Tarea 6: Haz que tu aplicación sea interactiva

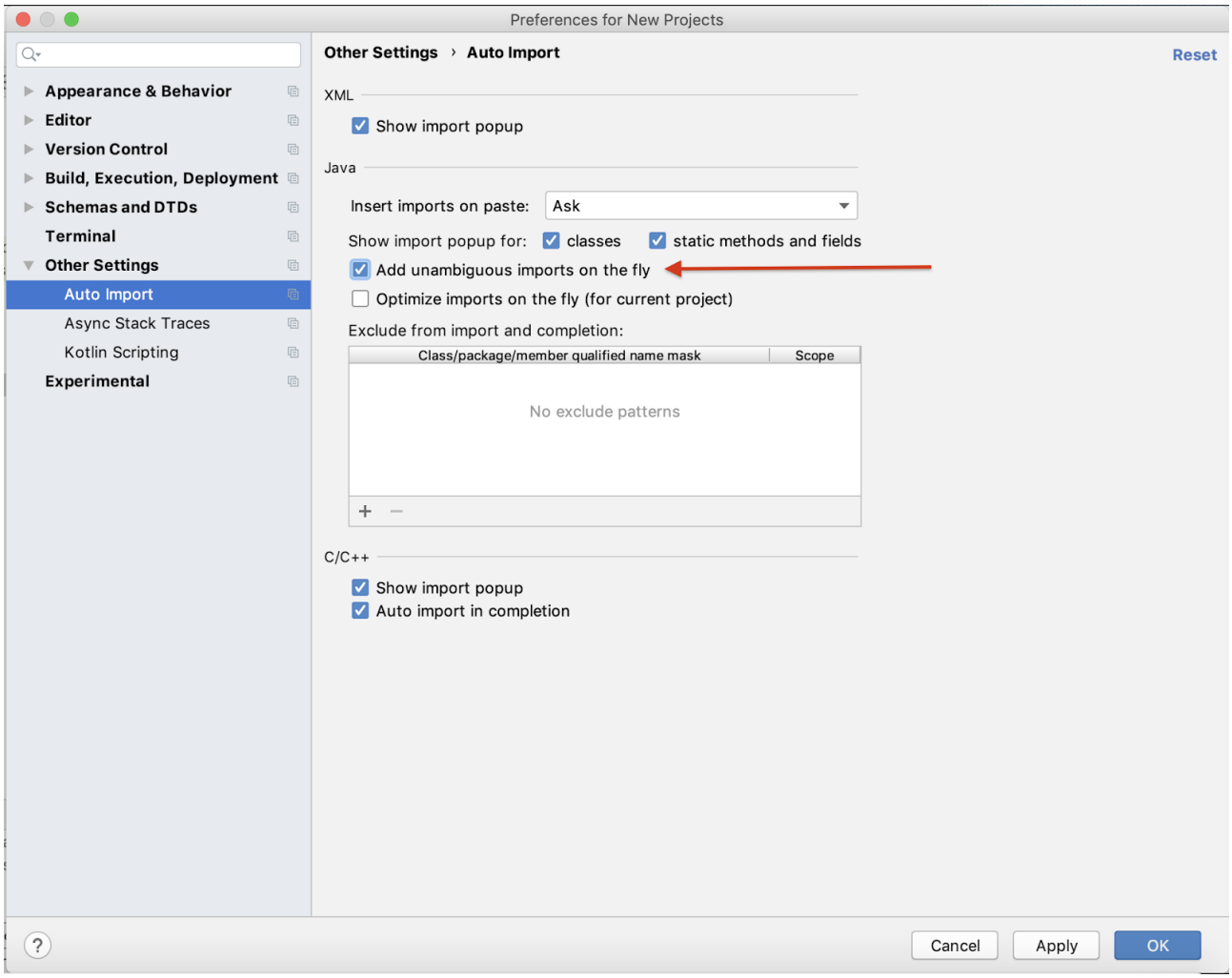
Ha agregado botones a la pantalla principal de su aplicación, pero actualmente los botones no hacen nada. En esta tarea, harás que tus botones respondan cuando el usuario los presione.

- Primero, hará que el **botón Toast** muestre un **mensaje emergente** llamado **toast**.
  A continuación, hará que el botón **Contar** actualice el número que se muestra en **TextView**.

Paso 1: habilite las importaciones automáticas

Para facilitarle la vida, puede habilitar las importaciones automáticas para que Android Studio importe automáticamente cualquier clase que necesite el código Java.

