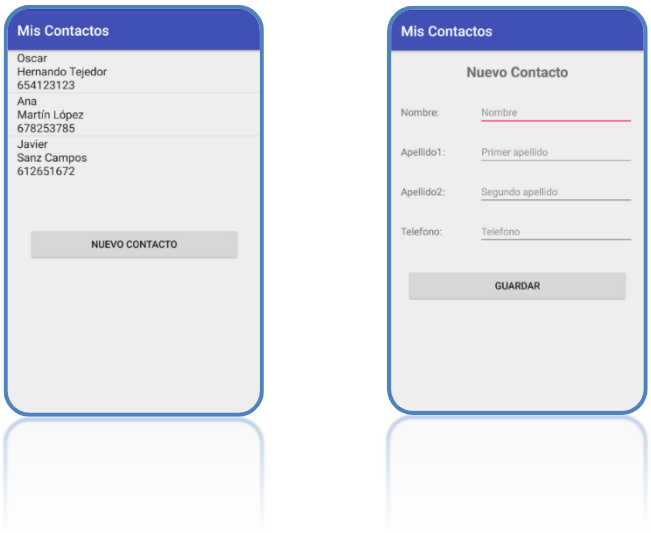
UNIDAD 5: Base de datos (Método 2)

Vamos a ver el funcionamiento de este segundo método creando una aplicación para mantener una lista de contactos muy similar a la anterior, pero utilizando una clase Contacto para gestionar la información de un contacto.

Al igual que la aplicación anterior, ésta estará compuesta de dos actividades:

- ***MainActivity***: mostrará una lista con los contactos actuales y un botón para añadir un nuevo contacto que lanzará la actividad *AddActivity*.
- ***AddActivity***: mostrará un formulario para introducir los datos del contacto y un botón "Guardar" que recogerá los datos del formulario y los almacenará como nuevo contacto en la BD.



CREACIÓN DE LA INTERFAZ

La interfaz va a ser prácticamente idéntica a la de la aplicación anterior, con la única diferencia de que para los elementos de la lista de contactos se va a utilizar un *layout* predefinido.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.example.oscar.exbasedatos02.MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/lvContactos"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="60dp" />

    <Button
        android:id="@+id/botNuevo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Nuevo Contacto"
        android:layout_gravity="center"
        android:layout_marginVertical="8dp"
        android:layout_marginHorizontal="32dp"/>

</LinearLayout>
```

activity_add.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">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="Nuevo Contacto"
        android:textSize="20sp"
        android:textStyle="bold"
        android:layout_margin="16dp"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginHorizontal="16dp"
        android:layout_marginVertical="8dp">
        <TextView
            android:id="@+id/txtNombre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Nombre: "
            android:textSize="14sp"
            android:layout_weight="2"/>
        <EditText
            android:id="@+id/editNombre"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Nombre"
            android:textSize="14sp"
            android:layout_weight="1"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginHorizontal="16dp"
        android:layout_marginVertical="8dp">
        <TextView
            android:id="@+id/txtApellido1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Apellido1: "
            android:textSize="14sp"
            android:layout_weight="2"/>
        <EditText
            android:id="@+id/editApellido1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Primer apellido"
            android:textSize="14sp"
            android:layout_weight="1"/>
    </LinearLayout>
```

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginHorizontal="16dp"
        android:layout_marginVertical="8dp">
        <TextView
            android:id="@+id/txtApellido2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Apellido2: "
            android:textSize="14sp"
            android:layout_weight="2"/>
        <EditText
            android:id="@+id/editApellido2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Segundo apellido"
            android:textSize="14sp"
            android:layout_weight="1"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginHorizontal="16dp"
        android:layout_marginVertical="8dp">
        <TextView
            android:id="@+id/txtTelefono"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Telefono: "
            android:textSize="14sp"
            android:layout_weight="2"/>
        <EditText
            android:id="@+id/editTelefono"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Telefono"
            android:textSize="14sp"
            android:layout_weight="1"/>
    </LinearLayout>

    <Button
        android:id="@+id/botGuardar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Guardar"
        android:layout_gravity="center"
        android:layout_margin="24dp" />

</LinearLayout>
```

CLASE: CONTACTO

La clase "Contacto" es la encargada de gestionar la información de un contacto, de manera que tendrá los atributos propios del contacto: id, nombre, apellido1, apellido2 y teléfono, y los métodos para establecer y obtener la información de cada atributo.

Puede observarse que hay dos constructores:

- `public Contacto()`: constructor vació que crea el objeto pero no establece el valor de ninguno de sus atributos, que se utilizará para crear objetos contacto cuya información se obtendrá de la base de datos.

