Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

MANUAL DE PRÁCTICAS DE LABORATORIO

Desarrollo de Aplicaciones Móviles

Laboratorio

Programa Académico	**INGENIERÍA EN SOFTWARE**
Plan de Estudios	**2021**
Fecha de elaboración	**27/05/ 2025**
Versión del Documento	**1**

Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

Dra. Martha Patricia Patiño Fierro
Rectora

Mtra. Ana Lisette Valenzuela Molina
Encargada del Despacho de la Secretaría General Académica

Mtro. José Antonio Romero Montaño
Secretario General Administrativo

Lic. Jorge Omar Herrera Gutiérrez
Encargado de Despacho de Secretario General de Planeación

Tabla de contenido

INTRODUCCIÓN

Como parte de las herramientas esenciales para la formación académica de los estudiantes de la Universidad Estatal de Sonora, se definen manuales de práctica de laboratorio como elemento en el cual se define la estructura normativa de cada práctica y/o laboratorio, además de representar una guía para la aplicación práctica del conocimiento y el desarrollo de las competencias clave en su área de estudio. Su diseño se encuentra alineado con el modelo educativo institucional, el cual privilegia el aprendizaje basado en competencias, el aprendizaje activo y la conexión con escenarios reales.

Con el propósito de fortalecer la autonomía de los estudiantes, su pensamiento crítico y sus habilidades para la resolución de problemas, las prácticas de laboratorio integran estrategias didácticas como el aprendizaje basado en proyectos, el trabajo colaborativo, la experimentación guiada y el uso de tecnologías educativas. De esta manera, se promueve un proceso de enseñanza-aprendizaje dinámico, en el que los estudiantes no solo adquieren conocimientos teóricos, sino que también desarrollan habilidades prácticas y reflexivas para su desempeño profesional.

Señalar en este apartado brevemente los siguientes elementos según corresponda:

- Propósito del manual
- Justificación de su uso en el programa académico
- Competencias a desarrollar
    - **Competencias blandas:** Habilidades transversales que se refuerzan en las prácticas, como la comunicación, el trabajo en equipo, el uso de tecnologías, etc.
    - **Competencias disciplinares:** Conocimientos específicos del área del laboratorio, incluyendo fundamentos teóricos y habilidades técnicas.
    - **Competencias profesionales:** Aplicación de los conocimientos adquiridos en escenarios reales o simulados, en concordancia con el perfil de egreso del programa.

IDENTIFICACIÓN

Nombre de la Asignatura		Desarrollo de Aplicaciones Móviles	
Clave	061CE011	Créditos	6
Asignaturas Antecedentes		Plan de Estudios	2021

Área de Competencia	Competencia del curso
Específicas o Especializantes	Aplicar las nuevas tecnologías de computación móvil para planificar, desarrollar e implementar soluciones en las organizaciones, bajo los estándares de calidad establecidos, de una manera innovadora y creativa.

Carga Horaria de la asignatura

Horas Supervisadas			Horas Independientes	Total de Horas
Aula	Laboratorio	Plataforma		
2	1	1	2	6

Consignación del Documento

Unidad Académica	Unidad Académica Magdalena
Fecha de elaboración	27/05/2025
Responsables del diseño	José Jesús Miranda Mirasol
Validación	
Recepción	Coordinación de Procesos Educativos

MATRIZ DE CORRESPONDENCIA

Señalar la relación de cada práctica con las competencias del perfil de egreso

PRÁCTICA	PERFIL DE EGRESO
Práctica No. 1: Instalación entorno de desarrollo móvil nativo.	– Desarrollar software con la finalidad de agilizar los procesos y la toma de decisiones en empresas públicas y privadas, bajo estándares de calidad nacional e internacional con enfoque de liderazgo
Práctica No. 2: Programación de aplicaciones Móviles, Intent Explicito e Implícito	– Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información en los departamentos que así lo requieran, poniendo en práctica sus habilidades de trabajo en equipo y planeación.
Práctica No. 3: Diseño de Interface en App	– Aplicar soluciones e innovaciones tecnológicas con la finalidad de automatizar los procesos, atendiendo los principios de la organización y gestión efectiva de la información en los departamentos que así lo requieran, poniendo en práctica sus habilidades de trabajo en equipo y planeación.
Práctica No. 4: Práctica de App C.R.U. D	– Crear bases de datos para una gestión eficiente de la información, garantizando la integridad y seguridad de los datos, atendiendo los requerimientos de la organización con un sentido de liderazgo e innovación.

NORMAS DE SEGURIDAD Y BUENAS PRÁCTICAS

Reglamento general del laboratorio

1. Objetivo

Establecer las normas de uso, seguridad y mantenimiento del centro de cómputo para garantizar su correcto funcionamiento, la integridad de los equipos y la productividad de los usuarios.

2. Ámbito de Aplicación

Aplica a todos los usuarios (estudiantes, docentes, personal administrativo y visitantes) que utilicen los recursos del centro de cómputo.

3. Horario de Operación

Lunes a viernes: 8:00 AM – 9:00 PM

4. Normas de Acceso y Uso

✓ Permitido:

Uso académico, investigativo o laboral.

Conexión de dispositivos personales (laptops, tablets) previa autorización.

Impresión de documentos académicos (con moderación).

✗ Prohibido:

Ingresar con alimentos o bebidas.

Instalar software sin autorización.

Jugar videojuegos o acceder a contenido inapropiado.

Modificar la configuración de hardware/software.

Fumar o usar dispositivos electrónicos no autorizados.

5. Seguridad y Mantenimiento

Equipos:

Apagar correctamente las computadoras después de su uso.

Reportar fallas técnicas al personal encargado.

Redes y datos:

No descargar archivos ilegales o maliciosos.

Respetar la política de privacidad de otros usuarios.

Emergencias:

En caso de incendio o falla eléctrica, seguir las indicaciones del personal.

6. Sanciones
Primera falta: Amonestación verbal.

Segunda falta: Suspensión temporal del acceso.

Tercera falta: Reporte a autoridades académicas o administrativas.

Daño intencional a equipos: Reposición económica y suspensión de acceso.

7. Responsabilidades del Personal
Supervisar el cumplimiento del reglamento.

Brindar soporte técnico básico.

Realizar mantenimiento preventivo mensual.

8. Disposiciones Finales
El incumplimiento del reglamento acarrea consecuencias disciplinarias.

Cualquier situación no prevista será resuelta por el administrador del centro.

Reglamento de uniforme

NA

Uso adecuado del equipo y materiales

Apagar correctamente las computadoras después de su uso.

Reportar fallas técnicas al personal encargado

Manejo y disposición de residuos peligrosos

NA

Procedimientos en caso de emergencia

En caso de incendio o falla eléctrica, seguir las indicaciones del personal.

RELACIÓN DE PRÁCTICAS DE LABORATORIO POR ELEMENTO DE COMPETENCIA

Elemento de Competencia al que pertenece la práctica	II
	Diseñar aplicaciones móviles, utilizando técnicas de modelado y programación en un entorno de desarrollo móvil para analizar y proponer soluciones a las organizaciones.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 1	Instalación entorno de desarrollo móvil nativo.	Instalar y configurar Android Studio para reconocer los componentes esenciales del IDE en el desarrollo de aplicaciones móviles de una forma correcta y profesional.
Práctica No. 2	Programación de aplicaciones Móviles, Intent Explicito e Implícito	Desarrollar aplicaciones básicas en Kotlin utilizando Intent explícito e implícito en Android Studio para interactuar con aplicaciones del sistema (como llamadas, navegación web o envío de correos) de una forma responsable y profesional.
Práctica No. 3	Diseño de Interface en App	Diseñar una interfaz gráfica en Android Studio utilizando Kotlin, aplicando controles básicos, con el fin de implementar una calculadora funcional que realice operaciones aritméticas básicas, bajo los criterios de usabilidad y diseño de Material Design, desarrollando habilidades de pensamiento lógico y trabajo en equipo.

Elemento de Competencia al que pertenece la práctica	III
	Codificar aplicaciones móviles para integrar soluciones al proceso del negocio, a fin de mejorarlo y/o optimizarlo bajo estándares de calidad con una resolución de problemas orientados al desarrollo creativo de soluciones.

PRÁCTICA	NOMBRE	COMPETENCIA
Práctica No. 4	App C.R.U. D.	Redactar competencia a desarrollar en la práctica de acuerdo con los criterios para la redacción de competencias: Verbo + objeto + finalidad + condición + contexto + competencia blanda

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

PRÁCTICAS

NOMBRE DE LA PRÁCTICA	Instalación entorno de desarrollo móvil nativo
COMPETENCIA DE LA PRÁCTICA	Instalar y configurar Android Studio para reconocer los componentes esenciales del IDE en el desarrollo de aplicaciones móviles de una forma correcta y profesional.

FUNDAMENTO TÉORICO

Introducción a Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones Android, desarrollado por Google y JetBrains. Está basado en IntelliJ IDEA, un IDE reconocido por su potente asistencia de código, refactorización y herramientas de productividad.

Android Studio proporciona un conjunto de herramientas esenciales para el desarrollo, depuración, prueba y publicación de aplicaciones móviles en la plataforma Android. Su integración con el SDK de Android y el sistema de compilación Gradle lo convierten en la opción más eficiente para desarrolladores.

