

UNIVERSIDAD ICEL

“CAMPUS ERMITA”

SEMANA 8

“PROYECTO INTEGRADOR”

“Creación de una App de ventas en Android Studio”

Nombre: Isaías Guadarrama Cristiani

Escolaridad: Licenciatura en Ingeniería en Sistemas Computacionales

Modalidad: Ejecutiva

ID: 000037914

Materia: Aplicaciones Móviles

Nombre del docente: Juan Carlos Pérez Cadena

Fecha de entrega: 4 de Julio de 2025



INDICE TEMATICO:

1. Introducción Creación de una App de ventas”	4
DEFINICIÓN DEL PROBLEMA Y OBJETIVOS.....	
.....4,5	
2. LEVANTAMIENTO DE INFORMACIÓN.....	
..... 3	
3. DEFINICIÓN DEL PROBLEMA.....	
.....5	
4.	
OBJETIVOS.....	
.....5	
5. REQUERIMIENTOS	
.....	
.....	
RESTRICCIONES.	9,10
7. DETERMINACIÓN DE VIABILIDAD	
.....	
..... 11,12	
10. PLANEACIÓN Y CONTROL DE ACTIVIDADES	
.....	
Como implementarlo en una Base de datos.....	8,9,10,11
12. DIAGRAMA	
UML.....	
.....12	
13. Diagrama E-R	
.....	

.....	14
15. ASPECTOS ÉTICOS Y LEGALES.....	25
16. Conclusión de proyecto	26

Documentación del Proyecto: App de Ventas Android

Introducción:

La presente aplicación de ventas desarrollada en Android Studio tiene como objetivo proporcionar una solución sencilla y eficiente para gestionar el inventario de productos, registrar ventas y visualizar la información de forma clara y ordenada. Está pensada para funcionar completamente de forma local, sin necesidad de conexión a internet, lo que la hace ideal para comerciantes que operan en entornos con recursos tecnológicos limitados.

Utilizando el lenguaje Kotlin y SQLite como base de datos local, la app permite registrar productos con su nombre, precio y cantidad disponible, así como eliminarlos y consultar el estado actual del inventario. La interfaz es intuitiva y está diseñada para usuarios sin experiencia técnica, brindando una herramienta útil y práctica para el día a día comercial.

1. Definición del Problema

En muchos pequeños negocios o vendedores independientes no existe una herramienta accesible, móvil y sencilla para registrar ventas, productos y controlar el inventario. Esto genera pérdidas por desorganización y falta de información en tiempo real.

2. Objetivos del Proyecto

Objetivo General:

Desarrollar una aplicación móvil en Android Studio para gestionar ventas, productos e inventarios de manera eficiente y accesible para pequeños comerciantes.

Objetivos Específicos:

Crear una interfaz amigable para registrar productos.

Implementar funciones para registrar y consultar ventas.

Generar un pequeño historial de transacciones.

Permitir la edición y eliminación de productos.

Mantener los datos localmente mediante SQLite (o Room).

3. Impacto Esperado

Tecnológico: Brindar una solución digital moderna a pequeños negocios.

Económico: Optimizar procesos de venta y reducir pérdidas por errores manuales.

Social: Facilitar la digitalización de negocios sin acceso a sistemas complejos.

4. Cronograma de Actividades (posible ejemplo)

Semana	Actividad	Responsable	Estado
1	Análisis de requerimientos	Desarrollador	✓
1	Diseño de interfaz (wireframes)	Desarrollador	✓
2	Implementación de base de datos (SQLite)	Desarrollador	✓
2	Registro y edición de productos	Desarrollador	✓
3	Módulo de ventas e historial	Desarrollador	⌚
4	Pruebas y ajustes	Desarrollador	□
4	Documentación final	Desarrollador	□

5. Funcionalidades de la App

Añadir productos con nombre, precio y cantidad.

Editar y eliminar productos.

Registrar una venta seleccionando productos.

Mostrar historial de ventas con fecha y total.

Base de datos local (SQLite/Room).

Diseño responsivo para pantallas pequeñas.

Preguntas Funcionales:

¿Qué información mínima necesita un producto para ser registrado?

¿Qué pasa si un producto se queda sin stock?

¿Cómo se registra una venta? ¿Hay un historial?

¿La aplicación permite deshacer una venta o corregirla?

¿Qué opciones tiene el usuario si quiere eliminar varios productos a la vez?