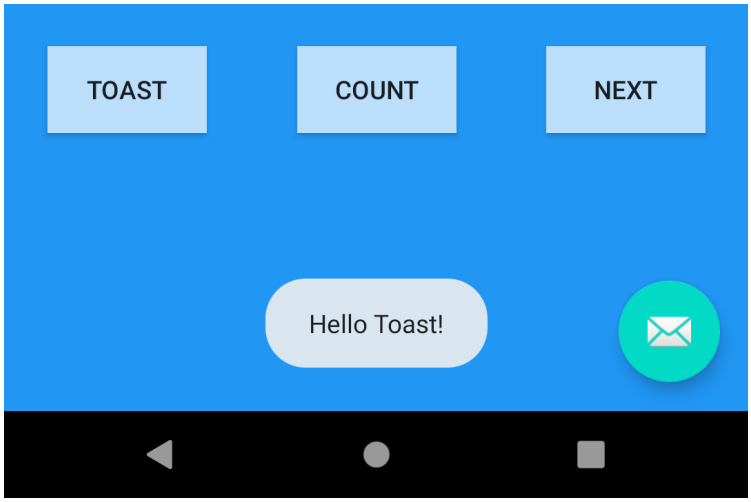
1. En Android Studio, abra el editor de configuraciones yendo a **Archivo > Otras configuraciones > Preferencias para nuevos proyectos.**
2. Seleccione **Importaciones automáticas**. En la sección **Java**, asegúrese de que esté marcada la opción **Agregar importaciones sin ambigüedades sobre la marcha**.
3. Cierre el editor de configuraciones presionando OK.



Paso 2: mostrar un toast

En este paso, adjuntará un método Java al **botón Toast** para mostrar un toast cuando el usuario presione el botón.

Un toast es un mensaje breve que aparece brevemente en la parte inferior de la pantalla.



1. Abra **FirstFragment.java** (**app** > **java** > **com.example.android.myfirstapp** > **FirstFragment**). Esta clase tiene solo dos métodos, **onCreateView()** y **onViewCreated()**. Estos métodos se ejecutan cuando se inicia el fragmento.

Como se mencionó anteriormente, la identificación de una vista lo ayuda a identificar esa vista claramente de otras vistas.
Usando el método **findViewByID()**, su código puede encontrar el botón aleatorio usando su **id**, **R.id.random_button**.

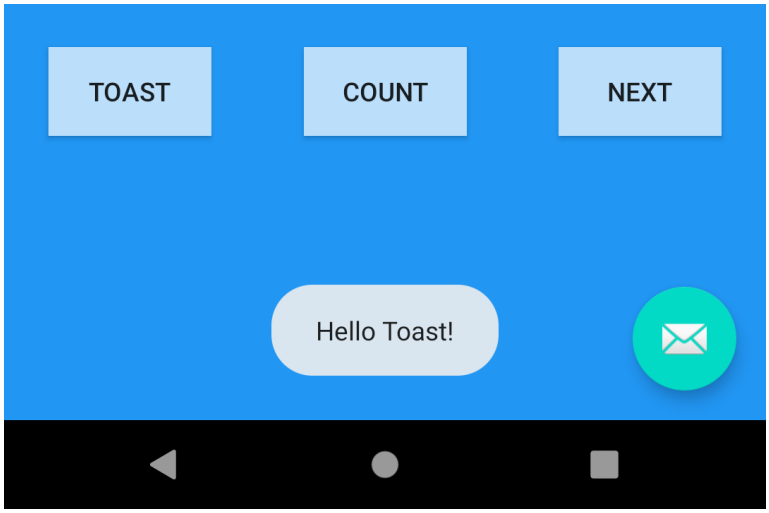
2. Eche un vistazo a **onViewCreated()**. Configura un **Listener (detector de clicks)** para el botón **Ramdom**, que se creó originalmente como el **botón Siguiente. (NEXT)**

```
view.findViewById(R.id.random_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        NavHostFragment.findNavController(FirstFragment.this)
                .navigate(R.id.action_FirstFragment_to_SecondFragment);
    }
});
```

Esto es lo que hace este código:
Use el método **findViewById()** con la identificación de la vista deseada como argumento, luego configure un detector de clics **(Listener**) en esa vista.
En el cuerpo del Listener, use una acción, que en este caso es para **navegar a otro fragmento**, y navegue hasta allí.
3. Justo debajo de ese Listener, agregue código para configurar un Listener para el botón de **Toast**, que crea y muestra un **toast**. Aquí está el código:

```
view.findViewById(R.id.toast_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Toast myToast = Toast.makeText(getActivity(), "Hello toast!",
Toast.LENGTH_SHORT);
        myToast.show();
    }
});
```

4. Ejecute la aplicación y presione el botón Toast. Ves el mensaje toast en la parte inferior de la pantalla?



4. Si lo desea, extraiga la cadena del mensaje en un recurso como lo hizo con las etiquetas de los botones.
Ha aprendido que para hacer que una vista sea interactiva, necesita configurar un Listener (detector de clics) para la vista **que dice qué hacer** cuando se hace clic en la vista (**botón**).
El Listener puede:

- Implementar una pequeña cantidad de código directamente.
- Llamar a un método que defina el comportamiento de clic deseado en la actividad.

