UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA MECÁNICA

"DESARROLLO DE UNA HERRAMIENTA DE AUTOEVALUACIÓN PARA EL SISTEMA OPERATIVO ANDROID"

TRABAJO FIN DE GRADO

Enero - 2019

AUTOR: Alfredo Fenoll Quirant

DIRECTOR/ES: César Fernádez Peris

ÍNDICE

TABLA DE ILUSTRACIONES

Agradecimientos

A mis padres, a los que agradezco haberme inspirado y respaldado durante toda mi educación.

A mi hermano, por apoyarme en los momentos difíciles.

Al profesor César Fernández Peris, tutor de mi TFG, por haberme guiado y ayudado durante estos últimos meses en la realización de mi trabajo y por todos esos conocimientos sobre programación que seguro que agradezco todavía más en el futuro.

A la Universidad Miguel Hernández y a todo su profesorado y demás trabajadores del campus, por crear este fantástico ambiente donde los estudiantes nos podemos desarrollar tanto a nivel académico como a nivel personal.

A todos los amigos que he hecho durante mis años de estudiante, han sido una vía de escape para poder sobrellevar los problemas con los que me he encontrado durante mi etapa universitaria.

A mi abuelo por darme una perspectiva diferente de la vida, de una persona que ha basado su vida en la cultura del esfuerzo y la humildad.

A toda la gente que me ayudó en mi etapa Erasmus, tanto desde la Universidad Miguel Hernández como desde la AGH university of science and technology, de Cracovia incluyendo también toda aquella gente maravillosa que conocí en Polonia.

Y a todas aquellas personas que, de una forma u otra, me han ayudado a lo largo de esta grata aventura que ha resultado ser mi vida en la universidad.

1 Introducción

Este documento tiene como objetivo describir el proyecto realizado por el alumno de ingeniería mecánica Alfredo Fenoll Quirant de la Universidad Miguel Hernández.

Dicho proyecto consiste en la realización de una aplicación para el sistema operativo Android, desarrollada en la plataforma de desarrollo oficial de dicho sistema operativo, Android Studio.

La aplicación es capaz de realizar exámenes tipo test adaptando el nivel de dificultad de las preguntas en relación al número de acierto que vaya registrando, dando una calificación final en función del número de fallos contabilizados y del nivel de dificultad de las preguntas alcanzado al final de dicho test.

Además, se podrán personalizar las preguntas de dicho test sin necesidad de programar directamente la app, esto será posible creando un fichero .csv (documento de texto separado por comas) guardándolo en google drive y obteniendo un enlace público tipo .url que será lo que tenga que ser introducido en la app.

1.1 Motivación y objetivos del proyecto

Hoy en día estamos rodeamos de aparatos inteligentes, gobernados por un software que tiene como objetivo simplificar los desafíos que nos encontramos día a día.

Este concepto, que ha ido desarrollándose desde los años 40 del siglo pasado, cuando aparecieron las primeras computadoras y que continúa evolucionando a una velocidad vertiginosa en nuestros días, se ha convertido en una pieza fundamental dentro de la concepción de la realidad socioeconómica de nuestro tiempo.

Por lo tanto, considerando que rama del conocimiento que estoy estudiando, me veo en la necesidad imperiosa de aprender, al menos, un mínimo de programación, pues estos conocimientos, los encuentro básicos para entender mejor como funciona mucha de la tecnología con la que voy a estar en contacto en mi vida profesional.

Por esto, cuando me llegó la información sobre el curso de desarrollo que César iba a impartir no dudé en apuntarme, considero que este curso ha sido un gran complemento a la educación que he recibido como ingeniero mecánico e, insto a los responsables del programa educativo de los grados de ingeniería a que refuercen el campo de la programación informática y el desarrollo de software, que tan escaso en tiempo y recursos he encontrado en mis ya muchos años como alumno de esta universidad.

Una vez quedaron impartidas las clases me surgió la duda de que aplicación podría crear, con la intención de poder darle una utilidad real y no solo se quedase en un proyecto académico.

A mi mente vino la posibilidad de crear un test, pero con una particularidad, que este fuese capaz de ir adaptando el nivel de las preguntas dependiendo del porcentaje de aciertos del usuario, y así poder afinar de una manera más precisa el nivel de conocimientos que posee el usuario.

Esta idea surgió de los test de nivel online de idioma que se pueden realizar en páginas web como la de Cambridge English (https://www.cambridgeenglish.org/es/test- your-english/).

En este tipo de test, conforme aciertas las preguntas van ganando mayor complejidad, ya que en ciertas materias tan profundas como un idioma acertar un porcentaje alto de preguntas de nivel básico no te convierte en un experto en el idioma, y viceversa, fallar muchas preguntas de alto nivel no significa que no sepas nada de dicha lengua.

De ahí surge la necesidad de categorizar por nivel de complejidad las diferentes habilidades que se necesita para dominar una lengua.

Esta forma de medir el nivel que se utiliza en este ejemplo me parece extrapolable a muchos otros campos del conocimiento entre ellos muchas de las ramas de las que se compone la ingeniería.

Por otra parte, la aplicación no sólo debe estar preparada para unas preguntas ya predefinidas, si no, que estas preguntas deben ser editables por cualquiera, independientemente de sus conocimientos de programación, así que estas preguntas son completamente programables sin necesidad de alterar el propio código de la aplicación.

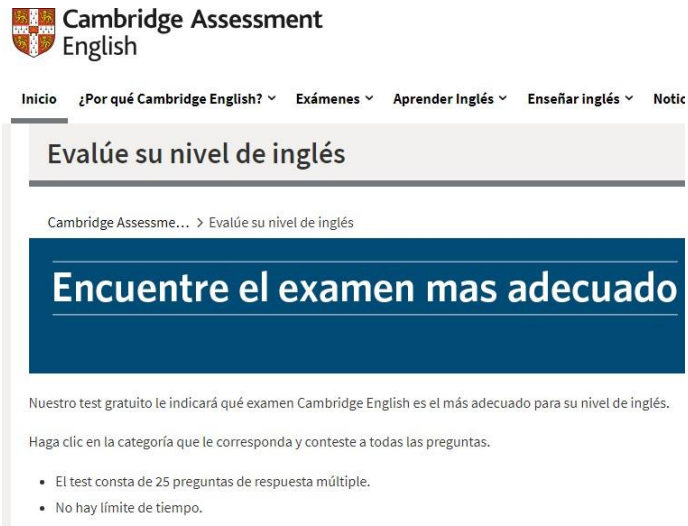


Ilustración 1: vista página web Cambridge

1.2 Contexto de Desarrollo

Como ya he mencionado anteriormente, la aplicación ha sido desarrollada para el sistema operativo Android.

Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes, tabletas y también para relojes inteligentes, televisores y automóviles.

Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró.

Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles, ese mismo año se anuncia la primera versión del sistema operativo: Android 1.0 Apple Pie.

Aunque terminales con Android no estarían disponibles hasta el año 2008.

El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008.

La implantación de Android ha sido progresiva desde unos inicios marcados por una gran fragmentación en la elección de cada fabricante en el sistema operativo escogido para sus dispositivos.

Actualmente Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado superior al 85 % al año 2018, muy por encima de IOS, el sistema operativo para sistemas móviles de la compañía Apple y segundo sistema operativo más extendido.

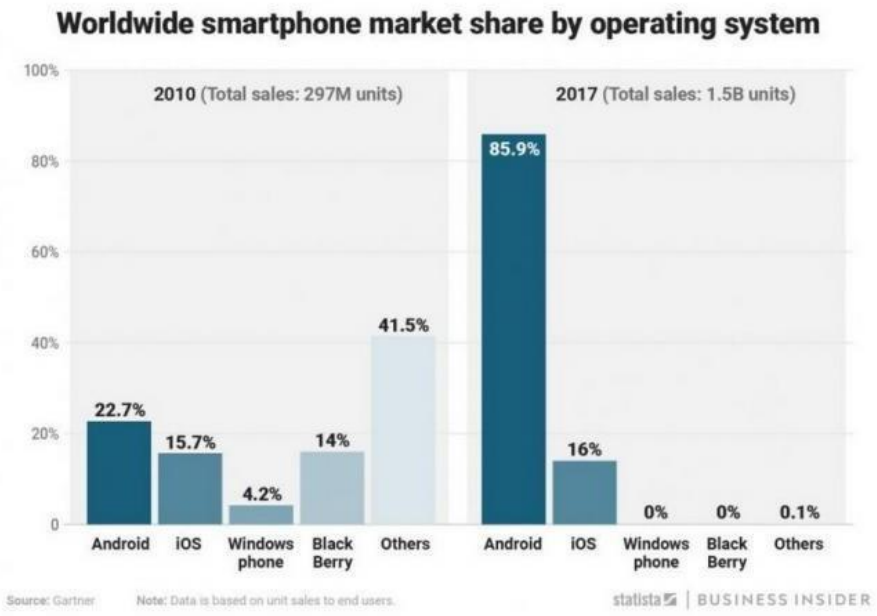


Ilustración 2: Evolución cuota de mercado Android

La historial de las versiones de sistema operativo Android se inició con el lanzamiento de Android beta en noviembre de 2007.

La primera versión comercial (de prueba), Android 1.0, fue lanzada en septiembre de 2008. Android es un sistema operativo móvil desarrollado por Google y la Open Handset Alliance, y ha visto un número de actualizaciones a su sistema operativo base desde su lanzamiento original. Estas actualizaciones típicamente corrigen fallos de programa y agregan nuevas funcionalidades.

Desde abril de 2009, las versiones de Android han sido desarrolladas bajo un nombre en clave y sus nombres siguen un orden alfabético: Cupcake, Donut, Éclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo y el último en sumarse a la lista fue Android Pie, la novena y última versión principal y la decimosexta versión del sistema operativo Android.

Android "Pie" fue lanzado oficialmente el 6 de agosto de 2018.

Nombre código	Número de versión	Fecha de lanzamiento	Nivel de API
Apple Pie[1]	1.0	23 de septiembre de 2008	1
Banana Bread[1]	1.1	9 de febrero de 2009	2
Cupcake	1.5	25 de abril de 2009	3
Donut	1.6	15 de septiembre de 2009	4
Eclair	2.0–2.1	26 de octubre de 2009	5-7
Froyo	2.2–2.2.3	20 de mayo de 2010	8
Gingerbread	2.3–2.3.7	6 de diciembre de 2010	9–10
Honeycomb[2]	3.0–3.2.6	22 de febrero de 2011	11–13
Ice Cream Sandwich	4.0–4.0.5	18 de octubre de 2011	14–15
Jelly Bean	4.1–4.3.1	9 de julio de 2012	16–18
KitKat	4.4–4.4.4, 4.4W–4.4W.2	31 de octubre de 2013	19–20
Lollipop	5.0–5.1.1	12 de noviembre de 2014	21–22
Marshmallow	6.0–6.0.1	5 de octubre de 2015	23
Nougat	7.0 - 7.1 - 7.1.1 - 7.1.2	15 de junio de 2016	24-25
Oreo	8.0 - 8.1	21 de agosto de 2017	26-27
Pie	9.0	6 de agosto de 2018	28

Ilustración 3: Versiones sistema operativo Android

El problema con las versiones de Android es que no todos los dispositivos se actualizan a la última versión, esto, unido al poco espacio temporal entre unas versiones y otras, hacen que haya muchos dispositivos operativos con distintas versiones de Android.

