UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES COMBINACIONALES BAJO ANDROID

JUAN BURGOS ABADIE

SEPTIEMBRE 2015

APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES COMBINACIONALES BAJO ANDROID

AUTOR: Juan Burgos Abadie
TUTOR: Federico García Salzmann
PONENTE: Eduardo Boemo Scalvinoni

Digital System Laboratory
Dpto. Tecnología Electrónica y de Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Septiembre de 2015

Agradecimientos

En primer lugar me gustaría agradecer a mi familia, especialmente a mi madre Carmen, que con su paciencia y apoyo constantes, han conseguido motivarme hasta finalizar, por fin, esta odisea que ha sido la carrera.

También me gustaría agradecer a todos los compañeros que he tenido a lo largo de estos numerosos años de estancia en la EPS, por todos los momentos compartidos tanto en el aula, el laboratorio o fuera de la escuela.

Muchas gracias a todos.

RESUMEN

Este proyecto consiste en el desarrollo de una aplicación para teléfonos móviles y tabletas (app) Android en el que se incluirán un tutorial, ejercicios, preguntas de test y funcionalidad para envío de soluciones por correo electrónico. El contenido teórico de los problemas, el tutorial y el test has sido seleccionado de guía de problemas de la asignatura Dispositivos Integrados Especializados impartida en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.

El fin del proyecto es proveer a los alumnos de una herramienta nueva y útil para la adquisición de conocimientos sobre DFT ad-hoc utilizando técnicas de stuck-at y la puesta en práctica de los mismos. La aplicación también permitirá compartir sus resultados con otros compañeros.

La aplicación se ha desarrollado en inglés y se encuentra publicada en Google Play.

Palabras clave

Smartphone, app, Android, móvil, tableta, stuck-at, DFT.

ABSTRACT

This project consists of the development of an Android mobile and tablet application (app) which will include a tutorial, exercises, a quiz and functionality for sending results via e-mail. The theoretical content of the exercises, the tutorial and the quiz has been selected from the exercise guide of the course Dispositivos Integrados Especializados imparted in the Escuela Politécnica Superior of the Universidad Autónoma de Madrid.

The goal of the project is to provide the students with a new and useful tool for the acquisition and implementation of knowledge about DFT ad-hoc using stuck-at techniques as well as to allow them to send their results with their colleagues.

The application has been developed in English and is published in Google Play.

Keywords

Smartphone, app, Android, mobile, tablet, stuck-at, DFT.

INDICE DE CONTENIDOS

1 Introducción 1
1.1 Motivación 1
1.2 Objetivos 1
1.3 Organización de la memoria 2
2 Estado del arte 3
2.1 Evolución de la telefonía móvil 3
2.2 Sistemas operativos para smartphones 4
2.2.1 Android 4
2.2.2 iOS 7
2.2.3 Elección del sistema operativo 8
2.2.3.1 Penetración de los smartphones en España 8
2.2.3.2 Cuota de mercado según el Sistema Operativo 9
2.2.3.3 Conclusión 10
2.3 Aplicaciones similares 10
2.3.1 Aplicaciones DSLab UAM 10
3 Diseño 11
3.1 Requisitos 11
3.1.1 Versión de Android 11
3.1.2 Tipos de pantalla 12
3.1.2.1 Tamaños de pantalla 13
3.1.2.2 Densidad de pantalla 14
3.1.2.3 Elección de tipos de pantalla soportados 15
3.1.3 Formato 15
3.1.3.1 Idioma 16
3.1.3.2 Colores 16
3.1.3.3 Tamaño de letra 16
3.1.3.4 Formato de las vistas 17
3.1.4 Resumen de requisitos 17
3.2 Herramientas 17
3.2.1 Eclipse 18
3.2.2 Android Studio 18
3.2.3 Elección de entorno de desarrollo 19
3.3 Módulos de la aplicación 19
3.3.1 Tutorial 19
3.3.2 Ayuda 19
3.3.3 Test 20
3.3.4 Ejercicios 20
3.3.4.1 Esquema 20
3.3.4.2 Guardar 21
3.3.4.3 Comprobación 21
3.3.4.4 Resolver 21
3.3.4.5 Reset 21
3.3.5 Envío por correo 21
3.4 Métodos de obtención de datos 21
4 Desarrollo 23
4.1 Material utilizado 23
4.2 Bloques básicos de Android 23
4.2.1 Activity 23
4.2.2 Fragment 25

4.2.3 Widgets ... 26
4.2.3.1 ViewPager ... 26
4.2.3.2 Tabs ... 27
4.2.3.3 ToggleButton ... 27
4.2.3.4 Dialogs ... 27
4.2.4 Layouts ... 28
4.2.5 Drawables ... 28
4.2.6 Assets ... 28
4.2.7 Values ... 29
4.2.8 Archivo Manifest ... 29
4.2.9 Librerías de apoyo ... 30
4.3 Estructura de la aplicación ... 31
4.4 Descripción de las clases ... 33
4.4.1 MenuInicio ... 33
4.4.2 DialogoEnviar ... 34
4.4.3 MenuEjercicios ... 35
4.4.4 EnunciadoEjercicio ... 36
4.4.5 DialogoCargar ... 37
4.4.6 EjercicioTabla ... 38
4.4.7 VistaEjercicioTabla ... 41
4.4.8 Esquema ... 42
4.4.9 DialogoGuardar ... 43
4.4.10 DialogoSobreescibir ... 44
4.4.11 DialogoReset ... 45
4.4.12 DialogoSinCambios ... 46
4.4.13 Test ... 47
4.4.14 VistaTest ... 48
4.4.15 RecuentoTest ... 49
4.4.16 Tutorial ... 50
4.4.17 PaginaTutorial ... 51
4.4.18 Ayuda ... 52
4.4.19 ListaAyuda ... 53
4.4.20 DetalleAyuda ... 54
4.4.21 AcercaDe ... 55
4.4.22 DBHelper ... 55
4.4.23 Ejercicio ... 56
4.4.24 Pregunta ... 56
4.4.25 Partida ... 56
4.5 Estructura de la base de datos ... 57
4.5.1 Tabla de preguntas ... 57
4.5.2 Tabla de ejercicios ... 57
5 Integración, pruebas y resultados ... 59
5.1 Pruebas realizadas ... 59
5.1.1 AVD Manager ... 59
5.1.1.1 Creación de un dispositivo virtual ... 59
6 Conclusiones y trabajo futuro ... 63
6.1 Conclusiones ... 63
6.2 Trabajo futuro ... 63
Referencias ... 64
Glosario ... - 65 -

Anexos ..... - 66 -
A PRESUPUESTO ..... - 66 -
B PLIEGO DE CONDICIONES ..... - 67 -

INDICE DE FIGURAS

FIGURA 2-1: SCR-194 3

FIGURA 2-2: ARQUITECTURA DE ANDROID 7

FIGURA 2-3: PENETRACIÓN SMARTPHONES EN ESPAÑA 8

FIGURA 2-4: CUOTA DE MERCADO POR SO 9

FIGURA 3-1: CUOTA DE DISPOSITIVOS POR VERSIÓN DE ANDROID 12

FIGURA 3-2: GRUPOS DE TAMAÑO Y DENSIDAD DE PANTALLA 13

FIGURA 3-3: DISTRIBUCIÓN POR TAMAÑOS DE PANTALLA 14

FIGURA 3-4: DISTRIBUCIÓN POR DENSIDADES DE PANTALLA 15

FIGURA 4-1: CICLO DE VIDA DE UNA ACTIVIDAD 24

FIGURA 4-2: CICLO DE VIDA DE UN FRAGMENTO 25

FIGURA 4-3: ESTRUCTURA DE LA APLICACIÓN 31

FIGURA 4-4: DISEÑO DE VISTA DE MENUINICIO 33

FIGURA 4-5: DISEÑO DE VISTA DE DIALOGOENVIAR 34

FIGURA 4-6: DISEÑO DE VISTA DE MENUEJERCICIOS 35

FIGURA 4-7: DISEÑO DE VISTA DE ENUNCIADOEJERCICIO 36

FIGURA 4-8: DISEÑO DE VISTA DE DIALOGOCARGAR 37

FIGURA 4-9: DISEÑO DE VISTA DE EJERCICIOTABLA 40

FIGURA 4-10: DISEÑO DE VISTA DE VISTAEJERCICIO 41

FIGURA 4-11: DISEÑO DE VISTA DE ESQUEMA 42

FIGURA 4-12: DISEÑO DE VISTA DE DIALOGOGUARDAR 43

FIGURA 4-13: DISEÑO DE VISTA DE DIALOGOSOBREESCRIBIR 44

FIGURA 4-14: DISEÑO DE VISTA DE DIALOGORESET 45

FIGURA 4-15: DISEÑO DE VISTA DE DIALOGOSINCAMBIOS 46

FIGURA 4-16: DISEÑO DE VISTA DE TEST 47

FIGURA 4-17: DISEÑO DE VISTA DE VISTATEST 48
FIGURA 4-18: DISEÑO DE VISTA DE RECUENTOTEST 49
FIGURA 4-19: DISEÑO DE VISTA DE TUTORIAL 50
FIGURA 4-20: DISEÑO DE VISTA DE PAGINATUTORIAL 51
FIGURA 4-21: DISEÑO DE VISTA DE AYUDA 52
FIGURA 4-22: DISEÑO DE VISTA DE LISTAAYUDA 53
FIGURA 4-23: DISEÑO DE VISTA DE DETALLEAYUDA 54
FIGURA 4-24: DISEÑO DE VISTA DE ACERCADE 55
FIGURA 5-1: AVD MANAGER 59
FIGURA 5-2: CONFIGURACIÓN DE PANTALLA 60
FIGURA 5-3: SELECCIÓN DE API 60
FIGURA 5-4: CONFIGURACIÓN DE DISPOSITIVO VIRTUAL 61
FIGURA 5-5: SELECCIÓN DE DISPOSITIVO 62

INDICE DE TABLAS

Tabla 1-1: Versiones de Android..... 14
Tabla 2.1: Versiones de iOS..... 16
Tabla 2.2: Aplicaciones anteriores..... 18
Tabla 3.1: Distribución de versiones de Android..... 19
Tabla 3.2: Distribución de tamaños de pantalla..... 21
Tabla 3.3: Distribución por densidades de pantalla..... 23
Tabla 3.4: Aplicaciones anteriores..... 24
Tabla 4.1: Estructura de la tabla para las preguntas..... 65
Tabla 4.2: Estructura de la tabla para los ejercicios..... 65

1 Introducción

1.1 Motivación

En la última década los dispositivos móviles, especialmente los smartphones y tablets, han experimentado un auge extraordinario. Hoy en día prácticamente todo el mundo dispone de uno, especialmente la gente joven, debido al éxito en paralelo que han tenido las redes sociales y los distintos servicios relacionados. Las aplicaciones para móviles (apps) se han convertido en un elemento cada vez más presente muchos aspectos de la vida diaria, especialmente para la comunicación y el intercambio de contenidos multimedia.

Este proyecto surge con la intención de aprovechar la penetración de los smartphones y las apps, para ofrecer a los estudiantes de ingeniería una manera alternativa de acceso a los contenidos, tanto teóricos como prácticos, de la asignatura Dispositivos Integrados Especializados.

Para ello se desarrollará una aplicación que permita ofrecer a los alumnos ventajas y funcionalidades que no son posibles con el formato tradicional de guías de problemas. Entre ellas cabría destacar:

- Eliminación del uso del papel.
- Accesibilidad
- Ubicuidad de la información
- Posibilidad de autocorrección
- Ayudas parciales a la resolución de problemas
- Envío de resultados a través del correo electrónico

1.2 Objetivos

El objetivo del PFC es diseñar una aplicación Android gratuita que permita ejercitarse en conceptos básicos de problemas de test de circuitos combinacionales sencillos utilizando la técnica de stuck-at. El objeto es disponer de una herramienta que permita resolver la guía de problemas del cuarto tema de la asignatura Dispositivos Integrados Especializados (DIE) del tercer curso del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid(EPS-UAM). Se ofrecerá una guía de diez problemas de circuitos combinacionales, así como un tutorial y un test.

La aplicación debe seguir estrictamente el estilo de las aplicaciones anteriores realizadas para la asignatura CIRDIG de primer curso del Grado en Ingeniería de Tecnologías y Servicios de Telecomunicaciones de la EPS-UAM. Estas restricciones incluyen taxativamente los aspectos siguientes:

- Tamaño de letra
- Colores
- Formato del menú inicial
- Icono
- Resolución de gráficos
- Relación de aspecto de los gráficos
- Formato de botones
- Menú de ayuda
- Página “about”
- Funcionamiento del tutorial

Esta aplicación debe ser un complemento a la guía de problemas de la asignatura DIE, por lo que en ella se mostrarán los ejercicios que estén también disponibles en la app.

Desde el punto de vista del estudiante de PFC el objetivo es aprender a programar aplicaciones para dispositivos Android y, por consiguiente, lenguaje Java, ya que ninguno de estos temas está cubierto por las asignaturas de la carrera y por lo tanto constituyen una formación adicional.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 1: Introducción:** Explica la motivación y objetivos del proyecto, así como la organización de la memoria.
- **Capítulo 2: Estado del arte:** Describe la evolución y la situación actual del mercado de la telefonía y aplicaciones móviles.
- **Capítulo 3: Diseño del proyecto:** Define con mayor detalle el problema al que nos enfrentamos y expone la forma en la que se va a conseguir solucionarlo.
- **Capítulo 4: Desarrollo del proyecto:** Precisa los pasos dados y las herramientas utilizadas para consecución de los objetivos del proyecto
- **Capítulo 5: Integración, pruebas y resultados:** Analiza los resultados obtenidos, así como las pruebas realizadas durante el desarrollo del proyecto.
- **Capítulo 6: Conclusiones y trabajo futuro:** Sintetiza las conclusiones que se han obtenido tras la realización del proyecto y propone posibles mejoras.

2 Estado del arte

En este capítulo se realizará un breve estudio de la evolución de los dispositivos móviles hasta la aparición de los smartphones y la penetración que han tenido en la población, También se compararan los sistemas operativos que utilizan, dado que el proyecto consiste en diseño de software, y haciendo una descripción más exhaustiva del sistema operativo Android puesto que es el escogido para el desarrollo del proyecto.

2.1 Evolución de la telefonía móvil

Los orígenes de los primeros dispositivos móviles de comunicación se remontan a la segunda guerra mundial. Se podrían considerar como tales a las primeras radios AM portables que desarrollo el ejército de los Estados Unidos de América como son el SCR-194 y 195. Estos dispositivos, a pesar de ser efectivamente dispositivos portátiles para comunicación a distancia, carecían de características que los hiciesen atractivos para el público en general, dado su peso de alrededor de 25 libras y su tamaño. A lo largo de la guerra surgieron mejores versiones como el SCR-300 o el SCR-536, que incorporaban mejoras como radio FM en el primer caso y con una reducción de peso y dimensiones considerable en el segundo.

