1. **Historial de cambios**

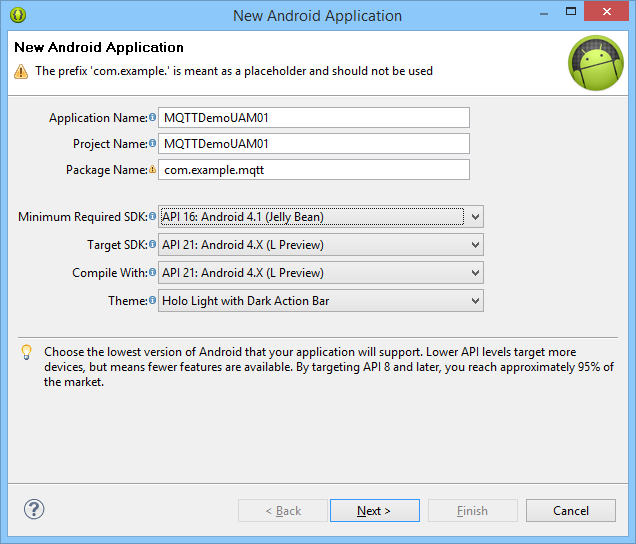
| **Fecha** | **Versión** | **Autor** | **Descripción del Cambio** |
| --- | --- | --- | --- |
| 06/05/2015 | 1.0 | Julián Andrés Henao Taborda | Creación del documento |

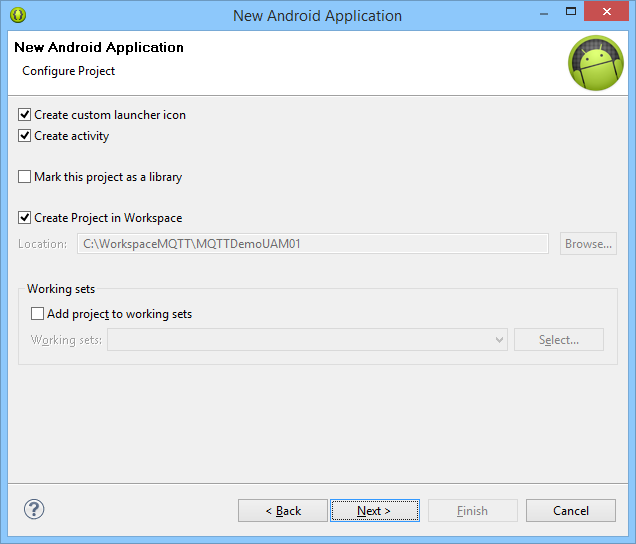
1. **Introducción**

El objetivo del tutorial es el desarrollo de una aplicación cliente para sistema operativo Android para hacer uso de Message Queue Telemetry Transport*.*

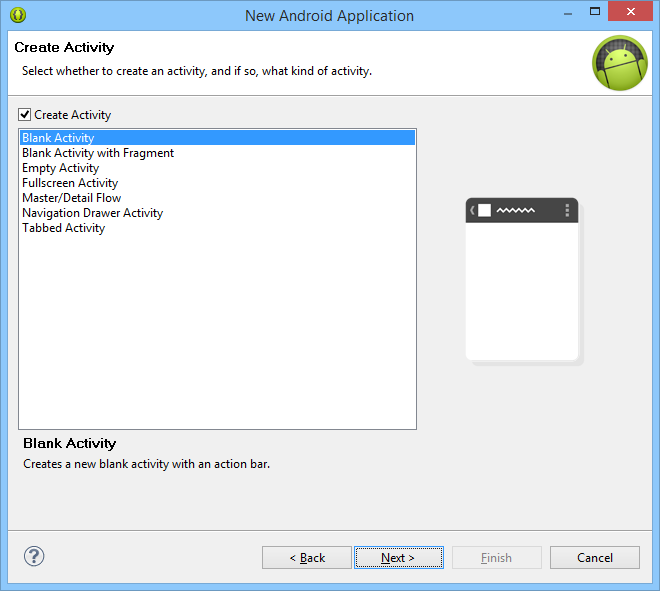
Después de seguir este tutorial se obtendrá una aplicación que una vez instalada en un dispositivo podrá enviar/recibir mensajes de suscripción.

