



Elementos de la interfaz, navegación e interacción

Elementos de interacción (Parte I)

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



¿Qué aprenderás en esta sesión?

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

¿Qué necesitamos para
poder usar una vista
(button, textview) a nivel
de código?

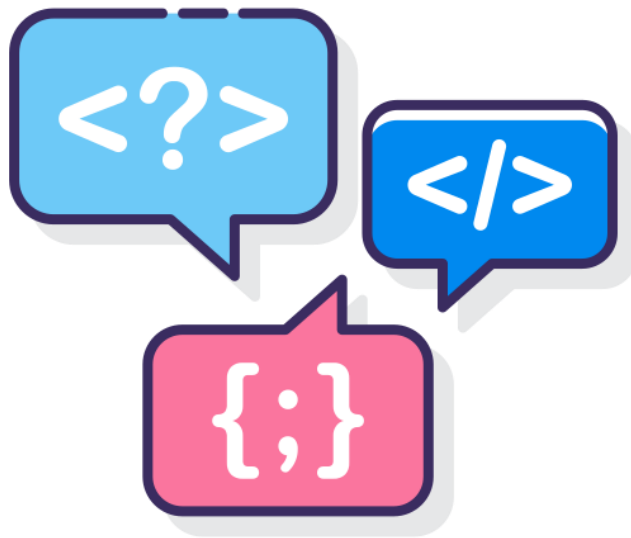


/* La clase R */

Clase R

Paquete

- Es una clase generada al compilar la app y contiene un ID único para cada recurso del directorio **res/** (XML).
- Contiene las definiciones de todos los recursos dentro del paquete.
- Por ejemplo, para el paquete **cl.dal.android** el archivo R generado está en el espacio de nombres **cl.dal.android.R**.



Clase R

Subclases

Para cada tipo de recurso hay una subclase de R que identifica los recursos de cada directorio de recursos.

Dentro de la subclase, se identifica con un número entero estático único a cada uno de los recursos disponibles.

Fuente:

<https://developer.android.com/reference/android/R>

R

R.anim
R.animator
R.array
R.attr
R.bool
R.color
R.dimen
R.drawable
R.fraction
R.id
R.integer
R.interpolator
R.layout
R.menu
R.mipmap
R.plurals
R.raw
R.string
R.style
R.styleable
R.transition
R.xml

/* View Bindings */

View Bindings

¿Cómo utilizarlo?

Declaramos nuestro diseño en el layout, para que luego en la actividad/fragmento, se enlace con el layout que se quiera utilizar.

¿Cómo ocurre esto?

- **findViewById** es la primera forma que se usó para recuperar una vista y programar su comportamiento.
- En la actualidad, fue reemplazada por **viewBinding** que se encarga de recuperar las vistas automáticamente.

setContentView

¿Qué es?

```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

setContentView() es el método encargado de enlazar el contenido visual (layout XML) con la actividad o fragmento.

findViewById

¿Cómo se usa?

- Desde el árbol de vistas, se obtiene una vista que coincide con el ID indicado.
- Se encuentra y retorna la primera vista con el ID indicado, de no ser así, se entrega **null**, y puede generar un `NullPointerException` si se utiliza.
- Una desventaja es que se debe llamar por cada una de las vistas que se van a utilizar.

```
TextView title =  
findViewById(R.id.storeTitle)  
;
```

findViewById

Layout XML

```
<TextView
    android:id="@+id/storeTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    ... />
```

```
android:id="@+id/storeTitle"
```

- Para poder utilizar findViewById, la vista debe tener declarado un **id**.
- El **null** se devuelve si el ID no se encuentra en el árbol de vistas.
- Es importante que los IDs sean autoexplicativos para su fácil reconocimiento.

findViewById

Activity / Fragment

- Luego de ejecutado setContentView(...), se debe recuperarla vista usando findViewById.
- `actionButton` es un atributo de la clase y puede ser reutilizado en diferentes métodos de la clase
- `TextView title` es una variable local y NO puede ser reutilizado.

