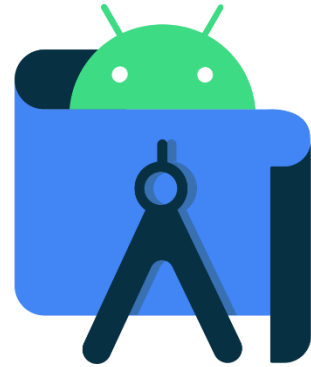




Proyecto de programación ANDROID STUDIO con SQLITE
Y Gestor DBBrowser

android studio



Gestión de ventas en restaurante



Autor: Carbajal Cusi Cristhian Gerardo



Introducción	4
Marco Teórico	4
Gestor de base de datos.....	4
Sqlite.....	4
DbBrowser.....	4
AndroidStudio	4
Aplicaciones Moviles	4
Desarrollo Proyecto.....	5
Requerimientos funcionales	5
Requerimientos no funcionales	5
Instalación de software	6
SQLite:	6
DB Browser:.....	10
Android Studio:	13
Arquitectura de la aplicación	16
Desarrollo del Proyecto.....	17
Creación de tablas en SQLite.....	17
Insertando registros en SQLite	19
Creación de un nuevo proyecto en Android studio	21
Creación de conexión SQLite en Android Studio	23
DatabaseHelper	23
ControladorDatabase	25
Codificando ACTIVITY LAYOUTS [CODE] [DESIGN] en Android Studio	27
Funcionamiento de la Clase activity layout login	27
Funcionamiento de la Clase activity layout menú_general	29
Funcionamiento de la Clase activity layout pedido	30
Funcionamiento de la Clase activity layout ResultadoBoleta	32
Considerando funcionalidades finales en Android Studio	33
Pruebas y depuración.....	33
Implementación y lanzamiento.....	33
Seguimiento y mantenimiento.....	33





Introducción

Desarrollaré una aplicación móvil en Android Studio que servirá como gestor de ventas abarrotes para la app 'mercado Rapificiente'. Utilizaré SQLite como base de datos para almacenar la información de los productos, y el Gestor DB Browser para administrar la base de datos. La aplicación permitirá registrar, buscar y actualizar detalles de los productos, generar informes de ventas y gestionar proveedores. El objetivo es mejorar la eficiencia y rentabilidad de las tiendas de abarrotes, simplificando la gestión del inventario y brindando un control preciso sobre los productos disponibles.

Marco Teórico

Gestor de base de datos

Gestor de base de datos: Un gestor de base de datos es un software que se utiliza para administrar y organizar datos. En el contexto de una aplicación en Android, un gestor de base de datos es necesario para almacenar y gestionar los datos de la aplicación, como la información de los usuarios, configuraciones, registros, etc. Algunos ejemplos comunes de gestores de bases de datos utilizados en Android son SQLite, MySQL

Sqlite

SQLite: SQLite es un gestor de base de datos relacional de código abierto ampliamente utilizado en aplicaciones móviles, incluidas las aplicaciones en Android. Es una base de datos liviana y autónoma que se almacena localmente en el dispositivo del usuario. SQLite es una opción popular para aplicaciones móviles debido a su eficiencia, facilidad de uso y capacidad para manejar grandes volúmenes de datos. L y Firebase Realtime Database.

DbBrowser

DB Browser: DB Browser (también conocido como DB Browser for SQLite) es una herramienta de software que permite visualizar, editar y administrar bases de datos SQLite de manera intuitiva. Proporciona una interfaz gráfica fácil de usar que facilita la exploración de los datos almacenados en una base de datos SQLite, así como la ejecución de consultas SQL. DB Browser es especialmente útil durante el desarrollo de aplicaciones en Android, ya que permite inspeccionar y manipular los datos almacenados en la base de datos SQLite de la aplicación, lo que facilita el proceso de depuración y verificación de los datos.

AndroidStudio

Android Studio: Android Studio es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones en Android. Proporciona un conjunto de herramientas y funciones que facilitan el desarrollo de aplicaciones, incluida la codificación, depuración, diseño de interfaces, pruebas y empaquetado de la aplicación. Android Studio se basa en IntelliJ IDEA y es compatible con los lenguajes de programación Java y Kotlin.

Aplicaciones Moviles

Aplicaciones Móviles: Las aplicaciones móviles son programas diseñados específicamente para ser ejecutados en dispositivos móviles, como teléfonos inteligentes y tabletas. Estas aplicaciones están destinadas a brindar funcionalidades y servicios que satisfagan las necesidades de los usuarios en movimiento. Pueden ser de diversos tipos, como aplicaciones de redes sociales, juegos, productividad, comercio electrónico, entre otros. El desarrollo de aplicaciones móviles



ha experimentado un crecimiento significativo debido a la creciente popularidad de los dispositivos móviles en todo el mundo.

Desarrollo Proyecto

Definición de requerimientos: En esta etapa, debes identificar y documentar los requerimientos y funcionalidades clave de tu aplicación. Determina qué características debe tener, cómo debe funcionar y qué público objetivo quieres alcanzar.

Requerimientos funcionales

ID	Requerimiento	Descripción
RF1	Registro de Clientes	Los usuarios deberían poder registrar nuevos clientes en la tabla "Clientes" con su nombre, apellidos y DNI.
RF2	Registro de Órdenes	Los usuarios deberían poder registrar nuevas órdenes en la tabla "Ordenes" con un número de orden único, una fecha y hora, y relacionar la orden con un cliente existente a través de su ClientelD.
RF3	Registro de Consumos	Los usuarios deberían poder agregar nuevos elementos de consumo en la tabla "Consumo" con nombre, tipo, descripción y precio.
RF4	Agregar Detalles a la Orden	Los usuarios deberían poder agregar detalles a una orden existente en la tabla "Detalles de la Orden". Esto implicaría vincular una orden a elementos de consumo y especificar la cantidad de cada elemento.
RF5	Visualización de Órdenes por Cliente	Los usuarios deberían poder ver todas las órdenes realizadas por un cliente específico, identificado por su ClientelD.
RF6	Calcular el Total de una Orden	El sistema debería calcular automáticamente el total de una orden sumando los precios de todos los elementos de consumo incluidos en los detalles de la orden.
RF7	Generación de Boleta	El sistema debería ser capaz de generar boletas automáticamente a partir de una orden registrada en la tabla "Ordenes". La boleta debe incluir detalles como el nombre del cliente, la fecha y hora de la orden, los elementos de consumo con sus cantidades y precios individuales, el total a pagar y cualquier impuesto aplicable. Las boletas deben ser fácilmente accesibles e imprimibles para proporcionar a los clientes como comprobante de su compra.

Requerimientos no funcionales

ID	Requerimiento	Descripción
RNF1	Usabilidad	La aplicación debe ser fácil de usar y tener una interfaz intuitiva.
RNF2	Rendimiento	La aplicación debe ser rápida y eficiente en la búsqueda y actualización de datos.
RNF3	Seguridad	Se deben implementar medidas de seguridad para proteger los datos de la aplicación.
RNF4	Disponibilidad	La aplicación debe estar disponible en todo momento, con un tiempo de inactividad mínimo.
RNF5	Escalabilidad	La aplicación debe ser escalable para manejar un crecimiento en la cantidad de productos y usuarios.
RNF6	Portabilidad	La aplicación debe ser compatible con diferentes dispositivos móviles con sistema operativo Android.
RNF7	Mantenibilidad	La aplicación debe ser fácil de mantener y permitir futuras actualizaciones.



Instalación de software

SQLite:

1. Visita el sitio web oficial de SQLite en <https://www.sqlite.org/index.html>.

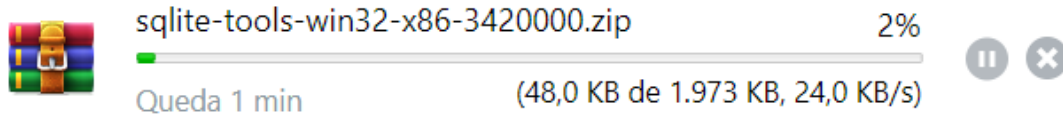
The screenshot shows the SQLite website. The 'What Is SQLite?' section describes it as a C-language library. The 'Latest Release' section shows version 3.42.0 (2023-05-16) with a 'Download' button highlighted by a red arrow.

2. Navega hasta la sección de descargas y elige la versión adecuada para tu sistema operativo (Windows, macOS, Linux, etc.).