Figura 2-1: SCR-194

Tras el fin de la guerra los avances tecnológicos que se produjeron en ella, especialmente en los campos de la electrónica y las telecomunicaciones, fueron aprovechados por diferentes compañías para ofrecer servicios inexistentes hasta la época. Ejemplos de ello son el Mobile Telephone System (MTS) de Bell Systems y los primeros teléfonos para el automóvil como son el Mobile System A (MTA) de Ericsson y el Improved Mobile Telephone Service (IMTS) de Bell. Estos dispositivos fueron disminuyendo paulatinamente en tamaño y peso, llevando a la aparición en 1973 del prototipo del DynaTAC de Motorola, el primer teléfono portable capaz de llamar sin necesidad de un automóvil. Debieron pasar diez años hasta que se comercializo al público, pero su precio resulto prohibitivo para la mayoría de la población, alrededor de 4000 $ de la época, por lo que no se extendió su uso. A lo largo de la siguiente década siguieron apareciendo dispositivos móviles, principalmente de la mano de Motorola, siendo estos cada vez más manejables y con mayor autonomía.

El año 1993 trajo consigo el que probablemente sea el primer smartphone de la historia, el IBM Simon. Este dispositivo incluía además de la posibilidad de realizar llamadas, todas las funcionalidades de una PDA, como son fax, agenda, reloj, correo electrónico y calculadora, y como punto más novedoso, disponía de pantalla táctil.
A pesar de la aparición del IBM Simon, los mayores avances en el diseño de dispositivos móviles no fueron en la integración de servicios y nuevas funcionalidades en los mismos, sino que se enfocaron en características físicas como son el tamaño y el peso. Otro factor importante en ello, fue que la tecnología de telefonía disponible en la época, el 2G, permitía el intercambio de datos, pero a unas tasas todavía demasiado bajas, por lo que su principal uso siguió siendo la comunicación por voz y los mensajes cortos.
Con la aparición de la tercera generación de tecnología de telecomunicación móvil, el 3G, se abrieron nuevas posibilidades en el mundo de los dispositivos móviles. La tendencia previa de enfocar las mejoras en las dimensiones de los dispositivos cambió y se empezaron a ofrecer cada vez más funcionalidades y servicios. Esta tecnología permitía unas tasas de transmisión de datos mucho mayores, lo que provocó un auge en servicios como las video llamadas, internet móvil y el GPS.
El incipiente uso de estos nuevos servicios conllevó el surgimiento de dispositivos como la BlackBerry, que permitía el acceso a internet y el correo electrónico, pero el verdadero estallido de los smartphones no se produjo hasta la salida del iPhone en 2007. El nuevo producto de Apple incorporaba todos estos servicios, además de una pantalla táctil a color y una interfaz innovadora. Fue un tremendo éxito de ventas que provoco un cambio en la concepción del diseño de dispositivos móviles.
El mismo año de la salida al mercado del iPhone, Google presentó su sistema operativo para móviles, Android, que acabó por convertirse en el más usado en el mundo.

2.2 Sistemas operativos para smartphones

Dadas las características, próximas a las de un ordenador personal, de los smartphones aparecieron varios sistemas operativos para gestionar el software y hardware, así como para proporcionar servicios comunes a los usuarios de los dispositivos.

Son dos de ellos, Android e iOS los que se han hecho con la mayor parte del mercado, debido en el primer caso principalmente a la variedad de dispositivos que lo incorporan, puesto que está basado en código libre, y en el segundo caso a las ventas masivas que se han producido de smarphones y tabletas de Apple.

En esta sección se realizará una descripción de las características principales de los más usados, extendiéndose especialmente en el escogido para el proyecto.

2.2.1 Android

El sistema operativo Android fue desarrollado inicialmente por Android Inc., empresa que fue financiada por Google entre otros y finalmente absorbida por el mismo. Se hizo público por primera vez en 2007 junto a la fundación del Open Handset Alliance, un consorcio de compañías tecnológicas, destinada al avance en los estándares abiertos de los dispositivos móviles. Un año más tarde salió a la venta el primer dispositivo que lo utilizó, el HTC Dream.

Android es un sistema operativo de código abierto y por lo tanto cualquier desarrollador puede crear aplicaciones en el lenguaje que desee y compilarlas a código nativo de ARM, así como acceder al código fuente para modificarlo o mejorarlo. La publicación de aplicaciones se realiza a través de la plataforma que ofrece Google, llamada Google Play, en la que sólo hace falta pagar una cuota al registrarse como desarrollador, lo que la comunidad de desarrolladores para Android y por tanto el número de aplicaciones, no han parado de crecer, desde su lanzamiento en 2007. Actualmente hay más de 1.3 millones de aplicaciones disponibles en Google Play.

Las principales características de Android que le diferencian de su principal competidor, iOS, son:

- **Código abierto**: El elemento más característico de Android es el estar desarrollado en código abierto, lo que permite que cualquier desarrollador acceda a él y lo modifique. Esto permite una continua mejora en el mismo.

- **Arquitectura de componentes:** Los elementos de una aplicación pueden ser usados por otras e incluso se pueden cambiar componentes integrados por otros, lo que activa la creatividad de los desarrolladores.

- **Servicios integrados:** El propio sistema operativo ofrece de serie una enorme variedad de servicios como puede ser el GPS, navegador, reconocimiento de voz o bases de datos.

- **Seguridad:** Gracias a la máquina virtual, cada aplicación tiene limitado su alcance, resultando como una caja negra a efectos del resto de aplicaciones.

- **Gráficos y sonido de alta calidad:** Android dispone de biblioteca de gráficos 2D y 3D basada en las especificaciones de la OpenGL ES 2.0 y soporta formatos multimedia como MP4, MPEG-4 SP, AMR, WebM, AAC, MP3, MIDI, Ogg Orbis, WAV, JPEG, PNG, GIF y BMP.

- **Portabilidad:** Prácticamente todos los dispositivos de hoy en día disponen de una máquina virtual de Java, y Android utiliza la máquina virtual Dalvik, basada en ella, esto permite que el código Java previamente compilado sea ejecutable una enorme variedad de dispositivos.

Otra consecuencia de su naturaleza de código abierto es el feedback recibido sobre los errores en el sistema operativo. Este hecho junto con la incorporación de nuevas funcionalidades en los dispositivos y la aparición de nuevo hardware, como son las tablets o los wearables, ha propiciado una actualización muy dinámica del código, apareciendo nuevas versiones cada poco tiempo. Cada nueva versión recibe el nombre de un postre en inglés siguiendo un orden alfabético.

Nombre de la versión	Numero de versión	API
Beta	-	-
Apple Pie	1.0	1
Banana Bread	1.1	2
Cupcake	1.5	3
Donut	1.6	4
Eclair	2.0	5
Eclair 0.1	2.0.1	6
Eclari MR1	2.1	7
Froyo	2.2	8
Gingerbread	2.3	9
Gingerbread MR1	2.3.3	10
Honeycomb	3.0	11
Honeycomb MR1	3.1	12
Honeycomb MR2	3.2	13
Ice Cream Sandwich	4.0	14
Ice Cream Sandwich MR1	4.0.3	15
Jelly Bean	4.1	16
Jelly Bean MR1	4.2	17
Jelly Bean MR2	4.3	18
Kitkat	4.4	19
Lollipop	5.0	21

Tabla 1-1: Versiones de Android

La arquitectura interna del sistema operativo está compuesta por cuatro capas principales de software libre. Cada capa representa un nivel de abstracción entre el software y el hardware y ofrece sus servicios a las capas superiores. Se describirá brevemente cada capa en orden de abstracción ascendente.

Núcleo de Linux: Esta es la capa que comunica el hardware con las capas superiores. Se compone de los controladores para el hardware de los dispositivos. Dispone de controladores para la pantalla, la conexión wifi, el audio, la interconexión con el pc, la gestión de energía, la gestión de procesos y la gestión de memoria.

Bibliotecas: Esta capa está dividida en dos partes, las bibliotecas nativas de Android, que proporcionan funcionalidad a las aplicaciones para las tareas más comunes y el runtime de Android. En el runtime se encuentran las bibliotecas esenciales de Android y la máquina virtual Dalvik. A diferencia de la máquina virtual de Java, no utiliza archivos tipo ".class", sino que usa otros específicos de Android tipo ".dex". Estos archivos están optimizados para Android y están más comprimidos que los de tipo ".class". Es una maquina basada en registros y no en pila, por lo que produce ejecuciones más rápidas.

Entorno de aplicación: Aquí están las clases y servicios necesarios para la realización de las funciones de las aplicaciones. Se apoya en los elementos del runtime y las bibliotecas para trabajar.

Aplicaciones: Éste es el nivel más alto de abstracción, donde se encuentran las aplicaciones. El sistema operativo incluye una serie de aplicaciones para uso del usuario, como son la guía de contactos, el navegador, la aplicación para realizar llamadas o la galería de fotos. Algunas

están escritas en C o C++, mientras que otras de tipo administradas están desarrolladas en Java.

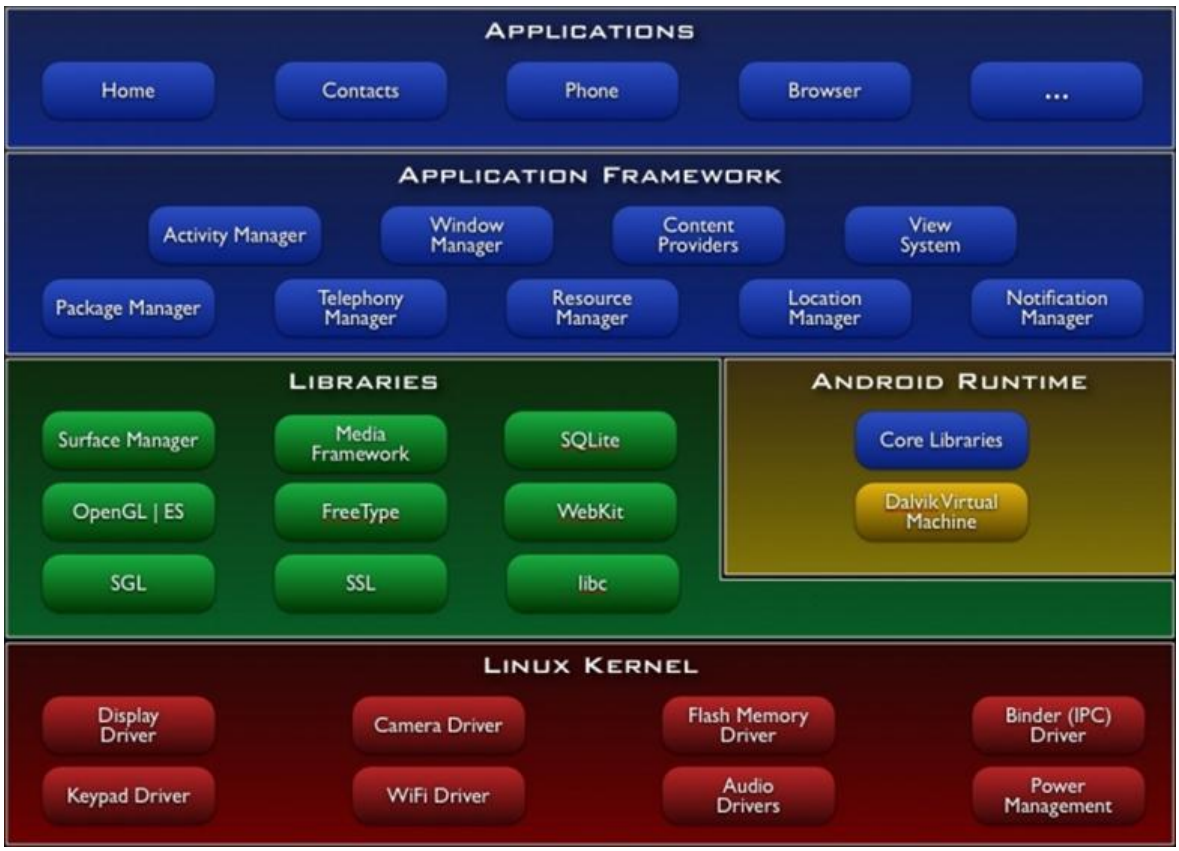


Figura 2-2: Arquitectura de Android

2.2.2 iOS

Este sistema operativo fue desarrollado por Apple Inc. para dar soporte a sus propios productos de hardware. Salió al mercado junto con el iPhone en 2007 con el nombre de iPhone OS y no fue hasta 2010, con la introducción del iPad y el iPod Touch, que fue renombrado a iOS, puesto que ya no solo era el sistema operativo para el iPhone sino para todos los dispositivos móviles de Apple.

A diferencia de Android, no está basado en código abierto, sino en el núcleo del sistema operativo de Apple, el Mac OS X. El lenguaje que utiliza es Objective-C, un superconjuto de C orientado a objetos.

En el caso de iOS las actualizaciones se han producido, en la mayoría de los casos, con la salida de algún nuevo producto de hardware de Apple. Al no disponer de una comunidad tan grande de desarrolladores como Android, y tener que restringirse exclusivamente a sus propios productos, no ha habido necesidad de realizar tantos cambios en el sistema operativo a lo largo del tiempo. En la siguiente tabla se exponen las versiones de iOS y sus fechas de salida.

Nombre de la versión	Fecha de salida
iPhone OS 1	Marzo 2008
iPhone OS 2	Julio 2008
iPhone OS 3	Junio 2009
iOS 4	Junio 2010
iOS 5	Junio 2011
iOS 6	Junio 2012
iOS 7	Junio 2013
iOS 8	Junio 2014

Tabla 2.1: Versiones de iOS

2.2.3 Elección del sistema operativo

2.2.3.1 Penetración de los smartphones en España

Puesto que el proyecto aspira a aprovechar el uso extendido de los dispositivos móviles entre los estudiantes para ofrecer una nueva manera de ejercitarse en los circuitos digitales, resulta útil analizar la penetración que han tenido tales dispositivos en la población.

Según el estudio de Deloitte "Consumo Móvil en España 2014", España se encuentra en el cuarto puesto de los países desarrollados en cuanto a penetración de smartphones se refiere, con un 85%.