¿Cómo se muestran los productos en la interfaz? ¿Hay filtros o buscadores?

Preguntas Reflexivas / de Evaluación

¿Qué fue lo más difícil de desarrollar en este proyecto?

¿Cómo mejoraría la experiencia de usuario (UX)?

Si tuvieras más tiempo o recursos, ¿qué funcionalidades agregarías?

¿Qué aprendiste a nivel técnico con este proyecto?

¿Qué feedback recibiste de posibles usuarios o testers?

Como implementarlo en una Base de datos

¿Cómo implementarlo en Android?

Opción 1: Usar SQLite directamente

- Usa SQLiteOpenHelper.
- Define tu clase DBHelper que cree y actualice la base de datos.
- Accede mediante SQLiteDatabase.

Opción 2: Usar Room (Jetpack)

- Define tus entidades con anotaciones @Entity.
- Usa @Dao para definir las operaciones (insertar, leer, borrar).
- Crea una clase AppDatabase que extienda RoomDatabase.

¿Qué incluye esta base?

- 5 entidades (Usuarios, Productos, Carrito, Pedidos, DetallePedido)
- DAOs con operaciones básicas
- Clase AppDatabase para conectar Room

Paso 1: Agregar dependencias a tu build.gradle

kotlin

CopiarEditar

dependencies {

 def room_version = "2.6.1" // Asegúrate de tener esta u otra versión actual

```
implementation "androidx.room:room-runtime:$room_version"
kapt "androidx.room:room-compiler:$room_version"
```

```
// Opcional: Coroutines
implementation "androidx.room:room-ktx:$room_version"
}
```

Y en el build.gradle (Project) asegúrate de tener kapt habilitado:

kotlin

CopiarEditar

apply plugin: 'kotlin-kapt'

Paso 2: Crear las Entidades

◆ Usuario.kt

kotlin

CopiarEditar

```
import androidx.room.Entity
import androidx.room.PrimaryKey
```

```
@Entity(tableName = "Usuarios")
```

data class Usuario(

```
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val nombre: String,
    val correo: String,
    val contrasena: String
```

)

◆ Producto.kt

kotlin

CopiarEditar

```
import androidx.room.Entity
```

```
import androidx.room.PrimaryKey
```

```
@Entity(tableName = "Productos")
```

data class Producto(

```
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val nombre: String,
    val descripcion: String?,
    val precio: Double,
```

```
    val stock: Int,  
    val imagen_url: String?  
)
```

◆ **Carrito.kt**

```
kotlin  
CopiarEditar  
import androidx.room.Entity  
import androidx.room.PrimaryKey
```

```
@Entity(tableName = "Carrito")  
data class Carrito(  
    @PrimaryKey(autoGenerate = true) val id: Int = 0,  
    val usuario_id: Int,  
    val producto_id: Int,  
    val cantidad: Int  
)
```

◆ **Pedido.kt**

```
kotlin  
CopiarEditar  
import androidx.room.Entity  
import androidx.room.PrimaryKey
```

```
@Entity(tableName = "Pedidos")  
data class Pedido(  
    @PrimaryKey(autoGenerate = true) val id: Int = 0,  
    val usuario_id: Int,  
    val fecha: String,  
    val total: Double,  
    val estado: String  
)
```

◆ **DetallePedido.kt**

```
kotlin  
CopiarEditar  
import androidx.room.Entity  
import androidx.room.PrimaryKey
```

```
@Entity(tableName = "DetallePedido")  
data class DetallePedido(
```

```
@PrimaryKey(autoGenerate = true) val id: Int = 0,  
val pedido_id: Int,  
val producto_id: Int,  
val cantidad: Int,  
val precio_unitario: Double  
)
```

■ Paso 3: Crear los DAOs

◆ UsuarioDao.kt

kotlin

CopiarEditar

```
import androidx.room.*
```

```
@Dao  
interface UsuarioDao {  
    @Insert  
    suspend fun insertar(usuario: Usuario)  
  
    @Query("SELECT * FROM Usuarios WHERE correo = :correo AND contrasena =  
:contrasena")  
    suspend fun login(correo: String, contrasena: String): Usuario?  
  
    @Query("SELECT * FROM Usuarios")  
    suspend fun obtenerTodos(): List<Usuario>  
}  
(¿Quieres que cree los otros DAOs también? Puedo darte todos los necesarios).
```