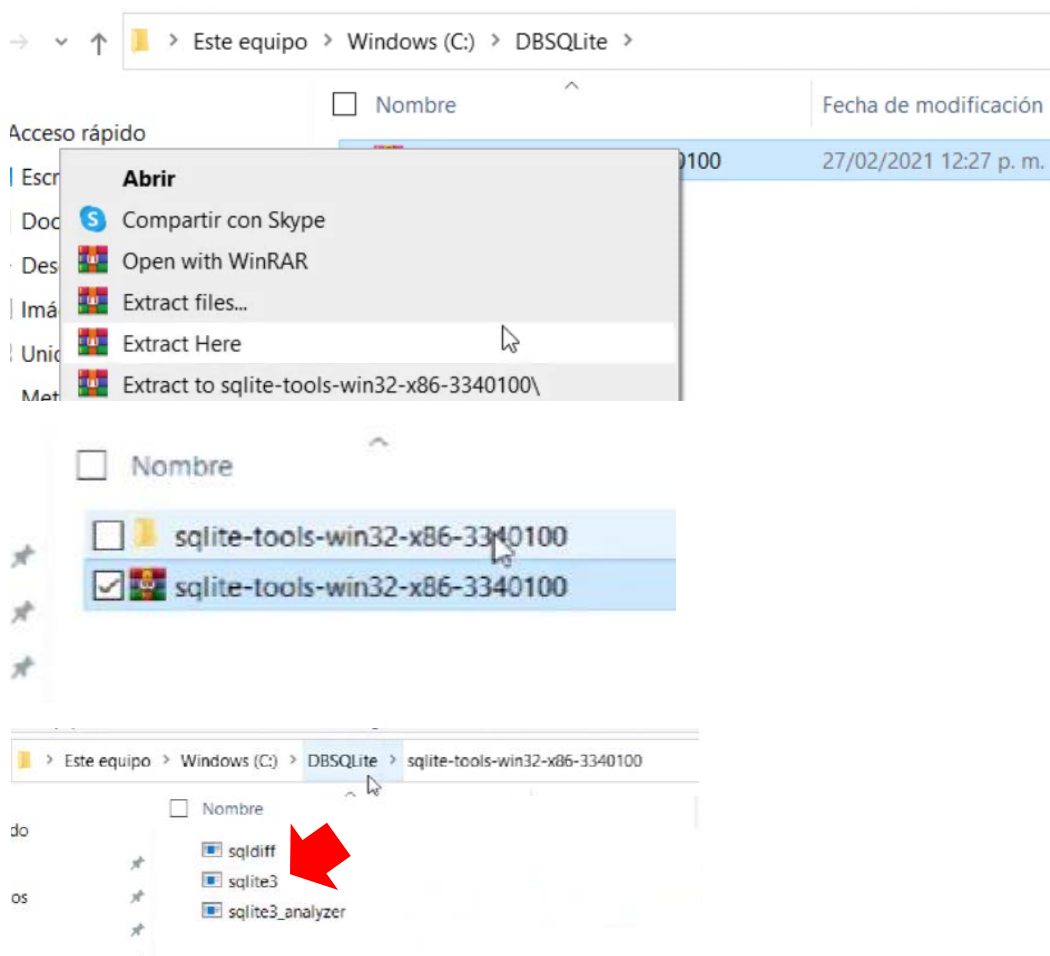
The screenshot shows the 'Precompiled Binaries for Windows' section. It lists two options: a 32-bit DLL (x86) and a 64-bit DLL (x64). A red arrow points to the 'sqlite-tools-win32-x86-3420000.zip' file, which is highlighted with a red box.



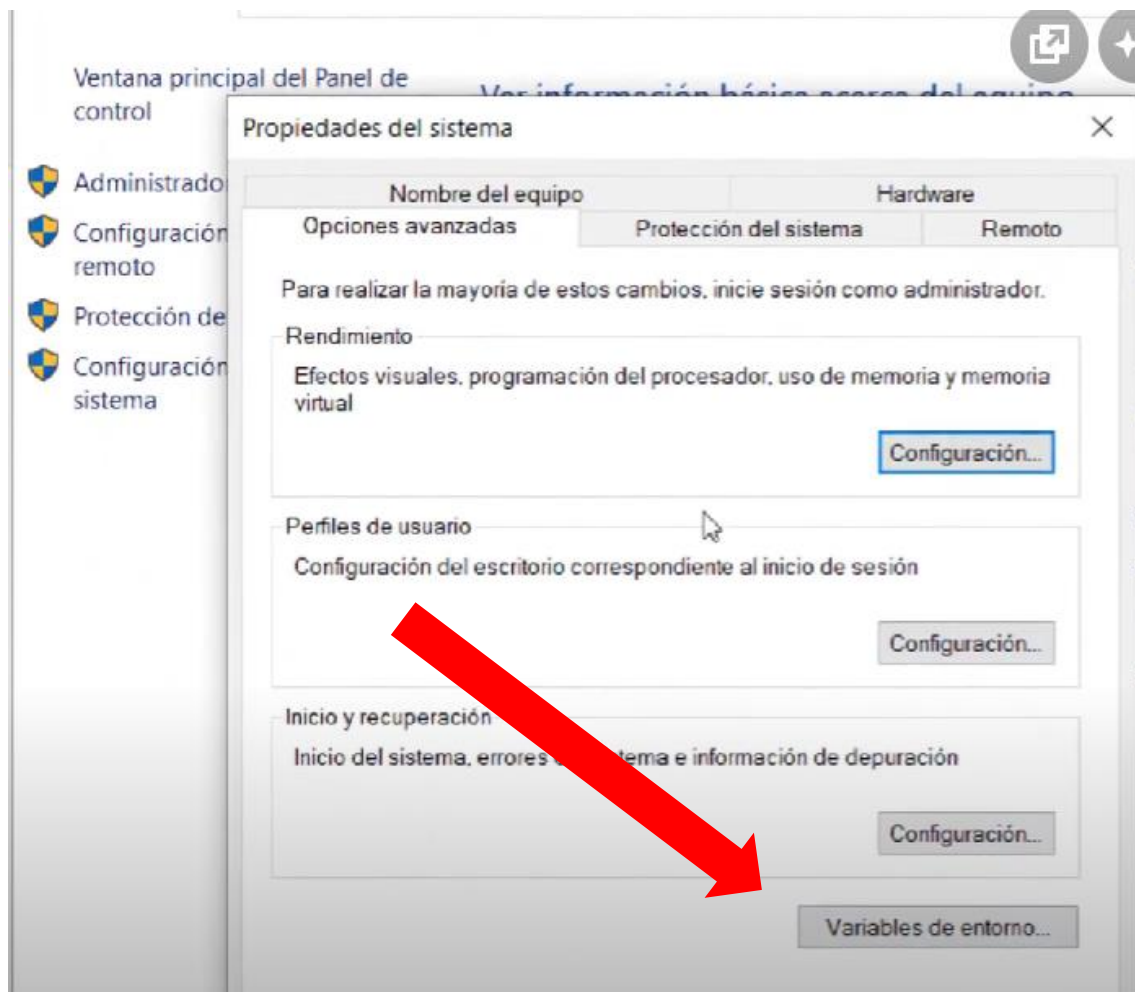
3. Descarga el archivo de instalación correspondiente.



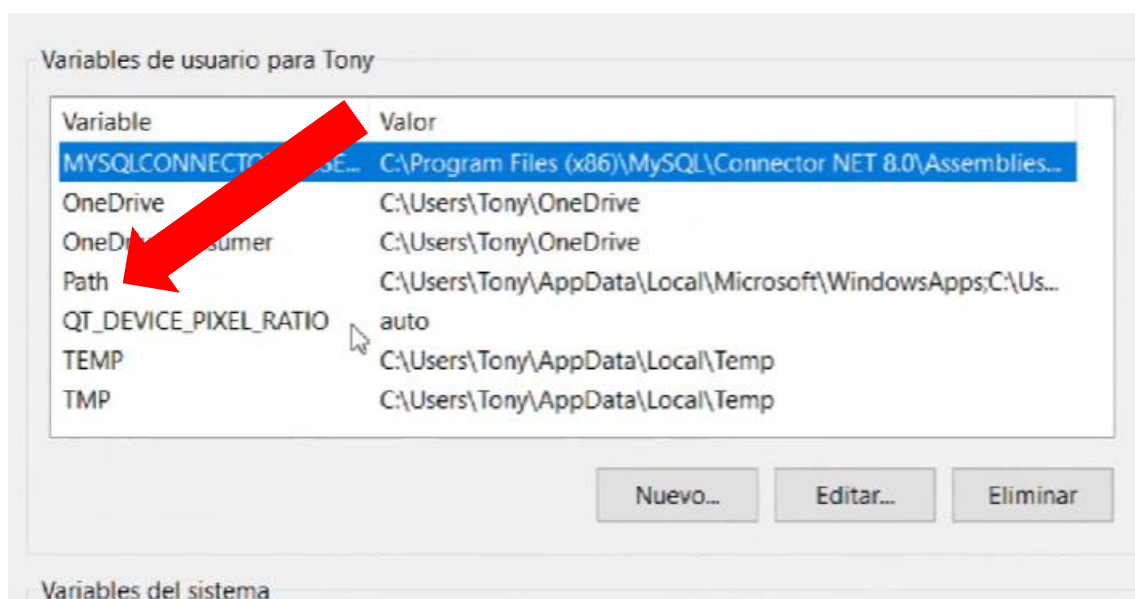
4. Ejecuta el archivo de instalación y sigue las instrucciones del asistente de instalación.



Agregar la aplicación de sqlite enlazado al path del sistema para que se pueda ejecutar desde consola.

