

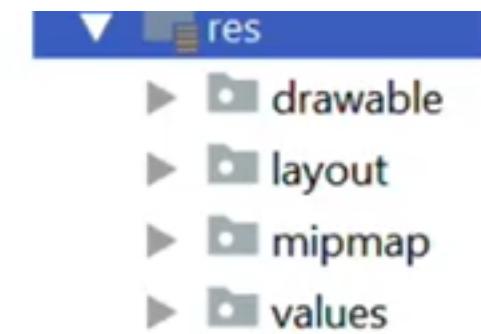
Programación Multimedia y Dispositivos Móviles



UF1: Análisis de tecnologías para app

Estructura de una app

CARPETAS DE RECURSOS



- **drawable**: ficheros de imágenes y descriptores de imágenes (.png, .jpg, .xml).
- **mipmap**: igual que drawables pero no pueden ser reescalados por el sistema.
- **layout**: ficheros XML con vistas de la aplicación. Permiten diseñar las pantallas de la aplicación.
- **values**: Valores de cadenas, colores y estilos ...
- **menu**: ficheros XML con los menús de la aplicación.
- **anim**: ficheros XML con descriptores de animaciones.
- **xml**: otros ficheros XML requeridos por la aplicación.
- **raw** ficheros adicionales que no se encuentran en formato XML.
- **doc**: documentación asociada al proyecto.

UF1: Análisis de tecnologías para app

Android Manifest

Su función es declarar una serie de metadatos de la aplicación que el dispositivo debe conocer antes de instalarla.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.favoritetoy">

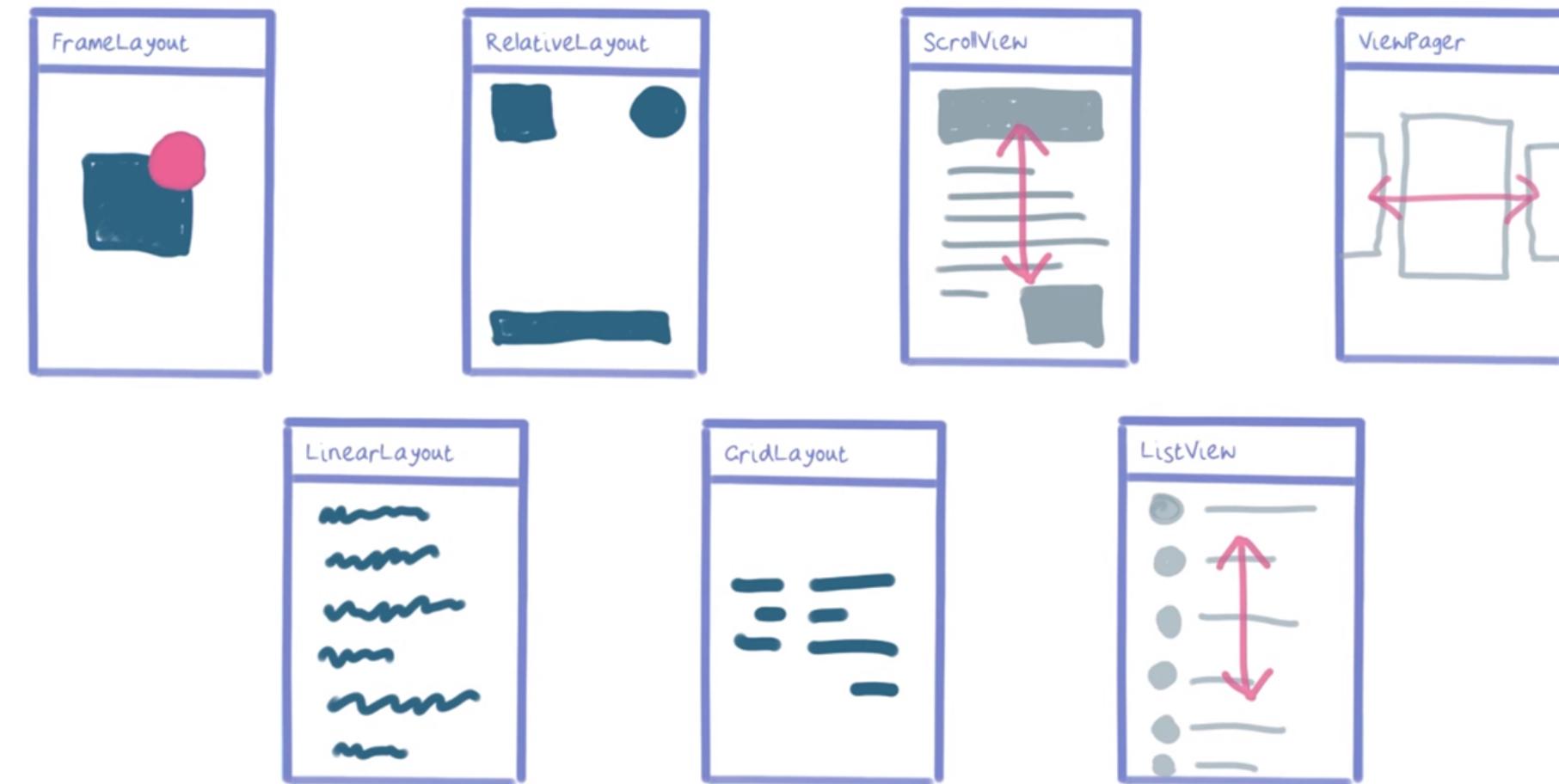
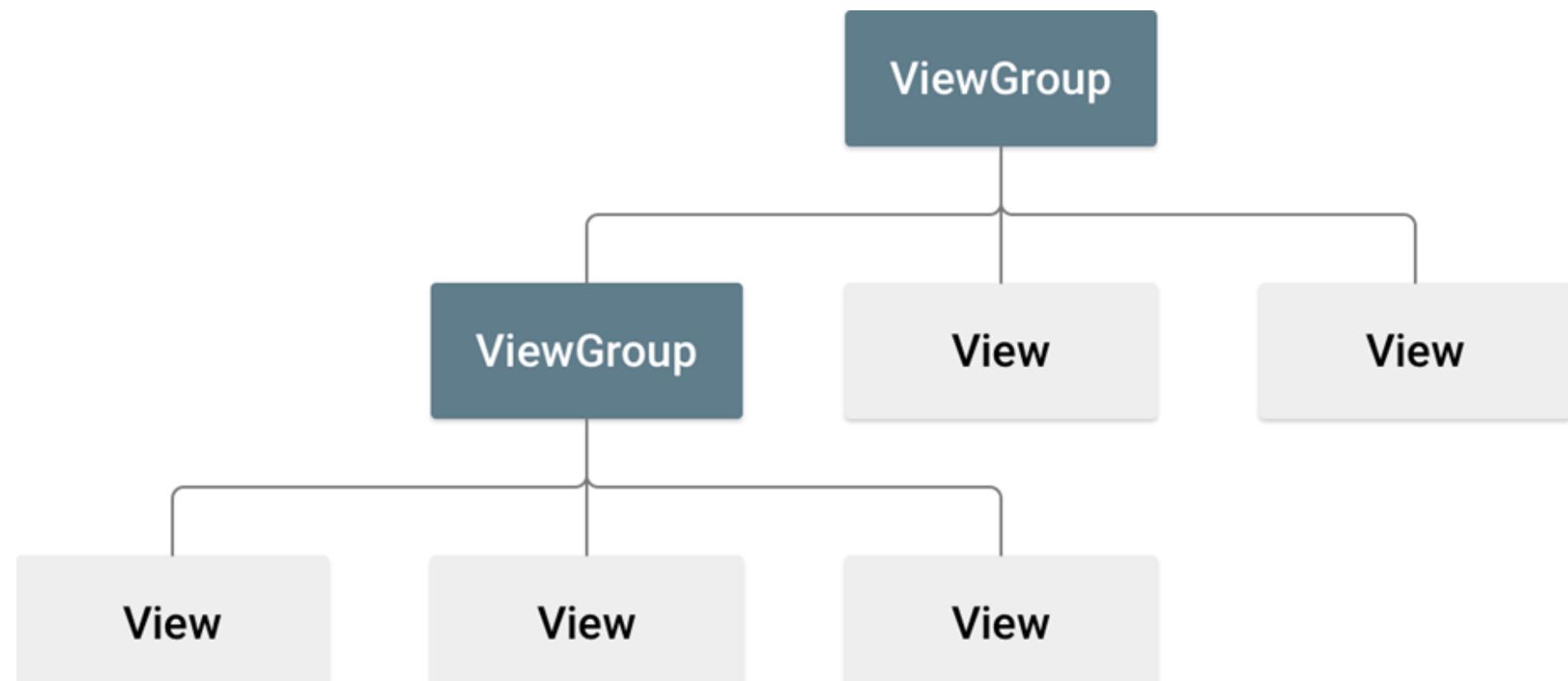
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name="com.example.android.favoritetoy.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