Paso 3: haga que el botón Contar actualice el número en la pantalla
El método que muestra el brindis es muy sencillo; no interactúa con ninguna otra vista en el diseño. En el siguiente paso, agrega comportamiento a su diseño para buscar y actualizar otras vistas.
Actualice el botón Contar para que cuando se presione, el número en la pantalla aumente en 1.

1.En el archivo **fragment_first.xml** note el **id** for the **TextView**:

```
<TextView
    android:id="@+id/textview_first"
```

2. En **FirstFragment.java**, agregue un detector de clics (**Listener**) para **count_button** debajo de los otros detectores de clics en on**ViewCreated()**. Debido a que tiene un poco más de trabajo por hacer, haga que llame a un nuevo método, **countMe().**

```
view.findViewById(R.id.count_button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        countMe(view);
    }
});
```

3. En la clase **FirstFragment**, agregue el método **countMe()** que toma un solo argumento **View**. Este método se invocará cuando se haga clic en el **botón Contar** y se llame al Listener (detector de clics).

```
private void countMe(View view) {
}
```

4. Obtenga el valor de **showCountTextView**. Lo definirá en el siguiente paso.

```
    ...
     // Get the value of the text view
     String countString = showCountTextView.getText().toString();
```

5. Convierta el valor en un número e increméntelo.

```
    ...
     // Convert value to a number and increment it
     Integer count = Integer.parseInt(countString);
     count++;
```

6. Muestre el nuevo valor en **TextView** configurando mediante programación la propiedad de texto de TextView.

```
    ...
    // Display the new value in the text view.
    showCountTextView.setText(count.toString());
```

Aquí está todo el método:

```
private void countMe(View view) {
    // Get the value of the text view
    String countString = showCountTextView.getText().toString();
    // Convert value to a number and increment it
    Integer count = Integer.parseInt(countString);
    count++;
    // Display the new value in the text view.
    showCountTextView.setText(count.toString());
}
```

Paso 4: Guarde en caché el TextView para uso repetido

Puede llamar a **findViewById()** en **countMe()** para encontrar **showCountTextView**. Sin embargo, se llama a **countMe()** cada vez que se hace clic en el botón, y **findViewById()** es un método relativamente lento para llamar.
Por lo tanto, es mejor encontrar la vista una vez y almacenarla en caché.

1. En la clase **FirstFragment** antes de cualquier método, agregue una variable miembro para **showCountTextView** de tipo **TextView.**

```
TextView showCountTextView;
```

2. En **onCreateView()**, llamará a **findViewById()** para obtener el **TextView** que muestra el conteo. El método **findViewById()** debe llamarse en una vista donde debe comenzar la búsqueda de la ID solicitada, así que asigne la vista de diseño que se devuelve actualmente a una nueva variable, **fragmentFirstLayout**, en su lugar.

```
// Inflate the layout for this fragment
View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container, false);
```

3. Llame a f**indViewById()** en **fragmentFirstLayout** y especifique el **id** de la vista que desea buscar, **textview_first**. Guarde en caché ese valor en **showCountTextView**.

```
    ...
     // Get the count text view
     showCountTextView = fragmentFirstLayout.findViewById(R.id.textview_first);
```

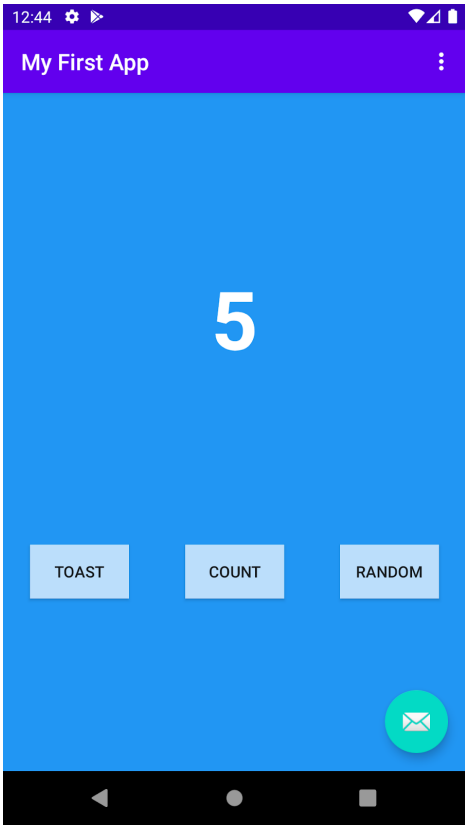
4. Devuelve **fragmentFirstLayout** desde **onCreateView()**.

```
return fragmentFirstLayout;
```