Este factor se ha de tener en cuenta a la hora de desarrollar en Android, cada versión de Android tiene un nivel de API o Application Programming Interface (interfaz de programación de aplicaciones), esto es importante porque utilizar un nivel u otro de API puede hacer que una aplicación no pueda ser utilizada en versiones más antiguas de Android, pero por otro lado cuanto mayor sea el nivel de API mayor número de ventajas y recursos para el desarrollador.

Por este motivo hay que alcanzar un compromiso entre las ventajas que puedan ofrecer las nuevas versiones de Android y el número de dispositivos en el mercado con un sistema operativo capaz de ejecutar tu aplicación.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	1.9%
4.2.x		17	2.9%
4.3		18	0.8%
4.4	KitKat	19	12.8%
5.0	Lollipop	21	5.7%
5.1		22	19.4%
6.0	Marshmallow	23	28.6%
7.0	Nougat	24	21.1%
7.1		25	5.2%
8.0	Oreo	26	0.5%
8.1		27	0.2%

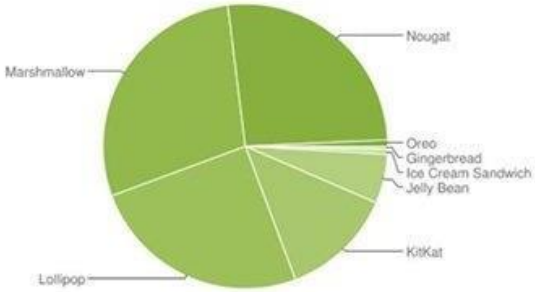


Ilustración 4: Fragmentación dispositivos Android 2018

2. Descripción de la App

La aplicación se llama IntelliTest y sirve para realizar test de hasta cuatro opciones por pregunta y un total de 15 preguntas, una vez respondidas las preguntas dará una calificación dependiendo de una serie de parámetros de describiré más adelante.

Dentro de una aplicación desarrollada en Android Studio tenemos distintas actividades, cada actividad consta de una interfaz html que es lo que se muestra en pantalla y de un archivo .java que es donde va el código asociado a dicha actividad.

La primera actividad de la app se llama "Main Activity", es la que nos aparece cuando abrimos la aplicación:

Ilustración 5: Vista de la pantalla principal de la app.

Como podemos ver en la ilustración, esta Activity se compone del nombre de la aplicación y tres botones.

El botón principal es el "Start Quiz", si pulsamos este botón la aplicación cargará las preguntas predefinidas y pasaremos a la segunda actividad "Test Activity", que describiremos más adelante. Además, disponemos de dos botones más pequeños en la parte inferior, el de la izquierda, es el botón "Elegir Preguntas". Si elegimos esta opción el programa lanzará la actividad "Choose Activity" donde podremos elegir nuestras propias preguntas.

Por último, tenemos el botón ayuda, donde obtendremos unas indicaciones del modo de funcionamiento de la aplicación.

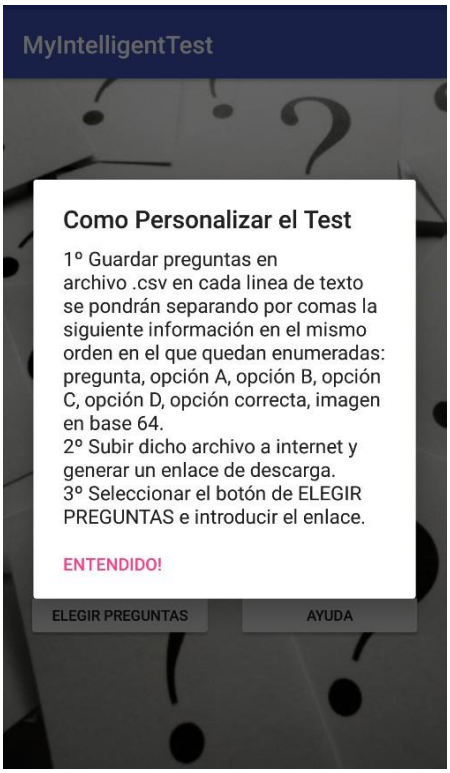


Ilustración 6: Vista de la pantalla ayuda de la app.

La segunda actividad que vamos a describir es “Choose Activity”:

Ilustración 7: Vista de la pantalla principal de la app.

En esta actividad tenemos dos elementos principales.

Por un lado, tenemos una casilla de texto editable, en ella introduciremos un enlace de google drive donde previamente habremos guardado nuestro archivo .csv con nuestras preguntas personalizadas.

El otro elemento que aparece es el botón “Empezar Test Personalizado”, una vez pulsemos este botón nos llevará a la actividad “Test activity” pero de esta forma en dicha actividad se cargarán las preguntas del archivo .csv en vez de las ya precargadas en la aplicación, que es lo que pasa si lanzamos “Test Activity” desde “Main Activity”.

Ilustración 8: Vista de la pantalla test de la app.

En la ilustración 8 vemos la interfaz “Test Activity”, en ella, contamos con 6 elementos principales, arriba del todo tenemos un cuadro de texto donde aparecerá la pregunta, el elemento de abajo es donde irá la ilustración que acompañe a la pregunta, y los otros 4 elementos que encontramos más abajo son los cuatro botones donde aparecerán las cuatro opciones que podemos contestar.

En esta actividad irán apareciendo 15 preguntas, que variarán dependiendo de si acertamos o fallamos de la siguiente manera:

- Las preguntas se organizan en bloques de 5 preguntas.
- Las preguntas están categorizadas en 3 categorías diferentes, aprobado, notable, y sobresaliente.
- El test empieza en las preguntas de la categoría aprobado, que es la básica, de la cual no se puede descender, si de este primer bloque de 5 preguntas acertamos 4 o más preguntas, el siguiente bloque de preguntas será de la categoría de notable
- Para ascender a una categoría superior hay que acertar al menos 4 preguntas del bloque de preguntas actual, si contestamos correctamente 2 o menos, descenderíamos, si acertamos 3 mantendríamos la misma categoría para el siguiente bloque.
- En el bloque de sobresaliente no tendríamos posibilidad de ascender más y al contrario pasaría con el bloque de aprobado donde no podríamos descender.

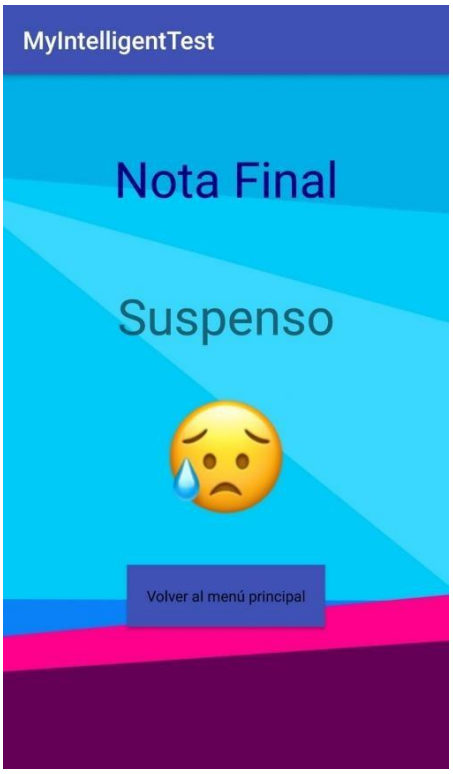


Ilustración 9: Vista de la pantalla puntuación de la app.

Una vez contestadas 15 preguntas se abriría automáticamente la actividad "Score activity" Donde nos aparecerían tres elementos principales:

-Un recuadro de texto donde aparecería la puntuación final, es calificación se obtiene dependiendo de dos variables, por un lado, la categoría final donde has acabado el test, y por otro el número total de preguntas acertadas, vamos a describir las diferentes posibilidades:

1 Acabamos en la categoría sobresaliente, en este caso la calificación sería de sobresaliente
2 Acabamos en la categoría notable, en este caso la calificación sería de notable
3 Acabamos en la categoría aprobado y el número total de preguntas acertadas es mayor de la mitad del total de preguntas, la calificación es de aprobado, en caso contrario, sería de suspenso.
4 Acabamos en la categoría aprobado y el número total de preguntas acertadas es menor de la mitad del total de preguntas, la calificación es de suspenso.

-Un recuadro de imagen donde nos aparecería un emoticono diferente dependiendo de nuestra calificación, hay 4 diferentes (tantos como número de posibles calificaciones):

Sobresaliente Aprobado

Suspenso Notable

Ilustración 10: Emojis asociados a cada puntuación.

-Un botón que nos permite regresar a la actividad de inicio da la aplicación “Main Activity”

3. Materiales y Métodos

3.1 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA.

Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu productividad durante la compilación de apps para Android, como las siguientes:

- Un sistema de compilación basado en Gradle flexible
- Un emulador rápido con varias funciones
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android
- Instant Run para aplicar cambios mientras tu app se ejecuta sin la necesidad de compilar un nuevo APK
- Integración de plantillas de código y GitHub para ayudarte a compilar funciones comunes de las apps e importar ejemplos de código
- Gran cantidad de herramientas y frameworks de prueba
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versión, etc.
- Compatibilidad con C++ y NDK
- Soporte incorporado para Google Cloud Platform, lo que facilita la integración de Google Cloud Messaging y App Engine.

3.1.1 Estructura del proyecto

Cada proyecto en Android Studio contiene uno o más módulos con archivos de código fuente y archivos de recursos. Entre los tipos de módulos se incluyen los siguientes:

-módulos de apps para Android

-módulos de bibliotecas

-módulos de Google App Engine

De manera predeterminada, Android Studio muestra los archivos de tu proyecto en la vista de proyectos de Android, como se muestra en la figura 1. Esta vista se organiza en módulos para proporcionar un rápido acceso a los archivos de origen clave de tu proyecto.

Todos los archivos de compilación son visibles en el nivel superior de Secuencias de comando de Gradle y cada módulo de la aplicación contiene las siguientes carpetas:

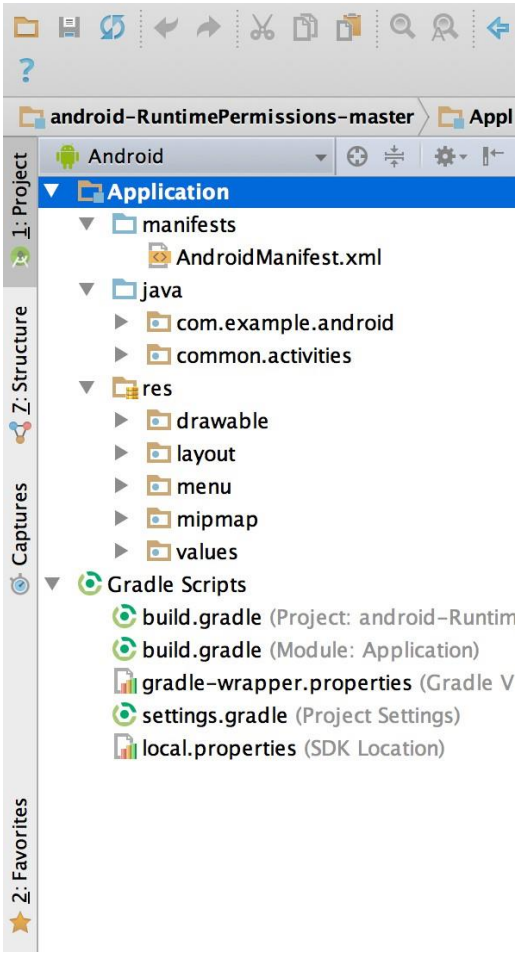


Ilustración 11: Archivos del proyecto

- Manifests: contiene el archivo AndroidManifest.xml.
- Java: contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.
- Res: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

La estructura del proyecto para Android en el disco difiere de esta representación plana. Para ver la estructura de archivos real del proyecto, selecciona Project en la lista desplegable Project (en la figura 1 se muestra como Android).

