UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA MECÁNICA

"DESARROLLO DE UNA APLICACIÓN ANDROID PARA PERSONAS DIABÉTICAS"

AUTOR: Guillermo Guirao Rodríguez

DIRECTOR: César Fernández Peris

INDICE

CAPÍTULO 1: INTRODUCCIÓN

Hablemos de la diabetes.

¿Qué es la diabetes?

La diabetes es una enfermedad crónica que se origina porque el páncreas no sintetiza la cantidad de insulina que el cuerpo humano necesita, la elabora de una calidad inferior o no es capaz de utilizarla con eficacia.

La insulina es una hormona producida por el páncreas. Su principal función es el mantenimiento de los valores adecuados de glucosa en sangre. Permite que la glucosa entre en el organismo y sea transportada al interior de las células, en donde se transforma en energía para que funcionen los músculos y los tejidos. Además, ayuda a que las células almacenen la glucosa hasta que su utilización sea necesaria.

La glucosa elevada puede ser perjudicial para todo el organismo, pero principalmente para el corazón, el riñón y las arterias, por lo que las personas que tienen diabetes y no lo saben o no la tratan tienen más riesgo de problemas renales, infartos, pérdida de visión y amputaciones de miembros inferiores.

Actualmente el porcentaje de personas con diabetes en España es de un 13.8%.

1.1 CONTEXTO Y JUSTIFICACIÓN DEL PROYECTO

La vida de un diabético requiere de un constante seguimiento de su enfermedad, es crucial que se adquiera un hábito diario en el que el paciente se realice controles glucémicos a lo largo del día para controlar cuáles son sus niveles de glucosa, a su vez, es necesario llevar un control exhaustivo de las inyecciones de insulina que debe aplicarse cada día.

El objetivo de esta aplicación es facilitar el control de su enfermedad a cualquier persona diabética:

-Ayudarle a organizar y recordar cuando debe aplicarse la próxima inyección de insulina.

-Guardar los registros de sus controles glucémicos diarios.

-Contactar de forma rápida con los servicios de emergencias o con algún otro contacto del usuario en caso de una urgencia.

La diabetes es una enfermedad que puede causar daños muy graves en el paciente si no se trata de forma adecuada llevando un control riguroso y continuo a lo largo de su vida. Es por ello por lo que nuestra aplicación alcanza su gran utilidad. Los pacientes diabéticos acaparan todo rango de edades, desde niños pequeños hasta personas de la tercera edad. La simplicidad y la facilidad con la que nuestra aplicación trabaja, hace de ella una herramienta muy útil para cualquier diabético y sobre todo para aquellas personas que no tengan nociones muy avanzadas en el uso y manejo de dispositivos electrónicos.

Diabeticday es una aplicación muy intuitiva y muy sencilla de utilizar por lo que nuestros pacientes más jóvenes e incluso los más adultos (que suelen mostrar rechazo a la hora de usar dispositivos electrónicos y herramientas más actuales) no van a tener ningún problema en usar de forma muy guiada y sencilla nuestra aplicación.

En el mercado existen numerosas aplicaciones enfocadas para ayudar a las personas diabéticas en su día a día, con infinidad de actividades, opciones y configuraciones, pero ¿por qué usar Diabeticday en lugar de cualquier otra de las disponibles en el mercado?

Si tuviésemos que definir el rango de edades para el que está enfocado nuestra aplicación, éste abarcaría desde los 10 años hasta los 90 años de edad, y el motivo de esto es que no requiere de ningún tipo de conocimiento en dispositivos electrónicos y teléfonos móviles, lo cual hace que nuestra aplicación sea una herramienta muy sencilla y de uso muy intuitivo, consiguiendo que el paciente se sienta muy cómodo al usarla y que cualquier persona pueda darle un uso adecuado de forma muy rápida.

1.2 OBJETIVOS

Aprender a programar en Android, Programación orientada a objetos, manejo de la base de datos embebida en Android.

1.3 OBJETIVOS ESPECÍFICOS

- Establecer alarma para la siguiente toma
- Guardar valores de una toma
- Registro histórico de las tomas realizadas
- Abrir el dial de llamada con número guardado en memoria
- Establecer número de contacto de emergencia

CAPÍTULO 2: DISEÑO DEL PROYECTO

2.1 ANÁLISIS DE LOS USUARIOS Y PÚBLICO OBJETIVO

Esta aplicación está pensada para toda persona de entre 10 años y 90 años que sufren diabetes. Muy recomendada para todos aquellos usuarios que no dominen con soltura las nuevas tecnologías o se muestren reacios a usar dispositivos electrónicos. También puede ser una herramienta muy útil para todos aquellos usuarios que acaben de ser diagnosticados como diabéticos recientemente, ya que de forma muy cómoda y guiada podrán apoyarse en esta aplicación para iniciarse en el cambio de estilo de vida que conlleva ser diabético, sabemos que al principio es cuando más cuesta adaptarse a un cambio radical en el estilo de vida que deben de llevar los pacientes para mantener controlada su diabetes.

2.2 BASE DE DATOS

-Tipo de base de datos

-Porque hemos elegido este tipo

-Diseño de la tabla

TIPO

Hemos elegido una base de datos tipo SQL basado en SQLite que es la base de datos con la que trabaja Android por defecto

SQL: tablas que tienen relaciones entre ellas, base de datos relacional. Un dato de una tabla puede estar en otra tabla.

PORQUE

Existen otros tipos de base de datos que no son SQL como por ejemplo Firebase que es de Google, AWS que es de Amazon, Azure que es de Microsoft o MongoDB en la que tendríamos que montar nosotros mismos el servidor, a diferencia de las demás, esta ni siquiera tiene una librería (framework) con sus métodos para poder facilitarnos el desarrollo de la actividad.

Una librería es una colección de clases, un framework es una colección de librerías.

Entonces hemos elegido SQLite porque es con la que por defecto Android Studio trabaja, es muy fácil encontrar documentación y tutoriales al respecto, y dado que nuestra aplicación trabaja en local no hay necesidad de complicarse con una base de datos que trabaje en la nube (que es como trabajan el resto de las anteriormente mencionadas).

Local: base de datos que existe dentro del teléfono y ya está. Existe vive y trabaja dentro del móvil.

Las que trabajan en la nube tienen su servidor con su propio mantenimiento y tendríamos que pagar por el servicio.

Una base de datos SQLite en definitiva, es una base de datos embebida, esto significa que está integrada dentro de la aplicación, cada aplicación tiene su propia base de datos independiente de las otras actividades y por consiguiente no entraría en conflicto con las otras. Al mismo tiempo son rápidas y contienen un número de registros lo suficientemente amplia para poder trabajar con ellas sin problemas.

DISEÑO DE LA TABLA

CONTROL GLUCEMICO DB			
ID	FECHA	VALOR TOMA	NOMBRE TOMA
1	22/06/2019	102	07:30 a.m.
2	22/06/2019	88	13:00 a.m.
…	…	…	…
N	dd/mm/aaaa	N	Texto

Siendo:

- La columna “ID” la que iría numerando la identidad de cada registro que añadimos a nuestra base de datos.
- La columna “fecha” la que permite al usuario anotar numéricamente la fecha en que se hizo la toma que va a guardar.
- La columna “valor toma” la que permite al usuario anotar numéricamente el valor de azúcar en sangre del control glucémico realizado.
- La columna “nombre toma” la que permite al usuario anotar libremente más información acerca de este registro. En este ejemplo hemos anotado la hora del día en la que se hizo el control glucémico.

2.3 ARBOL DE NAVEGACIÓN

2.3.1 DIAGRAMA GENERAL

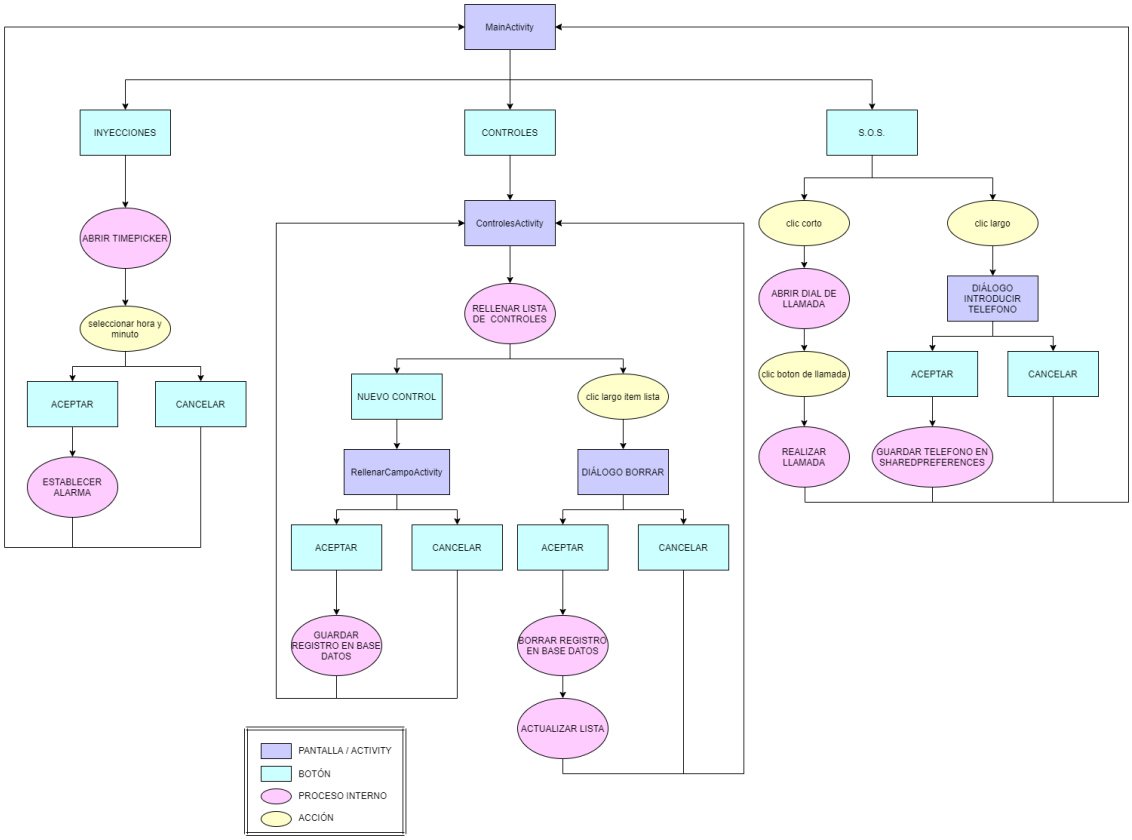
A continuación, se muestra el diagrama general de la aplicación, y posteriormente iremos viendo más detalladamente cada una de las partes:

-Azul: Pantallas/Activities.