1. Descargar Eclipse SDK.
2. En Eclipse crear un nuevo proyecto tipo -> Android Application Project.

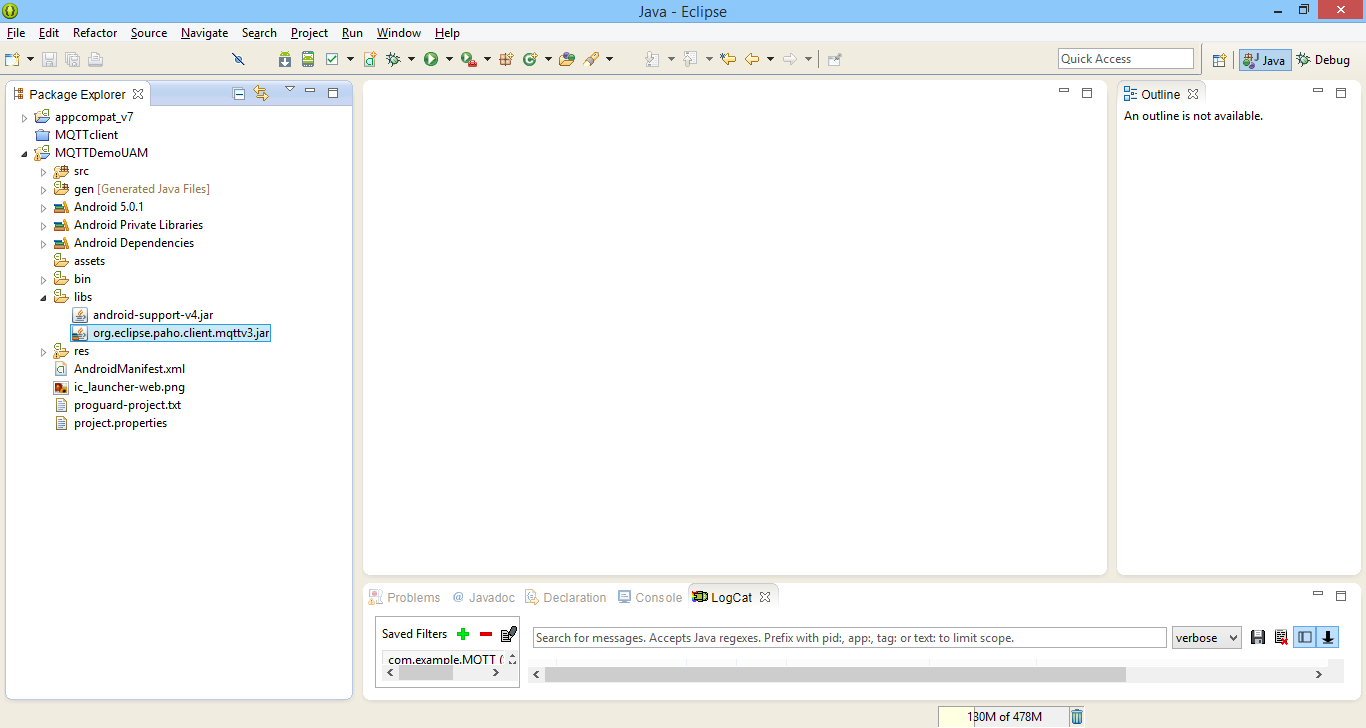




1. Crear “Blank Activity”



1. Copiar el archivo “org.eclipse.paho.client.mqttv3.jar” y pegarlo en la carpeta “libs” del proyecto recientemente creado.



1. Reemplazar el contenido de la clase creada por el siguiente código

**En esta clase se programan los eventos de suscripción y publicación, utilizados para el envío de los mensajes.**

**package** com.example.MQTT;

**import** android.support.v7.app.ActionBarActivity;

**import** android.app.Activity;

**import** android.content.BroadcastReceiver;

**import** android.content.ComponentName;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.content.IntentFilter;

**import** android.content.ServiceConnection;

**import** android.os.Bundle;

**import** android.os.Handler;

**import** android.os.IBinder;

**import** android.os.Message;

**import** android.os.Messenger;

**import** android.os.RemoteException;

**import** android.view.Menu;

**import** android.view.MenuItem;

**import** android.view.View;

**import** android.view.View.OnClickListener;

**import** android.view.inputmethod.InputMethodManager;

**import** android.widget.Button;

**import** android.widget.EditText;

**import** android.widget.TextView;

**import** android.widget.Toast;

**public** **class** MainActivity **extends** Activity {

**private** Messenger service = **null**;

**private** **final** Messenger serviceHandler = **new** Messenger(**new** ServiceHandler());

**private** IntentFilter intentFilter = **null**;

**private** PushReceiver pushReceiver;

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

intentFilter = **new** IntentFilter();

intentFilter.addAction("com.example.MQTT.PushReceived");

pushReceiver = **new** PushReceiver();

registerReceiver(pushReceiver, intentFilter, **null**, **null**);

startService(**new** Intent(**this**, MQTTservice.**class**));

addSubscribeButtonListener();

addPublishButtonListener();

}

@Override

**protected** **void** onStart()

{

**super**.onStart();

bindService(**new** Intent(**this**, MQTTservice.**class**), serviceConnection, 0);

}

@Override

**protected** **void** onStop()

{

**super**.onStop();

unbindService(serviceConnection);

}

@Override

**protected** **void** onResume()

{

**super**.onResume();

registerReceiver(pushReceiver, intentFilter);

}

@Override

**protected** **void** onPause()

{

**super**.onPause();

unregisterReceiver(pushReceiver);

}

**public** **class** PushReceiver **extends** BroadcastReceiver

{

@Override

**public** **void** onReceive(Context context, Intent i)

{

String topic = i.getStringExtra(MQTTservice.*TOPIC*);

String message = i.getStringExtra(MQTTservice.*MESSAGE*);

Toast.*makeText*(context, "Mensaje Push recibido - " + topic + ":" + message, Toast.*LENGTH\_LONG*).show();

}

}

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

// Inflate the menu; this adds items to the action bar if it is present.

getMenuInflater().inflate(R.menu.*main*, menu);

**return** **true**;

}

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

// Handle action bar item clicks here. The action bar will

// automatically handle clicks on the Home/Up button, so long

// as you specify a parent activity in AndroidManifest.xml.

**int** id = item.getItemId();

**if** (id == R.id.*action\_settings*) {

**return** **true**;

}

**return** **super**.onOptionsItemSelected(item);

}

**private** ServiceConnection serviceConnection = **new** ServiceConnection()

{

@Override

**public** **void** onServiceConnected(ComponentName arg0, IBinder binder)

{

service = **new** Messenger(binder);

Bundle data = **new** Bundle();

//data.putSerializable(MQTTservice.CLASSNAME, MainActivity.class);

data.putCharSequence(MQTTservice.*INTENTNAME*, "com.example.MQTT.PushReceived");

Message msg = Message.*obtain*(**null**, MQTTservice.*REGISTER*);

msg.setData(data);

msg.replyTo = serviceHandler;

**try**

{

service.send(msg);

}

**catch** (RemoteException e)

{

e.printStackTrace();

}

}

@Override

**public** **void** onServiceDisconnected(ComponentName arg0)

{

}

};

**private** **void** addSubscribeButtonListener()

{

Button subscribeButton = (Button) findViewById(R.id.*buttonSubscribe*);

subscribeButton.setOnClickListener(**new** OnClickListener()

{

InputMethodManager inputMethodManager = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);

@Override

**public** **void** onClick(View arg0)

{

TextView result = (TextView) findViewById(R.id.*textResultStatus*);

EditText t = (EditText) findViewById(R.id.*EditTextTopic*);

String topic = t.getText().toString().trim();

inputMethodManager.hideSoftInputFromWindow(result.getWindowToken(), 0);

**if** (topic != **null** && topic.isEmpty() == **false**)

{

result.setText("");

Bundle data = **new** Bundle();

data.putCharSequence(MQTTservice.*TOPIC*, topic);

Message msg = Message.*obtain*(**null**, MQTTservice.*SUBSCRIBE*);

msg.setData(data);

msg.replyTo = serviceHandler;

**try**

{

service.send(msg);