Aquí está el método completo y la declaración de showCountTextView:

```
TextView showCountTextView;
@Override
public View onCreateView(
        LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState
) {
    // Inflate the layout for this fragment
    View fragmentFirstLayout = inflater.inflate(R.layout.fragment_first, container, false);
    // Get the count text view
    showCountTextView = fragmentFirstLayout.findViewById(R.id.textview_first);
    return fragmentFirstLayout;
}
```

5. Ejecute su aplicación. Presione el botón Contar y observe cómo se actualiza el conteo.



Tarea 7: Implementar el segundo fragmento

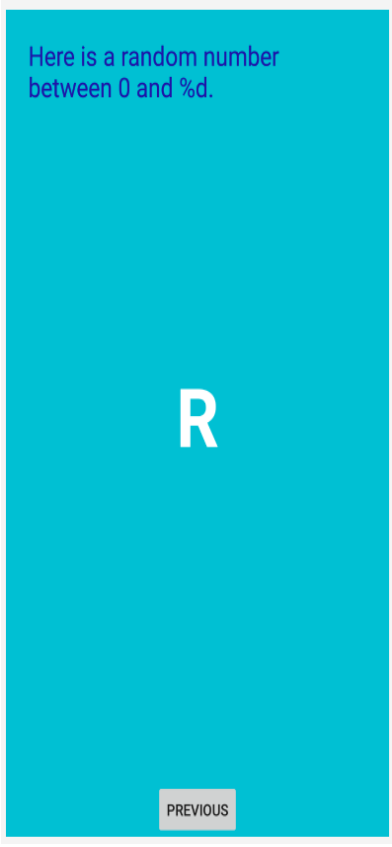
Hasta ahora, se ha centrado en la primera pantalla de su aplicación.
A continuación, actualizará el botón **Random** para mostrar un número aleatorio entre 0 y el recuento actual en una segunda pantalla.

Actualizar el diseño para el segundo fragmento.

La pantalla del nuevo fragmento mostrará un título de encabezado y el número aleatorio.

Así es como se verá la pantalla en la vista de diseño:

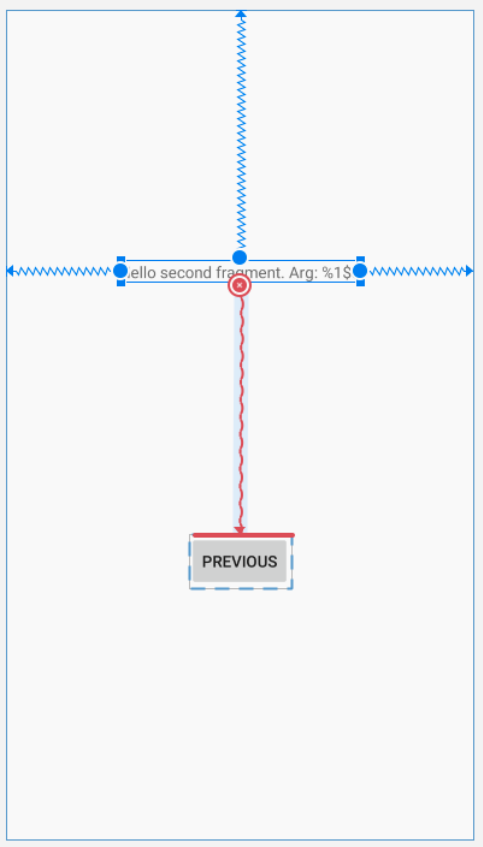
- El %d indica que parte de la cadena se reemplazará con un número.
- La R es solo un marcador de posición.



Paso 1: agregue un TextView para el número aleatorio

1. Abra f**ragmento_segundo.xml** (**aplicación > res > diseño > fragment_second.xml**) y cambie a Vista de diseño si es necesario.

Observe que tiene un **ConstraintLayout** que contiene un **TextView** y un **Button.**



1. Elimine las restricciones de cadena entre **TextView** y **Button**.
2. Agregue otro **TextView** de la paleta y suéltelo cerca del centro de la pantalla. Este **TextView** se utilizará para mostrar un número aleatorio entre 0 y el recuento actual del primer fragmento.
3. Establezca la identificación en **@+id/textview_random** (textview_random en el panel Atributos).
4. Restrinja el borde superior del nuevo **TextView** a la parte inferior del primer **TextView**, el borde izquierdo a la izquierda de la pantalla, el borde derecho a la derecha de la pantalla y la parte inferior a la parte superior del **botón Anterior.**
5. Establezca tanto el ancho como el alto en **wrap_content**.
6. Establezca **textColor** en **@android:color/white**, establezca **textSize** en **72sp** y **textStyle** en **negrita**.