-Verde: Botones.

-Rosa: Procesos internos.

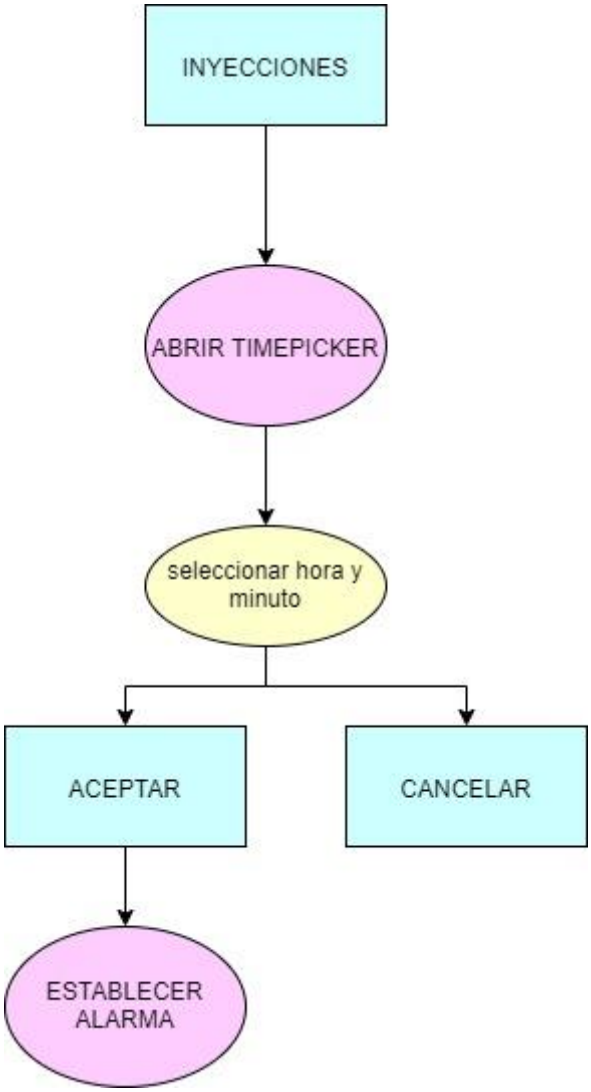
-Amarillo: Acciones.



2.3.2 INYECCIONES

La siguiente figura muestra el diagrama de navegación más detallado del botón inyecciones.

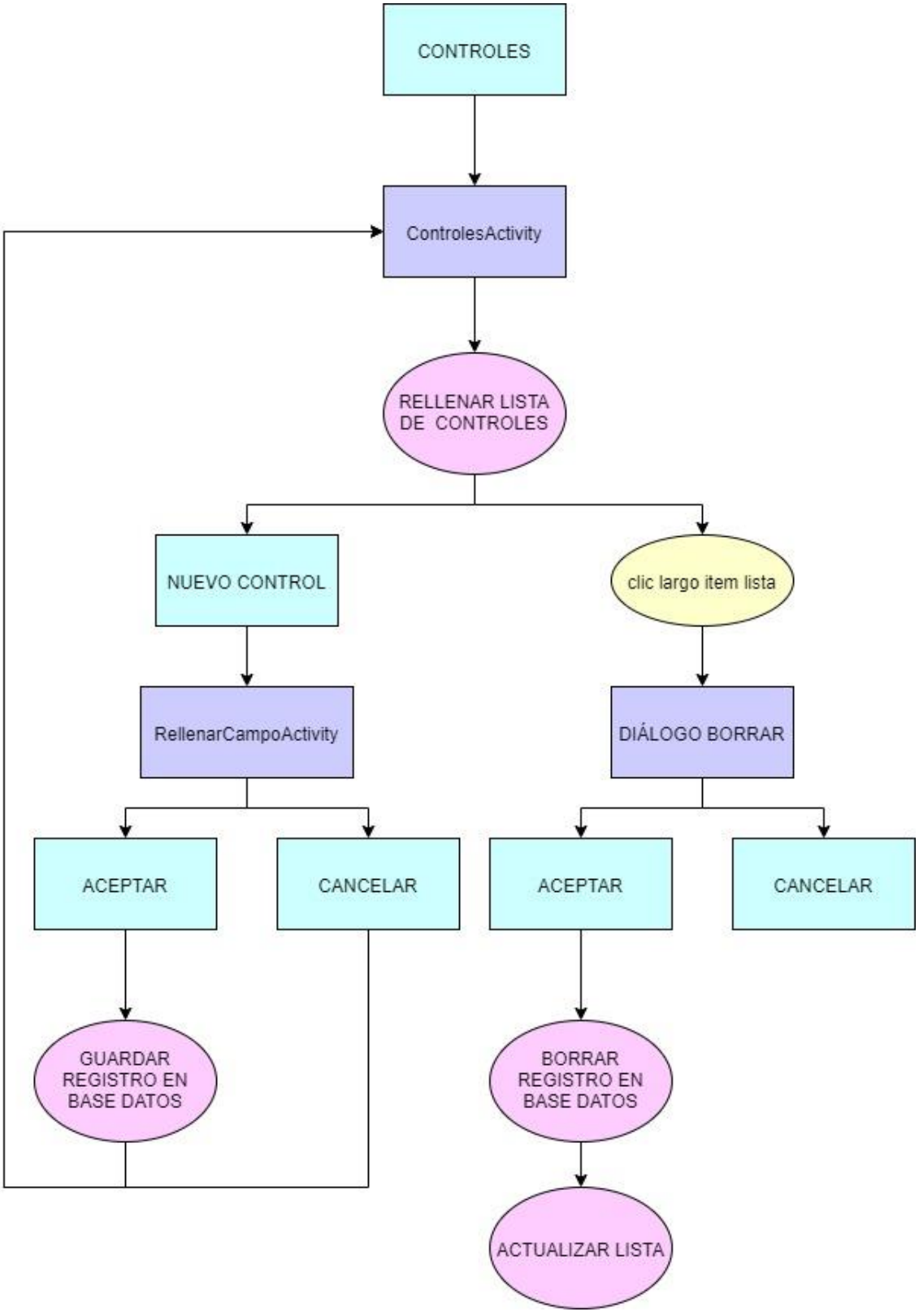
Al pulsar este, se nos abre el timepicker, que no es más que un reloj al que le podemos cambiar el formato a nuestro gusto como reloj analógico o reloj digital. En este podemos programar la alarma para que suene cuando tenemos que administrarnos la próxima inyección de insulina, seleccionando la hora y los minutos. Una vez ejecutado este paso se nos abrirá una pantalla en la que podremos elegir entre “Aceptar” (se establecerá la alarma programada) y “Cancelar” (regresaremos al timepicker).



2.3.3 CONTROLES

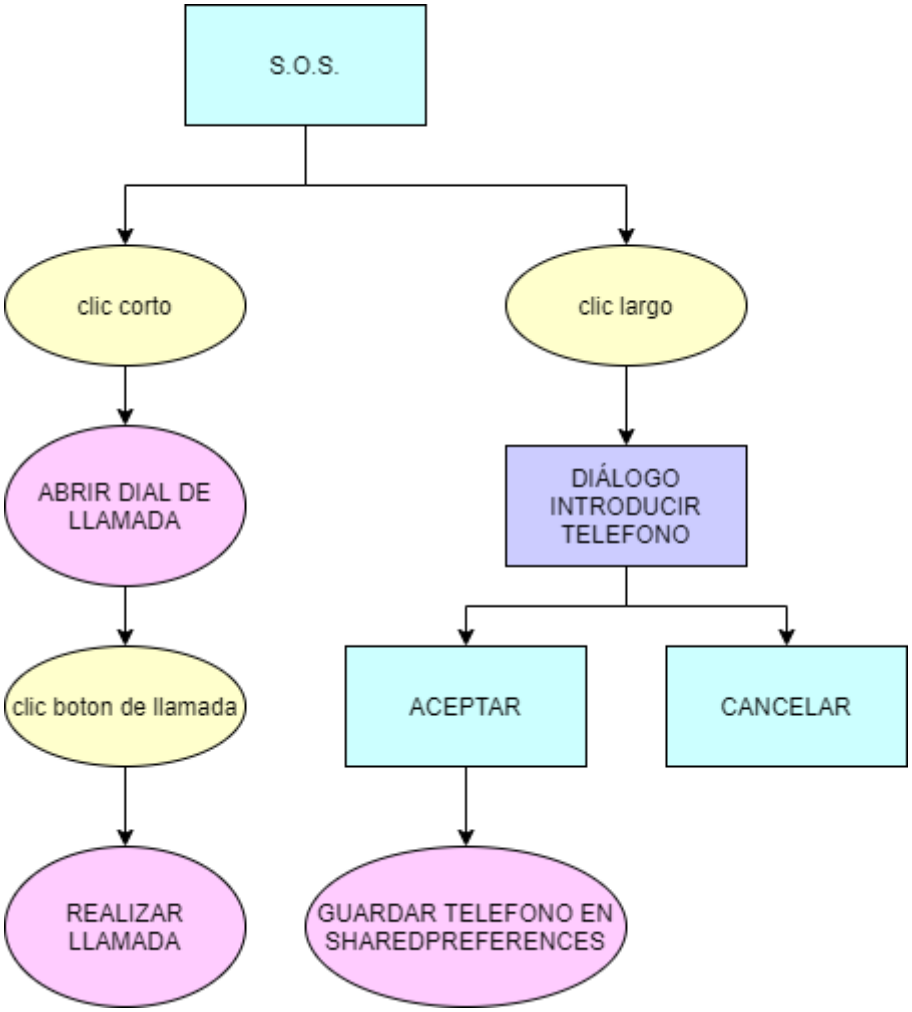
A continuación, se muestra el diagrama más detallado del botón Controles. Al pulsarlo nos lleva a la Activity "ControlesActivity", la cual es una base de datos interna que nos hace las funciones de agenda para agregar, modificar y eliminar los resultados de nuestros controles glucémicos permitiendo tener una lista en la que guardar los mismos. Al pulsar en el botón "Nuevo control" se abre la Activity "RellenarCampoActivity" donde podremos guardar los parámetros del control que deseamos registrar. Una vez hecho esto, se abre una pantalla donde podemos "Aceptar" (guarda el registro del control) o "Cancelar" (se descartan los parámetros introducidos.