}

**catch** (RemoteException e)

{

e.printStackTrace();

result.setText("Excepción al suscribirse: " + e.getMessage());

}

**catch** (Exception e)

{

e.printStackTrace();

result.setText("Excepción al suscribirse: " + e.getMessage());

}

}

**else**

{

result.setText("Tema requerido.");

}

}

});

}

**private** **void** addPublishButtonListener()

{

Button publishButton = (Button) findViewById(R.id.*buttonPublish*);

publishButton.setOnClickListener(**new** OnClickListener()

{

InputMethodManager inputMethodManager = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);

@Override

**public** **void** onClick(View arg0)

{

EditText t = (EditText) findViewById(R.id.*EditTextTopic*);

EditText m = (EditText) findViewById(R.id.*editTextMessage*);

TextView result = (TextView) findViewById(R.id.*textResultStatus*);

inputMethodManager.hideSoftInputFromWindow(result.getWindowToken(), 0);

String topic = t.getText().toString().trim();

String message = m.getText().toString().trim();

**if** (topic != **null** && topic.isEmpty() == **false** && message != **null** && message.isEmpty() == **false**)

{

result.setText("");

Bundle data = **new** Bundle();

data.putCharSequence(MQTTservice.*TOPIC*, topic);

data.putCharSequence(MQTTservice.*MESSAGE*, message);

Message msg = Message.*obtain*(**null**, MQTTservice.*PUBLISH*);

msg.setData(data);

msg.replyTo = serviceHandler;

**try**

{

service.send(msg);

}

**catch** (RemoteException e)

{

e.printStackTrace();

result.setText("Excepción al publicar: " + e.getMessage());

}

**catch** (Exception e)

{

e.printStackTrace();

result.setText("Excepción al publicar: " + e.getMessage());

}

}

**else**

{

result.setText("Tema y mensaje requerido.");

}

}

});

}

**class** ServiceHandler **extends** Handler

{

@Override

**public** **void** handleMessage(Message msg)

{

**switch** (msg.what)

{

**case** MQTTservice.*SUBSCRIBE*: **break**;

**case** MQTTservice.*PUBLISH*: **break**;

**case** MQTTservice.*REGISTER*: **break**;

**default**:

**super**.handleMessage(msg);

**return**;

}

Bundle b = msg.getData();

**if** (b != **null**)

{

TextView result = (TextView) findViewById(R.id.*textResultStatus*);

Boolean status = b.getBoolean(MQTTservice.*STATUS*);

**if** (status == **false**)

{

result.setText("Error");

}

**else**

{

result.setText("Éxito");

