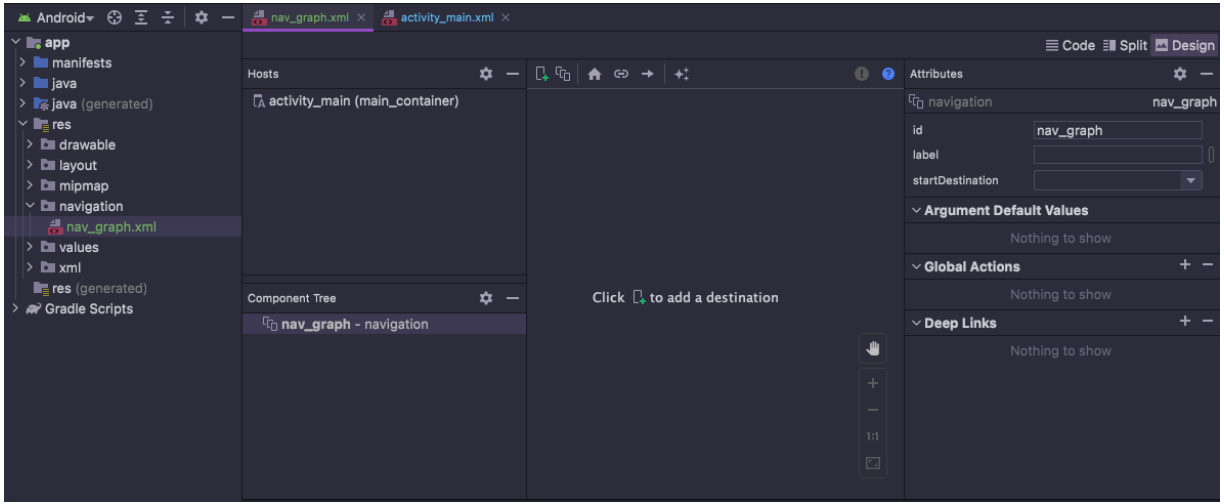
NavHostFragment

- Se le indica al FragmentContainerView que va a ser del tipo NavHostFragment.
- Se enlaza al gráfico de navegación nav_graph.
- Se indica que va a ser la pantalla de navegación por defecto.

```
<?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">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/main_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"

        android:name="androidx.navigation.fragment.NavHostFragment"
        app:navGraph="@navigation/nav_graph"
        app:defaultNavHost="true"

        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </androidx.fragment.app.FragmentContainerView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Editor de navegación

El editor de navegación reconoce que el FragmentContainerView está configurado como NavHostFragment y lo muestra en la lista de Hosts.

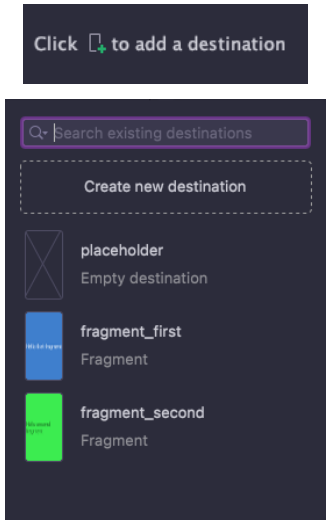


{desafío} latam_

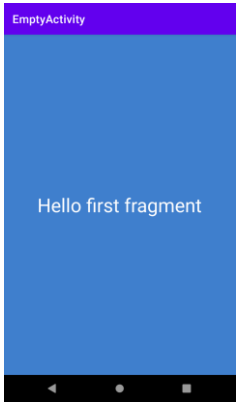
Editor de navegación

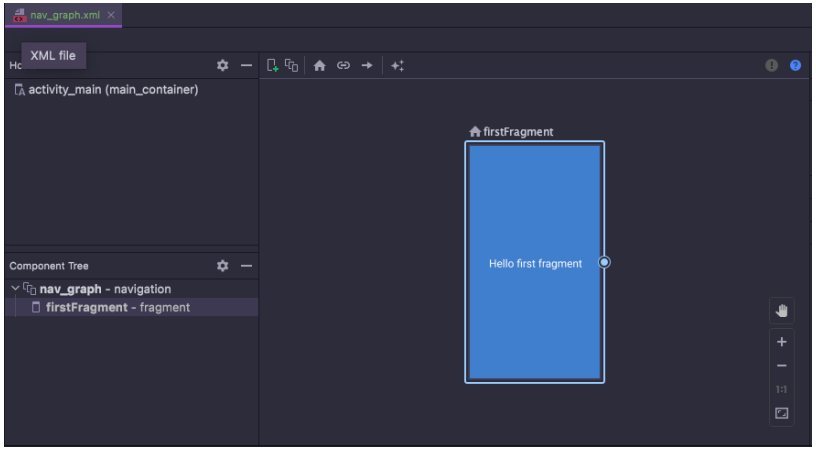
Agregar primer fragmento

- Se agrega el primer fragmento



- Se define como el Home. Es el primer fragmento que se muestra





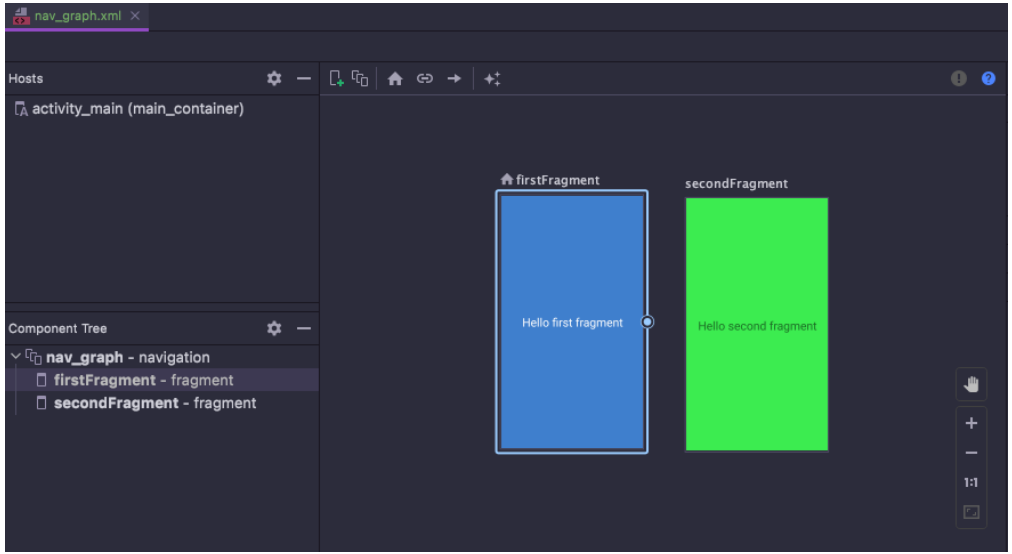
Fuente: ADL.

{desafío} latam_

Editor de navegación

Agregar segundo fragmento

- Repetimos el mismo proceso para agregar el SecondFragment.
- El primer fragment sigue siendo el Home, donde se inicia la app.
- En este momento, ambos fragmentos no tienen relación entre sí.



Fuente: ADL.

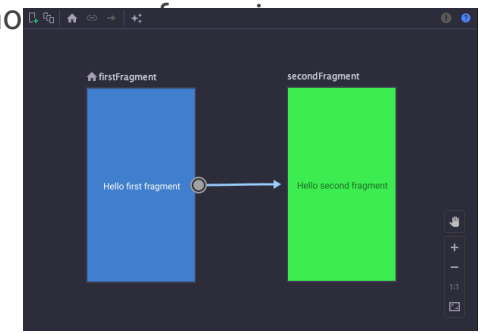
{desafío} latam_

Editor de navegación

Acción de navegación del primer a segundo fragmento

- Arrastrando el círculo en los bordes de cada fragmento se puede crear un link hacia otro fragmento.

- Se crea un <action> asociado a firstFragment que tiene como destino secondFragment con un ID único que usaremo



