INTENTS, IMÁGENES Y WEBVIEW

Carlos Maldonado

5IV7

1. Introducción

El desarrollo de aplicaciones móviles ha evolucionado significativamente en la última década, ofreciendo múltiples plataformas y frameworks para crear experiencias de usuario ricas e interactivas. Dos enfoques destacados son el desarrollo nativo con Android Studio (Java/Kotlin) y el desarrollo multiplataforma con Kivy (Python). Este trabajo explora tres componentes fundamentales en el desarrollo de aplicaciones móviles: los Intents para comunicación entre componentes, el manejo de imágenes para contenido visual, y los WebViews para integración web. Se presentan implementaciones en ambas plataformas, destacando similitudes, diferencias y mejores prácticas.

2. Intents en Android Studio

2.1 Definición y Tipos de Intents

En Android, un Intent es un objeto de mensajería que permite solicitar una acción de otro componente de la aplicación o de otras aplicaciones. Funcionan como un mecanismo de comunicación entre diferentes componentes de la aplicación (Activities, Services, Broadcast Receivers).

Tipos principales de Intents:

1. **Intents Explícitos:** Especifican exactamente qué componente debe manejar el intent, generalmente dentro de la misma aplicación.
2. **Intents Implícitos:** No especifican el componente objetivo, sino la acción a realizar, permitiendo que el sistema determine el componente apropiado.

2.2 Implementación de Intents Explícitos

Los Intents explícitos se utilizan para iniciar componentes específicos dentro de la misma aplicación. Requieren el nombre de la clase del componente destino.

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

2.3 Implementación de Intents Implícitos

Los Intents implícitos son más flexibles y permiten la interacción con componentes de otras aplicaciones.

```
Intent browserIntent = new Intent(Intent.ACTION_VIEW,
                        Uri.parse("https://www.ejemplo.com"));
startActivity(browserIntent);
```

2.4 Ejemplo Práctico: Navegación entre Activities

MainActivity.java:

```
package com.ejemplo.intentsdemo;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends AppCompatActivity {

  private EditText editTextMensaje;
  private Button btnEnviar;

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

    editTextMensaje = findViewById(R.id.editTextMensaje);
    btnEnviar = findViewById(R.id.btnEnviar);

    btnEnviar.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        String mensaje = editTextMensaje.getText().toString();

        // Intent explícito para abrir SecondActivity
```

```
            Intent intent = new Intent(MainActivity.this,
                                SecondActivity.class);

            // Pasar datos al intent
            intent.putExtra("MENSAJE_KEY", mensaje);

            // Iniciar la actividad
            startActivity(intent);
        }
    });
  }
}
```

SecondActivity.java:

```
package com.ejemplo.intentsdemo;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class SecondActivity extends AppCompatActivity {

  private TextView textViewMensaje;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    textViewMensaje = findViewById(R.id.textViewMensaje);
    // Recibir datos del intent
    Intent intent = getIntent();
    if (intent != null && intent.hasExtra("MENSAJE_KEY")) {
      String mensajeRecibido = intent.getStringExtra("MENSAJE_KEY");
      textViewMensaje.setText("Mensaje recibido: " + mensajeRecibido);
    }
  }
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="16dp">

  <EditText
    android:id="@+id/editTextMensaje"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Escribe un mensaje"
    android:inputType="text"/>

  <Button
    android:id="@+id/btnEnviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:text="Enviar a Segunda Pantalla"
    android:layout_marginTop="16dp"/>

</LinearLayout>
```

3. Manejo de Imágenes en Android Studio

3.1 Métodos para Añadir Imágenes

En Android, existen varias formas de agregar y manejar imágenes:

1. **Recursos estáticos:** Imágenes almacenadas en las carpetas res/drawable
2. **Recursos desde la web:** Descarga dinámica de imágenes
3. **Desde almacenamiento local:** Galería del dispositivo
4. **Desde cámara:** Captura de fotos

3.2 Carpetas de Recursos (res/)

Android organiza los recursos en carpetas específicas según densidad de pantalla:

- drawable-ldpi (120 dpi)
- drawable-mdpi (160 dpi)
- drawable-hdpi (240 dpi)
- drawable-xhdpi (320 dpi)
- drawable-xxhdpi (480 dpi)
- drawable-xxxhdpi (640 dpi)

3.3 Carga Dinámica de Imágenes

Para cargar imágenes desde URL, se recomienda usar bibliotecas como Picasso o Glide:

```
dependencies {
  implementation 'com.github.bumptech.glide:glide:4.12.0'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
}
```

3.4 Ejemplo Práctico: Galería de Imágenes

activity_image_gallery.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Galería de Imágenes"
    android:textSize="24sp"
    android:layout_gravity="center"
    android:layout_margin="16dp"/>

  <androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerViewGaleria"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="8dp"/>

</LinearLayout>
```

ImageGalleryActivity.java:

```
package com.ejemplo.imagengallery;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.os.Bundle;
import java.util.ArrayList;
import java.util.List;

public class ImageGalleryActivity extends AppCompatActivity {

  private RecyclerView recyclerView;
  private ImageAdapter adapter;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_image_gallery);

    recyclerView = findViewById(R.id.recyclerViewGaleria);

    // Configurar RecyclerView con GridLayoutManager
    GridLayoutManager layoutManager = new GridLayoutManager(this, 2);
    recyclerView.setLayoutManager(layoutManager);

    // Crear lista de recursos de imagen
    List<Integer> imageResources = new ArrayList<>();
    imageResources.add(R.drawable.imagen1);
    imageResources.add(R.drawable.imagen2);
    imageResources.add(R.drawable.imagen3);
    imageResources.add(R.drawable.imagen4);

    // Configurar adaptador
    adapter = new ImageAdapter(this, imageResources);
    recyclerView.setAdapter(adapter);
  }

}
```

ImageAdapter.java:

```
package com.ejemplo.imagengallery;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;

public class ImageAdapter extends
RecyclerView.Adapter<ImageAdapter.ImageViewHolder> {

  private Context context;
  private List<Integer> imageResources;

  public ImageAdapter(Context context, List<Integer> imageResources) {
    this.context = context;
    this.imageResources = imageResources;
  }

  @NonNull
  @Override
  public ImageViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
    View view = LayoutInflater.from(context)
        .inflate(R.layout.item_image, parent, false);
    return new ImageViewHolder(view);
  }

  @Override
  public void onBindViewHolder(@NonNull ImageViewHolder holder, int position) {
    int imageResource = imageResources.get(position);
    holder.imageView.setImageResource(imageResource);
  }
```

```
    @Override
    public int getItemCount() {
        return imageResources.size();
    }

    public static class ImageViewHolder extends RecyclerView.ViewHolder {
        ImageView imageView;

        public ImageViewHolder(@NonNull View itemView) {
            super(itemView);
            imageView = itemView.findViewById(R.id.imageViewItem);
        }
    }
}
```

4. WebView en Android Studio

4.1 Concepto y Usos del WebView

WebView es un componente que muestra contenido web dentro de una aplicación. Puede cargar páginas web locales o remotas, y permite la interacción entre JavaScript y código nativo.

4.2 Configuración Básica

activity_webview.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>

</LinearLayout>
```

4.3 Personalización y Manejo de Eventos

WebViewActivity.java:

```
package com.ejemplo.webviewdemo;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.webkit.WebChromeClient;
import android.webkit.WebSettings;
import android.widget.Toast;

public class WebViewActivity extends AppCompatActivity {

  private WebView webView;

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

    webView = findViewById(R.id.webView);

    // Configurar WebView
    WebSettings webSettings = webView.getSettings();
    webSettings.setJavaScriptEnabled(true);
    webSettings.setDomStorageEnabled(true);
    webSettings.setLoadWithOverviewMode(true);
    webSettings.setUseWideViewPort(true);

    // Configurar WebViewClient para manejar navegación interna
    webView.setWebViewClient(new WebViewClient() {
      @Override
      public boolean shouldOverrideUrlLoading(WebView view, String url) {
        view.loadUrl(url);
        return true;
      }
```

```
        @Override
        public void onPageFinished(WebView view, String url) {
            super.onPageFinished(view, url);
            Toast.makeText(WebViewActivity.this,
                    "Página cargada: " + url,
                    Toast.LENGTH_SHORT).show();
        }
    });

    // Configurar WebChromeClient para manejar eventos JS
    webView.setWebChromeClient(new WebChromeClient() {
        @Override
        public void onProgressChanged(WebView view, int newProgress) {
            // Mostrar progreso de carga si es necesario
        }
    });

    // Cargar una URL
    webView.loadUrl("https://www.google.com");
  }

  @Override
  public void onBackPressed() {
    if (webView.canGoBack()) {
      webView.goBack();
    } else {
      super.onBackPressed();
    }
  }
}
```

