Escuela Técnica Superior de Ingeniería Informática

Memoria de la aplicación Android *rSalud*

Alumna: Rocío Leal Díaz

Asignatura: Gestión de la Información y tecnologías Informáticas

Grado: Ingeniería de la Salud (4º Año)

Contenido

Introducción

Objetivo

El objetivo de este trabajo ha sido la creación de una aplicación en el entorno de desarrollo Android Studio. Con el mismo, se pretendía familiarizarse con la plataforma Android y, además, adquirir conocimiento para el desarrollo de aplicaciones futuras.

Además, se adquirieron conocimientos sobre XML y manipulación de la Base de Datos SQLite.

Descripción de la aplicación

La aplicación *rSalud* ha sido diseñada para la gestión de pacientes de los médicos de un centro de salud. Contiene una interfaz principal muy sencilla, en la cual se puede elegir entre 4 botones que hacen referencia a 4 funciones distintas:

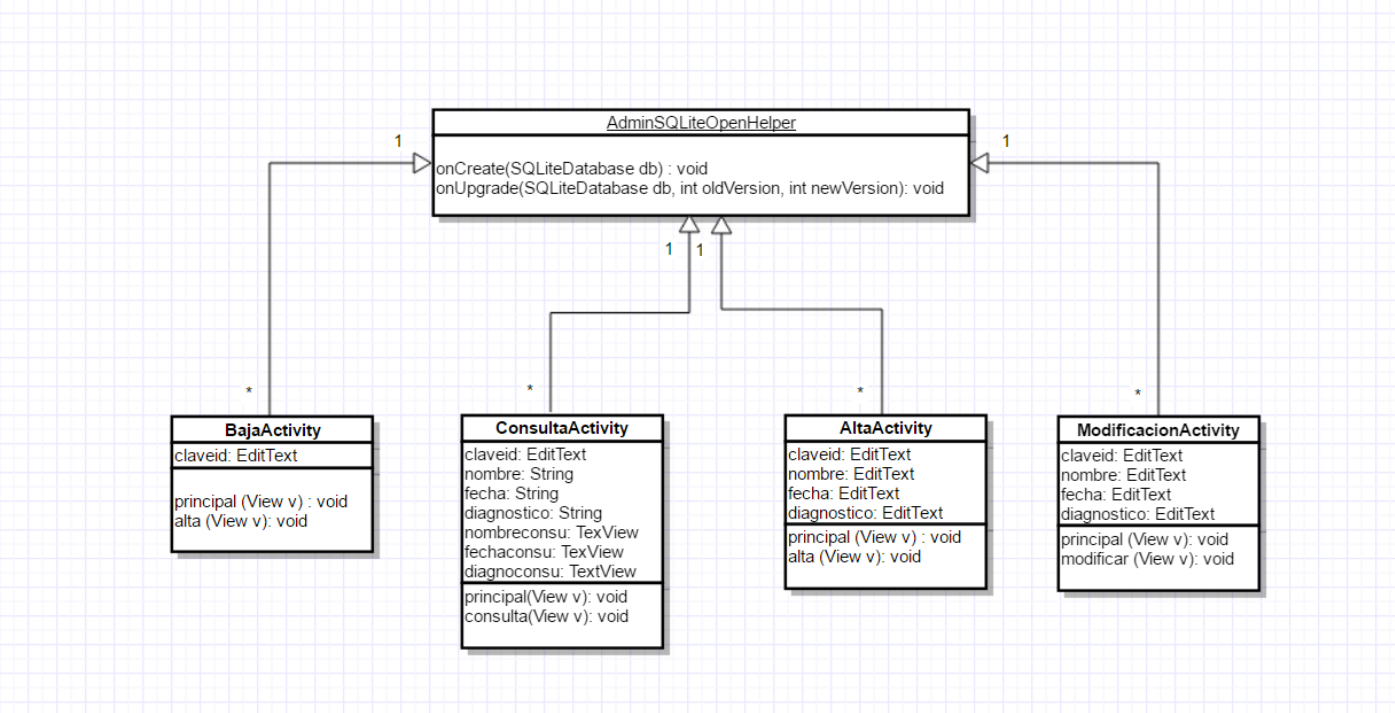
- Dar de alta a un paciente:
  Esta función permite la creación nuevos pacientes, los cuales se irán agregando como filas de la tabla creada en la base de datos.
  Cada paciente contiene:
  - Clave única de paciente: Es una clave que el médico le asigna para identificar al paciente en cuestión entre todos. Podría considerarse el código NHUSA, con la restricción que solo puede introducirse caracteres numéricos.
  - Nombre del paciente
  - Edad del paciente
  - Diagnóstico: En este apartado se incorpora toda la información de diagnóstico.