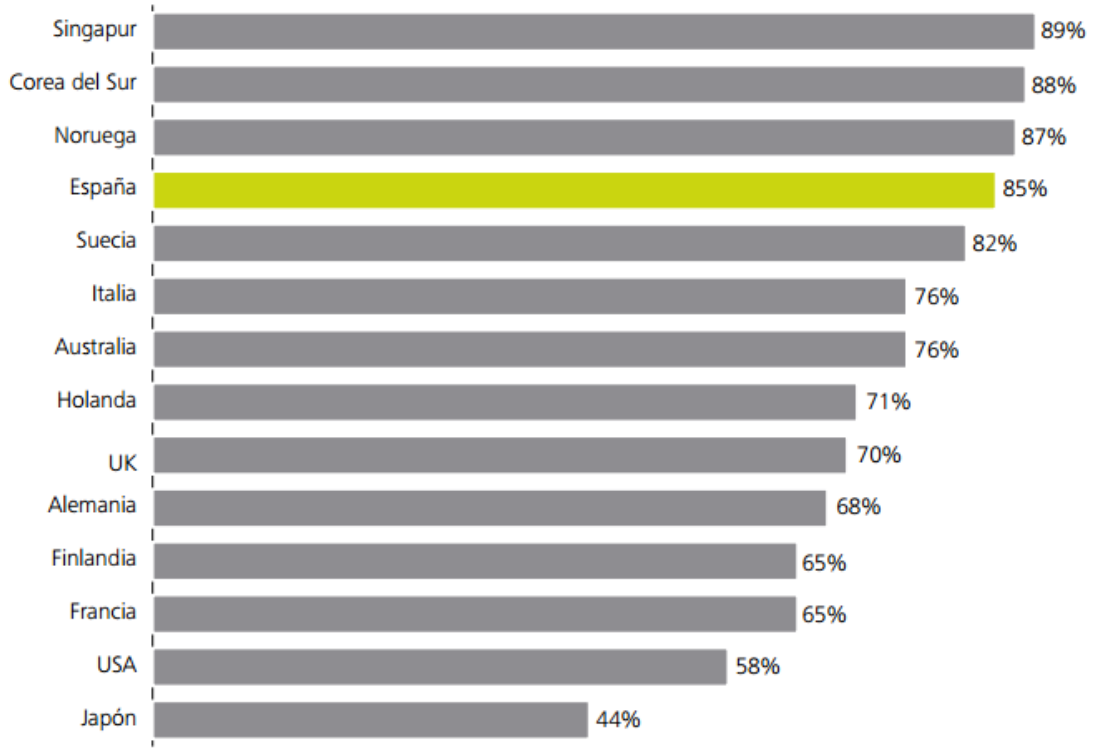


Figura 2-3: Penetración smartphones en España

Este dato demuestra el incipiente uso, en cada vez más disciplinas de la vida diaria, que tienen los smartphones.

Más allá del uso generalizado, los grupos de población de entre 18 y 24 años, y entre 25 y 34, son los mayores consumidores de esta nueva tecnología, lo que invita a pensar que, si se diese oferta de aplicaciones educativas como complemento al estudio, podrían tener un éxito considerable entre la comunidad universitaria.

2.2.3.2 Cuota de mercado según el Sistema Operativo

El principal elemento diferenciador a la hora de realizar una aplicación para smartphones es sin duda el sistema operativo para el que se va desarrollar. Cuando el mercado era todavía joven, el iPhone era prácticamente el único Smartphone en el mercado y por tanto acaparaba prácticamente la totalidad de ventas de este tipo de dispositivos. Tras la llegada de Android, y su capacidad de dar soporte sobre una mayor variedad de dispositivos, iOS fue perdiendo paulatinamente la hegemonía en esta materia. Hubo otros sistemas operativos desarrollados por Windows, BlackBerry y Nokia, que no terminaron de convencer a los consumidores, y fueron arrollados junto con iOS por Android. En la figura siguiente se puede apreciar cómo Android ha experimentado una subida constante desde su comienzo, superando a iOS en menos de tres años.

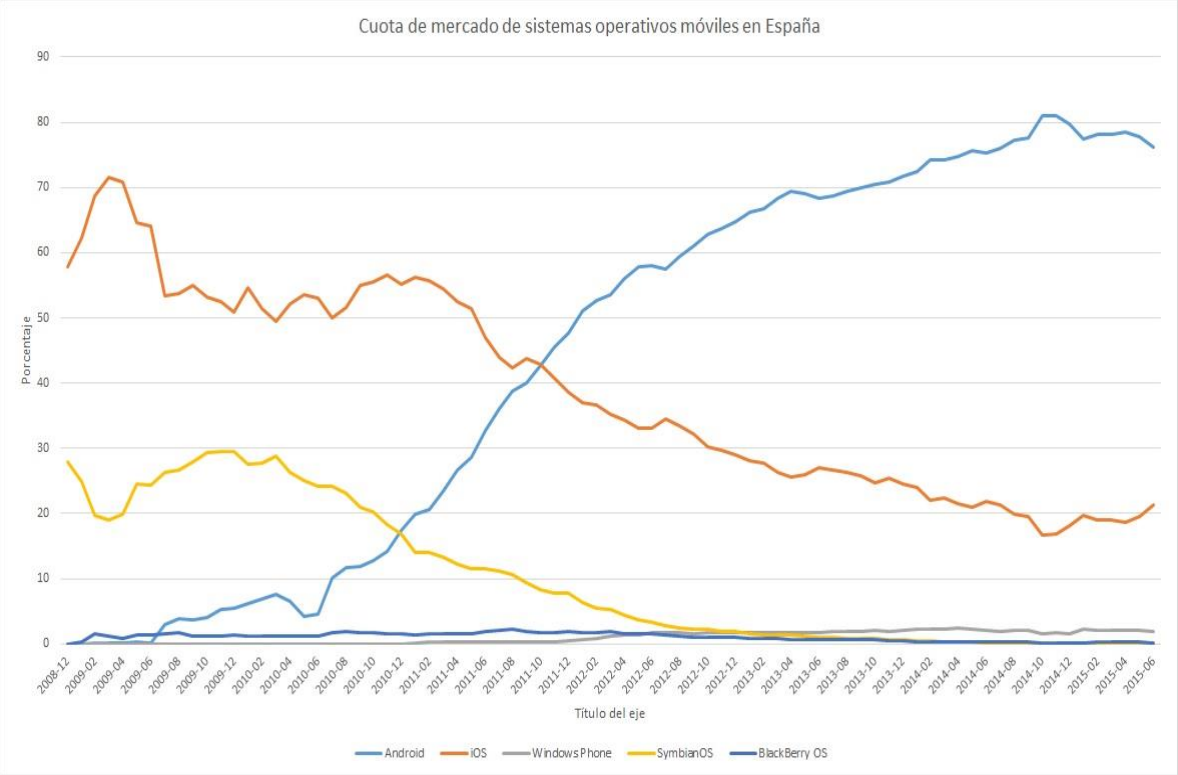


Figura 2-4: Cuota de mercado por SO

2.2.3.3 Conclusión

Visto el tremendo éxito que ha tenido Android como sistema operativo para smartphones, resulta una buena elección para el desarrollo del proyecto. Además, Android ofrece mayores facilidades a la hora de publicar aplicaciones en su plataforma Google Play que Apple, puesto que para crear una cuenta como desarrollador, tan solo hace falta registrarse y pagar una cuota inicial, que da licencia de por vida, mientras que en el AppStore de Apple es necesario el registro y un abono anual para mantener la licencia. Todos estos factores hacen que se haya escogido Android como sistema operativo sobre el que realizar el proyecto.

2.3 Aplicaciones similares

Antes de desarrollar la aplicación, resulta conveniente buscar aplicaciones similares ya existentes en el mercado. Por ello se ha buscado en las tiendas de aplicaciones de los sistemas operativos más usados. Desgraciadamente no se han encontrado aplicaciones que se centren en la comprobación de errores en circuitos lógicos mediante la técnica de stuck-at.

2.3.1 Aplicaciones DSLab UAM

Existen cuatro aplicaciones realizadas en el DSLab UAM como proyecto de fin de carrera por otros compañeros, que ofrecen la misma funcionalidad que la que se desarrolla en este proyecto, centrándose en otros temas de la asignatura.

	Nombre	Estado	Resumen
	Combinational Circuits	Publicada y finalizada	Ofrece una serie de ejercicios de electrónica combinacional.
	Sequential Circuits	Publicada y finalizada	Ofrece una serie de ejercicios de electrónica secuencial (máquinas de estados).
	KARN Map	Publicada y finalizada	Ofrece una serie de herramientas para la simplificación de funciones lógicas.
	MOS Circuits	Publicada y finalizada	Ofrece una serie de ejercicios de circuitos integrados MOS.

Tabla 2.2: Aplicaciones anteriores

Como se indica en las especificaciones del proyecto, la aplicación que se va a desarrollar debe ofrecer características de cara al usuario iguales a las desarrolladas en estas aplicaciones, por lo que gran parte de los diseños de las vistas se han tomado de ellas, especialmente Combinational Circuits y MOS Circuits, dada la similitud conceptual de los problemas que resuelven en ellas con los que se va a resolver en este proyecto, ya que en ambas se utilizan tablas para la resolución de problemas, técnica que será utilizada para la resolución de los problemas en esta aplicación.

3 Diseño

En esta sección se analizaran los métodos y herramientas a utilizar para la consecución de los objetivos marcados.

3.1 Requisitos

Puesto que se trata de una aplicación perteneciente a un grupo de aplicaciones ya desarrolladas previamente en otros proyectos de fin de carrera, deberá, como fue estipulado en la sección de objetivos, respetar las características de estilo y funcionamiento de cara al usuario que tienen el resto de aplicaciones del grupo.

Por lo tanto se pasará a desglosar las características y explicar su razón de ser.

3.1.1 Versión de Android

La primera característica que hay que decidir, ya que condiciona las herramientas que se van a poder utilizar para el desarrollo del proyecto, son las versiones de Android para las que va a diseñarse la aplicación. Para ello se analizará que versiones son las más usadas y, por tanto, las que van a permitir que la aplicación sea instalada en más dispositivos.

En el propio portal de Android disponen de estadísticas actualizadas del uso de sus productos, lo cual va a resultar muy cómodo para la toma de decisiones tanto de la versión como de otras características que se analizaran más adelante.

Versión	Nombre	API	Distribución
2.2	Froyo	8	0.3%
2.3.3-2.3.7	Gingerbread	10	5.6%
4.0.3-4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jellybean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

Tabla 3.1: Distribución de versiones de Android

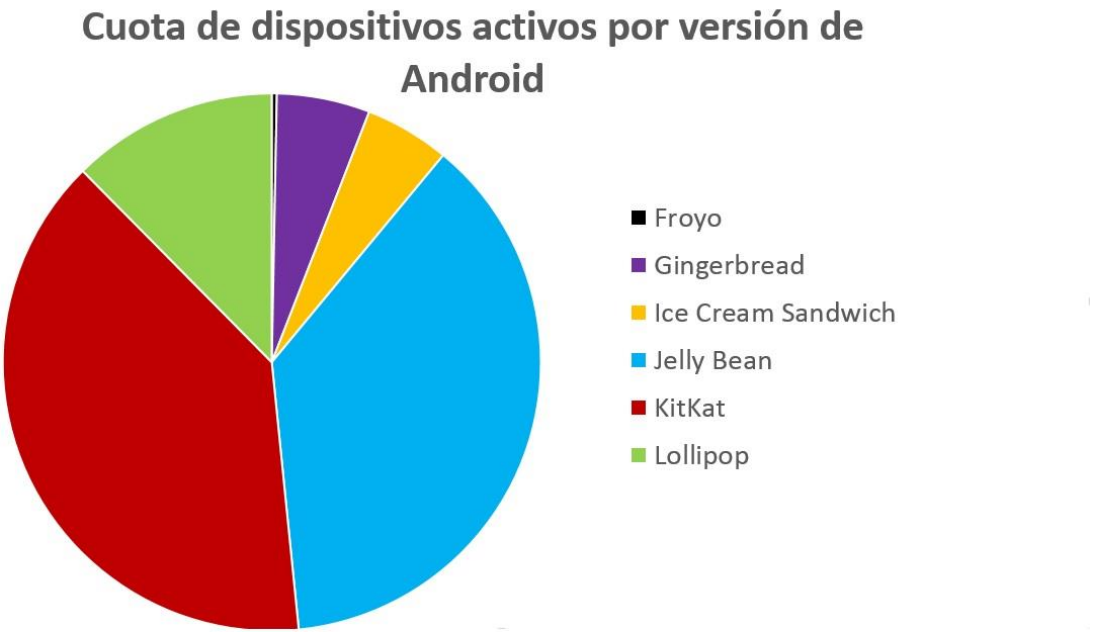


Figura 3-1: Cuota de dispositivos por versión de Android

Como se puede ver en la tabla 3.1 la mayoría de dispositivos utilizan versiones a partir de la 4.1 (Jelly Bean). Lo que va a condicionar el desarrollo de la aplicación va a ser la versión mínima para la que debe funcionar.

Si se diseñase la aplicación escogiendo como versión mínima la 4.1, se accedería a aproximadamente un 89% de los usuarios de Android. Puesto que Android ofrece librerías especiales de apoyo para poder utilizar elementos de las versiones más modernas en aplicaciones para versiones antiguas, no se perdería funcionalidad seleccionando una versión inferior. Por lo tanto, se ha decidido que la versión mínima sea la 2.2 (Froyo), para así poder acceder a una mayor cantidad de usuarios.

3.1.2 Tipos de pantalla

Otra característica vital para la aplicación son los tipos de pantalla que soportará. Ya que Android no restringe su uso a dispositivos de su marca, ofrece una estructura que permite construir *layouts* (diseños) específicas e incluir imágenes para cada tipo de pantalla.

Hay dos parámetros básicos que hay que tener en cuenta para ello, el tamaño, medido en pulgadas, y la densidad de pantalla, medida en dpi (*dots per inch*).

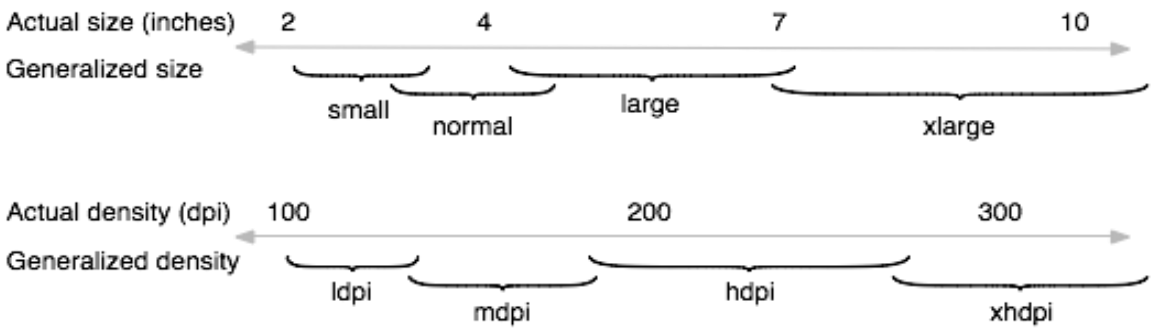


Figura 3-2: Grupos de tamaño y densidad de pantalla

3.1.2.1 Tamaños de pantalla

Puesto que Android da servicio a infinidad de dispositivos, teniendo cada fabricante sus propios estándares de tamaño de pantalla, sería prácticamente imposible, y en cualquier caso muy engorroso, tener que programar distintas versiones de cada aplicación que se desarrolla, para poder abarcar todos los dispositivos. Para estandarizar el desarrollo de las aplicaciones, Android clasifica los tamaños de pantalla en cuatro grupos generalizados: *small*, *normal*, *large*, *xlarge*.

