Desarrollo completo de una aplicación interactiva para dispositivos móviles

1 Introducción

En este documento se explica cómo desarrollar completamente un proyecto con Android Studio ©, que incluye las librerías adecuadas para un componente software de teléfonos móviles que ejecuten Android (versión posterior a la 4.4) como sistema operativo, de acuerdo con el esquema general que muestra la siguiente figura:

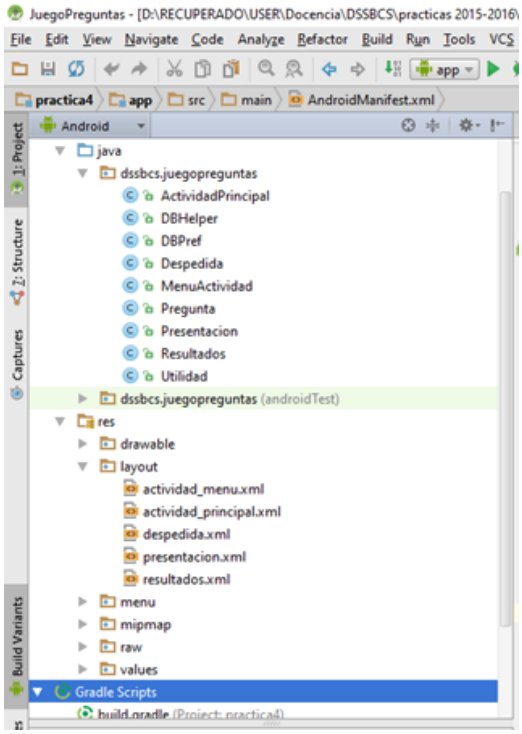

Figura 1: Estructura de carpetas de un proyecto de *Android Studio*

La estructura de un *app* para poder ser ejecutado en una plataforma Android ha de poseer la siguiente estructura de paquetes (o *'folders'* mostrados en la figura 1):

- *manifiestos*: `AndroidManifest.xml` (ver contenido del archivo en la tabla 1)
- *java*: nombre de paquetes y clases–java, que componen la parte controladora de nuestro *app*
- Un prototipo de clase para incluir las pruebas unitarias de validación para nuestra aplicación
- *res*: un paquete que contiene los recursos, expresados como archivos `.xml`, que se utilizarán en el proceso de elaboración del *desplegable* que finalmente constituirá nuestro *app*.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="dssbcs.juegopreguntas" >
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:name="dssbcs.juegopreguntas.Utilidad">
        <activity
            android:name="dssbcs.juegopreguntas.MenuActividad"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="dssbcs.juegopreguntas.ActividadPrincipal"
            android:label="@string/app_name"
            android:parentActivityName="dssbcs.juegopreguntas.MenuActividad">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="dssbcs.juegopreguntas.MenuActividad" />
        </activity>
        <activity
            android:name="dssbcs.juegopreguntas.Presentacion"
            android:label="@string/app_name"
            android:parentActivityName="dssbcs.juegopreguntas.MenuActividad">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="dssbcs.juegopreguntas.MenuActividad" />
        </activity>
        <activity
            android:name="dssbcs.juegopreguntas.Resultados"
            android:label="@string/app_name"
            android:parentActivityName="dssbcs.juegopreguntas.MenuActividad" >
        </activity>
        <activity
            android:name="dssbcs.juegopreguntas.Despedida"
            android:label="@string/app_name"
            android:parentActivityName="dssbcs.juegopreguntas.ActividadPrincipal" >
        </activity>
    </application>
</manifest>
```

Tabla 1: Archivo AndroidManifest.xml

El paquete de recursos más importante se denomina *layout*, que contiene las *vistas* o pantallas a las que dará lugar la ejecución de nuestra aplicación:

- `actividad_menu.xml`
- `actividad_principal.xml`
- `despedida.xml`
- `presentacion.xml`
- `resultados.xml`

La configuración completa de nuestro proyecto utilizando *Android Studio* viene descrita en un archivo `xml` denominado `AndroidManifest.xml`, que se puede ver en la tabla 1.

La carpeta Java de la figura 1 contendrá las clases que constituyen la aplicación del juego que vamos a desarrollar en esta práctica,

- `ActividadPrincipal`
- `Despedida`
- `MenuActividad`
- `Pregunta`
- `Presentacion`
- `Resultados`
- `Utilidad`

Así como también incluirá las clases "*helper*", por ejemplo: `DBHelper` y `DBPref`, para poder acceder a la base de datos (`database.sqlite`) de preguntas cuyo contenido se guardará como un recurso más en la carpeta: `res/raw/`.

La mencionada carpeta `res` contendrá los diferentes recursos: pantallas, base de datos, imágenes, iconos, ... a los que acceden las clases Java cuando necesitan obtener alguna información desde su entorno o bien modificar el estado de la vista o de la propia aplicación.

Figura 2: Pantalla correspondiente la vista `actividad_menu` de la aplicación

1.1 Las carpetas del proyecto

Vamos a pasar a describir los contenidos de la carpeta `\res` anterior (ver figura 1) por orden de aparición.

`\drawable`

Contiene todas las imágenes del juego en los formatos habituales (*jpg*, *png*, *gif*) y con los tamaños adecuados para ser mostrados en la pantalla de nuestro dispositivo móvil específico, que hemos seleccionado en el menú correspondiente `run` de la barra superior de menús del nuestro *app*.