Con la pulsación larga sobre uno de los registros ya guardados se nos abre la Activity "Dialogo Borrar" la cual nos despliega una pantalla con la opción de borrar control, pudiendo escoger entre "Aceptar" (borraríamos el control seleccionado y se actualiza la lista) o "Cancelar" (descarta la acción).


CONTROLES
ControlesActivity
RELLENAR LISTA DE CONTROLES
NUEVO CONTROL
clic largo item lista
RellenarCampoActivity
DIÁLOGO BORRAR
ACEPTAR
CANCELAR
ACEPTAR
CANCELAR
GUARDAR REGISTRO EN BASE DATOS
BORRAR REGISTRO EN BASE DATOS
ACTUALIZAR LISTA

2.3.4 S.O.S.

La siguiente figura muestra el diagrama más detallado del botón “S.O.S”. Con la acción clic corto sobre el botón nos abre el dial de llamada en el cual con un clic más en el botón de llamada realizaría una llamada de teléfono al número previamente guardado en sharedpreferences. Con la acción clic largo sobre el botón “S.O.S.” nos abre la pantalla del diálogo introducir teléfono para guardar un número de teléfono con la consecuente pantalla con las opciones “Aceptar” (guarda el número de teléfono en sharedpreferences) y “Cancelar” descartamos la acción.

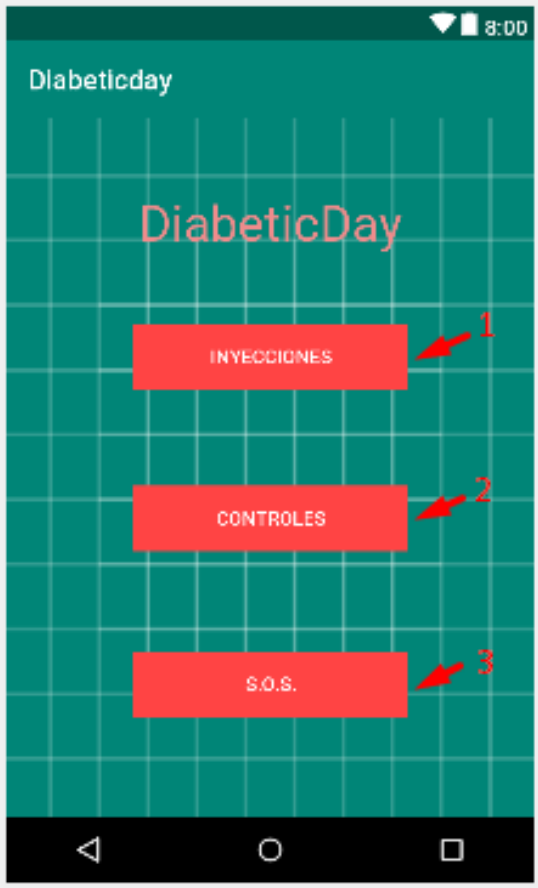


2.4 INTERFAZ GRÁFICA

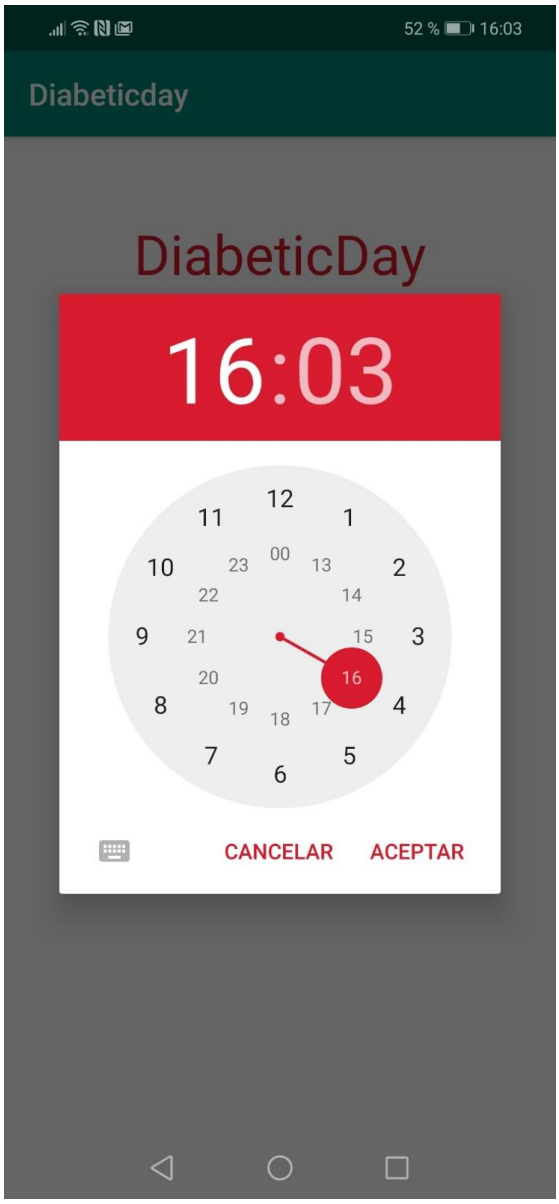
A continuación, navegaremos a lo largo de la aplicación recorriendo cada una de sus pantallas y de este modo familiarizándonos con la misma de forma gráfica.

En el menú principal de la aplicación contamos con 3 botones.

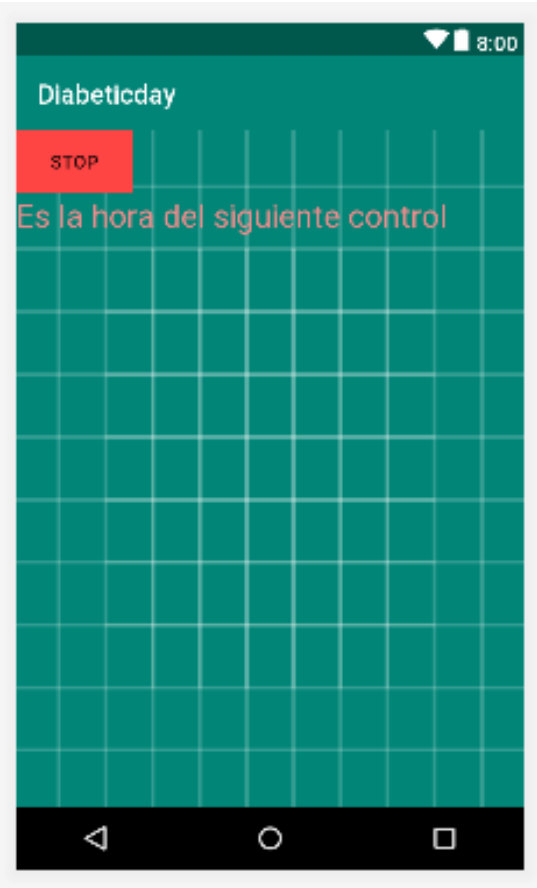
- Botón 1: Inyecciones
- Botón 2: Controles
- Botón 3: S.O.S.



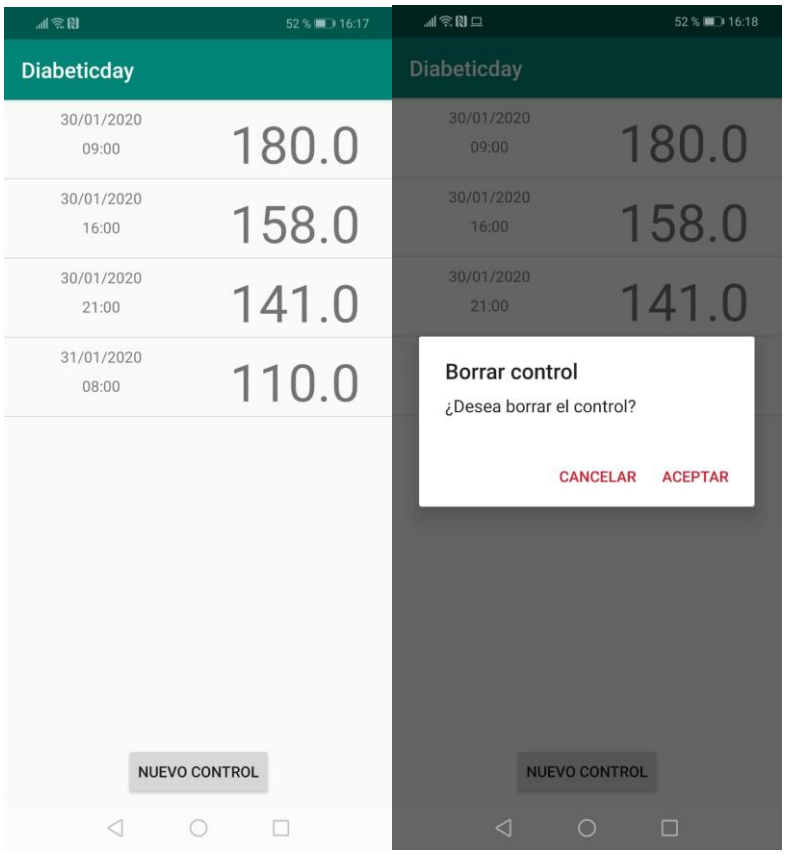
Si pulsamos sobre el primero de los botones (INYECCIONES) accederemos a la Activity Inyecciones donde nos aparecerá en pantalla un reloj en el cual podremos definir la alarma para la próxima inyección de insulina.