4.4 Ejemplo Práctico: Navegador Web Básico

Para un navegador más completo, se puede añadir una barra de dirección y controles de navegación:

```
public class BrowserActivity extends AppCompatActivity {

  private WebView webView;
  private EditText editTextUrl;
  private Button btnGo, btnBack, btnForward;

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

    // Inicializar componentes
    webView = findViewById(R.id.webView);
    editTextUrl = findViewById(R.id.editTextUrl);
    btnGo = findViewById(R.id.btnGo);
    btnBack = findViewById(R.id.btnBack);
    btnForward = findViewById(R.id.btnForward);

    // Configurar WebView
    WebSettings settings = webView.getSettings();
    settings.setJavaScriptEnabled(true);

    // Configurar navegación
    btnGo.setOnClickListener(v -> {
      String url = editTextUrl.getText().toString();
      if (!url.startsWith("http://") && !url.startsWith("https://")) {
        url = "https://" + url;
      }
      webView.loadUrl(url);
    });

    btnBack.setOnClickListener(v -> {
      if (webView.canGoBack()) {
        webView.goBack();
```

```
        }
    });

    btnForward.setOnClickListener(v -> {
        if (webView.canGoForward()) {
            webView.goForward();
        }
    });
  }
}
```

5. Equivalente en Kivy (Python)

5.1 Introducción a Kivy

Kivy es un framework de código abierto escrito en Python para el desarrollo de aplicaciones multitouch. Es multiplataforma, funcionando en Android, iOS, Windows, Linux y macOS.

5.2 Manejo de Pantallas vs Intents

En Kivy, la navegación entre pantallas se maneja mediante el ScreenManager, que es conceptualmente similar a los Intents en Android.

Estructura básica en Kivy:

```
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.lang import Builder

# Definir pantallas
class MainScreen(Screen):
  pass

class SecondScreen(Screen):
  pass

# Configurar ScreenManager
class MyScreenManager(ScreenManager):
  pass

# Aplicación principal
class MyApp(App):
  def build(self):
    return MyScreenManager()
```

5.3 Manejo de Imágenes en Kivy

Kivy maneja imágenes mediante el widget Image, que puede cargar imágenes desde recursos locales o URLs.

5.4 WebView en Kivy

Para WebView en Kivy, se utiliza la biblioteca kivywebview o se integra con componentes nativos según la plataforma.

6. Ejemplos Comparativos

6.1 Ejemplo: Navegación entre Pantallas

Android (Java):

```
Intent intent = new Intent(MainActivity.this, SecondActivity.class);
startActivity(intent);
```

Kivy (Python):

```
# main.py
from kivy.app import App
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.lang import Builder

Builder.load_string('''
<MainScreen>:
  BoxLayout:
    orientation: 'vertical'
    padding: 50

    Label:
      text: 'Pantalla Principal'
      font_size: 24

    Button:
      text: 'Ir a Segunda Pantalla'
      on_press: root.manager.current = 'second'
      size_hint_y: None
      height: 50
```

```
<SecondScreen>:
    BoxLayout:
        orientation: 'vertical'
        padding: 50

        Label:
            text: 'Segunda Pantalla'
            font_size: 24

        Button:
            text: 'Regresar'
            on_press: root.manager.current = 'main'
            size_hint_y: None
            height: 50
''')

class MainScreen(Screen):
    pass

class SecondScreen(Screen):
    pass

class ScreenManagerApp(App):
    def build(self):
        sm = ScreenManager()
        sm.add_widget(MainScreen(name='main'))
        sm.add_widget(SecondScreen(name='second'))
        return sm

if __name__ == '__main__':
    ScreenManagerApp().run()
```

