ANDROID

En este apartado destaco los conceptos que diferencian a Android de otros lenguajes de programación conocidos, que más trabajo me han costado comprender a la hora del desarrollo de este proyecto.
Para adquirir el nivel necesario para el desarrollo de las apps, se ha cursado: [UDEMY] y se han consultado las webs: [Learn-android-easily], [Android] y [AndroidCurso].

Un poco de historia

Android fue creado por Android Inc , basándose en el sistema operativo Linux , utilizando una máquina virtual Java Dalvik especializada en dispositivos móviles, con las librerías : C/C++, SQLite y las librerías propias de Android. Android fue comprada por google en el 2005 y se vendió el primer terminal con este sistema operativo en 2007. Actualmente el número de terminales con sistemas operativos Android supera a los de IOS, BrackBerry y Windows Phone juntos. Para saber más ver [https://es.wikipedia.org/wiki/Android]

Estructura de directorios, Gradle

He utilizado Android Studio, en el desarrollo del proyecto, el cual como herramienta de building viene con gradle, lo que nos impone la siguiente estructura de carpetas.

*Tabla **¡Error! No hay texto con el estilo especificado en el documento.**.1: Estructura de directorios Android Studio.*

Directorio	Uso
.idea	Contiene ficheros en formato XML con las opciones de configuración del proyecto para el entorno de desarrollo
app/build	Código fuente compilado de mi aplicación
app/libs	Código fuente de librerías importadas
app/src/main/java	Los directorios creados por nosotros que contienen las clases.java
app/src/main/AndroidManifest.xml	Fichero principal del programa, indica que clases son ejecutables, porque clase comienza la ejecución, que herramientas del sistema operativo usa la aplicación y los servicios en segundo plano que ha de ejecutar.
app/src/main/res/drawable	imagenes
app/src/main/res/values	Ficheros xml con definicion de variables
app/src/main/res/xml	Utilizado para definir en XML, menús
app/src/main/res/layout	Contiene ficheros XML, donde se definen las vistas.
build	Binarios
gradle	Builder tool
External Libraries	Aquí es donde está el SDK de Android y el jdk de java.

B.1 Ciclos de vida

A diferencia de las aplicaciones para escritorio, las aplicaciones para móvil han de estar preparadas para ser interrumpidas en cualquier instante por eventos como: llamada entrante, sms entrante, e-mail entrante, nuevo whatsApp, batería baja o botón de apagado. Para hacer esto posible, cuando se crea una aplicación Android, las clases que controlan la ejecución de la actividad han de heredar de otras clases que le marcan un ciclo de vida. Cada aplicación tiene uno o varios hilos, el sistema operativo decide en tiempo real, a que hilo asignarle recursos, dándole prioridad a los que están visibles en pantalla, en caso de falta de recursos, se empezara a destruir las aplicaciones en estado onStop() y después las que estén en onPause(). Por ello es importante al pasa a estado onPause( ), guardar el valor de las variables de la aplicación, lo cual se realiza utilizando como se indica en:

A continuación se muestran las máquinas de estado de las clases utilizadas en el proyecto.

Ciclo de vida de una actividad.

Nosotros no utilizamos la clase ActionBarActivity que extiende la clase activity permitiéndonos añadir una barra superior a la actividad, con el botón de Settings. Compruebo para esta clase, su ciclo de vida, imprimiendo en la consola un mensaje al entrar en cada uno de los métodos.

1) Abro la aplicación: Se ejecuta por orden los métodos: onCreate(), onStart(), onResumen().
2) Pulso el botón home: onPause(),onStop().
3) Pulso el botón Phone y abro de nuevo la Aplicación: OnRestart(), onStart(), onResumen().
4) Pulso el botón back, se ejecutan: onPause(),onStop y OnDestroy(). Se elimina el hilo de ejecución de la aplicación, perdiéndose las variables temporales.