UF1: Análisis de tecnologías para app

Layouts

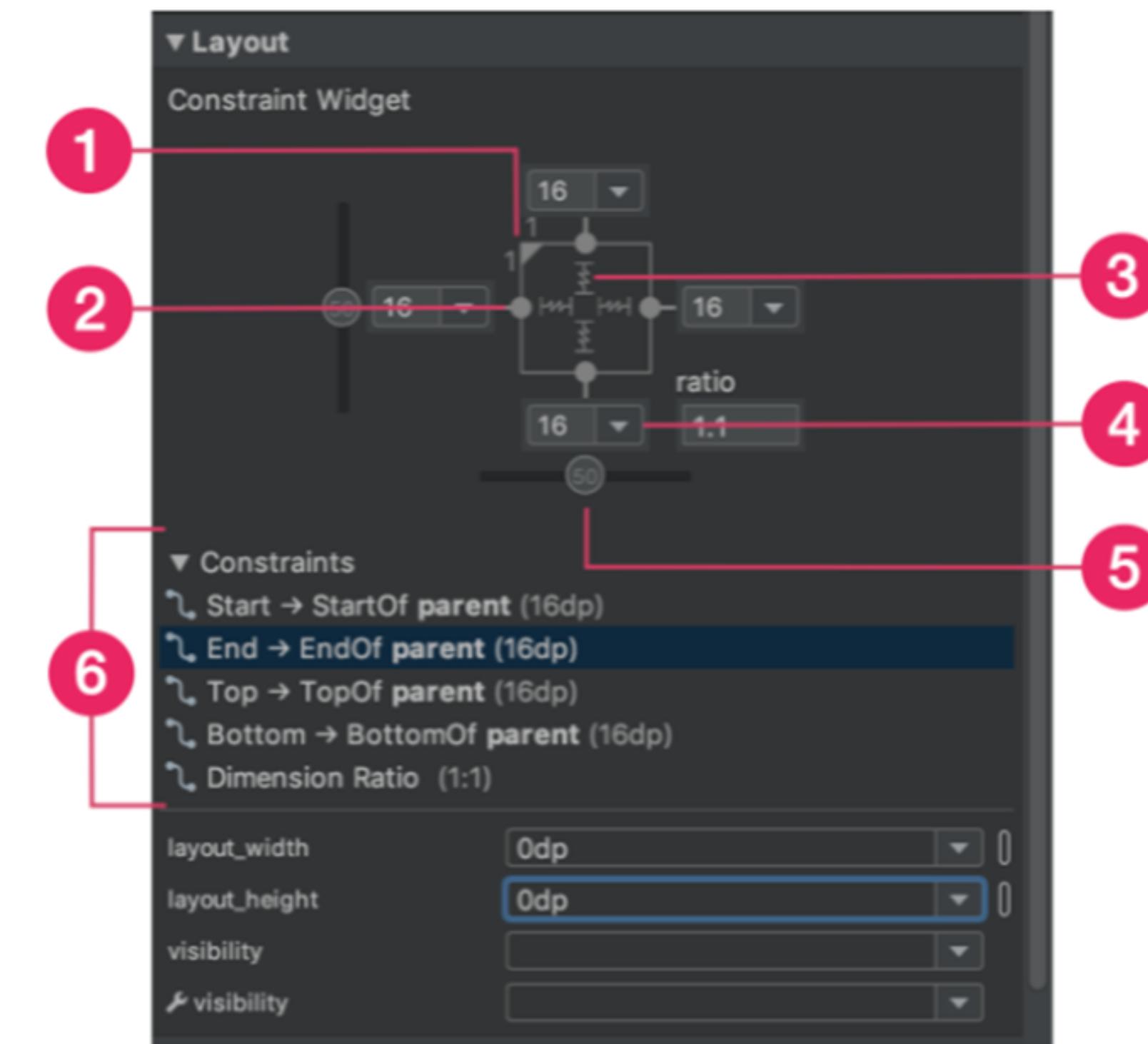
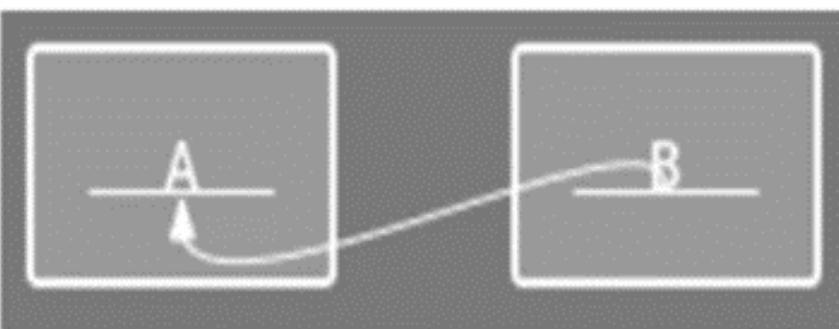
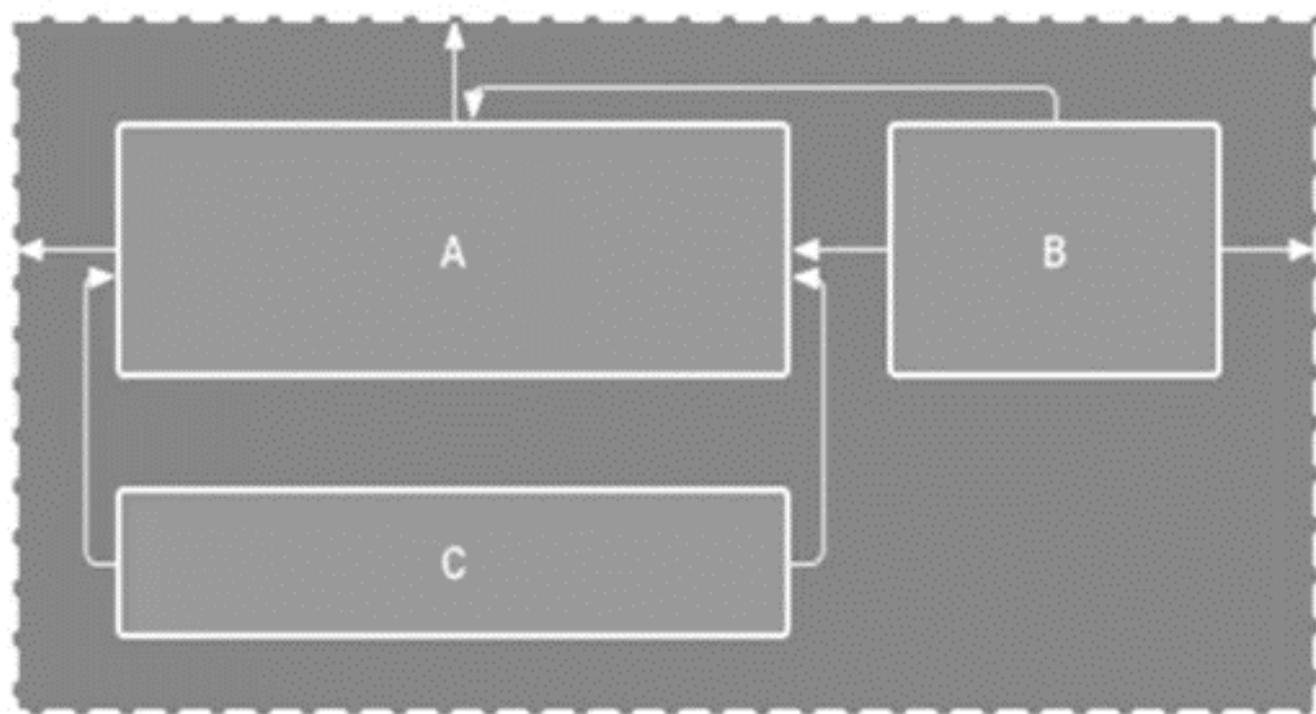
- La UI es un árbol anidado (jerarquía). Este árbol define la estructura completa de la interfaz de usuario (las vistas), desde la raíz hasta las hojas, como texto, botones o imágenes. Todo lo que ves en la pantalla de tu app es una vista en algún lugar de esta jerarquía.
- Puedes definir la UI creando estos objetos por código, o declarar su diseño en un archivo XML. Este enfoque, te permite separar la presentación del comportamiento de la aplicación.



UF1: Análisis de tecnologías para app

Layouts

- Pensado para diseño visual en lugar de diseño por medio de código XML.
- Relación de un punto de anclaje con:
 - Contenedor.
 - Otra vista.
 - Línea base de texto.



Cuando seleccionas una vista, la ventana **Attributes** incluye controles para ① proporción de tamaño, ② borrar restricciones, ③ modo de altura/ancho, ④ márgenes y ⑤ sesgo de restricciones. También puedes destacar restricciones individuales en el editor de diseño haciendo clic en ellas en la ⑥ lista de restricciones.