8. Establezca el texto en **"R"**. Este texto es solo un marcador de posición hasta que se genera el número aleatorio.
9. Establezca l**ayout_constraintVertical_bias en 0,45**.

Este **TextView** está restringido en todos los bordes, por lo que es mejor usar un sesgo vertical que los márgenes para ajustar la posición vertical, para ayudar a que el diseño se vea bien en diferentes orientaciones y tamaños de pantalla.

10. Si recibe una advertencia **"Sin restricción horizontal"**, agregue una restricción desde el inicio del botón al lado izquierdo de la pantalla y el final del botón al lado derecho de la pantalla.

Aquí está el código XML para TextView que muestra el número aleatorio:

```
<TextView
    android:id="@+id/textview_random"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="R"
    android:textColor="@android:color/white"
    android:textSize="72sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/button_second"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textview_second"
    app:layout_constraintVertical_bias="0.45" />
```

3.Set top, left and right margins to **24dp**. Left and right margins may also be referred to as "start" and "end" to support localization for right to left languages.
4.Remove any bottom constraint.
5.Set the text color to **@color/colorPrimaryDark** and the text size to **24sp**.
6.In **strings.xml**, change **hello_second_fragment** to "**Here is a random number between 0 and %d.**"
7.Use **Refactor > Rename...** to change the name of **hello_second_fragment** to **random_heading**.

Paso 2: actualice TextView para mostrar el encabezado

1. En **fragment_second.xml**, seleccione **textview_second**, que actualmente tiene el texto **"Hola, segundo fragmento. Arg: %1$s"** en el recurso de cadena **hello_second_fragment.**
2. Si **android:text** no está configurado, configúrelo en el recurso de cadena **hello_second_fragment**.

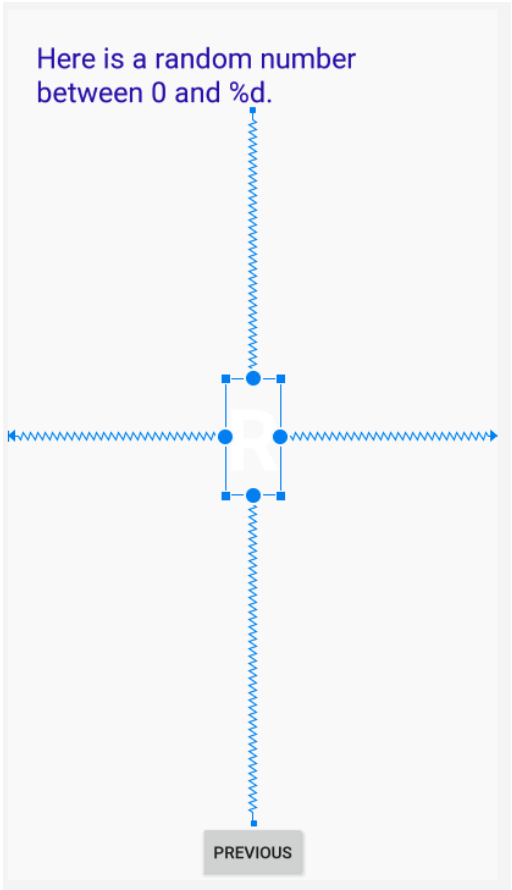
```
android:text="@cadena/hola_segundo_fragmento"
```

3. Cambie la identificación a **textview_header** en el panel Atributos.
4. Establezca el ancho en **match_constraint**, pero establezca la altura en **wrap_content**, de modo que la altura cambie según sea necesario para coincidir con la altura del contenido.

5. Establezca los márgenes superior, izquierdo y derecho en **24 dp**. Los márgenes izquierdo y derecho también pueden denominarse "inicio" y "final" para admitir la localización de idiomas de derecha a izquierda.
6. Elimine cualquier restricción inferior.
7. Establezca el color del texto en **@color/colorPrimaryDark** y el tamaño del texto en **24sp**.
8. En **strings.xml**, cambie **hello_second_fragment** a **"Aquí hay un número aleatorio entre 0 y %d"**.
9. Utilice **Refactorizar** > **Renombrar...** para cambiar el nombre de **hola_segundo_fragmento** a **random_heading**..