La documentación Oficial de Android nos indica que el ciclo de vida de las actividades es:

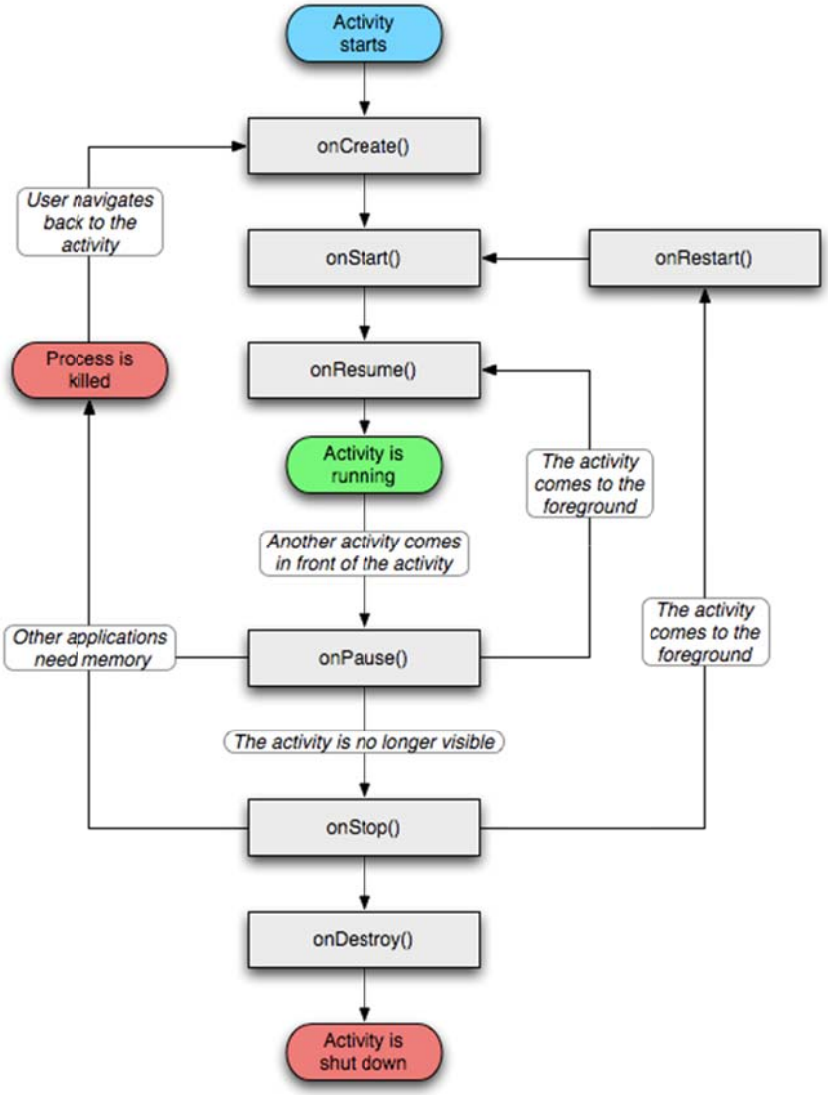

Figura ***¡Error! No hay texto con el estilo especificado en el documento.****.1 Ciclo de vida de una actividad [AndroidCurso].*

Servicios

Los servicios son tareas que se ejecutan en segundo plano, sin interfaz visible con el usuario, aunque suelen ser lanzados desde una tarea con interfaz visible**.** Corren por defecto en la misma hebra de ejecución que la tarea que los lanzo, pero, por defecto, no se destruyen al destruirse esta. Se distinguen entre dos tipos de Servicios: Los que no mantienen comunicación con la tarea que los lanzo, y los que sí.