}

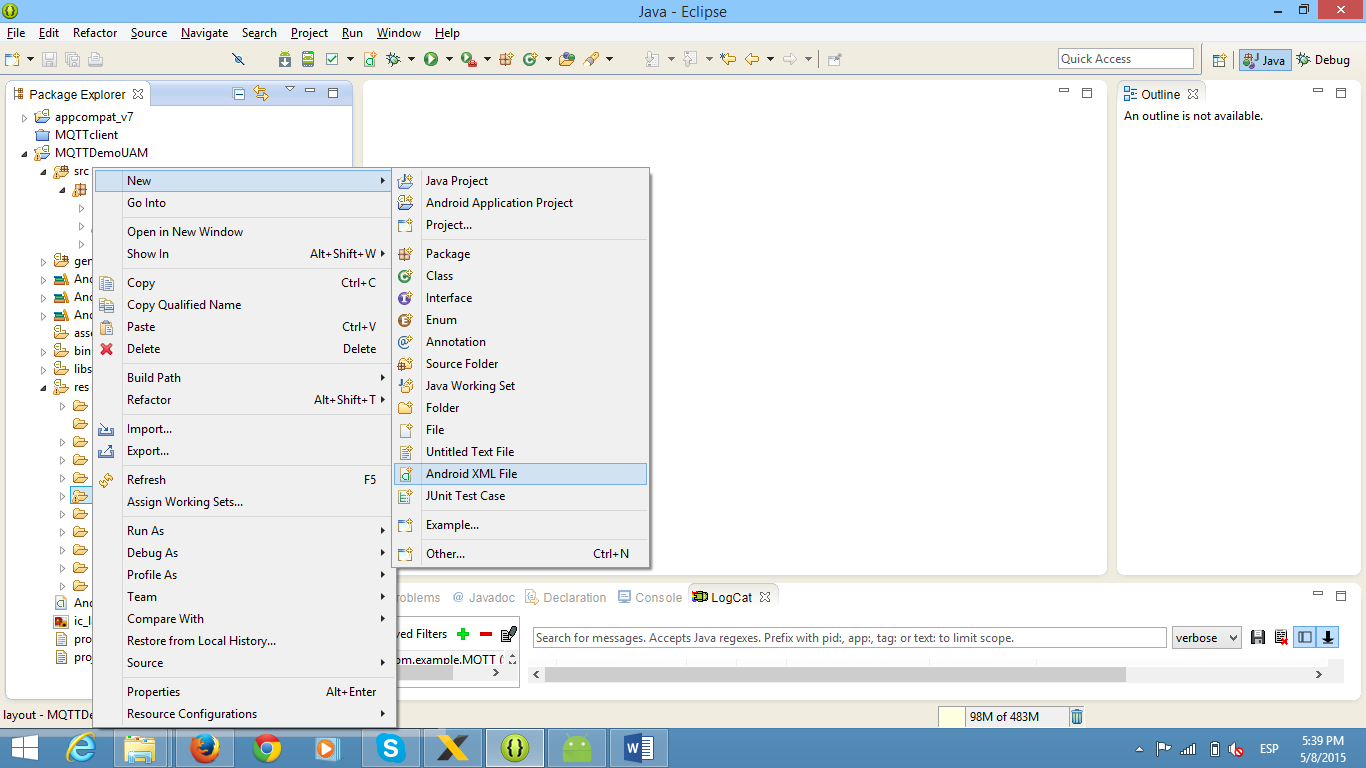
}

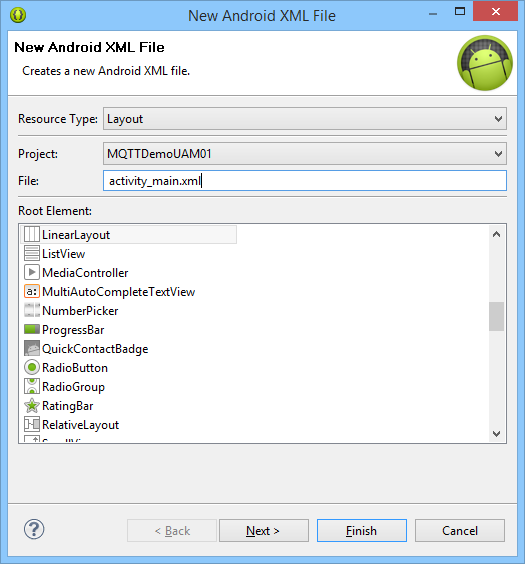
}

}

}

1. Proceder a crear la plantilla gráfica para la clase. Creando un “Android XML File” en el directorio de layouts:





1. Reemplazar el contenido del XML creado por el código:

**En el xml se definen los controles android de la interfaz gráfica.**

<RelativeLayout xmlns:android=*"http://schemas.android.com/apk/res/android"*

xmlns:tools=*"http://schemas.android.com/tools"*

android:id=*"@+id/textStatus"*

android:layout\_width=*"match\_parent"*

android:layout\_height=*"match\_parent"*

android:paddingBottom=*"@dimen/activity\_vertical\_margin"*

android:paddingLeft=*"@dimen/activity\_horizontal\_margin"*

android:paddingRight=*"@dimen/activity\_horizontal\_margin"*

android:paddingTop=*"@dimen/activity\_vertical\_margin"*

tools:context=*"com.example.MQTT.MainActivity"* >

<EditText

android:id=*"@+id/editTextMessage"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_below=*"@+id/buttonSubscribe"*

android:layout\_marginTop=*"18dp"*

android:ems=*"10"*

android:hint=*"Mensaje"*

android:width=*"175dp"* />

<Button

android:id=*"@+id/buttonPublish"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBottom=*"@+id/editTextMessage"*

android:layout\_alignLeft=*"@+id/buttonSubscribe"*

android:layout\_alignParentRight=*"true"*

android:text=*"Publicar"* />

<Button

android:id=*"@+id/buttonSubscribe"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBottom=*"@+id/EditTextTopic"*

android:layout\_alignParentRight=*"true"*

android:text=*"Suscribirse"* />

<EditText

android:id=*"@+id/EditTextTopic"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/editTextMessage"*

android:layout\_alignParentTop=*"true"*

android:layout\_marginTop=*"41dp"*

android:ems=*"10"*

android:hint=*"Tema"*

android:width=*"175dp"* >

<requestFocus />

</EditText>

<TextView

android:id=*"@+id/textView1"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignLeft=*"@+id/editTextMessage"*

android:layout\_below=*"@+id/editTextMessage"*

android:layout\_marginTop=*"39dp"*

android:text=*"Estado"* />

<TextView

android:id=*"@+id/textResultStatus"*

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

android:layout\_alignBaseline=*"@+id/textView1"*

android:layout\_alignBottom=*"@+id/textView1"*

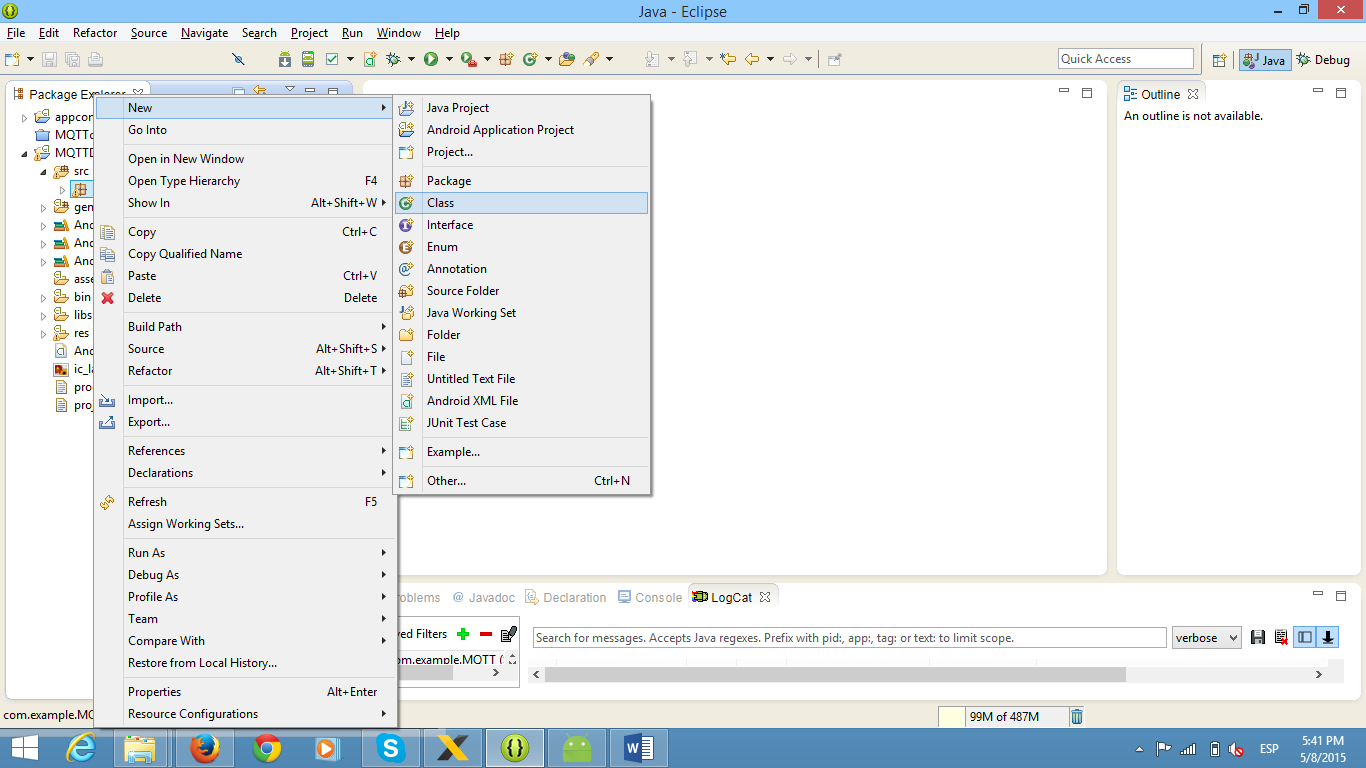
android:layout\_alignRight=*"@+id/buttonPublish"*