Aquí está el código XML para TextView que muestra el encabezado:

```
<TextView
    android:id="@+id/textview_header"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="24dp"
    android:layout_marginEnd="24dp"
    android:layout_marginRight="24dp"
    android:text="@string/random_heading"
    android:textColor="@color/colorPrimaryDark"
    android:textSize="24sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

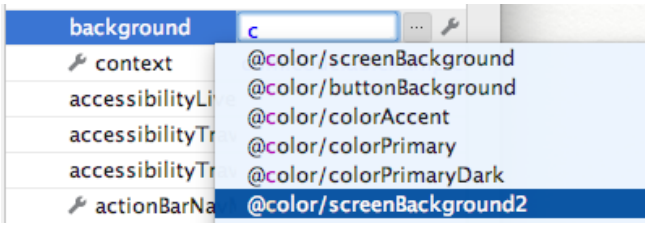


Paso 3: cambia el color de fondo del diseño

Dale a tu nueva actividad un color de fondo diferente al de la primera actividad:
1. En **colors.xml**, agregue un nuevo recurso de color:

```
<color name="screenBackground2">#26C6DA</color>
```

2. En el diseño de la segunda actividad, **fragment_second.xml**, establezca el fondo de **ConstraintLayout** en el nuevo color.
En el panel Atributos:



o en XML:

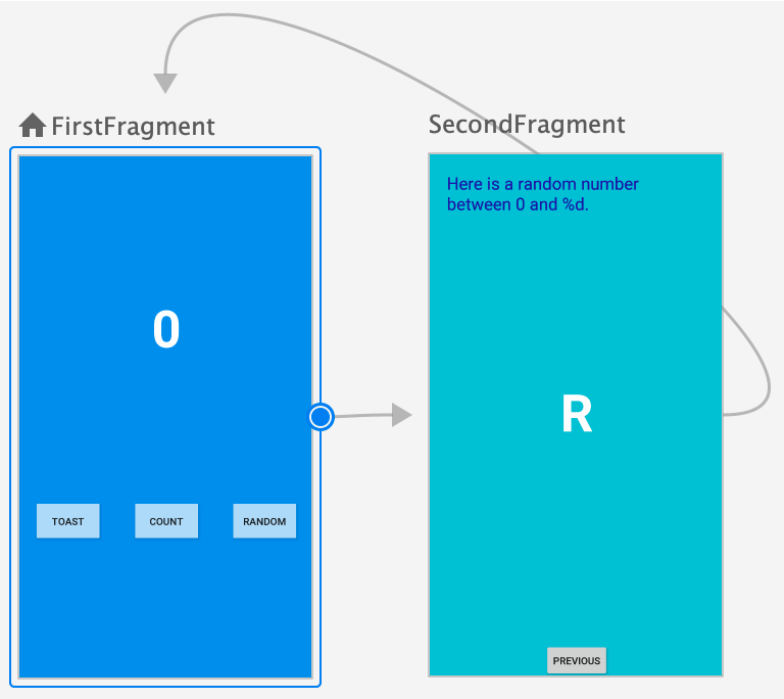
```
android:background="@color/screenBackground2"
```

Su aplicación ahora tiene un diseño completo para el segundo fragmento. Pero si ejecuta su aplicación y presiona el botón Random puede bloquearse.
El Listener (controlador de clics) que Android Studio configuró para ese botón necesita algunos cambios.
En la siguiente tarea, explorará y corregirá este error.

Paso 4: examina el gráfico de navegación

Cuando creó su proyecto, eligió la **Actividad básica** como plantilla para el nuevo proyecto. Cuando Android Studio usa la plantilla de actividad básica para un nuevo proyecto, configura dos fragmentos y un gráfico de navegación para conectar los dos. También configuró un botón para enviar un argumento de cadena desde el primer fragmento al segundo. Este es el botón que cambió al botón Aleatorio. Y ahora desea enviar un número en lugar de una cadena.

1. Abra **nav_graph.xml** (**aplicación > res > navegación > nav_graph.xml**).
Aparece una pantalla similar al Editor de diseño en la vista Diseño. Muestra los dos fragmentos con unas flechas entre ellos. Puede hacer zoom con los botones + y - en la parte inferior derecha, como lo hizo con el Editor de diseño.
2. Puede mover libremente los elementos en el editor de navegación. Por ejemplo, si los fragmentos aparecen con **SecondFragment** a la izquierda, arrastre **FirstFragment** a la izquierda de **SecondFragment** para que aparezcan en el orden en que trabaja con ellos.



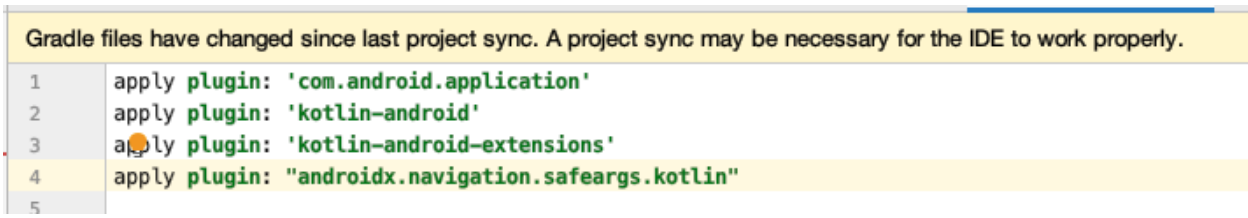
Paso 5: habilite SafeArgs
Esto habilitará SafeArgs en Android Studio.
1. Abra **Gradle Scripts > build.gradle** (**Proyecto: Mi primera aplicación**)
2. Busque la sección de **dependencias** en la sección **buildscript** y agregue las siguientes líneas después de las otras entradas de **classpath**:

```
def nav_version = "2.3.0-alpha04"
classpath "androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
```

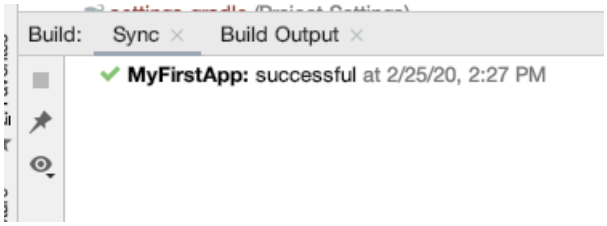
3. Abra **Gradle Scripts > build.gradle** (**Módulo: app**)
4. Justo debajo de las otras líneas que comienzan con aplicar el complemento, agregue una línea para habilitar SafeArgs:

```
aplicar complemento: 'androidx.navigation.safeargs'
```

5. Android Studio debería mostrar un mensaje sobre los archivos de Gradle que se están modificando. Haga clic en Sincronizar ahora en el lado derecho.



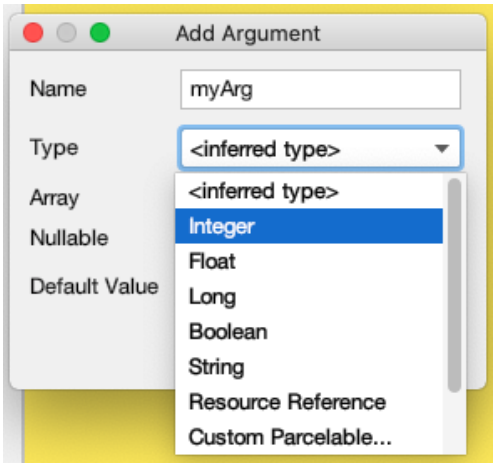
Después de unos momentos, Android Studio debería mostrar un mensaje en la pestaña Sincronizar que se realizó correctamente:



6. Elija **Build > Make Project**. Esto debería reconstruir todo para que Android Studio pueda encontrar **FirstFragmentDirections**.
Solución de problemas: si la sincronización no fue exitosa, confirme que agregó las líneas correctas al archivo Gradle correcto.

Paso 6: crea el argumento para la acción de navegación

1. En el gráfico de navegación, haga clic en **FirstFragment** y observe el panel Atributos a la derecha. (Si el panel no se muestra, haga clic en la etiqueta Atributos vertical a la derecha).
2. En la **sección Acciones**, muestra qué acción sucederá para la navegación, es decir, ir a **SecondFragment.**
3. Haga clic en SecondFragment y observe el panel Atributos.
La sección Argumentos muestra **Nada que mostrar**.
4. Haga clic en el + en la sección Argumentos.
5. En el cuadro de diálogo Agregar argumento, ingrese myArg para el nombre y establezca el tipo en Entero, luego haga clic en el botón Agregar.



Paso 7: envía el conteo al segundo fragmento

Android Studio configuró el botón **Next/Random** para pasar del primer fragmento al segundo, pero no envía ninguna información.
En este paso, lo cambiará para enviar un número para el conteo actual. Obtendrá el recuento actual de la vista de texto que lo muestra y lo pasará al segundo fragmento.

1. Abra **FirstFragment.java** (**aplicación > java > com.example.myfirstapp > FirstFragment**)
2. Busque el método **onViewCreated()** y observe el código que configura el detector de clics para pasar del primer fragmento al segundo.
3. Reemplace el código en ese detector de clics con una línea para encontrar la vista de texto de conteo, **textview_first**.