❖ Paso 4: Crear AppDatabase.kt

```
import androidx.room.Database  
import androidx.room.RoomDatabase
```

```
@Database(  
    entities = [Usuario::class, Producto::class, Carrito::class, Pedido::class,  
    DetallePedido::class],  
    version = 1  
)  
abstract class AppDatabase : RoomDatabase() {  
    abstract fun usuarioDao(): UsuarioDao
```

```
// Puedes agregar los demás DAOs aquí también
}
```

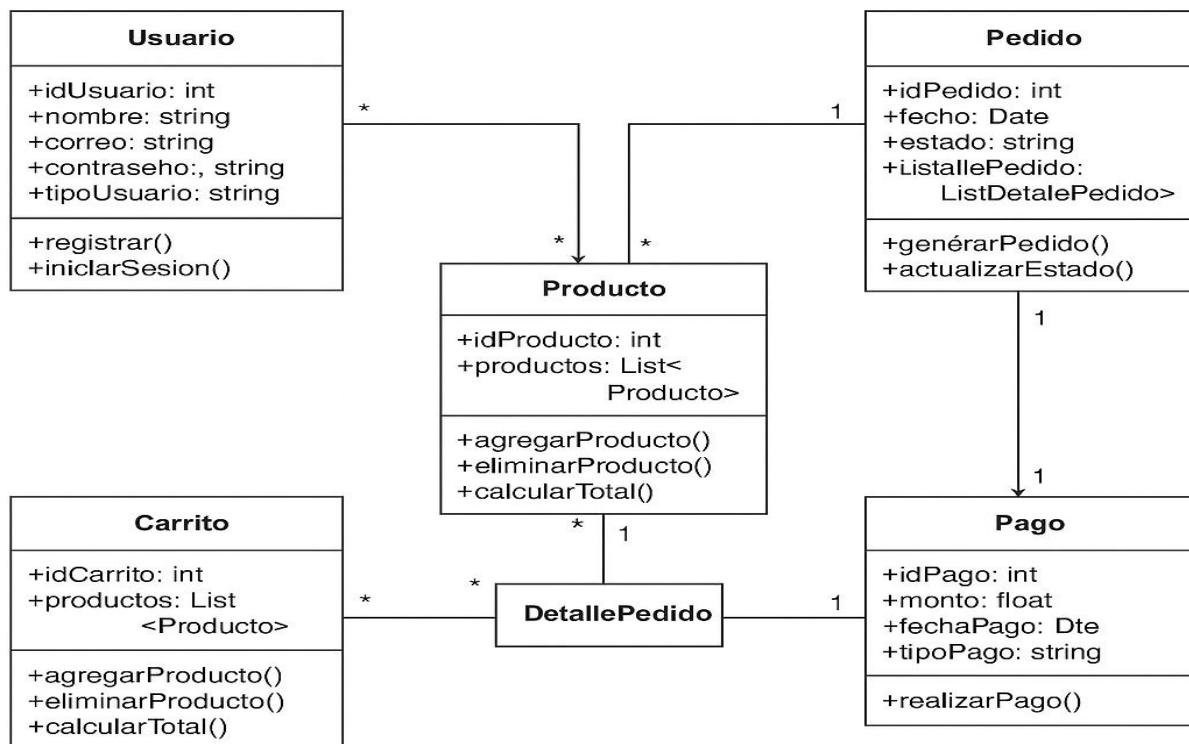
Paso 5: Inicializar Room en tu App

En tu MainActivity.kt o clase Application:

kotlin

```
val db = Room.databaseBuilder(
    applicationContext,
    AppDatabase::class.java, "ventas-db"
).build()
```

DIAGRAMA UML CON LUCIDCHART:



6. Ventajas y Desventajas

✓ Ventajas:

Accesible sin conexión a internet.

Fácil de usar, pensado para usuarios no técnicos.

Gratis y de código abierto.

Permite registrar ventas e inventario fácilmente.

✗ Desventajas:

No sincroniza datos en la nube.

Limitado a un solo dispositivo.

No tiene autenticación de usuarios ni múltiples roles.

7. Posibles Mejoras Futuras

Integración con Firebase para respaldo en la nube.