- Dar de baja a un paciente:
  Esta función elimina a los pacientes existentes en la tabla de la base de datos. Para ello debe introducir la clave del paciente y se borrará toda la fila de la tabla.

- Modificar datos a un paciente:
  Esta función permite actualizar toda la información del paciente desde el nombre hasta el diagnóstico; Para ello, se introduce la clave de identificación del paciente y se introduce de nuevo todos los datos con las modificaciones pertinentes.

- Consultar datos de un paciente:
  Esta función permite consultar todos los datos de un paciente mediante su clave de identificación.

Diagrama UML de clases



RSalud

Interfaces

Como se ha comentado anteriormente, la aplicación cuenta con 4 funciones y por lo tanto, cada función tiene su interfaz.

En ese apartado se comentará brevemente los componentes de cada interfaz.

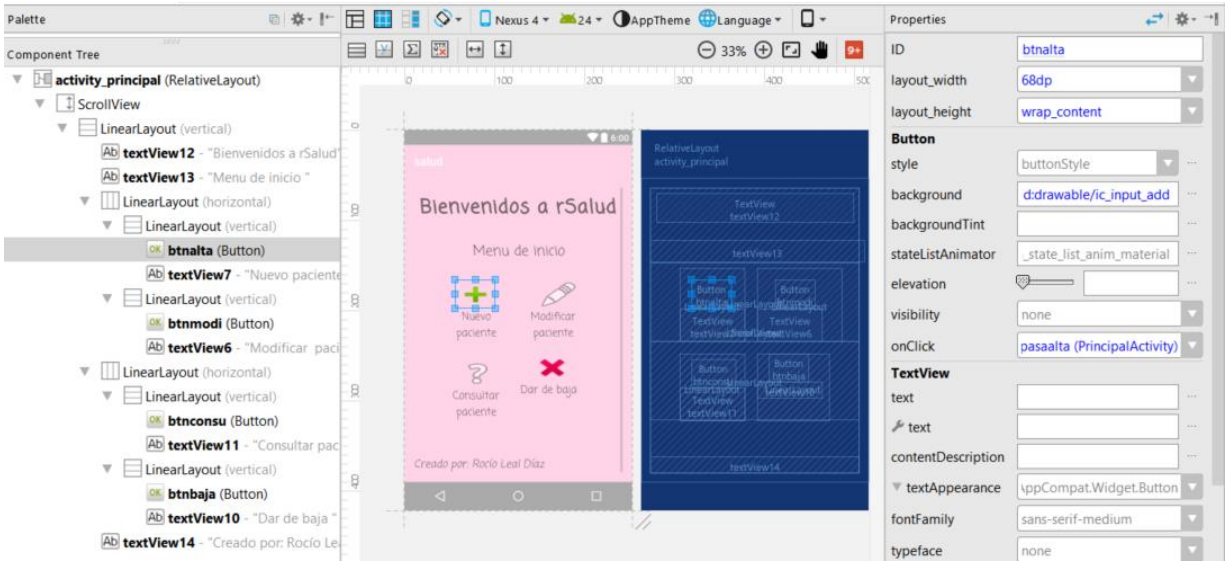
- Interface Principal
  Esta interfaz recibe el nombre de *activity_principal* en la sección de los layout. Como podemos ver en la imagen siguiente, la interfaz está formada por:

  - Un ScrollView que ayuda a que todo el contenido se ajuste a la pantalla del teléfono ya sea en horizontal o en vertical.

  - 3 LinearLayout principales:
    - El primer LinearLayout contiene 3 textView que sirven para poner el mensaje de bienvenida, el mensaje de menú y el mensaje de autor; además, también contiene los 2 LinearLayout principales siguientes.