Componentes Principales de Android Studio

a) Android SDK (Software Development Kit)
- Conjunto de herramientas que incluye librerías, depuradores, emuladores y APIs necesarias para desarrollar aplicaciones Android.
- Permite seleccionar versiones específicas de Android para garantizar compatibilidad con diferentes dispositivos.

b) Emulador de Android (AVD - Android Virtual Device)
- Simulador de dispositivos Android que permite probar aplicaciones sin necesidad de hardware físico.
- Configurable con diferentes versiones de Android, tamaños de pantalla y capacidades de hardware.

c) Gradle
- Sistema de construcción automatizado que gestiona dependencias, compilación y empaquetado de la aplicación.
- Permite definir configuraciones específicas para diferentes versiones (debug, release, etc.).

d) Layout Editor (Diseño Visual)
- Herramienta de diseño drag-and-drop para crear interfaces de usuario sin necesidad de escribir XML manualmente.
- Soporte para ConstraintLayout, Material Design y vistas responsivas.

e) Android Jetpack
- Conjunto de bibliotecas y herramientas que facilitan el desarrollo siguiendo las mejores prácticas.
- Incluye componentes como Room (base de datos), ViewModel (gestión de datos) y LiveData (observación de datos).

Requisitos del Sistema

Para un funcionamiento óptimo, Android Studio requiere:

- Sistema operativo: Windows 10/11, macOS 10.14+, o Linux (Ubuntu 18.04+).
- RAM: Mínimo 8 GB (recomendado 16 GB).
- Almacenamiento: 10 GB libres (más espacio para emuladores y SDK).
- Java Development Kit (JDK): Versión 11 o superior (incluido en Android Studio, pero puede requerir configuración manual).

Flujo de Trabajo en Android Studio

1. Creación de un proyecto: Selección de plantillas (Empty Activity).
2. Diseño de la interfaz: Uso del Layout Editor o edición manual de XML.
3. Programación en Kotlin: Implementación de la lógica de la aplicación.
4. Ejecución y depuración: Pruebas en emulador o dispositivo real.
5. Generación del APK/AAB: Compilación para distribución en Google Play.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Material/Equipo	Cantidad	Especificaciones
Computadora	1	Windows/macOS/Linux, mínimo 8GB RAM, 10GB espacio libre
Conexión a Internet	1	Banda ancha estable
Android Studio	1	Última versión estable descargable desde developer.android.com
JDK (Java Development Kit)	1	Versión 11 o superior

PROCEDIMIENTO O METODOLOGÍA

Descripción de Actividades

1. **Descarga e Instalación de Android Studio**

   1. Acceder al sitio oficial de Android Studio.
   2. Descargar la versión compatible con el sistema operativo.
   3. Ejecutar el instalador y seguir las instrucciones (asegurarse de incluir el SDK y el emulador).

2. **Configuración Inicial**

1. Abrir Android Studio y completar el asistente de configuración.
2. Instalar los paquetes necesarios del SDK.
3. Configurar un dispositivo virtual (AVD) para pruebas.

3. **Verificación del Entorno**

1. Crear un proyecto nuevo con la plantilla "Empty Activity".
2. Ejecutar la aplicación en el emulador o en un dispositivo físico conectado.

Precauciones y Advertencias

1. Verificar que el equipo cumpla con los requisitos mínimos.
2. Tener permisos de administrador para la instalación.
3. Evitar interrumpir la descarga o instalación para prevenir errores.

RESULTADOS ESPERADOS

1. Android Studio instalado correctamente sin errores.
2. Proyecto base compilado y ejecutado en el emulador.
3. Interfaz gráfica del IDE funcional y reconocimiento de herramientas principales.

ANÁLISIS DE RESULTADOS

¿Se completó la instalación sin errores?

¿El emulador funciona correctamente?

¿Qué problemas surgieron y cómo se resolvieron?

CONCLUSIONES Y REFLEXIONES

La correcta instalación de Android Studio es fundamental para el desarrollo de aplicaciones móviles. Esta práctica permitió familiarizarse con el entorno y verificar su funcionamiento, lo que es esencial para proyectos futuros en el ámbito profesional.

ACTIVIDADES COMPLEMENTARIAS
1. Investigar cómo instalar Android Studio en Linux. 2. Probar la ejecución de una aplicación en un dispositivo físico. 3. Explorar las herramientas de depuración y profiling de Android Studio.

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Desarrollo de la práctica 70%, reporte 30%
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica sobre Práctica de Laboratorio
Formatos de reporte de prácticas	Ejemplo de reporte en Anexos

FUENTES DE INFORMACIÓN

Google. (2023). Android Studio documentation. Android Developers. https://developer.android.com/studio/intro?hl=es-419

Google. (2023). Android Jetpack. Android Developers. https://developer.android.com/jetpack

JetBrains. (2023). IntelliJ IDEA: The Capable & Ergonomic Java IDE. https://www.jetbrains.com/idea/

Oracle. (2023). Java Development Kit (JDK). https://www.oracle.com/java/technologies/javase-downloads.html

Gradle. (2023). Gradle Build Tool. https://gradle.org/

NORMAS TÉCNICAS APLICABLES

NOM-019-STPS-2011 (Seguridad en Laboratorios de Computación)

NOM-163-SCFI-2022 (Especificaciones de Equipo de Cómputo)

NOMBRE DE LA PRÁCTICA	**Programación de aplicaciones Móviles, Intent Explicito e Implícito**
COMPETENCIA DE LA PRÁCTICA	Desarrollar aplicaciones básicas en Kotlin utilizando Intent explícito e implícito en Android Studio para interactuar con aplicaciones del sistema (como llamadas, navegación web o envío de correos) de una forma responsable y profesional.

FUNDAMENTO TÉORICO

Definición y Propósito de los Intents

En Android, un Intent es un mecanismo fundamental que permite la comunicación entre componentes de una aplicación (como Activities, Services y Broadcast Receivers) o incluso entre diferentes aplicaciones. Funciona como un mensaje que solicita una acción, ya sea iniciar una nueva pantalla, enviar datos, abrir una URL o interactuar con funciones del sistema.

Características principales:
- Flexibilidad: Pueden usarse para activar componentes internos (explícitos) o delegar acciones a otras apps (implícitos).
- Interoperabilidad: Facilitan la integración con aplicaciones de terceros (ejemplo: compartir en redes sociales, abrir mapas).
- Seguridad: Los permisos definidos en el `AndroidManifest.xml` controlan qué componentes pueden responder a un Intent.

Tipos de Intents:

1. Intent Explícito
- Definición: Especifica directamente el componente destino (usando su clase).
- Uso típico: Navegación entre Activities dentro de la misma app.
- Ejemplo en Kotlin:

```
val intent = Intent(this, SecondActivity::class.java)
startActivity(intent)
```

- Ventajas:
  - Mayor control y seguridad (no depende de apps externas).
  - Rendimiento optimizado al evitar búsquedas en el sistema.

2. Intent Implícito
- Definición: Describe una acción genérica (como "ver una página web") sin especificar el componente. El sistema busca apps compatibles.
- Uso típico:
  - Abrir enlaces (`ACTION_VIEW`).
  - Enviar correos (`ACTION_SEND`).
  - Realizar llamadas (`ACTION_DIAL`).

- Ejemplo en Kotlin:

```
val intent = Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.com"))
startActivity(intent)
```

- Ventajas:
  - Reutilización de funcionalidades existentes en el dispositivo.
  - Integración con el ecosistema Android (ej: Google Maps, WhatsApp).

La figura 1 muestra cómo se usa una intent para iniciar una actividad. Cuando el objeto Intent nombra un componente de actividad específico de forma explícita, el sistema inicia ese componente de inmediato.

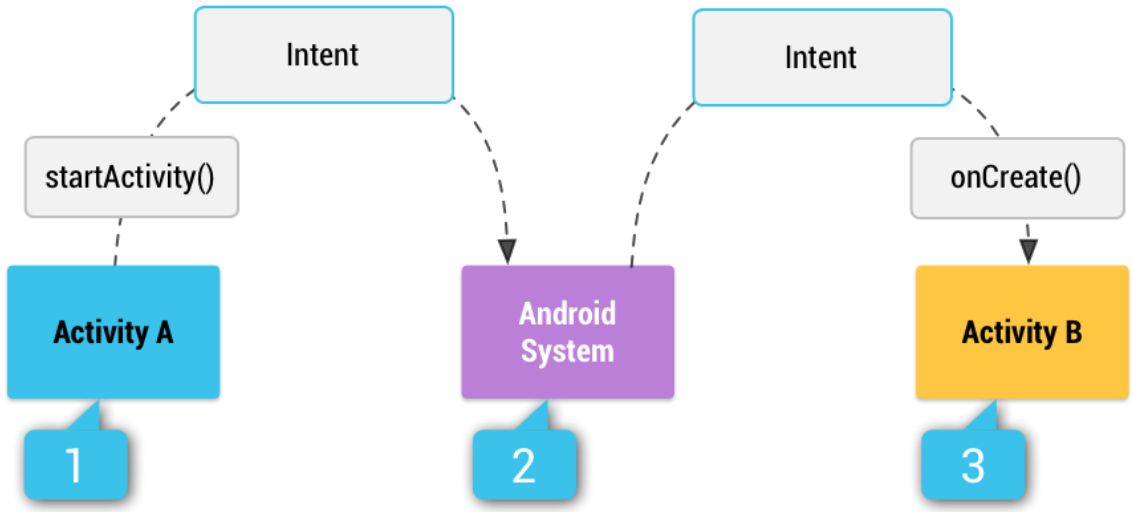


Figura 1: Cómo se entrega un intent implícito a través del sistema para iniciar otra actividad: **[1]** *La actividad A* crea un Intent con una descripción de acción y lo pasa a startActivity(). **[2]** El sistema Android busca en todas las apps un filtro de intents que coincida con el intent. Cuando se encuentra una coincidencia, **[3]** el sistema inicia la actividad coincidente (*Actividad B*) invocando su método onCreate() y pasándole el Intent.

