

**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**ESPOL**

**MATERIA: PROGRAMACION DE SISTEMAS TELEMATICOS**

**TÍTULO DEL INFORME:**

**“DISPENSADOR DE COMIDA PARA MASCOTAS”**

**INTEGRANTES:**

**CALI LOAIZA BRYAN FABRICIO**

**TORRES CEDEÑO ROBERTO JOSUE**

**AREVALO GOYES JONNATHAN STEVEN**

**ALVARADO PELAEZ DANIEL ANDRES**

**GARCIA ROMO TAMARA KARINA**

**PARALELO: 1**

**PROFESOR:**

**MSIG. COLLAGUAZO JARAMILLO ADRIANA ELISA**

**FECHA DE PRESENTACIÓN:**

**MARTES, 1 DE SEPTIEMBRE DEL 2020**

**GUAYAQUIL - ECUADOR**

**1. MARCO TEÓRICO**

**1.1. BASES TEÓRICAS**

**1.1.1. Bases teóricas veterinarias**

La alimentación de las mascotas es un tema muy importante que tratar, para una buena de la mascota. Así como el ser humano se llega a enfermar por una mala alimentación, nuestras mascotas no están exentas de ello, por lo cual pueden sufrir varias enfermedades como:

* **Diarrea:** Son cuando las heces se presentan en forma blanda o liquida, producto de alimentos descompuestos o por una variación en su dieta.
* **Gastritis:** Se produce por la ingesta de alimentos caducos o en malas condiciones, así también por alimentos caseros, que pueden hallar las mascotas en la basura.
* **Desnutrición:** Sucede cuando tienen una dieta desequilibrada por falta de nutrientes, por lo general suele presentarse en las mascotas sin hogar, pero también las de casa puede llegar a padecer.

**1.1.2. Bases teóricas electrónica**

**1.1.2.1. Microcontrolador**

Es un circuito integrado utilizado en todo tipo de inventos que sea automático dependiendo de la necesidad del usuario; mejor comparado como una minicomputadora que controla elementos de entrada y salida; al tener procesador, memoria flash y RAM.

**1.1.2.2 Principio motor DC**

Basado en la ley de Fuerza, al recibir una corriente eléctrica un conductor es sumergido en un campo magnético, por lo cual recibe una fuerza perpendicular al plano formado por la corriente y el campo, según la regla de la mano derecha.

**1.1.3. Bases teóricas sistemas**

**1.1.3.1. Lenguaje C++**

El lenguaje C++ esta fundado por C, siendo uno de los lenguajes más antiguos, desarrollado desde 1979. Se tuvo que un tiempo fuera de la disponibilidad del publico hasta 1985, siendo hasta la fecha un lenguaje muy popular; siendo un programa estándar ISO (Organización Internacional de Normalización) siendo desde bajo nivel y eficiente hasta ser complejo con optimas capacidades. Este lenguaje gana publico por su biblioteca estándar, estando a la mano para empresas y terceros que desean crear su propio programa mediante este lenguaje.

**1.1.3.2. Lenguaje Java**

Es una plataforma y lenguaje informático comercializado desde 1995 por Sun Microsystems, siendo en la actualidad necesario su instalación para muchas aplicaciones y sitios web; su popularidad se debió a que es seguro, rápido y fiable. Siendo un lenguaje multiplataforma, estando en portátiles, consolas y teléfonos.