```
public class MainActivity extends AppCompatActivity {  
    Button actionButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        TextView title = findViewById(R.id.storeTitle);  
        actionButton = findViewById(R.id.button);  
        actionButton.setText(title);  
    }  
}
```

Conversemos

1. ¿Qué pasa si el layout es más complejo?
2. ¿Qué pasa si queremos utilizar 20 vistas?
3. ¿Qué pasa si una vista no es inicializada y se ocupa?



`/* View Binding */`

ViewBinding

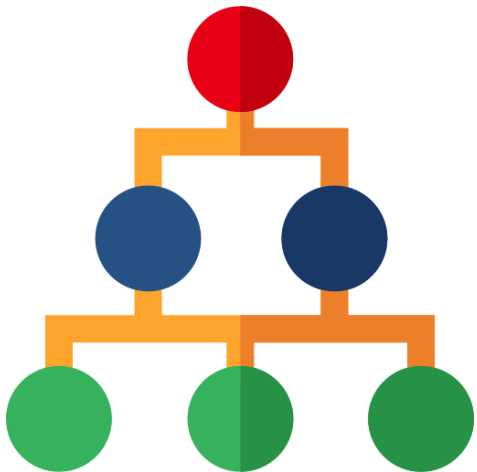
¿Qué es?

- View Binding es una biblioteca que permite escribir un código que interactúa con las vistas de forma más limpia.
- Es una **clase generada** (*binding class*) por cada layout XML.
- Viene activado en el archivo `build.gradle` del módulo, indicado en `buildFeatures`.

```
plugins {  
    id 'com.android.application'  
}  
android {  
    compileSdk 33  
    defaultConfig { . . . }  
    buildTypes { . . . }  
    compileOptions {  
        sourceCompatibility JavaVersion.VERSION_1_8  
        targetCompatibility JavaVersion.VERSION_1_8  
    }  
    buildFeatures {  
        viewBinding true  
    }  
}  
dependencies { . . . }
```


ViewBinding

¿En qué consiste?



- En el proceso de compilación se genera automáticamente una binding class por cada layout XML de la app.
- La binding class contiene una referencia a cada vista contenida en el layout. Las vistas están ordenadas por un árbol jerárquico con una raíz que es el contenedor principal.

ViewBinding

¿Cómo se genera el nombre?

El nombre de la clase de vinculación se genera convirtiendo el nombre del archivo XML, según la convención de mayúsculas y minúsculas, y agregando la palabra "Binding" al final. (Fuente: <https://developer.android.com/topic/libraries/view-binding>)

Por ejemplo:

Nombre de layout	Nombre de binding class generada
activity_main.xml	ActivityMainBinding
fragment_the_third.xml	FragmentTheThirdBinding
result_profile.xml	ResultProfileBinding

ViewBinding

Utilización

La binding class
ActivityMainBinding
permite acceder a
cualquier vista declarada
en el layout.

{desafío}
latam_

```
public class MainActivity extends AppCompatActivity {  
  
    // Se declara como atributo de la clase para tener  
    // acceso en distintos métodos de la actividad  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // Se infla la vista directamente usando la binding class  
        binding = ActivityMainBinding.inflate(getLayoutInflater());  
  
        // Se asigna el contenido de la binding class a la actividad.  
        setContentView(binding.getRoot());  
  
        // Se puede acceder a cualquier vista usando la binding class  
        // y el ID de la vista  
        setSupportActionBar(binding.toolbar);  
        binding.fab.setOnClickListener(new View.OnClickListener() {  
            ...  
        });  
    }  
}
```

ViewBinding

Error común



Llamar a **setContentView(...)** con el ID del layout:

- Se infla 2 veces el layout.
- Los listener se instalan en el layout equivocado.

```
public class MainActivity extends AppCompatActivity {  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = ActivityMainBinding.inflate(getLayoutInflater());  
  
        setContentView(R.layout.activity_main);  
  