También puedes personalizar la vista de los archivos del proyecto para concentrarte en aspectos específicos del desarrollo de tu app. Por ejemplo, al seleccionar la vista Problems de tu proyecto, aparecerán enlaces a los archivos de origen que contengan errores conocidos de codificación y sintaxis, como una etiqueta de cierre faltante para un elemento XML en un archivo de diseño.

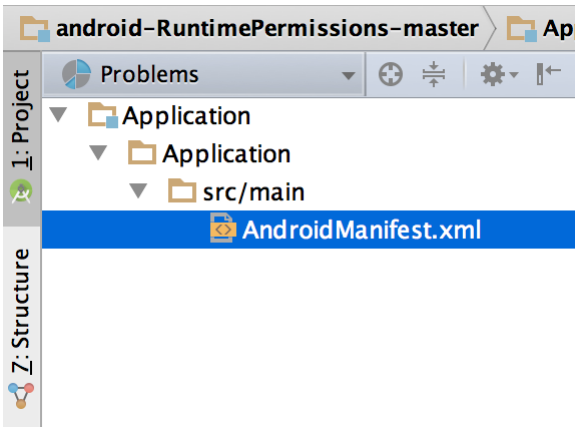


Ilustración 12: Archivos del proyecto en la vista Problems, en la que se muestra un archivo de diseño con un problema.

3.1.2 Interfaz de usuario

La ventana principal de Android Studio consta de varias áreas lógicas que se identifican en la figura 3.

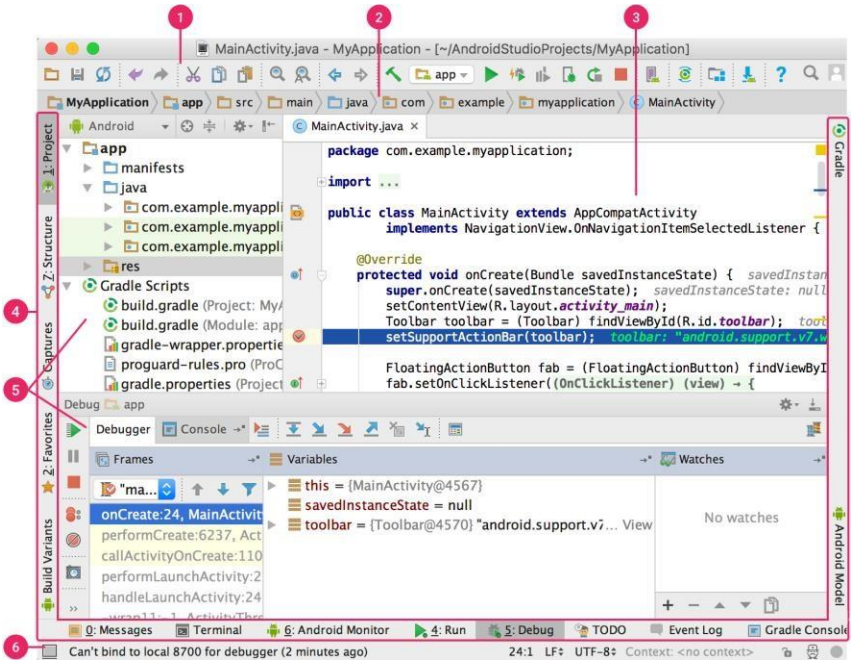


Ilustración 13: Ventana principal de Android Studio.

La barra de herramientas te permite realizar una gran variedad de acciones, como la ejecución de tu app y el inicio de herramientas de Android.

La barra de navegación te ayuda a explorar tu proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana Project.

La ventana del editor es el área donde puedes crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.

La barra de la ventana de herramientas se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que te permiten expandir o contraer ventanas de herramientas individuales.

Las ventanas de herramientas te permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Puedes expandirlas y contraerlas.

En la barra de estado, se muestra el estado de tu proyecto y del IDE en sí, como también cualquier advertencia o mensaje.

Puedes organizar la ventana principal para tener más espacio en pantalla ocultando o desplazando barras y ventanas de herramientas. También puedes usar combinaciones de teclas para acceder a la mayoría de las funciones del IDE.

En cualquier momento, puedes realizar búsquedas en tu código fuente, bases de datos, acciones, elementos de la interfaz de usuario, etc., presionando dos veces la tecla Shift o haciendo clic en la lupa que se encuentra en la esquina superior derecha de la ventana de Android Studio. Esto puede ser muy útil, por ejemplo, si intentas localizar una acción específica del IDE que olvidaste cómo activar.

3.2 Diseño y programación

Vamos a ir comentando el código de la app conforme lo vemos estructurado en el propio proyecto, primero, veremos el código de AndroidManifest, describiendo cuál es su función y cuáles son los elementos más importantes a tener en cuenta.

Seguidamente veremos los archivos java que se integran dentro de la app, aclarando el propósito de las líneas de código del que se componen los diferentes archivos.

A continuacion, dentro de la carpeta resources, veremos los elementos que forman el layout de la app, adjuntando tanto el código como los diseños resultantes de dicho código.

Por último, veremos 3 archivos más que se encuentran dentro de la carpeta values, importantes a la hora de complementar lo definido en el layout.

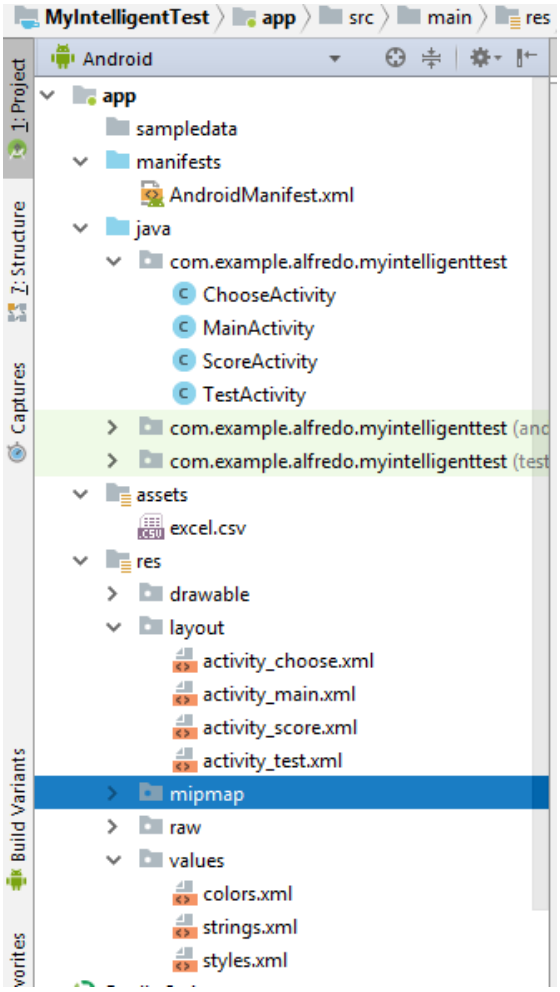


Ilustración 14: Estructura de mi proyecto

3.2.1 Android Manifest

Todas las aplicaciones deben tener un archivo AndroidManifest.xml (con ese nombre exacto) en el directorio raíz.

El archivo de manifiesto proporciona información esencial sobre tu aplicación al sistema Android, información que el sistema debe tener para poder ejecutar el código de la app.

Entre otras cosas, el archivo de manifiesto hace lo siguiente:

- Nombra el paquete de Java para la aplicación. El nombre del paquete sirve como un identificador único para la aplicación.
- Describe los componentes de la aplicación, como las actividades, los servicios, los receptores de mensajes y los proveedores de contenido que la integran. También nombra las clases que implementa cada uno de los componentes y publica sus capacidades, como los mensajes Intent con los que pueden funcionar. Estas declaraciones notifican al sistema Android los componentes y las condiciones para el lanzamiento.
- Determina los procesos que alojan a los componentes de la aplicación.
- Declara los permisos debe tener la aplicación para acceder a las partes protegidas de una API e interactuar con otras aplicaciones. También declara los permisos que otros deben tener para interactuar con los componentes de la aplicación.
- Enumera las clases Instrumentation que proporcionan un perfil y otra información mientras la aplicación se ejecuta. Estas declaraciones están en el manifiesto solo mientras la aplicación se desarrolla y se quitan antes de la publicación de esta.
- Declara el nivel mínimo de Android API que requiere la aplicación.
- Enumera las bibliotecas con las que debe estar vinculada la aplicación.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.alfredo.myintelligenttest">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="MyIntelligentTest"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />


                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".TestActivity"></activity>
        <activity android:name=".ScoreActivity"></activity>
        <activity android:name=".ChooseActivity"></activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

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

</manifest>
```

Ilustración 15: Manifest de mi App

De las líneas de código aquí escritas destacar las siguientes:

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

Con esta línea damos permiso a la aplicación a descargar documentos de internet, esto es muy importante ya que una de las funcionalidades básicas de la aplicación es que cualquiera pueda modificar las preguntas del test desde un archivo CSV guardado en Google Drive, para ello es necesario darle permiso a la app para que descargue ficheros de internet.

2. `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`

Permite que una aplicación escriba en almacenamiento externo.

3. `<uses-permission android:name="android.permission.WRITE_INTERNAL_STORAGE" />`

Permite que una aplicación escriba en almacenamiento interno.

3.2.2 Java

En la carpeta Java encontramos el código fuente de la aplicación, y cada uno de los archivos que encontramos en esta carpeta se corresponde con una de las interfaces de usuario que hemos creado, su función dentro de la aplicación es dotar de funcionalidades a todos los elementos que hemos creado anteriormente, como botones, visores de texto, ventanas editables de texto…

ActivityMain

Este archivo .java es el que complementa al archivo .xml Activity_main:

MainActivity.java

```
package com.example.alfredo.myintelligenttest;

import ...

public class MainActivity extends AppCompatActivity {

    //Creación de objetos
    ImageButton comenzar;
    Button elegir, ayuda;

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

        //Enlazado de mi objeto con .java
        comenzar = (ImageButton) findViewById(R.id.comenzar);
        elegir = (Button) findViewById(R.id.elegir);
        ayuda = (Button) findViewById(R.id.help);