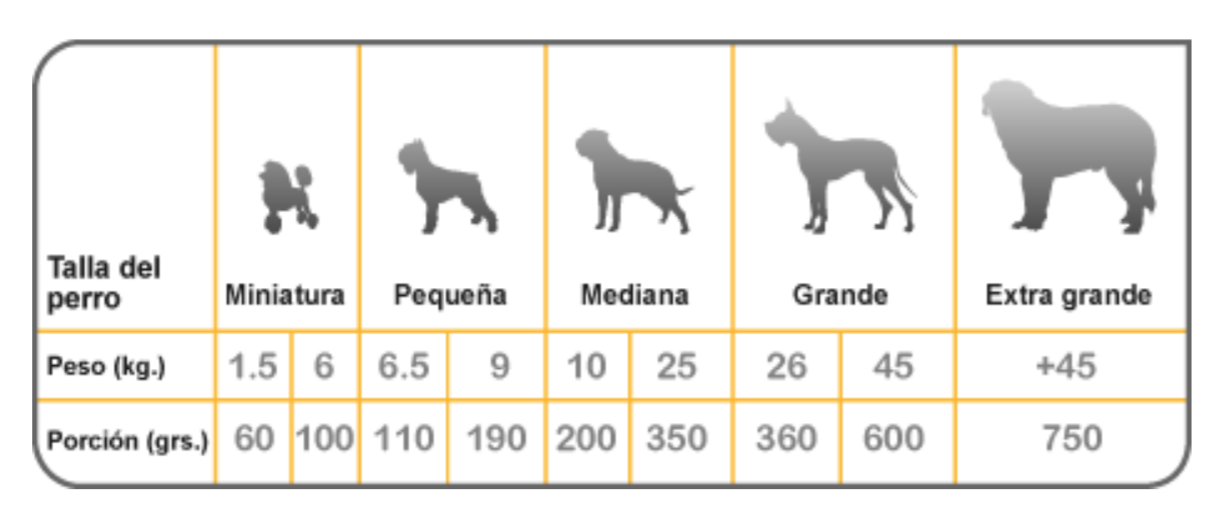
**1.2. MARCO CONCEPTUAL**

**1.2.1. Marco conceptual veterinario**

La alimentación de las mascotas depende del desgate físico y las calorías quemadas, para así determinar la cantidad de proteínas que faltan en su cuerpo; en la cual recuperara esa energía con comida especial para ellos y agua.

También es de tener muy en cuenta la edad de la mascota y su raza al saber si es pequeña, mediana o grande. Las vitaminas necesarias de una cría son mucho mayores que la de una mascota adulta.

**1.2.1.1. Tabla de alimentación de las mascotas según su tamaño y peso.**



**Figura 1** Porción de comida según el peso de la mascota

**Fuente:** (Nutrición Canis, 2010)

**1.2.2. Marco conceptual sistemas**

**1.2.2.1. Software Android**

Es un sistema operativo móvil para teléfonos inteligentes o tablets, que actualmente también lo encontramos en relojes inteligentes, televisores e incluso algunos modelos de carros. Fue desarrollado por Google y basado en Kernel de Linux, con códigos abiertos, para desarrolladores que desean crear diversas aplicaciones.

**1.2.3 Marco conceptual electrónico**

**1.2.3.1. Arduino UNO**

Es una placa muy utilizada al iniciar en el mundo de la programación y electrónica; siendo este una buena opción para tener una buena experiencia como principiantes, al tener mayor cantidad de información entre todos los demás tipos de Arduino.

Este tipo de Arduino UNO, esta basado en un microcontrolador ATmega328P. Esta compuesto por 14 pines de entrada y salida digitales, con 6 entradas analógicas, una conexión USB, conector Jack de alimentación, y un botón de RESET. El Arduino UNO se lo debe conectar con un transformador AC-DC o por el puerto USB.

**1.2.3.2 Motor DC**

Es un motor de corriente continua, que su principal función es convertir la energía eléctrica en mecánica, al producir un movimiento rotatorio, generado por un campo magnético. Este tipo de motor está compuesto por dos partes:

* **Estator:** Brinda soporte mecánico siendo la parte fija, conteniendo los polos siendo imanes permanentes o devanados.
* **Rotor:** LA mayoría de las veces lo encontramos en forma cilíndrica, con núcleo y devanado, nutrido con corriente a través de escobillas fijas de carbono.