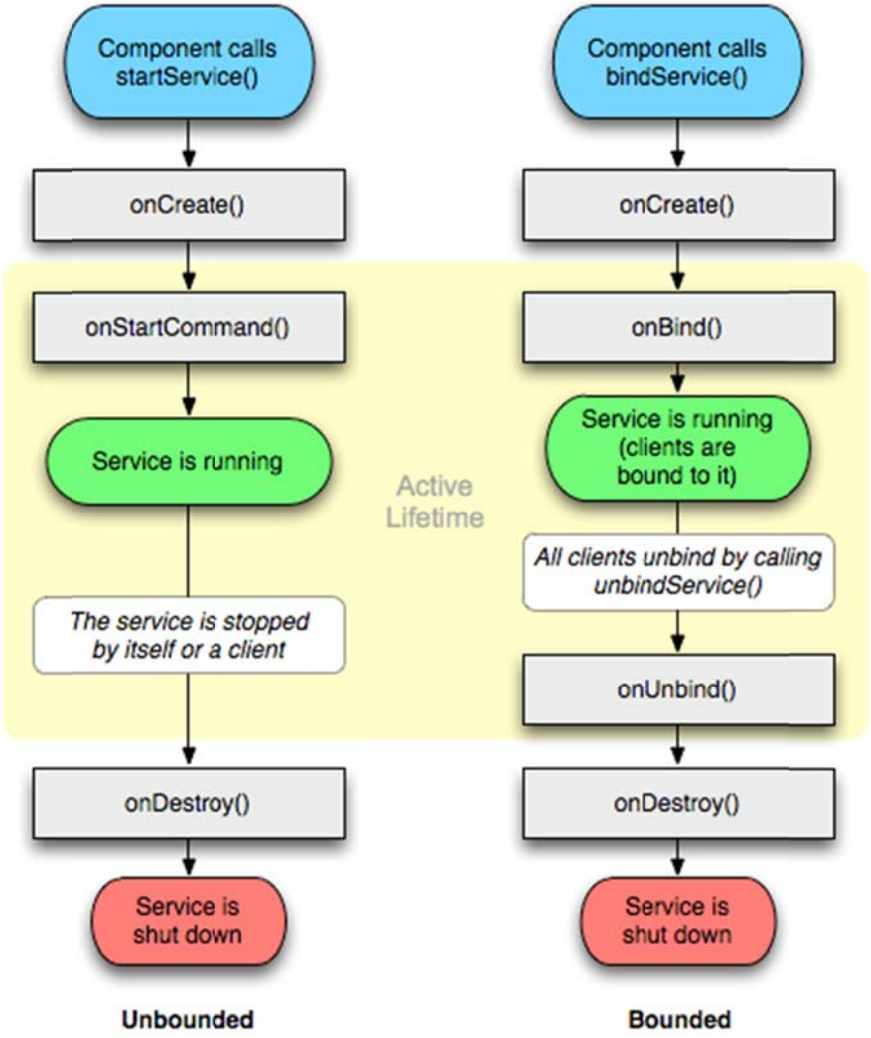

Figura ***¡Error! No hay texto con el estilo especificado en el documento.****..2: Ciclo de vida los servicios [AndroidCurso].*

En la aplicación del Cliente utilizo un servicio de tipo “Bounded”, lanzo el servicio en el OnCreate de la actividad del cliente * y establezco el canal de comunicación en el OnResumen( ) y lo destruyo en el OnPause( ). Inicio el escaneo de Marcas con el botón “Start” y lo paro con el botón “Stop”. El servicio es destruido por el sistema operativo en caso de falta de recursos, en otro caso, se mantiene a la escucha.

*Si el servicio ya está lanzado, no se crea de nuevo, sino que se obtiene un puntero hacia él. Si no está lanzado se relanza.

Intent Service

Es un tipo de servicio que se crea en su propia hebra de ejecución, no interactúa directamente con la hebra de la activity que la creo, además en cada llamada al servicio, comprueba si hay ya una hebra creada y se la hay enlaza la llamada con dicho proceso. Utilizamos este tipo de servicio, para evitar los mensajes ANR(Application not responding). Es el tipo de servicio que hemos utilizado.

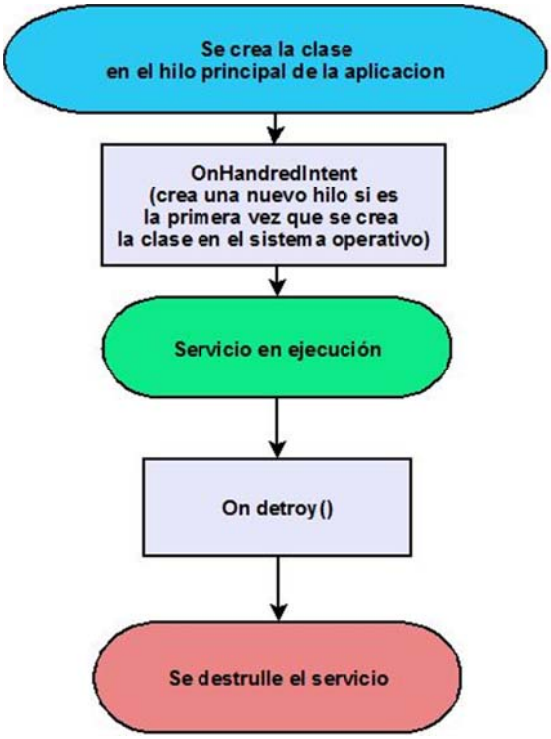

Figura ***¡Error! No hay texto con el estilo especificado en el documento.****.3: Intent Service.* .

Comunicación

Esta es una parte fundamental en cualquier sistema. Android utiliza para ello los siguientes elementos:

- Intens: Solicitudes de que una acción se realice.
- Intent Filters: Para registrar las Clases que ejecutan las acciones solicitadas por los Intent.
- Broadcast Receivers: Escuchan los Intens y deciden como tratarlos.

