TALLER 1 – ANDROID STUDIOS

Augusto Perrone

Programación dirigida por eventos

Contenido

Introducción

En este proyecto, se ha desarrollado una aplicación Android simple denominada *EventDrivenApp*, que implementa la Programación Dirigida por Eventos y una interfaz de usuario interactiva. El objetivo principal es permitir al usuario ingresar su nombre y recibir un saludo personalizado, además de proporcionar una experiencia visual dinámica al cambiar el color de fondo de la pantalla al presionar un botón. La aplicación se ha desarrollado utilizando Android Studio y el lenguaje de programación Kotlin.

Los objetivos específicos incluyen:

- Comprender el entorno de desarrollo Android Studio y sus componentes.
- Diseñar una interfaz de usuario interactiva.
- Implementar el manejo de eventos.
- Realizar pruebas automatizadas para asegurar la funcionalidad.

Esta guía describe los aspectos técnicos del proyecto, como el diseño de la interfaz, la implementación de eventos y el uso de herramientas de pruebas automatizadas.

Diseño de la Interfaz de usuario

La interfaz de la aplicación se diseñó para ser simple y funcional, con elementos que permiten al usuario interactuar fácilmente. El diseño se realizó utilizando XML, un lenguaje que facilita la creación de interfaces gráficas en aplicaciones Android.

Elementos de la interfaz:

- **Campo de texto (EditText):** Permite al usuario ingresar su nombre.
- **Texto de saludo (TextView):** Muestra un saludo personalizado que cambia dinámicamente en función de la entrada del usuario.
- **Botón (Button):** Al ser presionado, cambia el color de fondo de la aplicación.

Código XML de la interfaz

El siguiente código muestra cómo se estructuró la interfaz en el archivo activity_main.xml:

xml

Copiar código

```
<EditText
    android:id="@+id/editTextName"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="Ingrese su nombre"
    android:padding="16dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_margin="16dp"/>

<TextView
    android:id="@+id/textViewGreeting"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="¡Hola, Mundo!"
    android:textSize="20sp"
    app:layout_constraintTop_toBottomOf="@id/editTextName"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"/>

<Button
    android:id="@+id/buttonChangeColor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Cambiar Color"
    android:padding="16dp"
```

```
app:layout_constraintTop_toBottomOf="@id/textViewGreeting"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintEnd_toEndOf="parent"
android:layout_marginTop="16dp"/>
```

Implementación de eventos

La Programación Dirigida por Eventos es uno de los pilares de este proyecto. Se han implementado varios *listeners* que responden a las acciones del usuario, mejorando la interacción con la aplicación.

1. **Evento de cambio de texto:**
   Utilizamos un TextWatcher para detectar cuando el usuario ingresa texto en el campo de nombre. Dependiendo de si el campo está vacío o no, el saludo se actualiza dinámicamente.

```
binding.editTextName.addTextChangedListener(object : TextWatcher {
override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int, after: Int) {}

override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count: Int) {
    val name = s.toString()
    if (name.isNotEmpty()) {
        binding.textViewGreeting.text = "¡Hola, $name!"
    } else {
        binding.textViewGreeting.text = "¡Hola, Mundo!"
    }
}

override fun afterTextChanged(s: Editable?) {}
})
```

2. **Evento de cambio de color:**
   El botón cambia el color de fondo de la pantalla cada vez que es presionado, generando un color aleatorio mediante la función Color.rgb().

```
binding.buttonChangeColor.setOnClickListener {
  val randomColor = generateRandomColor()
  binding.root.setBackgroundColor(randomColor)
}

private fun generateRandomColor(): Int {
  val random = java.util.Random()
  return Color.rgb(random.nextInt(256), random.nextInt(256),
random.nextInt(256))
}
```

Pruebas automatizadas

Para asegurar que la aplicación funcione correctamente, se realizaron pruebas automatizadas utilizando **JUnit** para pruebas unitarias

Se implementaron pruebas para verificar que el saludo personalizado funcione correctamente.

```
@Test
fun testGreetingMessage() {
  val name = "Juan"
  val expectedGreeting = "¡Hola, $name!"
  assertEquals("¡Hola, Juan!", expectedGreeting)
}
```

Conclusión

Este proyecto de aplicación Android se centra en la implementación de eventos y en mejorar la experiencia del usuario mediante una interfaz simple pero

interactiva. Utilizando Android Studio y Kotlin, se ha logrado crear una aplicación funcional que responde de manera eficiente a las acciones del usuario.

Además, la integración de pruebas automatizadas ha sido fundamental para garantizar la calidad y estabilidad de la aplicación. Este enfoque no solo facilita la depuración, sino que asegura que la experiencia del usuario final sea fluida y libre de errores.

Con este proyecto, hemos reforzado nuestras habilidades en el manejo de eventos, el diseño de interfaces interactivas y la implementación de pruebas automatizadas, todos componentes esenciales en el desarrollo de aplicaciones Android modern