- Los dos LinearLayout restantes, cada uno contiene dos LinearLayout y a su vez cada uno contiene un button y un texView. Con esto se consigue que cada botón tenga debajo para qué sirve.

Cada uno de los botones creados, están asociados a una función de la clase *PrincipalActivity* indicada en el OnClick de cada uno de ellos. En la imagen siguiente, se puede ver como el botón de "Nuevo paciente" tiene asociado el método "pasaalta".

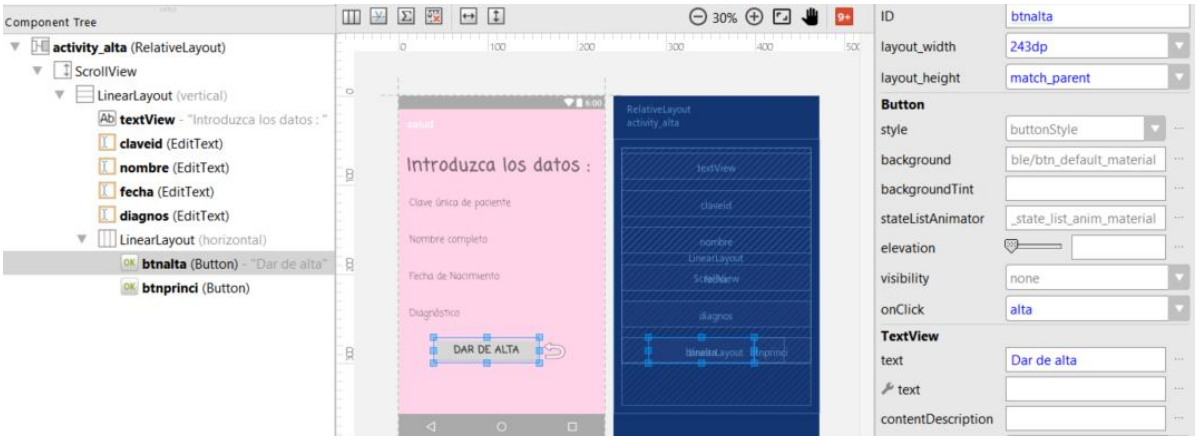


- Interface de Alta de paciente

  Esta interfaz recibe el nombre de *activity_alta* en la sección de los layout. Como podemos ver en la imagen siguiente, la interfaz está formada por:

  - Un ScrollView.

  - Un LinerLayout principal que contiene:
    - Un textView para indicar la sección.
    - 4 EditText, cada uno con su ID para poder recoger los datos que el usuario escriba en la clase de dicha interfaz.
    - Un LinearLayout con 2 botones y cada uno con su función descrita en la clase de la interfaz.

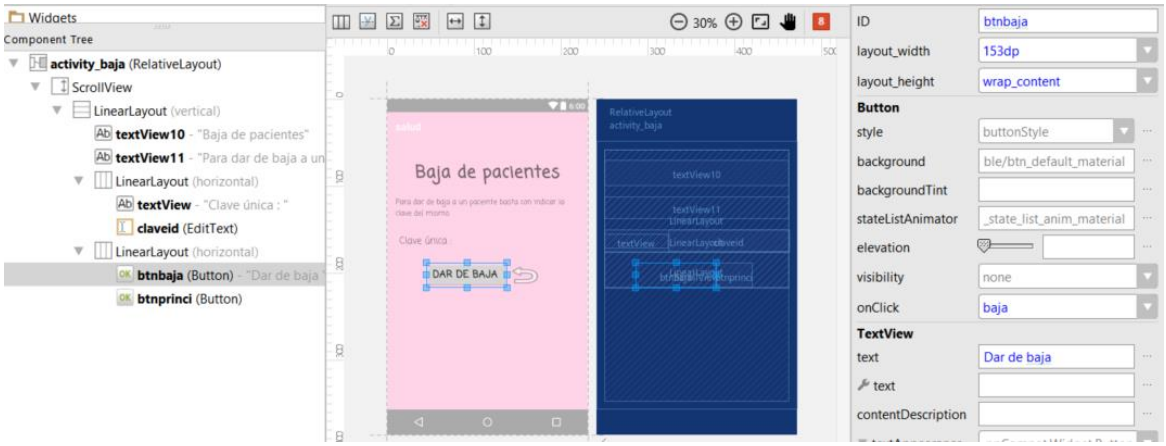


- Interface de Baja de paciente

  Esta interfaz recibe el nombre de *activity_baja* en la sección de los layout. Como podemos ver en la imagen siguiente, la interfaz está formada por:

  - Un ScrollView.