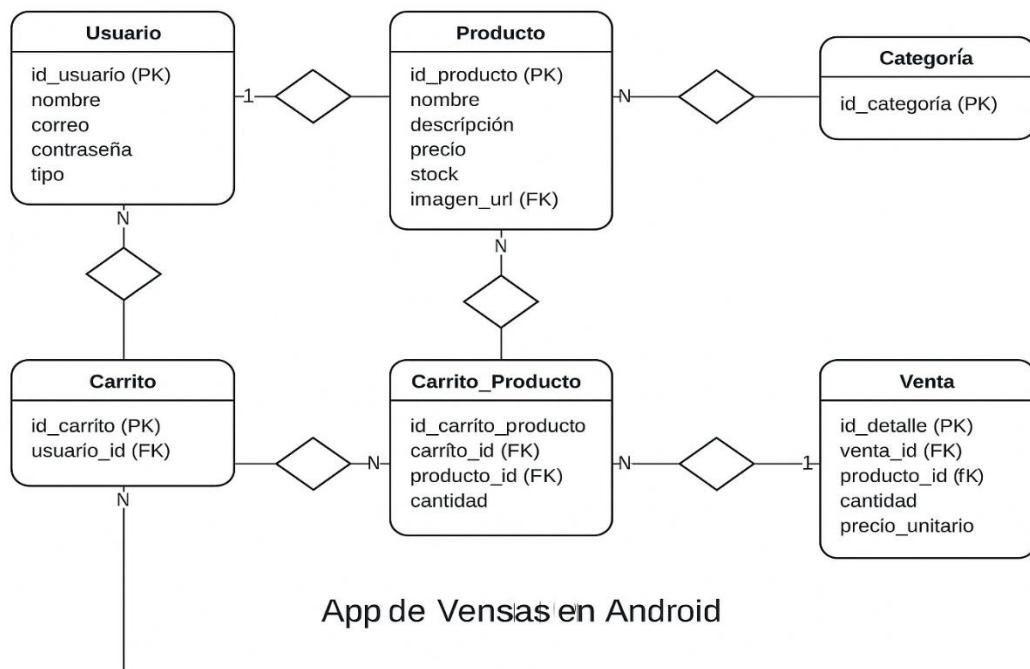
Módulo de reportes (PDF o Excel).

Notificaciones de productos bajos en stock.

Soporte multiusuario y acceso con contraseña.

Integración con código de barras.

Diagrama E-R Entidad Relación de mi App de ventas en Lucidchart



Restricciones del Proyecto: App de Ventas en Android Studio

Técnicas:

1. Solo almacenamiento local (SQLite)

No se usa una base de datos en la nube, por lo que los datos se pierden si el usuario borra la app o cambia de dispositivo.

2. Compatibilidad mínima con Android 5.0 (API 21)

Para asegurar mayor alcance, pero sin usar funciones muy modernas de Android.

3. Sin autenticación de usuarios

Todos los datos están disponibles sin login o roles de acceso, lo que limita seguridad y personalización.

4. Uso de arquitectura sencilla (sin MVVM avanzado)

El proyecto se construye con arquitectura básica para facilitar su comprensión, sacrificando escalabilidad.

5. Sin sincronización entre dispositivos

La app solo puede usarse en un dispositivo a la vez, lo cual limita su uso colaborativo.

Económicas:

1. Sin presupuesto para servicios externos

No se puede integrar pagos en línea, ni usar APIs de terceros de pago.

2. Hardware limitado para pruebas

Se prueba en un número reducido de dispositivos, lo que limita asegurar compatibilidad universal.

Del usuario:

1. Perfil técnico bajo del usuario objetivo

La interfaz debe ser simple y clara, sin funciones complicadas.

2. Solo idioma español

El sistema no está traducido a otros idiomas por simplicidad.

Determinación de Viabilidad:

La determinación de viabilidad en la creación de un sistema de ventas en una app desarrollada en Android Studio implica analizar múltiples aspectos técnicos, económicos, legales y de mercado. Aquí se detalla varios aspectos:

1. Viabilidad Técnica

a) Herramientas y Tecnologías

- **Android Studio:** IDE oficial para el desarrollo.
- **Lenguaje:** Kotlin o Java.
- **Base de datos:** Firebase, SQLite, Room, etc.
- **Backend:** Firebase, Node.js, Laravel, etc.
- **Pasarelas de pago:** Google Pay, Stripe, PayPal, MercadoPago.

b) Compatibilidad

- ¿La app será compatible con múltiples versiones de Android?
- ¿Qué tipo de dispositivos (teléfonos, tablets) se soportarán?