**REFERENCIAS BIBLIOGRÁFICAS**

* Osorio-Carmona, E., Giraldo-Carmona, J., & Narváez-Solarte, W. (2012). Metodologías para determinar la digestibilidad de los alimentos utilizados en la alimentación canina. *vet. zootec*, *6*(1), 87-97.
* Valdés, F., & Areny, R. P. (2007). *Microcontroladores fundamentos y aplicaciones con PIC* (Vol. 1149). Marcombo.
* Sl Didact. (2005). *Manual de programacion lenguaje C++*. MAD-Eduforma.
* Groussard, T. (2012). *JAVA 7: Los fundamentos del lenguaje Java*. Ediciones Eni.
* Dauvergne, C. (2018). *La alimentación del perro*. Parkstone International.
* Rountev, A., & Yan, D. (2014, February). Static reference analysis for GUI objects in Android software. In *Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization* (pp. 143-153).
* Arduino, S. A. (2015). Arduino. *Arduino LLC*.
* Yedamale, P. (2003). Brushless DC (BLDC) motor fundamentals. *Microchip Technology Inc*, *20*, 3-15.

PASO: HORA-NOTIFICACION

**AndroidManifest.xml:**

**Se otorgan los permisos necesario a la aplicación para poder mostrar las notificaciones, realizar el evento de alarma y poder recibir eventos de reinicio. Se declaran los servicios y receptores que reaccionaran a los eventos.**

*<?***xml version="1.0" encoding="utf-8"***?>*<**manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.example.proyecto"**>  
 <**uses-permission android:name="android.permission.INTERNET"** />  
 <**uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"**/>  
 <**uses-permission android:name="android.permission.WRITE\_CALENDAR"** />  
 <**uses-permission android:name="android.permission.READ\_CALENDAR"** />  
 <**uses-permission android:name="android.permission.FOREGROUND\_SERVICE"** />  
 <**uses-permission android:name="android.permission.WAKE\_LOCK"** />  
 <**uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"** />  
 <**uses-permission android:name="com.google.android.gms.permission.ACTIVITY\_RECOGNITION"** />  
 <**uses-permission android:name="android.permission.VIBRATE"** />  
 <**uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED"**/>  
 <**application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_icone"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_icone\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme"**>  
 <**activity android:name=".RegistroMascota"**/>  
 <**activity android:name=".RegistroUsuario"**/>  
 <**activity android:name=".UsuarioAc"**/>  
 <**activity android:name=".HorariodeComida"**/>  
 <**activity android:name=".DetalleMascotas"**/>  
 <**activity android:name=".ConsultarListaPersonas"**/>  
 <**activity android:name=".ListaMascotas"**/>  
 <**activity android:name=".Login"**/>  
 <**activity android:name=".DetallesdelUsuario"**/>  
 <**activity android:name=".MainActivity"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.MAIN"** />  
  
 <**category android:name="android.intent.category.LAUNCHER"** />  
 </**intent-filter**>  
 </**activity**>  
 <**receiver  
 android:name="com.example.proyecto.entidades.AlarmReceiver"  
 android:enabled="true"** />  
  
 <**service  
 android:name="com.example.proyecto.entidades.NotificationService"  
 android:enabled="true"** />  
  
 <**service android:name="com.example.proyecto.entidades.RebootServiceClass"**/>  
  
 <**receiver android:name="com.example.proyecto.entidades.MyRebootReceiver"**>  
 <**intent-filter**>  
 <**action android:name="android.intent.action.BOOT\_COMPLETED"**/>  
 </**intent-filter**>  
 </**receiver**>  
 </**application**>  
  
</**manifest**>

**AlarmReceiver.java:**

**Recibe y guarda una alarma y cuando salte el evento llama a la clase que se encarga de diseñar y mostrar la notificación.**

**package** com.example.proyecto.entidades;  
  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.net.Uri;  
**import** android.util.Log;  
  
**import** androidx.core.content.ContextCompat;**public class** AlarmReceiver **extends** BroadcastReceiver {  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 Intent service1 = **new** Intent(context, NotificationService.**class**);  
 service1.setData((Uri.*parse*(**"custom://"** + System.*currentTimeMillis*())));  
 ContextCompat.*startForegroundService*(context, service1 );  
 Log.*d*(**"WALKIRIA"**, **" ALARM RECEIVED!!!"**);