```
public Contacto() {}
```

- `public Contacto (String nombre, String ape1, String ape2, String tlf)`: recibe como parámetros todos sus atributos excepto el "id", ya que este atributo se va a generar de forma automática al insertar el objeto en la base de datos (autoincrement).

```
public Contacto (String nombre, String ape1, String ape2, String tlf) {
    this.nombre = nombre;
    this.apellido1 = ape1;
    this.apellido2 = ape2;
    this.telefono = tlf;
}
```

Además, se ha sobrescrito el método "*toString*" para facilitar una presentación del objeto en forma de *String* que estará compuesto por 3 líneas con el siguiente formato:

Nombre

Apellido1 Apellido2

Teléfono

```
@Override
public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append(nombre+"\n"+apellido1+" "+apellido2+"\n"+telefono);
    return sb.toString();
}
```

```
public class Contacto {
    private int id;
    private String nombre;
    private String apellido1;
    private String apellido2;
    private String telefono;

    public Contacto() {}

    public Contacto (String nombre, String ape1, String ape2, String tlf) {
        this.nombre = nombre;
        this.apellido1 = ape1;
        this.apellido2 = ape2;
        this.telefono = tlf;
    }

    public int getId() { return this.id; }

    public String getNombre() { return this.nombre; }

    public String getApellido1() { return this.apellido1; }

    public String getApellido2() { return this.apellido2; }

    public String getTelefono() { return this.telefono; }

    public void setId (int id) { this.id = id; }

    public void setNombre (String nombre) { this.nombre = nombre; }

    public void setApellido1 (String ape1) { this.apellido1 = ape1; }

    public void setApellido2 (String ape2) { this.apellido2 = ape2; }

    public void setTelefono (String tlf) { this.telefono = tlf; }

    @Override
    public String toString() {
      StringBuilder sb = new StringBuilder();
      sb.append(nombre+"\n"+apellido1+" "+apellido2+"\n"+telefono);
      return sb.toString();
    }
}
```

CLASE: CONTACTOHELPER

Esta clase extiende a *SQLiteOpenHelper* y tiene como finalidad facilitar la creación y la gestión de las versiones de la Base de datos. Para ello cuenta con los métodos *onCreate* y *onUpdate*.

```
public class ContactoHelper extends SQLiteOpenHelper {

  // Constantes TABLA
  private static final String TABLA_CONTACTOS = "TABLA_CONTACTOS";
  private static final String COL_ID = "ID";
  private static final String COL_NOMBRE = "NOMBRE";
  private static final String COL_APE1 = "APELLIDO1";
  private static final String COL_APE2 = "APELLIDO2";
  private static final String COL_TLF = "TELEFONO";

  // Constante creacion tabla
  private static final String CREATE_BD =
    "CREATE TABLE " + TABLA_CONTACTOS + " ("
    + COL_ID + " INTEGER PRIMARY KEY, "
    + COL_NOMBRE + " TEXT NOT NULL, "
    + COL_APE1 + " TEXT NOT NULL, "
    + COL_APE2 + " TEXT NOT NULL, "
    + COL_TLF + " TEXT NOT NULL);";

  // Constructor
  public ContactoHelper(Context context, String name,
        SQLiteDatabase.CursorFactory factory, int version) {
    super(context, name, factory, version);
  }

  @Override
  public void onCreate(SQLiteDatabase db) {
    // Creamos la BD
    db.execSQL(CREATE_BD);
  }

  @Override
  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
    // Gestionamos las actualizaciones de la BD
    db.execSQL("DROP TABLE IF EXISTS " + TABLA_CONTACTOS);
    onCreate(db);
  }
}
```

CLASE: CONTACTOBD

Esta clase se encarga de gestionar todas las operaciones que vamos a realizar con los datos de nuestra base de datos.

Para ello se crean constantes a todos los elementos de la BD (bd, tablas, campos, etc.), y se definen métodos para todas las operaciones a realizar: insertar, eliminar, actualizar, consultar...

Además, cuenta con los métodos *openForRead* y *openForWrite* que nos van a permitir abrir la base de datos para leer datos, y para leer y escribir respectivamente.