c) Requerimientos técnicos

- Conectividad a Internet
- Seguridad en transacciones (cifrado, tokens, HTTPS)
- Almacenamiento local vs. en la nube

2. Viabilidad Económica

a) Costos

- Desarrollo (personal o equipo)
- Infraestructura (hosting, bases de datos, APIs)
- Licencias (Google Play Developer: \$25 único)
- Costos de integración de pasarelas de pago

b) Retorno de Inversión (ROI)

- ¿Se monetiza por comisión de venta, suscripción, publicidad, etc.?
 - Estimación de usuarios activos y compras mensuales
-

3. Viabilidad Legal

a) Regulación

- Cumplir con la **Ley de Protección de Datos (GDPR o local)**
 - Políticas de reembolso
 - Términos y condiciones
 - Aceptación de medios de pago (requiere cumplimiento legal del país)
-

4. Viabilidad de Mercado

a) Análisis de competencia

- ¿Existen apps similares? ¿Qué ofrecen?
- ¿Qué valor agregado ofrecerás tú?

b) Perfil del usuario

- ¿Quién es tu usuario objetivo?

- ¿Qué dispositivo utiliza? ¿Con qué frecuencia compra?

c) Validación de la idea

- Encuestas, prototipos (mockups) y pruebas de concepto con usuarios reales
-

5. Prototipo y Prueba de Concepto

Antes de desarrollar la app completa:

- Haz un **MVP** (producto mínimo viable)
 - Incluye funciones básicas: catálogo, carrito, pago, historial
 - Prueba con un grupo reducido de usuarios
-

El código fuente base del proyecto con SQLite y pantallas funcionales:

```
// MainActivity.kt

package com.ejemplo.ventasapp

import android.content.Intent
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.google.android.material.floatingactionbutton.FloatingActionButton

class MainActivity : AppCompatActivity() {

    private lateinit var dbHelper: ProductoDbHelper
    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: ProductoAdapter
```

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    dbHelper = ProductoDbHelper(this)  
    recyclerView = findViewById(R.id.recyclerView)  
    recyclerView.layoutManager = LinearLayoutManager(this)  
  
    cargarProductos()  
    findViewById<FloatingActionButton>(R.id.fab).setOnClickListener {  
        startActivity(Intent(this, AddProductoActivity::class.java))  
    }  
}  
  
override fun onResume() {  
    super.onResume()  
    cargarProductos()  
}  
  
private fun cargarProductos() {  
    val productos = dbHelper.obtenerTodos()  
    adapter = ProductoAdapter(productos, dbHelper)  
    recyclerView.adapter = adapter  
}  
}  
  
//  Producto.kt  
  
data class Producto(val id: Int, val nombre: String, val precio: Double, val cantidad: Int)  
  
//  ProductoDbHelper.kt
```

```
package com.ejemplo.ventasapp

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

class ProductoDbHelper(context: Context) : SQLiteOpenHelper(context, "productos.db",
null, 1) {

    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL("""
            CREATE TABLE productos (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nombre TEXT NOT NULL,
                precio REAL NOT NULL,
                cantidad INTEGER NOT NULL
            )
        """.trimIndent())
    }

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

    fun insertar(producto: Producto): Long {
        val db = writableDatabase
        val values = ContentValues().apply {
            put("nombre", producto.nombre)
            put("precio", producto.precio)
        }
    }
}
```

```
        put("cantidad", producto.cantidad)
    }

    return db.insert("productos", null, values)
}

fun obtenerTodos(): List<Producto> {
    val db = readableDatabase
    val cursor = db.rawQuery("SELECT * FROM productos", null)
    val lista = mutableListOf<Producto>()
    with(cursor) {
        while (moveToNext()) {
            val producto = Producto(

```

```
        getInt(getColumnIndexOrThrow("id")),
        getString(getColumnIndexOrThrow("nombre")),
        getDouble(getColumnIndexOrThrow("precio")),
        getInt(getColumnIndexOrThrow("cantidad"))

    )
    lista.add(producto)
}

cursor.close()
return lista
}

fun eliminar(id: Int) {
    val db = writableDatabase
    db.delete("productos", "id=?", arrayOf(id.toString()))
}
}