Estos grupos abarcan tamaños de pantalla desde las dos pulgadas hasta por encima de diez. Resulta práctico saber cuáles de ellos son los más usados, para poder diseñar la aplicación ofreciendo al usuario la mejor experiencia posible.

Tamaño de pantalla	Distribución
Small	4.1%
Normal	83.3%
Large	8.6%
Xlarge	4.0%

Tabla 3.2: Distribución de tamaños de pantalla

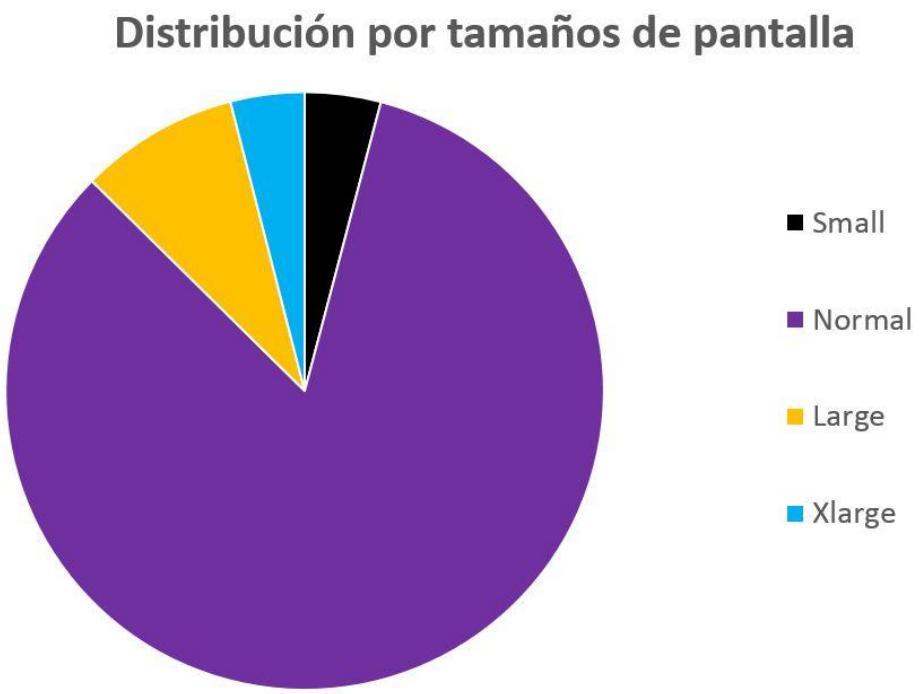


Figura 3-3: Distribución por tamaños de pantalla

Viendo los datos obtenidos de Android, queda patente que la inmensa mayoría de dispositivos que lo utilizan corresponden a la categoría *normal*.

3.1.2.2 Densidad de pantalla

El otro parámetro de la pantalla a tener en cuenta a la hora de diseñar la aplicación es la densidad de pantalla.

La densidad de pantalla son la cantidad de pixeles existentes en un área física de una pulgada de la pantalla. Android trabaja con este valor relativo en lugar de con la resolución, debido a que a la hora de diseñar las aplicaciones, si se utilizase la resolución, el tamaño de las imágenes no se mantendría constante en los distintos tamaños de pantalla. Utilizando la densidad de pantalla como parámetro con el que trabajar, el tamaño se mantendrá constante y por tanto las proporciones entre los distintos elementos de una vista también.

Igual que pasaba con los tamaños de pantalla, para poder dar soporte al mayor número de dispositivos, sin necesidad de incluir una cantidad exagerada de recursos para cada configuración, Android clasifica las densidades de pantalla en seis grandes grupos: *ldpi* (~120dpi), *mdpi* (~160dpi), *hdpi* (~240dpi), *xhdpi* (~320dpi), *xxhdpi* (~480dpi), *xxxhdpi* (~640dpi).

Densidad de pantalla	Distribución
ldpi	4.5%
mdpi	15.5%
tvdpi	2.3%
hdpi	40.8%

xhdpi	21.0%
xxhdpi	15.9%
xxxhdpi	-

Tabla 3.3: Distribución por densidades de pantalla

La densidad de pantalla *tvdpi* no está considerada una densidad primaria, puesto que se utiliza en televisores, por tanto no la incluiremos en la valoración final para la elección.

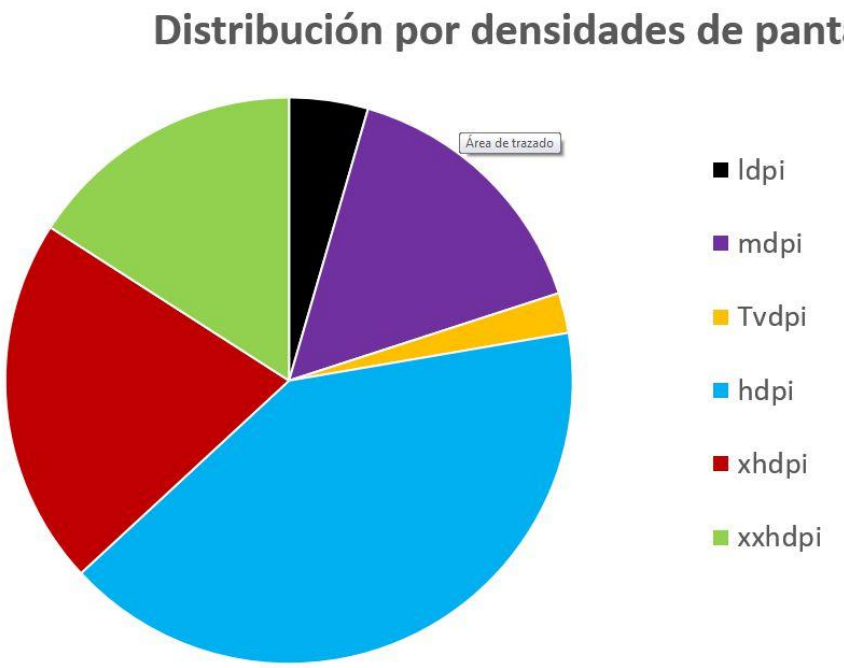


Figura 3-4: Distribución por densidades de pantalla

Observando la figura 3.4 se puede apreciar que las densidades usadas mayoritariamente son *mdpi*, *hdpi*, *xhdpi* y *xxhdpi*.

3.1.2.3 Elección de tipos de pantalla soportados

A la vista de los datos reflejados en las dos secciones previas y al ser objetivo del proyecto que la aplicación sea utilizable por la mayor cantidad de usuarios posible, se ha decidido dar soporte a las pantallas con los cuatro tamaños ofrecidos por Android y densidades *ldpi*, *mdpi*, *hdpi*, *xhdpi* y *xxhdpi*.

3.1.3 Formato

Al tratarse de una aplicación perteneciente a una colección de aplicaciones para la enseñanza de electrónica digital creada en el DSLab, uno de los requisitos básicos es también el formato que debe tener la aplicación.

Actualmente hay cuatro aplicaciones del DSLab UAM disponibles en Google Play. Se expondrán los aspectos de diseño principales que deben mantenerse en la aplicación a diseñar.

Nombre	Tamaño	Fecha de última actualización	Version	Idioma
Sequential Circuits	6.37 MB	23/02/2014	1.0	Inglés
Combinational Circuits	7.19 MB	25/09/2014	1.0	Inglés
Karn Map	1.97 MB	19/11/2014	2.0	Inglés y Castellano
MOS Circuits	6.74 MB	17/01/2015	2.0	Inglés

Tabla 3.4: Aplicaciones anteriores

3.1.3.1 Idioma

Como se puede ver en la tabla 3.4 el idioma escogido para las aplicaciones de electrónica del DSLab es el inglés.

A pesar de que la asignatura Dispositivos Integrados Especializados se imparte en castellano el inglés es el idioma más internacional en materias de electrónica, y es hablado prácticamente en todo el mundo. Puesto que la aplicación va a ser publicada en Google Play, y esta es una plataforma internacional para la distribución de aplicaciones, resulta práctico realizar la aplicación en este idioma si se pretende acceder al mayor número de usuarios posibles.

3.1.3.2 Colores

La aplicación utilizara las combinaciones de colores, tanto para los botones como para el fondo de pantalla, tutorial, ejercicios y test, que se ha utilizado en las cuatro aplicaciones ya publicadas. Esta medida es simplemente cuestión de coherencia estética, puesto que no afecta realmente a ningún aspecto práctico de la aplicación.

3.1.3.3 Tamaño de letra

La aplicación respetará las proporciones mantenidas en las aplicaciones ya publicadas. Los tamaños de letra utilizan como medida los *sp* (scale-independent pixels). Esta medida tiene en cuenta la configuración general del usuario, y escala los tamaños de letra consecuentemente. Utilizando esta medida se evita que las letras aparezcan con distinto tamaño en cada dispositivo con densidad y tamaño de pantalla distintos.

3.1.3.4 Formato de las vistas

Gracias a que Android utiliza archivos XML para los diseños de cada vista, se podrán reutilizar los ya creados en las aplicaciones del DSLab, para así manter una absoluta congruencia tanto en márgenes, espacios, posición y tamaño de los elementos. Obviamente habrá que adaptarlos apropiadamente para añadir las nuevas funcionalidades añadidas.

3.1.4 Resumen de requisitos

Resumiendo lo expuesto en la sección anterior la aplicación deberá cumplir los siguientes requisitos:

- Sistema operativo: Android
- Versiones del sistema operativo soportadas: 2.2 hasta 5.1
- Tipos de dispositivos válidos:
  - Smartphones
  - Tablets
- Tipos de pantalla:
  - Por tamaño:
    - Small
    - Normal
    - Large
    - Xlarge
  - Por densidad de pantalla:
    - ldpi
    - mdpi
    - hdpi
    - xhdpi
    - xxhdpi
- Idioma: Inglés
- Colores:
- Tamaños de letra:
  - Small:
  - Normal:
  - Large:
  - Xlarge:

3.2 Herramientas

Para el desarrollo de aplicaciones en Android existen principalmente dos entornos de desarrollo: Eclipse y Android Studio. Al comienzo del desarrollo de la aplicación objeto de este proyecto, se utilizó el Eclipse ADT para el desarrollo. Durante el mismo Google sacó al mercado un nuevo entorno para el desarrollo de aplicaciones, Android Studio. A raíz de la salida del nuevo entorno Google ha dejado de ofrecer actualizaciones del plug-in para Eclipse, por lo que esta Android Studio está destinado a convertirse en el entorno de desarrollo oficial para aplicaciones Android en el futuro y por tanto, resulta conveniente el

aprender su uso para futuros proyectos sobre la materia. Para una comprensión de sus diferencias se describirá brevemente ambos entornos de desarrollo.

3.2.1 Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) multiplataforma compuesto por un conjunto de herramientas de programación de código abierto. Está programado principalmente en Java y consta de un workspace y un sistema extensible de plug-ins que permite adaptar el entorno al lenguaje de programación deseado. Se trata de una herramienta muy versátil puesto que soporta un abanico inmenso de lenguajes y muchas herramientas adicionales ofrecidas a través de la comunidad eclipse.

En cuanto al desarrollo de aplicaciones para Android, Eclipse ha sido el entorno de desarrollo escogido por Google para el desarrollo de este tipo de aplicaciones. Lanzó el Eclipse ADT (Android Development Toolkit) como herramienta oficial para la programación de aplicaciones Android. Este conjunto de herramientas ofrecía, además del típico procesador de texto mejorado, con autocompletado de sentencias, para la escritura del código, una interfaz gráfica de desarrollo muy útil a la hora de diseñar vistas, puesto que permitía ver la interfaz diseñada al momento, sin necesidad de ejecutar la aplicación.

Los proyectos de eclipse se organizan en workspaces, cosa no tan deseable, ya que si se quiere ejecutar una aplicación que pertenece a otro workspace, hace falta reiniciar el entorno.

Los workspaces ofrecen una estructura de árbol como las carpetas de Windows, en las que se encuentran las bibliotecas, los archivos fuente (src) donde se encuentra el código java de la aplicación, la carpeta assests, donde se pueden añadir las bases de datos u otros recusos externos, la carpeta de recursos (res), donde se encuentran todos los archivos XML con las vistas de la aplicación, así como las imágenes adaptadas a cada tamaño y densidad de pantalla.

3.2.2 Android Studio

Android Studio es el entorno de desarrollo exclusivamente para Android, creado por Google. Está basado en IntelliJ IDEA de JetBrains y disponible bajo la licencia Apache 2.0. Desde diciembre de 2014 es el IDE oficial de Android y como tal, recibe actualizaciones constantes.

El sistema de construcción es más flexible ya que está basado en el novedoso Gradle. Gradle utiliza un gráfico acíclico dirigido (DAG) para determinar el orden en el que se realizaran las tareas. Con este nuevo sistema, resulta mucho más sencillo añadir bibliotecas de apoyo, así como establecer ciertos parámetros generales de la aplicación. Android Studio también ofrece unas herramientas de refactorización y autocompletado mejores que las de Eclipse, y la interfaz gráfica de diseño de vistas ofrece más posibilidades y muestra más cantidad de elementos.

Los proyectos en Android Studio están estructurados en módulos. Esto permite tener varios proyectos abiertos simultáneamente y poder ejecutarlos independientemente. Además, la estructura de archivos se simplifica, puesto que usando los scripts de gradle, no resulta necesario tener a la vista todas las bibliotecas de java utilizadas, ni los archivos ".jar" autogenerados por java, cosa que resulta muy cómoda, puesto que reduce notablemente el

número de carpetas y elementos del proyecto, permitiendo así un acceso más rápido a las secciones deseadas.

3.2.3 Elección de entorno de desarrollo

Dado que al comienzo del proyecto Android Studio se encontraba todavía en la fase beta y no era suficientemente estable, se optó por utilizar Eclipse ADT. A pesar de ello, debido a que Android dejó de ofrecer actualizaciones para Eclipse, e hizo a Android Studio su entorno de desarrollo oficial, se optó por cambiar a este último para probarlo y así poder barajar, con conocimiento de causa, un cambio de entorno. Para ello hubo que hacer una migración del proyecto. Una vez realizado el cambio, la facilidad de uso que ofrecía la nueva estructura y nuevas funcionalidades de Android Studio convencieron e hicieron que finalmente se decidiera utilizar este último como entorno final de desarrollo para la aplicación.

3.3 Módulos de la aplicación

Para mantener la coherencia entre las aplicaciones de enseñanza de electrónica del DSLab, la aplicación ofrece una estructura similar a la de las aplicaciones ya desarrolladas, al menos de cara al usuario. Por ello incluye los módulos básicos ya presentes en las aplicaciones desarrolladas previamente.