UF1: Análisis de tecnologías para app

Animaciones

Utilizando como nodo raíz <set>, es posible especificar múltiples animaciones que se ejecutarán todas a la vez (si no se especifica un offset).

```
<set xmlns:android="..."  
      android:duration="1000"  
      android:fillAfter="true">  
    <scale  
        android:fromXScale="100%"  
        android:fromYScale="100%"  
        android:toXScale="0%"  
        android:toYScale="0%"  
        android:pivotX="50%"  
        android:pivotY="50%" />  
    <rotate  
        android:fromDegrees="0"  
        android:toDegrees="360"  
        android:pivotX="50%"  
        android:pivotY="50%" />  
  </set>
```

1. Declarar un atributo del tipo correspondiente al view que se quiere animar o de tipo general View.
2. Inicializarla variable con la referencia al view, para ello buscar la view por su identificador.
3. Cargar el fichero de animación en un objeto Animation.
4. Enlazarla animación con el view.

```
View imagen;  
imagen = findViewById(R.id.imagen);  
Animation rotate = AnimationUtils.loadAnimation(this, R.anim.ficheroAnimacion);  
imagen.startAnimation(rotate);
```

codetomg.com

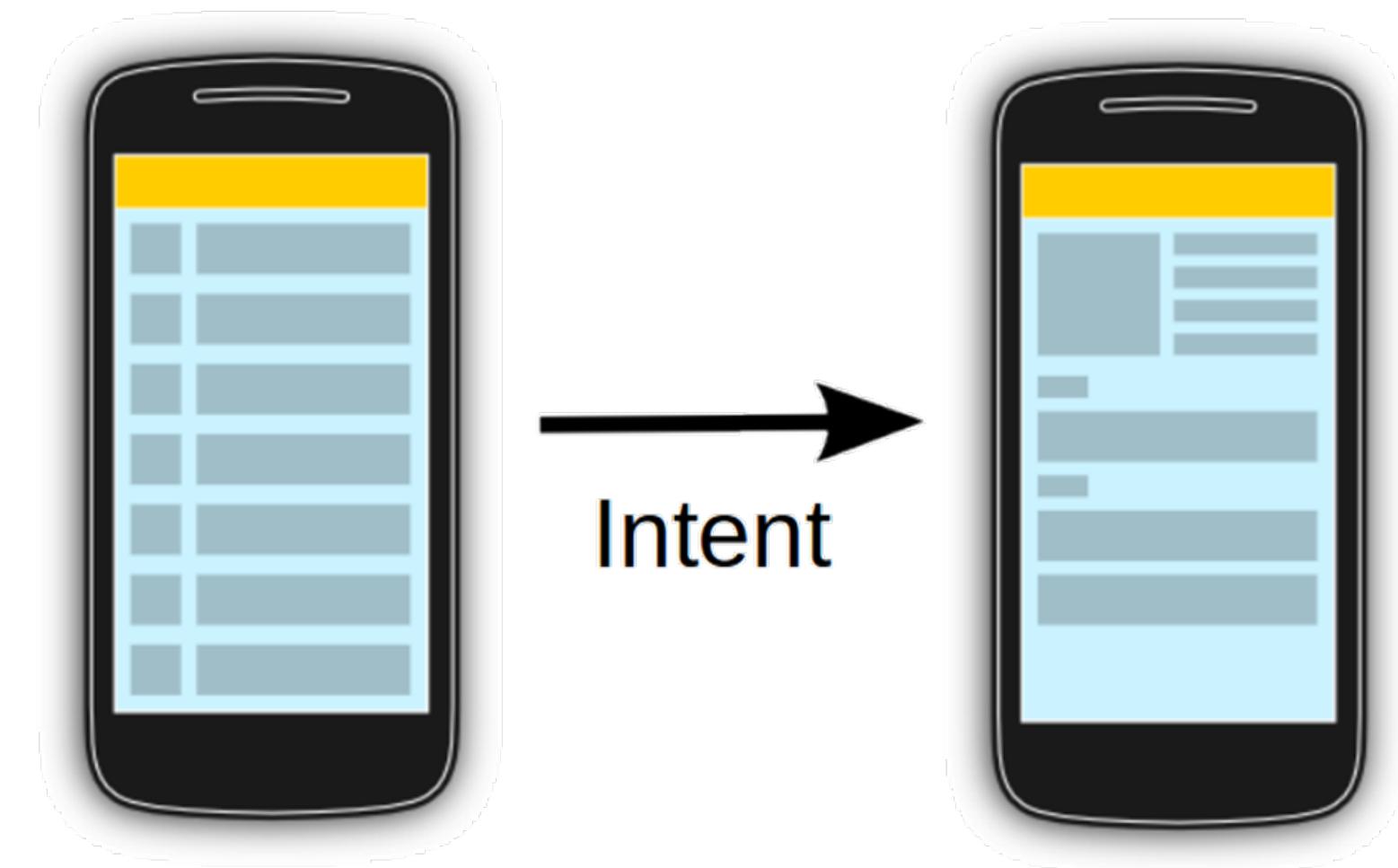
UF1: Análisis de tecnologías para app

Cambio de activity

La forma más simple de iniciar una *activity* usando un objeto *Intent* es pasarlo como un argumento al método ***startActivity()***.

```
startActivity(new Intent(Splash.this, MainActivity.class));
```

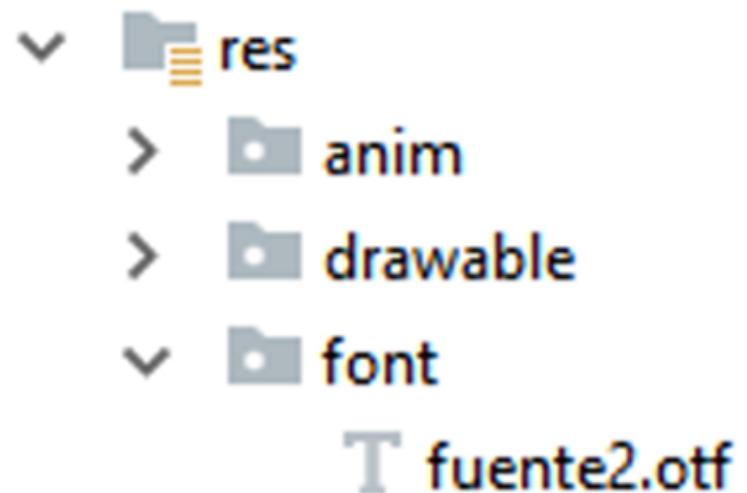
codetomg.com



UF2: Programación de apps para móviles

Creación de una fuente

1. Creación de un directorio de recursos dentro de la carpeta res de tipo Font
2. Copiar en ella el archivo de fuente:



3. En el archivo xml, en la view donde se vaya a poner la fuente:

```
android:fontFamily = "@font/fuente2"
```

UF2: Programación de apps para móviles

Paleta de colores: res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <color name="colorPrimary">#7B32C9</color>
    <color name="colorPrimaryDark">#491E65</color>
    <color name="colorAccent">#B829A7</color>

</resources>
```

UF2: Programación de apps para móviles

Cadenas de texto: res/values/strings.xml

```
<resources>
    <string name="app_name">My Application</string>
    <string name="mis_tareas">Mis Tareas</string>
    <string name="itt">ITT</string>
    <string name="usuario">Usuario</string>
    <string name="pass">Contraseña</string>
    <string name="login">LOGIN</string>
    <string name="crea_una_cuenta">Crea una cuenta</string>
    <string name="fondo">fondo</string>
</resources>
```

UF2: Programación de apps para móviles

Estilos: res/values/themes.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="etiquetas">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:layout_centerHorizontal">true</item>
        <item name="android:textColor">@android:color/white</item>
        <item name="android:textStyle">bold</item>
    </style>
</resources>
```

```
<TextView
    android:id="@+id/subtitle"
    style="@style/etiquetas"
    android:layout_below="@id/title"
    android:layout_marginTop="10dp"
    android:text="@string/itt"
    android:textSize="20sp" />
```

UF2: Programación de apps para móviles

Eventos: onClick Resource

Si todo lo que se requiere es **que se llame a un método *callback* cuando un usuario haga clic en una *view* de botón en la interfaz de usuario**, existe un acceso directo disponible.

Imaginemos un diseño de interfaz que contenga una vista de botón, llamada **button1**, con el requisito de que cuando el usuario toque el botón, se llame a un método llamado **buttonClick()** declarado en la clase de actividad.

Todo lo que se requiere para implementar este comportamiento es escribir el método **buttonClick()** (que toma como argumento una referencia a la vista que activó el evento de clic) y agrega una sola línea a la declaración de la vista de botón en el archivo XML).

```
<Button  
...  
    android:onClick = 'login' />
```

```
public void login(View view){  
...  
}
```

UF2: Programación de apps para móviles

Eventos: listener

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        findViewById(R.id.button).setOnClickListener(this);  
    }  
  
    @Override  
    public void onClick(View view) {  
    }  
}
```

UF2: Programación de apps para móviles

Notificaciones: Toast

Un **toast** es un **mensaje que se muestra en pantalla al usuario durante unos segundos para luego desaparecer automáticamente.**

```
Toast toast = Toast.makeText(this,"Funcionalidad no implementada", Toast.LENGTH_LONG );
toast.show();
```

Cómo modificar la posición del *toast* en la pantalla
método ***setGravity()***.

Podremos indicar a este método **en qué zona deseamos que aparezca la notificación** con alguna de las constantes definidas en la clase *Gravity*:

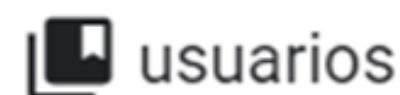
- **CENTER**
- **LEFT**
- **BOTTOM**