```xml
<?xml version="1.0" encoding="utf-8"?>
    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        style="@style/Background_">
        <ImageView
            android:layout_marginTop="100dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/arenusca"
            android:layout_gravity="center"/>
        <TextView
            style="@style/Texto_introductorio"
            android:text="@string/welcome1"
            android:textAllCaps="false"
            android:fontFamily="sans-serif"
            android:textSize="35dp"
            android:textColor="#003366"/>
        <TextView
            style="@style/Texto_introductorio"
            android:text="@string/welcome2"
            android:id="@+id/go"
            android:textAllCaps="false"
            android:fontFamily="sans-serif"
            android:textSize="35dp"
            android:textColor="#003366"/>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">
            <Button
                style="@style/Botones_menu"
                android:text="Jugar"
                android:id="@+id/btn_jugar"
                android:background="#bada55"
                android:padding="15dp"
                android:layout_gravity="center"/>
            <Button
                style="@style/Botones_menu"
                android:text="Resultados"
                android:id="@+id/btn_resultados"
                android:background="#7aaee6"
                android:padding="15dp"
                android:layout_gravity="center"/>
            <Button
                style="@style/Botones_menu"
                android:text="Otros␣juegos"
                android:id="@+id/btn_ajustes"
                android:background="#ff88ae"
                android:padding="15dp"
                android:layout_gravity="center"/>
        </LinearLayout>
    </LinearLayout>
```

Tabla 2: Archivo `actividad_menu.xml`

`\layout`

Contiene todos los formatos de diseño que van a utilizar las clases de la aplicación. Tenemos que definir un archivo `xml` para cada una de las clases que utilizan estos recursos, que en el caso de la aplicación del presente ejemplo se denominan:
`actividad_menu.xml`, `actividad_principal.xml`, `despedida.xml`, `presentacion.xml` y `resultado.xml`. Uno de estos archivos de diseño sería el de la vista del menú principal o página de inicio de la aplicación (ver figura 2), que hemos denominado `actividad_menu.xml` (o `MenuActividad` en el archivo *Android Manifest*), cuyo contenido se muestra en la tabla 2.

\menu

Aquí se genera automáticamente el siguiente archivo `menu_main.xml`:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" tools:context=".MenuActividad">
    <item android:id="@+id/menu_inicial"
        android:title="Menu inicial"
        android:orderInCategory="100"
        android:showAsAction="never" />
</menu>
```

\mipmap

Aquí se ubican los distintos iconos o "logos" con que aparecerá nuestra *app* en aquellos dispositivos en los que se despliegue. Por defecto, si no incluimos otra imagen representativa de nuestra *app*, nos creará imágenes con el "marciano verde" de Android, para 4 posibles resoluciones de imagen:

- `ic_launcher.png` (48 px x 48 px) (mdpi)
- `ic_launcher.png` (72 px x 72 px) (hdpi)
- `ic_launcher.png` (96 px x 96 px) (xhdpi)
- `ic_launcher.png` (144 px x 144 px) (xxhdpi)