        //event listener
        comenzar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent =  new Intent( packageContext: MainActivity.this, TestActivity.class);
                intent.putExtra( name: "tipo",  value: "interna");
                startActivity(intent);
                }
        });

        ayuda.setOnClickListener((view) → {
                AlertDialog alertDialog = new AlertDialog.Builder( context: MainActivity.this).create();
                alertDialog.setTitle("Como Personalizar el Test");
                alertDialog.setMessage("1° Guardar preguntas en archivo .csv en cada linea de texto se pondrán" +
                        " separando por comas la siguiente información en el mismo orden en el que quedan enumeradas:" +
                        " pregunta, opción A, opción B, opción C, opción D, opción correcta, imagen en base 64.\n" +
                        "2° Subir dicho archivo a internet y generar un enlace de descarga.\n" +
                        "3° Seleccionar el botón de ELEGIR PREGUNTAS e introducir el enlace.");
                alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL,  text: "Entendido!",
                        (OnClickListener) (dialog, which) → {
                                dialog.dismiss();
                        });
                alertDialog.show();
        });

        elegir.setOnClickListener((view) → {
                Intent intent =  new Intent( packageContext: MainActivity.this, ChooseActivity.class);
                startActivity(intent);
        });
    }
}
```

Ilustración 16: Código del MainActivity.java

Procedo a explicar el código aquí implementado:

Para empezar lo primero que hacemos en este .java es definir las variables con las que vamos a trabajar y el tipo de objeto al que hacen referencia:

```
ImageButton comenzar;
Button elegir, ayuda;
```

Seguidamente enlazaremos estas variables con los botones que hemos representado en nuestro archivo .xml Activity_main, en esta operación cobra importancia el atributo id que hemos elegido en el .xml para cada objeto:

```
comenzar = (ImageButton) findViewById(R.id.comenzar);
elegir = (Button) findViewById(R.id.elegir);
ayuda = (Button) findViewById(R.id.help);
```

Una vez realizado estos dos pasos, que son imprescindibles para cualquier .java que tengamos que programar pasamos a implementar las funciones que tenemos que realizar en esta pantalla:

- Pulsando el botón comenzar debemos pasar a la pantalla Activity_test esto se realiza mediante el siguiente código:

```
comenzar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent =  new Intent(MainActivity.this,
TestActivity.class);
        intent.putExtra("tipo", "interna");
        startActivity(intent);
        }
});
```

En esta orden hay una particularidad más, y es que al abrir la nueva pantalla pulsando este botón pasamos una variable llamada “tipo” con el valor de “interna”. Volveremos a hablar de esta variable en el .java ActivityTest.

- Pulsando el botón elegir debemos pasar a la pantalla Activity_Choose esto se realiza mediante el siguiente código:

```
elegir.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent =  new Intent(MainActivity.this,
ChooseActivity.class);
        startActivity(intent);
    }
});
```

- Pulsando el botón ayuda se despliega un texto mostrándonos los pasos que debemos seguir para personalizar las preguntas de nuestro test esto se realiza mediante el siguiente código:

```
ayuda.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog alertDialog = new
AlertDialog.Builder(MainActivity.this).create();
        alertDialog.setTitle("Como Personalizar el Test");
        alertDialog.setMessage("1° Guardar preguntas en archivo
.csv en cada linea de texto se pondrán" +
                " separando por comas la siguiente información
en el mismo orden en el que quedan enumeradas:" +
                " pregunta, opción A, opción B, opción C, opción
D, opción correcta, imagen en base 64.\n" +
                "2° Subir dicho archivo a internet y generar un
enlace de descarga.\n" +
                "3° Seleccionar el botón de ELEGIR PREGUNTAS e
introducir el enlace.");
        alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL,
"Entendido!",
                new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,
int which) {
                        dialog.dismiss();
                    }
                });
        alertDialog.show();
    }
});
```

ActivityChoose

Este archivo .java es el que complementa al archivo .xml Activity_Choose:

ChooseActivity.java

```java
package com.example.alfredo.myintelligenttest;

import ...

public class ChooseActivity extends AppCompatActivity {

    //Creación de objetos

    Button start;
    EditText URLName;
    private ProgressDialog pDialog;

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

        //Enlazado de mi objeto con .java
        start = (Button) findViewById(R.id.start);
        URLName = (EditText)findViewById(R.id.URL);

        //event listener
        start.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

                new DownloadFileFromURL().execute(URLName.getText().toString());
            }



            class DownloadFileFromURL extends AsyncTask<String, String, String> {

                /**
                 * Before starting background thread
                 * */
                @Override
                protected void onPreExecute() {
                    super.onPreExecute();
                    System.out.println("Empezando descarga");
                }

                /**
                 * Downloading file in background thread
                 * */
                @Override
                protected String doInBackground(String... f_url) {
                    int count;
                    try {
                        String root = Environment.getExternalStorageDirectory().toString();

                        System.out.println("Descargando");
                        URL url = new URL(f_url[0]);

                        URLConnection conection = url.openConnection();
                        conection.connect();
                        // getting file length
                        int lenghtOfFile = conection.getContentLength();

                        // input stream to read file - with 8k buffer
                        InputStream input = new BufferedInputStream(url.openStream(), size: 8192);

                        // Output stream to write file

                        FileOutputStream output =  openFileOutput( name: "excel_externo.csv", Context.MODE_PRIVATE);
```

```
88                  byte data[] = new byte[1024];
89
90                  long total = 0;
91                  while ((count = input.read(data)) != -1) {
92                      total += count;
93
94                      // writing data to file
95                      output.write(data,  off: 0, count);
96
97                  }
98
99                  // flushing output
100                 output.flush();
101
102                 // closing streams
103                 output.close();
104                 input.close();
105
106             } catch (Exception e) {
107                 Log.e( tag: "Error: ", e.getMessage());
108             }
109
110             return null;
111         }
112
113         /**
114          * After completing background task
115          * **/
116         @Override
117         protected void onPostExecute(String file_url) {
118
119             System.out.println("Descargado");
120
121             Intent intent = new Intent( packageContext: ChooseActivity.this, TestActivity.class);
122             intent.putExtra( name: "tipo",  value: "externa");
123             startActivity(intent);
124         }
125
126     }
127 });
128
129 }}
```

Ilustración 17: Código del ChooseActivity.java

Procedo a explicar el código aquí implementado:

Para empezar lo primero que hacemos en este .java es definir las variables con las que vamos a trabajar y el tipo de objeto al que hacen referencia:

```
Button start;
EditText URLName;
```

Seguidamente, enlazaremos estas variables con los botones que hemos representado en nuestro archivo .xml Activity_choose, en esta operación cobra importancia el atributo id que hemos elegido en el .xml para cada objeto:

```
start = (Button) findViewById(R.id.start);
URLName = (EditText)findViewById(R.id.URL);
```

A continuación, tendríamos el método View.OnClickListener sobre el botón start, este método se activa cuando el usuario pulsa el botón.

```
start.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        new DownloadFileFromURL().execute(URLName.getText().toString());
    }
```

Por último, con el fragmento que tenemos a continuación, lanzamos la siguiente ventana de la aplicación, que sería TestActivity, ademas pasamos una variable llamada “tipo” con el valor de “externa”.

```
Intent intent = new Intent(ChooseActivity.this, TestActivity.class);
intent.putExtra("tipo", "externa");
startActivity(intent);
```

Activity Test

TestActivity.java

```
package com.example.alfredo.myintelligenttest;

import ...

public class TestActivity extends AppCompatActivity {

    //Creación de Objetos

    TextView pregunta;
    Button botona, botonb, botonc, botond;
    ImageView imagen;

    private String mAnswer;
    private Questions preguntas = new Questions();
    private Questions preguntas2 = new Questions();
    private Questions preguntas3 = new Questions();
    private int mQuestionNumber = 0;
    private int mQuestionNumber1 = 0;
    private int mQuestionNumber2 = 15;
    private int mQuestionNumber3 = 25;
    public int mScore = 0;
    private int mBlockScore = 0;
    private int mBlockNumber = 0;
    private int Level = 1;
    private String tipo;
    private List<String[]> list;

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

        String tipo = getIntent().getStringExtra( name: "tipo");

        String next[] = {};
        list = new ArrayList<String[]>();
        try {
            CSVReader reader = null;
            if (tipo.equalsIgnoreCase( anotherString: "interna")) {
                reader = new CSVReader(new InputStreamReader(getAssets().open( fileName: "excel.csv")))
            } else {
                FileInputStream fis = openFileInput( name: "excel_externo.csv");
                reader = new CSVReader(new InputStreamReader(fis));
            }

            //in open();
            for(;;) {
                next = reader.readNext();
                if(next != null) {
                    list.add(next);
                } else {
                    break;
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }


        //Enlazado de mi objeto con .java
        pregunta = (TextView) findViewById(R.id.pregunta);
        botona = (Button) findViewById(R.id.botona);
        botonb = (Button) findViewById(R.id.botonb);
        botonc = (Button) findViewById(R.id.botonc);
```

```
100         botond = (Button) findViewById(R.id.botond);
101         imagen = (ImageView) findViewById(R.id.imagen);
102
103         updateQuestion();
104
105         //Botón A
106
107         botona.setOnClickListener((view) → {
110                 String textobotona = botona.getText().toString();
111                     if (textobotona.equalsIgnoreCase(mAnswer)) {
112                     mScore = mScore + 1;
113                     mBlockScore = mBlockScore + 1;
114                 }
115
116                 if (mQuestionNumber == 14){
117                     Intent intent = new Intent ( packageContext: TestActivity.this,ScoreActivity.class);
118                     intent.putExtra( name: "finalScore", mScore);
119                     intent.putExtra( name: "finalLevel", Level);
120                     startActivity(intent);
121                 }
122                 if (mBlockNumber == 5 && mBlockScore <= 2) {
123                     if (Level==2 || Level==3){Level = Level-1; }
124                     resetBlock();
125                 }
126
127                 if (mBlockNumber == 5 && mBlockScore >= 4) {
128                     Level = Level+ 1;
129                     resetBlock();
130                 }
131                 if (Level == 1) {updateQuestion();}
132                 if (Level == 2) {updateQuestion2();}
133                 if (Level == 3) {updateQuestion3();}
134         });
136         //Final Botón A
137
138
139         //Botón B
140
141         botonb.setOnClickListener((view) → {
144                 String textobotonb = botonb.getText().toString();
145                 if (textobotonb.equalsIgnoreCase(mAnswer)) {
146                     mScore = mScore + 1;
147                     mBlockScore = mBlockScore + 1;
148                 }
149
150                 if (mQuestionNumber == 14){
151                     Intent intent = new Intent ( packageContext: TestActivity.this,ScoreActivity.class);
152                     intent.putExtra( name: "finalScore", mScore);
153                     intent.putExtra( name: "finalLevel", Level);
154                     startActivity(intent);
155                 }
156                 if (mBlockNumber == 5 && mBlockScore <= 2) {
157                     if (Level==2 || Level==3){Level = Level-1; }
158                     resetBlock();
159                 }
160
161                 if (mBlockNumber == 5 && mBlockScore >= 4) {
162                     Level = Level+ 1;
163                     resetBlock();
164                 }
165                 if (Level == 1) {updateQuestion();}
166                 if (Level == 2) {updateQuestion2();}
167                 if (Level == 3) {updateQuestion3();}
168
169         });
171         //Final Botón B
```