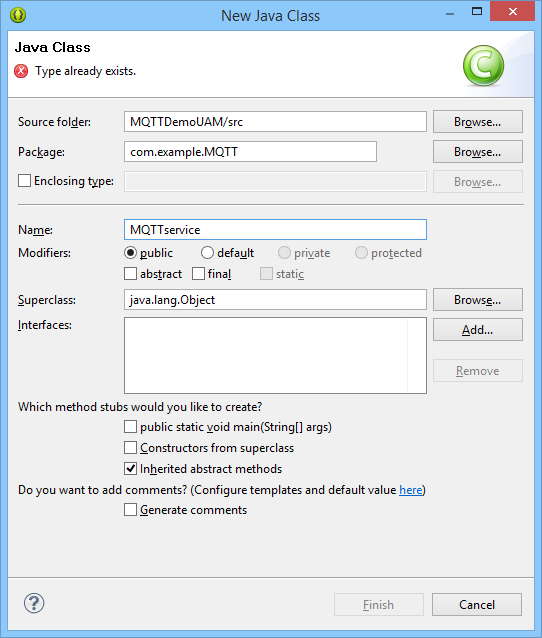
android:layout\_marginLeft=*"36dp"*

android:layout\_toRightOf=*"@+id/textView1"* />

</RelativeLayout>

1. Crear una nueva clase en el proyecto:





1. Reemplazar el contenido de la clase creada por el código:

**En esta clase se definen los servicios de suscripción, publicación desde el servicio MQTT.**

**package** com.example.MQTT;

**import** java.util.Iterator;

**import** java.util.Vector;

**import** org.eclipse.paho.client.mqttv3.IMqttDeliveryToken;

**import** org.eclipse.paho.client.mqttv3.MqttCallback;

**import** org.eclipse.paho.client.mqttv3.MqttClient;

**import** org.eclipse.paho.client.mqttv3.MqttConnectOptions;

**import** org.eclipse.paho.client.mqttv3.MqttException;

**import** org.eclipse.paho.client.mqttv3.MqttMessage;

**import** android.app.Notification;

**import** android.app.NotificationManager;

**import** android.app.PendingIntent;

**import** android.app.Service;

**import** android.app.Notification.Builder;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.os.Handler;

**import** android.os.IBinder;

**import** android.os.Message;

**import** android.os.Messenger;

**import** android.os.RemoteException;

**import** android.util.Log;

**public** **class** MQTTservice **extends** Service {

**private** **static** **boolean** *serviceRunning* = **false**;

**private** **static** **int** *mid* = 0;

**private** **static** MQTTConnection *connection* = **null**;

**private** **final** Messenger clientMessenger = **new** Messenger(**new** ClientHandler());

@Override

**public** **void** onCreate()

{

**super**.onCreate();

*connection* = **new** MQTTConnection();

}

@Override

**public** **int** onStartCommand(Intent intent, **int** flags, **int** startId)

{

**if** (*isRunning*())

{

**return** *START\_STICKY*;

}

**super**.onStartCommand(intent, flags, startId);

/\*

\* Start the MQTT Thread.

\*/

*connection*.start();

**return** *START\_STICKY*;

}

@Override

**public** **void** onDestroy()

{

*connection*.end();

}

@Override

**public** IBinder onBind(Intent intent)

{

/\*

\* Return a reference to our client handler.

\*/

**return** clientMessenger.getBinder();

}

**private** **synchronized** **static** **boolean** isRunning()

{

/\*

\* Only run one instance of the service.

\*/

**if** (*serviceRunning* == **false**)

{

*serviceRunning* = **true**;

**return** **false**;

}

**else**

{

**return** **true**;

}

}

/\*

\* These are the supported messages from bound clients

\*/

**public** **static** **final** **int** *REGISTER* = 0;

**public** **static** **final** **int** *SUBSCRIBE* = 1;

**public** **static** **final** **int** *PUBLISH* = 2;

/\*

\* Fixed strings for the supported messages.

\*/

**public** **static** **final** String *TOPIC* = "topic";

**public** **static** **final** String *MESSAGE* = "message";

**public** **static** **final** String *STATUS* = "status";

**public** **static** **final** String *CLASSNAME* = "classname";

**public** **static** **final** String *INTENTNAME* = "intentname";

/\*

\* This class handles messages sent to the service by

\* bound clients.

\*/

**class** ClientHandler **extends** Handler

{

@Override

**public** **void** handleMessage(Message msg)

{

**boolean** status = **false**;

**switch** (msg.what)

{

**case** *SUBSCRIBE*:

**case** *PUBLISH*:

/\*

\* These two requests should be handled by

\* the connection thread, call makeRequest

\*/

*connection*.makeRequest(msg);

**break**;

**case** *REGISTER*:

{

Bundle b = msg.getData();

**if** (b != **null**)

{

Object target = b.getSerializable(*CLASSNAME*);

**if** (target != **null**)

{

/\*

\* This request can be handled in-line

\* call the API

\*/

*connection*.setPushCallback((Class<?>) target);

status = **true**;

}

CharSequence cs = b.getCharSequence(*INTENTNAME*);

**if** (cs != **null**)

{

String name = cs.toString().trim();

**if** (name.isEmpty() == **false**)

{

/\*

\* This request can be handled in-line

\* call the API

\*/

*connection*.setIntentName(name);

status = **true**;

}

}

}

ReplytoClient(msg.replyTo, msg.what, status);

**break**;

}

}

}

}

**private** **void** ReplytoClient(Messenger responseMessenger, **int** type, **boolean** status)

{

/\*

\* A response can be sent back to a requester when

\* the replyTo field is set in a Message, passed to this

\* method as the first parameter.