  - Un LinerLayout principal que contiene:
    - Dos textView uno para indicar la sección y otro como mensaje informativo acerca de la baja de los pacientes
    - Dos LinearLayout, uno contiene un TextView y un EditText para recoger la clave de identificación del paciente que se tiene que dar de baja y otro, que contiene los botones de la interface, en concreto, un botón de regresar al menú principal y otro de dar de baja.

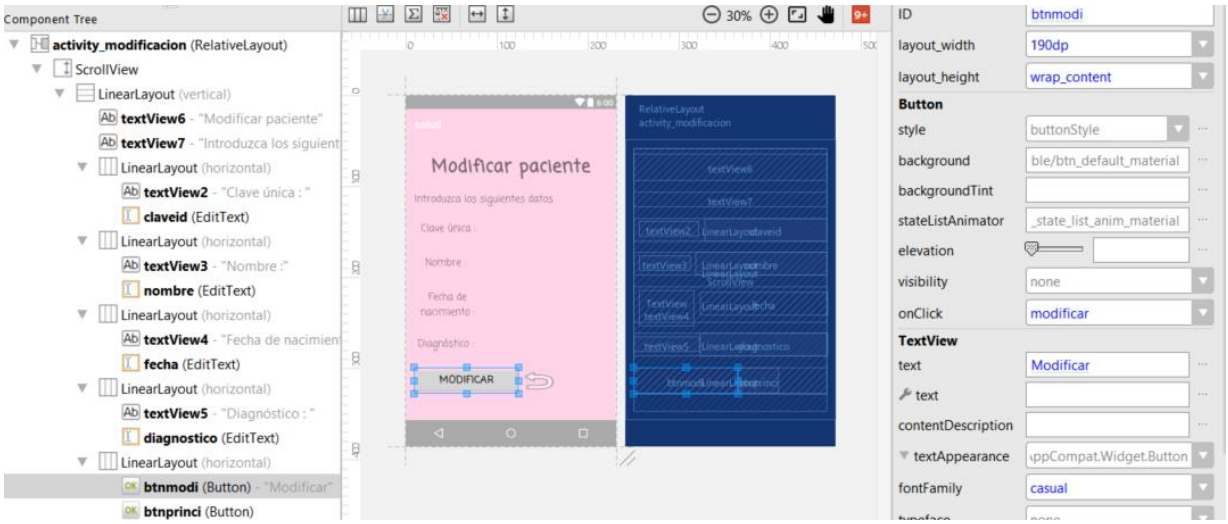


- Interface de Modificación de paciente

  Esta interfaz recibe el nombre de *activity_modificacion* en la sección de los layout. Como podemos ver en la imagen siguiente, la interfaz está formada por:

- Un ScrollView.

- Un LinerLayout principal que contiene:
  - 2 TextViews para introducir la sección y un texto informativo.
  - 5 LinearsLayouts, 4 con un TextView y un editText para recoger los nuevos parámetros y otro con dos botones, uno para volver al menú principal y otro para modificar al paciente.