3.3.1 Tutorial

La aplicación debe incluir un tutorial acerca de la materia sobre la que son los ejercicios y el test, para facilitar al alumno las bases teóricas necesarias para la resolución de los problemas de test y ejercicios. El tutorial funciona, como en el resto de aplicaciones del grupo, emulando un libro electrónico, pudiendo pasar de página o volver atrás con un gesto del dedo hacia derecha o izquierda respectivamente.

El contenido del texto del tutorial será suministrado por el profesor de la asignatura Eduardo Boemo.

3.3.2 Ayuda

Aparte del tutorial, que está enfocado al contenido teórico, la aplicación ofrece también una sección donde se explica el funcionamiento de la misma, para que el usuario pueda comprobar cómo proceder antes de empezar con uso de los ejercicios o test.

Esta sección consta de varias subsecciones que desglosaran las distintas funcionalidades y módulos de la aplicación.

Este menú de ayudas es accesible a través de dos vías, siendo una el menú principal, donde se ha asignado un botón a esta sección, y otro el botón menú de Android. Al presionar el botón menú se le ofrecen al usuario dos opciones: la ayuda y la página “about”.

3.3.3 Test

Como novedad dentro del grupo, esta aplicación ofrece una sección de preguntas tipo test de respuesta simple con cuatro opciones. Para comodidad del usuario se ha añadido una pestaña a la barra de acción de Android que permite navegar directamente a la pregunta deseada. Una vez terminado el test se le ofrece al usuario una tabla con la puntuación obtenida y los fallos, acierto y preguntas en blanco que ha tenido.

3.3.4 Ejercicios

El módulo principal de la aplicación son los ejercicios. La aplicación ofrece diez ejercicios de problemas de fallos siguiendo el modelo stuck-at, que son resueltos mediante tablas en las que el usuario puede marcar los fallos encontrados en las salidas o entradas según sea el vector de entrada utilizado. Al entrar en el ejercicio, el usuario puede ver el enunciado del problema y dispone de dos botones, uno para pasar a la interfaz para la resolución del mismo y otro para cargar soluciones almacenadas en la memoria del dispositivo.

Una vez entre en la interfaz para la resolución del problema dispondrá de la previamente dicha tabla, donde podrá cambiar el estado de los campos de la tabla para denotar el encuentro de un fallo en el pin correspondiente, marcado en ese caso con "X" o, por el contrario, indicar que el vector de entrada no encuentra el fallo en el pin correspondiente, utilizando en ese caso el símbolo "-".

El mayor problema al que nos enfrentamos a la hora de diseñar el módulo, es la anchura excesiva de las tablas en los ejercicios de este tipo. Para solucionarlo, se ha optado por descomponer la tabla en varias tablas de menor tamaño y ofrecer al usuario una manera similar a la utilizada en el tutorial para navegar entre ellas. Además, se han incluido indicadores sobre la tabla, para que con un solo click, se pueda acceder a cualquiera de ellas, para facilitar aún más el uso de la aplicación.

Se ha decidido incluir solo seis columnas en cada una de estas tablas, para que la superficie que ocupe cada botón sea suficientemente grande como para que resulten responsivos en cualquier tamaño de pantalla.

Todos los ejercicios comparten elementos comunes en la interfaz, accesibles mediante botones, que permiten el control de errores, ver el esquemático del circuito a analizar en el problema, resolución automática del problema, reseteo de los estados de la tabla a la situación inicial y guardado del estado actual de la tabla en un archivo ".csv" para el posterior envío por correo electrónico.

3.3.4.1 Esquema

El primer botón que ofrece la interfaz de los ejercicios es el de esquema. Este botón permite al usuario visualizar el esquema del circuito a analizar, ocupando este toda la pantalla del dispositivo para poder ver bien las conexiones y nombre de las variables implicadas en el mismo.

3.3.4.2 Guardar

El segundo botón ofrece la posibilidad de guardar el estado del ejercicio en cuestión en la memoria del teléfono para poder continuar con él más tarde o para compartirlo vía correo electrónico con otro usuario de la aplicación o con el profesor de la asignatura. Al presionar el botón el usuario podrá asignarle el nombre que él considere oportuno al ejercicio, respetando un prefijo que proporcionara la aplicación automáticamente para poder diferenciar tanto el número del ejercicio, como la localización en memoria (memoria interna o externa) del archivo resultante del proceso de guardado.

3.3.4.3 Comprobación

El tercer botón permite al usuario comprobar qué filas de la tabla contienen errores o en caso de estar todo correcto avisa de ello a través de un mensaje tipo "Toast".

3.3.4.4 Resolver

La cuarta opción en la interfaz de los ejercicios es la resolución automática del ejercicio. Una vez pulsado el botón las celdas de la tabla mostrarán la solución y se le avisará al usuario como en el caso de la comprobación a través de un mensaje tipo "Toast".

3.3.4.5 Reset

El último botón permite al usuario reiniciar el estado de las celdas del ejercicio, preguntándole previamente si está seguro de ello, puesto que los cambios realizados previamente se perderán una realizada esta acción.

3.3.5 Envío por correo

Por último la aplicación también ofrece un módulo para el envío de soluciones por correo electrónico para ofrecer la posibilidad de compartir resultados entre compañeros o envío de las soluciones al profesor para pedir consejo.

3.4 Métodos de obtención de datos

Para la obtención de los datos de los ejercicios y el test Android ofrece distintas formas de almacenamiento. En las aplicaciones previas se utilizan las SharedPreferences, para la comunicación de datos entre actividades, y archivos XML para el almacenamiento de las cadenas de caracteres.

Las SharedPreferences son un archivo con tablas tipo clave-valor, que permite el almacenamiento de datos primitivos como enteros, booleanos o cadenas de caracteres. Al

guardar datos en ellas, éstos se pueden utilizar en cualquier actividad de la aplicación haciendo referencia a la clave asignada a cada valor. Puesto que Android ofrece otras vías, como la inclusión de datos "extra" en los intents que llaman a las actividades, se ha optado por reducir el uso de las SharedPreferences.

En esta aplicación se utiliza una base de datos SQL para la obtención de los contenidos, con el fin de ofrecer una mayor facilidad para la actualización de la aplicación y con vistas del desarrollo de otra aplicación que aglomere las aplicaciones ya desarrolladas. Con esta medida se facilita enormemente la reusabilidad del código.

4 Desarrollo

En este capítulo se enumerará el material utilizado, los bloques básicos que ofrece Android para el desarrollo de aplicaciones móviles, se describirá la programación de la aplicación, explicando la función de las clases generadas y la estrategia seguida para el diseño de las mismas. Se comentarán también los problemas surgidos durante el proceso de desarrollo y las mejoras introducidas para subsanar los mismos.

4.1 Material utilizado

Para el desarrollo de la aplicación han hecho falta una serie de elementos físicos. Estos son:

- Ordenador personal
- Eclipse IDE con el ADT de Android (hasta la migración a Android Studio)
- Android Studio
- Java
- Smartphone Android

4.2 Bloques básicos de Android

Android ofrece varias herramientas para el diseño y desarrollo de las aplicaciones. Se describirán brevemente los bloques de construcción básicos utilizados en la aplicación.

4.2.1 Activity

Las Activity (actividades) son el elemento fundamental de cualquier aplicación Android. Ofrecen un marco donde gestionar tanto los elementos mostrados en la pantalla como las interacciones del usuario. Son una clase abstracta creada por el equipo de Android que gestiona automáticamente su ciclo de vida dependiendo de las acciones realizadas por el usuario.

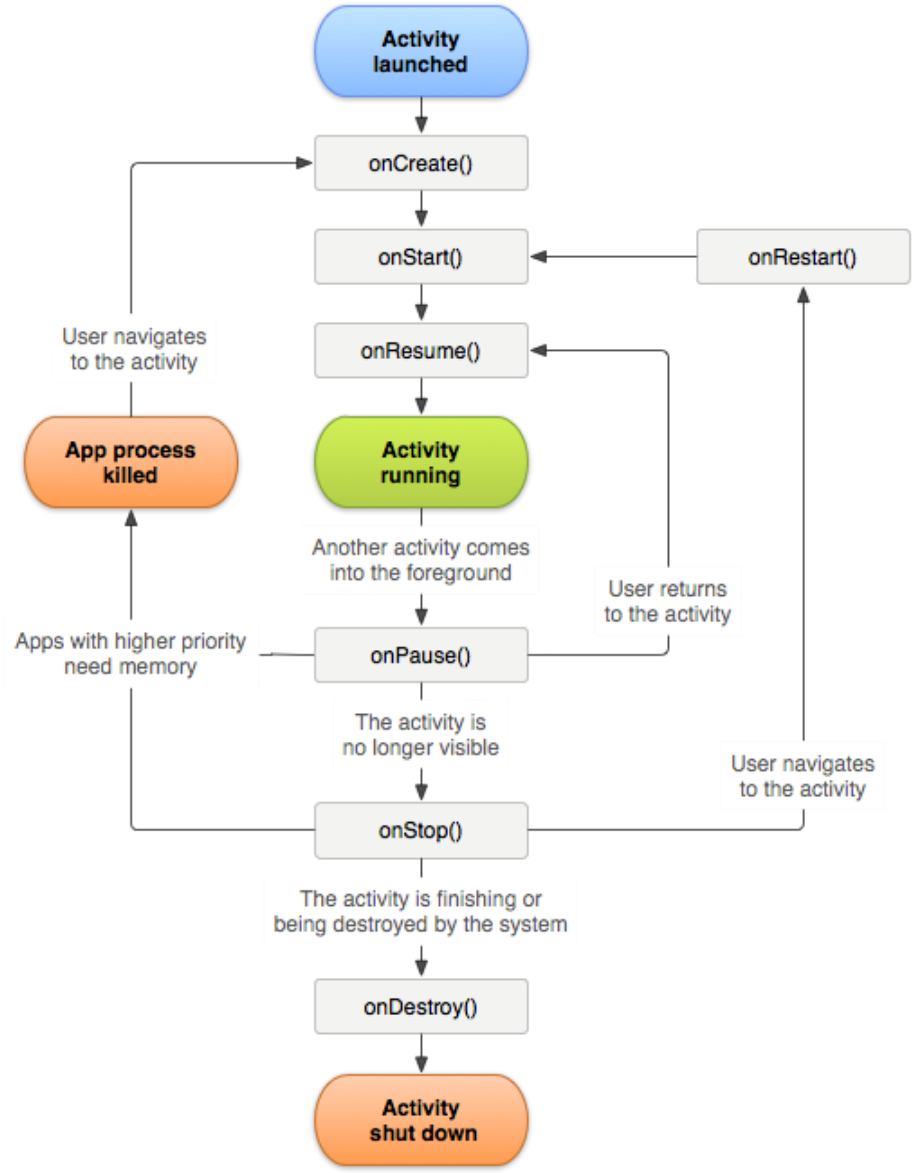


Figura 4-1: Ciclo de vida de una actividad

La gestión del ciclo de vida se realiza automáticamente por Android, pero hay métodos dentro de esta clase que permiten a cualquier desarrollador realizar acciones en cada uno de ellos, como puede ser el almacenamiento de estados de elementos de la interfaz, el inicio de otra actividad o la destrucción de la actividad en cuestión.

Toda actividad está ligada a un archivo XML que representa la interfaz asociada a la misma, aunque como se verá más adelante, se pueden crear elementos dinámicos en la interfaz utilizando otras clases ofrecidas por Android.

Las actividades también pueden llamar a otras actividades a través de la clase Intent que ofrece Android, sin que estas últimas sean necesariamente de la misma aplicación. Esto resulta especialmente útil para funcionalidades como el envió por correo electrónico.

Existen distintos tipos de actividades que extienden a las actividades estándar, pero ofreciendo funcionalidades especiales para el desarrollo más cómodo de actividades con

elementos típicos de Android. Puesto que en el proyecto se utilizaran componentes pertenecientes a las librerías de apoyo, las actividades deberán extender a la clase AppCompatActivity. Esta clase es una extensión de la clase base Activity diseñada específicamente para ofrecer acceso a las funcionalidades de los elementos de las librerías de soporte más actuales.

4.2.2 Fragment

Los Fragment (fragmentos) son muy parecidos a las actividades. Son elementos que pueden asumir la apariencia de una actividad o simplemente de un elemento de la interfaz. Disponen de un ciclo de vida muy similar a las actividades, pero no disponen de la independencia de éstas, ya que deben estar asociados obligatoriamente a una actividad, y su ciclo de vida depende directamente de la actividad que los aloja.

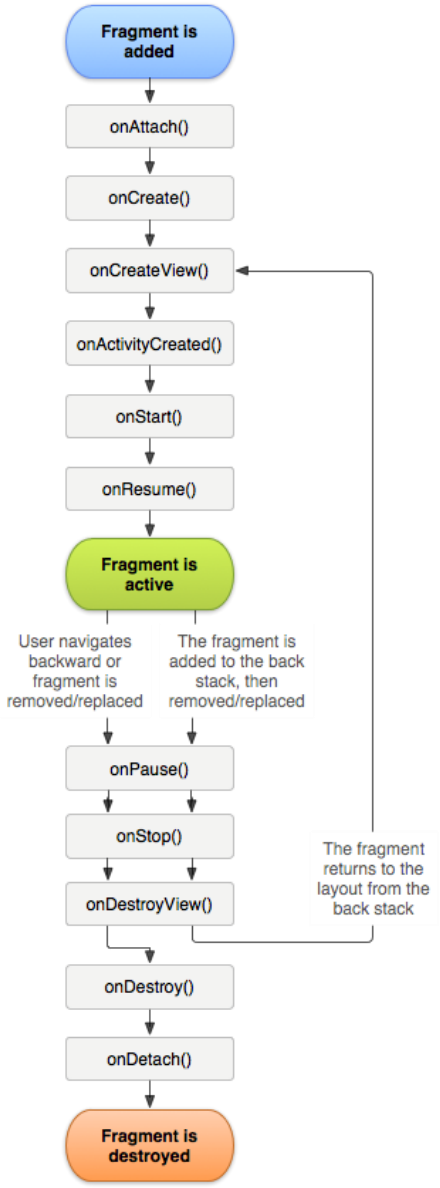


Figura 4-2: Ciclo de vida de un fragmento

Desde la aparición de esta clase en Android, se han desarrollado muchas clases que la extienden y la utilizan, puesto que permite un diseño mucho más dinámico de las interfaces de usuario. Muchos de los widgets que se describirán en la siguiente sección tienen un diseño basado en fragmentos o los utilizan para realizar sus funciones, por lo que se trata de una clase con una enorme potencia de uso.

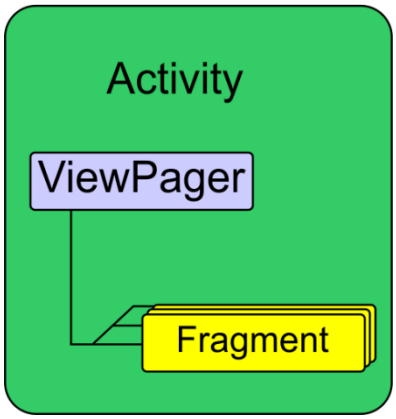
4.2.3 Widgets

Los widgets son los elementos de la interfaz diseñados por Android. Incluyen los botones, las imágenes, los textos e incluso elementos más avanzados como pueden ser el ViewPager o los diálogos.

