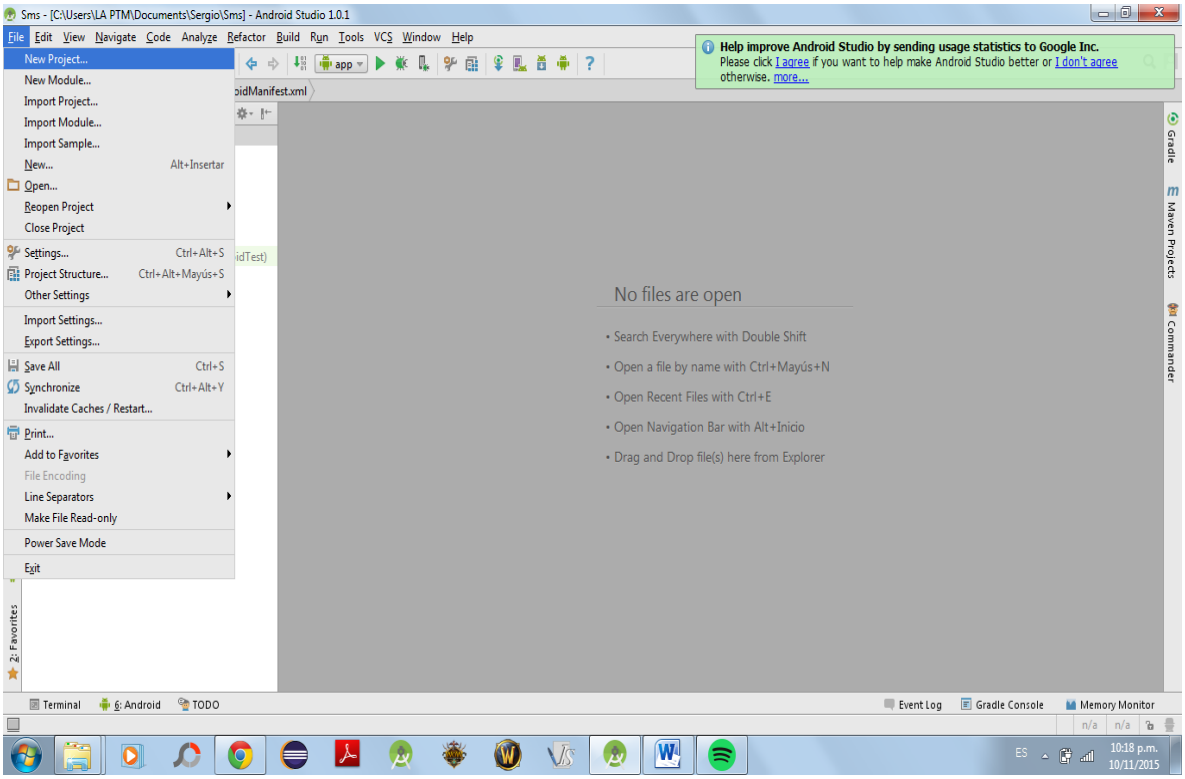
Envió y notificación de Sms en android

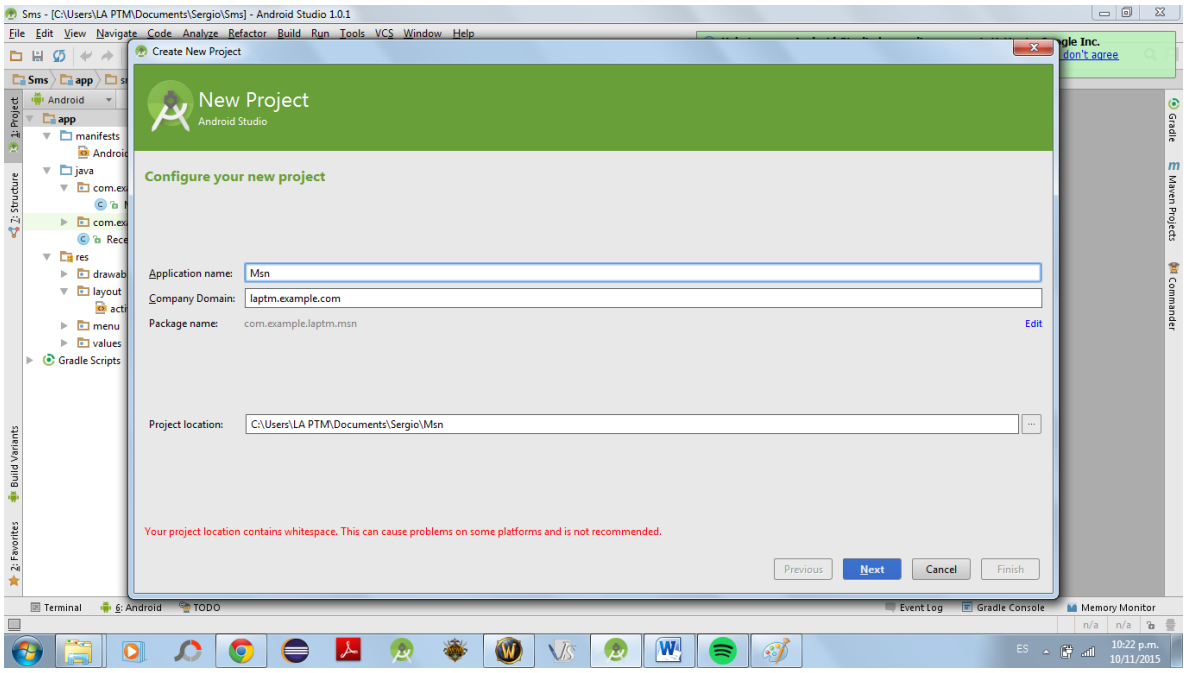
En este tutorial se describe la forma en la que se elabora una aplicación para mandar y recibir mensajes.

Para poder iniciar este tutorial es necesario tener la Plataforma “Android Studio”