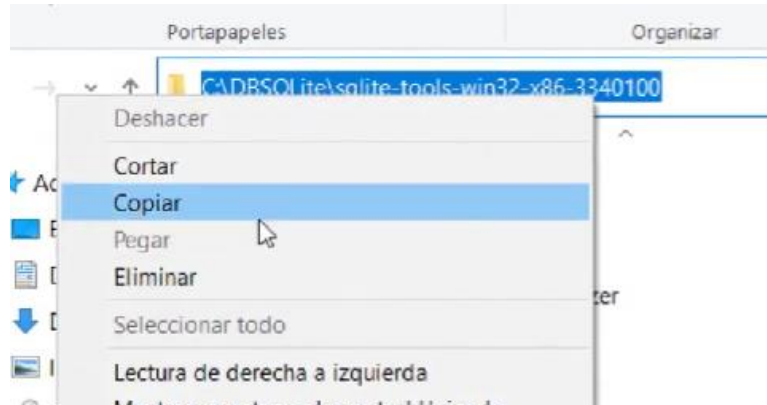


Le damos clic en PATH

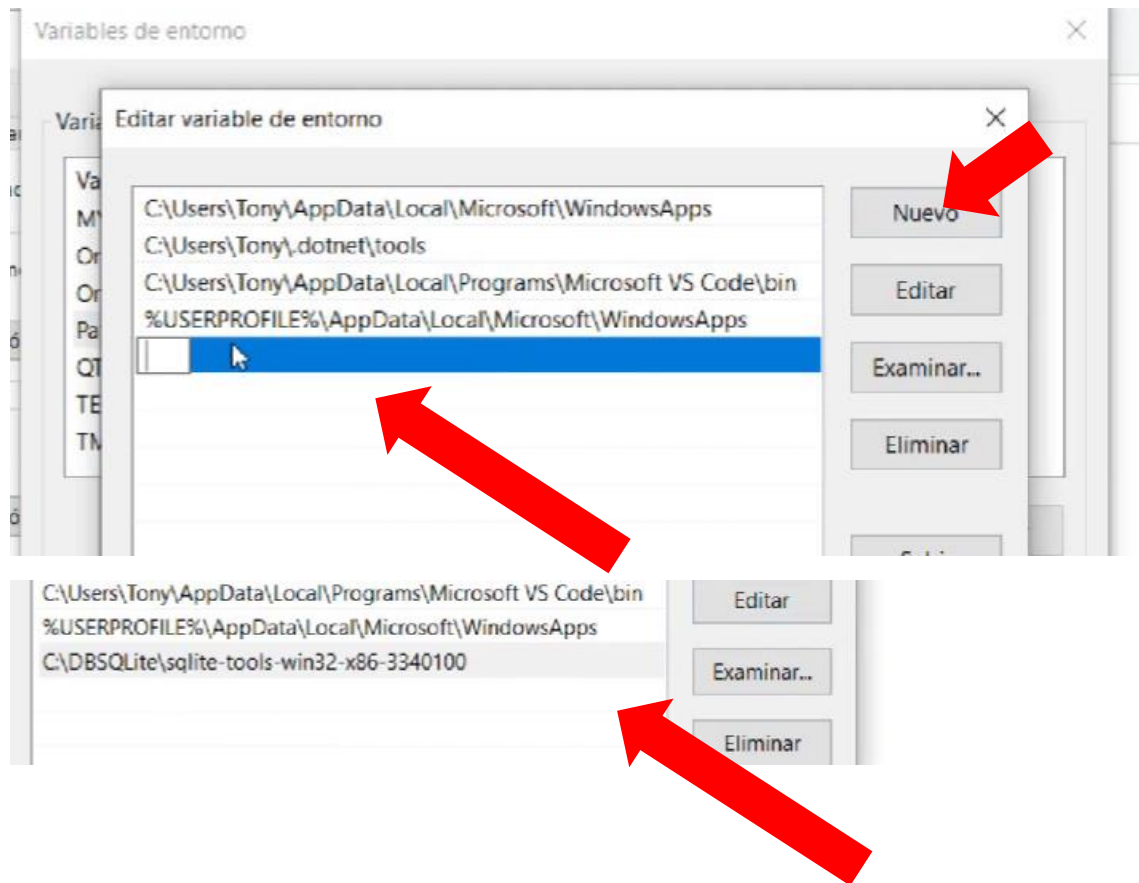




Copiamos la URL donde se ubica nuestro SQLite



Pegamos la URL en el PATH





En la consola de CMD (Windows + R) , colocamos ya directamente la ejecución de SQLITE3

```
Seleccionar C:\Windows\system32\cmd.exe - SQLITE3
Microsoft Windows [Versión 10.0.19045.3086]
(c) Microsoft Corporation. Todos los derechos reservados.

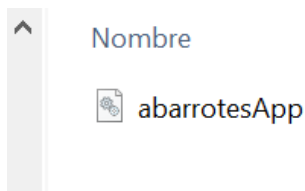
C:\Users\KRISTHIAN>SQLITE3
SQLite version 3.38.5 2022-05-06 15:25:27
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Creamos la base de datos desde consola **abarrotasApp.db**

```
Seleccionar C:\Windows\system32\cmd.exe - sqlite3 abarrotasApp.db
Microsoft Windows [Versión 10.0.19045.3086]
(c) Microsoft Corporation. Todos los derechos reservados.

