

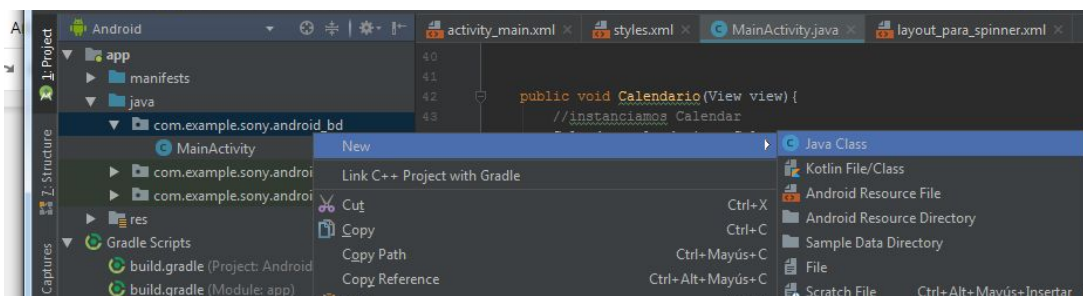
Índice

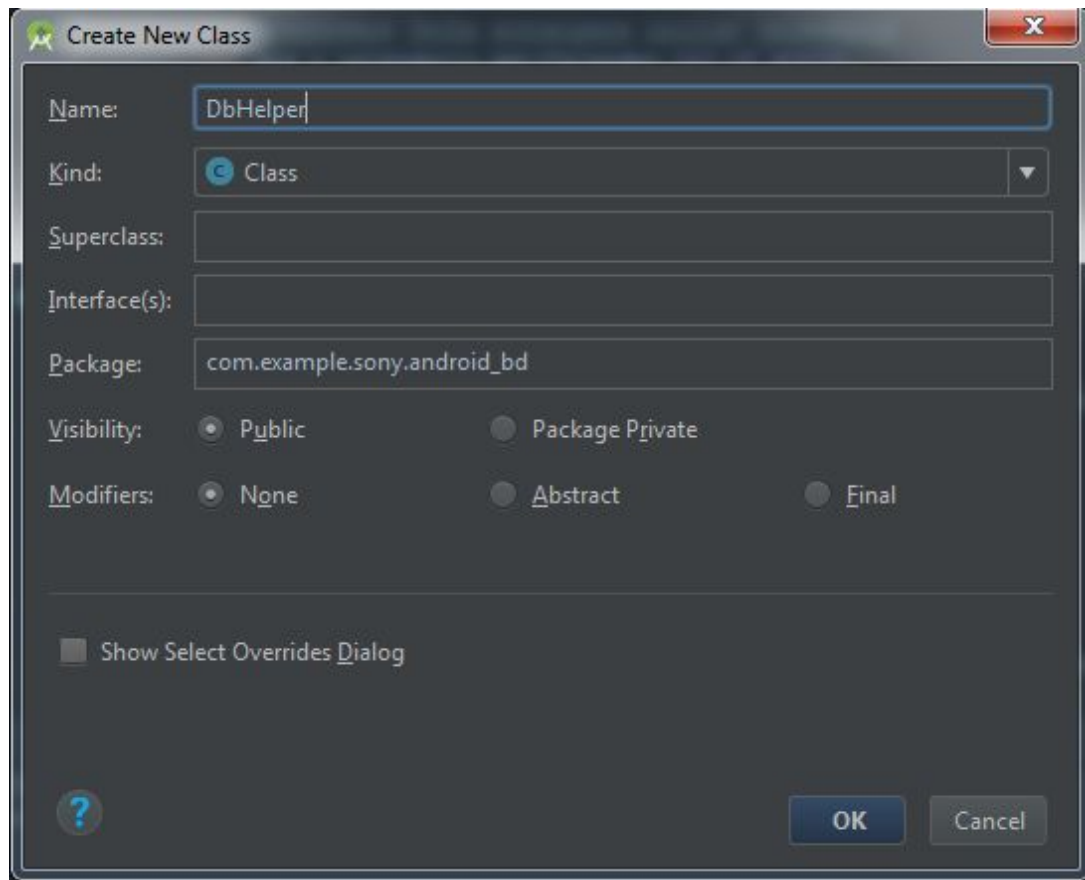
SQLITE	2
Crear Base de Datos por medio de DbHelper	2
Extraer BD creada dentro del Emulador	8
Layout para Spinner	10
Captura de Fechas	12
INSERTAR DATOS	13
Listar	14
Eliminar	17
Nota 1	18
Nota 2	19

SQLITE

Diferencia de otros Motores de Base de Datos, Sqlite es una versión pequeña , tiene funciones reducidas, no necesita conexion a servidor y administra de manera local los datos Crearemos una clase llamada DbHelper la cual reemplazará la clase coneccion que se utiliza en la interacción con los motores convencionales.

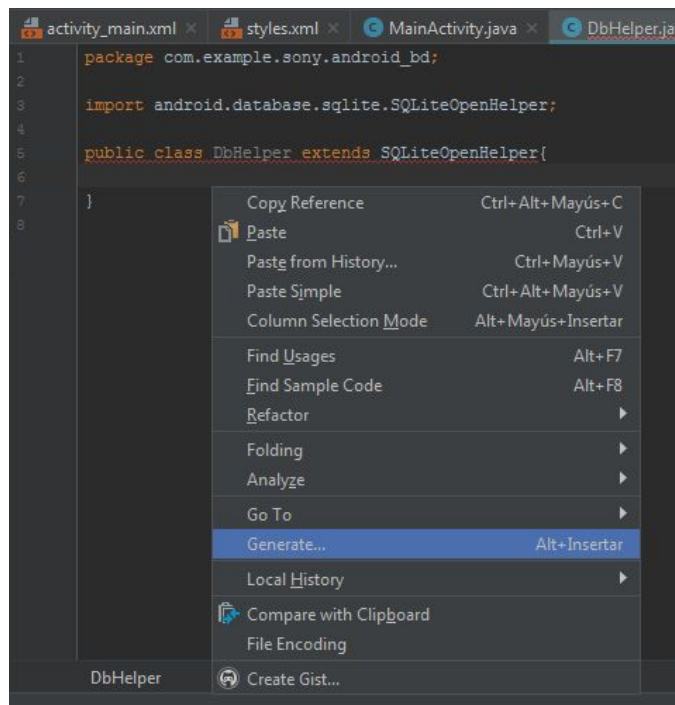
Crear Base de Datos por medio de DbHelper