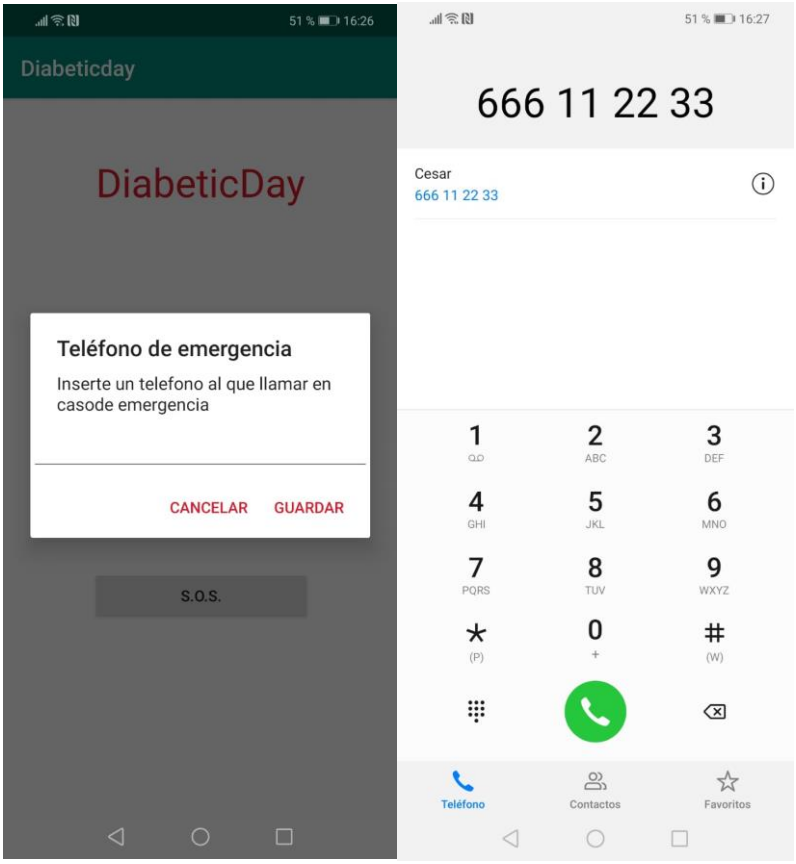
Llegado el momento en que la alarma se ejecute, al usuario le aparecerá en pantalla una ventana en la que únicamente podrá desactivar la alarma mediante un botón de STOP, dando a conocer a DiabeticDay que ha sido avisado de que es el momento de su siguiente inyección de insulina.



Si pulsamos sobre el segundo de los botones (CONTROLES) la aplicación llevará al usuario a la base de datos interna de DiabeticDay donde aparecerá una pantalla en la que el cliente podrá crear un nuevo registro, eliminar un registro ya guardado o simplemente consultar los registros almacenados en la base de datos.



Si pulsamos en el tercer botón del menú principal (S.O.S.), DiabeticDay nos lleva al dial de llamada en el cual con un clic más en el botón de llamada realizaría una llamada de teléfono al número previamente guardado. Con la acción clic largo sobre el botón "S.O.S." nos abre la pantalla del diálogo introducir teléfono para guardar un número de teléfono.



CAPÍTULO 3: IMPLEMENTACIÓN DEL PROYECTO

3.1 CONEXIÓN CON LA BASE DE DATOS

Como hemos dicho anteriormente la base de datos de SQLite es la que usa Android por defecto por lo que trabajar con ella resulta bastante sencillo.

Para realizar la conexión con la base de datos vamos a hacer uso de la clase ControlGlucemicoDbHelper, que hereda de SQLiteOpenHelper. Primero creamos un objeto de esta clase al que hemos llamado dbHelper. El siguiente paso es crear un objeto SQLiteDataBase mediante el método dbHelper.getWritableDatabase () al que vamos a llamar db. Con el objeto db tenemos una conexión directa con el gestor de la base de datos de SQLite.

```
public class ControlesActivity extends AppCompatActivity {
    private Button btNuevoControl;
    private ListView lvControles;
    private ControlGlucemicoDbHelper dbHelper;
    private SQLiteDatabase db;

    private ListaAdapter adaptador;

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

        dbHelper = new ControlGlucemicoDbHelper( context: this);
        db = dbHelper.getWritableDatabase();
```

3.2 LOGICA DE LA APLICACIÓN

MainActivity.class:

Es la pantalla principal de la aplicación, desde aquí vamos a acceder a todas las funcionalidades.

Contiene tres botones:

1. Inyecciones: abre el reloj para establecer la siguiente alarma que avisará al usuario de cuándo debe realizar la próxima toma
2. Controles: abre la actividad ControlesActivity.class. En ella visualizaremos un listado de las distintas tomas de control que tengamos almacenadas en la base de datos.
3. S.O.S.: con pulsación corta, abre el dial de llamada del teléfono con el número de teléfono guardado en la memoria de la aplicación. Con pulsación larga abre el formulario de entrada en el que podremos especificar el número de teléfono que queremos que aparezca en el dial de llamada cuando realizamos la pulsación corta.

ControlesActivity.class:

Muestra un listado con los registros almacenados en la base de datos referentes a los controles que hayamos realizado.

Contiene un ListView en el que se muestra el listado arriba mencionado. También encontramos un botón llamado “Nuevo Control” con el que abrimos la actividad “RellenarCampoActivity.class” donde generamos un nuevo registro en la base de datos y que se mostrará en la lista.

ListView "lvControles":

El ListView muestra una relación de los ítems o elementos que le pasamos a su adaptador por medio de un Array de objetos de ControlGlucémico.class.

Cada uno de los items muestra la información del control glucémico pertinente, que es la fecha en que se realizó el control, el valor de la toma y una breve descripción que el usuario puede introducir a su gusto y criterio.

RellenarCampoActivity.class:

Contiene 3 editText en los que podemos introducir los valores de un nuevo registro (fecha, valor, nombre) y dos botones (Aceptar y Cancelar).

Al pulsar Aceptar, con los valores de los EditText, se crea un nuevo registro en la base de datos y nos devuelve a la ControlesActivity.

Al pulsar Cancelar, se cancela la operación y te devuelve a ControlesActivity.

3.3 CLASES DE UTILIDAD:

ControlGlucémico.class

Una clase pojo son unos planos donde vamos a almacenar la información que vamos a obtener de la base de datos para poder acceder a ella de manera sencilla.

Digamos que esto consiste en organizar las variables en un conjunto y dejarlo definido como tal, de este modo cuando tengamos que volver a usar en cualquier parte del programa todas estas variables, no tendremos que volver a mencionarlas una a una ya que podremos echar mano de nuestro conjunto y así optimizar el tiempo y tenerlo todo más organizado.

Dentro de la clase Control Glucémico tenemos los siguientes atributos:

- Final int ID: el identificador del registro. Se obtiene del identificador de la tabla de la base de datos. Se ha declarado como final, porque una vez le demos un valor no lo vamos a cambiar y será auto incrementable.
- String fecha: usado para almacenar la fecha en que se crea el objeto
- Float valor_toma: para almacenar el valor del control glucémico realizado
- String nombre_toma: para almacenar una descripción del control glucémico realizado

Dentro de la clase Control Glucémico tenemos los siguientes métodos:

Getters y setters de todos los atributos para poder acceder a ellos desde el exterior de la clase.

Getters se utilizan para coger el valor de cada uno de los atributos y setters para darle valor.

```java
public class ControlGlucemico {
    private final int ID;
    private String fecha;
    private float valor_toma;
    private String nombre_toma;

    //CONSTRUCTOR
    public ControlGlucemico(int id, String fecha, float valor_toma, String nombre_toma) {
        this.ID = id;
        this.fecha = fecha;
        this.valor_toma = valor_toma;
        this.nombre_toma = nombre_toma;
    }
```

ControlGlucémicoContract.class:

La clase Contract sirve como un acuerdo en el que el objetivo es básicamente simplificar las posteriores modificaciones del código como por ejemplo las queries (consultas).

Dentro de la clase contract encontramos una clase interna al que añadiremos el sufijo entry para una mejor identificación por cada pojo que vayamos a usar.

```java
public class ControlGlucemicoContract {

    public static abstract class ControlGlucemicoEntry{
        public static final String TABLE_NAME = "ControlGlucemico";

        public static final String _ID = BaseColumns._ID; //necesario para android;

        public static final String FECHA = "Fecha";

        public static final String VALOR_TOMA = "valor_toma";

        public static final String NOMBRE_TOMA = "nombre_toma";

    }
}
```

ControlGlucemicoDbHelper.class:

Como su nombre indica (db data base, Helper ayudante) es una clase auxiliar para ayudarnos a acceder a la base de datos y que hereda de SQLiteOpenHelper, para así tener la funcionalidad de la clase padre.

Ayuda a que no se te olvide que ese método existe y tienes que ponerlo. Y la capacidad de darle distinta funcionalidad a un mismo método se llama polimorfismo.

POR EJEMPLO: método calcular área: sería distinta para un triángulo que para un cuadrado. “Medir distancia” sería el método del padre que heredamos y luego usaríamos

para las clases hijas cada una el suyo: triangulo.medir distancia y cuadrado.medir distancia.