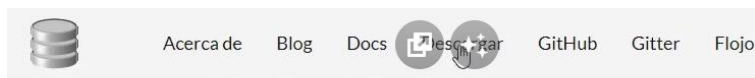
C:\Users\KRISTHIAN>CD DESKTOP
C:\Users\KRISTHIAN\Desktop>mkdir Proyecto-Android-Practicas-I
C:\Users\KRISTHIAN\Desktop>cd Proyecto-Android-Practicas-I
C:\Users\KRISTHIAN\Desktop\Proyecto-Android-Practicas-I>sqlite3 abarrotasApp.db
SQLite version 3.38.5 2022-05-06 15:25:27
Enter ".help" for usage hints.
sqlite> .databases
main: C:\Users\KRISTHIAN\Desktop\Proyecto-Android-Practicas-I\abarrotasApp.db r/w
sqlite>
```

5. Tenemos lista nuestra base de datos creada, ahora la podremos gestionar con DBBrowser for Sqlite.



DB Browser:

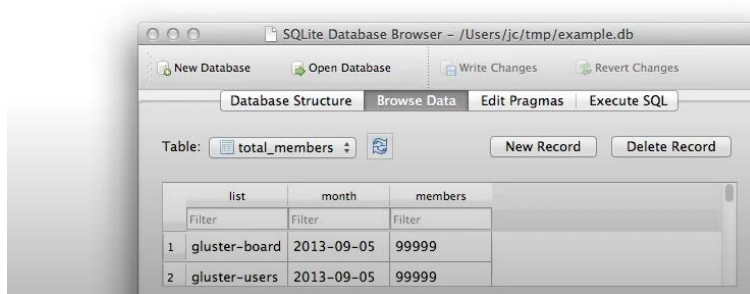
Accede al sitio web oficial de DB Browser en <https://sqlitebrowser.org/>.



DB Browser para SQLite

La página oficial del navegador DB para SQLite

Captura de pantalla



Dirígete a la sección de descargas y elige la versión compatible con tu sistema operativo.

Windows

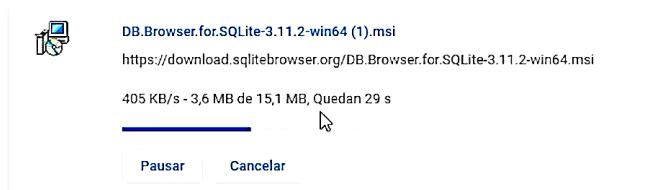


Nuestra última versión (3.11.2) para Windows:

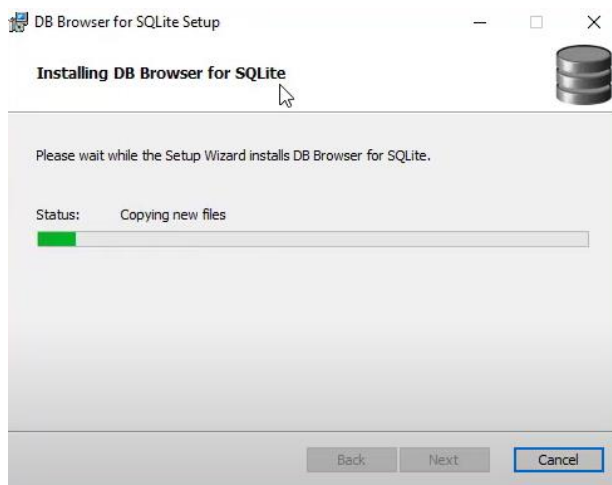
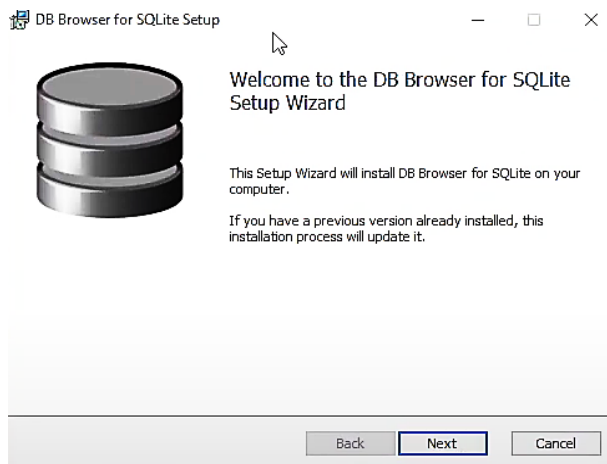
- [DB Browser for SQLite: instalador estándar para Windows de 32 bit y Windows XP](#)
- [DB Browser para SQLite: .zip \(sin instalador\) para Windows de 32 bits y Windows XP](#)
- [DB Browser for SQLite - Instalador estándar para Windows de 64 bits](#)
- [DB Browser para SQLite - .zip \(sin instalador\) para Windows de 64 bits](#)
- Nota: no hay una versión de PortableApp para 3.11.1 (todavía). Esperemos que esté listo en unos días.



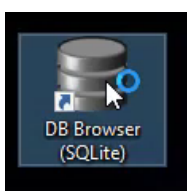
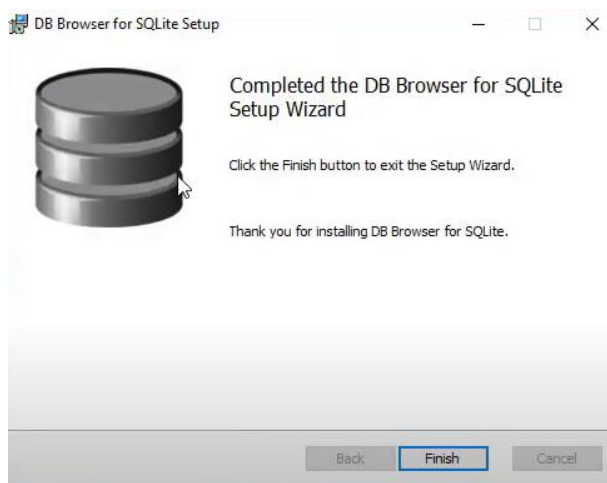
Descarga el archivo de instalación adecuado.



Ejecuta el archivo de instalación y sigue las instrucciones del asistente.



Una vez finalizada la instalación, podrás abrir DB Browser y utilizarlo para abrir y administrar bases de datos SQLite.





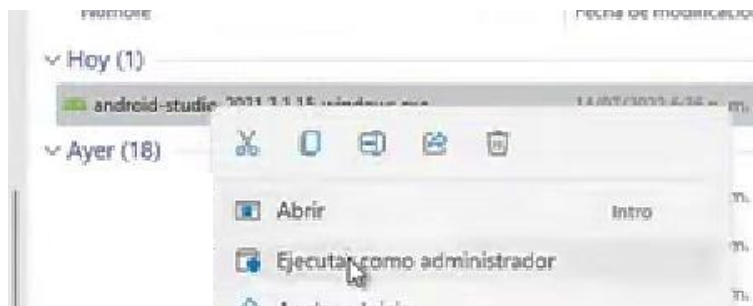
Android Studio:

Ve al sitio web oficial de Android Studio en <https://developer.android.com/studio>.

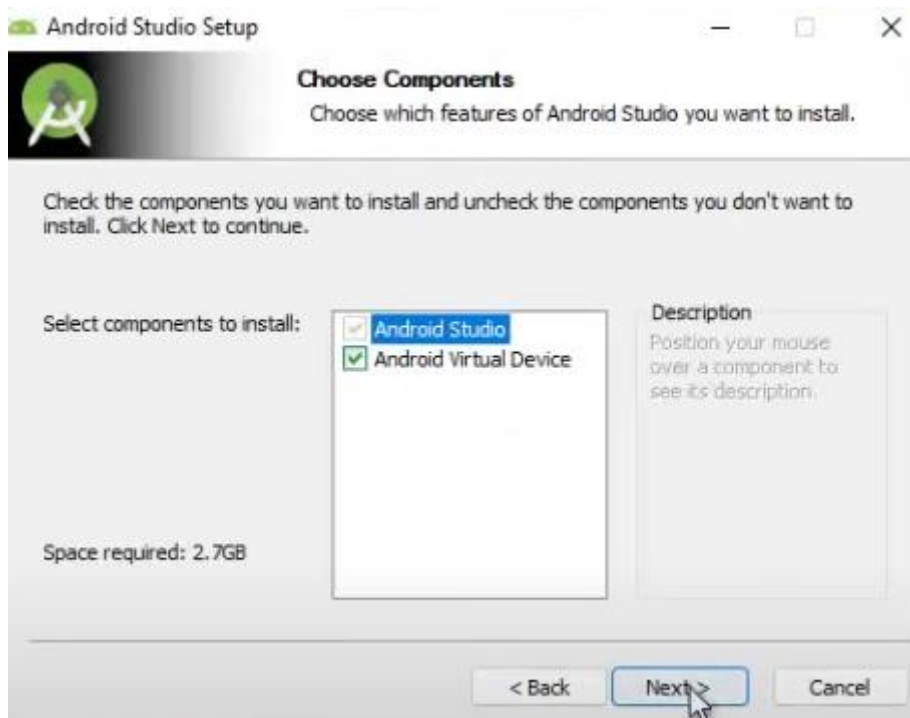
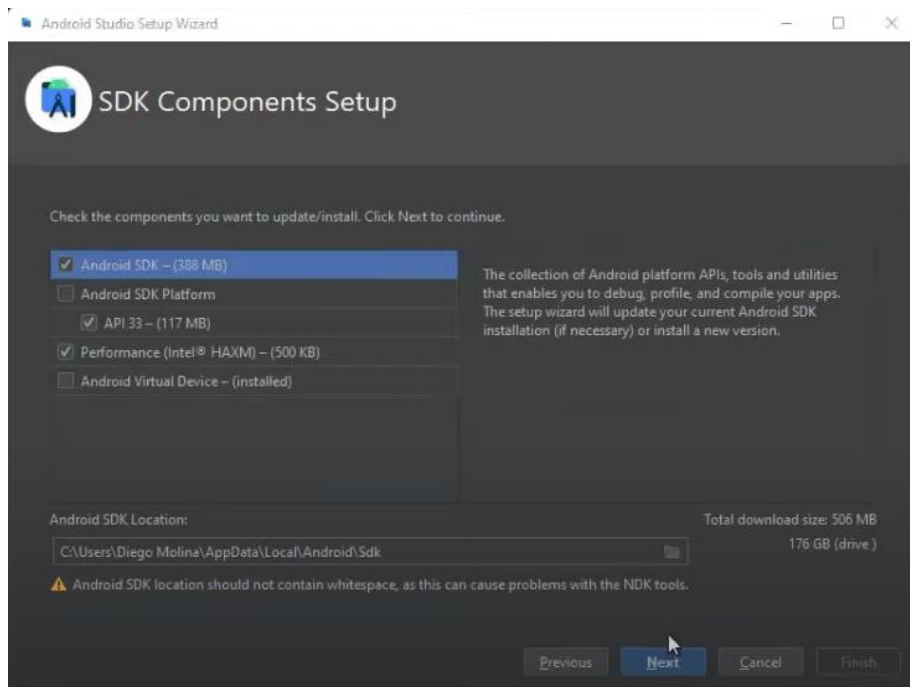


Haz clic en el botón "Download" para descargar el instalador de Android Studio.

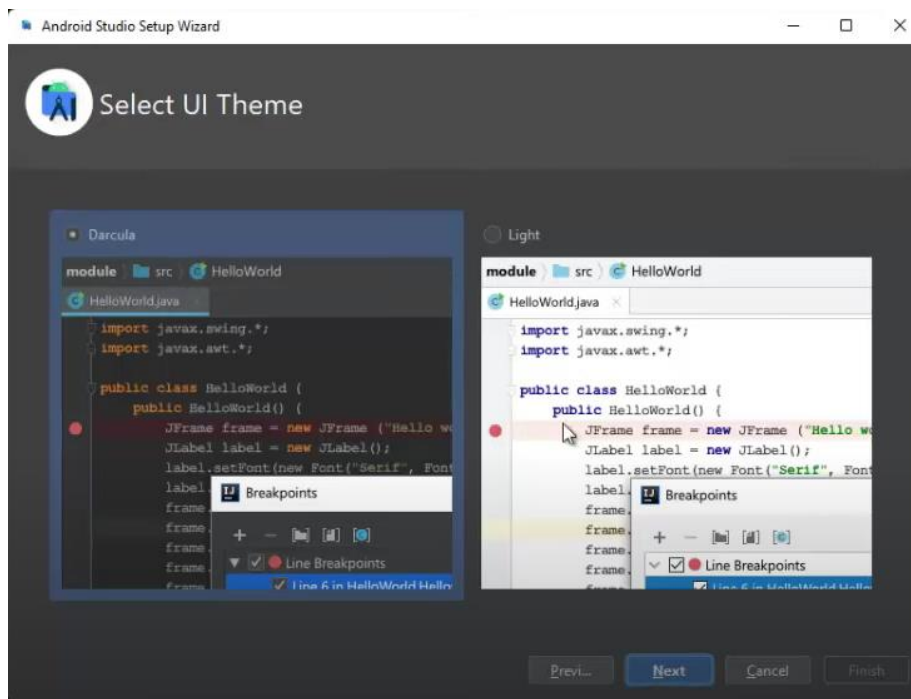
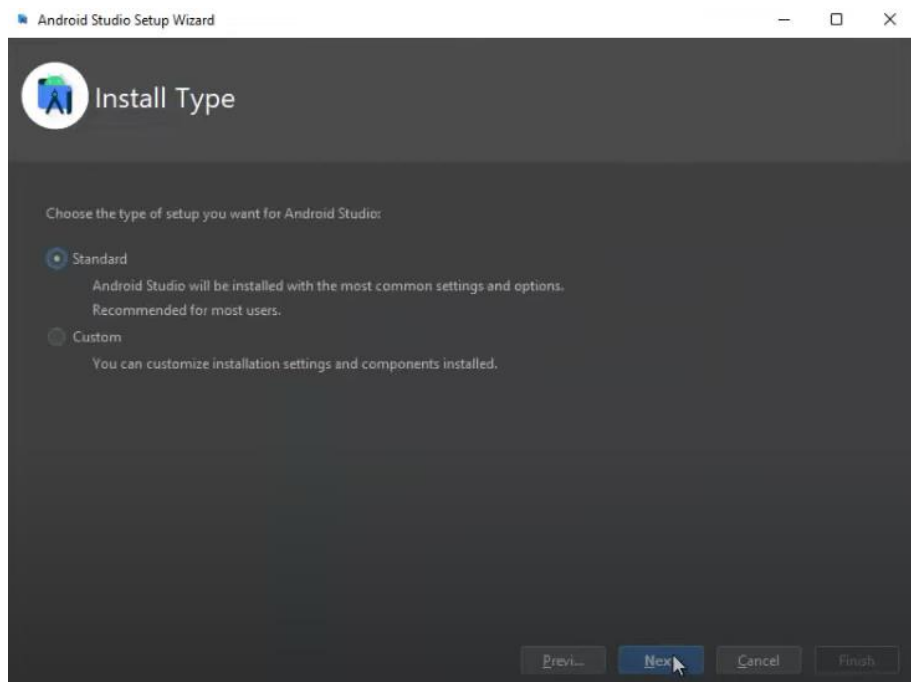