**MyRebootReceiver.java:**

**Se activa si recibe un evento de reinicio y llama a un servicio que configura la alarma guardada previamente.**

**package** com.example.proyecto.entidades;  
  
**import** android.content.BroadcastReceiver;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.os.Build;**public class** MyRebootReceiver **extends** BroadcastReceiver {  
  
 @Override  
 **public void** onReceive(Context context, Intent intent) {  
 Intent serviceIntent = **new** Intent(context, RebootServiceClass.**class**);  
 serviceIntent.putExtra(**"caller"**, **"RebootReceiver"**);  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***) {  
 context.startForegroundService(serviceIntent);  
 } **else** {  
 context.startService(serviceIntent);  
 }  
 }  
}

**RebootServiceClass.java:**

**Revisa la hora de la alarma para poder leer a qué hora debe sonar y activarla cuando se dispare el evento.**

**package** com.example.proyecto.entidades;  
  
**import** android.app.IntentService;  
**import** android.app.Notification;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
  
**import** com.example.proyecto.R;**public class** RebootServiceClass **extends** IntentService {  
 **public** RebootServiceClass(String name) {  
 **super**(name);  
 startForeground(1, **new** Notification());  
 }  
  
 **public** RebootServiceClass() {  
 **super**(**"RebootServiceClass"**);  
 }  
  
 @Override  
 **public void** onCreate() {  
 **super**.onCreate();  
 }  
  
 @Override  
 *//Cuando obtiene el extra y es “Rebootreceiver”, va a leer el id de la alarma y su hora y reguardarla* **protected void** onHandleIntent(Intent intent) {  
 String intentType = intent.getExtras().getString(**"caller"**);  
 **if** (intentType == **null**) **return**;  
 **if** (intentType.equals(**"RebootReceiver"**)) {  
 SharedPreferences settings = getSharedPreferences(getString(R.string.***app\_name***), Context.***MODE\_PRIVATE***);  
 Utils.*setAlarm*(settings.getInt(**"alarmID"**, 0), settings.getLong(**"alarmTime"**, 0), **this**);  
 }  
 }  
}

**Utils.java:**

**Recibe la información de la alarma y la guarda en AlarmaManager.**

**package** com.example.proyecto.entidades;  
  
**import** android.app.AlarmManager;  
**import** android.app.PendingIntent;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.net.Uri;  
  
**import static** android.content.Context.***ALARM\_SERVICE***;  
*//Guarda la alarma en AlarmaManager***public class** Utils {  
  
 **public static void** setAlarm(**int** i, Long timestamp, Context ctx) {  
 AlarmManager alarmManager = (AlarmManager) ctx.getSystemService(***ALARM\_SERVICE***);  
 Intent alarmIntent = **new** Intent(ctx, AlarmReceiver.**class**);  
 PendingIntent pendingIntent;  
 pendingIntent = PendingIntent.*getBroadcast*(ctx, i, alarmIntent, PendingIntent.***FLAG\_ONE\_SHOT***);  
 alarmIntent.setData((Uri.*parse*(**"custom://"** + System.*currentTimeMillis*())));  
 alarmManager.set(AlarmManager.***RTC\_WAKEUP***, timestamp, pendingIntent);  
 }  
}

**NotificationService.java:**

**Creará la notificación que se mostrara cuando se dispare el evento, además de darle un mensaje, el nombre de la app, un tono o vibración y un mini-logo.**

**package** com.example.proyecto.entidades;  
  
**import** android.annotation.TargetApi;  
**import** android.app.IntentService;  
**import** android.app.Notification;  
**import** android.app.NotificationChannel;  
**import** android.app.NotificationManager;  
**import** android.app.PendingIntent;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.res.Resources;  
**import** android.graphics.BitmapFactory;  
**import** android.media.RingtoneManager;  
**import** android.net.Uri;  
**import** android.os.Build;  
  
**import** androidx.core.app.NotificationCompat;  
  
**import** com.example.proyecto.HorariodeComida;  
**import** com.example.proyecto.R;  
*//Muestra la notificacion en la barra de tareas, cuando según la hora establecida***public class** NotificationService **extends** IntentService {  
  