\*/

**if** (responseMessenger != **null**)

{

Bundle data = **new** Bundle();

data.putBoolean(*STATUS*, status);

Message reply = Message.*obtain*(**null**, type);

reply.setData(data);

**try** {

responseMessenger.send(reply);

} **catch** (RemoteException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**enum** CONNECT\_STATE

{

*DISCONNECTED*,

*CONNECTING*,

*CONNECTED*

}

**private** **class** MQTTConnection **extends** Thread

{

**private** Class<?> launchActivity = **null**;

**private** String intentName = **null**;

**private** MsgHandler msgHandler = **null**;

**private** **static** **final** **int** *STOP* = *PUBLISH* + 1;

**private** **static** **final** **int** *CONNECT* = *PUBLISH* + 2;

**private** **static** **final** **int** *RESETTIMER* = *PUBLISH* + 3;

**private** CONNECT\_STATE connState = CONNECT\_STATE.*DISCONNECTED*;

MQTTConnection()

{

msgHandler = **new** MsgHandler();

msgHandler.sendMessage(Message.*obtain*(**null**, *CONNECT*));

}

**public** **void** end()

{

msgHandler.sendMessage(Message.*obtain*(**null**, *STOP*));

}

**public** **void** makeRequest(Message msg)

{

/\*

\* It is expected that the caller only invokes

\* this method with valid msg.what.

\*/

msgHandler.sendMessage(Message.*obtain*(msg));

}

**public** **void** setPushCallback(Class<?> activityClass)

{

launchActivity = activityClass;

}

**public** **void** setIntentName(String name)

{

intentName = name;

}

**private** **class** MsgHandler **extends** Handler **implements** MqttCallback

{

**private** **final** String HOST = "iot.eclipse.org";

**private** **final** **int** PORT = 1883;

**private** **final** String uri = "tcp://" + HOST + ":" + PORT;

**private** **final** **int** MINTIMEOUT = 2000;

**private** **final** **int** MAXTIMEOUT = 32000;

**private** **int** timeout = MINTIMEOUT;

**private** MqttClient client = **null**;

**private** MqttConnectOptions options = **new** MqttConnectOptions();

**private** Vector<String> topics = **new** Vector<String>();

MsgHandler()

{

options.setCleanSession(**true**);

**try**

{

client = **new** MqttClient(uri, MqttClient.*generateClientId*(), **null**);

client.setCallback(**this**);

}

**catch** (MqttException e1)

{

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

@Override

**public** **void** handleMessage(Message msg)

{

**switch** (msg.what)

{

**case** *STOP*:

{

/\*

\* Clean up, and terminate.

\*/

client.setCallback(**null**);

**if** (client.isConnected())

{

**try** {

client.disconnect();

client.close();

} **catch** (MqttException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

getLooper().quit();

**break**;

}

**case** *CONNECT*:

{

**if** (connState != CONNECT\_STATE.*CONNECTED*)

{

**try**

{

client.connect(options);

connState = CONNECT\_STATE.*CONNECTED*;

Log.*d*(getClass().getCanonicalName(), "Connected");

timeout = MINTIMEOUT;

}

**catch** (MqttException e)

{

Log.*d*(getClass().getCanonicalName(), "Connection attemp failed with reason code = " + e.getReasonCode() + e.getCause());

**if** (timeout < MAXTIMEOUT)

{

timeout \*= 2;

}

**this**.sendMessageDelayed(Message.*obtain*(**null**, *CONNECT*), timeout);

**return**;

}

/\*

\* Re-subscribe to previously subscribed topics

\*/

Iterator<String> i = topics.iterator();

**while** (i.hasNext())

{

subscribe(i.next());

}

}

**break**;

}

**case** *RESETTIMER*:

{

timeout = MINTIMEOUT;

**break**;

}

**case** *SUBSCRIBE*:

{

**boolean** status = **false**;

Bundle b = msg.getData();

**if** (b != **null**)

{

CharSequence cs = b.getCharSequence(*TOPIC*);

**if** (cs != **null**)

{

String topic = cs.toString().trim();

**if** (topic.isEmpty() == **false**)

{

status = subscribe(topic);

/\*

\* Save this topic for re-subscription if needed.

\*/

**if** (status)

{

topics.add(topic);

}

}

}

}

ReplytoClient(msg.replyTo, msg.what, status);

**break**;

}

**case** *PUBLISH*:

{

**boolean** status = **false**;

Bundle b = msg.getData();

**if** (b != **null**)

{

CharSequence cs = b.getCharSequence(*TOPIC*);

**if** (cs != **null**)

{

String topic = cs.toString().trim();

**if** (topic.isEmpty() == **false**)

{

cs = b.getCharSequence(*MESSAGE*);

**if** (cs != **null**)

{

String message = cs.toString().trim();

**if** (message.isEmpty() == **false**)

{

status = publish(topic, message);

}

}

}

}

}

ReplytoClient(msg.replyTo, msg.what, status);

**break**;

}

}

}

**private** **boolean** subscribe(String topic)

{

**try**

{

client.subscribe(topic);

}

**catch** (MqttException e)

{

Log.*d*(getClass().getCanonicalName(), "Subscribe failed with reason code = " + e.getReasonCode());

**return** **false**;

}

**return** **true**;

}

**private** **boolean** publish(String topic, String msg)

{

**try**

{

MqttMessage message = **new** MqttMessage();

message.setPayload(msg.getBytes());

client.publish(topic, message);

}

**catch** (MqttException e)

{

Log.*d*(getClass().getCanonicalName(), "Publish failed with reason code = " + e.getReasonCode());

**return** **false**;

}

**return** **true**;

}

@Override

**public** **void** connectionLost(Throwable arg0)

{

Log.*d*(getClass().getCanonicalName(), "connectionLost");

connState = CONNECT\_STATE.*DISCONNECTED*;

sendMessageDelayed(Message.*obtain*(**null**, *CONNECT*), timeout);

}

@Override

**public** **void** deliveryComplete(IMqttDeliveryToken arg0)

{

}

@Override

**public** **void** messageArrived(String topic, MqttMessage message) **throws** Exception

{

Log.*d*(getClass().getCanonicalName(), topic + ":" + message.toString());

**if** (intentName != **null**)

{

Intent intent = **new** Intent();

intent.setAction(intentName);

intent.putExtra(*TOPIC*, topic);

intent.putExtra(*MESSAGE*, message.toString());

sendBroadcast(intent);