```
public class ContactoBD {

  // Constantes BD
  private static final int VERSION = 1;
  private static final String BD_NOMBRE = "contactos.db";

  // Constantes TABLA
  private static final String TABLA_CONTACTOS = "TABLA_CONTACTOS";
  private static final String COL_ID = "ID";
  private static final int NUM_COL_ID = 0;
  private static final String COL_NOMBRE = "NOMBRE";
  private static final int NUM_COL_NOMBRE = 1;
  private static final String COL_APE1 = "APELLIDO1";
  private static final int NUM_COL_APE1 = 2;
  private static final String COL_APE2 = "APELLIDO2";
  private static final int NUM_COL_APE2 = 3;
  private static final String COL_TLF = "TELEFONO";
  private static final int NUM_COL_TLF = 4;

  // Variables gestión BD
  private SQLiteDatabase bd;
  private ContactoHelper contactos;


  public ContactoBD (Context context) {
    contactos = new ContactoHelper(context, BD_NOMBRE, null, VERSION);
  }

  public void openForWrite() {
    bd = contactos.getWritableDatabase();
  }

  public void openForRead() {
    bd = contactos.getReadableDatabase();
  }

  public SQLiteDatabase getBD() {
    return bd;
  }

  public int eliminarContacto (int id) {
    String where = COL_ID + " = " + Integer.toString(id);
    return bd.delete(TABLA_CONTACTOS, where, null);
  }
```

```
public long insertarContacto (Contacto contacto) {
    ContentValues content = new ContentValues();
    content.put(COL_NOMBRE, contacto.getNombre());
    content.put(COL_APE1, contacto.getApellido1());
    content.put(COL_APE2, contacto.getApellido2());
    content.put(COL_TLF, contacto.getTelefono());
    return  bd.insert(TABLA_CONTACTOS, null, content);
  }

  public int actualizarContacto (int id, Contacto contacto) {
    ContentValues content = new ContentValues();
    content.put(COL_NOMBRE, contacto.getNombre());
    content.put(COL_APE1, contacto.getApellido1());
    content.put(COL_APE2, contacto.getApellido2());
    content.put(COL_TLF, contacto.getTelefono());
    String where = COL_ID + " = " + id;
    return  bd.update(TABLA_CONTACTOS, content, where, null);
  }

  public Contacto obtenerContacto (int id) {
    String where = COL_ID + " = " + Integer.toString(id);
    String campos[] = new String[] {COL_ID,COL_NOMBRE,COL_APE1,COL_APE2,COL_TLF};
    Cursor cur = bd.query(TABLA_CONTACTOS, campos, where, null,null,null, COL_ID);
    Contacto contacto;
    if (cur.getCount() == 0) {
      contacto = null;
    } else {
      cur.moveToFirst();
      contacto = new Contacto();
      contacto.setId(cur.getInt(NUM_COL_ID));
      contacto.setNombre(cur.getString(NUM_COL_NOMBRE));
      contacto.setApellido1(cur.getString(NUM_COL_APE1));
      contacto.setApellido2(cur.getString(NUM_COL_APE2));
      contacto.setTelefono(cur.getString(NUM_COL_TLF));
    }
    cur.close();
    return contacto;
  }

  public ArrayList<Contacto> obtenerTodosContactos() {
    String campos [] = new String[] {COL_ID,COL_NOMBRE,COL_APE1,COL_APE2,COL_TLF};
    Cursor cur = bd.query(TABLA_CONTACTOS, campos, null, null, null, null, COL_ID);
    ArrayList<Contacto> listaContactos;
    if (cur.getCount() == 0) {
      listaContactos = null;
    } else {
      listaContactos = new ArrayList<Contacto>();
      while (cur.moveToNext()) {
        Contacto contacto = new Contacto();
        contacto.setId(cur.getInt(NUM_COL_ID));
        contacto.setNombre(cur.getString(NUM_COL_NOMBRE));
        contacto.setApellido1(cur.getString(NUM_COL_APE1));
        contacto.setApellido2(cur.getString(NUM_COL_APE2));
        contacto.setTelefono(cur.getString(NUM_COL_TLF));
        listaContactos.add(contacto);
      }
    }
    cur.close();
    return listaContactos;
  }

  public void close() {
    bd.close();
  }

}
```

CLASE: MAINACTIVITY

La clase "*MainActivity*" es casi idéntica a la de la práctica anterior salvo el método "listar" que va a ser mucho más simple, porque todos los accesos a la BD están encapsulados en la clase "ContactoBD".

Como puede verse basta con crear un objeto ContactoBD, abrir la BD para lectura (*openForRead*), y llamar al método "obtenerTodosContactos", para obtener un *ArrayList* con todos los contactos de nuestra BD. A partir de ahí sólo queda darle formato para mostrarlos en pantalla utilizando un *ListView*.

```
private void listar() {
  ContactoBD contactoBD = new ContactoBD(this);
  contactoBD.openForRead();
  ArrayList<Contacto> lista = contactoBD.obtenerTodosContactos();
  contactoBD.close();

  if (lista == null) {
    String mensa = "Lista vacía";
    int duracion = Toast.LENGTH_SHORT;
    Toast.makeText(getApplicationContext(), mensa, duracion).show();
  } else {
    ArrayAdapter<Contacto> adaptador = new ArrayAdapter<Contacto>
        (this,android.R.layout.simple_list_item_1, lista);;
    ListView lvContactos = (ListView)findViewById(R.id.lvContactos);
    lvContactos.setAdapter(adaptador);
  }
}
```

Además, se ha añadido un método "vaciar" por si en algún momento fuera necesario eliminar todos los registros de la tabla "Contactos".