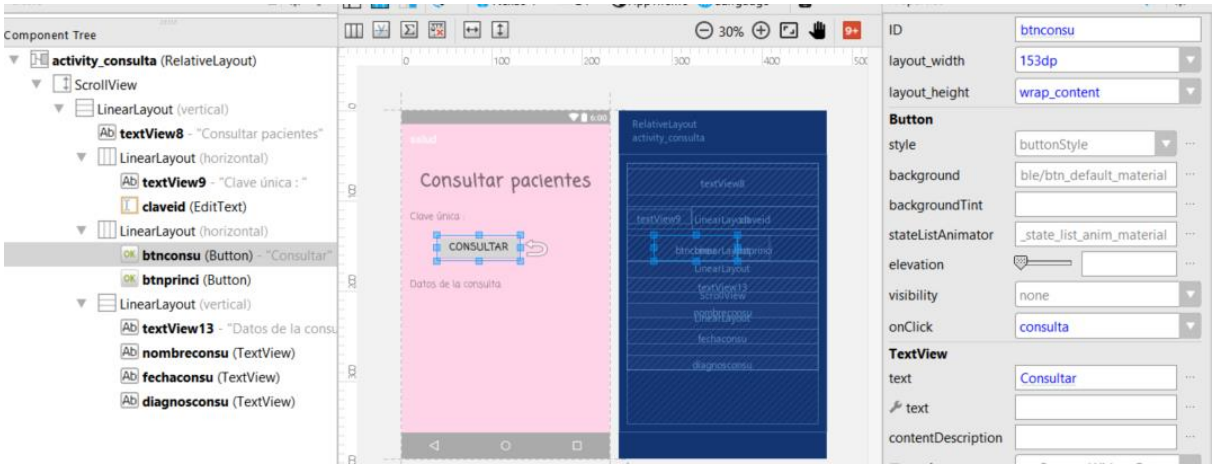


- Interface de Consulta de paciente

  Esta interfaz recibe el nombre de *activity_consulta* en la sección de los layout. Como podemos ver en la imagen siguiente, la interfaz está formada por:

  - Un ScrollView.

  - Un LinerLayout principal que contiene:
    - Un textView para indicar el nombre de la sección
    - 3 LinearLayaout; el primero sirve para recoger la clave del paciente que servirá para buscarlo, está compuesto por un textView y un EditText; el segundo contiene dos botones, uno para volver al menú principal y otros para realizar la búsqueda; el ultimo linearlayout está compuesto por 4 textViews en los que se mostrará la información de la consulta.



Clases

En este apartado, se comentará las clases asociadas a cada una de las interfaces anteriores y los métodos que contienen.

- Clase Principal

  Está relacionada con la interfaz principal.

  Contiene los métodos necesarios para que, al pulsar sobre algunos de los botones, se dirija a la interfaz de casa uno.

PrincipalActivity.java

```
public class PrincipalActivity extends AppCompatActivity {

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

    //Boton Alta: al pulsarlo nos dirigimos a la clase/interface Alta
    public void pasaalta (View v){
        Intent i = new Intent(this, AltaActivity.class);
        startActivity(i);
    }
    //Boton Bajaa: al pulsarlo nos dirigimos a la clase/interface Baja
    public void pasabaja (View v){
        Intent i = new Intent(this, BajaActivity.class);
        startActivity(i);
    }
    //Boton Modificar: al pulsarlo nos dirigimos a la clase/interface Modificar
    public void pasamodi (View v){
        Intent i = new Intent(this, ModificacionActivity.class);
        startActivity(i);
    }
    //Boton Consultar: al pulsarlo nos dirigimos a la clase/interface Cosultar
    public void pasaconsu (View v){
        Intent i = new Intent(this,ConsultaActivity.class);
        startActivity(i);
    }
}
```

- Clase Alta de paciente

  Está relacionada con la interfaz alta de paciente.

  Contiene los métodos para los dos botones de la interfaz:

  - Botón volver al menú principal.
  - Botón dar de alta al paciente:

    Para este botón, su código no es tan fácil como para el botón anterior.

    En primer lugar, fuera del botón se declara las variables que se van a usar y se indica que los datos introducidos en los EditTexts presentes en la interfaz se guarden en las mismas a través de sus IDs.

    En segundo lugar, dentro del botón, se indica que si algún campo de la interfaz está vacío no se puede continuar hasta que estén todos los campos llenos. Después, se conecta con la base de datos y para recuperar la información de la misma, se debe usar un objeto de tipo Cursor en el que se le mete una secuencia sql indicando que la primary key para relacionar será el dato claveid. Posteriormente, en un contenedor se recogerá por columnas de la tabla el valor a añadir y se insertará en la tabla.

    Por último, se cierra la conexión con la base de datos y se ponen todos los EditText de la interfaz vacíos por si el usuario quiere agregar más de un paciente a la vez.

  Para esta clase no se proporcionará imagen del código ya que es muy extenso y ocuparía mucho espacio. Si se quiere ver el código de las clases, se tendrá que cargar el fichero en Android o en la carpeta de la aplicación adjuntada, buscar la clase en cuestión.

- Clase Baja de paciente

  Está relacionada con la interfaz baja de paciente.