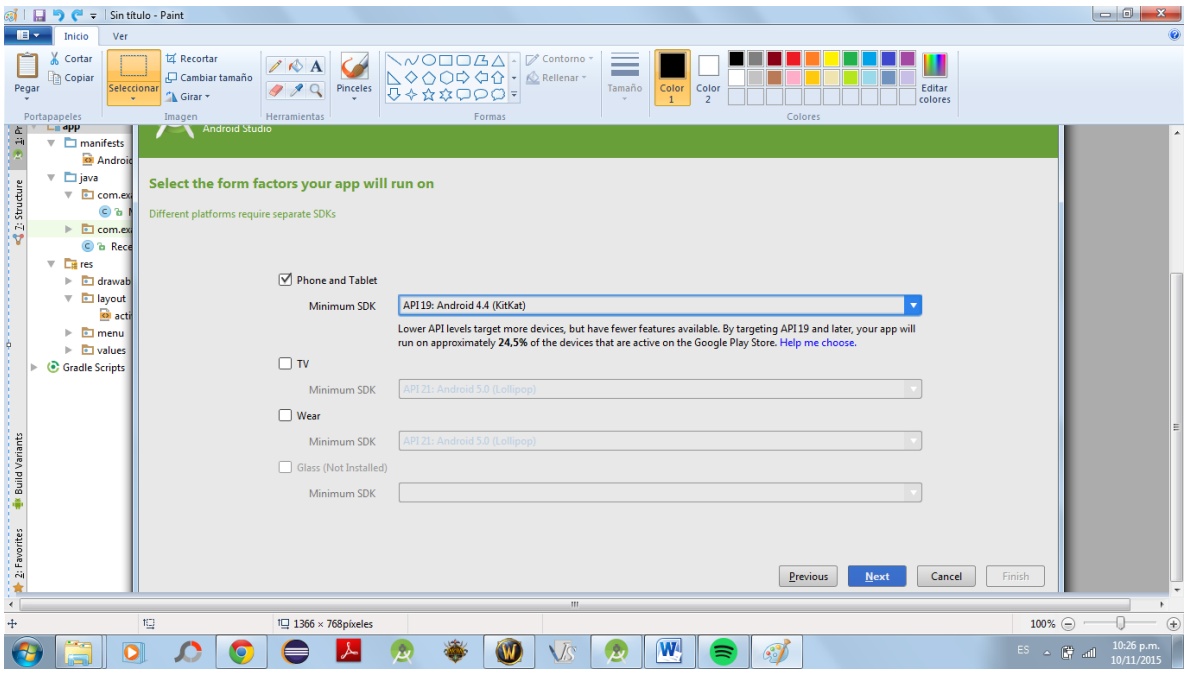
Como primer paso Iniciaremos el programa, después de hecho esto creamos un proyecto en Android Studio, para esto tendremos que ir a file en la parte superior izquierda de nuestra ventana, fijamos file luego de new proyect