```
172
173         //Botón C
174
175         botonc.setOnClickListener((view) → {
178                 String textobotonc = botonc.getText().toString();
179                 if (textobotonc.equalsIgnoreCase(mAnswer)) {
180                     mScore = mScore + 1;
181                     mBlockScore = mBlockScore + 1;
182                 }
183
184                 if (mQuestionNumber == 14){
185                     Intent intent = new Intent ( packageContext: TestActivity.this,ScoreActivity.class);
186                     intent.putExtra( name: "finalScore", mScore);
187                     intent.putExtra( name: "finalLevel", Level);
188                     startActivity(intent);
189                 }
190                 if (mBlockNumber == 5 && mBlockScore <= 2) {
191                     if (Level==2 || Level==3){Level = Level-1; }
192                     resetBlock();
193                 }
194
195                 if (mBlockNumber == 5 && mBlockScore >= 4) {
196                     Level = Level+ 1;
197                     resetBlock();
198                 }
199                 if (Level == 1) {updateQuestion();}
200                 if (Level == 2) {updateQuestion2();}
201                 if (Level == 3) {updateQuestion3();}
202         });
204         //Final Botón C
205
206         //Botón D
207
208         botond.setOnClickListener((view) → {
211                 String textobotond = botond.getText().toString();
212                 if (textobotond.equalsIgnoreCase(mAnswer)) {
213                     mScore = mScore + 1;
214                     mBlockScore = mBlockScore + 1;
215                 }
216
217                 if (mQuestionNumber == 14){
218                     Intent intent = new Intent ( packageContext: TestActivity.this,ScoreActivity.class);
219                     intent.putExtra( name: "finalScore", mScore);
220                     intent.putExtra( name: "finalLevel", Level);
221                     startActivity(intent);
222                 }
223                 if (mBlockNumber == 5 && mBlockScore <= 2) {
224                     if (Level==2 || Level==3){Level = Level-1; }
225                     resetBlock();
226                 }
227
228                 if (mBlockNumber == 5 && mBlockScore >= 4) {
229                     Level = Level+ 1;
230                     resetBlock();
231                 }
232                 if (Level == 1) {updateQuestion();}
233                 if (Level == 2) {updateQuestion2();}
234                 if (Level == 3) {updateQuestion3();}
235         });
237         //Final Botón D
238
239     }
240
```

```
241     private void updateQuestion() {
242
243         pregunta.setText(list.get(mQuestionNumber1)[0]);
244         botona.setText(list.get(mQuestionNumber1)[1]);
245         botonb.setText(list.get(mQuestionNumber1)[2]);
246         botonc.setText(list.get(mQuestionNumber1)[3]);
247         botond.setText(list.get(mQuestionNumber1)[4]);
248         mAnswer = list.get(mQuestionNumber1)[5];
249         byte[] decodedString = Base64.decode(list.get(mQuestionNumber1)[6], Base64.DEFAULT);
250         Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,  offset: 0, decodedString.length);
251         imagen.setImageBitmap(decodedByte);
252
253         mBlockNumber++;
254         mQuestionNumber++;
255         mQuestionNumber1++;
256     }
257
258     private void updateQuestion2() {
259         pregunta.setText(list.get(mQuestionNumber2)[0]);
260         botona.setText(list.get(mQuestionNumber2)[1]);
261         botonb.setText(list.get(mQuestionNumber2)[2]);
262         botonc.setText(list.get(mQuestionNumber2)[3]);
263         botond.setText(list.get(mQuestionNumber2)[4]);
264         mAnswer = list.get(mQuestionNumber2)[5];
265
266         byte[] decodedString = Base64.decode(list.get(mQuestionNumber2)[6], Base64.DEFAULT);
267         Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,  offset: 0, decodedString.length);
268         imagen.setImageBitmap(decodedByte);
269         mBlockNumber++;
270         mQuestionNumber++;
271         mQuestionNumber2++;
272     }
273     private void updateQuestion3() {
274         pregunta.setText(list.get(mQuestionNumber3)[0]);
275         botona.setText(list.get(mQuestionNumber3)[1]);
276         botonb.setText(list.get(mQuestionNumber3)[2]);
277         botonc.setText(list.get(mQuestionNumber3)[3]);
278         botond.setText(list.get(mQuestionNumber3)[4]);
279         mAnswer = list.get(mQuestionNumber3)[5];
280         byte[] decodedString = Base64.decode(list.get(mQuestionNumber3)[6], Base64.DEFAULT);
281         Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,  offset: 0, decodedString.length);
282         imagen.setImageBitmap(decodedByte);
283
284         mBlockNumber++;
285         mQuestionNumber++;
286         mQuestionNumber3++;
287     }
288
289     public void resetBlock(){
290         mBlockNumber = 0;
291         mBlockScore = 0;
292     }
293
294 }
```

Ilustración 18: Código del TestActivity.java

Para empezar lo primero que hacemos en este .java es definir las variables con las que vamos a trabajar:

```
TextView pregunta;
Button botona, botonb, botonc, botond;
ImageView imagen;

private String mAnswer;
private int mQuestionNumber = 0;
private int mQuestionNumber1 = 0;
private int mQuestionNumber2 = 15;
private int mQuestionNumber3 = 25;
public int mScore = 0;
private int mBlockScore = 0;
private int mBlockNumber = 0;
private int Level = 1;
private String tipo;
private List<String[]> list;
```

En este caso tenemos variables de muchos tipos, las variables internas son muy importantes ya que con ellas controlamos las preguntas que deben aparecer y vamos actualizando la puntuación que obtenemos, por ejemplo:

- mScore: esta variable controla la puntuación global del test, sin hacer distinción entre el nivel de las preguntas
- mBlockScore: esta variable lleva la puntuación de los "bloques" de preguntas y cada vez que pasamos de un bloque a otro la cuenta se reinicia, al terminar el bloque se tiene en cuenta la puntuación obtenida para modificar el nivel de las preguntas.
- Level: esta variable es por la que se rige la dificultad de las preguntas, ya que a cada valor entre 1 y 3 hay un método diferente de actualizar las preguntas.

Por otro lado, también tenemos otras dos variables fundamentales la variable tipo String (cadena de valores) "tipo" y la variable tipo, List<String[]> "list". La primera la utiliza el programa para saber si el test se ha abierto desde la pantalla principal (main activity) o desde la pantalla de personalizar el test (choose activity) dependiendo de esto el test funcionará de maneras distintas. La variable tipo List crea una matriz donde se van guardando los datos que lee de nuestro archivo .csv.

```
String tipo = getIntent().getStringExtra("tipo");
```

Con esta línea de código obtenemos el valor de la variable que hemos pasado desde el activity anterior.

```
String next[] = {};
list = new ArrayList<String[]>();
try {
    CSVReader reader = null;
    if (tipo.equalsIgnoreCase("interna")) {
        reader = new CSVReader(new
InputStreamReader(getAssets().open("excel.csv")));
    } else {
        FileInputStream fis = openFileInput("excel_externo.csv");
        reader = new CSVReader(new InputStreamReader(fis));
    }

    //in open();
    for(;;) {
        next = reader.readNext();
        if(next != null) {
            list.add(next);
        } else {
            break;
        }
    }
} catch (IOException e) {
    e.printStackTrace();
}
```

En esta parte del código generamos valores dentro de nuestra variable “list” a partir de nuestro archivo .csv. Es decir, necesitamos leer archivos con valores separados por comas (CSV, de su nombre en inglés: Comma Separated Values).

Esto lo podemos hacer nativamente en Java usando las API de lectura de archivos y procesando cada línea con código específico para separar los valores o para crear las líneas correspondientes. Pero hacerlo así es un trabajo muy ingrato, propenso a errores y que tiene poco sentido si podemos hacerlo de una manera mejor y más directa.

Esto precisamente es lo que nos proporciona la conocida biblioteca Open Source llamada opencsv. Por eso utilizamos la instrucción CSVReader.

También se puede apreciar una estructura condicional de tipo “if else” esto lo utilizamos para que dependiendo del valor de la variable “tipo” lea el archivo que corresponda y la lista se rellene con las preguntas que corresponden, ya sean, las precargadas en la app o las que descargue desde internet.

Aquí tenemos el enlazado de nuestros elementos de la interfaz con las variables en .java:

```
//Enlazado de mi objeto con .java
pregunta = (TextView) findViewById(R.id.pregunta);
botona = (Button) findViewById(R.id.botona);
botonb = (Button) findViewById(R.id.botonb);
botonc = (Button) findViewById(R.id.botonc);
botond = (Button) findViewById(R.id.botond);
imagen = (ImageView) findViewById(R.id.imagen);
```

A continuación, tenemos el método updateQuestion:

```
updateQuestion();
```

Este método sirve para actualizar las preguntas, lo ponemos dentro de onCreate para que cuando aparezca el .xml activity_test ya se puedan visualizar preguntas en él.

```
//Botón A

botona.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String textobotona = botona.getText().toString();
            if (textobotona.equalsIgnoreCase(mAnswer)) {
            mScore = mScore + 1;
            mBlockScore = mBlockScore + 1;
        }

        if (mQuestionNumber == 14){
            Intent intent = new Intent
(TestActivity.this,ScoreActivity.class);
            intent.putExtra("finalScore", mScore);
            intent.putExtra("finalLevel", Level);
            startActivity(intent);
        }
        if (mBlockNumber == 5 && mBlockScore <= 2) {
            if (Level==2 || Level==3){Level = Level-1; }
            resetBlock();
        }

        if (mBlockNumber == 5 && mBlockScore >= 4) {
            Level = Level+ 1;
            resetBlock();
        }
        if (Level == 1) {updateQuestion();}
        if (Level == 2) {updateQuestion2();}
        if (Level == 3) {updateQuestion3();}
    }
});
```

Este fragmento de código recoge la lógica que sigue el test cuando se pulsa el botón A, pero es idéntico para las otras opciones (B, C y D) por lo tanto solo lo explicaré para este caso en particular.

El primer condicional que nos encontramos (estructura con if) viene a significar que si el texto del botón A es igual a la respuesta hay que sumar una unidad a la puntuación global del test y otra unidad a la puntuación de bloque

En el segundo if es el que finaliza la app, cuando nuestro contador de preguntas llega a 14 significa que ya han aparecido las 15 preguntas (este desfase se produce porque el contador empieza en 0 y en 0 ya aparece una pregunta, si llegásemos hasta 15 en el contador de preguntas en realidad aparecerían 16 preguntas en nuestro test). Cuando esto ocurre finalizamos el activity actual y pasamos al Activity encargado de la pantalla final de puntuaciones “ScoreActivity”.

El tercer y cuarto if hacen referencia a las puntuaciones de bloque, cuando el número de bloque llega a 5 si la puntuación de ese bloque es menor o igual a dos y el nivel de

pregunta es mayor de 1, el nivel de pregunta disminuirá su valor en una unidad. Si, por el contrario, la puntuación de bloque al llegar a 5 es igual o mayor de 4, el nivel de pregunta aumentará en una unidad. Al final de ambos procesos se restaurarán los valores de puntuación de bloque y numero de bloque a sus valores predeterminados, que son 0 en ambos casos.

Por último, tenemos los condicionales que hacen referencia a la actualización de la pregunta. Dependiendo del nivel de pregunta utilizaran diferentes métodos para actualizarse.