  Contiene los métodos para los dos botones de la interfaz:

  - Botón volver al menú principal.
  - Botón dar de baja al paciente:

    En primer lugar, se declara fuera del método del botón una variable de tipo EditText que servirá para recoger la claveid que se introduce en la interfaz.

    En segundo lugar, dentro del botón, se indica que la sección de la clave no puede estar vacía. A continuación, se conecta con la base de datos y se indica que se le elimine la fila de la tabla que contenga la claveid introducida. Esta acción devuelve un valor de tipo entero que se recogerá en la variable conteo; Si esta variable es igual a 1 se lanzara un mensaje de que el paciente fue eliminado con existo.

  Del mismo modo que la clase anterior, no se proporcionará una imagen de la clase debido a su extensión.

- Clase Modificación de paciente

  Está relacionada con la interfaz modificación de paciente.

  Contiene los métodos para los dos botones de la interfaz:

  - Botón volver al menú principal.
  - Botón modificación de paciente:

    En primer lugar, se declara fuera del método del botón 4 variables de tipo EditTexts que servirán para recoger los datos que se introducen en la interfaz.

    En segundo lugar, dentro del botón, se indica que los apartados a rellenar con la información nueva no pueden quedar vacíos. A continuación, se conecta con la base de datos y de manera similar a la clase alta de paciente, se recogen los datos en un contenedor para actualizar la base de datos a través del mismo. Una vez actualizada la base de datos este devuelve un valor entero que si es igual a uno se lanzara un mensaje de actualización con éxito del paciente.

Del mismo modo que la clase anterior, no se proporcionará una imagen de la clase debido a su extensión.

- Clase Consulta de paciente

  Está relacionada con la interfaz consulta de paciente.

  Contiene los métodos para los dos botones de la interfaz:

  - Botón volver al menú principal.
  - Botón consulta de paciente:

    En primer lugar, se declara fuera del método del botón una variable del tipo EditText para recoger la claveid, 3 variables del tipo String necesarias para recoger los datos de la consulta y 3 variables del tipo TextView para mostrar los datos de la consulta.

    En segundo lugar, dentro del botón, se indica que el apartado de la clave no puede quedar vacío, ya que sin la clave no podrá realizarse ninguna consulta. A continuación, conectamos con la base de datos y recogemos la información de la consulta en un objeto de tipo Cursor. Después, guardamos la información del cursor en las variables declaradas para recoger la información y por último, indicamos a las variables de tipo TextView que muestren la información de las variables anteriores. En caso de que el paciente no haya sido dado de alta, se lanzará un mensaje para decir que no existe.

Del mismo modo que la clase anterior, no se proporcionará una imagen de la clase debido a su extensión.

Base de datos

La base de datos escogida para el desarrollo de esta aplicación ha sido SQLite ya que es una de las herramientas que proporciona directamente Android para el almacenamiento y consulta de datos estructurados.

SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características como su pequeño tamaño, no necesitar servidor, precisar poca configuración, ser transaccional y ser de código libre.

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una API para llevar a cabo de manera sencilla todas las tareas necesarias.

En Android, la forma típica para crear, actualizar, y conectar con una base de datos SQLite es a través de una clase auxiliar llamada SQLiteOpenHelper. La clase SQLiteOpenHelper tiene sólo un constructor, que normalmente no necesitaremos sobrescribir, y dos métodos abstractos, onCreate() y onUpgrade(), que han sido personalizados con el código necesario para crear nuestra base de datos y para actualizar su estructura respectivamente.

En el apartado Clase de la base de datos, podemos ver el código desarrollado para crear la base de datos. La sentencia sql que se le pasa al método onCreate, indica que se cree una tabla de nombre paciente que contenga las columnas de nombre, fecha, diagnóstico y claveid como primary key.