\raw

Aquí ubicaremos los registros de la base de datos, en formato texto, que llamaremos `database.sqlite`:

```
CREATE TABLE `preguntas` (
    _id INTEGER AUTO INCREMENT PRIMARY KEY,
    `pregunta` VARCHAR(256),
    `respuesta_correcta` VARCHAR(32) NOT NULL,
    `respuesta_incorrecta_1` VARCHAR(32) NOT NULL,
    `respuesta_incorrecta_2` VARCHAR(32) NOT NULL,
    `respuesta_incorrecta_3` VARCHAR(32) NOT NULL,
    `respuesta_incorrecta_4` VARCHAR(32) DEFAULT NULL,
    `respuesta_incorrecta_5` VARCHAR(32) DEFAULT NULL,
    `respuesta_incorrecta_6` VARCHAR(32) DEFAULT NULL,
    `respuesta_incorrecta_7` VARCHAR(32) DEFAULT NULL,
    `dificultad` CHARACTER NOT NULL DEFAULT 'e',
    `categoria` CHARACTER NOT NULL DEFAULT 'Z',
    `pregunta_tipo` VARCHAR(32) NOT NULL
);

-- historia    - A
-- geografia  - B
-- literatura - C
-- arte        - D
-- deportes      - E
-- ciencia    - F
-- tecnologia - G
-- astronomia - H
-- otras      - Z

-- incluir algunas preguntas para depuracion
INSERT INTO `preguntas` (`pregunta`, `respuesta_correcta`, `respuesta_incorrecta_1`,
 `respuesta_incorrecta_2`, `respuesta_incorrecta_3`, `categoria`, `pregunta_tipo`)
VALUES("El agua es esencial para la vida: en que porcentaje forma parte del peso total
del cuerpo humano?", "60-80%", "10-30%", "40-50%", "90-110%", 'G', '0');

--resto de preguntas
```

\values

En esta carpeta, además del archivo `dimens.xml` que contiene los márqenes de las pantallas por defecto y pantallas con más de 820dp de anchura disponible[1], se incluyen los 2 recursos siguientes:

1. `strings.xml`: contiene las cadenas globales que se utilizan en los archivos de la carpeta `layout` para las distintas pantallas de la aplicación.
2. `styles.xml`: contiene defincciones de los estilos que vamos a utilizar en nuestra aplicación para los diferentes *widgets*, botones, texto en la pantalla de presentación, etc.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="Version">Version 2.0 (c) Qualiastech</string>
    <string name="app_name">Juego de Agua</string>
    <string name="menu_inicial">Menu Inicial</string>
    <string name="despedida">Enhorabuena has respondido a todas las preguntas!</string>
    <string name="despedida1">Has respondido correctamente: \n</string>
    <string name="despedida2">Soy ARENUSCA de nuevo, muchas gracias por tu atencion. \n </string>
    <string name="despedida3"> Me has ayudado muchisimo, espero que tu tambien hayas aprendido!
    \n</string>
    <string name="play" >Jugar</string>
    <string name="stats" >Resultados</string>
    <string name="settings" >Otros juegos</string>
    <string name="welcome1" >Hola, \n Soy ARENUSCA y me gustaria retarte a que me demuestres
    cuanto sabes del agua &#8230; </string>
    <string name="welcome2">Te atreves?</string>
    <string name="tip" color="#F62217" textSize="33dp">Entonces, &#8230; esta envenenada el
    agua del embalse? \n  \n No, porque el agua no esta estancada, esta en continua renovacion,
    le esta entrando y saliendo agua permanentemente .
    </string>
</resources>
```

Tabla 3: Archivo `strings.xml` de la aplicación *Juego de Agua*