```
private void updateQuestion() {

    pregunta.setText(list.get(mQuestionNumber1)[0]);
    botona.setText(list.get(mQuestionNumber1)[1]);
    botonb.setText(list.get(mQuestionNumber1)[2]);
    botonc.setText(list.get(mQuestionNumber1)[3]);
    botond.setText(list.get(mQuestionNumber1)[4]);
    mAnswer = list.get(mQuestionNumber1)[5];
    byte[] decodedString =
Base64.decode(list.get(mQuestionNumber1)[6], Base64.DEFAULT);
    Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,
0, decodedString.length);
    imagen.setImageBitmap(decodedByte);

    mBlockNumber++;
    mQuestionNumber++;
    mQuestionNumber1++;
}
```

Este fragmento de texto hace referencia al modo en que la preguntas se van actualizando, como se puede observar los texto se cogen de la list que antes hemos creado, la variable mQuestionNumber hace referencia a la fila de donde se coge el texto y el número que va entre corchetes [] hace referencia a la comnua de donde se extrae el texto.

Este proceso presenta una particularidad, y es que en esta matriz que solo nos deja introducir texto debemos almacenar también recursos de imágenes, para ello pasamos la imagen a base64 antes de introducirla en la matriz, y una vez extraemos el fragmento de texto que contiene la imagen para introducirla en el imageview la decodificamos otra vez para darle otra vez su formato original.

Esto es lo hacemos con el siguiente fragmento:

```
    byte[] decodedString =
Base64.decode(list.get(mQuestionNumber1)[6], Base64.DEFAULT);
    Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,
0, decodedString.length);
    imagen.setImageBitmap(decodedByte);
```

Ahora bien, ¿qué es lo que cambia si el updateQuestion es para preguntas de otros niveles?

Pues la verdad es que es muy sencillo, solo cambiamos el mQuestionNumber, todas las preguntas de nuestro test se hayan en el mismo Excel, y están ordenadas por grado de dificultad, por lo tanto, lo que cambia entre los métodos de actualización solo son los números de pregunta desde los que empiezan a contar. Lo podemos ver en el ejemplo de updateQuestion2:

```
private void updateQuestion2() {
    pregunta.setText(list.get(mQuestionNumber2)[0]);
    botona.setText(list.get(mQuestionNumber2)[1]);
    botonb.setText(list.get(mQuestionNumber2)[2]);
    botonc.setText(list.get(mQuestionNumber2)[3]);
    botond.setText(list.get(mQuestionNumber2)[4]);
    mAnswer = list.get(mQuestionNumber2)[5];

    byte[] decodedString =
Base64.decode(list.get(mQuestionNumber2)[6], Base64.DEFAULT);
    Bitmap decodedByte = BitmapFactory.decodeByteArray(decodedString,
0, decodedString.length);
    imagen.setImageBitmap(decodedByte);
    mBlockNumber++;
    mQuestionNumber++;
    mQuestionNumber2++;
}
```

El único cambio apreciable es que ahora tenemos mQuestionNumber2 donde antes teníamos mQuestionNumber1.

El valor predeterminado de mQuestionNumber1 es 0 que es desde donde empiezan las preguntas de nivel 1 y el valor predeterminado de mQuestionNumber2 es 15 que es desde conde empiezan las preguntas de nivel 2.

Al tener el test 15 preguntas mQuestionNumber1 jamás llegará a ofrecer preguntas de un nivel diferente al nivel 1 y así está pensado para el resto de mQuestionNumbers.

```
public void resetBlock(){
    mBlockNumber = 0;
    mBlockScore = 0;
}
```

Por último, nos queda por mencionar el método que aplicamos para reiniciar los bloques, este es tan sencillo como darle el valor inicial a las variables que nosotros queremos, en este caso a mBlockNumber y a mBlockScore.

Activity Score

ScoreActivity.java

```java
package com.example.alfredo.myintelligenttest;

import ...

public class ScoreActivity extends AppCompatActivity {

    //Creación de objetos

    Button volver;
    TextView puntuacion;
    ImageView emoji;
    private int Level;
    private int mScore;



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

        Intent intent = getIntent();
        mScore = intent.getIntExtra( name: "finalScore",  defaultValue: 0);
        Level = intent.getIntExtra( name: "finalLevel",  defaultValue: 0);

        volver = (Button) findViewById(R.id.volver);
        puntuacion = (TextView) findViewById(R.id.puntuacion);
        emoji = (ImageView) findViewById(R.id.emoji);

        if (Level == 1) {
            if (mScore >= 8){
                puntuacion.setText("Aprobado");
                emoji.setImageResource(R.mipmap.emojiaprobado);}

                if (mScore <=7)
                puntuacion.setText("Suspenso");
                emoji.setImageResource(R.mipmap.emojisuspenso1);
            }

            if (Level == 2)

            {
                puntuacion.setText("Notable");
                emoji.setImageResource(R.mipmap.emojimatricula);

            }

            if (Level == 3)
            {
                puntuacion.setText("Sobresaliente");
                emoji.setImageResource(R.mipmap.emojisobresaliente);

            }


            volver.setOnClickListener((view) → {
                    Intent intent = new Intent( packageContext: ScoreActivity.this, MainActivity.class);
                    startActivity(intent);

            });

    }

}
```

Ilustración 19: Código del ScoreActivity.java

Para empezar lo primero que hacemos en este .java es definir las variables con las que vamos a trabajar:

```
Button volver;
TextView puntuacion;
ImageView emoji;
private int Level;
private int mScore;
```

En segundo lugar, recuperamos las variables que hemos utilizado en el .java TestActivity y que necesitaremos a la hora de establecer las calificaciones finales:

```
Intent intent = getIntent();
mScore = intent.getIntExtra("finalScore", 0);
Level = intent.getIntExtra("finalLevel", 0);
```

Seguidamente, enlazaremos estas variables con los botones que hemos representado en nuestro archivo .xml Activity_test, en esta operación cobra importancia el atributo id que hemos elegido en el .xml para cada objeto:

```
volver = (Button) findViewById(R.id.volver);
puntuacion = (TextView) findViewById(R.id.puntuacion);
emoji = (ImageView) findViewById(R.id.emoji);
```

En este fragmento de código estableceremos los criterios de calificación por los que se guiará nuestra aplicación, si nos hemos quedado en el nivel 1 al finalizar el test lo que decidirá si estamos aprobados o no será la puntuación global, en nuestra aplicación este valor lo recoge la variable mScore, si hemos acertado más de la mitad de las preguntas obtendremos un aprobado y nos aparecerá su emoji correspondiente. Si, por el contrario, hemos acertado menos de la mitad nuestra calificación será de suspenso y nos saldrá un emoticono “triste”.

```
if (Level == 1) {
    if (mScore >= 8){
        puntuacion.setText("Aprobado");
        emoji.setImageResource(R.mipmap.emojiaprobado);}

    if (mScore <=7)
    puntuacion.setText("Suspenso");
    emoji.setImageResource(R.mipmap.emojisuspenso1);
}
```

Si nos hemos quedado en el nivel 2, nuestro nivel en la materia es de notable.

```
if(Level==2){

{
    puntuacion.setText("Notable");
    emoji.setImageResource(R.mipmap.emojimatricula);

}
```

Finalmente, si hemos alcanzado el nivel 3 nuestra calificación será de sobresaliente.

```
if(Level==3)
{
    puntuacion.setText("Sobresaliente");
    emoji.setImageResource(R.mipmap.emojisobresaliente);

}
```

Al final del archivo ScoreActivity.java tenemos la parte que conecta esta última ventana con la ventana inicial de la aplicación por medio de un botón que pasará de ScoreActivity a MainActivity

```
volver.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(ScoreActivity.this,
MainActivity.class);
        startActivity(intent);

    }
});
```

3.2.3 Layout

En el layout se encuentra la interfaz de la aplicación basada en archivos xml, el usuario accederá a las funcionalidades de la aplicación interactuando con dicha interfaz, por eso es necesario planificarla, para obtener una interfaz plenamente funcional a la vez que sencilla de utilizar para el usuario.

Mi aplicación consta de los siguientes archivos correspondientes al layout:

- Activity_main
- Activity_choose
- Activity_test
- Activity_score

Estas cuatro pantallas y los objetos que aparecen en ellas son todo lo que necesitamos para poder personalizar nuestro propio test, hacer dicho cuestionario y por ultimo recibir una puntuación.

Para que nos sea posible entender el código de estos xml que veremos a continuación vamos a definir rápidamente los elementos más importantes de los que se componen los archivos:

-Linear Layout: Utilizando este comando estamos ordenando todos los elementos que incluyamos dentro del linear layout de forma que queden todos en una única columna.

-Button: Un elemento de la interfaz de usuario que el usuario puede pulsar para realizar una acción.

-TextView: Un elemento de la interfaz de usuario que muestra texto al usuario.

-EditText: Un elemento de la interfaz de usuario que muestra texto que el usuario puede modificar.

-ImageView: Un elemento de la interfaz que sirve para mostrar al usuario imágenes.

Estos elementos pueden ser modificados o personalizados a través de sus distintos atributos.

Atributos más importantes en mi aplicación:

-Tamaño:
Todos los grupos de vistas incluyen un ancho y una altura (layout_width y layout_height), y cada vista debe definirlos. Tenemos dos opciones diferentes para definirlos.
Por un lado, podemos indicar a nuestra vista que modifique su tamaño conforme a los requisitos de un elemento en particular escribiendo wrap_content. Por otro lado, podemos indicar a tu vista que se agrande tanto como lo permita su grupo de vistas principal (en nuestro caso nuestro linear layout) escribiendo match_parent.
También se puede especificar el ancho y la altura con medidas exactas.

-Orientación: Es importante especificarle a tu layout si quieres que los elementos que introduzcas dentro de él los quieres ordenar de manera vertical u horizontal esto se consigue introduciendo android:orientation=”vertical” o “horizontal”

-Gravity: Nos servimos de este atributo para colocar nuestro elemento en el lugar que nosotros le queremos asignar dentro de un elemento contenedor de mayor tamaña que el elemento contenido. Podemos indicar si queremos nuestro elemento en el centro del espacio asignado, en la parte superior, inferior, a la derecha, a la izquierda…etc.

-Android ID: este atributo es muy importante dentro del funcionamiento global de nuestra aplicación pues que sirve para conectar los elementos de la interfaz con el código java que dota de utilidad a dichos elementos, dotando de un id a un botón, por ejemplo, estamos dándole una identidad con la que lo podremos llamar en nuestro java para poder dotarlo de algún tipo de función cuando sea pulsado. Digamos que nuestro id sirve de enlace entre el layout y el java.

Una vez ya puestos en contexto de las particularidades básicas de los layouts veamos de los que se componen cada uno:

Activity_Main:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity"
    android:background="@drawable/enigma"
    tools:layout_editor_absoluteY="25dp">

    <TextView
        android:id="@+id/titulo"
        style="@style/Base.TextAppearance.AppCompat.Widget.ActionBar.Title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:gravity="center"
        android:text="Test Inteligente by Alfredo Fenoll"
        android:textColor="#000080"
        android:textSize="30sp"
        android:textStyle="bold" />

    <ImageButton
        android:id="@+id/comenzar"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:background="@mipmap/startbutton"

        android:layout_gravity="center"
         />


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="2"
        android:orientation="horizontal">

        <Button
            android:layout_width="150sp"
            android:layout_height="40sp"
            android:id="@+id/elegir"
            style="@style/Widget.AppCompat.Button"
            android:layout_gravity="center"
            android:layout_margin="20sp"
            android:textSize="12sp"
            android:text="Elegir Preguntas"/>

        <Button
            android:layout_width="150sp"
            android:layout_height="40sp"
            android:layout_gravity="center"
            style="@style/Widget.AppCompat.Button"
            android:text="Ayuda"
            android:textSize="12sp" />


    </LinearLayout>

</LinearLayout>
```