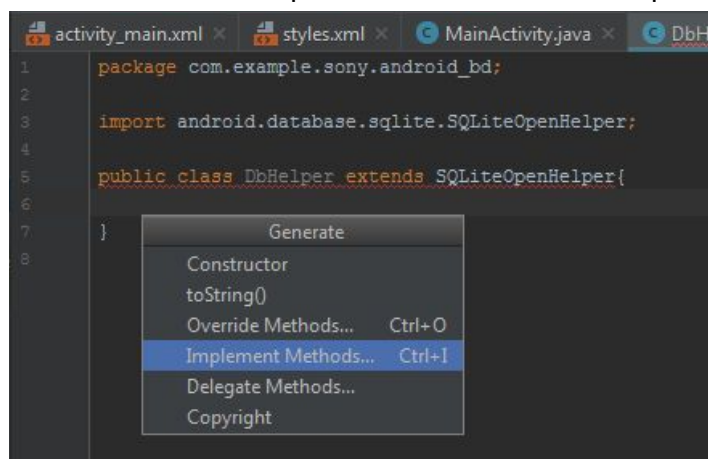


Creamos la clase y **Generamos Código** para realizar un trabajo más rápido

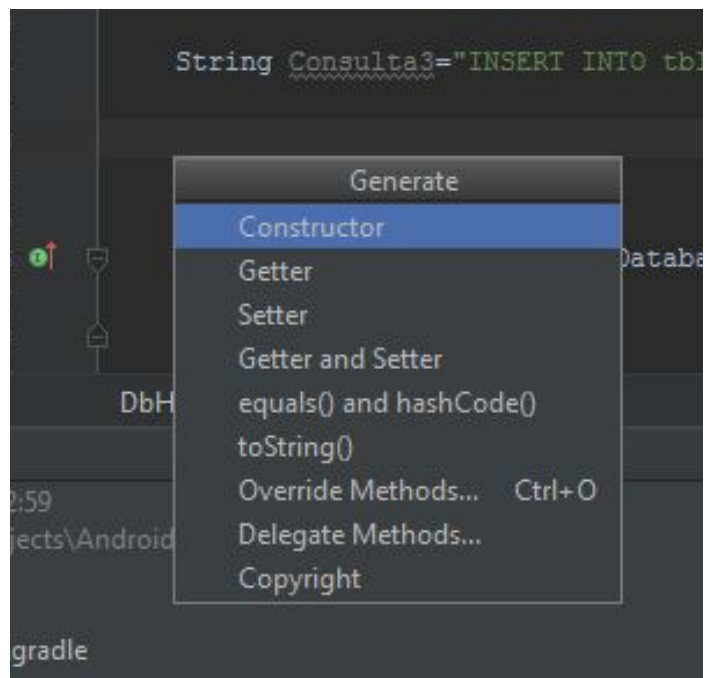
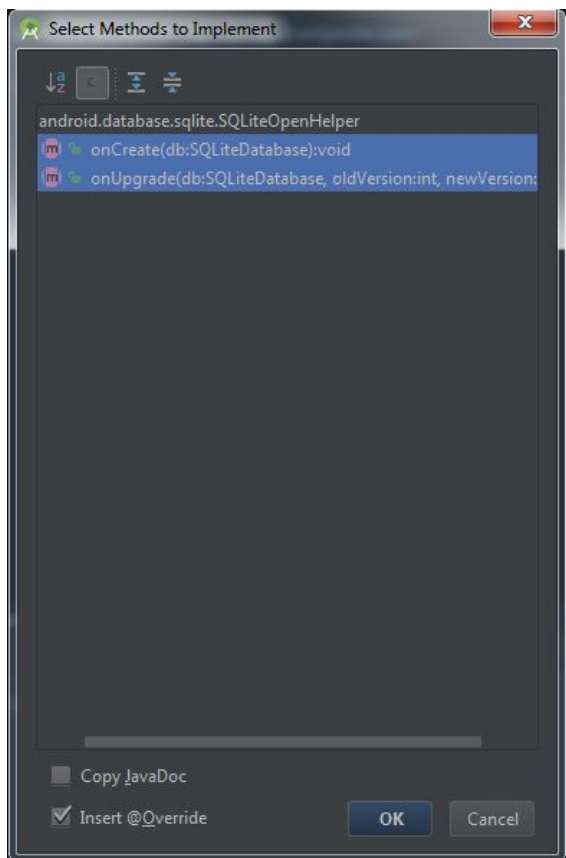
```
activity_main.xml x styles.xml x MainActivity.java x DbHelper.java x
1 package com.example.sony.android_bd;
2
3 import android.database.sqlite.SQLiteOpenHelper;
4
5 public class DbHelper extends SQLiteOpenHelper{
6     |
7 }
8
```



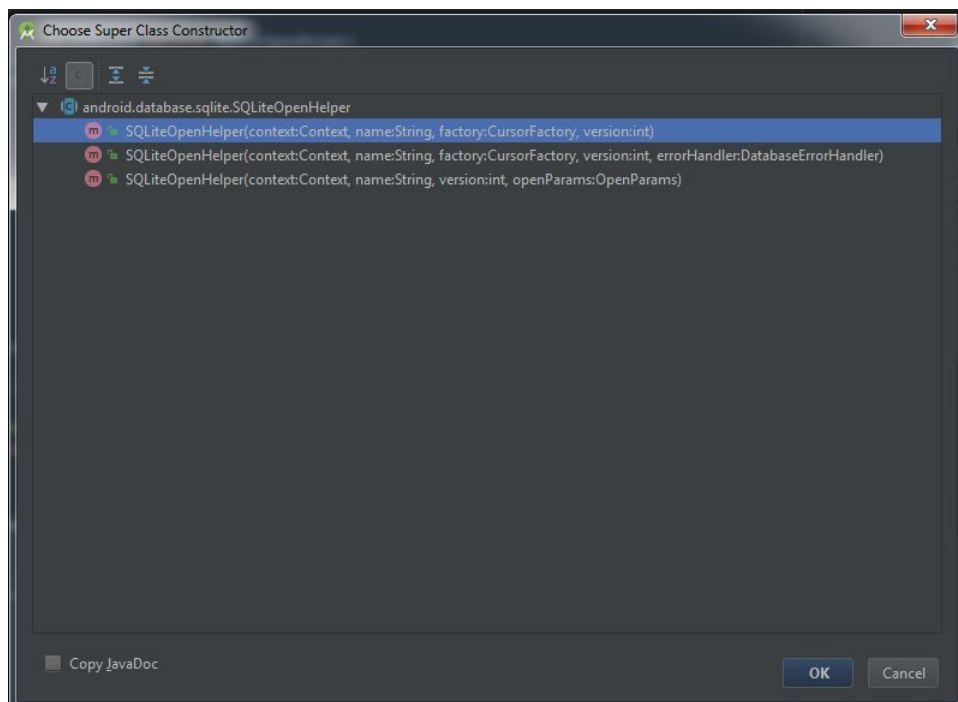
De la misma forma implementamos los métodos que serán heredados



Seleccionamos onCreate y onUpgrade, además de generar el código del Constructor



Seleccionamos el que contiene argumentos: NameString ,Factory,CursorFactory,VersionInt



Ya con la Estructura formada podemos crear las consultas que crearan el esquema de la base de datos:

```

1 package com.example.sony.android_bd;
2
3 import ...
4
5
6
7 public class DbHelper extends SQLiteOpenHelper{
8
9
10
11     public DbHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
12         super(context, name, factory, version);
13     }
14
15     @Override
16     public void onCreate(SQLiteDatabase db) {
17
18     }
19
20     @Override
21     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
22
23     }
24 }

```

en las siguientes consultas se crean tablas Ítem y Gasto, recordar que se debe crear primero la tabla que no tiene dependencia (Foreign Key), mientras que en el método onCreate irán las consultas que se ejecutarán a través del método execSQL al momento de ser instanciada **SQLiteDatabase**.

```

7 public class DbHelper extends SQLiteOpenHelper{
8     // text o integer en sql lite
9     String consulta1 =
10         "CREATE TABLE IF NOT EXISTS tbl_item (" +
11         "id_item INTEGER PRIMARY KEY AUTOINCREMENT," +
12         "nombre_item TEXT)";
13
14     String consulta2 =
15         "CREATE TABLE IF NOT EXISTS tbl_gasto (" +
16         "id_gasto INTEGER PRIMARY KEY AUTOINCREMENT," +
17         "monto_gasto TEXT," +
18         "descripcion TEXT," +
19         "fecha TEXT," +
20         "id_item INTEGER," +
21         "FOREIGN KEY (id_item) REFERENCES tbl_item(id_item))";
22
23     String consulta3 =
24         "INSERT INTO tbl_item VALUES" +
25         "(null,'TIENDAS')," +
26         "(null,'SUPERMERCADOS')," +
27         "(null,'VESTUARIO')," +
28         "(null,'CUENTAS')," +
29         "(null,'AUTOPISTAS')," +
30         "(null,'OTRO')";
31
32     public DbHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
33         super(context, name, factory, version);
34     }
35
36     @Override
37     public void onCreate(SQLiteDatabase db) {
38         db.execSQL(consulta1);
39         db.execSQL(consulta2);
40         db.execSQL(consulta3);
41     }
42
43     @Override
44     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
45
46     }
47 }