Ejecuta el archivo de instalación descargado y sigue las instrucciones del asistente de instalación.



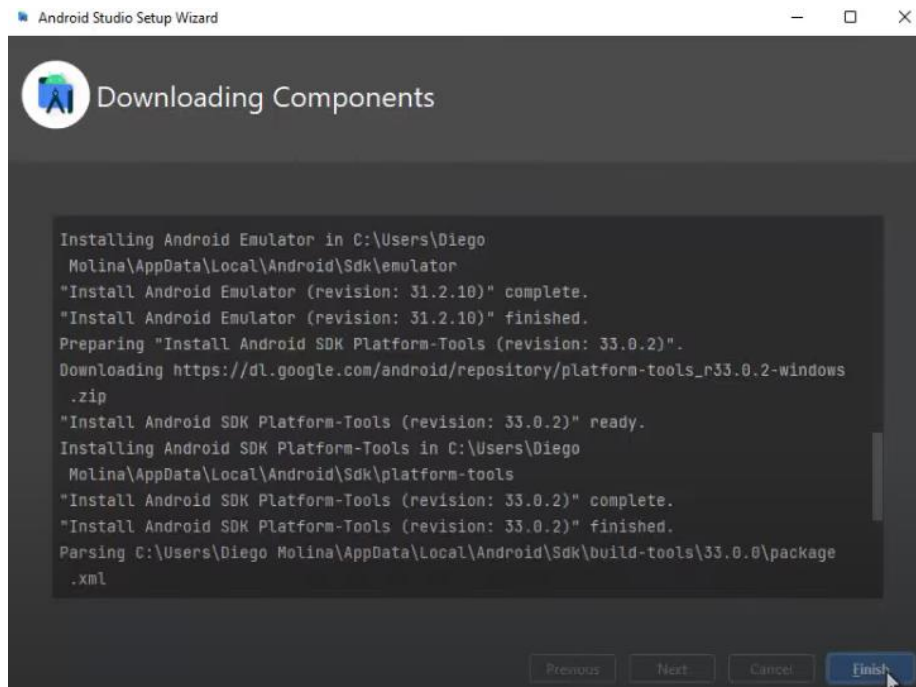
Durante el proceso de instalación, se te pedirá que elijas los componentes que deseas instalar. Asegúrate de seleccionar "Android SDK" y "Android Virtual Device".



Sigue los pasos de configuración inicial para instalar los componentes necesarios y configurar tu entorno de desarrollo.

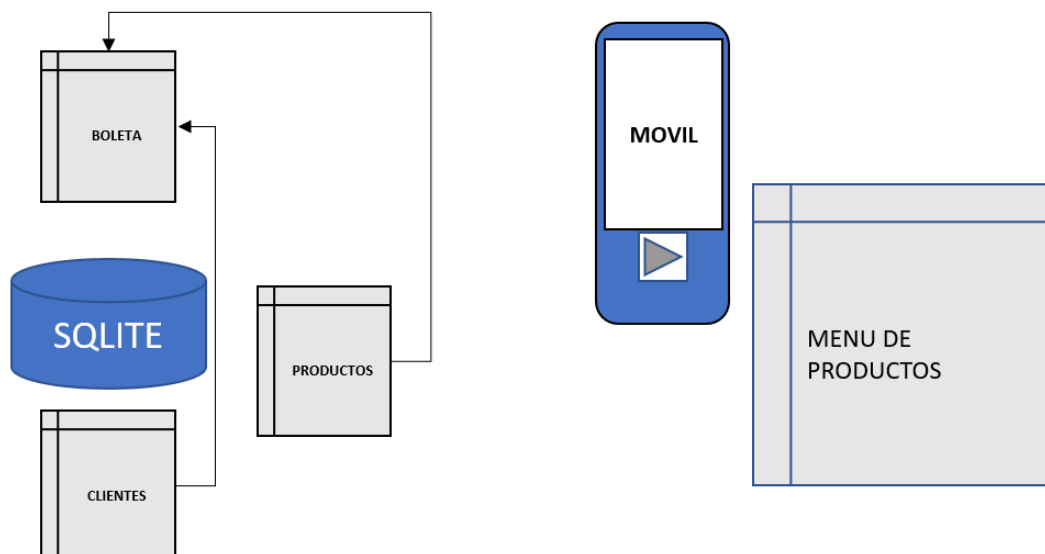


Finalizamos y estará listo para comenzar a usar Android Studio y generar nuevos proyectos.



Arquitectura de la aplicación: Aquí se realiza el diseño de la interfaz de usuario (UI) y la arquitectura de la aplicación. Define los diferentes componentes de la interfaz y cómo se relacionan entre sí. Además, considera los aspectos de usabilidad y experiencia de usuario.

Diseño de la aplicación Android con SQLite:





Desarrollo del Proyecto: Durante esta fase, se lleva a cabo la codificación de la aplicación. Seleccionamos las herramientas y tecnologías adecuadas para el desarrollo de la aplicación en Android, como el lenguaje de programación Java, y utilizamos el entorno de desarrollo integrado (IDE) de Android, como Android Studio.

*** Para nuestra aplicación de restaurante hemos escogido el lenguaje de programación JAVA.**

Creación de tablas en SQLite

Para que cumpla con el procedimiento de nuestra aplicación hemos creado 5 tablas:

1. Login_Principal
2. Clientes
3. Ordenes
4. Consumo
5. Detalles_de_la_Orden

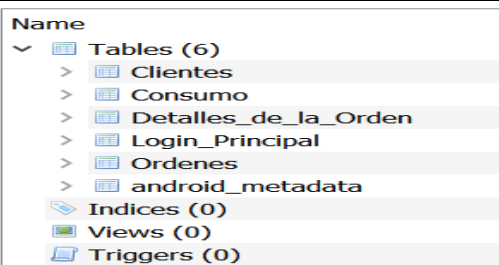
<p>Tabla "Clientes":</p> <p>ClienteID (clave primaria) nombre apellidos DNI</p>	<p>Tabla "Ordenes":</p> <p>OrdenID (clave primaria) NumeroDeOrden FechaHora ClienteID (clave foránea que se relaciona con la tabla "Clientes")</p>
<p>Tabla "Consumo":</p> <p>ConsumoID (clave primaria) nombre tipo descripcion precio</p>	<p>Tabla "Detalles de la Orden":</p> <p>DetalleID (clave primaria) OrdenID (clave foránea que se relaciona con la tabla "Órdenes") ConsumoID (clave foránea que se relaciona con la tabla "Consumo") Cantidad</p>
<p>Tabla "Login":</p> <p>loginID nombre contraseña</p> <p>El 'android_metadata' no se considera parte del proyecto, es una tabla de soporte que se crea en sqlite3 automáticamente.</p>	



Tabla "Clientes"