Componentes Clave en un Intent

Elemento	Descripción	Ejemplo
Action	Define la acción a realizar (constantes como ACTION_VIEW, ACTION_SEND).	Intent.ACTION_VIEW
Data	URI que especifica los datos sobre los que actuar (ej: una URL, teléfono).	Uri.parse("tel:123456789")
Category	Proporciona contexto adicional (ej: CATEGORY_BROWSABLE para web).	intent.addCategory(Intent.CATEGORY_DEFAULT)
Extras	Datos adicionales en pares clave-valor (usando putExtra()).	intent.putExtra("key", "value")

Filtros de Intents (Intent Filters)

Para que un componente (como una Activity) pueda responder a Intents implícitos, debe declarar un <intent-filter> en el AndroidManifest.xml

Ejemplo: Activity que maneja enlaces web

```
<activity android:name=".WebActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="https" />
    </intent-filter>
</activity>
```

Importante:
- Si múltiples apps pueden manejar el mismo Intent, Android muestra un diálogo para que el usuario elija ("Selector de Intents").
- Para evitar errores, siempre verificar si existe al menos una app compatible:

```
if (intent.resolveActivity(packageManager) != null) {
    startActivity(intent)
} else {
    Toast.makeText(this, "No hay apps compatibles", Toast.LENGTH_SHORT).show()
}
```

Diferencias Clave entre Intents Explícitos e Implícitos

Criterio	Intent Explícito	Intent Implícito
Destino	Componente específico (ej. SecondActivity)	Acción genérica (ej. “abrir mapa”).
Uso	Navegación interna en la app	Integración con apps de terceros
Declaración	No requiere <*intent-filter*>	Necesita <*intent-filter*> en el destino
Rendimiento	Más rápido (sin búsqueda en el sistema)	Más lento (consulta apps disponibles).

Aplicaciones Prácticas en el Desarrollo Profesional
- Redes sociales: Compartir contenido desde tu app hacia Facebook o X.
- Pagos: Integración con PayPal o Google Pay mediante Intents implícitos.
- Geolocalización: Abrir Google Maps con una dirección específica.
- Personalización: Crear apps que respondan a Intents customizados (ejemplo: "com.example.OPEN_CAMERA").

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Material/Equipo	Cantidad	Especificaciones
Computadora	1	Con Android Studio instalado
Dispositivo Android o Emulador	1	API nivel 21 o superior
Cable USB (opcional)	1	Para conexión física del dispositivo

Software Requerido

- Android Studio (versión estable más reciente).
- SDK de Android con API nivel 21+.
- Emulador configurado (si no se usa dispositivo físico).

PROCEDIMIENTO O METODOLOGÍA

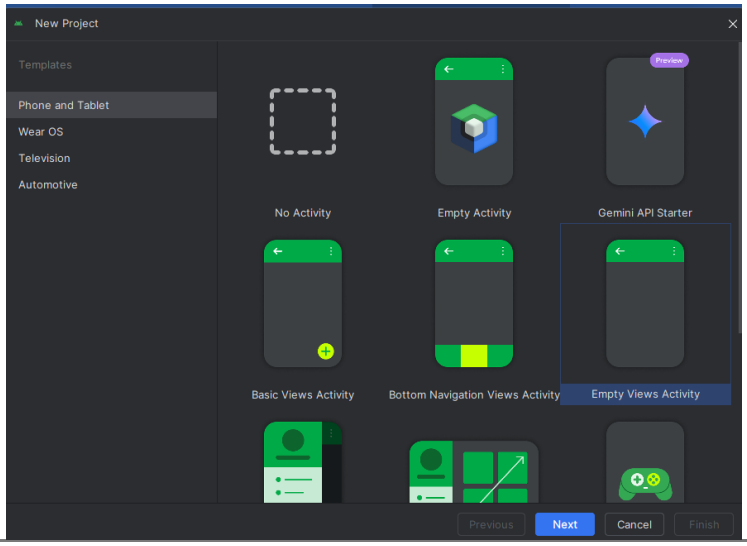
Descripción de Actividades:

Intents Explícitos e Implícitos en Android con Kotlin
En este ejercicio, crearemos una aplicación con dos actividades:

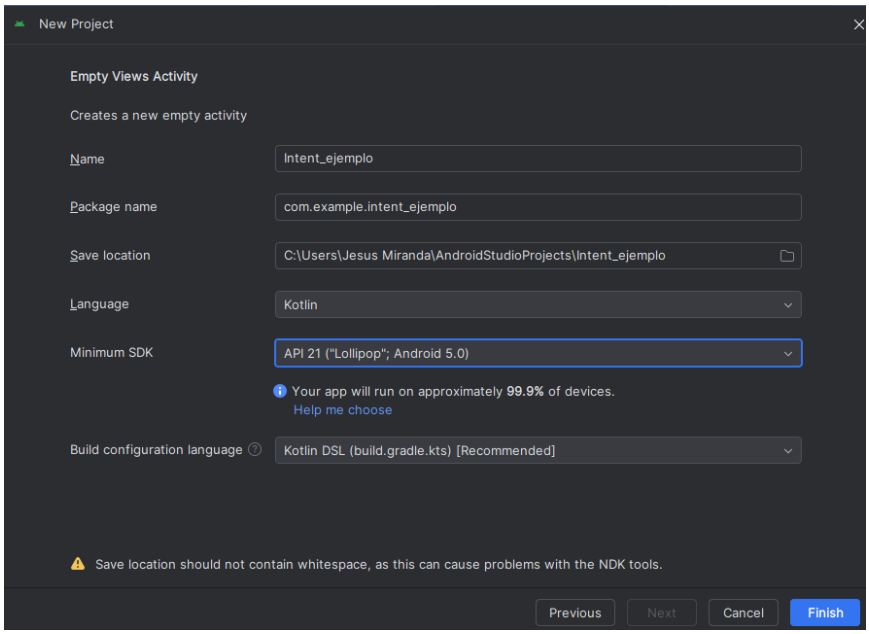
MainActivity: Tendrá botones para abrir una segunda actividad (Intent Explícito) y para abrir un enlace web (Intent Implícito).

SecondActivity: Mostrará un mensaje simple y un botón para regresar.

Paso 1: Crear un nuevo proyecto en Android Studio
Abre Android Studio y crea un nuevo proyecto (Empty Activity o Empty Views Activity según la versión de Android Studio).



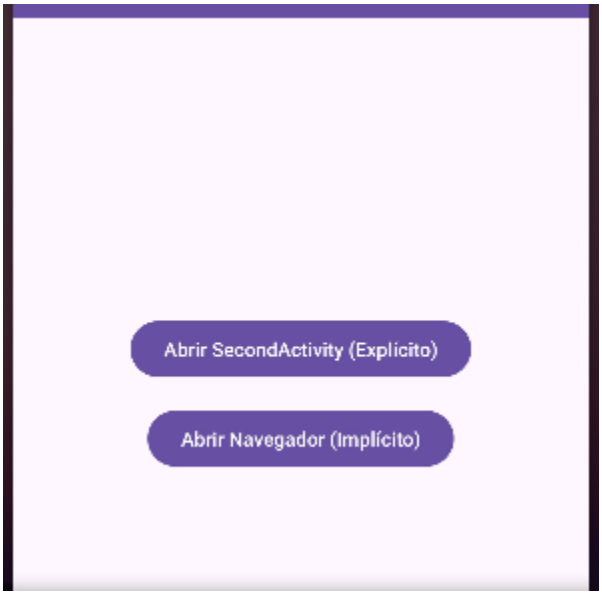
Configura el nombre (ej: Intent_ejemplo), lenguaje Kotlin y SDK mínimo (API 21+).



Paso 2: Diseñar la interfaz de MainActivity:
Abre activity_main.xml y añade dos botones:

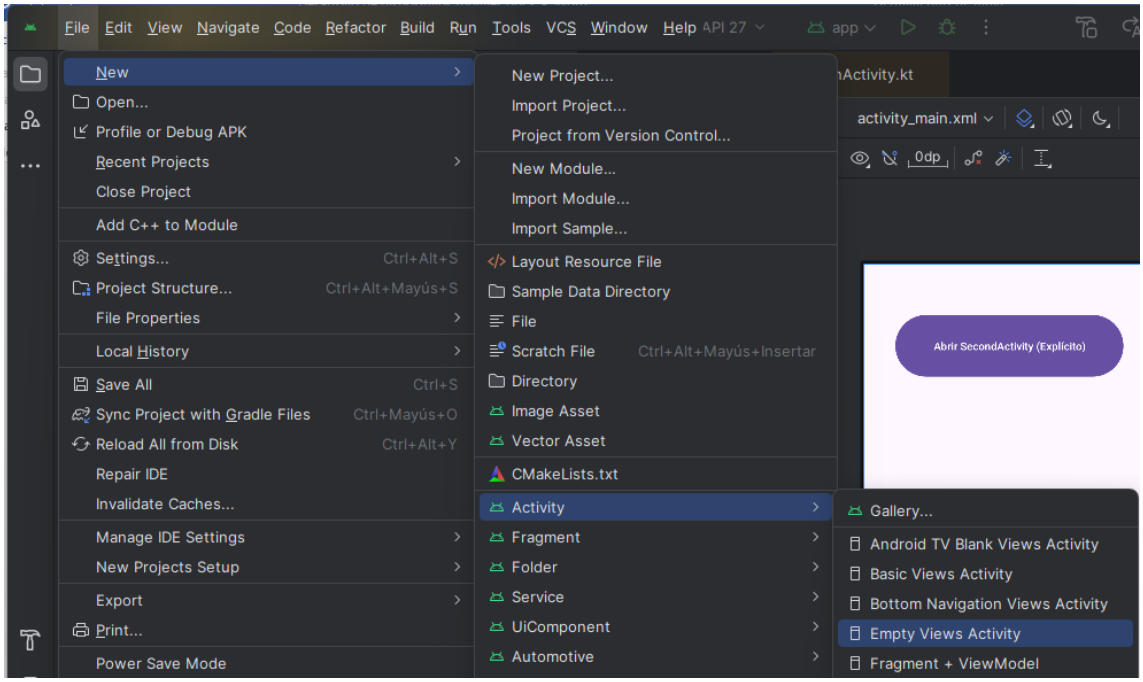
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="16dp">