La otra ventaja de la herencia, que es la que nosotros queremos usar realmente, es usar la funcionalidad de los métodos del padre sin el sobrescribir (@override), usándolos directamente.

Esta clase contiene los siguientes componentes:

VARIABLES:

- Static final int DATABASE_VERSION: Nos indica en que versión de la app nos encontramos actualmente.

  Si más adelante realizásemos cambios con una nueva versión, la app al comprobar que la última versión es distinta a la primera lo que haría sería actualizar.

- Static final String DATABASE_NAME: Da nombre al archivo que contiene la base de datos, "Control_Glucemico.db".

- Static final String CREATE_TABLE: Escribimos de manera concreta la sentencia en estructura de lenguaje SQL para la creación de la tabla que más adelante llamaremos con el método execSQL.

- Static final String DROP_TABLE: Sentencia SQP para borrar la tabla.

```
//version de la base de datos
private static final int DATABASE_VERSION = 1;

//nombre del archivo de la base de datos
private static final String DATABASE_NAME ="Control_Glucemico.db";

//creamos las sentencias SQL que crea la tabla
private static final String CREATE_TABLE = "CREATE TABLE if not exists "
        + ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME + " ("
        + ControlGlucemicoContract.ControlGlucemicoEntry._ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
        + ControlGlucemicoContract.ControlGlucemicoEntry.FECHA + " TEXT NOT NULL,"
        + ControlGlucemicoContract.ControlGlucemicoEntry.VALOR_TOMA + " REAL NOT NULL,"
        + ControlGlucemicoContract.ControlGlucemicoEntry.NOMBRE_TOMA + " TEXT NOT NULL"
        + " )";

//borrar la tabla
private static final String DROP_TABLE = "DROP TABLE IF EXISTS " + ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME;
```

MÉTODOS:

- Constructor: al crear la base de datos se usará el nombre que aquí le hemos dado (DATABASE_NAME) y guardará la información de la versión que le pasemos como argumento al constructor (DATABASE_VERSION).

- onCreate(): Cuando este método se ejecuta (heredado del padre) se ejecuta lo que hay dentro y en este caso es, crear la tabla.

- onUpgrade(): este método se llama cuando se actualiza la base de datos

  Cuando la versión nueva es distinta de la vieja versión, borra la tabla y crea una nueva.

```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_TABLE);

}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL(DROP_TABLE);
    onCreate(db);

}
```

- insert(): inserta una fila en la tabla con la información obtenida del objeto ControlGlucémico que le pasamos como parámetro.

```
//metodo para insertar datos en la tabla
public void insert (ControlGlucemico controlGlucemico, SQLiteDatabase db) {
    String fecha = ControlGlucemicoContract.ControlGlucemicoEntry.FECHA;
    String valor_toma = ControlGlucemicoContract.ControlGlucemicoEntry.VALOR_TOMA;
    String nombre_toma = ControlGlucemicoContract.ControlGlucemicoEntry.NOMBRE_TOMA;
    String insertar = "INSERT INTO " + ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME
            + " (" + fecha + ", " + valor_toma + ", " + nombre_toma
            + ") VALUES (\'" + controlGlucemico.getFecha() + "\', "
            + controlGlucemico.getValor_toma() + ", \'" + controlGlucemico.getNombre_toma() + "')";

    db.execSQL(insertar);

}
```

- delete(): borra una fila de la tabla donde el ID coincida con el del objeto ControlGlucémico que le pasamos como parámetro.

```
//metodo para borrar datos en la tabla
public void delete (ControlGlucemico controlGlucemico, SQLiteDatabase db) {
    int id = controlGlucemico.getID();
    String delete = "DELETE FROM " + ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME + " WHERE _ID =" + id;
    db.execSQL(delete);

}
```

- obtenCursorControles(): devuelve un objeto de tipo cursor con la información de todas las filas de la tabla

  Va a la tabla, obtiene todas las filas de la tabla, las guarda en un objeto de tipo cursor y devuelve este cursor que acaba de rellenar.

```
//metodo para leer datos de la tabla
public Cursor obtenCursorControles (SQLiteDatabase db) {
    String select = "SELECT * FROM " + ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME;
    Cursor cursor = db.rawQuery(select,  selectionArgs: null);
    return cursor;
}
```

- obtenListaControles(): devuelve un objeto de tipo ArrayList que dentro va a contener objetos de tipo ControlGlucémico. O sea, coge un cursor y guarda los valores de esta fila en una lista, y después pasaría a la siguiente fila de la tabla a la que haría referencia el cursor. Y cuando no tiene más dónde coger se detiene el "while" y entonces devuelve la lista (ArrayList).

```java
//metodo que devuelve una lista
public ArrayList <ControlGlucemico> obtenListaControles (SQLiteDatabase db) {
    ArrayList <ControlGlucemico> lista = new ArrayList<~>();
    Cursor cursor = obtenCursorControles(db);
    if (cursor.getCount() > 0) {
        cursor.moveToFirst();
        do {
            int id = cursor.getInt( columnIndex: 0);
            String fecha = cursor.getString( columnIndex: 1);
            float valor = Float.parseFloat(cursor.getString( columnIndex: 2));
            String nombre = cursor.getString( columnIndex: 3);
            ControlGlucemico cg = new ControlGlucemico(id, fecha, valor, nombre);
            lista.add(cg);
        } while (cursor.moveToNext());
    }
    return lista;
}
```

ListaAdapter.class:

En esta aplicación solo tenemos una ListaAdapter pero podríamos tener muchas más.

La función del adaptador es coger los objetos de la lista para rellenar el ítem con ese formato que le hemos dado y luego meter el ítem en el listView, y este ya lo muestra en pantalla.

Tiene una variable ArrayList, un método constructor y un método getView.

- ArrayList<ControlGlucemico> lista: va a contener una relación de los controles glucémicos que tenemos guardados en la base de datos.
- Constructor: asigna a la ArrayList la lista que se le pasa como parámetro al constructor (que será la lista obtenida de la base de datos mediante dbHelper.obtenListaControles().
- getView(): coge cada uno de los objetos guardados en la lista, obtiene los valores de sus atributos y rellena los campos del ítem.

```
public class ListaAdapter extends ArrayAdapter<ControlGlucemico> {
    private ArrayList <ControlGlucemico> lista;

    public ListaAdapter(Context context, ArrayList<ControlGlucemico> lista) {
        super(context, R.layout.item_lista, lista);
        this.lista = lista;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {

        //Inflate rellena los datos de los items
        LayoutInflater inflater = LayoutInflater.from(getContext());

        View item = inflater.inflate(R.layout.item_lista,  root: null);

        TextView tvFecha = item.findViewById(R.id.tvFecha);
        tvFecha.setText(lista.get(position).getFecha());

        TextView tvNombre = item.findViewById(R.id.tvNombre);
        tvNombre.setText(lista.get(position).getNombre_toma());

        TextView tvValor = item.findViewById(R.id.tvValor);
        tvValor.setText(lista.get(position).getValor_toma() + ""); //concateno

        return item;
    }
```

3.4 HERRAMIENTAS UTILIZADAS

Las herramientas utilizadas que han servido de apoyo para la elaboración de este proyecto se mencionan a continuación:

-JetBrains Android Studio, AS: para la implementación del proyecto.

-AVD integrado en AS: emulación de dispositivos.

-Google Chrome: navegador web.

-SQLite base de datos: sistema de gestión de base de datos relacional.

-Draw.io: diseño de diagramas.

-Lightshot: programa para hacer capturas de pantalla.

CAPITULO 4: CONCLUSIONES

He de mencionar que el presente proyecto además de introducirme en la programación en AndroidStudio, me ha servido para aprender a desenvolverme mejor a la hora de intentar solucionar problemas. Al inicio del proyecto, mis conocimientos sobre este tipo de programación eran nulos y he de reconocer que en ocasiones me he sentido un poco abrumado durante su creación. A pesar de que parece una aplicación con unas funciones muy sencillas, cuenta con diversas actividades y varios elementos de programación que para los alumnos de ingeniería mecánica se presentan como totalmente nuevos. Así se crea la necesidad de comenzar con un período de investigación en canales de YouTube, foros y largas listas con dudas para plantear a mi tutor César Fernández Peris. El método ensayo-error ha sido el pan de cada día a la hora de elaborar este proyecto. La finalización de este proyecto ha resultado muy satisfactoria a nivel personal y me ha enseñado mucho, más allá de la programación en AndroidStudio.

4.1 TRABAJOS FUTUROS

Como posibles y futuras mejoras para esta aplicación cabría destacar:

-La posibilidad de mejorar la función del botón de S.O.S. añadiendo una vinculación con Google Maps en la que se podría enviar la ubicación exacta del usuario que tiene una emergencia además de la actual función de llamada.

-La mejora del aspecto visual, con el fin de diseñar una aplicación más bonita y llamativa para el usuario.

-La posibilidad de incluir un botón nuevo en el que se pueda mostrar un calendario de las farmacias de guardia y centros de salud de urgencias para cada día del año.