6.2 Ejemplo: Carga y Visualización de Imágenes

Android (XML + Java):

```
<!-- En layout XML -->
<ImageView
  android:id="@+id/imageView"
  android:layout_width="200dp"
  android:layout_height="200dp"
  android:scaleType="centerCrop"
  android:src="@drawable/mi_imagen"/>
```

Kivy (Python + KV Language):

```
# main.py
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.image import Image

class ImageApp(App):
  def build(self):
    layout = BoxLayout(orientation='vertical', padding=20)

    # Cargar imagen desde archivo local
    image = Image(source='imagen.jpg',
            size_hint=(None, None),
            size=(400, 400))

    layout.add_widget(image)
    return layout

if __name__ == '__main__':
  ImageApp().run()
```

6.3 Ejemplo: Integración Web

Android WebView:

java
```
WebView webView = findViewById(R.id.webView);
webView.getSettings().setJavaScriptEnabled(true);
webView.loadUrl("https://www.ejemplo.com");
```
Kivy con WebView (usando android.webkit.WebView en Android):
```
# Para WebView en Kivy en plataforma Android
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from android.runnable import run_on_ui_thread
from jnius import autoclass

WebView = autoclass('android.webkit.WebView')
WebViewClient = autoclass('android.webkit.WebViewClient')
activity = autoclass('org.kivy.android.PythonActivity').mActivity

class AndroidWebView(BoxLayout):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.create_webview()

    @run_on_ui_thread
    def create_webview(self):
        webview = WebView(activity)
        webview.getSettings().setJavaScriptEnabled(True)
        webview.setWebViewClient(WebViewClient())
        webview.loadUrl('https://www.ejemplo.com')

        # Convertir a widget Kivy
        from kivy.uix.widget import Widget
        from kivy.core.window import Window
        from kivy.graphics import Fbo, Color, Rectangle

        self.webview = webview
        self.canvas = Canvas()
```

```
        with self.canvas:
            self.fbo = Fbo(size=Window.size)
            Color(1, 1, 1)
            Rectangle(texture=self.fbo.texture, size=self.size)

    class WebViewApp(App):
        def build(self):
            return AndroidWebView()

    if __name__ == '__main__':
        WebViewApp().run()
```

7. Análisis Comparativo

CARACTERÍSTICA	ANDROID STUDIO (JAVA/KOTLIN)	KIVY (PYTHON)
NAVEGACIÓN	Intents (explícitos/implícitos)	ScreenManager
MANEJO DE IMÁGENES	ImageView, Glide/Picasso	Widget Image
WEBVIEW	WebView nativo (android.webkit)	Integración con componentes nativos o kivywebview
RENDIMIENTO	Alto (nativo)	Variable (depende de la plataforma)
DESARROLLO MULTIPLATAFORMA	Solo Android	Android, iOS, Windows, Linux, macOS
CURVA DE APRENDIZAJE	Moderada a alta	Moderada
COMUNIDAD Y DOCUMENTACIÓN	Extensa	Creciente pero más limitada

Ventajas de Android Studio:

- Rendimiento nativo óptimo
- Acceso completo a APIs del dispositivo
- Amplia documentación y comunidad
- Herramientas de desarrollo maduras

Ventajas de Kivy:

- Desarrollo multiplataforma con un solo código base
- Lenguaje Python (fácil de aprender)
- Rápido prototipado
- Licencia MIT (open source)

8. Conclusión

El desarrollo de aplicaciones móviles presenta múltiples enfoques, cada uno con sus ventajas y desventajas. Android Studio ofrece un entorno de desarrollo nativo robusto con componentes bien definidos como Intents, ImageView y WebView, proporcionando un control granular sobre el comportamiento de la aplicación y acceso directo a las capacidades del dispositivo.

Por otro lado, Kivy representa una alternativa interesante para desarrolladores que buscan una solución multiplataforma utilizando Python, un lenguaje accesible y poderoso. Aunque algunos componentes como WebView requieren integración con componentes nativos, Kivy ofrece un framework coherente para la mayoría de las necesidades de desarrollo de UI.