```
int currentCount = Integer.parseInt(showCountTextView.getText().toString());
```

4. Cree una acción con currentCount como argumento para **actionFirstFragmentToSecondFragment().**

```
FirstFragmentDirections.ActionFirstFragmentToSecondFragment action =
FirstFragmentDirections.actionFirstFragmentToSecondFragment(currentCount);
```

5. Agregue una línea para encontrar el controlador de navegación y navegue con la acción que creó.

```
NavHostFragment.findNavController(FirstFragment.this).navigate(acción);
```

Aquí está el método completo, incluido el código que agregó anteriormente:

```
public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    view.findViewById(R.id.random_button).setOnClickListener(new View.OnClickListener()
{
        @Override
        public void onClick(View view) {
            int currentCount = Integer.parseInt(showCountTextView.getText().toString());
            FirstFragmentDirections.ActionFirstFragmentToSecondFragment action =
FirstFragmentDirections.actionFirstFragmentToSecondFragment(currentCount);
            NavHostFragment.findNavController(FirstFragment.this).navigate(action);
        }
    });
    view.findViewById(R.id.toast_button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Toast myToast = Toast.makeText(getActivity(), "Hello toast!",
Toast.LENGTH_SHORT);
            myToast.show();
        }
    });
    view.findViewById(R.id.count_button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            countMe(view);
        }
    });
}
```

6. Ejecute su aplicación. Haga clic en el botón Contar varias veces. Ahora, cuando presiona el botón **Random**, la segunda pantalla muestra la cadena correcta en el encabezado, pero aún no cuenta ni tiene un número aleatorio, porque necesita escribir un código para hacer eso.

Paso 8: actualice SecondFragment para calcular y mostrar un número aleatorio
Ha escrito el código para enviar el recuento actual al segundo fragmento. El siguiente paso es agregar código a **SecondFragment.java** para recuperar y usar el conteo actual.

1. En **SecondFragment.java**, agregue una importación para navArgs a la lista de bibliotecas importadas.

```
importar androidx.navigation.fragment.navArgs;
```

2.En el método **onViewCreated()** debajo de la línea que comienza con **super**, agregue código para obtener el conteo actual, obtenga la cadena y formatéela con el conteo, y luego configúrelo para textview_header.

```
Integer count = SecondFragmentArgs.fromBundle(getArguments()).getMyArg();
String countText = getString(R.string.random_heading, count);
TextView headerView = view.getRootView().findViewById(R.id.textview_header);
headerView.setText(countText);
```

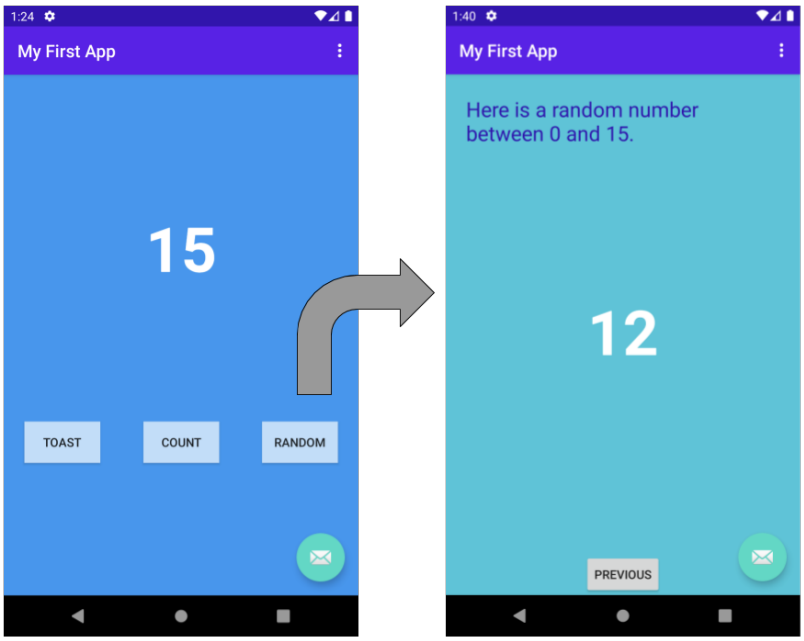
3. Obtenga un número aleatorio entre 0 y el contador

```
Random random = new java.util.Random();
Integer randomNumber = 0;
if (count > 0) {
    randomNumber = random.nextInt(count + 1);
}
```

4. Agregue código para convertir ese número en una cadena y configúrelo como texto para textview_random.

```
TextView randomView = view.getRootView().findViewById(R.id.textview_random);
randomView.setText(randomNumber.toString());
```

5. Ejecute la aplicación. Presione el botón Contar varias veces, luego presione el botón Aleatorio. ¿La aplicación muestra un número aleatorio en la nueva actividad?



¡Si llego hasta acá ha finalizado la app!