    <Button
        android:id="@+id/btnExplicitIntent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Abrir SecondActivity (Explícito)"
        android:layout_marginBottom="16dp"/>

    <Button
        android:id="@+id/btnImplicitIntent"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Abrir Navegador (Implícito)"/>
</LinearLayout>
```

El diseño queda:

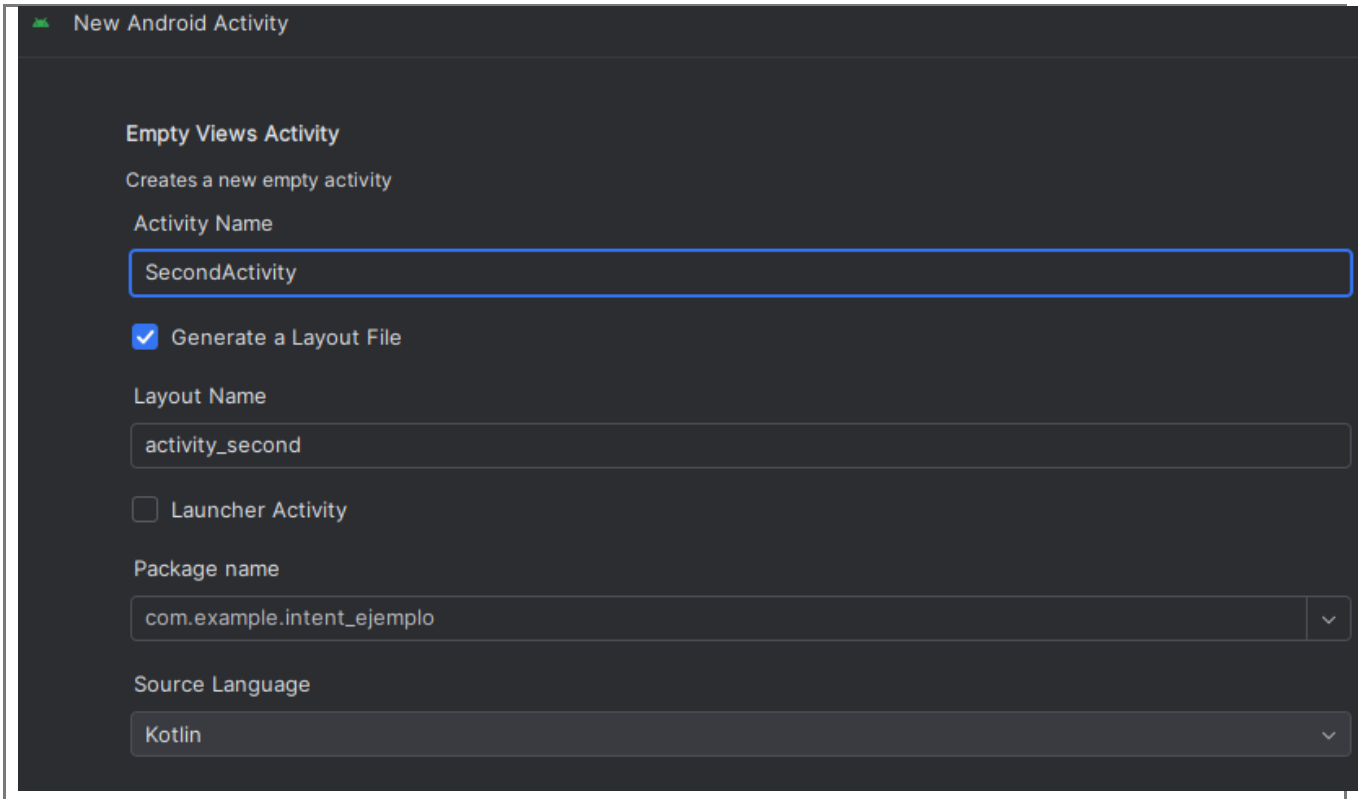


Paso 3: Crear SecondActivity

Haz clic derecho en tu paquete principal → New → Activity → Empty Activity.

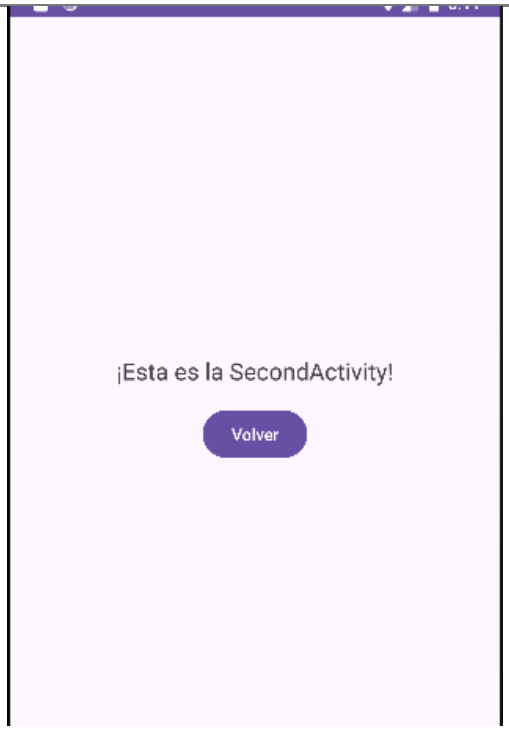


Nómbrala SecondActivity.



Edita activity_second.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:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/esta_es_el_segundoactivity"
        android:textSize="20sp"/>

    <Button
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/Regersar"
        android:layout_marginTop="16dp"/>
</LinearLayout>
```



Paso 4: Implementar Intents en MainActivity.kt
Abre MainActivity.kt y añade el siguiente código:

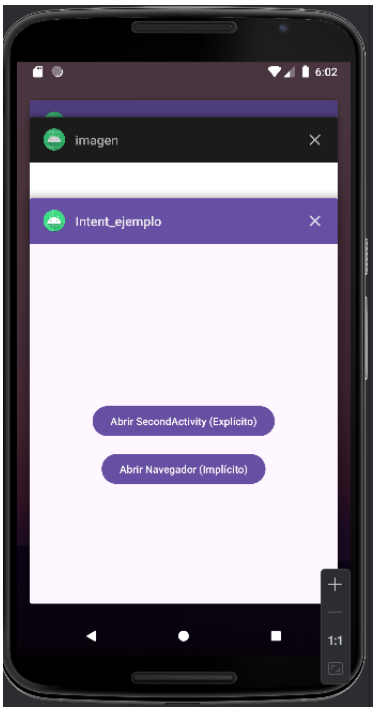
```
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Intent Explícito (Abrir SecondActivity)
        val btnExplicit = findViewById<Button>(R.id.btnExplicitIntent)
        btnExplicit.setOnClickListener {
            val intent = Intent(this, SecondActivity::class.java)
            startActivity(intent)
        }

        // Intent Implícito (Abrir navegador web)
        val btnImplicit = findViewById<Button>(R.id.btnImplicitIntent)
```

```
        btnImplicit.setOnClickListener {
            val url = "https://www.google.com"
            val intent = Intent(Intent.ACTION_VIEW, Uri.parse(url))
            startActivity(intent)
        }
    }
}
```

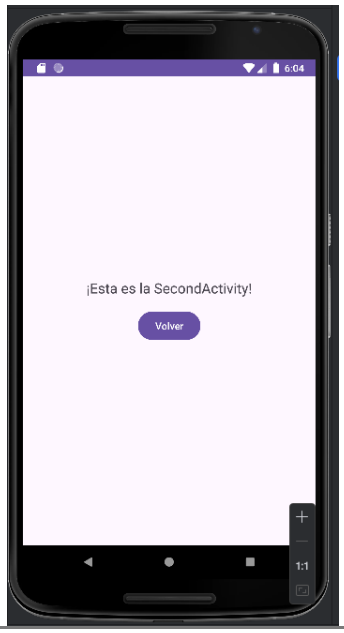
Paso 5: Implementar el botón de regreso en SecondActivity.kt
Abre SecondActivity.kt y añade:

```
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button

class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)

        val btnBack = findViewById<Button>(R.id.btnBack)
        btnBack.setOnClickListener {
            // Cierra la actividad actual y regresa a MainActivity
            finish()
        }
  }

 }
```

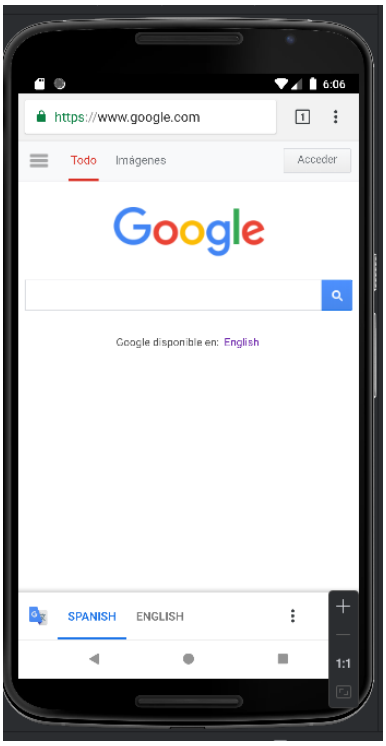
Paso 6: Probar la aplicación
Ejecuta la aplicación en un emulador o dispositivo físico.