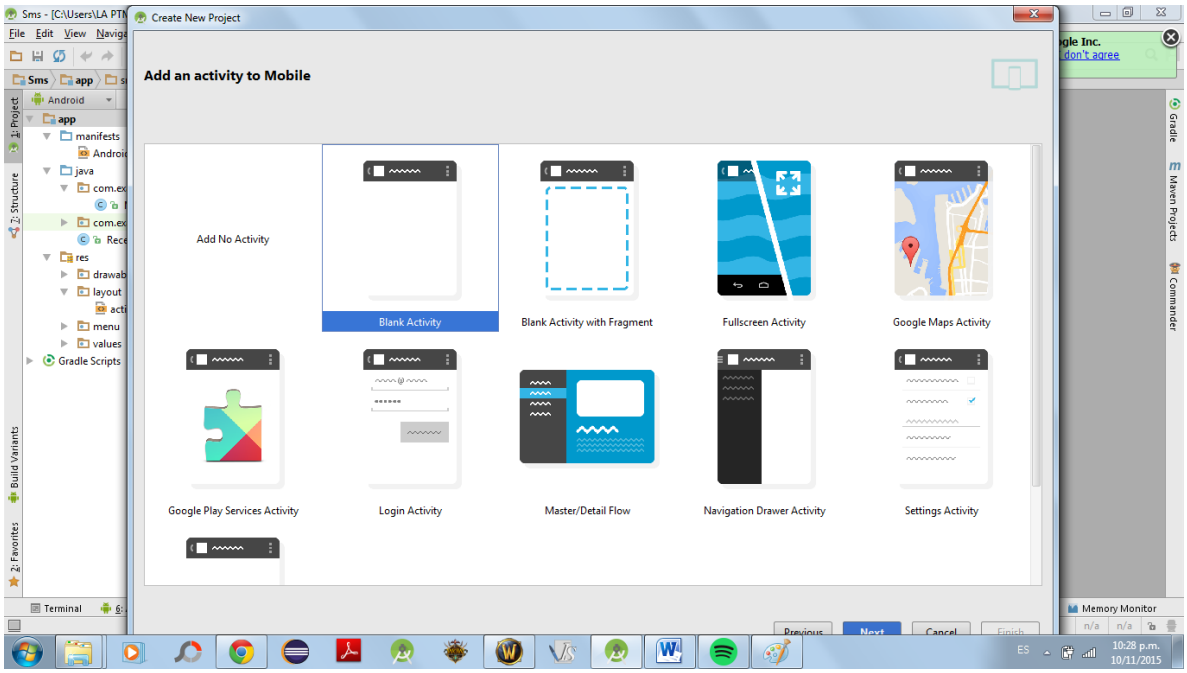
A continuación nos saldrá una ventana con la cual le pondremos el nombre a nuestro proyecto, en este caso “Msn” y seguido de esto pulsamos “Next”