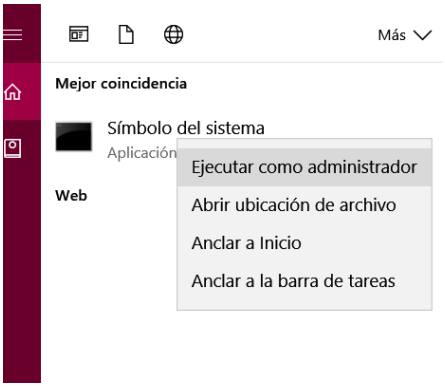
Clase de la base de datos

```
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper {
    public AdminSQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //Creamos la tabla con las columnas en la base de datos
        db.execSQL("CREATE TABLE pacientes(claveid INTEGER primary key, nombre TEXT, fecha TEXT, diagnostico TEXT)");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("drop TABLE if exists pacientes");
        db.execSQL("CREATE TABLE pacientes(claveid INTEGER primary key, nombre TEXT, fecha TEXT, diagnostico TEXT)");
    }
}
```

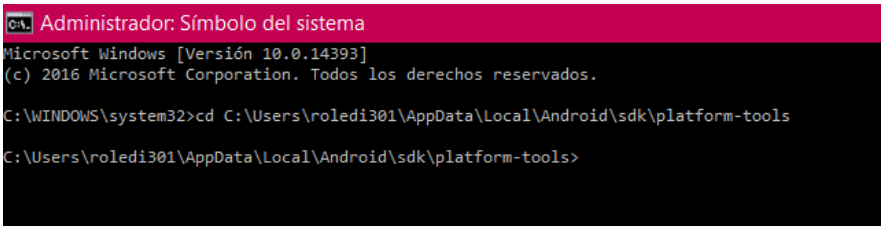
Información de la base de datos

Para consultar la base de dato SQLite creada por Android tendremos que seguir los pasos descritos a continuación:

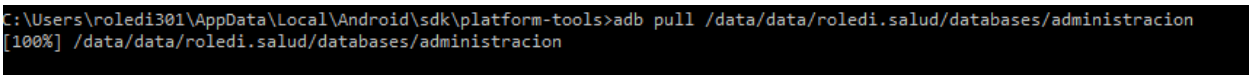
1. Ejecutamos la consola en modo administración.



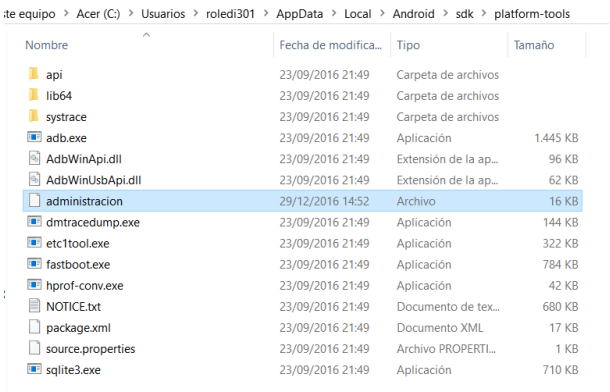
2. Movernos a la ruta donde que encuentra la carpeta plataform tools de Android. En mi caso la ruta es: C:\Users\roledi301\AppData\Local\Android\sdk\platform-tools



3. Ejecutar la orden: adb root
4. Ejecutar la orden: pull /data/data/paqueteDeLaAplicacion/databases/nombreDB. En mi caso la orden es: adb pull /data/data/roledi.salud/databases/administración.



5. Buscar en la ruta del paso 2 un archivo que tenga de nombre administración. Seleccionar el archivo y abrirlo con la herramienta Excel o Wordpad.



6. La información no se visualizará con el formato de base de datos pero como el objetivo es ver la información que contiene, se puede realizar un procesamiento de datos a posteriori que ayude a la visualización del contenido.

```
Juan Lopez 03/12/1956 1 Bandera, Temblores moderados
BegoÃ±a Orozco  07/08/1978 Sana
```

Bibliografía

1. http://www.sgoliver.net/blog/curso-de-programacion-android/
2. http://www.sgoliver.net/blog/bases-de-datos-en-android-i-primeros-pasos/
3. http://www.hermosaprogramacion.com/2014/10/android-sqlite-bases-de-datos/
4. https://androidayuda.com/tutorial-de-ayuda-y-primeros-pasos/
5. http://www.elandroidelibre.com/2013/02/diez-comandos-de-adb-que-deberias-conocer.html
6. https://developer.android.com/guide/index.html?hl=es
7. https://developer.android.com/training/basics/data-storage/databases.html?hl=es
8. http://stackoverflow.com/questions/12995030/how-to-use-adb-pull-command