Haz clic en "Abrir SecondActivity" para probar el Intent Explícito.



Haz clic en "Abrir Navegador" para probar el Intent Implícito (se abrirá Google Chrome o el navegador predeterminado).



RESULTADOS ESPERADOS

1. Funcionamiento Básico

– Interfaz inicial (MainActivity):

- Mostrará un botón con el texto " Abrir SecondActivity (Explícito)".
- Al presionarlo, la aplicación debe navegar sin errores a SecondActivity (Intent explícito).
- Mostrará un botón con el texto " Abrir Navegador (Implícito)".
- Al presionarlo, la aplicación debe abrir el navegador web con Google (Intent Implícito).

– SecondActivity:

- Mostrará un TextView con el mensaje: "Esta es la secondActivity".
- Incluirá un botón "Volver" que, al presionarlo, regresará a MainActivity (usando finish()).

ANÁLISIS DE RESULTADOS

¿Se completó la compilación sin errores?

¿El emulador funciona correctamente?

¿Qué problemas surgieron y cómo se resolvieron?

CONCLUSIONES Y REFLEXIONES

Los resultados esperados demuestran que la aplicación:

- Gestiona correctamente la navegación entre Activities (Intent explícito).
- (Si aplica) Interactúa con otras apps del sistema (Intent implícito).
- Ofrece una experiencia de usuario estable y sin errores críticos.

Estos resultados validan el dominio de los conceptos básicos de Intents en Android con Kotlin.

ACTIVIDADES COMPLEMENTARIAS

Descripción de la Actividad

Crea una aplicación llamada **"Tareas"** con las siguientes pantallas:

1. MainActivity (Pantalla Principal)

Contenido:

- Un Button con el texto **"Ver Tareas"** (Intent explícito).
- Un Button con el texto **"Llamar a Soporte"** (Intent implícito para marcado telefónico).
- Un Button con el texto **"Abrir Mapa"** (Intent implícito para Google Maps).

2. TareasActivity (Pantalla Secundaria)

Contenido:

- Un TextView que muestre: **"Mis Tareas Pendientes"**.
- Un Button con el texto **"Volver"** para regresar a MainActivity (usando finish()).

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	Desarrollo de la práctica 70%, reporte 30%
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica sobre Práctica de Laboratorio
Formatos de reporte de prácticas	Ejemplo de reporte en Anexos

FUENTES DE INFORMACIÓN

Google. (2023). Intents and intent filters. Android Developers. https://developer.android.com/guide/components/intents-filters?hl=es-419

Phillips, B., Hardy, B., & Marsicano, K. (2022). Android programming: The Big Nerd Ranch guide (5th ed.). Pearson.

NORMAS TÉCNICAS APLICABLES

Norma/Estándar	Ámbito de Aplicación en el Proyecto
ISO/IEC 25010	Calidad de la navegación entre Activities y manejo de errores.
NOM-008-SCFI-2002	Usabilidad en botones e interfaz.

NOMBRE DE LA PRÁCTICA	Diseño de Interface en App
COMPETENCIA DE LA PRÁCTICA	Diseñar una interfaz gráfica en Android Studio utilizando Kotlin, aplicando controles básicos, con el fin de implementar una calculadora funcional que realice operaciones aritméticas básicas, bajo los criterios de usabilidad y diseño de Material Design, desarrollando habilidades de pensamiento lógico y trabajo en equipo.

FUNDAMENTO TÉORICO

1. Introducción a las Interfaces en Android

En el desarrollo de aplicaciones móviles, el diseño de interfaces es fundamental para garantizar una experiencia de usuario intuitiva y eficiente. Android Studio proporciona herramientas para crear interfaces mediante XML (para el diseño) y Kotlin/Java (para la lógica).

- Componentes Básicos de una Interfaz en Android

- Activity: Representa una pantalla en la aplicación. Contiene la lógica y maneja la interacción del usuario.
- Views (Vistas): Elementos visuales como botones, campos de texto, imágenes, etc.
- Layouts: Estructuras que organizan las vistas en la pantalla (ej. ConstraintLayout, LinearLayout).
- **Controles Esenciales para una Calculadora**
  - EditText
- Función: Permite al usuario ingresar texto o números.
- Atributos clave:
- android:inputType="numberDecimal" → Solo permite números decimales.
- android:hint="Ingrese un número" → Texto de ayuda.
  - Button
- Función: Ejecuta acciones al ser presionado.
- Evento clave: setOnClickListener → Define qué ocurre al hacer clic.
  - TextView
- Función: Muestra texto estático o dinámico (como resultados).

- Uso en la calculadora: Mostrar el resultado de las operaciones.

3. Layouts en Android

Los layouts definen cómo se organizan los elementos en pantalla.

3.1. ConstraintLayout

- Ventaja: Permite posicionar elementos con restricciones (alinear respecto a otros componentes o bordes).
- **Ejemplo:**

```
<Button
    android:id="@+id/btnSumar"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/etNumero1"/>
```

3.2. LinearLayout

- Ventaja: Organiza elementos en fila (horizontal) o columna (vertical).
- **Ejemplo:**

```
<LinearLayout
  android:orientation="vertical">
  <EditText .../>
  <Button .../>
</LinearLayout>
```

4. Manejo de Eventos en Kotlin

- **Lógica de botones:** Se programa en el Activity usando listeners.
- **Ejemplo con Kotlin:**

```
btnSumar.setOnClickListener {
  val num1 = etNumero1.text.toString().toDouble()
    val num2 = etNumero2.text.toString().toDouble()
    tvResultado.text = "Resultado: ${num1 + num2}"
}
```

5. Material Design en Android

- **Objetivo:** Crear interfaces consistentes y atractivas.
- **Componentes clave:**
  - **MaterialButton:** Botones con efectos de elevación y formas redondeadas.
  - **Paleta de colores:** Personalización en res/values/colors.xml.
  - **Temas:** Definidos en res/values/themes.xml.

6. Paleta de Colores en Android

6.1. Importancia del Uso de Colores

En el diseño de interfaces móviles, la paleta de colores cumple funciones esenciales:

- **Identidad visual:** Refleja la marca de la aplicación
- **Usabilidad:** Mejora la legibilidad y jerarquía visual
- **Experiencia de usuario:** Guía la atención del usuario hacia elementos importantes

6.2 Implementación Técnica

Android maneja los colores a través de:

- **Archivo colors.xml:** Define todos los colores en **res/values/colors.xml**

```
<color name="primary">#3F51B5</color>
<color name="accent">#FF4081</color>
```

- **Color Primary (Primario)**
  `<color name="primary">#3F51B5</color>`
  Qué representa:
  - **Color principal** de identidad de la aplicación
  - **Color de marca** que domina la interfaz
  - Se usa en elementos clave como:
    - Barra de la aplicación (AppBar/Toolbar)
    - Botones importantes
    - Elementos de navegación
- **Características:**
  - Normalmente es un color **azul, morado o verde** (en Material Design)
  - En el ejemplo: #3F51B5 es un azul índigo estándar de Material
  - Debe tener buen contraste con texto blanco (colorOnPrimary)

- **Color Accent (Acento)**
  `<color name="accent">#FF4081</color>`
  Qué representa:
  - **Color de énfasis** para elementos interactivos
  - **Puntos focales** que atraen la atención
  - Se aplica a:
    - Botones flotantes (FAB)
    - Campos de formulario activos
    - Elementos seleccionados
    - Animaciones y efectos

  Características:
  - Suele ser un color **más vibrante** que el primario
  - En el ejemplo: #FF4081 es un rosa brillante
  - Idealmente debe contrastar bien con ambos fondos claros y oscuros

- **Visualización en la Interfaz**

Elemento UI	Color típico	Ejemplo en Calculadora
AppBar/Toolbar	primary	-
Botones principales	primary	Botones operaciones
Switch activo	accent	-
FAB	accent	-
Bordes de campos activos	accent	Bordes de EditText al seleccionar

- **Relación con el Tema (themes.xml)**

Estos colores se asignan en el tema de la aplicación:

```
<style name="Theme.MyApp" parent="Theme.MaterialComponents">
  <item name="colorPrimary">@color/primary</item>
  <item name="colorPrimaryVariant">@color/primary_dark</item>
  <item name="colorOnPrimary">@color/white</item>

  <item name="colorSecondary">@color/accent</item> <!-- Nota: "accent" se llama "secondary" en versiones recientes -->
  <item name="colorSecondaryVariant">@color/accent_dark</item>
  <item name="colorOnSecondary">@color/white</item>
</style>
```

7. **Componente Toast**

7.1 Concepto y Propósito

Mensaje emergente breve que:

- Proporciona retroalimentación inmediata
- No interrumpe el flujo de la aplicación
- Desaparece automáticamente

7.2. Implementación Básica:

```
Toast.makeText(context, "Mensaje", Toast.LENGTH_SHORT).show()
```

- Duración:
  - LENGTH_SHORT (2 segundos)
  - LENGTH_LONG (3.5 segundos)

- **Posicionamiento:**

```
val toast = Toast.makeText(...)
toast.setGravity(Gravity.TOP, 0, 100)
toast.show()
```

7.3 Casos de Uso en la Calculadora

1. **Validación de entrada:**

```
if (etNumero1.text.isEmpty()) {
    Toast.makeText(this, "Ingrese el primer número", Toast.LENGTH_SHORT).show()
}
```

2. **Error matemático:**

```
try {
    val resultado = num1 / num2
} catch (e: ArithmeticException) {
    Toast.makeText(this, "No se puede dividir por cero", Toast.LENGTH_LONG).show()
}
```