```

ahora dentro del Activity realizamos las declaraciones de objetos,

```

1 package com.example.sony.android_bd;
2
3 import ...
4
20
21 public class MainActivity extends AppCompatActivity {
22
23     //declaracion de objetos
24     private EditText edtMonto, edtDescripcion;
25     private Spinner spItem;
26     private TextView tvFecha;
27

```

dentro del método onCreate además de la referencias correspondientes se instanciara la clase DbHelper, en la cual va el nombre de la base de datos que crearemos (db_ejercicio) y la clase SQLiteDatabase que nos permitirá leer escribir en la base de datos, a través del método getReadableDatabase o getWritableDatabase según sea el caso, en esta ocasión leeremos datos, por lo cual creamos un cursor (Cursor c) y la respectiva consulta en el método rawquery(), este cursor recibirá la estructura de la tabla y los datos, para posteriormente extraerlos.

```

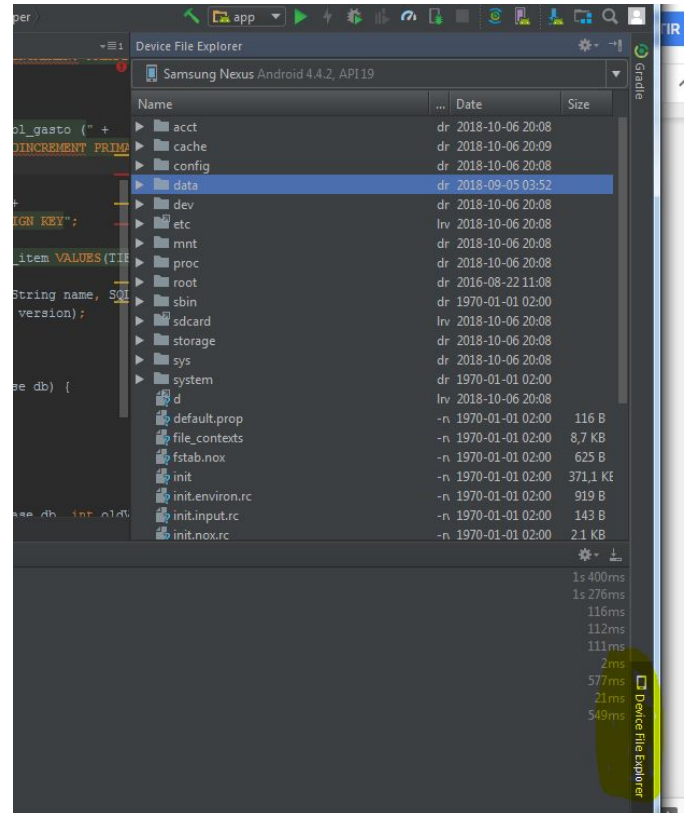
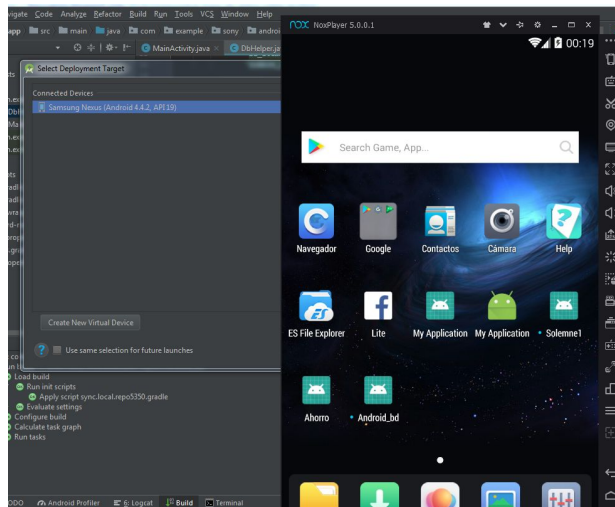
28 @Override
29 protected void onCreate(Bundle savedInstanceState) {
30     super.onCreate(savedInstanceState);
31     setContentView(R.layout.activity_main);
32     //asignacion de referencias
33     edtMonto = (EditText) findViewById(R.id.editTextMonto);
34     edtDescripcion = (EditText) findViewById(R.id.editTextDescripcion);
35     tvFecha = (TextView) findViewById(R.id.textViewFecha);
36     spItem = (Spinner) findViewById(R.id.spinnerItem);
37
38     //se crea la base de datos db_ejercicio
39     DbHelper dbHelper = new DbHelper( context: this, name: "db_ejercicio", factory: null, version: 1);
40
41     //indicamos que sera de tipo lectura la consulta "getReadableDatabase"
42     SQLiteDatabase db = dbHelper.getReadableDatabase();
43
44     //se declara cursor y se le ingresa el valor del metodo rawquery que es un resultset
45     // en db.rawQuery los argumentos son la consulta select y los argumentos para filtrar en este caso null
46     Cursor c = db.rawQuery( sql: "SELECT * FROM tbl_item", selectionArgs: null);
47
48     //getCount devuelve la cantidad de registros obtenidos en la consulta
49     int cantidad= c.getCount();
50
51     String arreglo[] = new String[cantidad];
52
53     //indicamos el mensaje inicial del spinner en la primera posicion
54     arreglo[0]="Seleccione Item";
55
56     if(cantidad>0) {
57         //llenamos el spinner a partir de la segunda posicion "1"
58         for (int i = 1; i < arreglo.length; i++) {
59             c.moveToNext();
60             arreglo[i] = c.getInt( columnIndex: 0)+"-"+c.getString( columnIndex: 1);
61         }
62     }
63     ArrayAdapter<String>adaptador;
64     adaptador=new ArrayAdapter<String>( context: this, R.layout.layout_para_spinner,arreglo);
65     spItem.setAdapter(adaptador);
66 }

```

ejecutamos un ciclo y a través del método movetonext() avanzaremos de registro en registro mientras que getInt o getString nos permite extraer los datos de esos registros, dependiendo si son TEXT o INTEGER. dentro estos métodos van los índices de las columnas de los registros (0 para la primera), esto es almacenado en un array y luego a través de un adaptador se incorpora a un **spinner**.

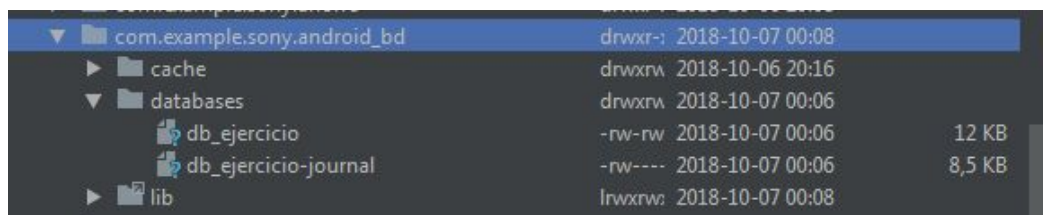
Extraer BD creada dentro del Emulador

mientras se emula podemos acceder a los datos del móvil emulado presionando en la pestaña derecha inferior, como se ve en la imagen.