Preview

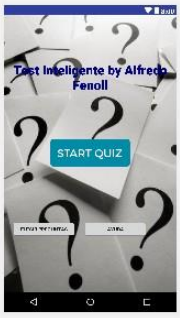
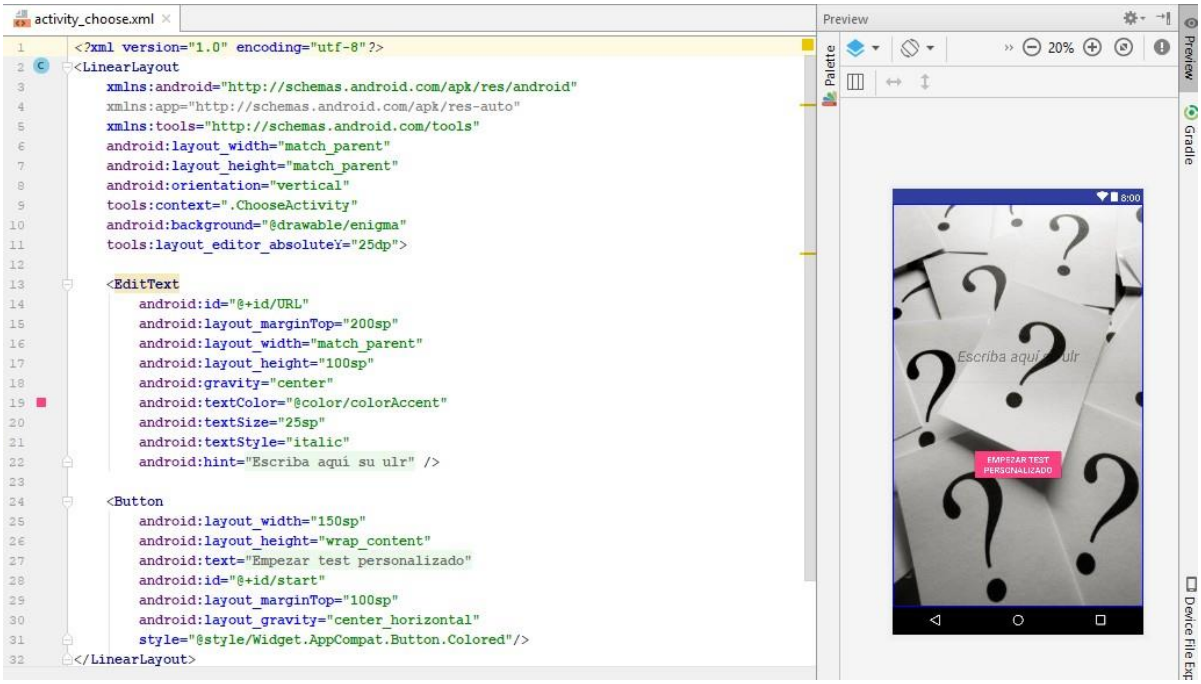


Ilustración 20: Código del activity_main.xml

Activity Choose:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ChooseActivity"
    android:background="@drawable/enigma"
    tools:layout_editor_absoluteY="25dp">

    <EditText
        android:id="@+id/URL"
        android:layout_marginTop="200sp"
        android:layout_width="match_parent"
        android:layout_height="100sp"
        android:gravity="center"
        android:textColor="@color/colorAccent"
        android:textSize="25sp"
        android:textStyle="italic"
        android:hint="Escriba aquí su ulr" />

    <Button
        android:layout_width="150sp"
        android:layout_height="wrap_content"
        android:text="Empezar test personalizado"
        android:id="@+id/start"
        android:layout_marginTop="100sp"
        android:layout_gravity="center_horizontal"
        style="@style/Widget.AppCompat.Button.Colored"/>
</LinearLayout>
```

Ilustración 21: Código del activity_choose.xml

Activity_Test:

activity_test.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:background="@drawable/degradacion"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".TestActivity">

    <TextView
        android:id="@+id/pregunta"
        android:layout_width="wrap_content"
        android:layout_height="100sp"
        android:layout_margin="20sp"
        android:text="¿En que año se descubrió América?"
        android:layout_gravity="center"
        android:gravity="center"
        android:textSize="25sp"
        android:layout_weight="1"/>

    <LinearLayout
        android:layout_width="200sp"
        android:layout_height="150sp"
        android:layout_gravity="center"
        android:layout_weight="1"
        >

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/imagen" />

    </LinearLayout>


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="300sp"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_marginTop="20sp">


        <Button
            android:id="@+id/botona"
            android:background="@drawable/button_test"
            android:layout_width="300sp"
            android:layout_height="45sp"
            android:layout_margin="15sp"
            android:text="1492"/>


        <Button
            android:id="@+id/botonb"
            android:layout_width="300sp"
            android:layout_height="45sp"
            android:background="@drawable/button_test"
            android:layout_margin="15sp"
            android:text="1313"/>


        <Button
            android:id="@+id/botonc"
            android:layout_width="300sp"
            android:layout_height="45sp"
            android:background="@drawable/button_test"
            android:layout_margin="15sp"
            android:text="2001"/>

        <Button
            android:id="@+id/botond"
            android:layout_width="300sp"
            android:layout_height="45sp"
            android:layout_margin="15sp"
            android:background="@drawable/button_test"
            android:text="1984"/>

    </LinearLayout>

</LinearLayout>
```

Preview

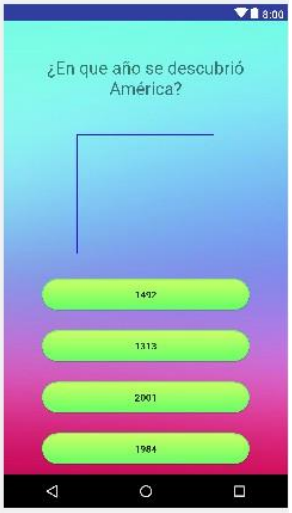
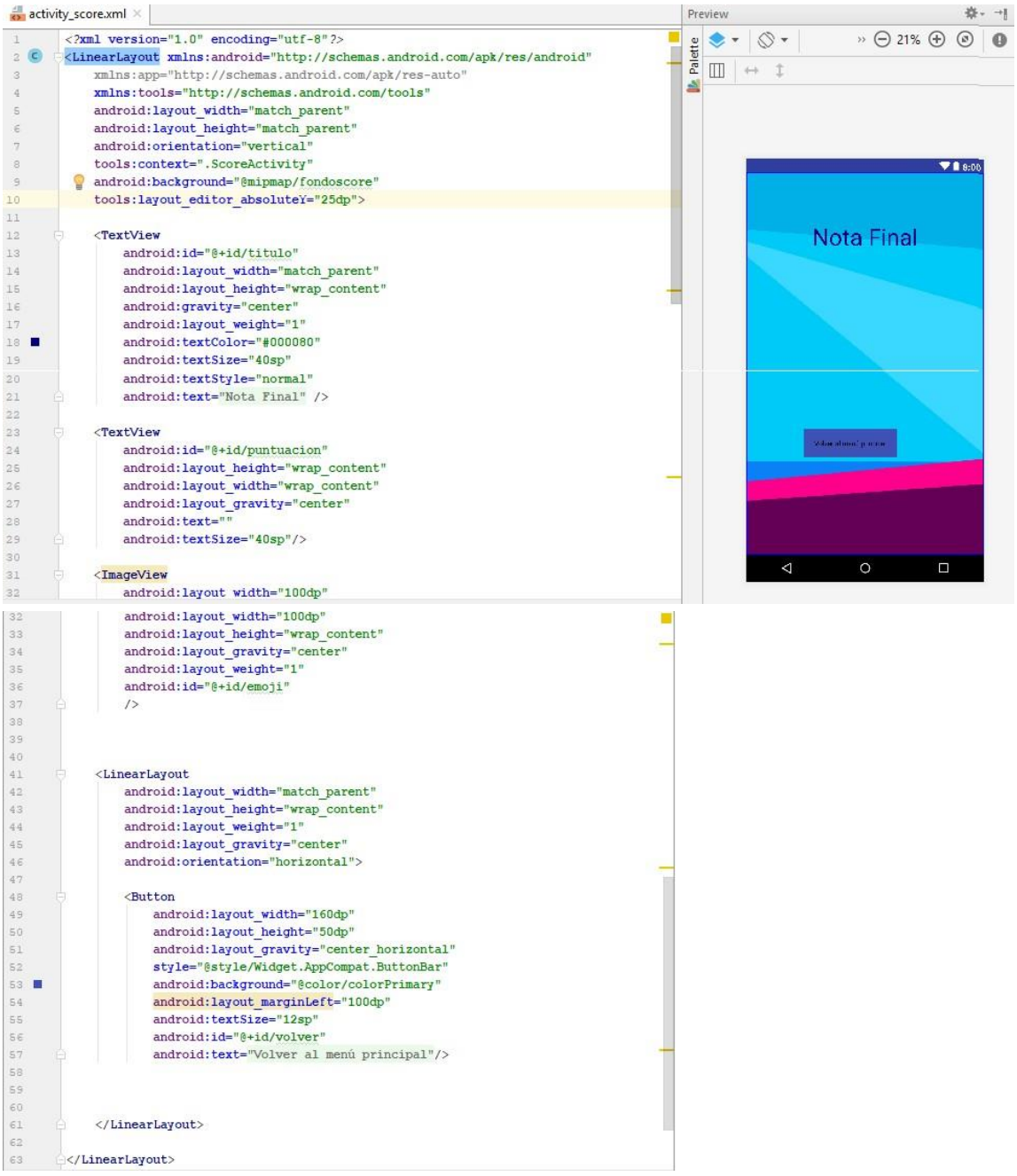


Ilustración 22: Código del activity_test.xml

En este archivo .xml hay una particularidad que procedo a comentar, al no convencerme ninguno de los estilos predeterminados de los botones para los botones para las preguntas creé mi propio botón a partir de un archivo .xml.