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Material/Equipo	Cantidad	Especificaciones
Computadora	1	Windows/macOS/Linux, mínimo 8GB RAM, 10GB espacio libre
Conexión a Internet	1	Banda ancha estable
Android Studio	1	Última versión estable descargable desde developer.android.com

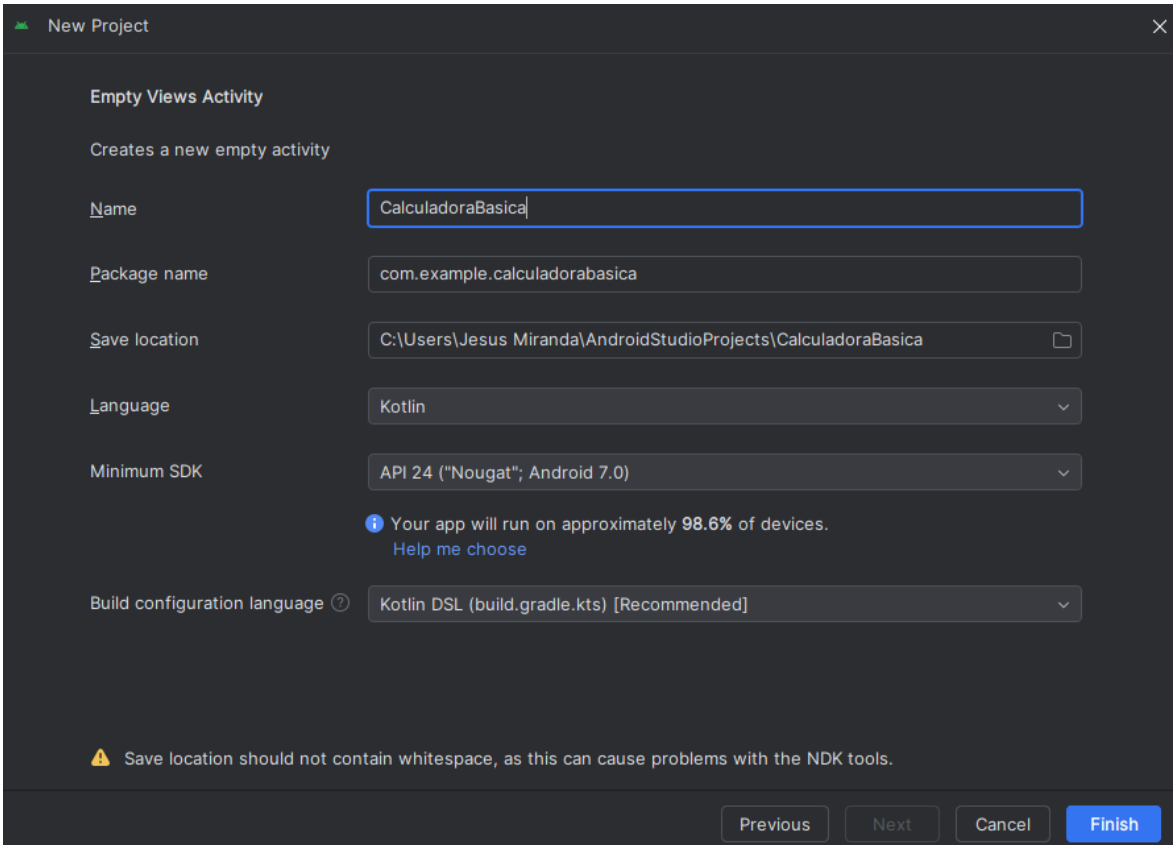
PROCEDIMIENTO O METODOLOGÍA

Descripción de Actividades

Paso 1: Crear un Proyecto en Android Studio

1. Abrir Android Studio → New Project → Empty Activity.

2. Nombrar el proyecto: "CalculadoraBasica".
3. Seleccionar lenguaje: Kotlin.



Paso 2: Diseñar la Interfaz en activity_main.xml

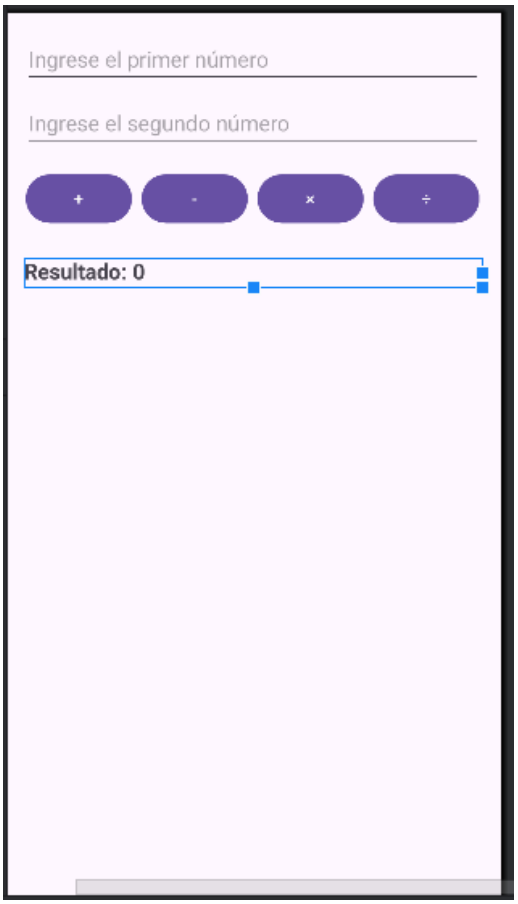
- Usar ConstraintLayout para organizar los elementos.
- Agregar:
    - Dos EditText para números.
    - Cuatro Button (Suma, Resta, Multiplicación, División).
    - Un TextView para mostrar el resultado.

El código queda:

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

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    tools:context=".MainActivity">

    <!-- Campo para el primer número -->
    <EditText
        android:id="@+id/etNumero1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Ingrese el primer número"
        android:inputType="numberDecimal"/>

    <!-- Campo para el segundo número -->
    <EditText
        android:id="@+id/etNumero2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:hint="Ingrese el segundo número"
        android:inputType="numberDecimal"/>

    <!-- Botones de operaciones -->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:orientation="horizontal"
        android:gravity="center">

        <Button
            android:id="@+id/btnSumar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="+"/>

        <Button
            android:id="@+id/btnRestar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:text="-"/>

        <Button
            android:id="@+id/btnMultiplicar"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
```

```
            android:text="×"/>

        <Button
            android:id="@+id/btnDividir"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:text="÷"/>
    </LinearLayout>

    <!-- Resultado -->
    <TextView
        android:id="@+id/tvResultado"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:text="Resultado: 0"
        android:textSize="18sp"
        android:textStyle="bold"/>
</LinearLayout>
```



Archivo MainActivity.kt **(Lógica de la Calculadora)**

```
package com.example.calculadorabasica

import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Referencias a los elementos de la interfaz
        val etNumero1 = findViewById<EditText>(R.id.etNumero1)
        val etNumero2 = findViewById<EditText>(R.id.etNumero2)
        val btnSumar = findViewById<Button>(R.id.btnSumar)
        val btnRestar = findViewById<Button>(R.id.btnRestar)
        val btnMultiplicar = findViewById<Button>(R.id.btnMultiplicar)
        val btnDividir = findViewById<Button>(R.id.btnDividir)
        val tvResultado = findViewById<TextView>(R.id.tvResultado)

        // Función para validar si los campos están vacíos
        fun validarCampos(): Boolean {
            return etNumero1.text.isNotEmpty() && etNumero2.text.isNotEmpty()
        }

        // Función para obtener los números ingresados
        fun obtenerNumeros(): Pair<Double, Double> {
            val num1 = etNumero1.text.toString().toDouble()
            val num2 = etNumero2.text.toString().toDouble()
            return Pair(num1, num2)
        }

        // Evento para el botón de suma
        btnSumar.setOnClickListener {
            if (validarCampos()) {
                val (num1, num2) = obtenerNumeros()
                val resultado = num1 + num2
                tvResultado.text = "Resultado: $resultado"
            } else {
                Toast.makeText(this, "Ingrese ambos números", Toast.LENGTH_SHORT).show()
            }
        }
```

```
        // Evento para el botón de resta
        btnRestar.setOnClickListener {
            if (validarCampos()) {
                val (num1, num2) = obtenerNumeros()
                val resultado = num1 - num2
                tvResultado.text = "Resultado: $resultado"
            } else {
                Toast.makeText(this, "Ingrese ambos números", Toast.LENGTH_SHORT).show()
            }
        }

        // Evento para el botón de multiplicación
        btnMultiplicar.setOnClickListener {
            if (validarCampos()) {
                val (num1, num2) = obtenerNumeros()
                val resultado = num1 * num2
                tvResultado.text = "Resultado: $resultado"
            } else {
                Toast.makeText(this, "Ingrese ambos números", Toast.LENGTH_SHORT).show()
            }
        }

        // Evento para el botón de división
        btnDividir.setOnClickListener {
            if (validarCampos()) {
                val (num1, num2) = obtenerNumeros()
                if (num2 != 0.0) {
                    val resultado = num1 / num2
                    tvResultado.text = "Resultado: $resultado"
                } else {
                    Toast.makeText(this, "No se puede dividir entre cero", Toast.LENGTH_SHORT).show()
                }
            } else {
                Toast.makeText(this, "Ingrese ambos números", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

RESULTADOS ESPERADOS

- Operaciones básicas de suma, resta, multiplicación y división funcionando correctamente.
- Datos de entrada validados por la App.