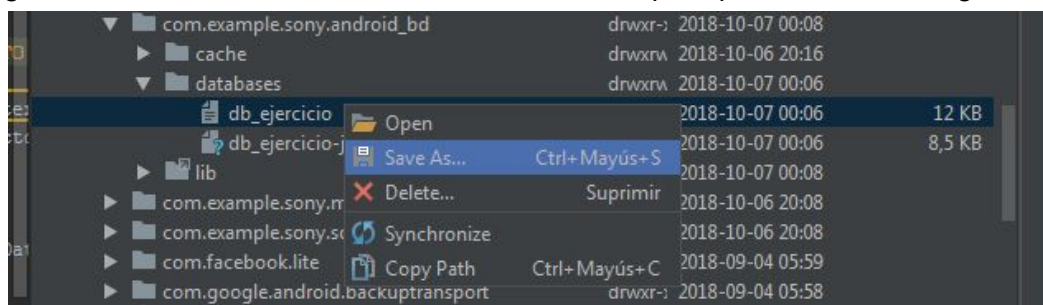


Buscamos la Ruta:

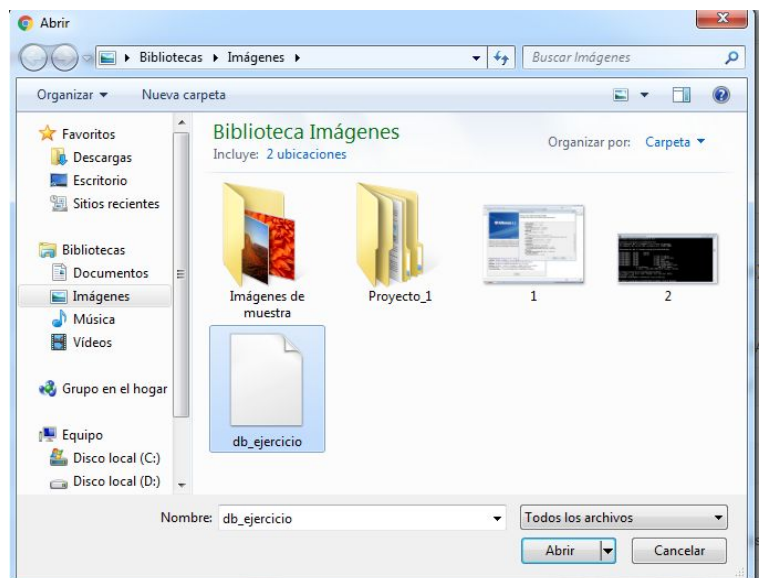
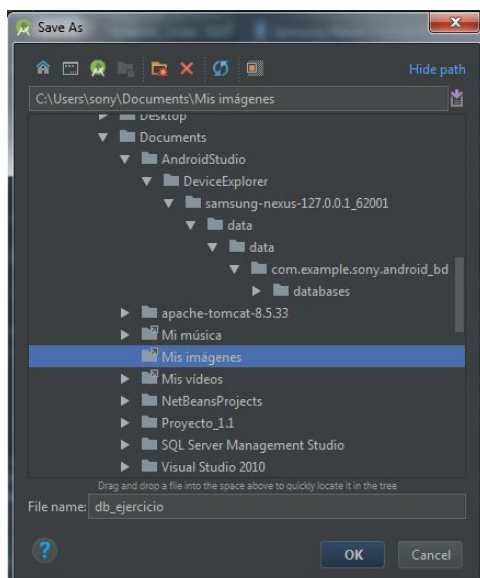
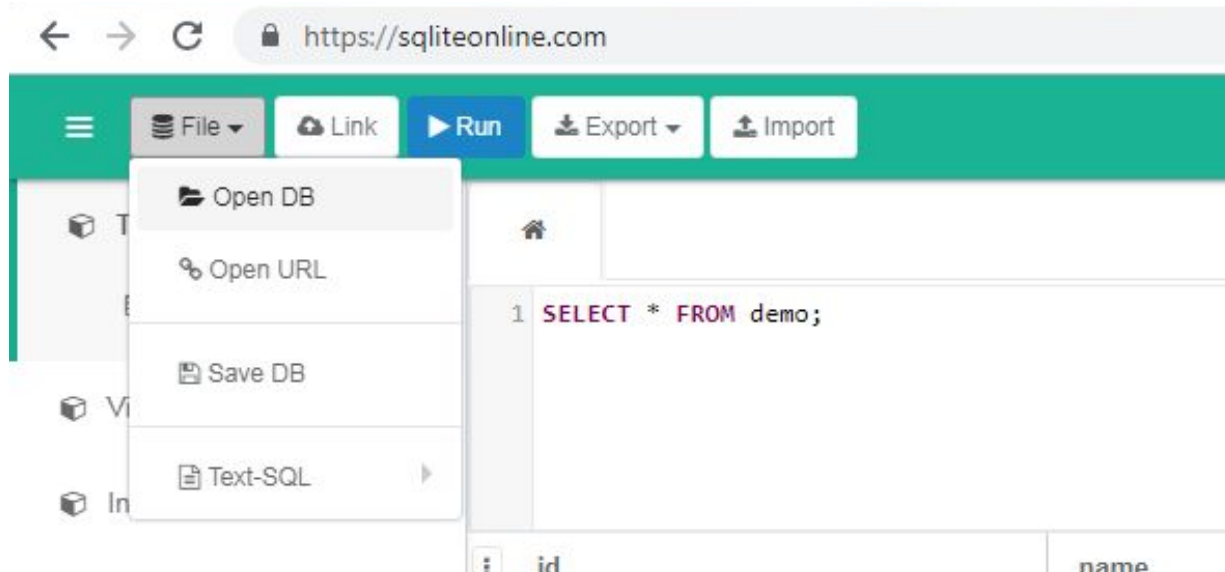
`/data/data/[Nombre del Proyecto]/databases/`



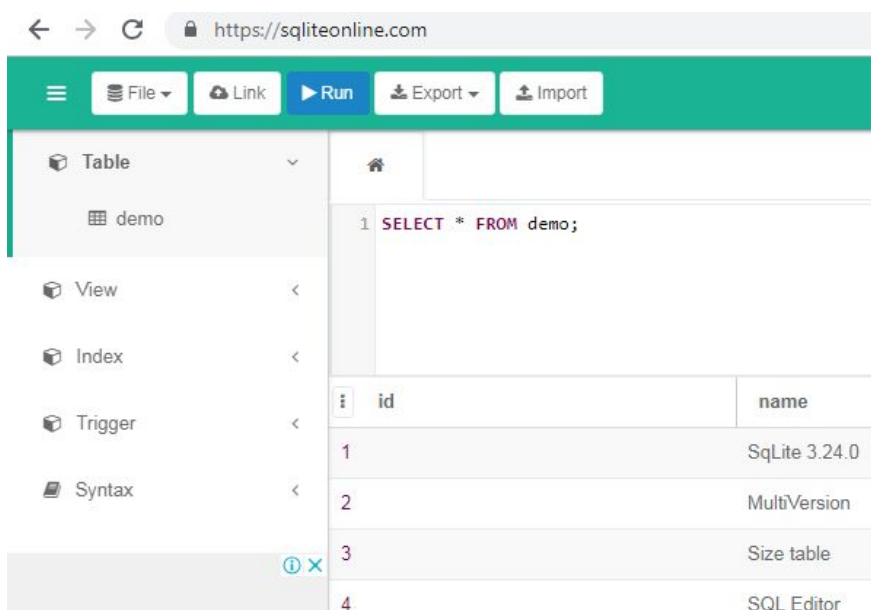
guardamos el archivo en una ruta de facil acceso para poder utilizarlo luego,



Abrimos el explorador y nos vamos a la dirección <https://sqliteonline.com/> donde es posible subir nuestra base de datos sqlite dandole la ruta donde la guardamos previamente