        ...  
    }  
}
```

```
setContentView(binding.getRoot());
```

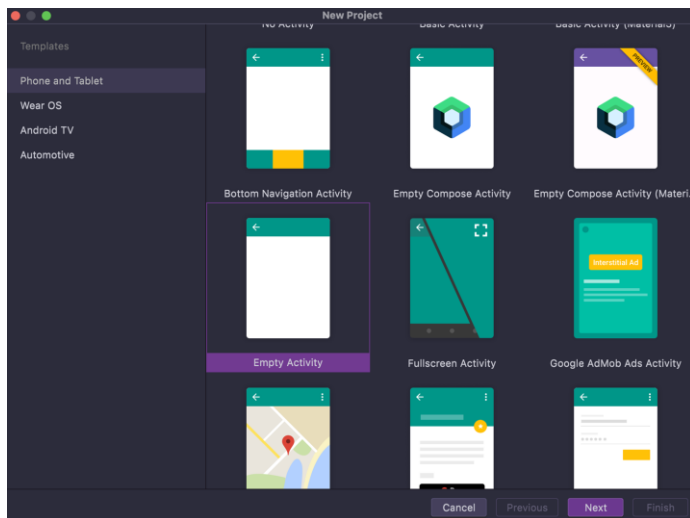
Recuerda usar binding para inicializar setContentView.

Ejemplo de View Binding

1. Crear proyecto

ViewBinding

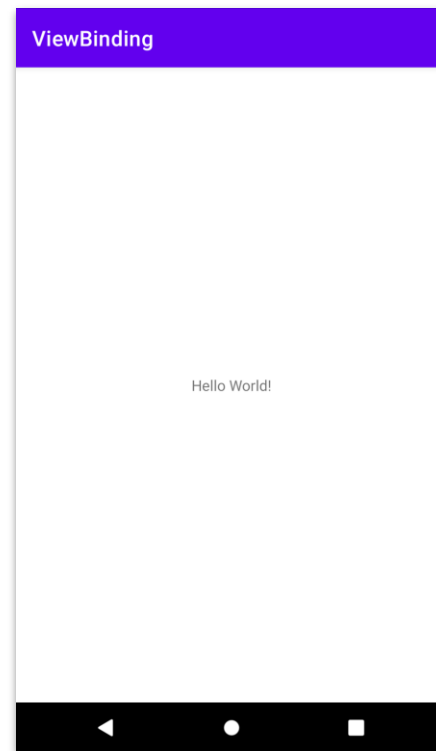
Nuevo proyecto desde Template Empty Activity



{desafío}
latam_

Fuente: ADL.

La app corriendo



Fuente: ADL.

2. Habilitar view binding

ViewBinding

- Se debe agregar la configuración buildFeatures al archivo build.gradle del módulo.
- Recuerda que siempre hay que sincronizar luego de modificar el build.gradle.

```
android {  
    compileSdk 32  
    . . .  
    compileOptions {  
        . . .  
    }  
    buildFeatures {  
        viewBinding true  
    }  
}
```

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

[Sync Now](#)

[Ignore these changes](#)

Fuente: ADL.

3. Instanciar binding class

ViewBinding

```
public class MainActivity extends AppCompatActivity {  
  
    private ActivityMainBinding binding;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding =  
        ActivityMainBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());  
    }  
}
```

- Modificamos MainActivity para tener la *binding class* como atributo.
- En onCreate se debe inflar la vista usando la *binding class*, para luego, pasar el árbol de vistas a **setContentView(...)**

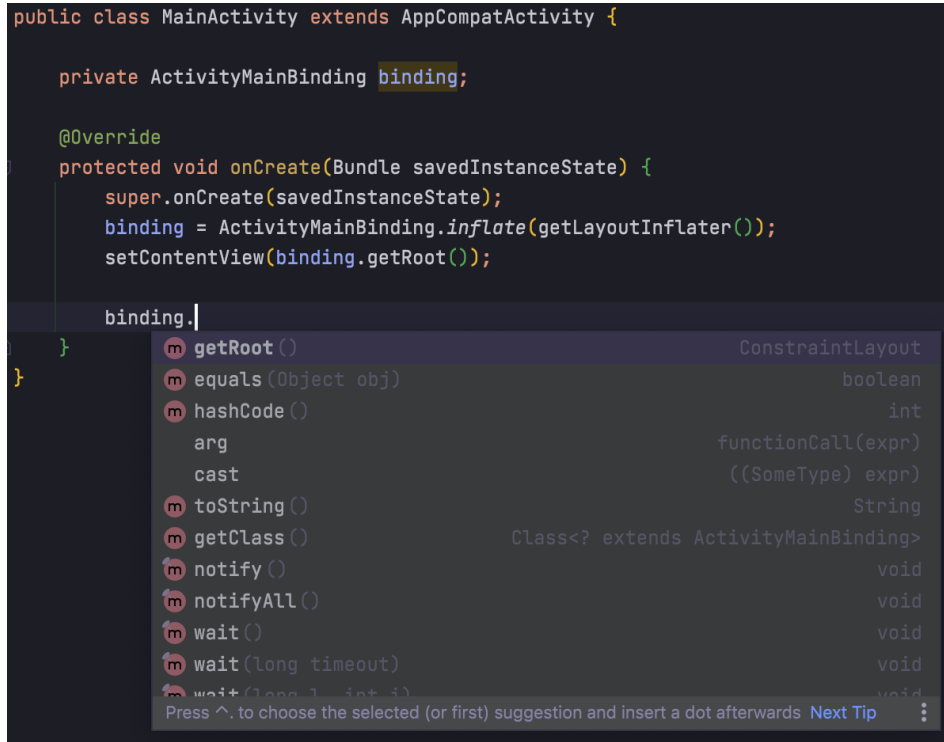
4. Utilizar la binding class

ViewBinding

- Al escribir la variable binding, se debe agregar el punto y esperar que el autocompletador nos muestre lo que está disponible de la variable.
- Sin embargo, no aparece el TextView que tiene el mensaje “hello world”, ¿por qué?

{desafío}
latam_

```
public class MainActivity extends AppCompatActivity {  
  
    private ActivityMainBinding binding;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding = ActivityMainBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());  
  
        binding.|  
    }  
}
```



Fuente: ADL.

4. Utilizar la binding class usando ID

ViewBinding

- El TextView del layout activity_main NO tiene un id definido, por lo que NO se define dentro de la binding class.
- Agregando el ID estará disponible en la binding class.