- Detección de errores matemáticos

ANÁLISIS DE RESULTADOS

Prueba	Entrada	Operación	Salida Esperada	Resultado
1	5, 3	Suma	8	✓ Correcto
2	10, 2	División	5	✓ Correcto
3	"abc", 5	Suma	Error	✗ Validar entrada

CONCLUSIONES Y REFLEXIONES

Se logró implementar una calculadora funcional con Kotlin.

Se aplicaron conceptos de diseño de interfaz (EditText, Button, TextView).

Es importante validar entradas para evitar errores.

ACTIVIDADES COMPLEMENTARIAS

Desarrollar una aplicación Android que:

1. Muestre una lista de productos usando ListView
2. Permita seleccionar un elemento
3. Muestre los detalles en una nueva Activity

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE

Criterios de evaluación	Desarrollo de la práctica 70%, reporte 30%
Rúbricas o listas de cotejo para valorar desempeño	Rúbrica sobre Práctica de Laboratorio
Formatos de reporte de prácticas	Ejemplo de reporte en Anexos

FUENTES DE INFORMACIÓN

Google. (2023). *Android Developers Documentation*. https://developer.android.com/docs

Google. (2022). *Material Design 3*. https://m3.material.io/

Google. (2023). *Toast | Android Developers*. https://developer.android.com/reference/android/widget/Toast

Google. (2023). *Material Components for Android*. https://github.com/material-components/material-components-android

Google. (2023). *Color System in Material Design*. https://m2.material.io/design/color/the-color-system.html

NORMAS TÉCNICAS APLICABLES

Tabla de cumplimiento normativo

Aspecto	Norma aplicable	Implementación en la práctica
Funcionalidad	ISO 25010	Operaciones aritméticas precisas
Símbolos matemáticos	NOM-003-SCFI-2014	Uso de "×" y "÷" estándar
Manejo de errores	ISO 9241-110	Toast para divisiones por cero
Contraste visual	ISO/IEC 40500	Paleta de colores accesible
Tamaño interactivo	NOM-034-STPS-2016	Botones de 48x48 dp mínimo

NOMBRE DE LA PRÁCTICA	App C.R.U. D.
COMPETENCIA DE LA PRÁCTICA	Implementar operaciones CRUD (Create, Read, Update, Delete) para gestionar datos persistentes utilizando SQLite en Android Studio, mediante el lenguaje Kotlin y buenas prácticas de desarrollo, trabajando de manera colaborativa y resolviendo problemas técnicos con pensamiento crítico"

FUNDAMENTO TÉORICO

SQLite en Android

SQLite es una base de datos relacional ligera y autocontenida que se integra directamente en las aplicaciones Android. A diferencia de otras bases de datos como MySQL o PostgreSQL, SQLite no requiere un servidor separado, lo que la hace ideal para aplicaciones móviles donde los recursos son limitados. En Android, SQLite es la opción predilecta para el almacenamiento local de datos estructurados.

Características clave de SQLite en Android:

Sin servidor: Toda la base de datos se almacena en un solo archivo dentro del dispositivo.

Transaccional: Soporta transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

Ligera: Ocupa poco espacio y consume pocos recursos, ideal para dispositivos móviles.

Integrada: Viene incluida en el sistema operativo Android, por lo que no requiere instalación adicional.

Componentes Principales de SQLite en Android

Para trabajar con SQLite en Android, se utilizan varias clases y componentes proporcionados por el SDK de Android:

SQLiteOpenHelper:
Es una clase auxiliar que facilita la creación y gestión de la base de datos. Proporciona dos métodos principales:

onCreate(): Se llama cuando la base de datos se crea por primera vez. Aquí se define la estructura de la base de datos (tablas, índices, etc.).

onUpgrade(): Se llama cuando la base de datos necesita ser actualizada (por ejemplo, al cambiar la versión de la base de datos).

SQLiteDatabase:
Representa la base de datos y proporciona métodos para ejecutar consultas SQL, como:

insert(): Para insertar nuevos registros.

query(): Para realizar consultas.

update(): Para actualizar registros existentes.

delete(): Para eliminar registros.

ContentValues:
Es una clase utilizada para almacenar pares clave-valor que representan los datos a insertar o actualizar en la base de datos. Es especialmente útil para operaciones de inserción y actualización.

Cursor:
Es un objeto que permite recorrer los resultados de una consulta. Proporciona métodos como moveToFirst(), moveToNext(), getString(), getInt(), etc., para acceder a los datos.

- Operaciones CRUD en SQLite

Las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) son la base de cualquier sistema de gestión de bases de datos. En SQLite, estas operaciones se implementan de la siguiente manera:

Crear (Create):
Se utiliza el método insert() de la clase SQLiteDatabase para agregar nuevos registros a una tabla. Los datos se pasan en un objeto ContentValues.

Leer (Read):
Se utiliza el método query() para realizar consultas. Los resultados se obtienen en un objeto Cursor, que permite recorrer los registros.

Actualizar (Update):
Se utiliza el método update() para modificar registros existentes. Al igual que en la inserción, los datos se pasan en un objeto ContentValues.

Eliminar (Delete):
Se utiliza el método delete() para borrar registros de una tabla.

Kotlin en Android
Kotlin es un lenguaje de programación moderno y conciso que se ha convertido en el lenguaje preferido para el desarrollo de aplicaciones Android. Algunas de sus ventajas incluyen:

Interoperabilidad con Java: Kotlin es completamente interoperable con Java, lo que permite utilizar bibliotecas y frameworks existentes.

Sintaxis simplificada: Kotlin reduce la cantidad de código boilerplate, haciendo que el código sea más legible y mantenible.

Seguridad nula: Kotlin incluye soporte para tipos nulos, lo que ayuda a evitar errores comunes como NullPointerException.

En el contexto de SQLite, Kotlin simplifica la implementación de operaciones de base de datos gracias a su sintaxis concisa y su capacidad para trabajar con tipos de datos de manera segura.

- **Arquitectura de una Aplicación con SQLite**

Para mantener una estructura limpia y mantenible, es recomendable seguir el patrón de arquitectura Modelo-Vista-Presentador (MVP) o Modelo-Vista-ViewModel (MVVM) al trabajar con SQLite en Android. Esto implica:

Capa de Datos:

Aquí se encuentra la clase DatabaseHelper que gestiona la base de datos.

También se pueden definir clases DAO (Data Access Object) para encapsular las operaciones de base de datos.

Capa de Presentación:

Contiene la lógica de la interfaz de usuario y se comunica con la capa de datos para obtener o modificar información.

Capa de Vista:

Es la interfaz de usuario (Activity, Fragment, RecyclerView, etc.) que muestra los datos al usuario.

Ventajas y Desventajas de SQLite en Android

Ventajas	Desventajas
Fácil de usar y configurar.	No es adecuado para aplicaciones con alta concurrencia.
No requiere un servidor separado.	Limitado en cuanto a escalabilidad
Almacena toda la base de datos en un archivo	No soporta usuarios y permisos avanzados
Soporta transacciones ACID.	Menos funcionalidades avanzadas comparado con otros RDBMS

- **Alternativas a SQLite en Android**

Aunque SQLite es la opción predeterminada para el almacenamiento local en Android, existen alternativas que pueden ser más adecuadas dependiendo del caso de uso:

Room:
Es una biblioteca de persistencia que forma parte de Android Jetpack. Room proporciona una capa de abstracción sobre SQLite, haciendo que el código sea más seguro y fácil de mantener.

Realm:
Es una base de datos NoSQL que ofrece un enfoque más moderno y orientado a objetos para el almacenamiento local.

Firebase Realtime Database:
Es una base de datos en la nube que sincroniza datos en tiempo real entre dispositivos.

MATERIALES, EQUIPAMIENTO Y/O REACTIVOS

Material/Equipo	Cantidad	Especificaciones
Computadora	1	Windows/macOS/Linux, mínimo 8GB RAM, 10GB espacio libre
Conexión a Internet	1	Banda ancha estable
Android Studio	1	Última versión estable descargable desde developer.android.com
SQLite	1	Última versión estable descargable desde https://sqlite.org/

PROCEDIMIENTO O METODOLOGÍA

Descripción de Actividades

Actividad 1: Crear la Base de Datos y la Tabla

1. Crea una clase DatabaseHelper que herede de SQLiteOpenHelper.
2. Define una tabla llamada usuarios con las siguientes columnas:
    - id (INTEGER, PRIMARY KEY, AUTOINCREMENT)
    - nombre (TEXT)
    - edad (INTEGER)
    - email (TEXT)

.