No se describirán los elementos más básicos como los botones o las imágenes por su trivialidad, pero sí algunos de los elementos avanzados utilizados en la aplicación.

4.2.3.1 ViewPager

Esta clase permite simular el paso de páginas dentro de una misma actividad utilizando fragmentos anidados para la representación de cada página. Los métodos que incluye gestionan la creación y destrucción de fragmentos del tipo que se especifique y la creación de la animación correspondiente al paso de página al deslizar el dedo en un sentido u otro.
Se utiliza en el tutorial por su idoneidad para la simulación de un libro electrónico, en el test y también en los ejercicios de tablas, puesto que al aparecer un número tan alto de variables en los mismos, resultaría imposible representar tablas tan grandes en una pantalla de móvil.



4.2.3.2 Tabs

Las Tabs (pestañas) permiten acceder a distintos fragmentos con un solo click. Resulta especialmente útil en el caso de representaciones horizontales de elementos similares, pero de un tamaño demasiado grande como para caber en la pantalla. Normalmente el cambio de las vistas lo gestiona un ViewPager asociado a las pestañas.

4.2.3.3 ToggleButton

Los ToggleButton son un tipo especial de botones que pueden estar en dos estados. Al pulsar el usuario el botón, éste cambia de estado y al repetirse la pulsación vuelve al estado inicial.

Estos botones son los que se utilizan en las tablas para marcar los estados de las variables. Resultan idóneos para esta función, puesto que al tratarse de circuitos digitales los estados de las variables serán booleanos.

4.2.3.4 Dialogs

Los Dialogs (diálogos) son un widget que consiste en una ventana emergente superpuesta a la vista que los contiene. Los diálogos suelen ofrecer dos botones para aceptar la acción descrita en el mismo o cancelarla. Dependiendo de la función que se le quiera dar a los diálogos, pueden ser simplemente informativos, como en caso de querer una confirmación por parte del usuario de una acción realizada previamente, o incluir otros elementos en su vista, como puede ser un campo de texto rellenable por el usuario, como en el caso de guardado de archivos, o mostrar una lista con opciones. Con la aparición de los fragmentos, Android diseño una clase basada en Fragment para la representación de diálogos: DialogFragment. Haciendo uso de esta clase es posible diseñar diálogos independientes, que puedan ser llamados desde cualquier actividad. Esto ofrece grandes ventajas, sobretodo en el caso de diálogos cuyo diseño se repite ostensiblemente dentro de una misma aplicación, ya que tan sólo es necesario programarlos una vez y, en los lugares del código que sea necesario utilizarlos, crear una instancia de ellos.

4.2.4 Layouts

En Android la disposición y características de los elementos de las interfaces que componen una aplicación pueden ser definidas por dos vías: dinámicamente utilizando las clases java correspondientes a dichos elementos, o de manera estática definiéndolas en archivos XML asociados a la actividad o fragmento que corresponda.

Para la mayoría de interfaces resulta mucho más cómodo y práctico definirlas en XML, puesto que ahorramos memoria y carga de procesamiento a la aplicación y podemos, gracias a la herramienta que ofrece Android Studio "Graphical Layout", ver una representación de la interfaz en el entorno de desarrollo sin necesidad de ejecutar la aplicación en el emulador o un dispositivo.

Dada la enorme diversidad de tamaños de pantalla y resoluciones de los dispositivos que usan Android, definir un solo layout para todos los tipos de pantalla resultaría prácticamente imposible y, por tanto, Android ofrece una solución a este problema.

En el proyecto que engloba la aplicación existen varias carpetas, aparte de la que contiene los archivos .java con las actividades, fragmentos y demás clases que se hayan creado para la aplicación, para incluir recursos como imágenes, bases de datos, valores constantes y diseños de interfaces. A las carpetas de recursos se les puede añadir calificativos adicionales para que al compilar la aplicación Android escoja, dependiendo de la configuración del dispositivo en el que se ejecute, los recursos correspondientes.

En el caso de los layouts la característica del dispositivo que influye de manera más importante es el tamaño de pantalla, por lo que existen calificativos para cada uno de los grupos de tamaños de pantalla expuestos en la sección 3.1.2.1.

4.2.5 Drawables

Como en el caso de los layouts, las imágenes que se utilizan en la aplicación deben estar optimizadas para las características de los dispositivos en los que se va a ejecutar. A diferencia de los layouts, en el caso de las imágenes, la característica del dispositivo que influirá en la correcta representación de las mismas no es el tamaño de pantalla, sino la densidad de pantalla. Por ello los calificativos a añadir a las carpetas donde se alojan las imágenes son los nombres de los grupos de densidades de pantalla definidos en la sección 3.1.2.2.

4.2.6 Assets

Para archivos que no sean imágenes o layouts Android ofrece esta otra carpeta para su almacenamiento. En el caso de esta aplicación se usa para el almacenamiento de la base de datos con los contenidos de los ejercicios y las preguntas de test. Utilizando este método para la adquisición de los datos resulta mucho más sencillo actualizar la aplicación, puesto que sólo se requiere cambiar los campos correspondientes en la base de datos.

4.2.7 Values

En la carpeta values se incluyen varios archivos XML con definiciones estáticas de valores de la aplicación. Al definir estos valores constantes en un archivo por separado, se pueden utilizar en cualquier parte del código de la aplicación utilizando una referencia a los mismos. Resulta muy útil para valores que se repiten mucho, puesto que sólo es necesario cambiarlos en los archivos de la carpeta values y no en cada lugar que son utilizados. Esta carpeta normalmente incluye al menos tres archivos XML: dimens, strings y styles. En el archivo dimens se almacenan valores de dimensiones, tanto de tamaños de letra como de elementos de la interfaz como pueden ser botones o campos de texto y en strings se almacenan todas las cadenas de caracteres de la aplicación. El archivo styles permite definir estilos, esto es, características comunes, de elementos de la interfaz. Haciendo uso de este archivo se pueden definir características específicas de ciertos elementos, heredando sus características básicas de los elementos ya definidos por Android. De este modo sólo es necesario definir una vez cómo va a ser un elemento de la interfaz y utilizar una referencia al estilo definido en los layouts. Con esto se consigue consistencia entre los elementos utilizados en las interfaces y aumenta sobremanera la legibilidad de los layouts. En el caso de nuestra aplicación se han creado estilos para los elementos que más se utilizan, como por ejemplo los botones de los menús, o los ToggleButton de las tablas.

```
<!--Estilos para los botones de los menus  -->
<style name="BotonMenu" parent="@android:style/Widget.Button">
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
    <item name="android:layout_marginBottom">2dp</item>
    <item name="android:textSize">@dimen/font_medium</item>
</style>

<style name="CeldaTabla" parent="@android:style/Widget.TextView">
    <item name="android:layout_width">30dp</item>
    <item name="android:layout_height">30dp</item>
    <item name="android:gravity">center</item>
    <item name="android:layout_weight">1</item>
</style>
```

4.2.8 Archivo Manifest

Por último cabe remarcar el archivo Manifest. Se trata de un archivo XML en el que se definen las actividades que compondrán la aplicación así como sus características, sean estas, por ejemplo, las orientaciones de pantalla que soportan, el estilo del que deben heredar su apariencia y su jerarquía dentro del flujo de la aplicación.

También se definen en él permisos especiales que puede necesitar la aplicación, como la posibilidad de acceder, tanto para lectura como escritura, a la memoria externa (tarjeta SD), los tamaños de pantalla y las versiones del sistema operativo Android que soporta la aplicación.

4.2.9 Librerías de apoyo

Para que sea posible el uso de los fragmentos y ciertos widgets, es necesario incluir en nuestro proyecto las librerías de soporte. Los fragmentos por ejemplo fueron añadidos a Android en la API 11, y por lo tanto no están disponibles en las aplicaciones desarrolladas para APIs anteriores. Android creo librerías de apoyo para solventar este problema y poder dar así acceso a las mismas características de que disponían los usuarios de versiones más modernas del sistema operativo a los usuarios de versiones antiguas.

Para incluir las librerías de apoyo es necesario instalarlas previamente en el entorno de desarrollo utilizando la herramienta que incluye para actualización y descarga de complementos, el SDK Manager. Una vez descargadas, hay que incluir en el archivo build.gradle del proyecto las dependencias correspondientes, siendo en este caso:

```
dependencies {
    compile 'com.android.support:support-v4:22.2.1'
    compile 'com.android.support:appcompat-v7:22.2.1'
    compile 'com.android.support:design:22.2.1'
}
```

Con estos sencillos pasos, adquirimos acceso a las librerías de apoyo y por tanto a toda la potencia que ofrece el sistema operativo para desarrollar nuestra aplicación.

4.3 Estructura de la aplicación

Una vez vistos los elementos que componen las aplicaciones Android, se ha definido la estructura de la aplicación. De cara al usuario es idéntica a las aplicaciones ya desarrolladas para la asignatura, pero ofrece ventajas desde el punto de vista del desarrollador, sobre todo para su actualización y la reutilización de los bloques que la componen.

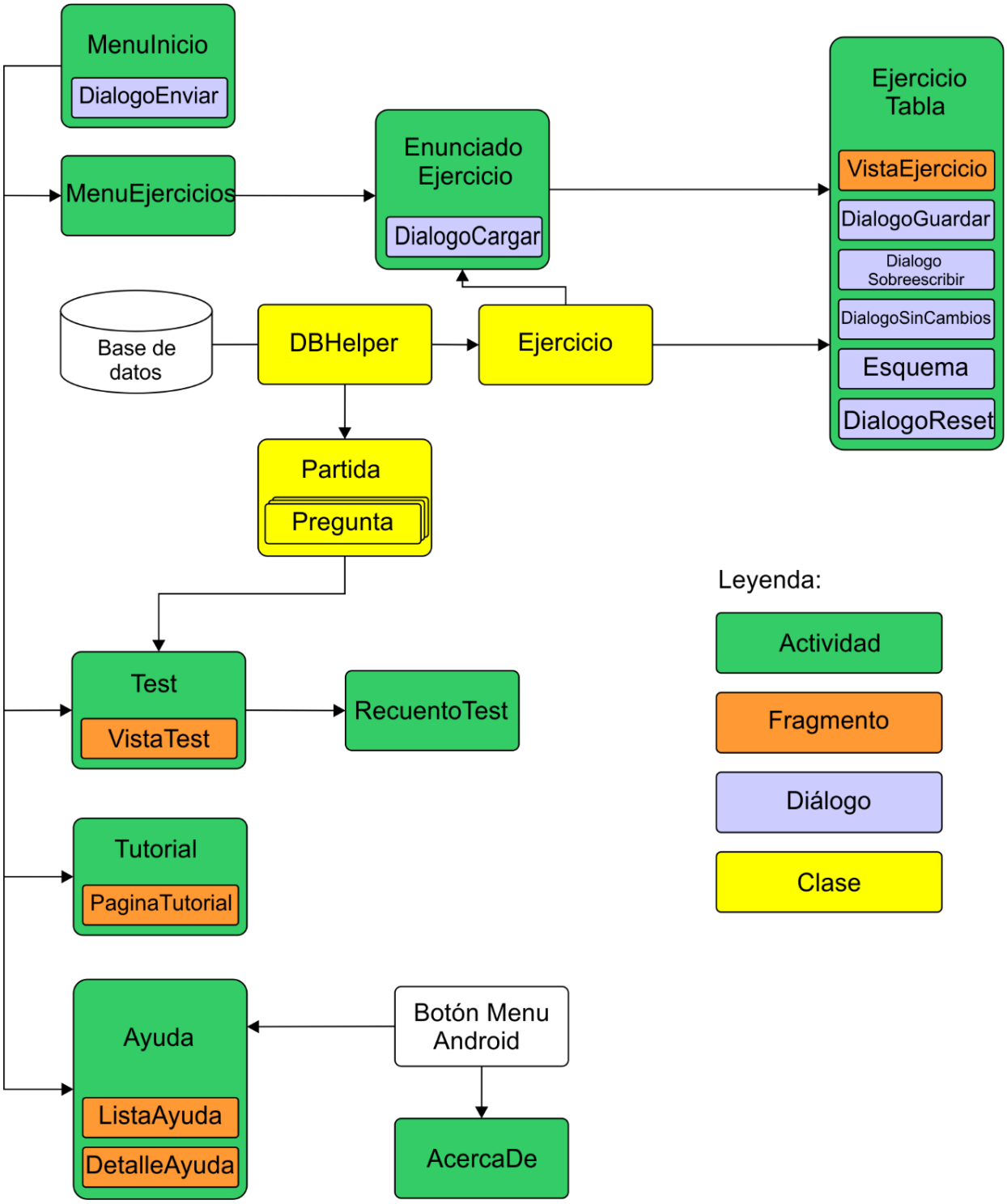


Figura 4-3: Estructura de la aplicación

Como se puede apreciar en la figura, se han definido todos los diálogos en clases separadas, puesto que su uso es común en todas las aplicaciones de la asignatura. Definiéndolos de esta manera, en el desarrollo de aplicaciones futuras de la asignatura, tan sólo será necesario incluir estas clases tipo Dialog en el proyecto, y se podrá hacer uso de ellas sin necesidad de desarrollarlas de nuevo.

También se puede apreciar que los contenidos de los ejercicios y preguntas de test se obtienen de la base de datos, encapsulándolos en clases específicas que gestionan los contenidos de cada elemento. De este modo en caso de necesitar actualizar la aplicación tan sólo será necesario actualizar la base de datos y añadir las imágenes en su carpeta de recursos correspondiente.

4.4 Descripción de las clases

Tras haber visto la estructura de aplicación y los tipos de elementos utilizados se pasará a describir las clases que la componen.