4.2 BIBLIOGRAFÍA

-Youtube - https://www.youtube.com/

-Stackoverflow - https://es.stackoverflow.com/

-Píldoras informáticas
https://www.youtube.com/watch?v=pdYkmCcQFd8&list=PLU8oAlHdN5Bkn-KS1sRFlSEnXXcAtAJ9P

-Documentación oficial de Android - https://developer.android.com/guide?hl=es-419

5 ANEXO CÓDIGO FUENTE

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diabeticday">

    <uses-permission android:name = "android.permission.VIBRATE"/><!--
video profesor-->
    <uses-permission android:name="android.permission.WAKE_LOCK"/><!--
de stackoverflow-->


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".activities.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".activities.ControlesActivity" />
        <activity
android:name=".activities.RellenarCampoActivity"></activity>
        <activity
android:name=".activities.AlarmSoundActivity"></activity>

        <receiver android:process=":remote"
android:name=".model.Alarm"/><!--de stackoverflow-->
    </application>

    <!--PERMISO DE ACCESO A LAS LLAMADAS-->
    <uses-permission android:name="android.permission.CALL_PHONE" />

</manifest>
```

Java

Activities:

AlarmSoundActivity

```
package com.example.diabeticday.activities;

import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Bundle;
import android.os.Vibrator;
import android.support.v7.app.AppCompatActivity;
import android.view.View;

import com.example.diabeticday.R;

public class AlarmSoundActivity  extends AppCompatActivity {

    private int hora, minutos, horasTotales, minutosTotales,
segundosTotales;
    private long milisegundosAlPulsarTimePicker, milisegundosTotales;

    private Ringtone ringtone;

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

        //hacer que vibre
        Vibrator vibrator =
(Vibrator)getSystemService(VIBRATOR_SERVICE);
        vibrator.vibrate(2000);

        //Hacer que se reproduzca un sonido
        Uri uri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
        ringtone = RingtoneManager.getRingtone(this, uri);
        ringtone.play();

        //mostrar un cuadro de dialogo para poder parar la alarma, si
lo descomento no va el sonido y no se muestra el dialogo tampoco
        /*AlertDialog.Builder cancelar = new
AlertDialog.Builder(context);
        cancelar.setTitle("Parar Alarma");
        cancelar.setMessage("Pulsa Cancelar para parar la alarma");

        cancelar.setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                CancelAlarm(context);
                dialog.cancel();
            }
        });
        cancelar.show();*/

    }
```

```
    public void stopAlarm(View view) {
        ringtone.stop();
    }
```

ControlesActivity

```
package com.example.diabeticday.activities;

import android.content.DialogInterface;
import android.content.Intent;
import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ListView;

import com.example.diabeticday.R;
import com.example.diabeticday.model.ControlGlucemico;
import com.example.diabeticday.model.ControlGlucemicoDbHelper;
import com.example.diabeticday.model.ListaAdapter;

public class ControlesActivity extends AppCompatActivity {
    private Button btNuevoControl;
    private ListView lvControles;
    private ControlGlucemicoDbHelper dbHelper;
    private SQLiteDatabase db;

    private ListaAdapter adaptador;

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

        dbHelper = new ControlGlucemicoDbHelper(this);
        db = dbHelper.getWritableDatabase();

        //Asignación VIEW
        btNuevoControl = findViewById(R.id.btNuevoControl);
        lvControles = findViewById(R.id.lvControles);

        //adaptador = new ListaAdapter(this,
dbHelper.obtenListaControles(db));
        // lvControles.setAdapter(adaptador);
//        lvControles.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
//            @Override
//            public void onItemClick(AdapterView<?> parent, View
view, int position, long id) {
//
//            }
//        });
```

```
        btNuevoControl.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
Intent(ControlesActivity.this,RellenarCampoActivity.class);
                startActivity(intent);
            }
        });

    }
    @Override
    public void onResume() {
        super.onResume();;
        adaptador = new ListaAdapter(this,
dbHelper.obtenListaControles(db));
        lvControles.setAdapter(adaptador);

        lvControles.setOnItemLongClickListener(new
AdapterView.OnItemLongClickListener() {
            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View
view, int position, long id) {

                final ControlGlucemico controlGlucemico =
(ControlGlucemico) lvControles.getItemAtPosition(position);

                AlertDialog.Builder builder =
                        new
AlertDialog.Builder(ControlesActivity.this);

                builder.setMessage("¿Desea borrar el control?")
                        .setTitle("Borrar control")
                        .setPositiveButton("Aceptar", new
DialogInterface.OnClickListener()  {
                            public void onClick(DialogInterface
dialog, int id) {
                                //borro el registro
                                dbHelper.delete(controlGlucemico,db);
                                adaptador = new
ListaAdapter(ControlesActivity.this,
dbHelper.obtenListaControles(db));
                                lvControles.setAdapter(adaptador);
                                dialog.cancel();
                            }
                        })
                        .setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface
dialog, int id) {
                                //no hace nada
                                dialog.cancel();
                            }
                        });

                builder.show();
                return false;
            }
        });
```

```
    }
}
```

MainActivity

```
package com.example.diabeticday.activities;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.app.TimePickerDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.InputType;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import com.example.diabeticday.R;
import com.example.diabeticday.model.Alarm;
import com.example.diabeticday.model.ControlGlucemico;
import com.example.diabeticday.model.ControlGlucemicoDbHelper;
import com.example.diabeticday.model.ListaAdapter;
import com.example.diabeticday.model.TelefonoEmergencia;

import java.sql.Date;
import java.util.Calendar;

public class MainActivity extends AppCompatActivity {

    private Button btInyecciones;
    private Button btControles;
    private Button btSOS;

    private int hora, minutos, horasTotales, minutosTotales,
segundosTotales, tlf;
    private long milisegundosAlPulsarTimePicker, milisegundosTotales;

    private ControlGlucemicoDbHelper dbHelper;
    private SQLiteDatabase db;

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


        dbHelper = new ControlGlucemicoDbHelper(this);
        db = dbHelper.getWritableDatabase();

        /*
        try {
            //Obenter el telefono de emergencia a que llamar desde la
```

```
base de datos
            TelefonoEmergencia telefonoEmergencia =
dbHelper.obtenTelefonoEmergencia(db);
            tlf = telefonoEmergencia.getTlf();
        }catch(Exception e){

        }
        */

        //Asignacion VIEW
        btInyecciones = findViewById(R.id.btInyecciones);
        btControles = findViewById(R.id.btControles);
        btSOS = findViewById(R.id.btSOS);

        btControles.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
ControlesActivity.class);
                startActivity(intent);
            }
        });

        btInyecciones.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // obtener la hora actual del telefono para que se
abra el timePicker en la hora actual
                final Calendar calendar = Calendar.getInstance();
                milisegundosAlPulsarTimePicker =
System.currentTimeMillis();
                hora = calendar.get(Calendar.HOUR_OF_DAY);
                minutos = calendar.get(Calendar.MINUTE);

                //Mostrar dialogo seleccion de hora, timePicker
                TimePickerDialog timePickerDialog = new
TimePickerDialog(MainActivity.this, new
TimePickerDialog.OnTimeSetListener() {
                    @Override
                    public void onTimeSet(TimePicker view, int
hourOfDay, int minute) {

                        //mensaje de alarma establecida
                        Toast.makeText(MainActivity.this, "Alarma
establecida: " + hourOfDay + ":" + minute, Toast.LENGTH_LONG).show();

                        //calculo los segundos totales que faltan
hasta que suene la alarma. se obtienen del timePiecker
                        horasTotales = hourOfDay - hora;
                        minutosTotales = minute - minutos;
                        segundosTotales = (horasTotales * 3600) +
(minutosTotales * 60);

                        //para establecer la alarma necesito pasarle l
tiempo en milisegundos. los calculo de los segundos totales que me da
el timePicker
                        milisegundosTotales =
milisegundosAlPulsarTimePicker + (segundosTotales * 1000); //hora a la
que tiene que sonar

                        //llamar al metodo que establece la alarma. le
```

```
paso por parametro el contexto y los milisegundos
                        Alarm alarm = new Alarm();
                        alarm.SetAlarm(MainActivity.this,
milisegundosTotales);

                    }
                }, hora, minutos, true);
                //muestro el timePicker
                timePickerDialog.show();
            }
        });

        //pulsacion corta del boton de llamada demergencia
        btSOS.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                SharedPreferences prefs =
getSharedPreferences("preferences", MODE_PRIVATE);
                tlf = prefs.getInt("emergency", 112);

                //Lanzar llamada
                Intent intent = new Intent(Intent.ACTION_DIAL);
                intent.setData(Uri.parse("tel:" + tlf));
                if(intent.resolveActivity(getPackageManager()) !=
null){
                    startActivity(intent);
                }
            }
        });

        //pulsacion larga del boton de llamada de emergencia
        btSOS.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {

                //mostrar el dialogo para introducir el numero de
telefono
                AlertDialog.Builder builder =
                        new AlertDialog.Builder(MainActivity.this);

                // meter el numero
                final EditText etTlf = new
EditText(MainActivity.this);
                etTlf.setInputType(InputType.TYPE_CLASS_PHONE);
                builder.setView(etTlf);

                builder.setMessage("Inserte un telefono al que llamar
en casode emergencia")
                        .setTitle("Teléfono de emergencia")
                        .setPositiveButton("Guardar", new
DialogInterface.OnClickListener()  {
                            public void onClick(DialogInterface
dialog, int id) {

                                /*
                                //borrar telefono de la base de datos

                                TelefonoEmergencia
telefonoEmergenciaBorrar = null;
```

```
                                try {
                                    telefonoEmergenciaBorrar =
dbHelper.obtenTelefonoEmergencia(db);
//                                  System.err.println("@@@@" +
telefonoEmergenciaBorrar.getId());
                                    //
System.err.println("@@@@" + telefonoEmergenciaBorrar.getTlf());
                                }catch(Exception e){

                                }

                                //si la bas de datos contiene un
telefono guardado lo borra (solo quermos tener uno guardado a la vez)
                                if(telefonoEmergenciaBorrar != null){

dbHelper.deleteTlf(telefonoEmergenciaBorrar,db);
                                }

                                //guardar telefono en la base de datos
                                int tlfNumero =
Integer.parseInt(String.valueOf(etTlf.getText()));
                                System.err.println("@@@@" +
tlfNumero);

                                TelefonoEmergencia
telefonoEmergencia1Guardar = new TelefonoEmergencia(1, tlfNumero);

dbHelper.insertTlf(telefonoEmergencia1Guardar,db);

                                //dar valor a la variable que alberga
el numeo de telefono al que llamar
                                tlf = tlfNumero;
                                */

                                int tlfNumero =
Integer.parseInt(String.valueOf(etTlf.getText()));
                                SharedPreferences.Editor editor =
getSharedPreferences("preferences", MODE_PRIVATE).edit();
                                editor.putInt("emergency", tlfNumero);
                                editor.apply();