```
toast1.setGravity(Gravity.CENTER|Gravity.LEFT, 0, 0);
```

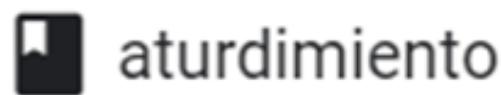
UF2: Programación de apps para móviles

BBDD: Cloud Firestore

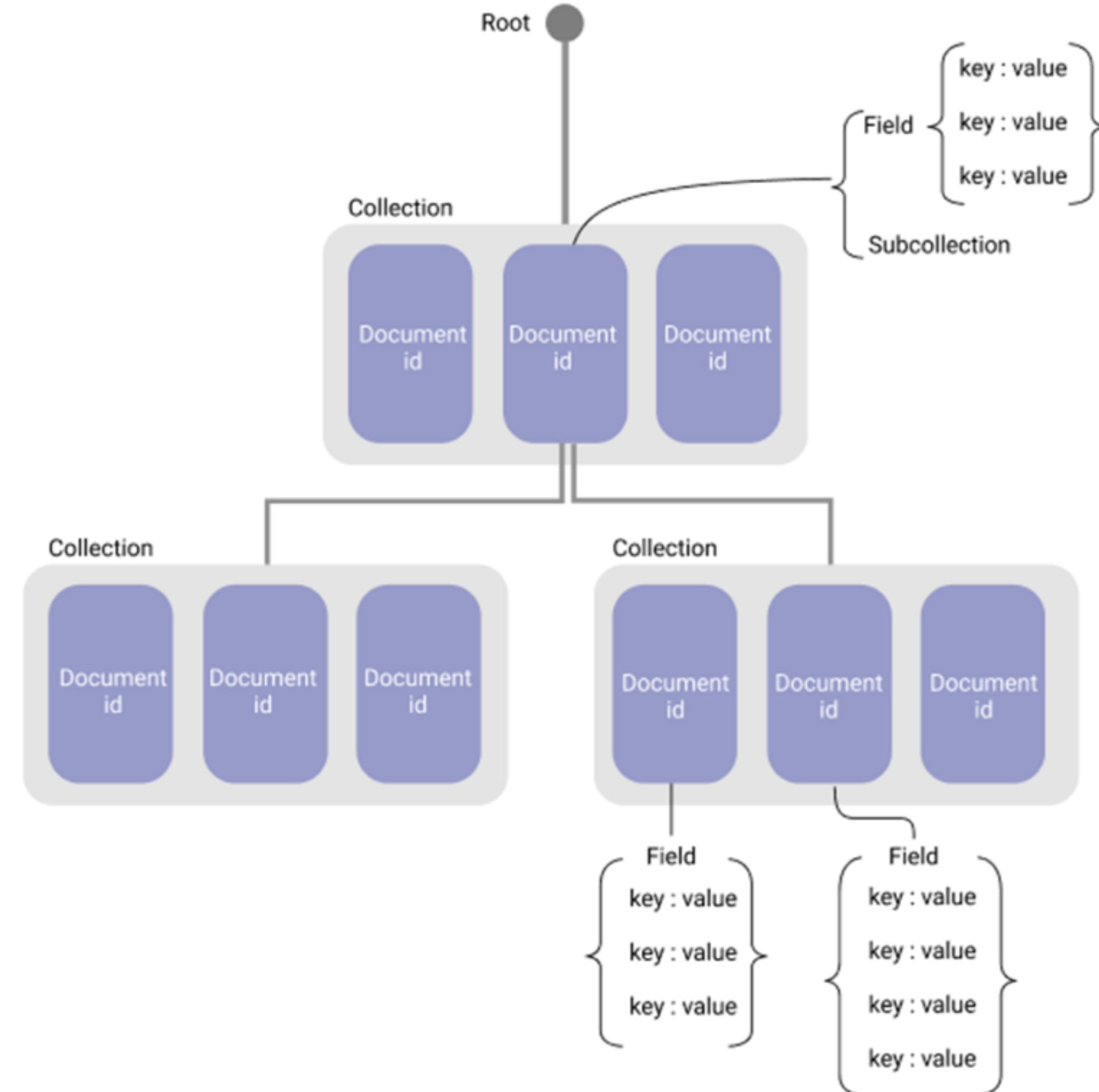
Los documentos viven en colecciones, que son simplemente contenedores para documentos. Por ejemplo, podría tener una colección de `users` para contener varios usuarios, cada uno representado por un documento:



```
first : "Ada"  
last : "Lovelace"  
born : 1815
```



```
first : "Alan"  
last : "Turing"  
born : 1912
```



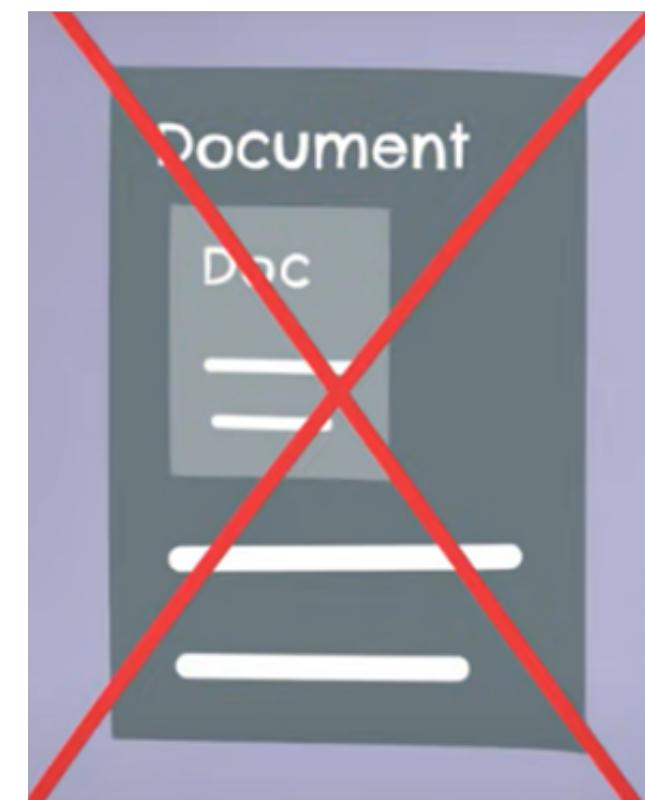
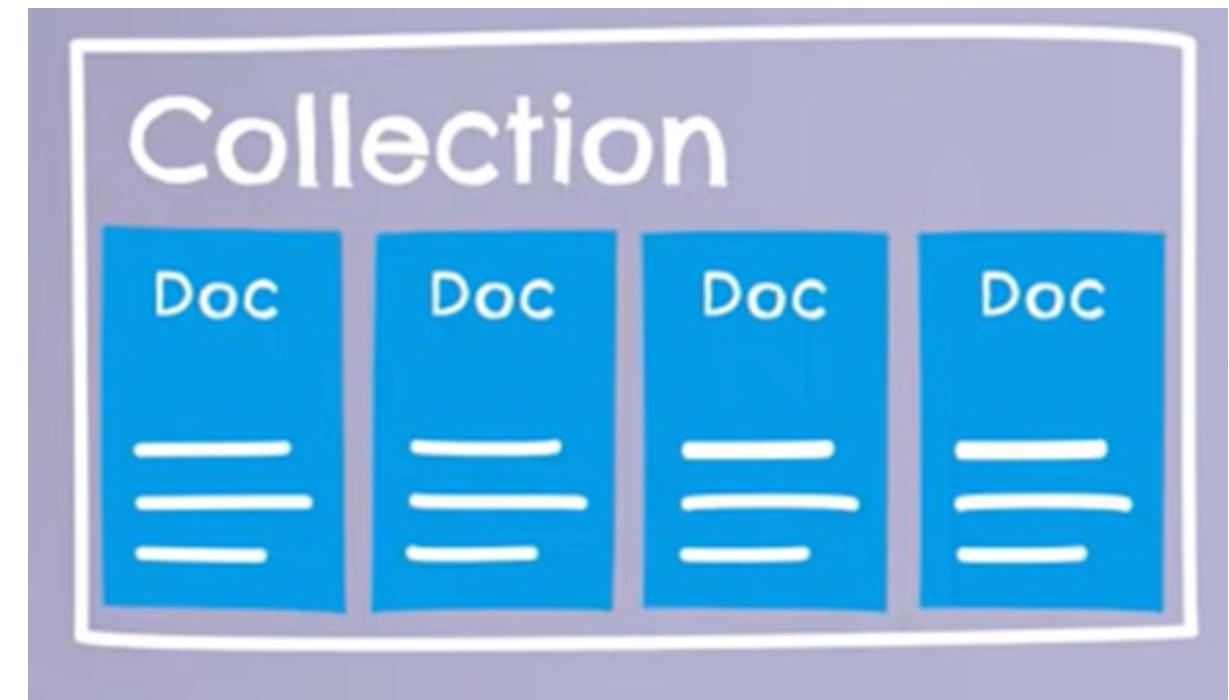
UF2: Programación de apps para móviles

BBDD: Cloud Firestore

1. Una colección solo puede contener documentos.
2. Un documento debe tener un tamaño menor de 1MB.



3. Un documento no puede contener otro documento, un doc puede apuntar a otra colección, pero nunca a otro doc directamente.



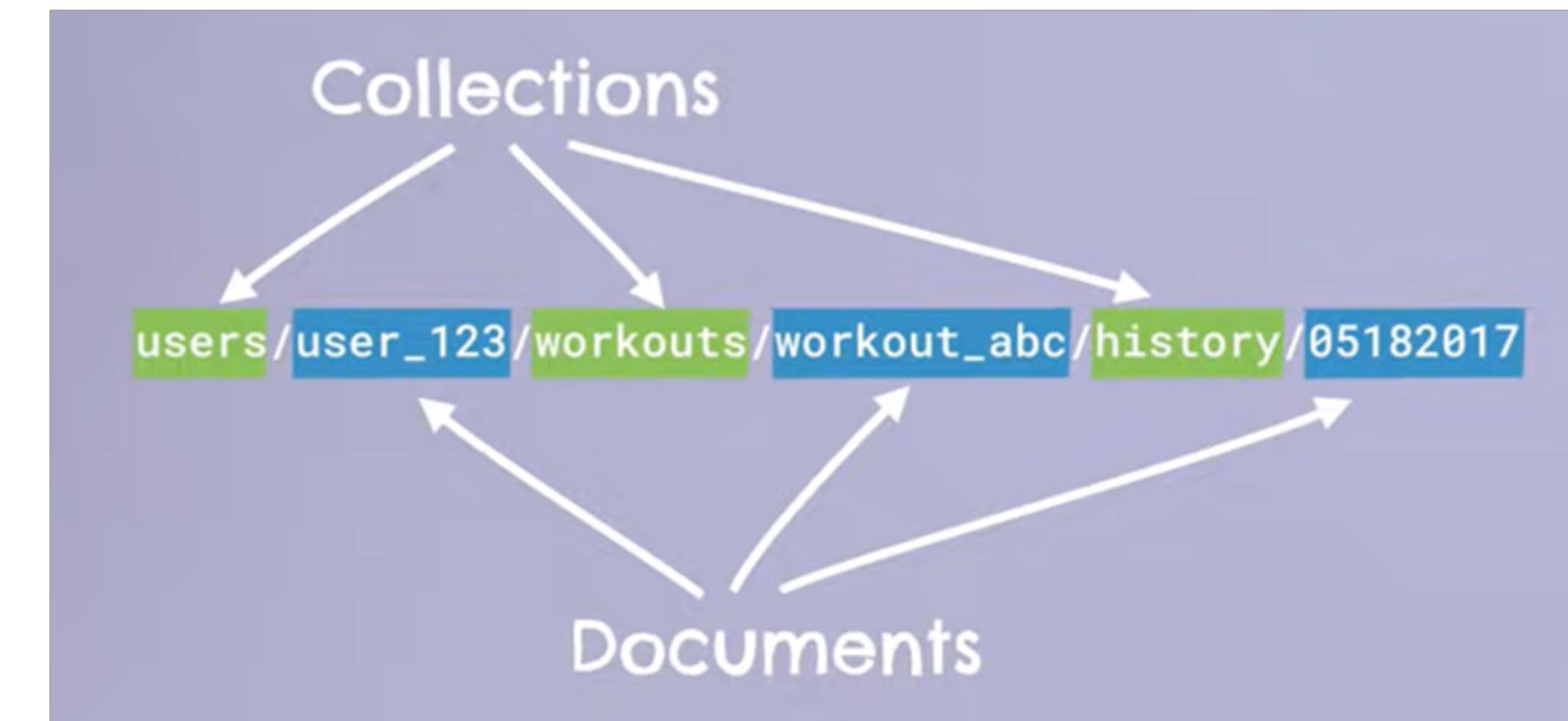
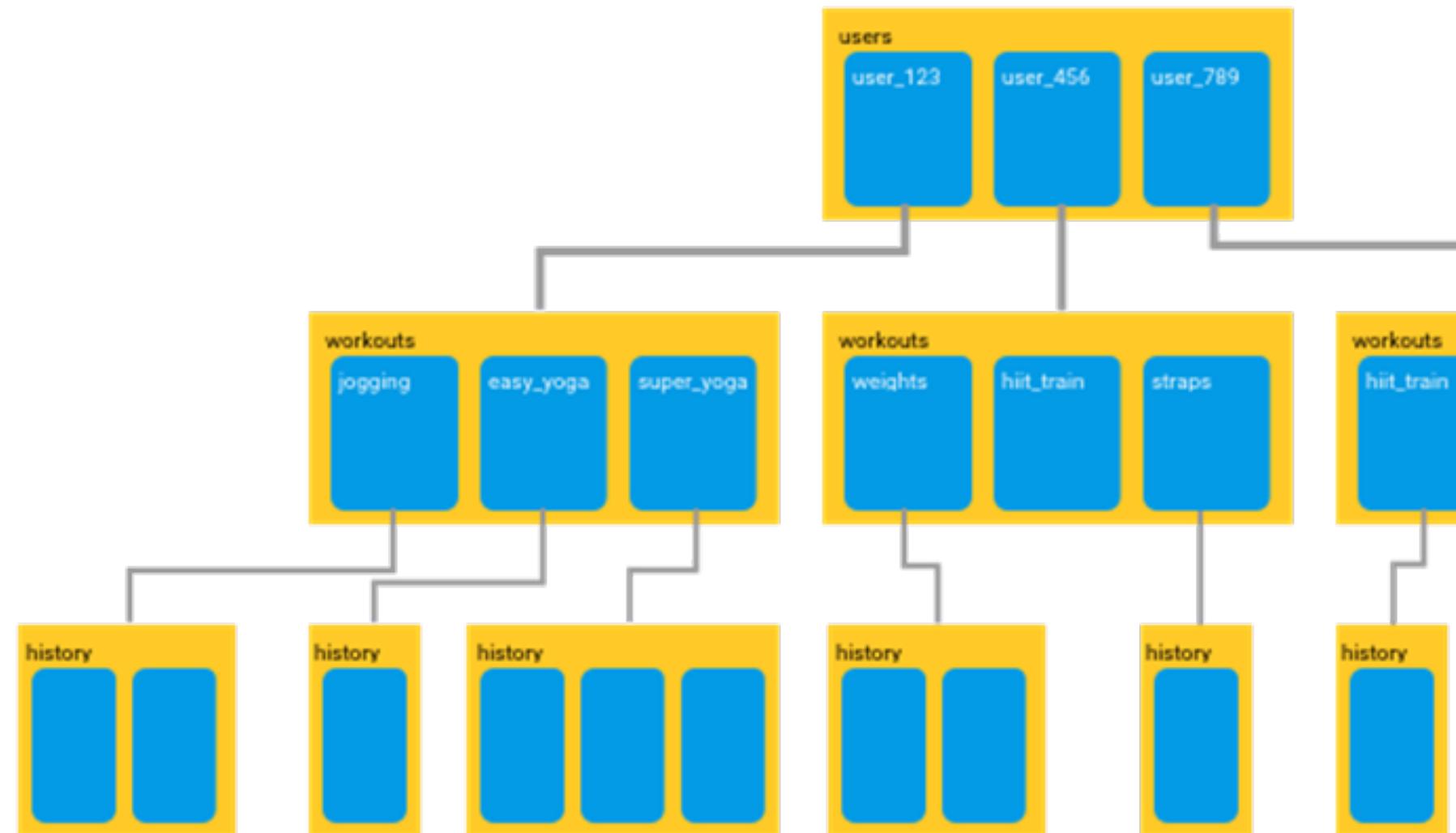
UF2: Programación de apps para móviles

BBDD: Cloud Firestore