Código de referencia:

```
import android.content.ContentValues
```

```
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "usuarios.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "usuarios"
        private const val COLUMN_ID = "id"
        private const val COLUMN_NOMBRE = "nombre"
        private const val COLUMN_EDAD = "edad"
        private const val COLUMN_EMAIL = "email"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE $TABLE_NAME (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_NOMBRE TEXT,
                $COLUMN_EDAD INTEGER,
                $COLUMN_EMAIL TEXT
            )
        """
        db.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertarUsuario(nombre: String, edad: Int, email: String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NOMBRE, nombre)
            put(COLUMN_EDAD, edad)
            put(COLUMN_EMAIL, email)
        }
        val result = db.insert(TABLE_NAME, null, values)
        return result != -1L
    }

    fun obtenerUsuarios(): List<Usuario> {
        val usuarios = mutableListOf<Usuario>()
        val db = readableDatabase
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
```

```
        if (cursor.moveToFirst()) {
            do {
                val id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_ID))
                val nombre = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NOMBRE))
                val edad = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_EDAD))
                val email = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_EMAIL))

                usuarios.add(Usuario(id, nombre, edad, email))
            } while (cursor.moveToNext())
        }

        cursor.close()
        return usuarios
    }

    fun eliminarUsuario(id: Int): Boolean {
        val db = writableDatabase
        val result = db.delete(TABLE_NAME, "$COLUMN_ID = ?", arrayOf(id.toString()))
        return result > 0
    }
}
```

2. Crea la clase Usuarios y agregarle el siguiente código:

Clase Usuario.kt

```
data class Usuario(
    val id: Int,
    val nombre: String,
    val edad: Int,
    val email: String
)
```

3. Agregar el siguiente còdigo en el Activity_main.xml:

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

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
```

```
    <EditText
        android:id="@+id/nombre_input"
        android:hint="Nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/edad_input"
        android:hint="Edad"
        android:inputType="number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/email_input"
        android:hint="Email"
        android:inputType="textEmailAddress"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/boton_agregar"
        android:text="Agregar Usuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <EditText
        android:id="@+id/id_eliminar_input"
        android:hint="ID a eliminar"
        android:inputType="number"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

    <Button
        android:id="@+id/boton_eliminar"
        android:text="Eliminar Usuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>

    <TextView
        android:id="@+id/lista_usuarios"
        android:layout_marginTop="16dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
  </LinearLayout>
</ScrollView>
```

4. Agregar el siguiente còdigo al archivo MainActivity.kt:

```
import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DatabaseHelper(context: Context) :
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "usuarios.db"
        private const val DATABASE_VERSION = 1
        private const val TABLE_NAME = "usuarios"
        private const val COLUMN_ID = "id"
        private const val COLUMN_NOMBRE = "nombre"
        private const val COLUMN_EDAD = "edad"
        private const val COLUMN_EMAIL = "email"
    }

    override fun onCreate(db: SQLiteDatabase) {
        val createTable = """
            CREATE TABLE $TABLE_NAME (
                $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
                $COLUMN_NOMBRE TEXT,
                $COLUMN_EDAD INTEGER,
                $COLUMN_EMAIL TEXT
            )
        """
        db.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int, newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertarUsuario(nombre: String, edad: Int, email: String): Boolean {
        val db = writableDatabase
        val values = ContentValues().apply {
            put(COLUMN_NOMBRE, nombre)
            put(COLUMN_EDAD, edad)
            put(COLUMN_EMAIL, email)
        }
        val result = db.insert(TABLE_NAME, null, values)
        return result != -1L
    }
```

```
    fun obtenerUsuarios(): List<Usuario> {
        val usuarios = mutableListOf<Usuario>()
        val db = readableDatabase
        val cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)

        if (cursor.moveToFirst()) {
            do {
                val id = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_ID))
                val nombre = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_NOMBRE))
                val edad = cursor.getInt(cursor.getColumnIndexOrThrow(COLUMN_EDAD))
                val email = cursor.getString(cursor.getColumnIndexOrThrow(COLUMN_EMAIL))

                usuarios.add(Usuario(id, nombre, edad, email))
            } while (cursor.moveToNext())
        }

        cursor.close()
        return usuarios
    }

    fun eliminarUsuario(id: Int): Boolean {
        val db = writableDatabase
        val result = db.delete(TABLE_NAME, "$COLUMN_ID = ?", arrayOf(id.toString()))
        return result > 0
    }
}
```

5. Ejecutar la Aplicaciòn y hacer las pruebas de inserciòn y eliminaciòn de registros.

RESULTADOS ESPERADOS

La aplicación se ejecuta correctamente en el emulador.
La App ejecuta las operaciones CRUD según lo esperado.
.

ANÁLISIS DE RESULTADOS

Casos de prueba funcionales

Prueba N°	Acción	Entrada	Resultado esperado	Resultado obtenido	Estado
1	Insertar usuario válido	Nombre: Juan, Edad: 28, Email: juan@...	Usuario agregado, se muestra en lista con ID autogenerado	□ ID: 1, Juan...	Éxito
2	Insertar sin edad	Nombre: Ana, Edad: "", Email: ana@...	Edad asignada como 0 por `toIntOrNull() ?: 0`	□ ID: 2, Ana...	Éxito
3	Eliminar usuario existente	ID: 1	Usuario eliminado, lista actualizada	□ ID 1 eliminado	Éxito
4	Eliminar usuario inexistente	ID: 99	Mensaje de error, sin cambios en la lista	□ Error esperado	Éxito
5	Ver lista vacía	Sin usuarios en DB	Se muestra lista vacía	□ Lista vacía	Éxito

- La app ejecuta correctamente operaciones CRUD básicas.
- Uso efectivo de SQLite con autoincremento.
- Visualización clara de los usuarios insertados.
- Interfaz sencilla y funcional.

CONCLUSIONES Y REFLEXIONES

SQLite es una solución eficiente para almacenamiento local en Android.

El uso de Kotlin simplifica la implementación de operaciones CRUD.

La práctica reforzó habilidades en persistencia de datos, programación en Android y depuración.

ACTIVIDADES COMPLEMENTARIAS

Desarrolla una aplicación Android en **Kotlin** que permita administrar una biblioteca personal mediante operaciones **CRUD** sobre una base de datos SQLite.

Tabla: libros

Columna	Tipo de dato	Descripción
`id`	`INTEGER PRIMARY KEY AUTOINCREMENT`	Identificador único del libro
`titulo`	`TEXT`	Título del libro
`autor`	`TEXT`	Nombre del autor
`paginas`	`INTEGER`	Número de páginas
`leido`	`INTEGER` (0 = No leído, 1 = Leído)	Estado de lectura del libro

Interfaz sugerida

- **EditTexts**:
  - Título
  - Autor
  - Páginas
  - ID (para actualizar/eliminar)
- **Switch o Checkbox**: para indicar si el libro fue leído
- **Botones**:
  - Agregar libro
  - Actualizar libro
  - Eliminar libro
  - Mostrar libros
- **Lista de libros** en un TextView o RecyclerView

EVALUACIÓN Y EVIDENCIAS DE APRENDIZAJE	
Criterios de evaluación	Desarrollo de la práctica 70%, reporte 30%
Rúbricas o listas de cotejo para valorar	Rúbrica sobre Práctica de Laboratorio

desempeño	
Formatos de reporte de prácticas	Ejemplo de reporte en Anexos

FUENTES DE INFORMACIÓN

Google. (2023). Android Studio documentation. Android Developers. https://developer.android.com/studio/intro?hl=es-419

Android Developers. (2023). *SQLite Database*. Google. https://developer.android.com/training/data-storage/sqlite

NORMAS TÉCNICAS APLICABLES

ISO/IEC 25010: Calidad de software (usabilidad, funcionalidad).

NOM-151-SCFI-2016: Prácticas de desarrollo seguro en aplicaciones móviles (México).

UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu

ANEXOS

1.- Ejemplos de reportes

Ingeniería en Software

Asignatura:

Desarrollo de Aplicaciones Móviles

Práctica No. 2

Nombre de la práctica:

Implementación de Intents Explícitos e Implícitos

Nombre del alumno:

Fecha:

Nombre del maestro:

Introducción:

En el desarrollo de aplicaciones móviles para Android, los Intents son un mecanismo fundamental para la comunicación entre componentes de una aplicación o entre diferentes aplicaciones. Esta práctica tuvo como objetivo implementar dos tipos de Intents:

Explícitos: Para navegar entre Activities dentro de la misma aplicación.

Implícitos: Para interactuar con aplicaciones externas (como navegadores web).

Se utilizó Kotlin en Android Studio para crear una aplicación con una interfaz sencilla que demostrara el funcionamiento de ambos tipos de Intents.

Materiales y Métodos:

- **Materiales Utilizados**

Material/Herramienta	Especificaciones
Computadora Windows 11	8GB RAM
Android Studio	Versión 2023.1.1
Emulador Android	Pixel 4, API 33

- **Desarrollo de la Práctica**

1. **Creación del proyecto**:
   - Se inició un nuevo proyecto en Android Studio con **Empty Activity** y lenguaje Kotlin.
2. **Diseño de interfaces**:
   - activity_main.xml: Contiene un botón para abrir la segunda Activity.
   - activity_second.xml: Muestra un TextView con el nombre del desarrollador y un botón para cerrar.
3. **Implementación de lógica**:
   - **MainActivity.kt**: Configura el botón para lanzar SecondActivity usando un Intent explícito.
   - **SecondActivity.kt**: Cierra la Activity actual al presionar el botón.
4. **Pruebas**:
   - Se ejecutó la aplicación en el emulador para verificar el funcionamiento correcto.
5. **Anexos:**
   - Código Fuente

   (Disponible en: [Enlace a drive/repositorio privado/ GitHub].)

Resultados y Evidencias:

- **Capturas de Pantalla:**

1. **MainActivity**:

   (Imagen del MainActivity)

2. **SecondActivity**:

   (Imagen del SecondActivity)

- **Observaciones:**

  — El Intent explícito funcionó correctamente para navegar entre Activities.
  — Al presionar "Cerrar", la SecondActivity se finalizó y se regresó a MainActivity.

Análisis de Resultados

- **¿Por qué es importante declarar Activities en el AndroidManifest.xml?**

Porque el sistema Android necesita conocer los componentes de la aplicación para gestionarlos correctamente.

- **¿Qué pasaría si no se usa finish() en el botón "Cerrar"?**

La SecondActivity permanecería en la pila de Activities, consumiendo recursos innecesarios.

- **Diferencia entre startActivity() y startActivityForResult()**

startActivityForResult() permite recibir datos de retorno de la Activity llamada.

Conclusiones

1. Los **Intents explícitos** son esenciales para la navegación interna en una aplicación.
2. Los **Intents implícitos** permiten integrar funcionalidades de otras apps (como navegadores).
3. Kotlin simplifica la implementación de Intents con sintaxis concisa.


UES
Universidad Estatal de Sonora
La Fuerza del Saber Estimulará mi Espíritu