4.4.1 MenuInicio

- **Tipo de clase:** Activity.
- **Layout asociado:** menu_inicio.xml
- **Diálogos o fragmentos asociados:** DialogoEnviar.
- **Descripción de la clase:** MenuInicio es la actividad principal de la aplicación. En el archivo Manifest se le asigna el atributo LAUNCHER, por lo que al presionar el icono de la aplicación en la pantalla principal del dispositivo en el que esté instalada la aplicación, ésta comenzará por esta clase. Se trata de una actividad con una interfaz compuesta por cinco botones, de los cuales cuatro llevan a nuevas actividades y uno llama a la clase DialogoEnviar. Cada botón tiene un método asociado mediante el atributo XML "android:onClick", que inicia la actividad correspondiente, o en el caso del botón enviar, genera una instancia de la clase DialogoEnviar.
- **Diseño de la vista:**

Figura 4-4: Diseño de vista de MenuInicio

4.4.2 DialogoEnviar

- **Tipo de clase:** Dialogo.
- **Layout asociado:** Generado dinámicamente.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta clase crea un diálogo (o pop-up) que se superpone a la actividad que acoge a la clase. Dependiendo de si hay o no ejercicios guardados previamente por el usuario en memoria muestra un mensaje u otro. En el caso de no haber ningún ejercicio, se lo comunica al usuario, y, en caso de haberlo, permite escoger dentro de una lista con CheckButtons los ejercicios que desea enviar. Una vez ha seleccionado los archivos, el propio dialogo se encarga de generar un intent para comunicar al sistema operativo que desea enviar el archivo seleccionado por correo electrónico y éste se encarga de ofrecer al usuario una lista con las aplicaciones de correo que dispone para el envío. Tras que el usuario seleccione el programa de correo electrónico que desea utilizar, automáticamente se genera en el mismo un mensaje con el archivo del ejercicio como archivo adjunto y se avisa en el cuerpo del mensaje de que el receptor del mismo debe mover el archivo una vez descargado a la carpeta de la aplicación para que ésta pueda acceder a él.
- **Diseño de la vista:**

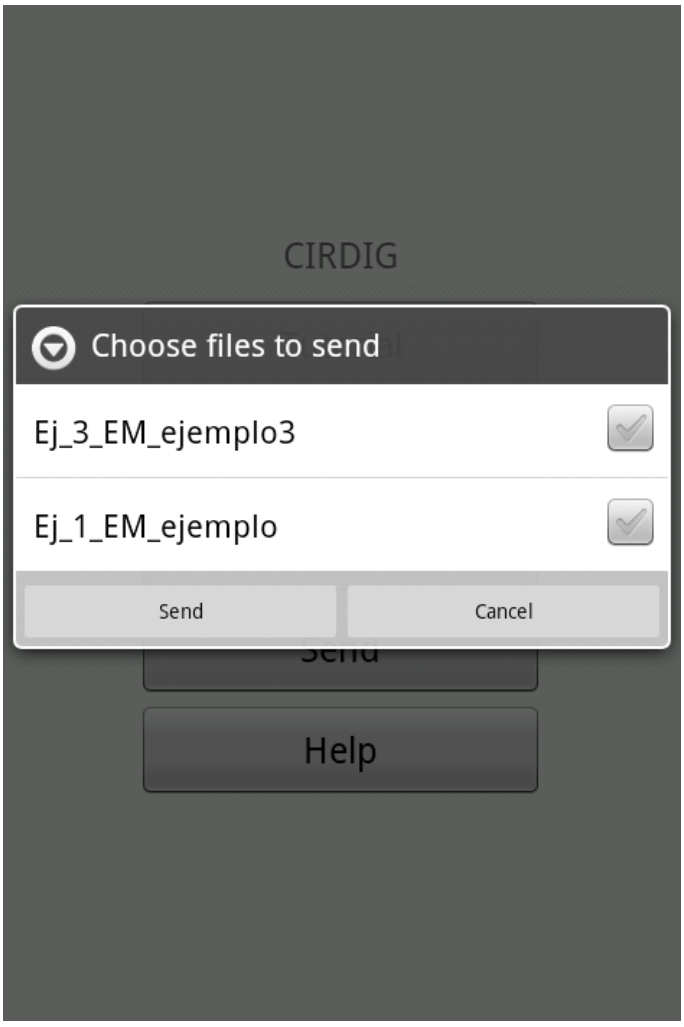


Figura 4-5: Diseño de vista de DialogoEnviar

4.4.3 MenuEjercicios

- **Tipo de clase:** Activity.
- **Layout asociado:** menu_ejercicios.xml
- **Dialogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Como en el caso del menú de inicio, esta clase genera un menú con botones para acceder a los ejercicios de la aplicación. Sin embargo, en este caso siempre llama a la misma actividad, EnunciadoEjercicio, pasándole como parámetro en el intent el número del ejercicio asociado al botón presionado por el usuario. La gestión de las pulsaciones de los botones se realiza también utilizando el atributo XML "android:onClick".
- **Diseño de la vista:**

Figura 4-6: Diseño de vista de MenuEjercicios

4.4.4 EnunciadoEjercicio

- **Tipo de clase:** Activity.
- **Layout asociado:** enunciado_ejercicio.xml
- **Diálogos o fragmentos asociados:** DialogoCargar.
- **Descripción de la clase:** Esta actividad recibe al ser llamada el número del ejercicio y con él genera una clase tipo Ejercicio, de la que obtiene el enunciado y la imagen del esquema del circuito a analizar. Ofrece dos botones para que interactúe el usuario, uno para comenzar el ejercicio en cuestión, que llama a la actividad EjercicioTabla a través de un intent pasándole como parámetro el número del ejercicio, y otro que llama al diálogo DialogoCargar.
- **Diseño de la vista:**

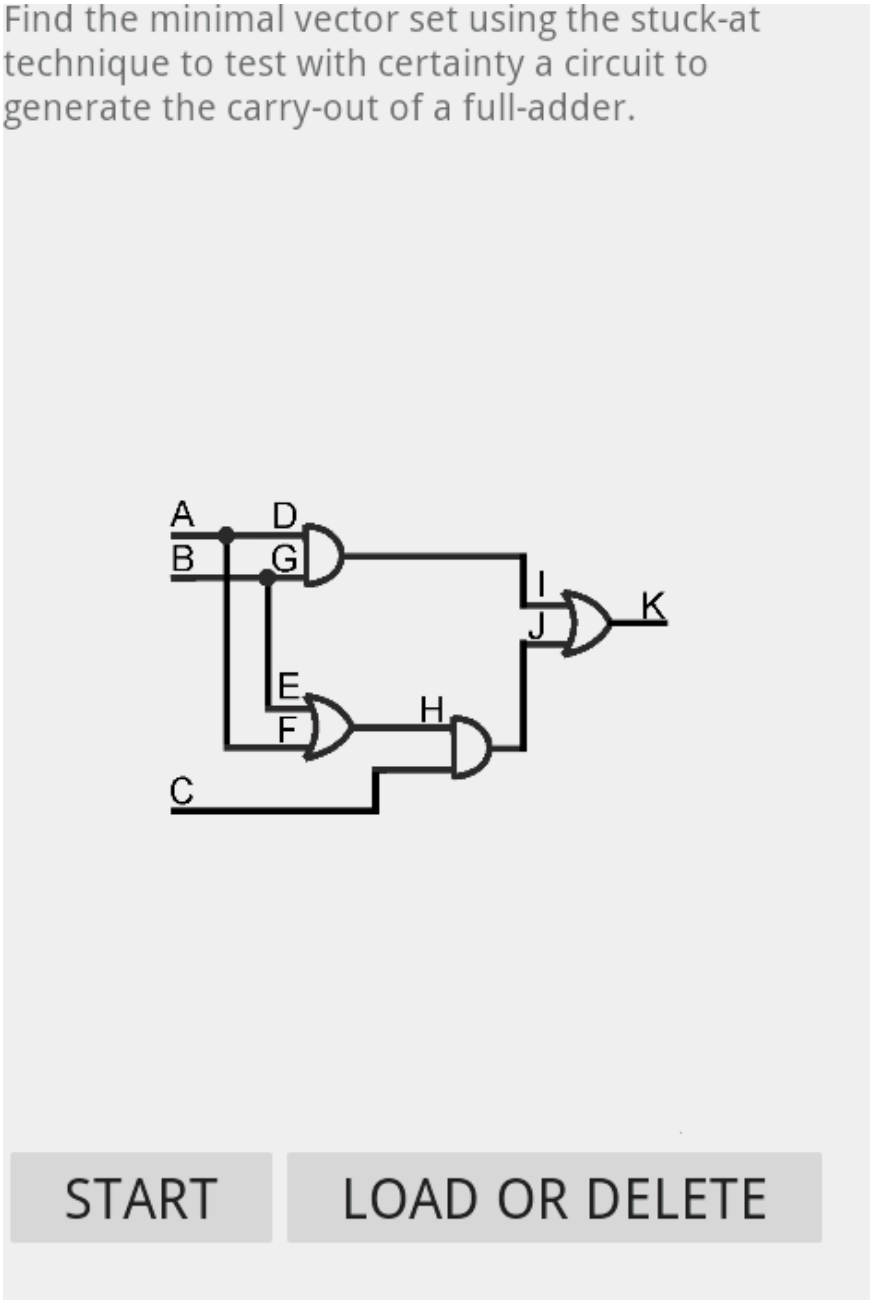


Figura 4-7: Diseño de vista de EnunciadoEjercicio

4.4.5 DialogoCargar

- **Tipo de clase:** Dialogo.
- **Layout asociado:** Generado dinámicamente.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta clase crea un diálogo (o pop-up) que se superpone a la actividad que acoge a la clase. Dependiendo de si hay o no archivos del ejercicio en el que se encuentra guardados previamente por el usuario en memoria muestra un mensaje u otro. En el caso de no haber ningún ejercicio, se lo comunica al usuario, y, en caso de haberlo, permite escoger dentro de una lista los ejercicios que desea cargar o borrar. En el caso de la carga de un ejercicio el propio diálogo se encarga de almacenar en SharedPreferences el estado de las variables del ejercicio y llama a la clase EjercicioTabla para que el usuario pueda comenzar con la resolución del ejercicio.
- **Diseño de la vista:**

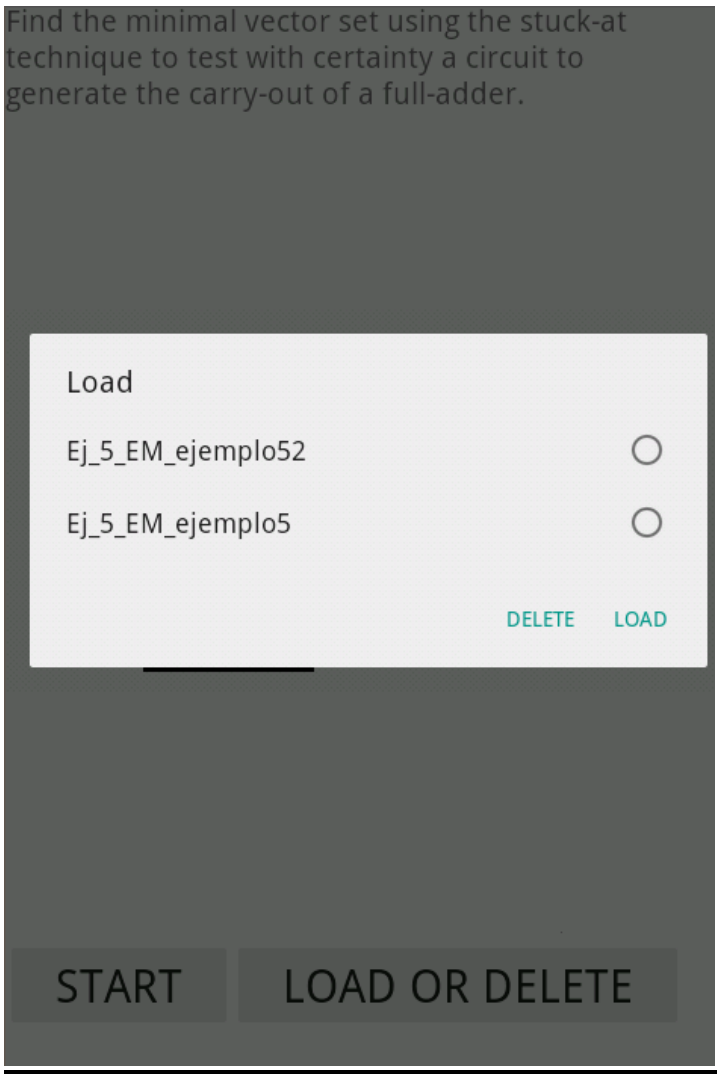


Figura 4-8: Diseño de vista de DialogoCargar

4.4.6 EjercicioTabla

- **Tipo de clase:** Activity.
- **Layout asociado:** ejercicio_tabla.xml
- **Diálogos o fragmentos asociados:** VistaEjercicioTabla, DialogoReset, DialogoGuardar, DialogoSobreescribir, DialogoSinCambios, Esquema.
- **Descripción de la clase:** Esta actividad genera la interfaz de los ejercicios tipo tabla y gestiona las acciones que se deben realizar a consecuencia de las interacciones del usuario.

  La interfaz consiste en una tabla con tabs, para permitir navegar horizontalmente por la misma dada la longitud que suelen tener los ejercicios del tema que abarca la aplicación. Las acciones asociadas a los clicks sobre las pestañas (tabs), son gestionadas por un ViewPager, que también gestiona los deslizamientos por parte del usuario sobre la parte de la tabla que alberga las variables. Esta parte de la tabla se crea utilizando fragmentos tipo VistaEjercicioTabla anidados, cuya creación y destrucción, así como las animaciones al pasar de uno a otro, son gestionadas también por el ViewPager.

  Debajo de la tabla están los cinco botones que aparecen en las aplicaciones realizadas previamente que contienen ejercicios de tipo tabla. Estos cinco botones permiten al usuario realizar varias acciones, siendo éstas gestionadas en su mayoría por diálogos, para ofrecer la mayor modularidad y reusabilidad posible.

  El primer botón, "Scheme", permite al usuario ver a pantalla completa el esquema del circuito a analizar en el ejercicio. Ofrece la misma funcionalidad que en las aplicaciones previas, pero lo hace llamando a la clase Esquema de tipo diálogo, facilitando así la gestión del ciclo de vida de la actividad además de las razones expuestas en el párrafo anterior.

  El segundo botón, "Save", llama a la clase DialogoGuardar, de tipo diálogo, que gestiona el guardado del estado de las variables en un archivo para su posterior carga o envío por correo electrónico. Dependiendo de si ya existen archivos con el nombre introducido por el usuario para el guardado, el diálogo se comunica a través de una interfaz con la actividad para que ésta llame a los métodos correspondientes de los fragmentos VistaEjercicioTabla donde se encuentran las variables. Una vez confirmado el guardado, se llama al método guardadoEnArchivo, que realiza el guardado en memoria y comunica el guardado exitoso al usuario a través de un mensaje tipo Toast. Éste método pertenece a la actividad, puesto que cada ejercicio contiene distinta cantidad de variables y resultaría un gasto enorme de memoria referenciar otra vez todas ellas en el diálogo.