 **private** NotificationManager **notificationManager**;  
 **private** PendingIntent **pendingIntent**;  
 **private static int** *NOTIFICATION\_ID* = 1;  
 Notification **notification**;  
  
  
 **public** NotificationService(String name) {  
 **super**(name);  
 }  
  
 **public** NotificationService() {  
 **super**(**"SERVICE"**);  
 }  
  
 @TargetApi(Build.VERSION\_CODES.***O***)  
 @Override  
 **protected void** onHandleIntent(Intent intent2) {  
 String NOTIFICATION\_CHANNEL\_ID = getApplicationContext().getString(R.string.***app\_name***);  
 Context context = **this**.getApplicationContext();  
 **notificationManager** = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 Intent mIntent = **new** Intent(**this**, HorariodeComida.**class**);  
 Resources res = **this**.getResources();  
 Uri soundUri = RingtoneManager.*getDefaultUri*(RingtoneManager.***TYPE\_ALARM***);  
  
 String message = getString(R.string.***new\_notification***);  
  
 **if** (Build.VERSION.***SDK\_INT*** >= Build.VERSION\_CODES.***O***) {  
 **final int** NOTIFY\_ID = 0; *// ID of notification* String id = NOTIFICATION\_CHANNEL\_ID; *// default\_channel\_id* String title = NOTIFICATION\_CHANNEL\_ID; *// Default Channel* PendingIntent pendingIntent;  
 NotificationCompat.Builder builder;  
 NotificationManager notifManager = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 **if** (notifManager == **null**) {  
 notifManager = (NotificationManager) context.getSystemService(Context.***NOTIFICATION\_SERVICE***);  
 }  
 **int** importance = NotificationManager.***IMPORTANCE\_HIGH***;  
 NotificationChannel mChannel = notifManager.getNotificationChannel(id);  
 *//Se encarga de dar un patron de vibración a la notificación* **if** (mChannel == **null**) {  
 mChannel = **new** NotificationChannel(id, title, importance);  
 mChannel.enableVibration(**true**);  
 mChannel.setVibrationPattern(**new long**[]{100, 200, 300, 400, 500, 400, 300, 200, 400});  
 notifManager.createNotificationChannel(mChannel);  
 }  
 *//Se crea el mensaje a mostrar en la notificacion y un icono* builder = **new** NotificationCompat.Builder(context, id);  
 mIntent.setFlags(Intent.***FLAG\_ACTIVITY\_CLEAR\_TOP*** | Intent.***FLAG\_ACTIVITY\_SINGLE\_TOP***);  
 pendingIntent = PendingIntent.*getActivity*(context, 0, mIntent, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
 builder.setContentTitle(getString(R.string.***app\_name***)).setCategory(Notification.***CATEGORY\_SERVICE***)  
 .setSmallIcon(R.drawable.***ic\_notification***) *// required* .setContentText(message)  
 .setLargeIcon(BitmapFactory.*decodeResource*(res, R.drawable.***ic\_notification***))  
 .setDefaults(Notification.***DEFAULT\_ALL***)  
 .setAutoCancel(**true**)  
 .setSound(soundUri)  
  
 .setContentIntent(pendingIntent)  
 .setVibrate(**new long**[]{100, 200, 300, 400, 500, 400, 300, 200, 400});  
 Notification notification = builder.build();  
 notifManager.notify(NOTIFY\_ID, notification);  
  
 startForeground(1, notification);  
  
 } **else** {  
 **pendingIntent** = PendingIntent.*getActivity*(context, 1, mIntent, PendingIntent.***FLAG\_UPDATE\_CURRENT***);  
 **notification** = **new** NotificationCompat.Builder(**this**)  
 .setContentIntent(**pendingIntent**)  
 .setSmallIcon(R.drawable.***ic\_notification***)  
 .setLargeIcon(BitmapFactory.*decodeResource*(res, R.drawable.***ic\_notification***))  
 .setSound(soundUri)  
 .setAutoCancel(**true**)  
 .setContentTitle(getString(R.string.***app\_name***)).setCategory(Notification.***CATEGORY\_SERVICE***)  
 .setContentText(message).build();  
 **notificationManager**.notify(*NOTIFICATION\_ID*, **notification**);  
 }  
 }  
}