La elección entre Android Studio y Kivy depende de factores como los requisitos de rendimiento, el alcance multiplataforma, la experiencia del equipo de desarrollo y los recursos disponibles. Ambas plataformas continúan evolucionando, ofreciendo herramientas cada vez más sofisticadas para crear experiencias de usuario ricas e interactivas.

Bibliografía

1. **Android Developers — Intents y filtros de intents (Guía oficial)** — Explica qué es un Intent, tipos (explícito/implícito), cómo iniciar Activities y pasar datos; esencial para la definición y ejemplos de código en Android (Kotlin/Java). Android Developers
2. **Android Developers — Codelab "Intents y actividades" (tutorial paso a paso)** — Ejemplo práctico para implementar intents entre Activities (ideal para incluir un ejemplo funcional en el trabajo). Android Developers
3. **Android Developers — Cómo agregar imágenes a tu app (codelab / ImageView)** — Guía oficial y práctica sobre añadir ImageView, recursos drawable, y buenas prácticas (útil para la sección de imágenes en Android). Android Developers+1
4. **PabloMonteserin — Curso Android: Insertar imágenes en un proyecto** — Tutorial paso a paso (español) con ejemplos de XML y código para cargar imágenes locales y desde código (útil para capturas y variantes). Pablo Monteserín
5. **Android Developers — WebView (cómo crear apps web en WebView)** — Documentación oficial sobre WebView, carga de URL, seguridad y gestión del componente; imprescindible para la parte de WebView. Android Developers
6. **Android Developers — Cómo administrar objetos WebView (gestión y seguridad)** — Buen recurso para explicar limitaciones, manejo de navegación y recomendaciones de seguridad al usar WebView en apps reales. (fecha actualizada; útil para justificar cuándo usar WebView). Android Developers
7. **MoureDev / GitHub — Tutorial "Crear un WebView con Kotlin" (repo/tutorial en español)** — Tutorial práctico con código completo (Kotlin) para implementar un navegador simple en tu app; ideal para incluir como ejemplo paso a paso. GitHub
8. **DaniSanchez.net — Crear un WebView con Android Studio (tutorial práctico en español)** — Guía clara con ejemplo de proyecto, ajustes de WebView y consejos de compatibilidad; útil para capturas y código comentado. danisanchez.net
9. **Develou — Comunicar Actividades a través de Intents (tutorial en español)** — Explicación con ejemplos y casos prácticos (inicio de Activities, pasar datos); buena fuente para ejercicios y ejemplos adicionales. develou.com

10. **Stack Overflow en español — preguntas y soluciones (ImageView / WebView ejemplos concretos)** — Hilos con soluciones a problemas frecuentes (asignar imágenes por código, navegación en WebView). Útil para incluir "FAQ / errores comunes" en tu trabajo. Stack Overflow en español+1
11. **Kivy — Documentación / Ejemplos (Basic Picture Viewer)** — Ejemplo oficial de Kivy que muestra cómo cargar y mostrar imágenes (útil para la sección "equivalente en Python (Kivy)" sobre imágenes). kivy.org
12. **Kivy-Webview — Repositorio / ejemplo (GitHub: Kivy-Webview)** — Implementación comunitaria (repositorio) que muestra cómo integrar un componente WebView en Kivy (útil como ejemplo práctico de WebView en Kivy). GitHub
13. **python-for-android / documentación (bootstrap webview)** — Información sobre cómo crear apps que arranquen con una vista web (webview bootstrap) usando python-for-android y Flask — útil para la alternativa a WebView nativa en Kivy/packaged Python. Scribd
14. **Foros y StackOverflow — Abrir URL / navegar desde Kivy (soluciones prácticas)** — Hilos que muestran cómo abrir URLs en el navegador nativo desde Kivy (webbrowser, plyer, o usar pyjnius para invocar Intents en Android). Muy útil para demostrar "equivalente a Intent" en Python/Kivy. Stack Overflow+1
15. **PyJnius / tutoriales (introducción en español)** — Introducción al uso de pyjnius para invocar APIs Java/Android desde Python; imprescindible para mostrar cómo en Kivy puedes llamar Intents nativos en Android. YouTube