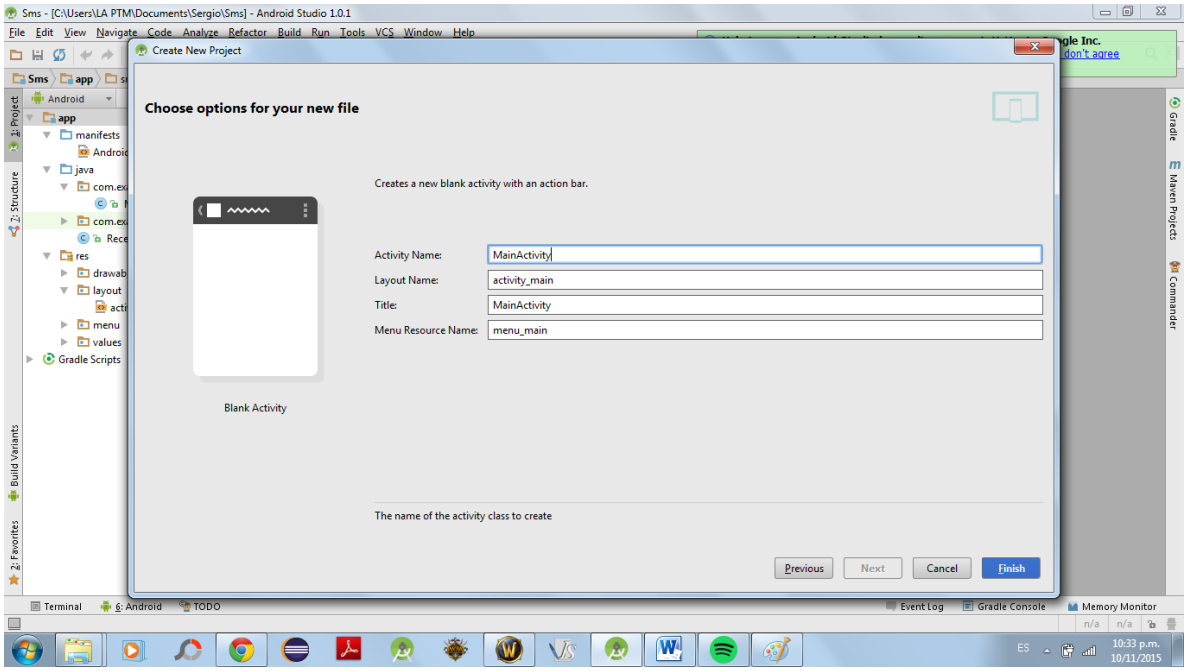
Nos pedirá establecer la versión hasta la cual se podra soportar la aplicación, en este caso la deje predeterminada (4.4 KitKat)



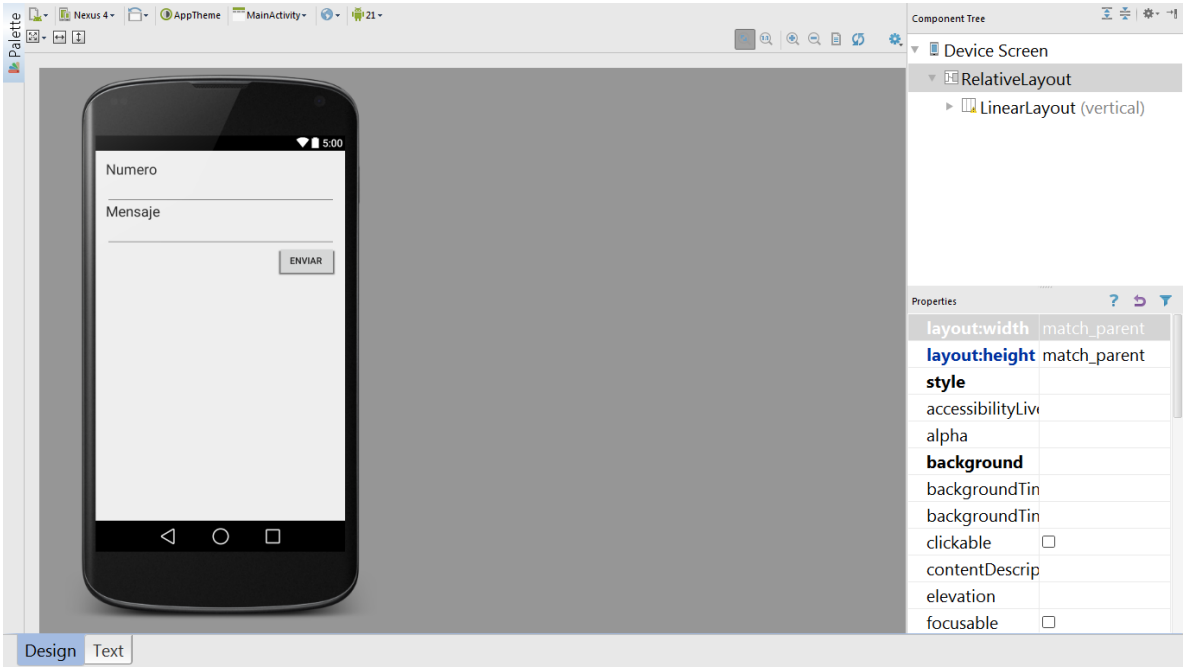
Escogemos Actividad en blanco, ya que no usaremos las demás herramientas y pulsamos “Next”