**HorariodeComida.java:**

**Se permite agregar y eliminar horas en los que se mandaran las notificaciones, además de guardarlas en la base de datos de horario y mostrarlas en tiempo real, como parámetro en caso de que se reinicie el sistema.**

**package** com.example.proyecto;  
  
**import** androidx.appcompat.app.AppCompatActivity;  
  
**import** android.app.TimePickerDialog;  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.content.Intent;  
**import** android.content.SharedPreferences;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.AdapterView;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.EditText;  
**import** android.widget.ListView;  
**import** android.widget.TextView;  
**import** android.widget.TimePicker;  
**import** android.widget.Toast;  
  
**import** com.example.proyecto.entidades.AdminSQLiteOpenHelper;  
**import** com.example.proyecto.entidades.Horario;  
**import** com.example.proyecto.entidades.Usuario;  
**import** com.example.proyecto.entidades.Utils;  
**import** com.example.proyecto.utilidades.Utilidades;  
  
**import** java.util.ArrayList;  
**import** java.util.Calendar;  
  
**import static** android.view.View.\*;  
*//Ventana que permite manejar a que hora se desea las notificaciones***public class** HorariodeComida **extends** AppCompatActivity {  
 **private** TextView **notificationsTime**;  
 **private int alarmID**=1;  
 **private** SharedPreferences **settings**;  
 **private** EditText **descripcion**;  
 ListView **listaH**;  
 ArrayList<String> **listaInformacion**;  
 ArrayList<Horario> **listahora**;  
 AdminSQLiteOpenHelper **admin**;  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//Se declaran las variables* setContentView(R.layout.***activity\_horariode\_comida***);  
 **settings** = getSharedPreferences(getString(R.string.***app\_name***), Context.***MODE\_PRIVATE***);  
  
 **listaH** = (ListView) findViewById(R.id.***ListaHora***);  
  
 **notificationsTime** = (TextView) findViewById(R.id.***notifications\_time***);  
  
 **admin** = **new** AdminSQLiteOpenHelper(**this**,**"PST\_G6"**,**null**,1);  
  
 **descripcion** = (EditText) findViewById(R.id.***edit***);  
 String hour, minute;  
 hour = **settings**.getString(**"hour"**, **""**);  
 minute = **settings**.getString(**"minute"**, **""**);  
 *//Se revisa si ya existen alarmas para poder mostrarlas en la misma ventana* consultarListahoras();  
 *//Crea un arreglo que mostrara informacion de cada alarma* ArrayAdapter adaptador=**new** ArrayAdapter(**this**,android.R.layout.***simple\_list\_item\_1***,**listaInformacion**);  
 **listaH**.setAdapter(adaptador);  
 **listaH**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> adapterView, View view, **int** pos, **long** l) {  
 String informacion= **listahora**.get(pos).getHora();  
 **notificationsTime**.setText(informacion);  
 }  
 });  
 }  
  
 **private void** consultarListahoras() {  
*//Se lee la base de datos y se consulta que horas hay existentes* SQLiteDatabase db=**admin**.getReadableDatabase();  
 Horario hora =**null**;  
 **listahora** =**new** ArrayList<Horario>();  
 Cursor cursor=db.rawQuery(**"SELECT \* FROM "**+ Utilidades.***TABLA\_HORARIO***,**null**);  
  
 **while** (cursor.moveToNext()){  
 hora=**new** Horario();  
 hora.setId(cursor.getInt(0));  
 hora.setHora(cursor.getString(1));  
 hora.setDescripcion(cursor.getString(2));  
  
  
 **listahora**.add(hora);  
 }  
 obtenerLista();  
 }  
 **public void** obtenerLista() {  
 *//Se crea una lista con informacion sobre la alarma* **listaInformacion** = **new** ArrayList<String>();  
  