                                //mostrar mensaje de telefono guardado
                                Toast.makeText(MainActivity.this,
"Número guardado " + tlfNumero,Toast.LENGTH_LONG).show();
                            }
                        })
                        .setNegativeButton("Cancelar", new
DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface
dialog, int id) {
                                dialog.cancel();
                            }
                        });

                builder.show();
                return false;
            }
        });
    }
}
```

RellenarCampoActivity

```
package com.example.diabeticday.activities;

import android.database.sqlite.SQLiteDatabase;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.example.diabeticday.R;
import com.example.diabeticday.model.ControlGlucemico;
import com.example.diabeticday.model.ControlGlucemicoDbHelper;

public class RellenarCampoActivity extends AppCompatActivity {

    private EditText etFecha;
    private EditText etValor;
    private EditText etNombre;
    private Button btAceptar;
    private Button btCancelar;
    private ControlGlucemicoDbHelper dbHelper;
    private SQLiteDatabase db;

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

        //Asignacion VIEW
        etFecha = findViewById(R.id.etFechaRellenar);
        etValor = findViewById(R.id.etValorRellenar);
        etNombre = findViewById(R.id.etNombreRellenar);
        btAceptar = findViewById(R.id.btAceptarRellenar);
        btCancelar = findViewById(R.id.btCancelarRellenar);
        dbHelper = new ControlGlucemicoDbHelper(this);
        db = dbHelper.getWritableDatabase();

        btAceptar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String fecha = etFecha.getText().toString();
                float valor =
Float.parseFloat(etValor.getText().toString());
                String nombre = etNombre.getText().toString();
                ControlGlucemico controlGlucemico = new
ControlGlucemico(0, fecha, valor, nombre);


                dbHelper.insert(controlGlucemico, db);

                finish();
            }
        });

        btCancelar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
```

```
        });
    }

}
```

Model:

Alarm

```
package com.example.diabeticday.model;

import android.app.AlarmManager;
import android.app.AlertDialog;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.media.Ringtone;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.PowerManager;
import android.os.Vibrator;
import android.util.Log;
import android.widget.Toast;

import com.example.diabeticday.R;
import com.example.diabeticday.activities.AlarmSoundActivity;

public class Alarm extends BroadcastReceiver {
    @Override
    public void onReceive(final Context context, Intent intent) {

        //no se que hace
        PowerManager pm = (PowerManager)
context.getSystemService(Context.POWER_SERVICE);
        PowerManager.WakeLock wl =
pm.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,"Alarm:");
        wl.acquire();

        /*
        //hacer q vibre
        Vibrator vibrator =
(Vibrator)context.getSystemService(context.VIBRATOR_SERVICE);
        vibrator.vibrate(2000);

        //Hacer que se reproduzca un sonido
        Uri uri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);
        Ringtone ringtone = RingtoneManager.getRingtone(context, uri);
        ringtone.play();

        //Mensaje que se muestra cuando se cumpla el tiempo
        Toast.makeText(context, "Esta sonando la alarma",
Toast.LENGTH_LONG).show();
        */

        Log.d("ALARM", "Esta sonando la alarma");
```

```
        Intent alarmIntent = new
Intent(context.getApplicationContext(), AlarmSoundActivity.class);
        alarmIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(alarmIntent);


        wl.release();
    }



    //método que establece la alarma
    public void SetAlarm(Context context, long tiempoTotal)
    {
        //gestor de alarmas
        AlarmManager
am=(AlarmManager)context.getSystemService(Context.ALARM_SERVICE);
        //intento de hacer algo, establacer la alarma en este caso
        Intent i = new Intent(context, Alarm.class);
        //intento de hacer algo pendiente, se lanza cuando se cumpla
el tiempo
        PendingIntent pi = PendingIntent.getBroadcast(context, 0, i,
0);
        //decirle al gestor que establezca la alarma, empieza a contar
en el momento que le doy el boton
        am.setRepeating(AlarmManager.RTC_WAKEUP, tiempoTotal,0, pi);
    }

    //para la alarma
    public void CancelAlarm(Context context)
    {
        Intent intent = new Intent(context, Alarm.class);
        PendingIntent sender = PendingIntent.getBroadcast(context, 0,
intent, 0);
        AlarmManager alarmManager = (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE);
        alarmManager.cancel(sender);
    }
```

ControlGlucemico

```
package com.example.diabeticday.model;



public class ControlGlucemico {
    private final int ID;
    private String fecha;
    private float valor_toma;
    private String nombre_toma;

    //CONSTRUCTOR
    public ControlGlucemico(int id, String fecha, float valor_toma,
String nombre_toma) {
        this.ID = id;
        this.fecha = fecha;
        this.valor_toma = valor_toma;
        this.nombre_toma = nombre_toma;
    }

    //GETTERS AND SETTERS


    public int getID() {
        return ID;
    }

    public String getFecha() {
        return fecha;
    }

    public void setFecha(String fecha) {
        this.fecha = fecha;
    }

    public float getValor_toma() {
        return valor_toma;
    }

    public void setValor_toma(float valor_toma) {
        this.valor_toma = valor_toma;
    }

    public String getNombre_toma() {
        return nombre_toma;
    }

    public void setNombre_toma(String nombre_toma) {
        this.nombre_toma = nombre_toma;
    }
}
```

ControlGlucemicoContract

```
package com.example.diabeticday.model;

import android.provider.BaseColumns;


public class ControlGlucemicoContract {

    public static abstract class ControlGlucemicoEntry{
        public static final String TABLE_NAME = "ControlGlucemico";

        public static final String _ID = BaseColumns._ID; //necesario
para android;

        public static final String FECHA = "Fecha";

        public static final String VALOR_TOMA = "valor_toma";

        public static final String NOMBRE_TOMA = "nombre_toma";

    }

    public static abstract class TelefonoEmergenciaEntry{

        public static final String TABLE_NAME = "TelefonoEmergencia";
        public static final String _ID = BaseColumns._ID;
        public static final String TLF = "tlf";
    }
}
```

ControlGlucemicoDbHelper

```
package com.example.diabeticday.model;

import android.content.Context;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import java.util.ArrayList;

public class ControlGlucemicoDbHelper extends SQLiteOpenHelper {

    //version de la base de datos
    private static final int DATABASE_VERSION = 1;

    //nombre del archivo de la base de datos
    private static final String DATABASE_NAME ="Control_Glucemico.db";

    //creamos las sentencias SQL que crea la tabla de ontroles
glucemicos
    private static final String CREATE_TABLE = "CREATE TABLE if not
exists "
            +
ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME + " ("
            + ControlGlucemicoContract.ControlGlucemicoEntry._ID + "
INTEGER PRIMARY KEY AUTOINCREMENT,"
            + ControlGlucemicoContract.ControlGlucemicoEntry.FECHA + "
TEXT NOT NULL,"
            +
ControlGlucemicoContract.ControlGlucemicoEntry.VALOR_TOMA + " REAL NOT
NULL,"
            +
ControlGlucemicoContract.ControlGlucemicoEntry.NOMBRE_TOMA + " TEXT
NOT NULL"
            + " )";

    //Creamos las sentencias SQL que crea la tabla del numero de
telefono de emergencia
    private static final String CREATE_TABLE_TLF = "CREATE TABLE if
not exists "
            +
ControlGlucemicoContract.TelefonoEmergenciaEntry.TABLE_NAME + " ("
            + ControlGlucemicoContract.TelefonoEmergenciaEntry._ID + "
INTEGER PRIMARY KEY AUTOINCREMENT,"
            + ControlGlucemicoContract.TelefonoEmergenciaEntry.TLF + "
INTEGER NOT NULL"
            + " )";

    //borrar la tablas
    private static final String DROP_TABLE = "DROP TABLE IF EXISTS " +
ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME;
    private static final String DROP_TABLE_TLF = "DROP TABLE IF EXISTS
" + ControlGlucemicoContract.TelefonoEmergenciaEntry.TABLE_NAME;