```
CREATE TABLE Clientes (  
    ClienteID INTEGER PRIMARY KEY,  
    nombre TEXT,  
    apellidos TEXT,  
    DNI TEXT  
);
```

-- Tabla "Ordenes"

```
CREATE TABLE Ordenes (  
    OrdenID INTEGER PRIMARY KEY,  
    NumeroDeOrden TEXT,  
    FechaHora DATETIME,  
    ClienteID INTEGER,  
    FOREIGN KEY (ClienteID) REFERENCES Clientes(ClienteID)  
);
```

-- Tabla "Consumo"

```
CREATE TABLE Consumo (  
    ConsumoID INTEGER PRIMARY KEY,  
    nombre TEXT,  
    tipo TEXT,  
    descripcion TEXT,  
    precio float  
);
```

-- Tabla "Detalles de la Orden"

```
CREATE TABLE Detalles_de_la_Orden (  
    DetalleID INTEGER PRIMARY KEY,  
    OrdenID INTEGER,  
    ConsumoID INTEGER,
```



```
Cantidad INTEGER,  
FOREIGN KEY (OrdenID) REFERENCES Ordenes(OrdenID),  
FOREIGN KEY (ConsumoID) REFERENCES Consumo(ConsumoID)  
);
```

```
-- Tabla "Login"  
CREATE TABLE Login (  
    loginID INTEGER PRIMARY KEY,  
    nombre TEXT,  
    contraseña TEXT  
);
```

Insertando registros en SQLite

Para que la aplicación cumpla con sus objetivos se insertaron 'Consumos' en nuestros productos del restaurante de ellos tendremos diferentes categorías:

1. Bebidas
2. Entradas
3. Segundo
4. Sopas

-- bebidas

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Coca-Cola', 'Bebida',  
'Refresco de cola', 2.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Jugo de naranja', 'Bebida',  
'Jugo natural de naranja', 3.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Cerveza IPA', 'Bebida',  
'Cerveza India Pale Ale', 4.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Agua mineral con gas',  
'Bebida', 'Agua con gas natural', 1.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Mojito', 'Bebida', 'Cóctel  
refrescante con menta y ron', 6.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Vino tinto Cabernet  
Sauvignon', 'Bebida', 'Vino tinto de uva Cabernet Sauvignon', 5.00);
```



```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sprite', 'Bebida', 'Refresco de lima-limón', 2.50);
```

-- entradas

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Ensalada César', 'Entrada', 'Lechuga romana, crutones y aderezo César', 7.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Calamares a la romana', 'Entrada', 'Calamares fritos con salsa de tomate', 8.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Bruschetta', 'Entrada', 'Pan tostado con tomate, ajo y albahaca', 6.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Tacos de pescado', 'Entrada', 'Tortillas de maíz con pescado a la parrilla', 9.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa de tomate', 'Entrada', 'Sopa a base de tomates, ajo y hierbas', 5.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Queso y uvas', 'Entrada', 'Tabla de quesos con uvas frescas', 10.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Nachos con guacamole', 'Entrada', 'Nachos con salsa de guacamole', 8.50);
```

-- segundo

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Filete de res a la parrilla', 'Segundo', 'Filete jugoso de carne de res con acompañamientos', 15.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Pollo a la parrilla', 'Segundo', 'Pechuga de pollo a la parrilla con verduras asadas', 12.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Salmón a la plancha', 'Segundo', 'Salmón fresco cocinado a la plancha con salsa de limón', 14.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Pasta Alfredo con pollo', 'Segundo', 'Fettuccine en una cremosa salsa Alfredo con pollo', 11.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Lomo de cerdo asado', 'Segundo', 'Lomo de cerdo asado con puré de papas', 16.00);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Ternera a la Stroganoff', 'Segundo', 'Ternera en salsa Stroganoff con arroz', 13.50);
```

```
INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Tofu salteado con verduras', 'Segundo', 'Tofu salteado con verduras en salsa de soja', 10.50);
```



-- sopas

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa de lentejas', 'Sopa', 'Sopa nutritiva de lentejas, zanahorias y cebollas', 6.00);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Gazpacho', 'Sopa', 'Sopa fría de tomate, pepino, pimiento y cebolla', 5.50);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa de pollo con fideos', 'Sopa', 'Caldo de pollo con fideos y verduras', 7.00);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Crema de champiñones', 'Sopa', 'Crema de champiñones con crutones', 6.50);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa tailandesa de coco', 'Sopa', 'Sopa tailandesa de coco con pollo y hierbas', 8.00);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa de tomate y albahaca', 'Sopa', 'Sopa de tomate con hojas frescas de albahaca', 6.50);

INSERT INTO Consumo (nombre, tipo, descripcion, precio) VALUES ('Sopa de miso', 'Sopa', 'Sopa japonesa de miso con tofu y algas', 5.00);

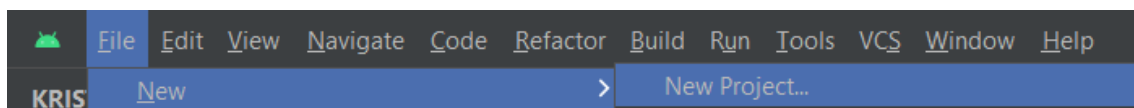
Creación de un nuevo proyecto en Android studio

Ejecutamos nuestra aplicación instalada de Android studio y procedemos con la creación del proyecto.



Android Studio

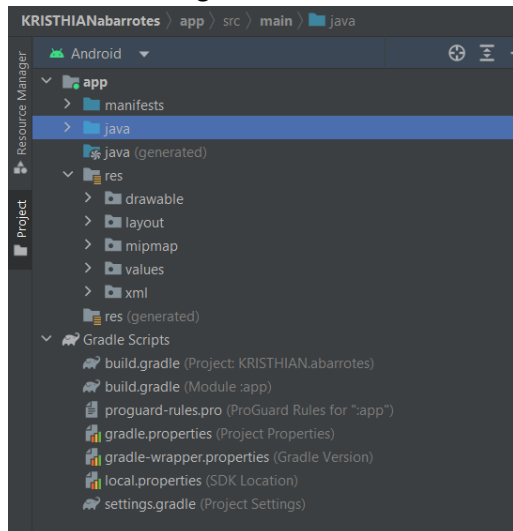
Aplicación



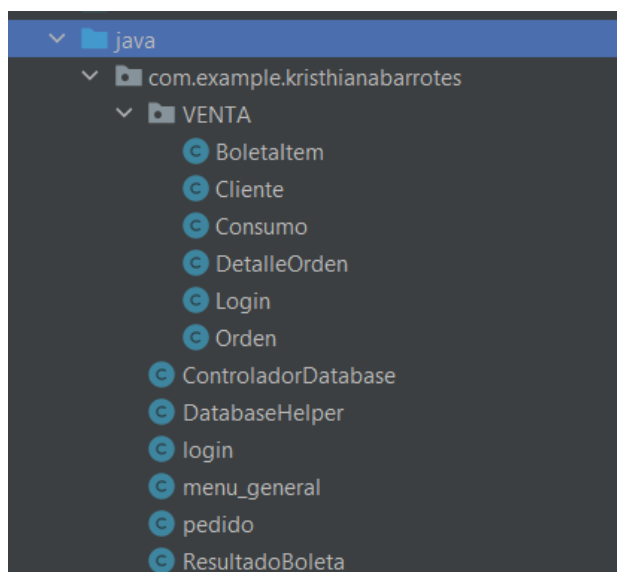
Paso 1 File – paso 2 New – paso 3 New Project.



Tendremos la siguiente estructura:



En la carpeta [java] crearemos nuestras entidades, así como las [Clase-layouts] del funcionamiento de nuestro aplicativo.



Dentro de nuestro paquete **com.example.krithianabarrotos** , dos bloques.

1. Las entidades, almacenadas en la carpeta [VENTA].
 - a. BoletaItem: Clase para guardar los pedidos generados por cliente.
 - b. Cliente: Clase para guardar datos del cliente.
 - c. Consumo: Clase para guardar datos de los productos del restaurante.
 - d. DetalleOrden: Clase que guarda las id de consumo, orden y cantidad de consumos pedidos para la boleta del cliente.
 - e. Login: Clase que guarda al usuario dueño de la aplicación en su restaurante.
 - f. Orden: Clase que guarda el cliente la fecha y numero de orden del pedido.
2. Las clases de los layouts y controladores de base de datos.



- ControladorDatabase: es donde se llevan los métodos que ejecuten desde la clase DatabaseHelper con la aplicación, en resumen nos ayuda a mejorar el proceso de ejecuciones del aplicativo.
- DatabaseHelper: contiene todos los métodos a realizar con la base de datos sqlite , incluido creación de tablas y controlador de versiones onCreate , onUpdate.
- Login: **Clase Layout** – la principal interfaz de la aplicación donde se muestra el ingreso de usuario y contraseña para poder acceder al sistema.
- Menú_general: **Clase Layout** – es el menú del sistemas al ingresar correctamente el login , nos muestra un resumen del día , monto del POS , cantidad mas vendidas ,etc.
- Pedido: **Clase Layout** – es el menú donde podemos elegir nuestro producto, cantidad y registrarnos como cliente para generar nuestra boleta.
- ResultadoBoleta: : **Clase Layout** – es donde se resume todo el pedido para generar la boleta de lo que vamos a elegir en el restaurante.