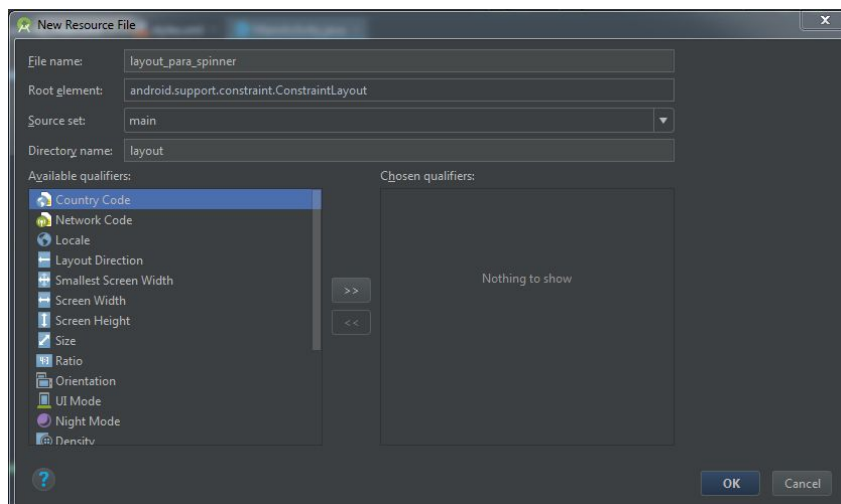
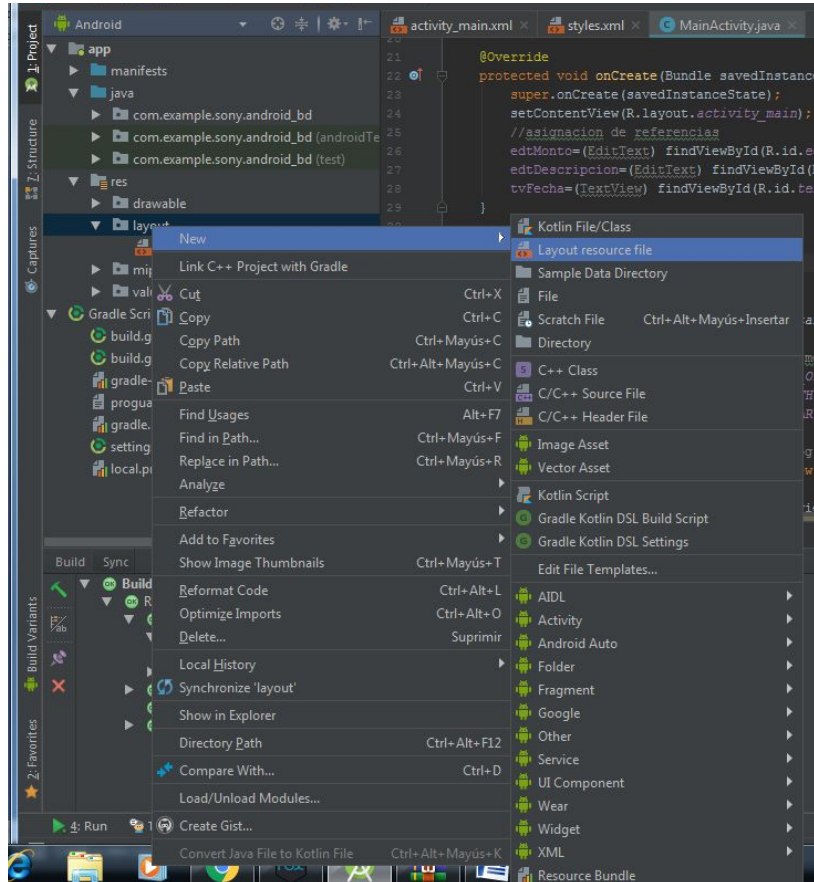


una vez arriba podemos realizar consultas y verificar sintaxis de queries.

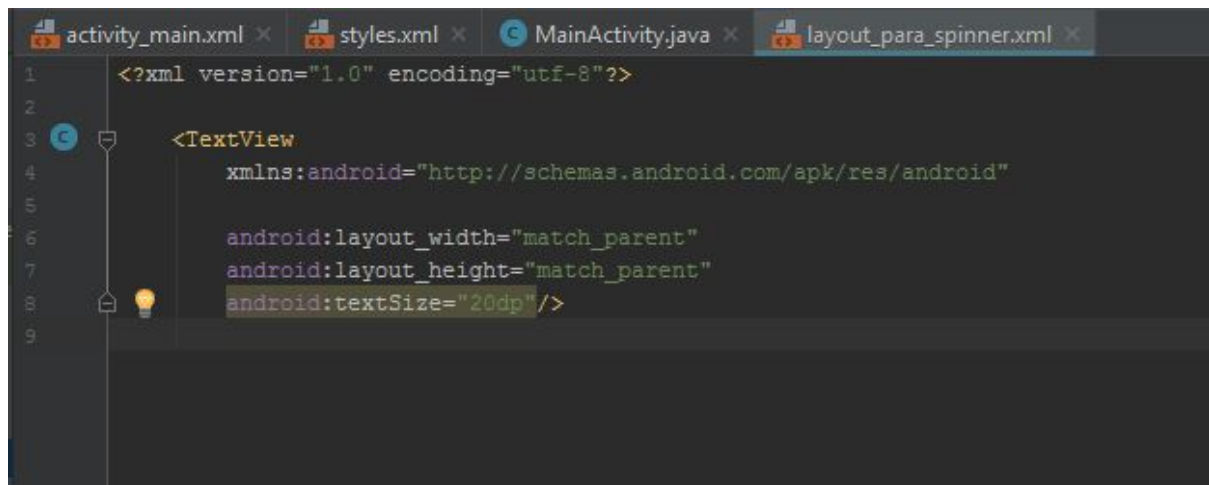


Layout para Spinner

Es posible crear nuestros Custom Layouts, para ellos presionamos boton derecho en la carpeta layout,new,Layout resource file



una vez creado es posible modificar o agregar distintos formatos, en este caso sera un formato de textview

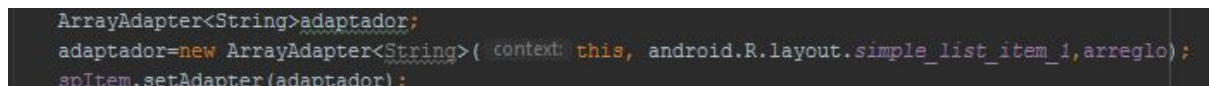
A screenshot of the Android Studio IDE. The top toolbar shows icons for activity_main.xml, styles.xml, MainActivity.java, and layout_para_spinner.xml. The layout_para_spinner.xml file is open in the editor, showing an XML snippet for a TextView. The code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:textSize="20dp"/>
```

para incorporarlo debemos indicarlo como argumento dentro del método constructor del ArrayAdapter.

Estilo de Spinner Predeterminado:

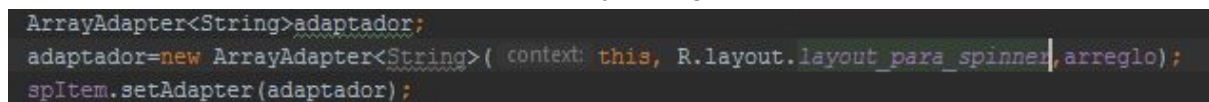
es el que se utiliza por defecto generalmente y su argumento se escribe así.

A screenshot of a Java code editor showing the following code:

```
ArrayAdapter<String>adaptador;
adaptador=new ArrayAdapter<String>( context: this, android.R.layout.simple_list_item_1,arreglo);
spItem.setAdapter(adaptador);
```

Estilo de Spinner Custom

es el estilo personalizado que podemos darle y su argumento se escribe así

A screenshot of a Java code editor showing the following code:

```
ArrayAdapter<String>adaptador;
adaptador=new ArrayAdapter<String>( context: this, R.layout.layout_para_spinner,arreglo);
spItem.setAdapter(adaptador);
```

Captura de Fechas

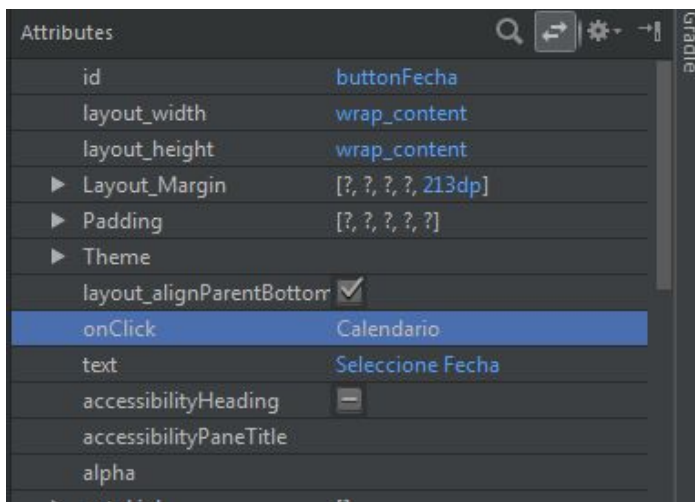
para capturar las fechas de una manera adecuada existen Herramientas como la Clase DatePickerDialog la cual llamaremos a través de un botón que contendrá el evento onclick que ejecutará este método

```
public void Calendario(View view){
    //instanciamos Calendar
    Calendar calendario = Calendar.getInstance();

    // obtendremos fecha actual para iniciar calendario
    int dia = calendario.get(Calendar.DAY_OF_MONTH);
    int mes = calendario.get(Calendar.MONTH);
    int anno = calendario.get(Calendar.YEAR);

    // instanciamos metodo DatePickerDialog
    DatePickerDialog datePickerDialog = new DatePickerDialog( context: this, (view, year, month, dayOfMonth) -> {
        tvFecha.setText(dayOfMonth+"/"+(month+1)+"/"+year);
    }, anno, mes, dia); //se le agregan los argumentos año, mes, dia para fecha default sea la actual

    // se muestra el calendario
    datePickerDialog.show();
}
```



ciisa Instituto de Ciencias Tecnológicas

Ingrese Monto

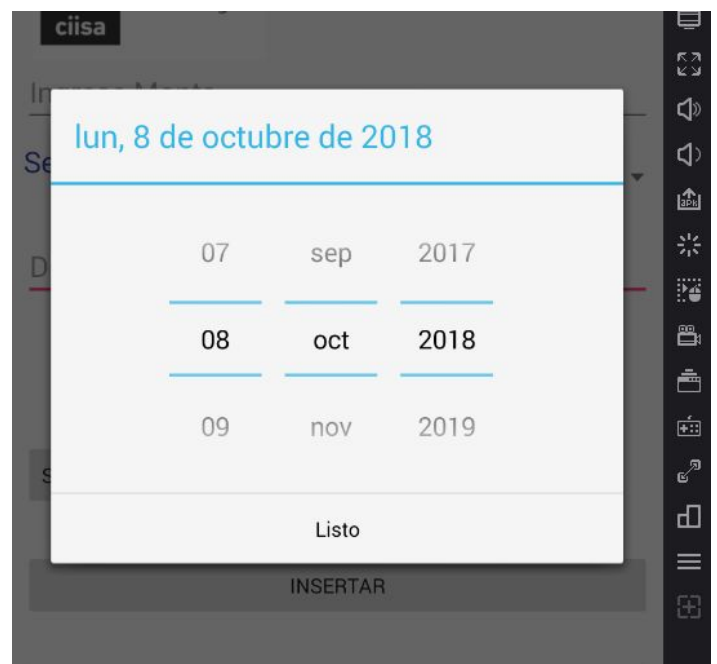
Seleccione Item

Descripcion

SELECCIONE FECHA 00/00/0000

INSERTAR

LISTAR



INSERTAR DATOS

Para Insertar Datos crearemos un Método Insertar en el cual primero validamos los datos.

```
public void insertar(View view) {
    //declaramos variables que reciban datos
    int monto;
    String descripcion, fecha, item;

    if (edtMonto.getText().toString().trim().equalsIgnoreCase( " " )) {
        edtMonto.setError( "Debe Ingresar Monto" );
    } else if (spItem.getSelectedItemPosition() == 0) {
        Toast.makeText( context, this, "Debe Seleccionar Item", Toast.LENGTH_SHORT ).show();
    } else if (edtDescripcion.getText().toString().trim().equalsIgnoreCase( " " )) {
        edtDescripcion.setError( "Debe Ingresar Descripcion" );
    } else if (tvFecha.getText().toString().trim().equalsIgnoreCase( "00/00/0000" )) {
        tvFecha.setError( "Debe Ingresar Fecha" );
        Toast.makeText( context, this, "Debe Seleccionar Fecha", Toast.LENGTH_SHORT ).show();
    } else {
        monto = Integer.parseInt( edtMonto.getText().toString().trim() );
        descripcion = edtDescripcion.getText().toString().trim();
        fecha = tvFecha.getText().toString().trim();
        item = spItem.getSelectedItem().toString();
    }
}
```

Nuevamente Instanciamos las Clases DbHelper para indicar la base de datos que utilizaremos y la SQLite que en esta ocasión utilizaremos el método getWritableDatabase para escribir sobre ella, así como también crearemos un contenedor de valor a través de la clase ContentValues, en el objeto creado (registro) utilizamos el método put donde indicaremos el nombre de la columna del registro de la base de datos que será insertado, junto a la variable que contiene el valor que será ingresado al objeto, (recordar que el tipo de dato debe coincidir con el que recibirá la base de datos).

```
//se crea la base de datos db_ejercicio
DbHelper dbHelper = new DbHelper( context, this, name: "db_ejercicio", factory: null, version: 1 );

//indicamos que sera de tipo Escritura la consulta "getWritableDatabase"
SQLiteDatabase db = dbHelper.getWritableDatabase();

//creamos contenedor
ContentValues registro = new ContentValues();
//asignamos valores al contenedor
//indicamos el nombre de la columna de la tabla y la variable que contiene el valor que insertaremos
registro.put( "monto_gasto", monto );
registro.put( "descripcion", descripcion );
registro.put( "fecha_dia", dia );
registro.put( "fecha_mes", mes );
registro.put( "fecha_año", año );
registro.put( "id_item", Integer.parseInt( idItem ) );