}

Context context = getBaseContext();

PendingIntent pendingIntent = **null**;

**if** (launchActivity != **null**)

{

Intent intent = **new** Intent(context, launchActivity);

intent.setAction(Intent.*ACTION\_MAIN*);

intent.addCategory(Intent.*CATEGORY\_LAUNCHER*);

//build the pending intent that will start the appropriate activity

pendingIntent = PendingIntent.*getActivity*(context, 0, intent, 0);

}

//build the notification

Builder notificationCompat = **new** Builder(context);

notificationCompat.setAutoCancel(**true**)

.setContentIntent(pendingIntent)

.setContentText( message.toString())

.setSmallIcon(R.drawable.*ic\_launcher*);

Notification notification = notificationCompat.build();

NotificationManager nm = (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);

nm.notify(*mid*++, notification);

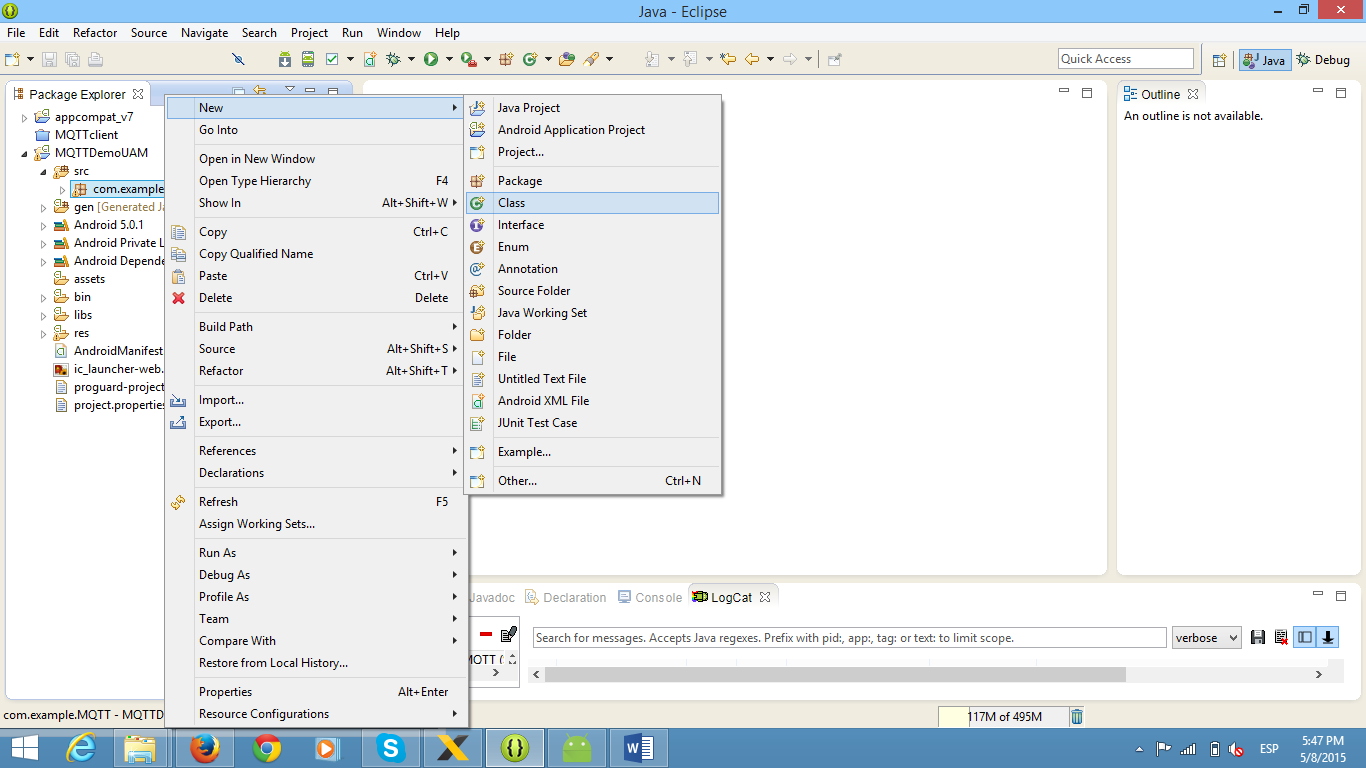
}

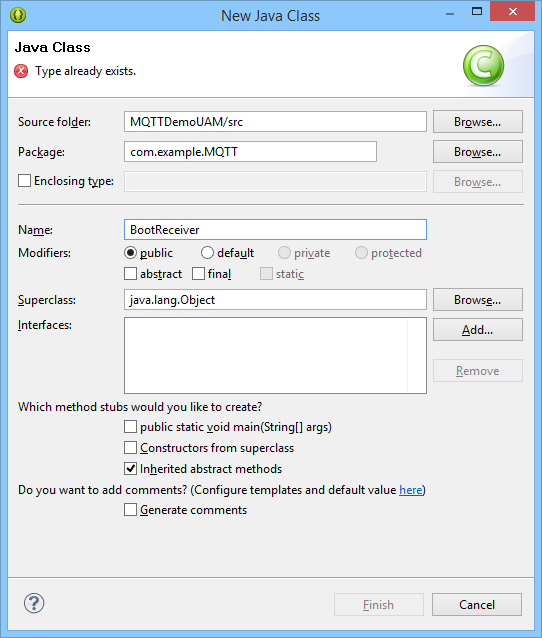
}

}

}

1. Crear una nueva clase en el proyecto:





1. Reemplazar el contenido de la clase creada por el código:

**En esta clase se definen la recepción de mensajes desde el servicio MQTT.**

**package** com.example.MQTT;

**import** android.content.BroadcastReceiver;

**import** android.content.Context;

**import** android.content.Intent;

**import** android.util.Log;

**public** **class** BootReceiver **extends** BroadcastReceiver

{

@Override

**public** **void** onReceive(Context context, Intent intent)

{

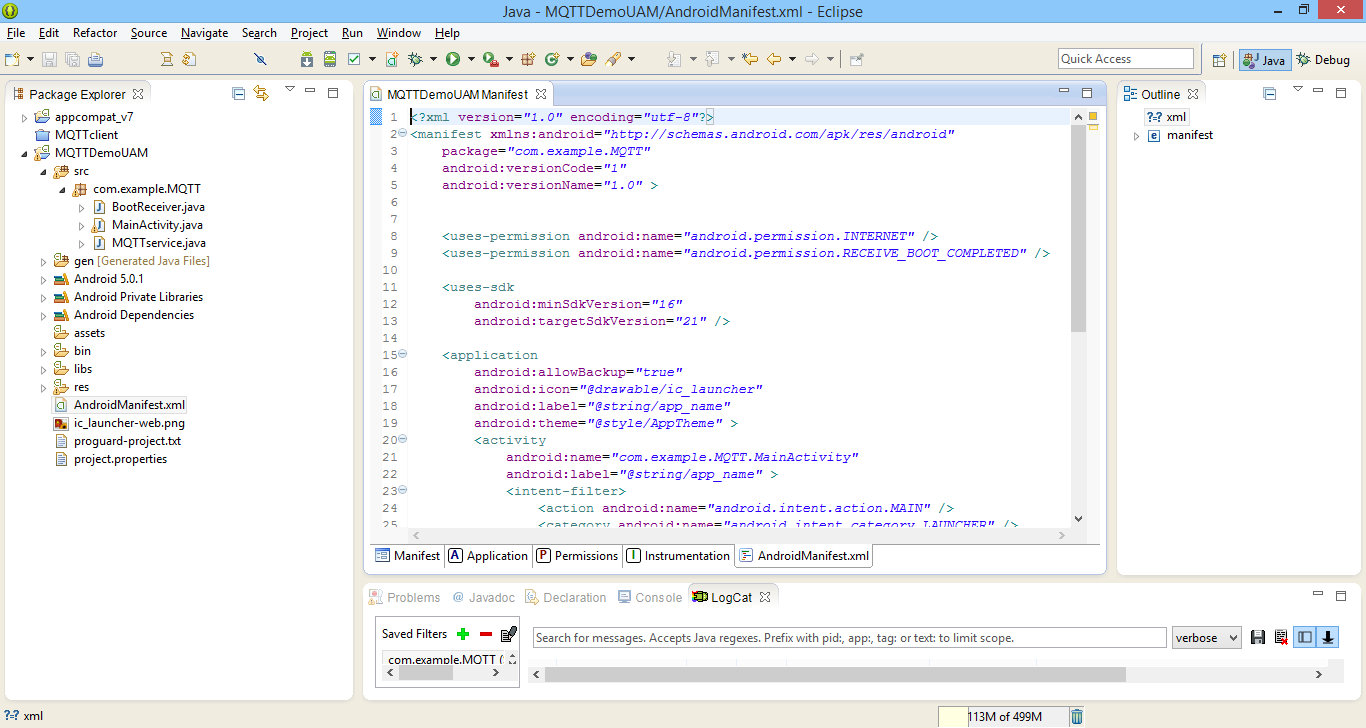
Log.*d*(getClass().getCanonicalName(), "onReceive");

context.startService(**new** Intent(context, MQTTservice.**class**));

}

}