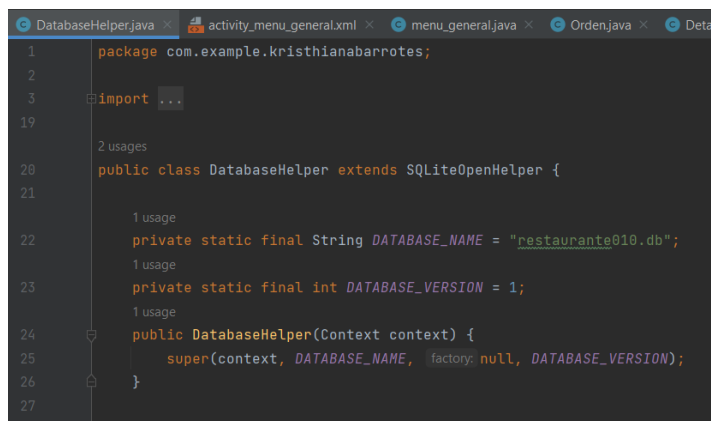
Creación de conexión SQLite en Android Studio

La conexión a sqlite en conjunto con Android y el lenguaje jva se debe generar mediante 2 clases siguientes:

DatabaseHelper

Debemos de crear la clase DatabaseHelper con herencia de SQLiteOpenHelper, tiene un constructor y dos métodos importantes:

Constructor



```
1 package com.example.krishianabarotes;
2
3 import ...
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28 public class DatabaseHelper extends SQLiteOpenHelper {
29
30     1 usage
31     private static final String DATABASE_NAME = "restaurante010.db";
32     1 usage
33     private static final int DATABASE_VERSION = 1;
34     1 usage
35     public DatabaseHelper(Context context) {
36         super(context, DATABASE_NAME, null, DATABASE_VERSION);
37     }
38 }
```

DATABASE_NAME: para colocar el nombre de la base de datos.

DATABASE_VERSION: para generar cambios en la creación de tablas y controlar sus versiones.

Cuando se instancia el Objeto **DatabaHelper** se tiene el parámetro Context context:



Algunos de los datos e información que puedes obtener a través de un objeto Context incluyen:

Cuadro de funcionalidades del Context.

- Paquete de la Aplicación: Puedes obtener información sobre el paquete de la aplicación, como su nombre, nombre del paquete, versión de la aplicación, etc.
- Recursos: Puedes acceder a recursos de la aplicación, como layouts de interfaz de usuario, imágenes, archivos XML, cadenas de texto, y más.
- Directorios de Archivos: Puedes obtener rutas a directorios específicos donde puedes almacenar y acceder a archivos de la aplicación, como el almacenamiento interno o externo.
- Recursos de Sistema: Puedes acceder a recursos y servicios del sistema, como la ubicación del dispositivo, la conectividad de red, la administración de alarmas, el administrador de contactos y otros servicios del sistema.
- Acceso a la Base de Datos SQLite: Puedes utilizar el contexto para crear instancias de clases que te ayuden a interactuar con bases de datos SQLite dentro de la aplicación.

1. Método onCreate

Es donde vamos a colocar las CREATE de las tablas del aplicativo , tiene como parámetro una instancia de SQLiteDatabase db , para poder iniciar la conexión a sqlite3.

```
@Override
public void onCreate(SQLiteDatabase db) {

    String TABLA_LOGIN = "CREATE TABLE Login_Principal(" +
        "loginID INTEGER PRIMARY KEY," +
        "nombre TEXT UNIQUE," +
        "contraseña TEXT" +
        ");";

    String TABLA_CLIENTES = "CREATE TABLE Clientes (" +
        "ClienteID INTEGER PRIMARY KEY," +
        "nombre TEXT," +
        "apellidos TEXT," +
        "DNI TEXT UNIQUE" +
        ");";
```

2. Método onUpgrade

Al final de la clase DatabaseHelper se encuentra este método para actualizar y generar control de versiones, usando la variable DATABASE_VERSION.



```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    db.execSQL("DROP TABLE IF EXISTS " + "Login_Principal");
    db.execSQL("DROP TABLE IF EXISTS " + "Clientes");
    db.execSQL("DROP TABLE IF EXISTS " + "Ordenes");
    db.execSQL("DROP TABLE IF EXISTS " + "Consumo");
    db.execSQL("DROP TABLE IF EXISTS " + "Detalles_de_la_Orden");

    onCreate(db);
}
```

Nos damos cuenta de la sentencia DROP la cual elimina la tabla si es que existe en la aplicación, para poder conseguir una actualización general.

* La Clase **ControladorDatabase** es importante ya que contiene no solo a la conexión más clara de la base de datos dentro de **DatabaseHelper**, sino presenta los métodos que se van a usar en el funcionamiento de la aplicación.

ControladorDatabase

La clase ControladorDatabase tiene 3 parámetros:

1. dbHelper
Instancia del objeto DatabaseHelper.
2. context
instancia Context. ver [Cuadro de funcionalidades del Context].
3. Database
Instancia de la conexión a base de datos.

```
public class ControladorDatabase {

    19 usages
    private DatabaseHelper dbHelper;
    2 usages
    private Context context;
    1 usage
    static SQLiteDatabase database;

    4 usages
    public ControladorDatabase(Context c) { context = c; }

    4 usages
    public ControladorDatabase open() throws SQLException {
        dbHelper = new DatabaseHelper(context);
        database = dbHelper.getWritableDatabase();

        return this;
    }

    public void close() { dbHelper.close(); }
```

Uso Básico:

1. Creación de una instancia:



Para utilizar la clase `ControladorDatabase`, primero debes crear una instancia de la misma en tu aplicación. Puedes hacerlo de la siguiente manera:

```
ControladorDatabase controladorDB = new ControladorDatabase(context);
```

Donde `context` es el contexto de tu aplicación, generalmente pasado desde una actividad o fragmento de Android.

2. Apertura de la base de datos:

Antes de realizar cualquier operación en la base de datos, debes abrirla. Esto se hace llamando al método `open()`:

```
controladorDB.open();
```

3. Inserción de Datos:

Puedes utilizar los métodos `insertar_login_usuario`, `insertar_consumo`, `insertarCliente`, `insertarOrden`, y `insertarDetalleOrden` para insertar datos en la base de datos. Por ejemplo:

```
long resultado = controladorDB.insertar_login_usuario("usuario", "contraseña");
```

Esto insertará un registro en la tabla de usuarios.

4. Consulta de Datos:

La clase `ControladorDatabase` proporciona métodos para consultar datos. Puedes utilizar `comprobar_login`, `mostrar_consumo`, `obtenerTodosLosConsumosPorID`, `comprobar_dni_cliente`, y otros métodos para obtener información de la base de datos.

```
boolean existeUsuario = controladorDB.comprobar_login("usuario", "contraseña");
```

5. Cierre de la base de datos:

Una vez que hayas terminado de trabajar con la base de datos, asegúrate de cerrarla llamando al método `close()` para liberar recursos:

```
controladorDB.close();
```

Ejemplos Adicionales:

- Para obtener una lista de todos los consumos:

```
List<Consumo> consumos = controladorDB.mostrar_consumo();
```



- Para calcular el monto total de una orden:

```
double montoTotal = controladorDB.calcularMontoTotal("numeroDeOrden");
```

- Para verificar si un número de orden existe:

```
boolean existeOrden = controladorDB.comprobar_numero_de_orden("numeroDeOrden");
```

- Para obtener una lista de los elementos de una boleta por número de orden:

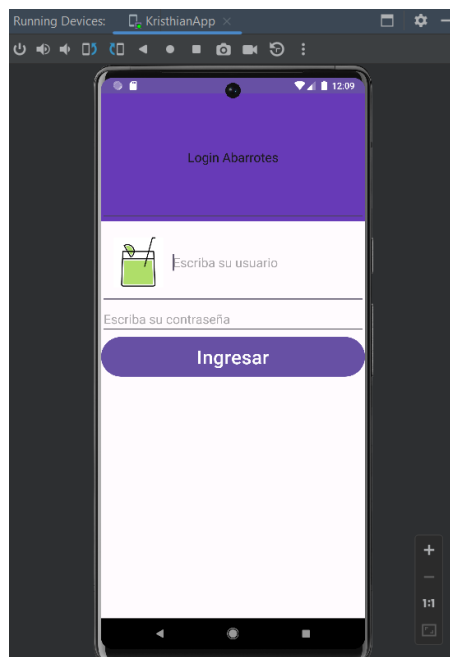
```
List<BoletaItem> boletaItems =  
controladorDB.obtenerBoletaPorNumeroDeOrden("numeroDeOrden");
```

*** Notas Importantes:**

- Debemos de asegurarnos de manejar excepciones adecuadamente al utilizar esta clase, ya que algunos métodos pueden arrojar excepciones de SQL.

Codificando **ACTIVITY LAYOUTS** [CODE] [DESIGN] en Android Studio

Funcionamiento de la Clase activity layout login





Descripción:

La clase login es una actividad de Android que representa la pantalla de inicio de sesión de la aplicación. Los usuarios o administradores del negocio pueden ingresar su nombre de usuario y contraseña para acceder a la aplicación.

Uso Básico

Creación de una instancia:

Cuando un usuario inicia la aplicación, se crea una instancia de la clase login. En su método onCreate(), se realizan las siguientes acciones:

- Se ejecuta el diseño de la actividad desde R.layout.activity_login.
- Se obtienen referencias a los elementos de la interfaz de usuario, como botones y campos de texto.
- Se crea una instancia de ControladorDatabase para interactuar con la base de datos.

Apertura de la base de datos:

- En el método onCreate(), se llama al método open() de ControladorDatabase para abrir la base de datos.

Intento de Inicio de Sesión:

- Cuando el usuario hace clic en el botón "Ingresar", se obtienen los valores ingresados en los campos de usuario y contraseña. Luego, se llama al método **comprobando_login** de **ControladorDatabase** para verificar las credenciales del usuario en la base de datos.

Resultado del Inicio de Sesión:

Si las credenciales son válidas (Cond es true), se inicia la actividad menu_general y se muestra un mensaje de inicio de sesión exitoso.