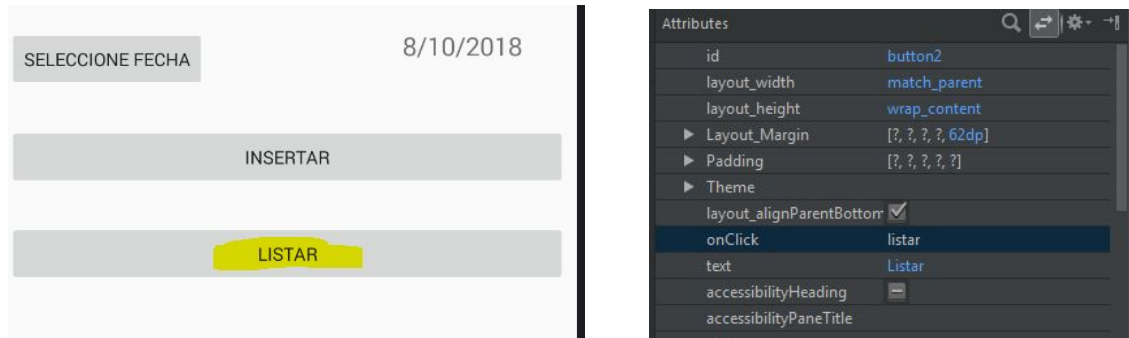
long filas = db.insert( table: "tbl_gasto", nullColumnHack: null, registro );
// Toast.makeText(this,"/n monto "+monto+"/n descripcion "+descripcion+"/n fecha "+fecha+"/n item "+idItem,Toast.LENGTH_SHORT).show();
if (filas > 0) {
    Toast.makeText( context, this, "registro insertado", Toast.LENGTH_SHORT ).show();
} else {
    Toast.makeText( context, this, "Error al insertar", Toast.LENGTH_SHORT ).show();
}
}
```

db.insert() nos permite insertar los datos, indicando la tabla, y el tercer argumento es el objeto Contenedor de valores que creamos previamente.

El método insert nos devolverá la cantidad de filas afectadas con lo cual podemos realizar distintas condiciones y mensajes para indicar si fue o no registrado el dato.

Listar

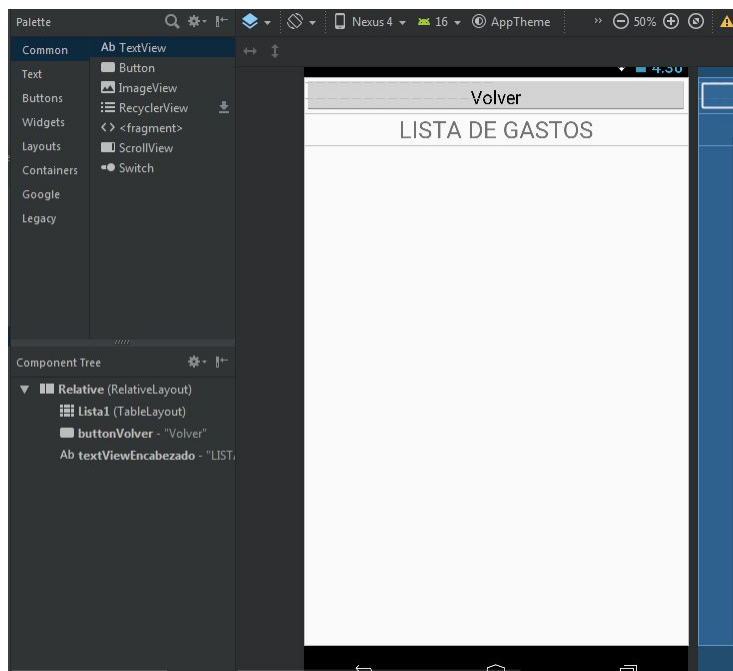
Desde el Main Activity crearemos un método llamado **listar** al cual accederemos mediante un boton LISTAR.



para Listar crearemos otro Main Activity al cual nos transportaremos mediante la clase **Intent**, en el argumento de del constructor se indica el contexto, (this para indicar esta clase) y el segundo argumento es la clase a la cual nos transportaremos es decir el Main activity 2

```
public void listar(View view) {  
    Intent intent=new Intent( packageContext this,Main2Activity.class);  
    startActivity(intent);  
}
```

en este layout Activity 2 crearemos un RelativeLayout y dentro de este un botón seguido de un textview que servira de encabezado para una tabla,ademas agregaremos un **TableLayout** que nos proporciona la estructura de tabla cuando mostremos los datos.



Ahora en la clase activity 2 realizamos algo similar a lo visto en el llenado del Spinner, Instanciando las clases de BD y el cursor

```
package com.example.sony.android_bd;

import ...

public class Main2Activity extends AppCompatActivity {

    private TableLayout tlLista;
    private TextView tvEncab;
    // private String valor;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        tlLista = (TableLayout) findViewById(R.id.Lista1);
        tvEncab = (TextView) findViewById(R.id.textViewEncabezado);

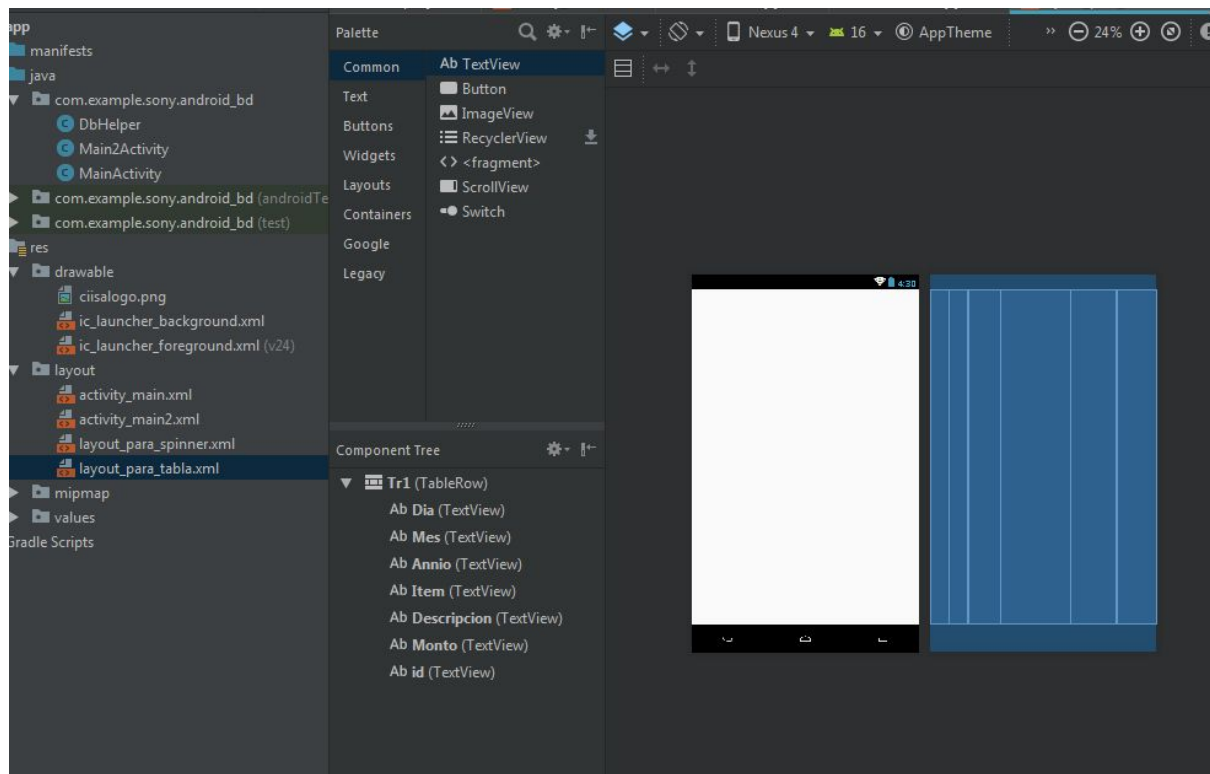
        final DbHelper dbHelper = new DbHelper( context: this, name: "db_ejercicio", factory: null, version: 1);

        //indicamos que sera de tipo lectura la consulta "getReadableDatabase"
        final SQLiteDatabase db = dbHelper.getReadableDatabase();

        //se declara cursor y se le ingresa el valor del metodo rawquery que es un resultSet
        // en db.rawQuery los argumentos son la consulta select y los argumentos para filtrar en este caso null
        final Cursor c = db.rawQuery( sql: "" +
            "SELECT tg.monto_gasto,tg.descripcion,tg.fecha_dia,tg.fecha_mes,tg.fecha_año,ti.nombre_item,tg.id_gasto " +
            "FROM tbl_gasto tg,tbl_item ti " +
            "where ti.id_item=tg.id_item",
            selectionArgs: null);

        //getCount devuelve la cantidad de registros obtenidos en la consulta
        int cantidad= c.getCount();
    }
}
```

Creamos un layout para la Estructura de tabla,para ellos creamos un tablerow y le agregamos textviews que guardaran posteriormente cada columna de los registros que mostraremos



Con las clases View y LayoutInflater podemos traer los componentes de otras Vistas o Layouts a al Activity 2, crearemos un ciclo en el cual en cada vuelta se creará un TableRow y los respectivos TextViews Referenciados a la los layout importados (tomar atención a que el método findViewById proviene el objeto v de la clase View el cual trae el Layout importado),

luego en cada ciclo a partir del segundo (el primero generará los nombres de las columnas) llenaremos los textviews.

[en cada ciclo además avanzamos de registro con el cursor y el método moveToNext()].

Finalmente Agregamos la vista importada a nuestro TableLayout con el método addView.