Ilustración 23: Código del button_test.xml

Activity_Score:




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ScoreActivity"
    android:background="@mipmap/fondoscore"
    tools:layout_editor_absoluteY="25dp">

    <TextView
        android:id="@+id/titulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:layout_weight="1"
        android:textColor="#000080"
        android:textSize="40sp"
        android:textStyle="normal"
        android:text="Nota Final" />

    <TextView
        android:id="@+id/puntuacion"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_gravity="center"
        android:text=""
        android:textSize="40sp"/>

    <ImageView
        android:layout width="100dp"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_weight="1"
        android:id="@+id/emoji"
        />



    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:layout_gravity="center"
        android:orientation="horizontal">

        <Button
            android:layout_width="160dp"
            android:layout_height="50dp"
            android:layout_gravity="center_horizontal"
            style="@style/Widget.AppCompat.ButtonBar"
            android:background="@color/colorPrimary"
            android:layout_marginLeft="100dp"
            android:textSize="12sp"
            android:id="@+id/volver"
            android:text="Volver al menú principal"/>



    </LinearLayout>

</LinearLayout>
```

Ilustración 24: Código del activity_score.xml

3.2.4 Excel.csv

En este archivo es donde se almacenan las preguntas predeterminadas que aparecen en este test.

La ubicación del archivo dentro de la aplicación es la siguiente:

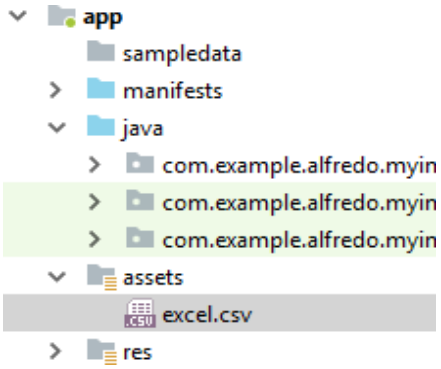


Ilustración 25: Ubicación del Excel.csv

Y el contenido es el siguiente:

excel.csv

```
1  Texto de la Pregunta 1,A R1,B R1,C R1,D R1,A R1,iVBORw0KGgoAAAANSUhEUgAAADEAAADg
2  Texto de la Pregunta 2,A R2,B R2,C R2,D R2,B R2,
3  Texto de la Pregunta 3,A R3,B R3,C R3,D R3,C R3,iVBORw0KGgoAAAANSUhEUgAAADEAAADg
4  Texto de la Pregunta 4,A R4,B R4,C R4,D R4,D R4,
5  Texto de la Pregunta 5,A R5,B R5,C R5,D R5,A R5,
6  Texto de la Pregunta 6,A R6,B R6,C R6,D R6,B R6,
7  Texto de la Pregunta 7,A R7,B R7,C R7,D R7,C R7,
8  Texto de la Pregunta 8,A R8,B R8,C R8,D R8,D R8,
9  Texto de la Pregunta 9,A R9,B R9,C R9,D R9,A R9,
10 Texto de la Pregunta 10,A R10,B R10,C R10,D R10,B R10,
11 Texto de la Pregunta 11,A R11,B R11,C R11,D R11,C R11,
12 Texto de la Pregunta 12,A R12,B R12,C R12,D R12,D R12,
13 Texto de la Pregunta 13,A R13,B R13,C R13,D R13,A R13,
14 Texto de la Pregunta 14,A R14,B R14,C R14,D R14,B R14,
15 Texto de la Pregunta 15,A R15,B R15,C R15,D R15,C R15,
16 Texto de la Pregunta 16,A R15,B R15,C R15,D R15,C R15,
17 Texto de la Pregunta 17,A R15,B R15,C R15,D R15,C R15,
18 Texto de la Pregunta 18,A R15,B R15,C R15,D R15,C R15,
19 Texto de la Pregunta 19,A R15,B R15,C R15,D R15,C R15,
20 Texto de la Pregunta 20,A R15,B R15,C R15,D R15,C R15,
21 Texto de la Pregunta 21,A R15,B R15,C R15,D R15,C R15,
22 Texto de la Pregunta 22,A R15,B R15,C R15,D R15,C R15,
23 Texto de la Pregunta 23,A R15,B R15,C R15,D R15,C R15,
24 Texto de la Pregunta 24,A R15,B R15,C R15,D R15,C R15,
25 Texto de la Pregunta 25,A R15,B R15,C R15,D R15,C R15,
26 Texto de la Pregunta 26,A R15,B R15,C R15,D R15,C R15,
27 Texto de la Pregunta 27,A R15,B R15,C R15,D R15,C R15,
28 Texto de la Pregunta 28,A R15,B R15,C R15,D R15,C R15,
29 Texto de la Pregunta 29,A R15,B R15,C R15,D R15,C R15,
30 Texto de la Pregunta 30,A R15,B R15,C R15,D R15,C R15,
31 Texto de la Pregunta 31,A R15,B R15,C R15,D R15,C R15,
32 Texto de la Pregunta 32,A R15,B R15,C R15,D R15,C R15,
33 Texto de la Pregunta 33,A R15,B R15,C R15,D R15,C R15,
```

Ilustración 26: Contenido del archivo Excel.csv

El contenido de la imagen son las preguntas que he utilizado para ir comprobando que aparecían en el orden correcto, por lo tanto, podrían cambiar en la presentación, aunque el cambio sería meramente estético.

Lo que no podría cambiar sería el orden en las que parece la información en el archivo ya que esto es clave para que luego la aplicación pueda mostrar las preguntas en el orden correcto.

Las preguntas se organizan por filas, en cada fila vendría una pregunta diferente, y los diferentes elementos que componen cada pregunta van organizados en columnas de la siguiente manera:

-En la primera columna tendríamos el texto de las preguntas.

-En la segunda columna tendríamos el texto de la opción A.

-En la tercera columna tendríamos el texto de la opción B.

-En la cuarta columna tendríamos el texto de la opción C.

-En la quinta columna tendríamos el texto de la opción D.

-En la sexta columna tendríamos el texto de la opción correcta.

-En la séptima columna tendríamos la imagen correspondiente en Base64.

4. Resultados y discusión

Actualmente considero muy pobres los conocimientos de programación que se imparten nuestro grado, particularmente, solo en la asignatura de primer año recibí formación en dicho campo.

Creo que queda camino por recorrer y mejoras que aplicar en nuestro grado. Por ello en este apartado de resultados, más allá de la implementación en las funcionalidades de mi aplicación, quiero destacar como han mejorado mis conocimientos y mis habilidades en el campo de la programación.

Durante el desarrollo de mi app, me he visto forzado a dominar los conceptos básicos de programación en java, uno de los lenguajes más extendidos y que estoy seguro que será un gran complemento en el futuro de cara a mis habilidades profesionales.

Aunque considero más importantes mis avances en materia de aprendizaje en el desarrollo de aplicaciones también quiero comentar los resultados obtenidos en cuanto a funcionalidades de la aplicación, aunque he de remarcar que es difícil comparar el nivel de complejidad de mi app con otros test de dificultad adaptable, ya que no he podido encontrar ninguno.

Tampoco puedo compararlo directamente con el test de nivel de la página de Cambridge, ya que este tiene una duración mucho más extensa y un repertorio de preguntas mucho más amplio, aunque indudablemente su lógica a la hora de seleccionar que preguntas debe mostrar tiene que ser mucho más compleja.

Es razonable considerar que esto test que utilizan grandes empresas están a otro nivel, incluso empiezan a aparecer test con inteligencias artificiales avanzadas, recientemente el Instituto Tecnológico de Massachusetts (MIT) ha desarrollado junto a ABA English, una escuela de inglés online, una aplicación que funciona con Alexa, asistente virtual desarrollado por Amazon, y que mediante una conversación por voz determina tu nivel de inglés.

Por último, mencionar que esta memoria también tiene un valor académico, la he redactado explicando con detalle mi código para que otros alumnos de esta universidad o cualquier aficionado a la programación pueda aprovechar todo lo que yo he aprendido de la forma más efectiva posible.

5. Conclusiones

Al hilo de lo comentado en el apartado anterior podríamos concluir con la siguiente reflexión, a nivel académico este proyecto ha sido un éxito, durante el desarrollo de esta aplicación he tenido que superar problemas de distinta naturaleza (problemas lógicos, bases de datos, apps que se sirven de recursos de internet…) que me han ayudado a adquirir un conocimiento bastante amplio de la herramienta de programación Android Studio.

En contraposición nos encontramos con una app muy sencilla que intenta realizar una tarea muy compleja, crear un test verdaderamente inteligente es un proyecto de gran envergadura que requiere de una cantidad de recursos tanto a nivel de conocimientos como a niveles técnicos que se escapa del alcance de lo que podría ser un proyecto de este nivel.

Aun así, me gustaría redactar un último apartado dando salida a ideas que se me han quedado en el tintero y que, quizás alguien en un futuro pueda llevar a la práctica en un nuevo trabajo de fin de grado.

5.1 Propuestas de mejora

Hablar de propuestas de mejora es un apartado que me motiva, sería muy satisfactorio para mí ve en unos años como otros alumnos de la carrera han mejorado esta aplicación.

Por lo tanto, voy a plantear una serie de mejoras que podrán llevar a mi aplicación un poco más lejos:

-Randomizar las preguntas que aparecen. Actualmente las preguntas siempre aparecen en el mismo orden, sería interesante alterar el orden de aparición de las preguntas de forma que sí pareciesen en el nivel de dificultad correspondiente, pero dentro de este nivel de una manera aleatoria.

Este cambio, aunque a simple vista resulte sencillo, de la forma que está estructurado el código de mi app se vuelve una tarea compleja.

-También se podría cambiar el método de introducción de las preguntas en la app, pudiéndolas introducir desde la propia aplicación.

-Por otro lado, con el objetivo de hace el test todavía más personalizable, se podría implementar un sistema que permitiese alterar el número de preguntas del test a nuestro gusto. Este apartado también requiere de grandes a cambios estructurales de mi código.

Combinando todas estas mejoras tendríamos una app muy completa y estoy seguro de que es trabajo suficiente como para plantear otro trabajo de Fin de Grado.

BIBLIOGRAFÍA

Recopilación de las fuentes de información utilizadas en este proyecto:

Cambridge English.
[1] URL: https://www.cambridgeenglish.org/es/test-your-english/

Wikipedia. La enciclopedia libre.
[2] URL: https://es.wikipedia.org/wiki/Android

ComputerHoy: Web de actualidad tecnológica.
[3] URL: https://computerhoy.com/reportajes/industria/android-vs-iphone-guerra-smartphones-cifras-271447

Wikipedia. La enciclopedia libre.
[4] URL: https://es.wikipedia.org/wiki/Android_Studio

Andr4all: Web especializada en noticias de Android.
[5] URL: https://andro4all.com/2018/01/distribucion-android-2018

Android Developers. Web oficial de Android para desarrolladores.
[6] URL: https://developer.android.com/studio/intro/?hl=es-419

Xataka: Web de aficionados a la tecnología.
[7] URL: https://www.xataka.com/aplicaciones/segun-este-test-de-ingles-de-alexa-y-el-mit-tengo-un-nivel-advanced-y-cambridge-dice-que-es-verdad

Wikipedia. La enciclopedia libre.
[8] URL: https://es.wikipedia.org/wiki/Amazon_Alexa

Android Developers. Web oficial de Android para desarrolladores.
[9] URL: https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419

Wikipedia. La enciclopedia libre.
[10] URL: https://es.wikipedia.org/wiki/Android_Pie