```
<androidx.constraintlayout.widget.ConstraintLayout ...>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        .../>
</androidx.constraintlayout.widget.ConstraintLayout>
```

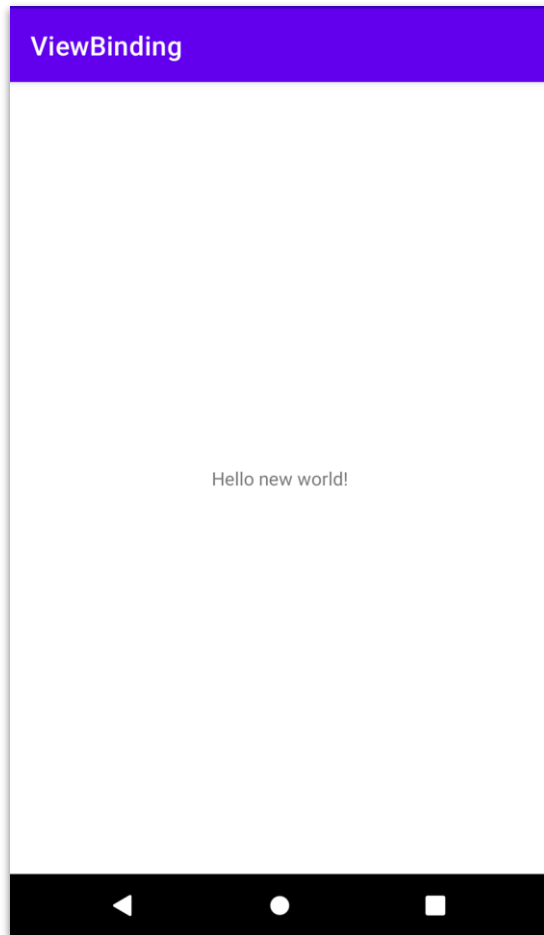
```
<androidx.constraintlayout.widget.ConstraintLayout ...>
    <TextView
        android:id="@+id/helloWorldText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        .../>
</androidx.constraintlayout.widget.ConstraintLayout>
```

5. Cambiar texto del TextView

ViewBinding

Accedemos al TextView con la *binding class*, y usando el método `setText()` podemos cambiar el texto a mostrar.

```
public class MainActivity extends AppCompatActivity {  
    private ActivityMainBinding binding;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        binding =  
        ActivityMainBinding.inflate(getLayoutInflater());  
        setContentView(binding.getRoot());  
        binding.helloWorldText.setText("Hello new world!");  
    }  
}
```



Conversemos

Queremos agregar una nueva vista y modificar su texto cuando se ejecute la app:

- ¿Cuáles son los pasos que necesito?
- ¿Cuál es el elemento clave para poder identificar la vista en la binding class?



Comparación

	findViewById()	View Binding
Legibilidad	<pre>Button button = findViewById(R.id.button); TextView textView = findViewById(R.id.textViewMain); button.setVisibility(View.GONE); textView.setVisibility(View.VISIBLE);</pre>	<pre>binding = ActivityMainBinding.inflate(getLayoutInflater()); binding.button.setVisibility(View.GONE); binding.textViewMain.setVisibility(View.VISIBLE);</pre>
Type safety	<pre>TextView button = findViewById(R.id.button); TextView textView = findViewById(R.id.textViewMain);</pre>	<pre>binding.button.setVisibility(View.VISIBLE);</pre>
Null safety	Para layouts definidos en múltiples configuraciones, si una vista está presente sólo en algunas.	
	Se cae durante la ejecución lanzando <u>NullPointerException</u> .	Las detecta y crea propiedades <u>@Nullable</u> .

En tus palabras,
¿Qué es y para qué sirve la
binding class?





Próxima sesión...

- *Utilizar Event Listeners en las vistas para el manejo de la interacción del usuario y del comportamiento a la pantalla.*

{desafío}
latam_

*Academia de
talentos digitales*