```
<resources>
    <style name="Background_"
        parent="android:Theme.Holo.Light.DarkActionBar">
        <item name="android:background">@drawable/background_juego</item>
    </style>
    <style name="Texto_introductorio">
        <item name="android:textAppearance">?android:attr/textAppearanceLarge</item>
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textSize">35dp</item>
        <item name="android:textColor">#003366</item>
     </style>
    <style name="Botones_menu"
        parent="@android:style/TextAppearance">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">match_parent</item>
        <item name="android:layout_marginLeft">100dp</item>
        <item name="android:layout_marginRight">100dp</item>
        <item name="android:layout_marginTop">5dp</item>
        <item name="android:layout_marginBottom">0dp</item>
        <item name="android:textSize">50dp</item>
        <item name="android:textStyle">bold</item>
        <item name="android:textAllCaps">false</item>
        <item name="android:fontFamily">sans-serif</item>
        <item name="android:visibility">visible</item>
    </style>
```

Tabla 4: Archivo `styles.xml` de la aplicación *Juego de Agua*

[1]por ejemplo, dispositivos de 7‘‘ y 8‘‘ en modo apaisado, que poseen sobre 960dp y 1280dp, respectivamente

1.2 Algunas clases de la aplicación

La base de datos anteriormente mencionada se leerá desde la clase–Java `DBHelper` de nuestro proyecto, utilizando para ello la instrucción `execSQL` dentro de un método de creación.

```
1  import android.content.Context;
2  import android.database.SQLException;
3  import android.database.sqlite.SQLiteDatabase;
4  import android.database.sqlite.SQLiteOpenHelper;
5  import android.util.Log;
6  import java.io.File;
7  import java.io.IOException;
8  import java.util.Scanner;
9  /**
10  * Created by Manuel on 17/12/2015.
11  */
12 public class DBHelper extends SQLiteOpenHelper {
13     public static final String NOMBRE_BD="Juego-preguntasDB";
14     public static final int VERSION_ACTUAL_BD=2;
15     protected SQLiteDatabase db;
16     protected Context ctx;
17
18     public DBHelper(Context context) {
19         super(context, NOMBRE_BD, null, VERSION_ACTUAL_BD);
20         this.ctx = context;
21         this.open();
22     }
23     public void onCreate(SQLiteDatabase sqLiteDatabase) {
24         Log.d("debug", "Creando la Base de Datos");
25         StringBuilder sb = new StringBuilder();
26
27         Scanner sc = new Scanner( this.ctx.getResources().openRawResource(R.raw.database) );
28         while(sc.hasNextLine()) {
29             sb.append(sc.nextLine());
30             sb.append('\n');
31             if (sb.indexOf(";") != -1) {
32                 sqLiteDatabase.execSQL(sb.toString());
33                 sb.delete(0, sb.capacity());
34             }
35         }
36     }
37     public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i2) {
38         Log.d("debug", "Actualizando la base de datos desde la version " + i + " a la version " + i2);
39     }
40     public void open() {
41         Log.d("debug", "Abriendo la base de datos");
42         this.db = this.getWritableDatabase();
43     }
44     public void close() {
45         Log.d("debug", "Cerrando la base de datos");
46         this.db.close();
47     }
48     public void addSampleData() {
49       try {
50             this.db.execSQL("DELETE FROM 'preguntas';");
51             this.db.execSQL("INSERT INTO 'preguntas' ('pregunta', 'respuesta_correcta',
52 'respuesta_incorrecta_1', 'respuesta_incorrecta_2', 'respuesta_incorrecta_3',
53 'categoria', 'pregunta_tipo')" +
54              "VALUES(\"El agua es esencial para la vida: En que porcentaje forma parte
55 del peso total del cuerpo humano?\", \"60-80%\", \"10-30%\", \"40-50%\",
56 \"90-110%\", 'G', '0');");
57              //RESTO DE PREGUNTAS
58       }
59       catch (Exception e) {
60             e.printStackTrace();
61             Log.d("debug", e.toString());
62         }
63     }
64 }
```

La clase Java `ActividadPrincipal` obtiene una nueva pregunta de la base de datos llamando al método `getPreguntas(...)` de una instancia de `DBPref` en la siguiente instrucción:

```
Cursor cuestiones =
        this.db.getPreguntas(DBPref.Categoria.TECNOLOGIA, DBPref.Dificultad.FACIL, PREGUNTAS);
```

La clase `DBPref` actúa como un adaptador de la instancia de la base de datos `sqlLiteDatabase`, que aparece en la clase `DBHelper` para que pueda ser utilizada por `ActividadPrincipal`:

```
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
/**
 * Created by Manuel on 17/12/2015.
 */
public class DBPref extends DBHelper {
    public static enum Categoria {
        HISTORIA('A'),
       //...
        TECNOLOGIA('G'),
        OTROS('Z');
        public final char C;
        Categoria(char c) {
            this.C = c;
        }
    }
    public static enum Dificultad {
        FACIL('e'),
        MEDIA('m'),
        DIFICIL('h');
        public final char D;
        Dificultad(char d) {
            this.D = d;
        }
    }
    public DBPref(Context contexto) {
        super(contexto);
    }


    public void addRegistro(Pregunta pregunta) {
        ContentValues valores = new ContentValues();
        valores.put("pregunta", pregunta.getPregunta());
        valores.put("respuesta_correcta", pregunta.getRespuesta());

        String[] respuestas_erroneas = pregunta.getRespuestasErroneas();
        for(int i = 0; i < respuestas_erroneas.length; i++) {
            valores.put("respuesta_incorrecta_" + i, respuestas_erroneas[i]);
        }
        db.insert("preguntas", null, valores);
    }
    public void addRegistros(Pregunta[] preguntas) {
        for(int i = 0; i < preguntas.length; i++) {
            this.addRegistro(preguntas[i]);
        }
    }
    public Cursor getPreguntas(Categoria c, Dificultad d, int limit) {
        return this.db.rawQuery("SELECT `pregunta`, `respuesta_correcta`,
        `respuesta_incorrecta_1`, `respuesta_incorrecta_2`," + "
        `respuesta_incorrecta_3`, `pregunta_tipo` FROM `preguntas`
        WHERE Categoria = ? AND Dificultad = ?" + "ORDER BY RANDOM() LIMIT ?",
          new String[]{String.valueOf(c.C), String.valueOf(d.D), String.valueOf(limit)});
    }
    //...
}
```

1.3 Despliegue de la aplicación móvil

Un `app` correctamente diseñado, desarrollado y probado ha de ser desplegado en un espacio público para que los potenciales usuarios de la aplicación puedan acceder a ella. El despliegue de una aplicación de este tipo se puede realizar utilizando los servicios de un Cloud o en un SW.

En el caso de `apps` desarrolladas para dispositivos con Android se puede hacer el despliegue mediante un archivo específico (`.apk`) en *Play Store* de Google.

La generación del archivo `apk` y ejecución de la aplicación en dispositivos móviles (teléfonos y tablets) que asuman el sistema operativo Android con un API igual o superior a 21 (versión $>= 4.4$ de Android), se puede realizar en el IDE *Android Studio* directamente, siguiendo el siguiente esquema general:

1. En modo *debug*, generar el *keystore* (para guardar las claves de la aplicación) y el archivo `apk`
2. Borrar el `apk`
3. Abrir un terminal o consola (`cmd` de Windows) y ejecutar la siguiente orden:
   ```
   keytool -list -v -keystore
    C:\<tu directorio para guardar keystores de
   Android Studio>\nombreKey.jks
   ```
4. Copiar la secuencia hexadecimal completa que aparece con el nombre `SHA`
5. Dar de alta en *Google Console Developer* el SHA anterior
6. Regresando a nuestro proyecto en Android Studio, ir a `Project Structure/signing` y seleccionar el *keystore* generado en el paso (1)
7. En Project *Structure/Flavors/signing config* seleccionar el keystore del paso (6)
8. Cambiar al modo *release*
9. En el menú *Build Project*, realizar un `clean/rebuild` y probar a ejecutar (run)
10. Utilizando el keystore del paso (1), generar el denominado *signed apk* que es el desplegable para dispositivos móviles

Créditos

- `http://doc.qt.io/qt-5/deployment-android.html`
- `http://www.vogella.com/tutorials/Android/article.html`
- Android developer homepage `http://developer.android.com/index.html`
- Wiki "Android bug tracker" `https://code.google.com/p/android/issues/list`
- Deploying an Android app `http://docs.twixlmedia.com/m/twixl-publisher-app/l/243432-deploying-an-android-app-in-google-play`