Fuente: ADL.

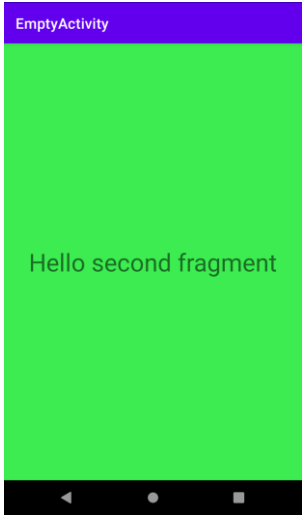
```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/firstFragment">
    <fragment
        android:id="@+id/firstFragment"
        android:name="cl.dal.emptyactivity.FirstFragment"
        android:label="fragment_first"
        tools:layout="@layout/fragment_first" >
        <action
            android:id="@+id/action_firstFragment_to_secondFragment"
            app:destination="@id/secondFragment" />
    </fragment>
    <fragment
        android:id="@+id/secondFragment"
        android:name="cl.dal.emptyactivity.SecondFragment"
        android:label="fragment_second"
        tools:layout="@layout/fragment_second" />
</navigation>
```

{desafío} latam_

Editor de navegación

Acción de navegación del primer a segundo fragmento

```
public class MainActivity extends AppCompatActivity {

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

  private void navigateToSecondFragment() {
     NavHostFragment navHostFragment =
           (NavHostFragment)getSupportFragmentManager().findFragmentById(R.id.main_container);
     NavController navController = navHostFragment.getNavController();
     navController.navigate(R.id.action_firstFragment_to_secondFragment);
  }
}
```



{desafío} latam_

Recordemos

Para finalizar recordemos los siguientes conceptos:

- **Navigation graph** o Gráfico de navegación es un archivo de recursos (XML) con toda la información relacionada con la navegación.
- El **NavController** es un objeto que maneja la navegación del NavHost.
- El **NavHost** es un contenedor vacío que muestra los destinos del gráfico de navegación.

¿Cuáles son las ventajas de utilizar Fragmentos?

Próxima sesión...

- *Guía de ejercicios.*

{desafío}
latam_

Academia de
talentos digitales