1. Procedemo a modificar el archivo “**AndroidManifest.xml**”.



1. Reemplazamos el contenido del archivo XML por el código:

<?xml version=*"1.0"* encoding=*"utf-8"*?>

<manifest xmlns:android=*"http://schemas.android.com/apk/res/android"*

package=*"com.example.MQTT"*

android:versionCode=*"1"*

android:versionName=*"1.0"* >

<uses-permission android:name=*"android.permission.INTERNET"* />

<uses-permission android:name=*"android.permission.RECEIVE\_BOOT\_COMPLETED"* />

<uses-sdk

android:minSdkVersion=*"16"*

android:targetSdkVersion=*"21"* />

<application

android:allowBackup=*"true"*

android:icon=*"@drawable/ic\_launcher"*

android:label=*"@string/app\_name"*

android:theme=*"@style/AppTheme"* >

<activity

android:name=*"com.example.MQTT.MainActivity"*

android:label=*"@string/app\_name"* >

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

<!-- Mqtt Service -->

<service android:name=*"org.eclipse.paho.android.service.MqttService"* >

</service>

<receiver android:name=*"com.example.MQTT.BootReceiver"*>

<intent-filter>

<action android:name=*"android.intent.action.BOOT\_COMPLETED"*/>

</intent-filter>

</receiver>

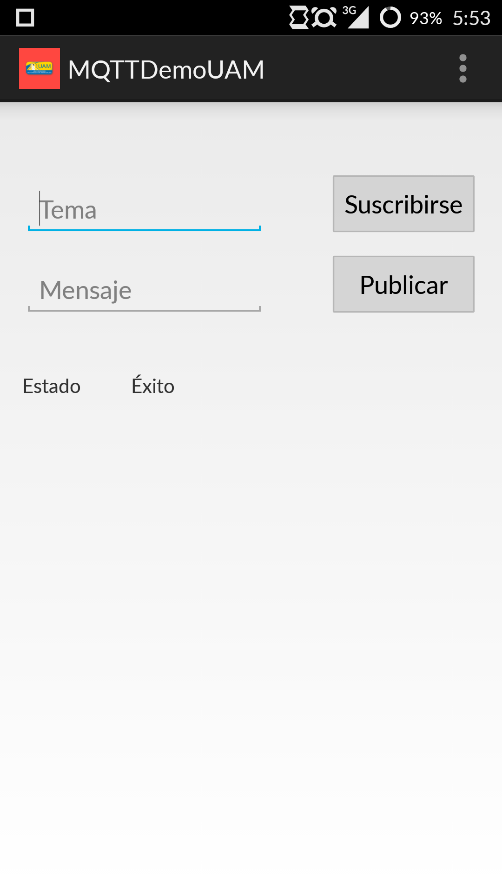
<service android:name=*"com.example.MQTT.MQTTservice"* >

</service>

</application>

</manifest>

1. Instalar la aplicación en el dispositivo.



1. Se debe digitar un tema y seleccionar el botón “Suscribirse”.
2. Una vez realizada la suscripción se puede escribir un mensaje y seleccionar el botón “Publicar”. Una vez publicado un mensaje este será enviado a todos los dispositivos que tengan la aplicación instalada y se encuentren suscritos al tema seleccionado.