```
firestore.collection(...).document(...)  
.collection(...).document(...)
```

→

```
firestore.document  
("/users/user_123/workouts/workout_abc/history/05182017")
```



UF2: Programación de apps para móviles

BBDD: Cloud Firestore

1. Inicializar Cloud Firestore

```
FirebaseFirestore db = FirebaseFirestore.getInstance();
```

2. Agregar datos

```
// Create a new user with a first and last name
Map<String, Object> user = new HashMap<>();
user.put("first", "Ada");
user.put("last", "Lovelace");
user.put("born", 1815);

// Add a new document with a generated ID
db.collection("users").add(user);
```

También se puede agregar un documento con el método **set()**, especificando un ID para el documento que se va a crear:

```
db.collection("users")
.document("ada-lovelace")
.set(user);
```

UF2: Programación de apps para móviles

BBDD: Cloud Firestore

3. Obtener un documento

Obtención del documento

Ruta al documento

Obtención de los datos del documento

```
DocumentReference docRef = db.collection("cities").document("SF");
docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
        if (task.isSuccessful()) {
            DocumentSnapshot document = task.getResult();
            if (document.exists()) {
                Log.d(TAG, "DocumentSnapshot data: " + document.getData());
            } else {
                Log.d(TAG, "No such document");
            }
        } else {
            Log.d(TAG, "get failed with ", task.getException());
        }
    }
});
```

UF2: Programación de apps para móviles

BBDD: Cloud Firestore

4. Obtener varios documentos

```
db.collection("cities")
    .get()
    .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
        @Override
        public void onComplete(@NonNull Task<QuerySnapshot> task) {
            if (task.isSuccessful()) {
                for (QueryDocumentSnapshot document : task.getResult()) {
                    Log.d(TAG, document.getId() + " => " + document.getData());
                }
            } else {
                Log.d(TAG, "Error getting documents: ", task.getException());
            }
        }
    });

```

Obtención de todos los documentos de la colección

Recorrer cada uno de los documentos y obtener sus datos

UF2: Programación de apps para móviles

BBDD: Cloud Firestore

5. Obtener documentos que cumplan una condición