 **for** (**int** i = 0; i < **listahora**.size(); i++) {  
 **listaInformacion**.add(**listahora**.get(i).getHora()+**" - "**+**listahora**.get(i).getDescripcion());  
 }  
 }  
 **public void** eliminarUsuario(View view) {  
 *//Se elimina la alarma que el usuario haya escogido , y se vuelve a cargar el activity para mostrar el cambio* SQLiteDatabase db=**admin**.getWritableDatabase();  
 String[] parametros={**notificationsTime**.getText().toString()};  
  
 db.delete(Utilidades.***TABLA\_HORARIO***,Utilidades.***KEY\_HORARIO\_HORA***+**"=?"**,parametros);  
 Toast.*makeText*(getApplicationContext(),**"Se eliminó el recordatorio "**,Toast.***LENGTH\_LONG***).show();  
 db.close();  
 cargar();  
 }  
 **public void** agregarUsuario(View view) {  
 *//Utiliza la clase calendar para poder mostrar y obterner hora y minutos que se desea la alarma* Calendar mcurrentTime = Calendar.*getInstance*();  
 **int** hour = mcurrentTime.get(Calendar.***HOUR\_OF\_DAY***);  
 **int** minute = mcurrentTime.get(Calendar.***MINUTE***);  
 TimePickerDialog mTimePicker;  
 mTimePicker = **new** TimePickerDialog(HorariodeComida.**this**, **new** TimePickerDialog.OnTimeSetListener() {  
 @Override  
 **public void** onTimeSet(TimePicker timePicker, **int** selectedHour, **int** selectedMinute) {  
 String finalHour, finalMinute=**"0"**;  
 *//Se obtienen las variables del tiempo y se procede a mostrarla* finalHour = **""** + selectedHour;  
 finalMinute = **""** + selectedMinute;  
 **if** (selectedHour < 10) finalHour = **"0"** + selectedHour;  
 **if** (selectedMinute < 10) finalMinute = **"0"** + selectedMinute;  
 **notificationsTime**.setText(finalHour + **":"** + finalMinute);  
  
 Calendar today = Calendar.*getInstance*();  
  
 today.set(Calendar.***HOUR\_OF\_DAY***, selectedHour);  
 today.set(Calendar.***MINUTE***, selectedMinute);  
 today.set(Calendar.***SECOND***, 0);  
  
 SharedPreferences.Editor edit = **settings**.edit();  
 edit.putString(**"hour"**, finalHour);  
 edit.putString(**"minute"**, finalMinute);  
  
 *//Se guarda la hora seleccionada para que se cree la alarma si se reinicia el sistema* edit.putInt(**"alarmID"**, **alarmID**);  
 edit.putLong(**"alarmTime"**, today.getTimeInMillis());  
  
 edit.commit();  
 *//Se guarda la alarma en la base de datos* SQLiteDatabase db= **admin**.getWritableDatabase();  
  
 ContentValues values=**new** ContentValues();  
 values.put(Utilidades.***KEY\_HORARIO\_HORA***,finalHour + **":"** + finalMinute);  
 values.put(Utilidades.***KEY\_HORARIO\_DESCRIPCION***,**descripcion**.getText().toString());  
  
  
 Long idResultante=db.insert(Utilidades.***TABLA\_HORARIO***,Utilidades.***KEY\_HORARIO\_ID***,values);  
  
 Toast.*makeText*(getApplicationContext(),**"Alarma registrada "**+idResultante+**"\nHora: "**+finalHour + **":"** + finalMinute,Toast.***LENGTH\_SHORT***).show();  
 db.close();  
 *//Se programa la notificacion acorde la alarma* Utils.*setAlarm*(**alarmID**, today.getTimeInMillis(), HorariodeComida.**this**);  
 **if** (finalHour != **"0"** && finalMinute != **"0"**){  
 cargar();  
 }  
  
 }  
 }, hour, minute, **true**);*//Yes 24 hour time* mTimePicker.setTitle(getString(R.string.***select\_time***));  
 mTimePicker.show();  
  
 }  
 **private void** cargar() {  
 *//Vuelve a cargar el activity para que se muestre los cambios* overridePendingTransition(0, 0);  
 finish();  
 overridePendingTransition(0, 0);  
 startActivity(getIntent());  
 overridePendingTransition(0, 0);  
 }  
}