```
if(cantidad>0) {  
    for (int i = 0; i <= cantidad; i++) {  
        //Importamos la vista del layout creado para la estructura de tabla  
        View v = LayoutInflater.from(this).inflate(R.layout.layout_para_tabla, root: null, attachToRoot: false);  
        //creamos el objeto para la TableRow y le indicamos como referencia el id del tablerow de la vista importada  
        TableRow tr = (TableRow) v.findViewById(R.id.Tr1);  
  
        TextView tvMonto = (TextView) v.findViewById(R.id.Monto);  
        TextView tvDescripcion = (TextView) v.findViewById(R.id.Descripcion);  
        TextView tvFechaDia = (TextView) v.findViewById(R.id.Dia);  
        TextView tvFechaMes = (TextView) v.findViewById(R.id.Mes);  
        TextView tvFechaAnnio = (TextView) v.findViewById(R.id.Annio);  
        TextView tvItem = (TextView) v.findViewById(R.id.Item);  
        TextView tvId = (TextView) v.findViewById(R.id.id);  
  
        if(i==0){  
            tvMonto.setText("MONTO");  
            tvDescripcion.setText("DESCRIPCION");  
            tvFechaDia.setText("DIA");  
            tvFechaMes.setText("MES");  
            tvFechaAnnio.setText("AÑO");  
            tvItem.setText("ITEM");  
        }else{  
            c.moveToNext();  
            tvMonto.setText(String.valueOf(c.getInt( columnIndex: 0)));  
            tvDescripcion.setText(c.getString( columnIndex: 1));  
            tvFechaDia.setText(c.getString( columnIndex: 2));  
            tvFechaMes.setText(c.getString( columnIndex: 3));  
            tvFechaAnnio.setText(c.getString( columnIndex: 4));  
            tvItem.setText(c.getString( columnIndex: 5));  
            tvId.setText(String.valueOf(c.getInt( columnIndex: 6)));  
        }  
  
        tlLista.addView(v);  
    }  
}
```

quedando de la siguiente forma:

Android_bd					
VOLVER					
LISTA DE GASTOS					
DIA	MES	AÑO	ITEM	DESCRIPCION	MONTO
9	10	2018	SUPERMERCADOS	Compras del mes	120000
10	10	2018	AUTOPISTAS	Multas	180000
8	8	2018	VESTUARIO	Traje Formal	180000

Eliminar

Para Eliminar realizaremos un Método `setOnClickListener` dentro del ciclo donde creamos las filas de la tabla anterior, para ello debemos crear una variable auxiliar (valor) que almacenará los valores de id para poder eliminar cada fila a través del método `delete`, que toma argumentos primero, la tabla, segundo condición, y el tercero lo dejaremos en null. con lo anterior ya se elimina un registro, pero debemos refrescar la vista para poder visualizar los cambios, para ellos utilizamos la clase `intent` pero con el método `getIntent()` que toma esta misma clase, por lo tanto me enviara de esta clase a la misma, provocando el refresh.

```

if(i==0){
    tvMonto.setText("MONTO");
    tvDescripcion.setText("DESCRIPCION");
    tvFechaDia.setText("DIA");
    tvFechaMes.setText("MES");
    tvFechaAnio.setText("AÑO");
    tvItem.setText("ITEM");
}else{
    c.moveToNext();
    tvMonto.setText(String.valueOf(c.getInt( columnIndex: 0)));
    tvDescripcion.setText(c.getString( columnIndex: 1));
    tvFechaDia.setText(c.getString( columnIndex: 2));
    tvFechaMes.setText(c.getString( columnIndex: 3));
    tvFechaAnio.setText(c.getString( columnIndex: 4));
    tvItem.setText(c.getString( columnIndex: 5));
    tvId.setText(String.valueOf(c.getInt( columnIndex: 6)));

    //valor de variable auxiliar
    final String valor=tvId.getText().toString();
    tr.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            SQLiteDatabase db2 = dbHelper.getWritableDatabase();
            db2.delete( table: "tbl_gasto", whereClause: "id_gasto="+valor, whereArgs: null);
            Intent intent = getIntent();
            startActivity(intent);
        }
    });
}
tlLista.addView(v);
}
}

```

EJ: al hacer clic en AUTOPISTA se elimina toda la fila

Android_bd						
VOLVER						
LISTA DE GASTOS						
DIA	MES	AÑO	ITEM	DESCRIPCION	MONTO	
9	10	2018	SUPERMERCADOS	Compras del mes	120000	
10	10	2018	AUTOPISTAS	Multas	180000	
8	8	2018	VESTUARIO	Traje Formal	180000	

Android_bd						
VOLVER						
LISTA DE GASTOS						
DIA	MES	AÑO	ITEM	DESCRIPCION	MONTO	
9	10	2018	SUPERMERCADOS	Compras del mes	120000	
8	8	2018	VESTUARIO	Traje Formal	180000	

Nota 1

Nota: para eliminar se utilizó el id de la tabla gasto, sin embargo este no se visualiza ya que existe pero está oculto con el atributo Visibilidad

Tr1 (TableRow)	
Ab Dia (TextView)	
Ab Mes (TextView)	
Ab Anio (TextView)	
Ab Item (TextView)	
Ab Descripcion (TextView)	
Ab Monto (TextView)	
Ab id (TextView)	

Attributes	
id	id
layout_width	wrap_content
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
★ visibility	gone

no está visible

Android_bd						
VOLVER						
LISTA DE GASTOS						
DIA	MES	AÑO	ITEM	DESCRIPCION	MONTO	
9	10	2018	SUPERMERCADOS	Compras del mes	120000	
10	10	2018	AUTOPISTAS	Multas	180000	
8	8	2018	VESTUARIO	Traje Formal	180000	

Nota 2

Además de los códigos para captura de fechas, encontré material de captura de Hora, se dejan ambos para posterior uso.

CAPTURA DE HORA

```

final Calendar c= Calendar.getInstance();
hora=c.get(Calendar.HOUR_OF_DAY);
minutos=c.get(Calendar.MINUTE);

TimePickerDialog timePickerDialog = new TimePickerDialog(this, new
    @Override
    public void onTimeSet(TimePicker view, int hourOfDay, int minute
        ehora.setText(hourOfDay+": "+minute);
    }
),hora,minutos,false);
timePickerDialog.show();

```

CAPTURA DE FECHA

```

public void Calendario(View view){
    //instanciamos Calendar
    Calendar calendario = Calendar.getInstance();

    // obtendremos fecha actual para iniciar calendario
    int dia = calendario.get(Calendar.DAY_OF_MONTH);
    int mes = calendario.get(Calendar.MONTH);
    int anno = calendario.get(Calendar.YEAR);

    // instanciamos metodo DatePickerDialog
    DatePickerDialog datePickerDialog = new DatePickerDialog( context: this, new DatePickerDialog.OnDateSetListener()
        @Override
        public void onDateSet(DatePicker view, int year, int month, int dayOfMonth) {
            tvFecha.setText(dayOfMonth+"/"+(month+1)+"/"+year);
        }
    ),anno,mes,dia); //se le agregan los argumentos año,mes,dia para fecha default sea la actual

    // se muestra el calendario
    datePickerDialog.show();
}

```