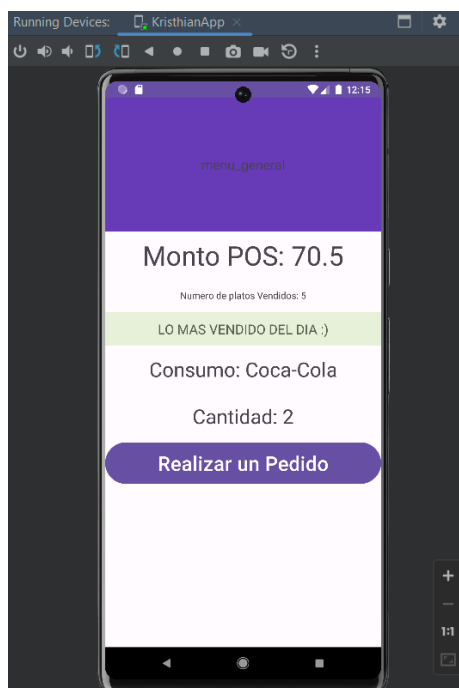
Si las credenciales son incorrectas (Cond es false), se muestra un mensaje de error.

Notas Importantes:

- La apertura y cierre de la base de datos deben manejarse adecuadamente para evitar pérdida de datos o recursos no liberados.
- El código incluye un intent para iniciar la actividad menu_general después de un inicio de sesión exitoso.



Funcionamiento de la Clase activity layout menú_general



Descripción

La clase `menu_general` representa la pantalla principal de la aplicación después de iniciar sesión. Muestra información como el monto POS, el número de platos vendidos, el consumo del día y la cantidad del día.

Uso Básico

Creación de una instancia:

Cuando un usuario inicia sesión correctamente y se inicia la actividad `menu_general`, se crea una instancia de esta clase. En su método `onCreate()`, se realizan las siguientes acciones:

- Se ejecuta el diseño de la actividad desde `R.layout.activity_menu_general`.
- Se obtienen referencias a elementos de la interfaz de usuario, como `TextViews` y botones.
- Se crea una instancia de `ControladorDatabase` para interactuar con la base de datos.

Apertura de la base de datos:

- En el método `onCreate()`, se llama al método `open()` de `ControladorDatabase` para abrir la base de datos.



Mostrar Información:

- Se obtiene información de la base de datos utilizando métodos como **mostrar_pos_del_dia**, **mostrar_cantidad_de_consumos_vendidos** y **mostrar_consumo_mas_vendido**.

*La información se muestra en los TextViews correspondientes en la interfaz de usuario.

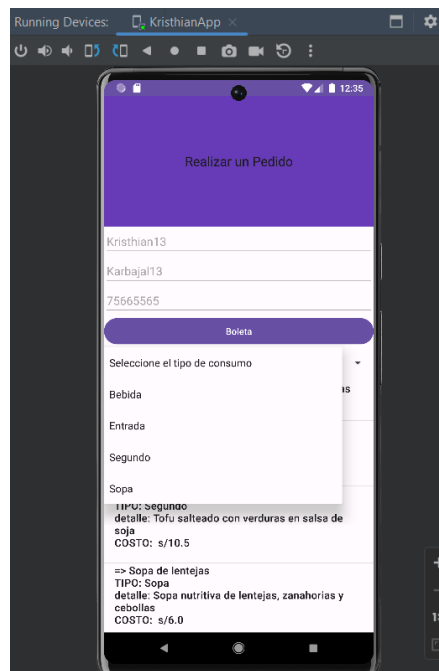
Realizar Acciones:

Si el usuario hace clic en el botón "Pedido", se inicia la actividad pedido.

Notas Importantes:

- La apertura y cierre de la base de datos deben manejarse adecuadamente para evitar pérdida de datos o recursos no liberados.
- El código muestra información en la interfaz de usuario basada en los datos obtenidos de la base de datos.
- Se proporciona un botón para permitir al usuario realizar pedidos.

Funcionamiento de la Clase activity layout pedido



Descripción



La clase pedido representa una actividad en la aplicación que permite a los usuarios realizar pedidos. Los usuarios pueden seleccionar consumos de una lista, agregarlos a un carrito de compras y generar una orden de venta.

Uso Básico.

Creación de una instancia:

Cuando un usuario accede a la pantalla de pedidos, se crea una instancia de la clase pedido. En su método onCreate(), se realizan las siguientes acciones:

- Se ejecuta el diseño de la actividad desde R.layout.activity_pedido.
- Se obtienen referencias a los elementos de la interfaz de usuario, como campos de texto, botones, un spinner y una lista (ListView).
- Se crea una instancia de ControladorDatabase para interactuar con la base de datos.

Selección de Consumos:

- Se utiliza un Spinner (spinTIPO) para permitir al usuario seleccionar un tipo de consumo (por ejemplo, "Bebida" o "Entrada").
- Se muestra una lista de consumos del tipo seleccionado en un ListView (listViewPersonas).

Añadir al Carrito:

- Cuando el usuario hace clic en un elemento de la lista de consumos, se agrega ese consumo al carrito de compras (carrito).
- Se valida que se hayan ingresado el nombre, apellido y DNI del cliente antes de agregar consumos al carrito.

Generar Venta:

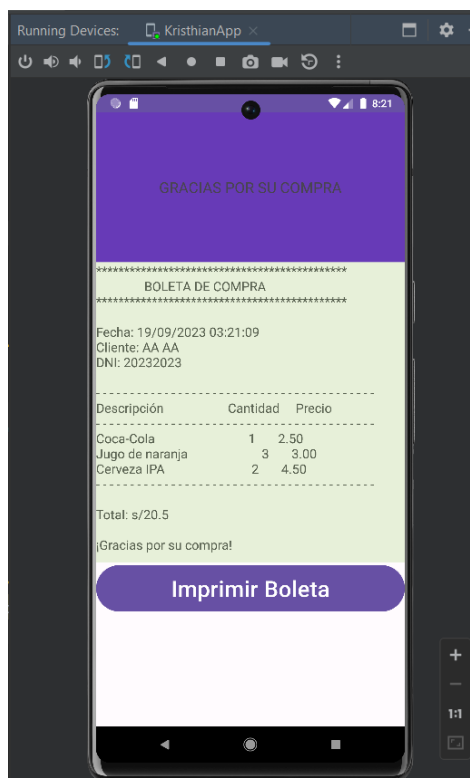
- Cuando el usuario hace clic en el botón "Carrito", se genera una orden de venta.
- Se genera un número de orden aleatorio y se verifica que sea único en la base de datos.
- Se cuenta la cantidad de cada consumo en el carrito y se almacena en una lista de DetalleOrden.
- Se crea un nuevo cliente en la base de datos si no existe.
- Se almacena la orden de venta y sus detalles en la base de datos.
- Se inicia la actividad ResultadoBoleta para mostrar los detalles de la boleta.

Notas Importantes:

- El código incluye manejo de excepciones y validación de campos antes de agregar consumos al carrito.
- Se utiliza un Spinner para filtrar los consumos por tipo y una ListView para mostrarlos.
- Se generan números de orden aleatorios únicos para cada venta.



Funcionamiento de la Clase activity layout ResultadoBoleta



Descripción

La clase ResultadoBoleta representa una actividad en la aplicación que muestra los detalles de una boleta de venta, incluido el monto total de la compra.

Uso Básico:

Creación de una instancia:

Cuando se inicia la actividad ResultadoBoleta, se crea una instancia de esta clase. En su método onCreate(), se realizan las siguientes acciones:

- Se infla el diseño de la actividad desde R.layout.activity_resultado_boleta.
- Se obtiene el número de orden de la boleta desde la actividad anterior mediante un Intent.
- Se crea una instancia de ControladorDatabase para interactuar con la base de datos.

Obtener Datos de la Boleta:



- Se utiliza el número de orden obtenido del Intent para recuperar los detalles de la boleta desde la base de datos.
- Se calcula el monto total de la compra llamando al método `calcularMontoTotal` de `ControladorDatabase`.

Mostrar Información:

- Se muestra la información de la boleta, incluido el monto total, en la interfaz de usuario.

Notas Importantes:

- El código utiliza un Intent para recibir el número de orden de la boleta de la actividad anterior.
- Se obtienen los detalles de la boleta y se calcula el monto total utilizando `ControladorDatabase`.
- La información se muestra en la interfaz de CLIENTE para que el usuario la vea.

Considerando funcionalidades finales en Android Studio

Pruebas y depuración: Una vez que la aplicación está desarrollada, es importante realizar pruebas exhaustivas para asegurarte de que funcione correctamente y cumpla con los requerimientos establecidos. Identifica y soluciona cualquier error o falla en la aplicación.

Implementación y lanzamiento: En esta fase, se prepara la aplicación para su implementación en el entorno de producción. Genera un archivo de instalación (APK) y realiza las pruebas finales antes de lanzar la aplicación en la tienda de aplicaciones de Android, como Google Play Store.

Seguimiento y mantenimiento: Después del lanzamiento, debes realizar un seguimiento continuo de la aplicación para asegurarte de que esté funcionando correctamente y responder a cualquier problema o retroalimentación de los usuarios. Además, puedes realizar actualizaciones y mejoras en la aplicación para agregar nuevas características o corregir errores.