Y para terminar con la creación del proyecto, le damos el nombre a la actividad o clase principal, en este caso la deje como “MainActivity”, y pulsamos “Finish”



Quedando de esta manera, lo siguiente será crear nuestra interfaz de usuario, simplemente se usaron 2 campos de texto (TextField) y un botón (Button) Quedando de esta manera




```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Numero"
    android:id="@+id/textView" />

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/editNumero" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
    android:text="Mensaje"
    android:id="@+id/textView2" />

<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

Design Text

```
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:text="Mensaje"
            android:id="@+id/textView2" />

        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/editMensaje" />

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Enviar "
            android:id="@+id/botonEnviar"
            android:layout_gravity="right" />
    </LinearLayout>

</RelativeLayout>
```

Design Text

Vamos al MainActivity y Declaramos las variables globales y asignamos los controles a esas variables globales

```
kage com.example.laptm.sms;

import ...


public class MainActivity extends ActionBarActivity {

    EditText editNumero;
    EditText editMensaje;
    Button botonEnviar;

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

        editNumero = (EditText)findViewById(R.id.editNumero);
        editMensaje = (EditText)findViewById(R.id.editMensaje);
        botonEnviar =  (Button)findViewById(R.id.botonEnviar);

        botonEnviar.setOnClickListener(new View.OnClickListener() {

            @Override
```

Continuamos configurando el botón con el método setOnClickListener para que cada vez que oprimamos el botón sea escuchado y llame a un método que llamamos envioSms el cual convertirá los valores de los campos de texto en String para asi poder ser enviados y puedan de igual manera ser recibidos., uno es el numero del teléfono del móvil al cual vamos a enviar el SMS y el otro es el mensaje que enviaremos

```
    EditText editNumero;
    EditText editMensaje;
    Button botonEnviar;

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

        editNumero = (EditText)findViewById(R.id.editNumero);
        editMensaje = (EditText)findViewById(R.id.editMensaje);
        botonEnviar =  (Button)findViewById(R.id.botonEnviar);

        botonEnviar.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {


                envioSms();
            }
```

Para terminar con el código de esta actividad, creamos el método que llamamos envioSms, en el cual creamos un objeto SmsManager llamado manager y con el método sendTextMessage de este objeto enviaremos nuestro mensaje. Finalmente si el mensaje ha sido enviado lo pondremos en el Toast

```
            @Override
            public void onClick(View v) {


                envioSms();
            }


            private void envioSms() {

                String numero = editNumero.getText().toString();
                String mensaje = editMensaje.getText().toString();