```
db.collection("...").whereEqualTo("key", valor).get();
```

6. Actualizar un documento

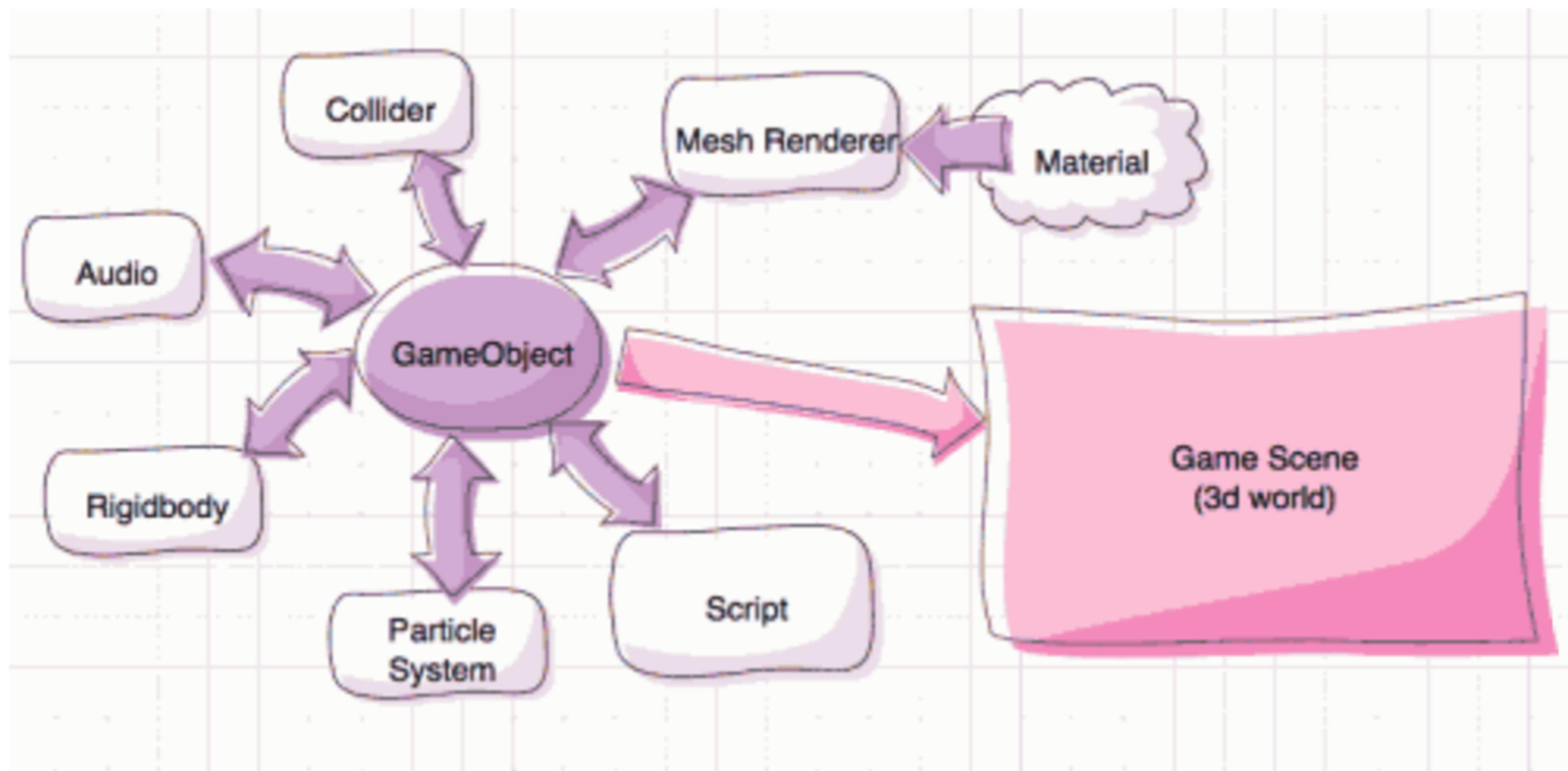
```
db.collection("...").document("...").update("key", valor);
```

7. Borrar un documento

```
db.collection("...").document("...").delete();
```

UF4 y 5: Desarrollo de juegos 2D y 3D

GameObjects y componentes



UF4 y 5: Desarrollo de juegos 2D y 3D

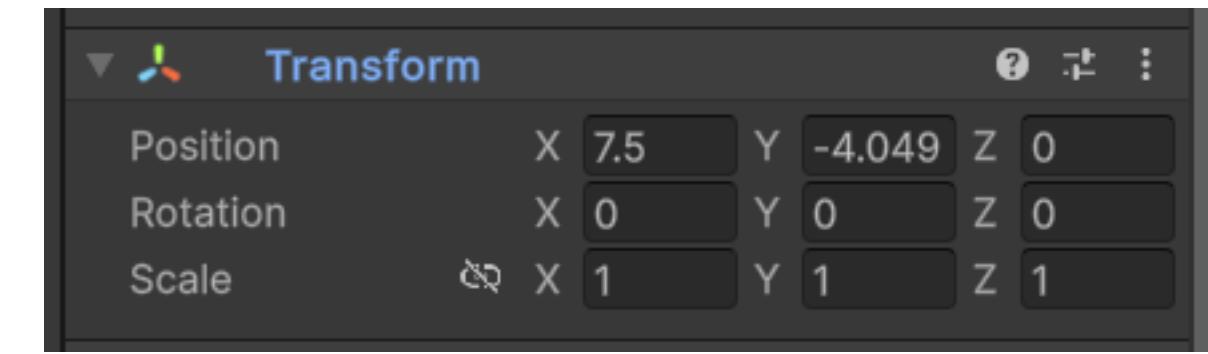
Movimiento

```
moveHorizontal = Input.GetAxis("Horizontal");

if(moveHorizontal < 0)
{
    render.flipX = true;
}

if (moveHorizontal > 0)
{
    render.flipX = false;
}

transform.position += new Vector3(moveHorizontal,0,0) * Time.deltaTime * speed;
```



UF4 y 5: Desarrollo de juegos 2D y 3D

Colisiones: componente collider

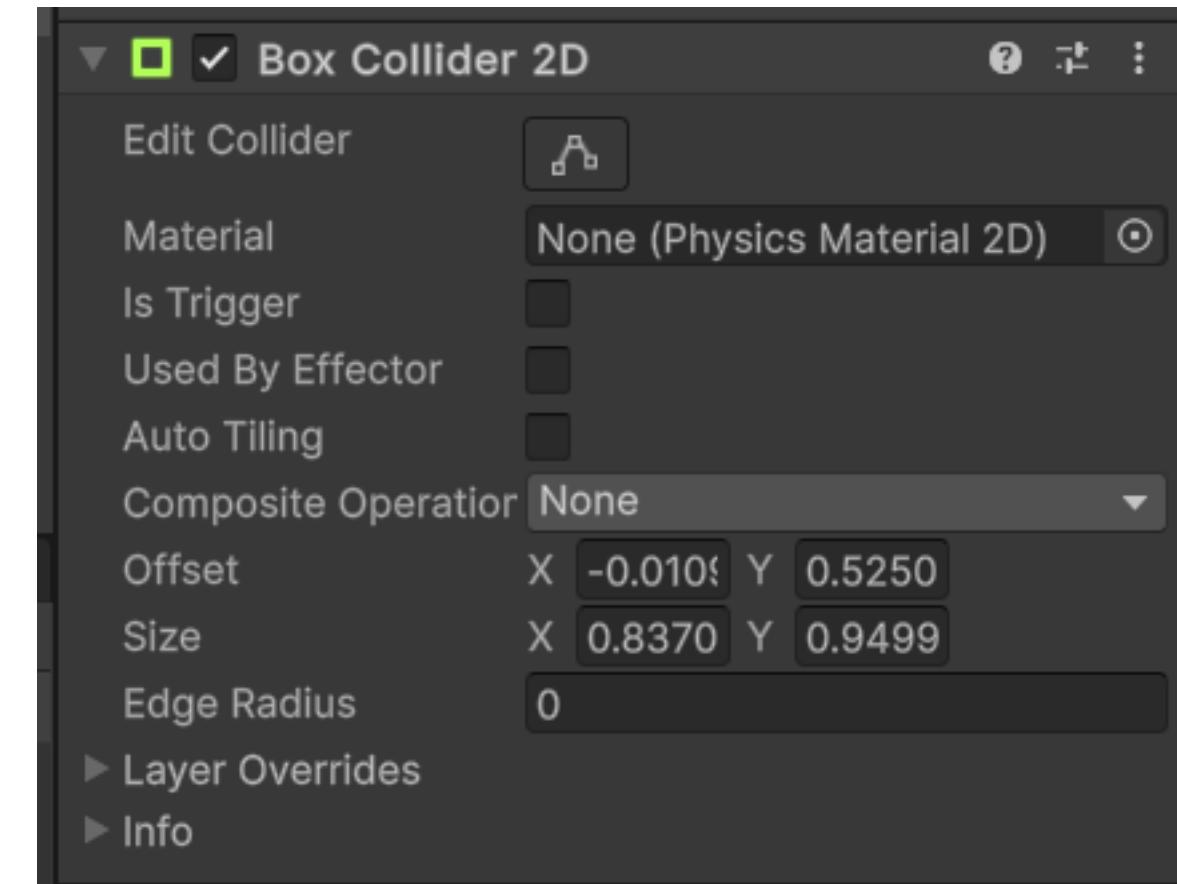
Para detectar colisiones entre dos gameobjects deberán tener componente Collider. Además, el collider espera chocar con un cuerpo rígido, por lo que, al menos uno de los dos, deberá tener un rigidbody.

Un collider define una superficie en la que se detectaran colisiones. Si se detecta una colisión, el collider emite el evento OnCollisionEnter:

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Ground")
    {
        isOnGround = true;
    }
    ...
}
```

Un trigger, no es sólido, deja que los objetos lo atraviesen y define un volumen en el que si entra otro collider o trigger, se disparará el evento OnTriggerEnter:

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision.tag == "Diamond")
    {
        audioSourcePlayer.PlayOneShot(diamondAudioClip);
        points++;
    }
    ...
}
```

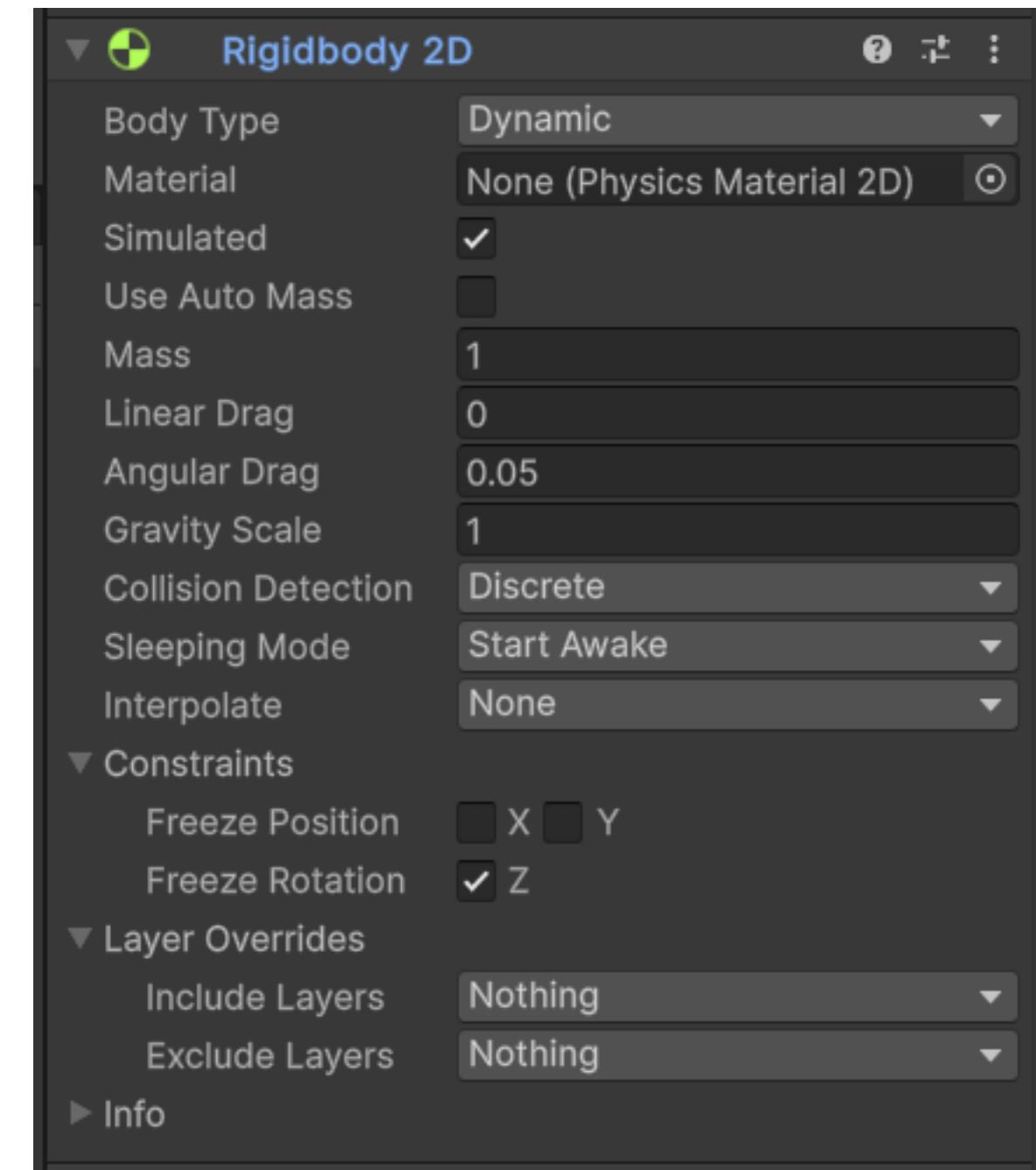


UF4 y 5: Desarrollo de juegos 2D y 3D

Motor físico: componente Rigidbody

1. Declarar una variable del tipo de la componente.
2. Inicializarla variable en el método start().
3. Utilizarla en cualquier otro método.

```
private Rigidbody2D rigidPlayer;  
  
rigidPlayer = GetComponent<Rigidbody2D>();  
  
if (Input.GetKeyDown(KeyCode.UpArrow) && isOnGround){  
    rigidPlayer.AddForce(Vector2.up * jumpForce, ForceMode2D.Impulse);  
}
```



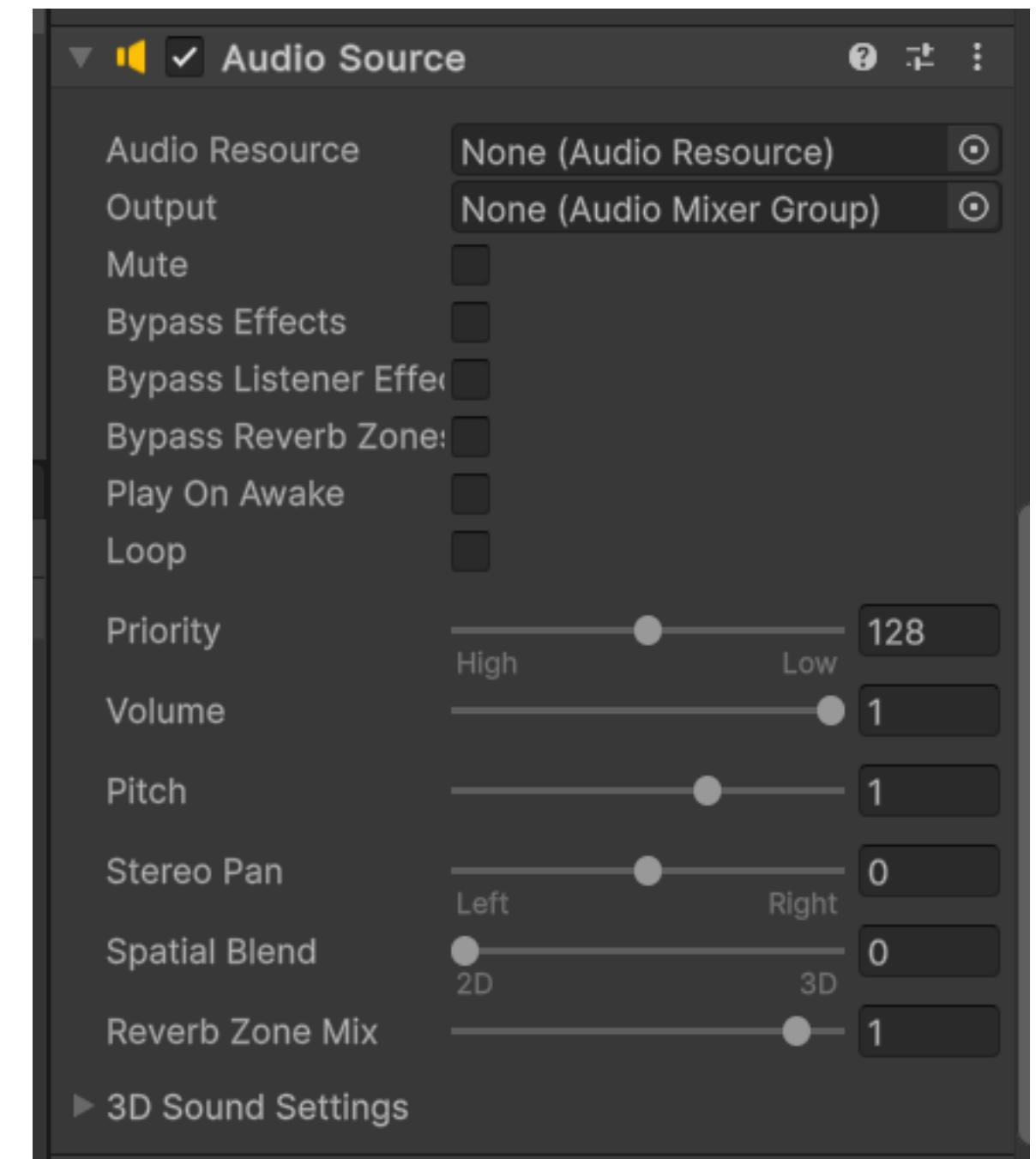
UF4 y 5: Desarrollo de juegos 2D y 3D

Audio: componente AudioSource

El AudioSource dispone de un AudioClip (archivo de sonido).

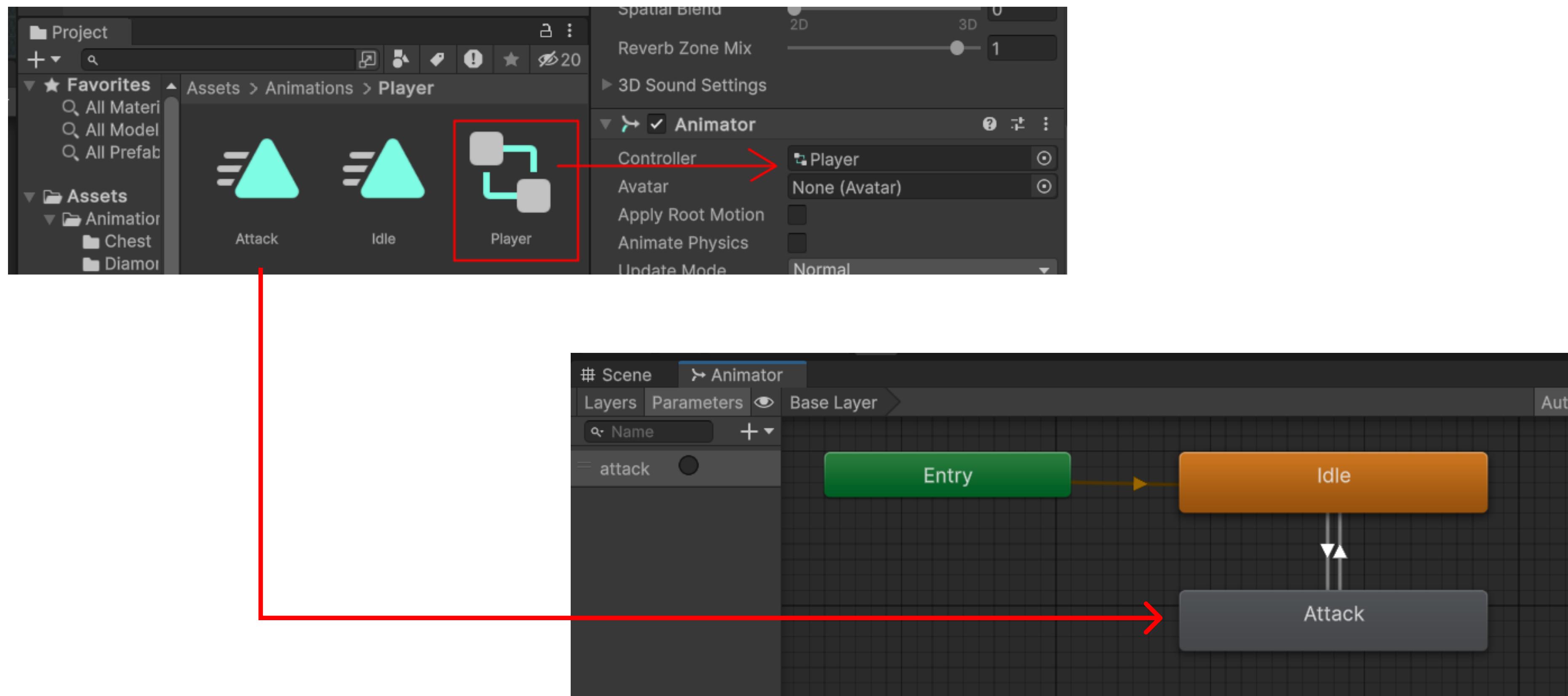
El audioClip se puede asignar por Unity o por código (si hay más de uno).