// ProductoAdapter.kt
package com.ejemplo.ventasapp

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
class ProductoAdapter(private val productos: List<Producto>, private val dbHelper: ProductoDbHelper)
    : RecyclerView.Adapter<ProductoAdapter.ProductoViewHolder>() {

    class ProductoViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        var id: TextView = itemView.findViewById(R.id.textViewId)
        var nombre: TextView = itemView.findViewById(R.id.textViewNombre)
        var precio: TextView = itemView.findViewById(R.id.textViewPrecio)
        var cantidad: TextView = itemView.findViewById(R.id.textViewCantidad)
        var eliminar: Button = itemView.findViewById(R.id.buttonEliminar)
    }
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProductoViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_producto, parent, false)
        return ProductoViewHolder(view)
    }
    override fun onBindViewHolder(holder: ProductoViewHolder, position: Int) {
        holder.id.text = productos[position].id.toString()
        holder.nombre.text = productos[position].nombre
        holder.precio.text = productos[position].precio.toString()
        holder.cantidad.text = productos[position].cantidad.toString()
        holder.eliminar.setOnClickListener { eliminarProducto(holder.adapterPosition) }
    }
    override fun getItemCount(): Int = productos.size
    private fun eliminarProducto(position: Int) {
        dbHelper.eliminar(productos[position].id)
        notifyDataSetChanged()
    }
}
```

```
inner class ProductoViewHolder(view: View): RecyclerView.ViewHolder(view) {  
    val nombre: TextView = view.findViewById(R.id.txtNombre)  
    val precio: TextView = view.findViewById(R.id.txtPrecio)  
    val cantidad: TextView = view.findViewById(R.id.txtCantidad)  
    val btnEliminar: Button = view.findViewById(R.id.btnEliminar)  
}  
  
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):  
    ProductoViewHolder {  
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_producto, parent,  
        false)  
        return ProductoViewHolder(view)  
    }  
  
override fun onBindViewHolder(holder: ProductoViewHolder, position: Int) {  
    val producto = productos[position]  
    holder.nombre.text = producto.nombre  
    holder.precio.text = "$${producto.precio}"  
    holder.cantidad.text = "Stock: ${producto.cantidad}"  
    holder.btnEliminar.setOnClickListener {  
        dbHelper.eliminar(producto.id)  
        notifyItemRemoved(position)  
    }  
}  
  
override fun getItemCount() = productos.size  
}  
// AddProductoActivity.kt  
package com.ejemplo.ventasapp
```

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

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

        val dbHelper = ProductoDbHelper(this)

        val edtNombre = findViewById<EditText>(R.id.edtNombre)
        val edtPrecio = findViewById<EditText>(R.id.edtPrecio)
        val edtCantidad = findViewById<EditText>(R.id.edtCantidad)
        val btnGuardar = findViewById<Button>(R.id.btnGuardar)

        btnGuardar.setOnClickListener {
            val nombre = edtNombre.text.toString()
            val precio = edtPrecio.text.toString().toDoubleOrNull()
            val cantidad = edtCantidad.text.toString().toIntOrNull()

            if (nombre.isNotBlank() && precio != null && cantidad != null) {
                val producto = Producto(0, nombre, precio, cantidad)
                dbHelper.insertar(producto)
                Toast.makeText(this, "Producto guardado", Toast.LENGTH_SHORT).show()
                finish()
            } else {

```

```
        Toast.makeText(this, "Completa todos los campos correctamente",  
        Toast.LENGTH_SHORT).show()  
    }  
}  
}  
}
```

8. Tecnologías Utilizadas

Lenguaje: Kotlin (o Java) IDE: Android Studio
Base de Datos: SQLite (con Room)
UI: XML
Arquitectura: MVVM (opcional)

Recomendaciones Éticas y Legales

1. Incluir una política de privacidad en la app (aunque sea sencilla).
2. Incluir créditos o atribuciones de recursos externos (íconos, fuentes, librerías).
3. Evitar el uso de datos sensibles sin consentimiento.
4. Documentar adecuadamente las funciones que afectan datos.
5. Considerar una licencia de distribución clara (ej. MIT, Creative Commons, etc.).

9. Conclusión

Esta aplicación de ventas ofrece una solución práctica para pequeños comerciantes que buscan una herramienta móvil simple para llevar el control de sus productos y transacciones. Aunque básica en su primera versión, su diseño modular permite ampliarla en el futuro con más funcionalidades como respaldo en la nube y múltiples usuarios. Es un primer paso hacia la transformación digital de microempresas.

Elaboro: Isaias Guadarrama Cristiani

Alumno de Ingeniería en Sistemas Computacionales