                SmsManager manager = SmsManager.getDefault();
                manager.sendTextMessage(numero, null, mensaje,null, null);
                Toast.makeText(getApplicationContext(), "Mensaje Enviado", Toast.LENGTH_LONG)
            }
        });
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
```

Ya tenemos nuestra parte del envío de SMS, ahora veremos la parte para recibir mensajes, para esto creamos una nueva clase a la cual llamaremos Receptor.java y la cual no tendrá layout asociado pero tendrá que extenderse de la clase BroadcastReceiver la cual habilita a nuestra aplicación a recibir *intents* de otras aplicaciones usando el método sendBroadcast(). Y para poder manejar estos *intents* debemos crear un manejador *Bundle* que nos facilita la tarea ya que los mensajes SMS son enviados en un formato llamado PDU (por sus siglas en ingles *protocol data unit*) y necesitamos una método especial llamado createFromPdu() que usaremos intrínsecamente con el bundle.

Necesitaremos importar algunas librerias que hagan posible la recepción de los SMS dentro de los que son mas destacados son, android.content.BroadcastReceiver para escuchar los SMS y android.os.Bundle para el manejo de los SMS, quedando de la siguiente manera

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Message;
import android.telephony.SmsMessage;
import android.widget.Toast;

import com.example.laptm.sms.R;

import java.util.Objects;

/**
 * Created by LA PTM on 21/10/2015.
 */
public class Receptor extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        SmsMessage [] mensajes = null;
        String string = "";
```

Para recibir los SMS necesitaremos un solo método llamado onReceive() al cual se le pasan 2 parámetros, uno es el contexto y el otro es un intent; dentro de este método creamos el objeto bundle pasándole los extras que necesitamos, asi como otro objeto SmsMessage llamado mensajes el cual estará null o vacio y es donde se almacenara cada SMS que llegue a nuestro dispositivo y por ultimo también crearemos un String llamado string que usaremos en el Toast para mostrar nuestro mensaje en formato de texto

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Message;
import android.telephony.SmsMessage;
import android.widget.Toast;

import com.example.laptm.sms.R;

import java.util.Objects;

/**
 * Created by LA PTM on 21/10/2015.
 */
public class Receptor extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Bundle bundle = intent.getExtras();
        SmsMessage [] mensajes = null;
        String string = "";
```

Ahora vamos a escribir el código que se activa cuando ha llegado un mensaje SMS, empezando por evaluar si el bundle no está vacío, creamos un objeto pdus ,e instanciamos nuestro objecto mensajes con el largo total del SMS que nos ha llegado

```
if (bundle != null){
    Object [] pdus = (Objects []) bundle.get("pdus");
    mensajes = new SmsMessage[pdus.length];
```

Ahora creamos un for que va evaluando el largo del SMS y va creando primero la dirección de quien ha enviado el mensaje, con el texto previo “SMS de:”, después crea el cuerpo del mensaje que ha llegado, Y para terminar con el código de esta clase, creamos un Toast que nos mostrara el String string cada vez que nos llegue un mensaje

```
public class Receptor extends BroadcastReceiver {;

        for (int i = 0; i<mensajes.length ; i++){

            mensajes [i] = SmsMessage.createFromPdu((byte[])pdus[i]);
            string +="Mensaje de" + mensajes[i].getOriginatingAddress();
            string +=":";
            string +=mensajes[i].getMessageBody().toString();
            string +="/n";
        }
        Toast.makeText(context, string, Toast.LENGTH_LONG).show();

    }


}
}
```

Para que la aplicación funcione correctamente necesitamos indicar en el manifiesto de nuestra aplicación que necesitamos el permiso para enviar y recibir SMS asi como declarar que nuestra actividad Receptor
.java es del tipo receiver, lo que indica que no debe estar corriendo en nuestra pantalla para que se encuentre activa y realice su tarea correspondiente

```
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />

<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="Sms"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MainActivity"
        android:label="Sms" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    <receiver android:name=".Receptor">
        <intent-filter>
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
    </receiver>
    </activity>
    <activity
```