```
private void vaciar() {
  ContactoBD contactoBD = new ContactoBD(getApplicationContext());
  contactoBD.openForWrite();
  ArrayList<Contacto> lista = contactoBD.obtenerTodosContactos();
  int id;
  for (int i=0; i<lista.size(); i++) {
    id = lista.get(i).getId();
    contactoBD.eliminarContacto(id);
  }
  contactoBD.close();
}
```

La clase "*MainActivity*" completa quedaría así:

```
public class MainActivity extends AppCompatActivity {

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

    // Mostramos el contenido de la BD en el ListView
    listar();

    // Añadir Contacto
    Button botNuevo = (Button)findViewById(R.id.botNuevo);
    botNuevo.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Intent intento = new Intent(MainActivity.this, AddActivity.class);
        startActivity(intento);
      }
    });
  }

  @Override
  protected void onRestart() {
    super.onRestart();
    listar();
  }

  private void listar() {
    ContactoBD contactoBD = new ContactoBD(getApplicationContext());
    contactoBD.openForRead();
    ArrayList<Contacto> lista = contactoBD.obtenerTodosContactos();
    contactoBD.close();

    if (lista == null) {
      String mensa = "Lista vacía";
      int duracion = Toast.LENGTH_SHORT;
      Toast.makeText(getApplicationContext(), mensa, duracion).show();
    } else {
      ArrayAdapter<Contacto> adaptador = new ArrayAdapter<Contacto>
          (this,android.R.layout.simple_list_item_1, lista);
      ListView lvContactos = (ListView)findViewById(R.id.lvContactos);
      lvContactos.setAdapter(adaptador);
    }
  }

  private void vaciar() {
    ContactoBD contactoBD = new ContactoBD(getApplicationContext());
    contactoBD.openForWrite();
    ArrayList<Contacto> lista = contactoBD.obtenerTodosContactos();
    int id;
    for (int i=0; i<lista.size(); i++) {
      id = lista.get(i).getId();
      contactoBD.eliminarContacto(id);
    }
    contactoBD.close();
    }
}
```

CLASE: ADDACTIVITY

La clase "*AddActivity*" también es casi idéntica a la de la práctica anterior, salvo en el acceso a la BD que es mucho más sencillo.

```
public class AddActivity extends AppCompatActivity {

  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_add);

    // Guardar datos en la BD
    Button botGuardar = (Button)findViewById(R.id.botGuardar);
    botGuardar.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        EditText editNombre = (EditText)findViewById(R.id.editNombre);
        EditText editApe1 = (EditText)findViewById(R.id.editApellido1);
        EditText editApe2 = (EditText)findViewById(R.id.editApellido2);
        EditText editTlf = (EditText)findViewById(R.id.editTelefono);
        String nombre = editNombre.getText().toString().trim();
        String ape1 = editApe1.getText().toString().trim();
        String ape2 = editApe2.getText().toString().trim();
        String tlf = editTlf.getText().toString().trim();
        String mensa = "";
        int dur = Toast.LENGTH_SHORT;

        if (!nombre.equals("") && !ape1.equals("") && !ape2.equals("")
            && !tlf.equals("")) {
          // Creamos un contacto y le asignamos valores
          Contacto contacto = new Contacto(nombre, ape1, ape2, tlf);
          ContactoBD contactoBD = new ContactoBD(getApplicationContext());
          contactoBD.openForWrite();
          if (contactoBD.insertarContacto(contacto) != -1) {
            // Alerta: contacto añadido
            mensa = "Contacto añadido";
            Toast.makeText(getApplicationContext(), mensa, dur).show();
            editNombre.setText("");
            editApe1.setText("");
            editApe2.setText("");
            editTlf.setText("");
          } else {
            // Alerta: contacto añadido
            mensa = "No se ha podido añadir el contacto";
            Toast.makeText(getApplicationContext(), mensa, dur).show();
          }
          contactoBD.close();
        } else {
          // Alerta: faltan datos
          mensa = "Faltan datos";
          Toast.makeText(getApplicationContext(), mensa, dur).show();
        }
      }
    });
  }
}
```

AMPLIACIÓN

Consulta la API de Android y completa esta documentación explicando cómo funcionan los métodos: *insert*, *delete*, *update* y *query*, utilizados en la clase "ContactoDB".

Explica también qué es la clase "*ContentValues*" y cómo funciona su método *put*.