```
private AudioSource audioSourcePlayer;  
public AudioClip diamondAudioClip;  
  
audioSourcePlayer = GetComponent<AudioSource>();  
  
audioSourcePlayer.PlayOneShot(diamondAudioClip);
```



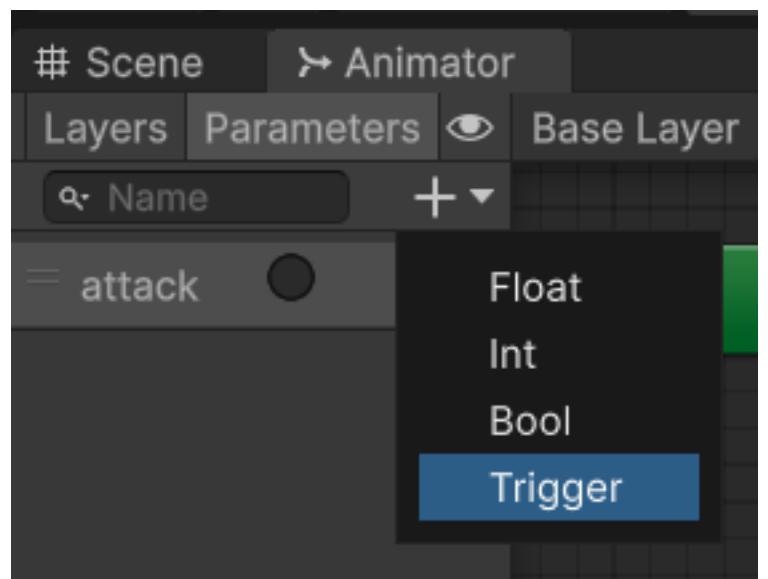
UF4 y 5: Desarrollo de juegos 2D y 3D

Animaciones: componente Animator

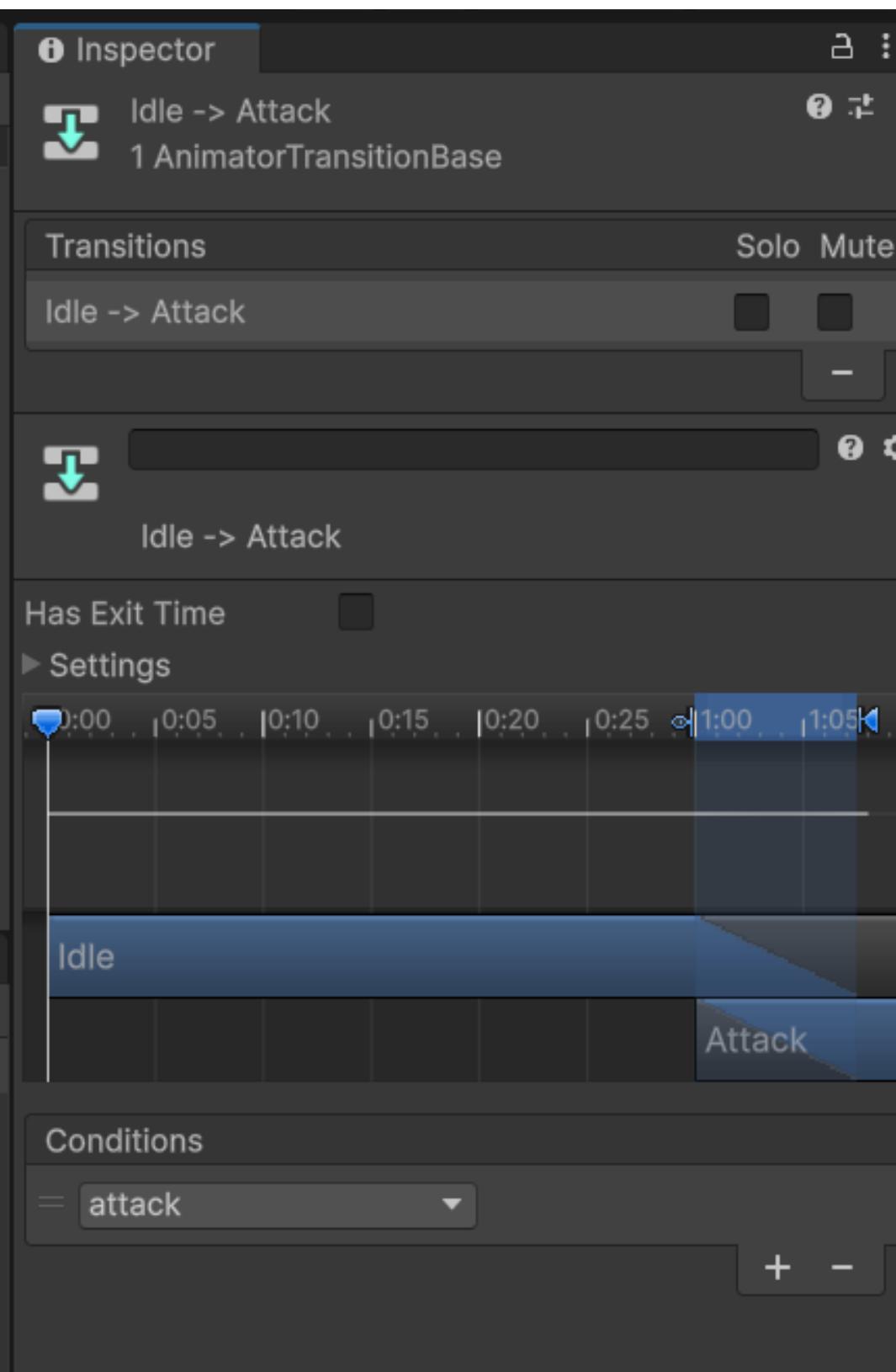


UF4 y 5: Desarrollo de juegos 2D y 3D

Animaciones: componente Animator



Entre animaciones se definen **transiciones** cuya ejecución puede estar condicionada a algún tipo de **parámetro**.



1. Declarar una variable del tipo de componente que va a guardar.
2. Inicializarla variable en el método start.
3. Usarla variable para activar/desactivar parámetros o animaciones.

```
private Animator animator;  
  
animator = GetComponent<Animator>();  
  
animator.SetTrigger("attack");
```

UF4 y 5: Desarrollo de juegos 2D y 3D

Prefabs: instanciar objetos dinámicamente

Prefab es un tipo de asset que permite almacenar un objeto *GameObject* con componentes y propiedades.

El *prefab* actúa como **una plantilla a partir de la cual se pueden crear nuevas instancias del objeto en la escena**. Cualquier edición hecha a un *prefab* será inmediatamente reflejada en todas las instancias que produzca, pero también se puede ajustar para cada instancia individualmente.

```
Instantiate(prefab, [position, rotation]);
```

UF4 y 5: Desarrollo de juegos 2D y 3D

Comunicación entre gameobjects

1. Declarar una variable del tipo correspondiente.
2. Buscar al gameobjecten el método start().
3. Ejecutar algún método de su script y/o usar sus componentes.

```
NombreScript otroObjetoScript;

otroObjetoScript = GameObject.Find("Name").GetComponent<NombreScript>();
otroObjetoScript = GameObject.FindGameObjectWithTag("tag").GetComponent<NombreScript>();

otroObjetoScript.Metodo();
otroObjetoScript.variable;
```

UF4 y 5: Desarrollo de juegos 2D y 3D

Corrutinas

Una corrutina es una función que tiene la habilidad de pausar su ejecución y devolver el control a Unity para luego continuar donde lo dejó en el siguiente frame o después de un tiempo dado.

```
void Start()
{
    StartCoroutine("CrearEnemigos");
}

IEnumerator CrearEnemigos()
{
    yield return new WaitForSeconds(2f);
    while(true)
    {
        // código de la corrutina
        yield return new WaitForSeconds(Random.Range(4f, 10f));
    }
}

public void CancelarEnemigos()
{
    StopCoroutine("CrearEnemigos");
}
```

unir LA UNIVERSIDAD
EN INTERNET