Ejemplos:

Enviar Información de una Clase a Otra

Pata ello se combina los tres elementos descritos anteriormente. En este proyecto se a utilizado para enviar al servicio encargado de reproducir el mensaje asociado a cada marca, el texto recogido del servidor remoto, para ello:

1. Crear dentro de la clase que implementa el servicio un BroadcastReceiver que tratara la información enviada por el Intent.

```
private BroadcastReceiver mMessageReceiver=new BroadcastReceiver(){

    @Override
    public void onReceive(Context context, Intent intent) {
        toSpeak=intent.getStringExtra("message");
        Log.i("----------INTENT received----------",toSpeak);
        speakTheText(toSpeak);
    }
};
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.4: BroadcastReceiver.*

2. Registrar en la creación de la clase que implementa el servicio el BroadcastReceiver, indicando con un IntentFilter a que intent va a atender y des registrarlo al destruir la clase. Dado que no se va a lanzar el intent al sistema operativo, sino a la propia aplicación utilizo "LocalBroadcastManager" en vez de "BroadcastReceiver".

```
@Override
public void onCreate() {
    LocalBroadcastManager.getInstance(this).registerReceiver(mMessageReceiver, new IntentFilter(Constants.DEVICE_MESSAGE));
    @Override
    public void onDestroy() {
        LocalBroadcastManager.getInstance(this).unregisterReceiver(mMessageReceiver);
        super.onDestroy();
    }
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.5: IntentFilter.*

3. Enviar la información deseada, desde la otra clase. En este caso, cada clase se ejecuta en un hilo diferente, pues estamos lanzando el intent desde la clase HTTP_JSON_POST que extiende de AsyncTask. Para enviar la información utilizo un "sendBroadcastSync" en vez de un "sendBroadcast".

```
Intent intent = new Intent(Constants.DEVICE_MESSAGE);
intent.putExtra("message", specifications_text);
LocalBroadcastManager.getInstance(context).sendBroadcastSync(intent);
Log.i("----------INTENT Was SEND------------",specifications_text);
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.6: Intent.*

<u>Estando en una clase que ofrece un interfaz de Usuario (Activity), pasar a otra.</u>

4. La clase a la que se quiere conmutar ha de estar declarada en el AndrodidManifest.xml.

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="Tracbursys"
    android:theme="@style/AppTheme">
    <activity
        android:name=".Activitys.User_Activity"
        android:label="@string/app_name"></activity>
    <activity
        android:name=".Activitys.Installer_Activity"
        android:label="@string/app_name"></activity>
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.7: AndroidManifest registro Clases.*

5. En el siguiente fragmento de código, según el valor de la variable "workmode", se selecciona la ejecución de la clase Installer o User.

```
if (workMode.equals("0")) {
    startActivity(new Intent(getApplicationContext(), User_Activity.class));
} else {
    startActivity(new Intent(getApplicationContext(), Installer_Activity.class));
}
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.8: Selección de Actividad a ejecutar.*

Salvar el estado del programa.

Para ello he utilizado la clase “PreferenceManager”, que nos permite almacenar valores asociados a la aplicación y recuperarlos, desde cualquier punto de la aplicación.

Ejemplo:

1. Almacenar el valor.

```
//set the value of the last device detected
PreferenceManager.getDefaultSharedPreferences(getApplicationContext())
        .edit()
        .putString(Constants.DEVICE_ADDRESS, address)
        .commit();
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.9: Almacenar valores con PreferenceManager.*

2. Recuperarlo.

```
address = PreferenceManager.getDefaultSharedPreferences(getApplicationContext()).getString(Constants.DEVICE_ADDRESS, "0");
```

*Figura **¡Error! No hay texto con el estilo especificado en el documento.**.10: Recuperar valores, guardados con PreferenceManager.*

REFERENCIAS

[Androidsis] www.androdsis.com

[AndroidCurso] Alejandro Alcalde, http://elbauldelprogramador.com/fundamentos-programacion-android-ciclo/,2015

[UDEMY] Juan Jose Ramirez, https://www.udemy.com/curso-android-online-agbotraining/#/, 2013.

[AndroidSource] https://source.android.com/security/

[Learn-android-easily] Kamlesh Yadav, http://www.learn-android-easily.com/, 2013.

[BLEenAndroid] http://developer.android.com/intl/es/guide/topics/connectivity/bluetooth-le.htm

[AndroidDeveloper] http://developer.android.com/guide/ services.html components/

[Gradle] Hans Dockter, Adam Murdoch, https://docs.gradle.org/current/userguide/userguide, 2007