    //Constructor
    public ControlGlucemicoDbHelper(Context context) {
        //lo que hace super es llamar al constructor del padre
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

```
    //al heredar de SQLiteOpenHelper, es condición necesaria
implementar estos dos métodos
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE);
        db.execSQL(CREATE_TABLE_TLF);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {
        db.execSQL(DROP_TABLE);
        db.execSQL(DROP_TABLE_TLF);

        onCreate(db);
    }

    //metodo para insertar datos en la tabla de los
controlesglucemicos
    public void insert (ControlGlucemico controlGlucemico,
SQLiteDatabase db) {
        String fecha =
ControlGlucemicoContract.ControlGlucemicoEntry.FECHA;
        String valor_toma =
ControlGlucemicoContract.ControlGlucemicoEntry.VALOR_TOMA;
        String nombre_toma =
ControlGlucemicoContract.ControlGlucemicoEntry.NOMBRE_TOMA;
        String insertar = "INSERT INTO " +
ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME
                + " (" + fecha + ", " + valor_toma + ", " +
nombre_toma
                + ") VALUES (\'" + controlGlucemico.getFecha() + "\',
"
                + controlGlucemico.getValor_toma() + ", \'" +
controlGlucemico.getNombre_toma() + "')";

        db.execSQL(insertar);
    }

    //metodo que inserta datos en la tabla del telefono de emergencia
    public void insertTlf(TelefonoEmergencia telefonoEmergencia,
SQLiteDatabase db){
        String telefono =
ControlGlucemicoContract.TelefonoEmergenciaEntry.TLF;
        String insertar = "INSERT INTO " +
ControlGlucemicoContract.TelefonoEmergenciaEntry.TABLE_NAME
                + " (" + telefono
                + ") VALUES (\'" + telefonoEmergencia.getTlf()
                + "')";

        db.execSQL(insertar);
    }

    //metodo para borrar datos en la tabla de los controlesglucemicos
    public void delete (ControlGlucemico controlGlucemico,
SQLiteDatabase db) {
        int id = controlGlucemico.getID();
        String delete = "DELETE FROM " +
```

```
ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME + " WHERE
_ID =" + id;
        db.execSQL(delete);

    }


    //metodo para borrar datos en la tabla del telefono de emergencia
    public void deleteTlf (TelefonoEmergencia telefonoEmergencia,
SQLiteDatabase db) {
        int id = telefonoEmergencia.getId();
        String delete = "DELETE FROM " +
ControlGlucemicoContract.TelefonoEmergenciaEntry.TABLE_NAME + " WHERE
_ID =" + id;
        db.execSQL(delete);
/*
        db.execSQL(DROP TABLE TLF);
        db.execSQL(CREATE_TABLE_TLF);
*/
    }

    //metodo para leer datos de la tabla de los controlesglucemicos
    public Cursor obtenCursorControles (SQLiteDatabase db) {
        String select = "SELECT * FROM " +
ControlGlucemicoContract.ControlGlucemicoEntry.TABLE_NAME;
        Cursor cursor = db.rawQuery(select, null);
        return cursor;
    }

    //metodo para leer datos de la tabla de los telefono de emergencia
    public Cursor obtenCursorTelefonoEmergencia (SQLiteDatabase db) {
        String select = "SELECT * FROM " +
ControlGlucemicoContract.TelefonoEmergenciaEntry.TABLE_NAME;
        Cursor cursor = db.rawQuery(select, null);
        return cursor;
    }

    //metodo que devuelve una lista de los controlesglucemicos
    public ArrayList <ControlGlucemico> obtenListaControles
(SQLiteDatabase db) {
        ArrayList <ControlGlucemico> lista = new
ArrayList<ControlGlucemico>();
        Cursor cursor = obtenCursorControles(db);
        if (cursor.getCount() > 0) {
            cursor.moveToFirst();
            do {
                int id = cursor.getInt(0);
                String fecha = cursor.getString(1);
                float valor = Float.parseFloat(cursor.getString(2));
                String nombre = cursor.getString(3);
                ControlGlucemico cg = new ControlGlucemico(id, fecha,
valor, nombre);
                lista.add(cg);
            } while (cursor.moveToNext());
        }
        return lista;
    }


    //metodo que devuelve una lista de los controlesglucemicos
    public TelefonoEmergencia obtenTelefonoEmergencia (SQLiteDatabase
```

```
db) {
        Cursor cursor = obtenCursorTelefonoEmergencia(db);
        if (cursor.getCount() > 0) {
            cursor.moveToFirst();
            do {
                int id = cursor.getInt(0);
                int tlf = cursor.getInt(1);
                TelefonoEmergencia telefonoEmergencia = new
TelefonoEmergencia(id, tlf);
                return  telefonoEmergencia;
            } while (cursor.moveToNext());
        }
        return null;
    }
}
```

ListaAdapter

```
package com.example.diabeticday.model;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;

import com.example.diabeticday.R;

import java.util.ArrayList;

public class ListaAdapter extends ArrayAdapter<ControlGlucemico> {
    private ArrayList <ControlGlucemico> lista;

    public ListaAdapter(Context context, ArrayList<ControlGlucemico>
lista) {
        super(context, R.layout.item_lista, lista);
        this.lista = lista;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup
parent) {

        //Inflate rellena los datos de los items
        LayoutInflater inflater = LayoutInflater.from(getContext());

        View item = inflater.inflate(R.layout.item_lista, null);

        TextView tvFecha = item.findViewById(R.id.tvFecha);
        tvFecha.setText(lista.get(position).getFecha());

        TextView tvNombre = item.findViewById(R.id.tvNombre);
        tvNombre.setText(lista.get(position).getNombre_toma());

        TextView tvValor = item.findViewById(R.id.tvValor);
        tvValor.setText(lista.get(position).getValor_toma() + "");
//concateno con comillas para hacer la conversión a String

        return item;
    }


}
```

TelefonoEmergencia

```
package com.example.diabeticday.model;

public class TelefonoEmergencia {
    final int id;
    int tlf;

    public TelefonoEmergencia(int id, int tlf) {
        this.id = id;
        this.tlf = tlf;
    }

    public int getId() {
        return id;
    }

    public int getTlf() {
        return tlf;
    }

    public void setTlf(int tlf) {
        this.tlf = tlf;
    }
}
```

Resources

Layout:

Activity_alarm_sound.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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/ic_launcher_background"
    android:orientation="vertical">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="@android:color/holo_red_light"
        android:onClick="stopAlarm"
        android:text="STOP"
        tools:layout_editor_absoluteX="148dp"
        tools:layout_editor_absoluteY="217dp" />

    <TextView
        android:id="@+id/tvActivityAlarmHora"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/tvHoraSigControl"
        android:textColor="@color/colorAccent"
        android:textSize="24sp" />

</LinearLayout>
```

Activity_controles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.ControlesActivity">

    <Button
        android:id="@+id/btNuevoControl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="@string/btNuevoControl"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <ListView
        android:id="@+id/lvControles"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@+id/btNuevoControl"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:listitem="@layout/item_lista"/>
</android.support.constraint.ConstraintLayout>
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.MainActivity"
    android:background="@drawable/ic_launcher_background">

    <TextView
        android:id="@+id/tvTitulo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="52dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="@string/tvTitulo"
        android:textColor="@color/colorAccent"
        android:textSize="36sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btInyecciones"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="7dp"
        android:layout_marginRight="7dp"
        android:background="@android:color/holo_red_light"
        android:maxWidth="300dp"
        android:minWidth="200dp"
        android:text="@string/btInyecciones"
        android:textColor="@android:color/background_light"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline2" />

    <Button
        android:id="@+id/btControles"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="70dp"
        android:background="@android:color/holo_red_light"
        android:maxWidth="300dp"
        android:minWidth="200dp"
        android:text="@string/btControles"
        android:textColor="@android:color/background_light"
        app:layout_constraintEnd_toEndOf="@+id/btInyecciones"
        app:layout_constraintStart_toStartOf="@+id/btInyecciones"
        app:layout_constraintTop_toBottomOf="@+id/btInyecciones" />

    <Button
```

```
        android:id="@+id/btSOS"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="73dp"
        android:background="@android:color/holo_red_light"
        android:maxWidth="300dp"
        android:minWidth="200dp"
        android:text="@string/btSOS"
        android:textColor="@android:color/background_light"
        app:layout_constraintEnd_toEndOf="@+id/btInyecciones"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="@+id/btInyecciones"
        app:layout_constraintTop_toBottomOf="@+id/btControles" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.28" />

</android.support.constraint.ConstraintLayout>
```

Activity_rellenar_campo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".activities.RellenarCampoActivity">

    <TextView
        android:id="@+id/tvFechaRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:text="@string/tvFechaRellenar"
        app:layout_constraintBottom_toBottomOf="@+id/etFechaRellenar"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/etFechaRellenar" />

    <TextView
        android:id="@+id/tvNombreRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:text="@string/tvNombreRellenar"
        app:layout_constraintBottom_toBottomOf="@+id/etNombreRellenar"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/etNombreRellenar" />

    <TextView
        android:id="@+id/tvValorRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:text="@string/tvValorRellenar"
        app:layout_constraintBottom_toBottomOf="@+id/etValorRellenar"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/etValorRellenar" />

    <EditText
        android:id="@+id/etFechaRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tvFechaRellenar"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etValorRellenar"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:ems="10"
        android:inputType="numberDecimal"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tvValorRellenar"
        app:layout_constraintTop_toBottomOf="@+id/etFechaRellenar" />

    <EditText
        android:id="@+id/etNombreRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:ems="10"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/tvNombreRellenar"
        app:layout_constraintTop_toBottomOf="@+id/etValorRellenar" />

    <Button
        android:id="@+id/btAceptarRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="@string/btAceptarRellenar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/guideline3"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btCancelarRellenar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        android:text="@string/btCancelarRellenar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="@+id/guideline3" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
```

```
        app:layout_constraintGuide_percent="0.5" />
</android.support.constraint.ConstraintLayout>
```

Item_lista.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/tvFecha"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="Fecha"
        app:layout_constraintEnd_toStartOf="@+id/guideline"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Nombre"
        app:layout_constraintEnd_toEndOf="@+id/tvFecha"
        app:layout_constraintStart_toStartOf="@+id/tvFecha"
        app:layout_constraintTop_toBottomOf="@+id/tvFecha" />

    <TextView
        android:id="@+id/tvValor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:text="10"
        android:textSize="@android:dimen/app_icon_size"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.487"
        app:layout_constraintStart_toStartOf="@+id/guideline"
        app:layout_constraintTop_toTopOf="parent" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.5" />
</android.support.constraint.ConstraintLayout>
```

Values:

Colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#008577</color>
    <color name="colorPrimaryDark">#00574B</color>
    <color name="colorAccent">#ef8c8c</color>
</resources>
```

Strings.xml

```
<resources>
    <string name="app_name">Diabeticday</string>
    <string name="tvTitulo">DiabeticDay</string>
    <string name="btControles">Controles</string>
    <string name="btInyecciones">Inyecciones</string>
    <string name="btSOS">S.O.S.</string>
    <string name="btNuevoControl">Nuevo Control</string>
    <string name="tvFechaRellenar">Fecha</string>
    <string name="tvValorRellenar">Valor</string>
    <string name="tvNombreRellenar">Nombre</string>
    <string name="btAceptarRellenar">Aceptar</string>
    <string name="btCancelarRellenar">Cancelar</string>
    <string name="tvHoraSigControl">Es la hora del siguiente
control</string>

</resources>
```

Styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme"
parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```

Build.gradle (Proyect: Diabeticday)

```
// Top-level build file where you can add configuration options common
to all sub-projects/modules.

buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.0'


        // NOTE: Do not place your application dependencies here; they
belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Build.gradle (Module: app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.example.diabeticday"
        minSdkVersion 15
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
}
```