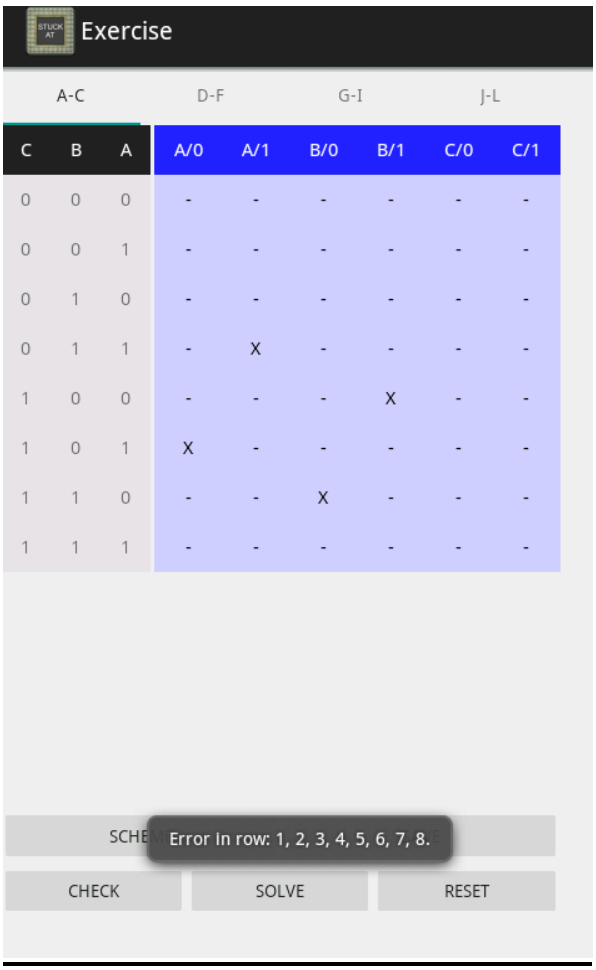
  El tercer botón, "Check", comprueba el estado de las variables, comparándolas con la solución del ejercicio. Para ello llama al método check, que comprueba las variables alojadas en los fragmentos VistaEjercicioTabla creados por la actividad, y los compara a la solución del ejercicio. La comprobación se realiza dentro de cada fragmento, recorriendo la matriz de variables de cada uno por filas y comparando sus valores con la matriz solución. En caso de encontrar diferencias entre ambas matrices, comunica al usuario con un mensaje tipo Toast, las filas en las que hay

errores, o en caso de ser idénticas, también se lo comunica con un mensaje del mismo tipo.

El cuarto botón, “Solve”, hace un proceso muy parecido al botón “Check”, pero esta vez asigna a las variables de cada fragmento VistaEjercicioTabla los valores correspondientes alojados en la matriz de soluciones del ejercicio. Una vez completada la asignación se lo comunica al usuario a través de un mensaje tipo Toast.

Por último, el quinto botón, “Reset”, llama al método reset de la actividad, dónde se comprueba si el usuario ha introducido cambios en las variables y consecuentemente, genera un diálogo, en caso de no haberlo hecho, comunicándole que no ha habido cambios en las variables, u otro diálogo, pidiéndole una confirmación de que desea restablecer el ejercicio al estado inicial. Una vez el usuario acepta la confirmación, se ejecuta el método onDialogoResetAceptado en el que se recorren los fragmentos con las variables y se les asigna el valor inicial a todas ellas.

- **<u>Diseño de la vista:</u>**



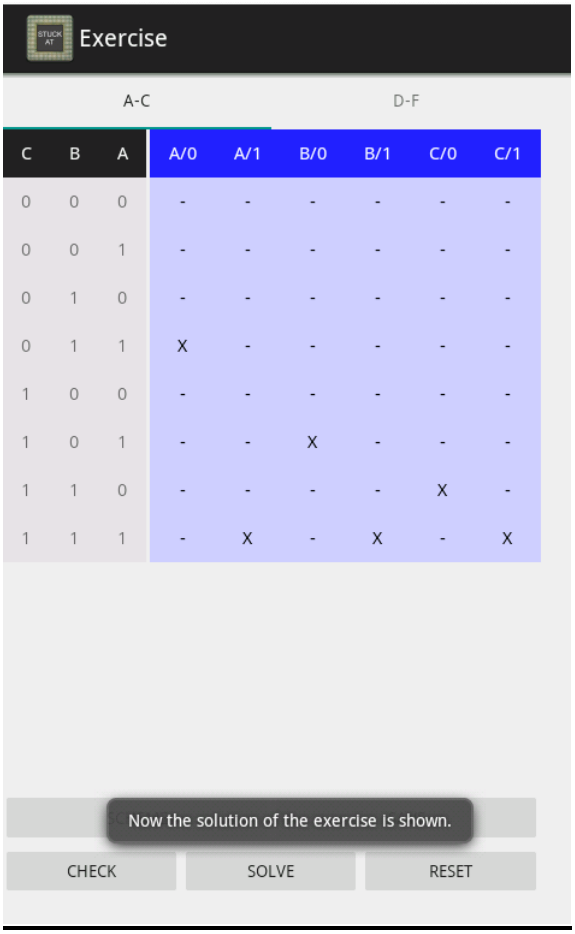


Figura 4-9: Diseño de vista de EjercicioTabla

4.4.7 VistaEjercicioTabla

- **Tipo de clase:** Fragmento.
- **Layout asociado:** vista_ejercicio_tabla.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este fragmento se compone de una vista en forma de tabla donde se alojan una parte de las variables del ejercicio. La actividad que lo aloja, EjercicioTabla, genera a través de su ViewPager tantos fragmentos de este tipo como necesite para la cantidad de variables que tenga el ejercicio seleccionado. Se ha diseñado esta clase de manera que aloje siempre seis columnas de variables, dado que las filas siempre son ocho, debido a que los bits de entrada siempre son tres. Posee métodos para la eliminación de las columnas sobrantes en las vistas que no necesiten las seis columnas, así como métodos para la comprobación del estado de las variables, el reinicio y la asignación de valores desde la matriz de soluciones de las mismas. Estos métodos son llamados por la actividad que aloja al fragmento para ejecutar las acciones realizadas por el usuario al presionar los botones de la interfaz de la actividad.
- **Diseño de la vista:**

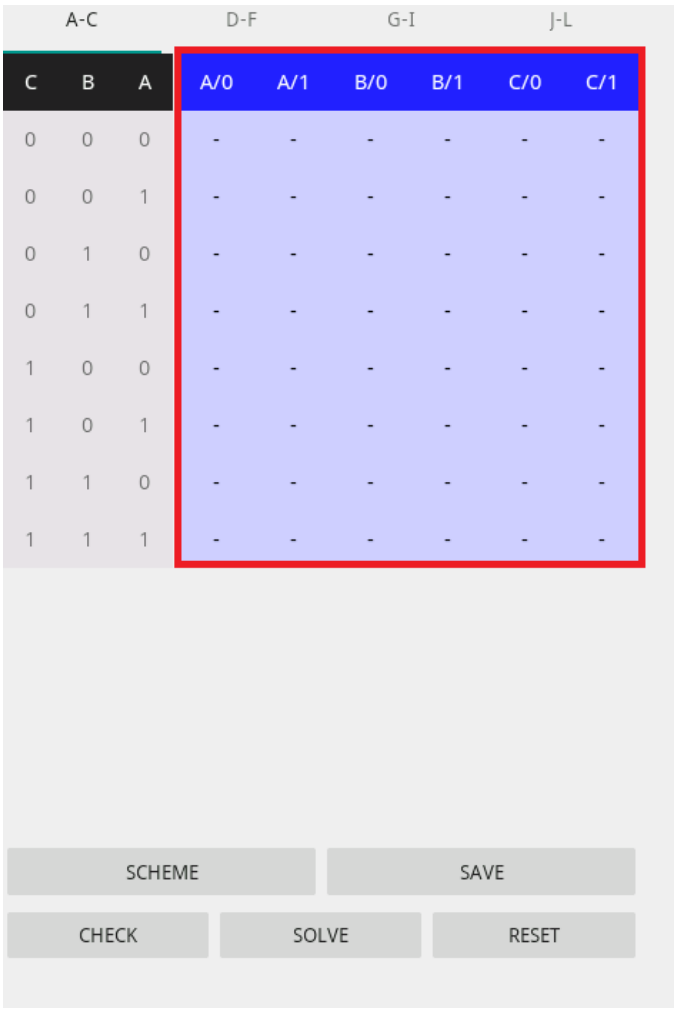


Figura 4-10: Diseño de vista de VistaEjercicio

4.4.8 Esquema

- **Tipo de clase:** Diálogo.
- **Layout asociado:** esquema.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este diálogo, a diferencia de la mayoría de los usados en la aplicación, dispone de un layout asociado, compuesto de un ImageView que abarca todo el espacio disponible de la pantalla. De este modo, al ser llamado el método de la actividad que lo aloja que lo inicia, se superpone una imagen del esquemático del circuito a resolver en el ejercicio sobre la interfaz de la actividad. El diálogo no dispone de botones para la interacción con el usuario para emular el comportamiento del botón "Scheme" de las aplicaciones realizadas previamente, ya que la única manera en la que puede el usuario cancelar la vista y volver al ejercicio es presionar el botón "Back" del dispositivo.
- **Diseño de la vista:**

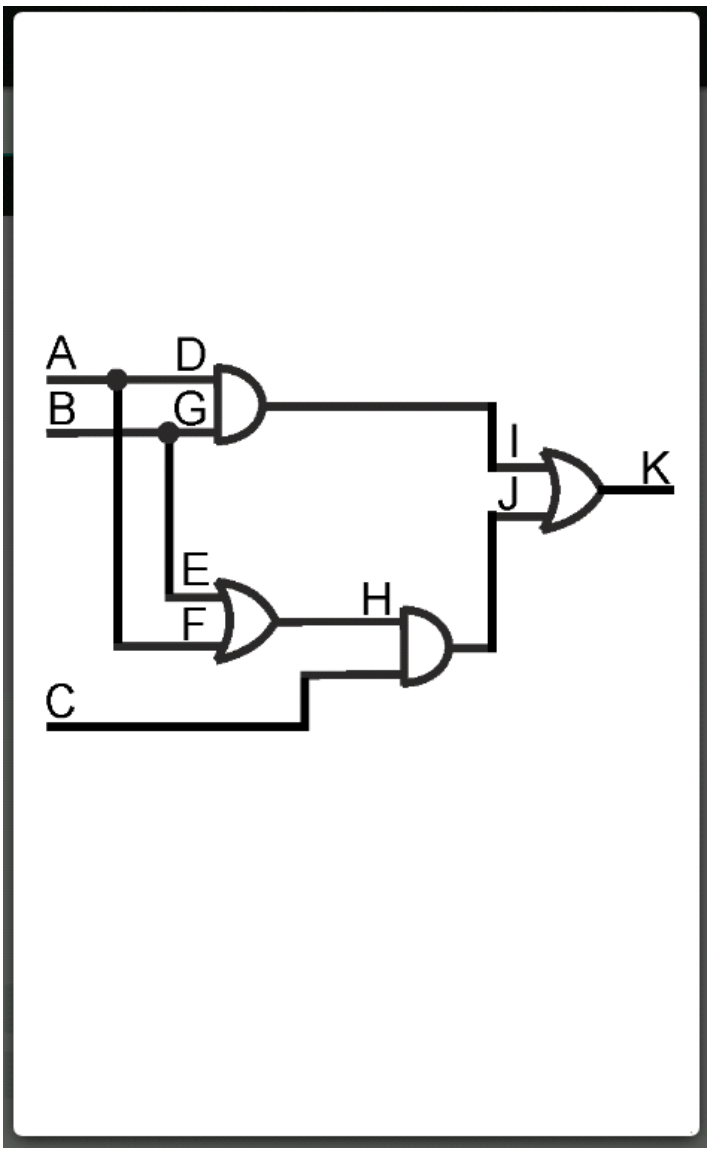


Figura 4-11: Diseño de vista de Esquema

4.4.9 DialogoGuardar

- **Tipo de clase:** Diálogo.
- **Layout asociado:** guardar_en_archivo.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este diálogo, igual que en el caso del diálogo Esquema, dispone de un layout asociado. Esto se debe a que los métodos de la clase Dialog, permiten generar dinámicamente distintos diseños de diálogos, pero no añadir un campo de texto rellenable por el usuario. El diálogo ofrece al usuario un campo de texto para que introduzca el nombre que desea asignarle al archivo donde se guardara el estado del ejercicio, añadiéndole automáticamente un prefijo que usa la aplicación para identificar el número de ejercicio guardado y su localización en memoria, sea esta memoria interna o externa (tarjeta SD). En caso de que el nombre introducido por el usuario sea igual que el de algún archivo guardado previamente, el diálogo se lo comunica a la actividad, para que ésta gestione la llamada a los diálogos correspondiente para avisar de ello al usuario.
- **Diseño de la vista:**

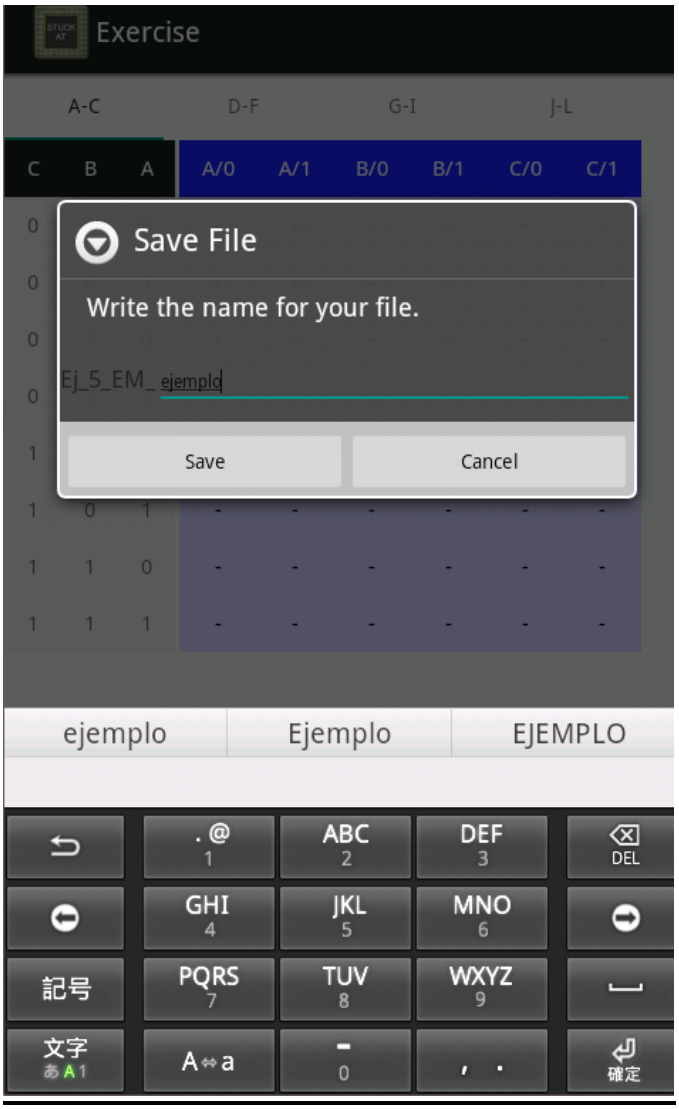


Figura 4-12: Diseño de vista de DialogoGuardar

4.4.10 DialogoSobreescibir

- **Tipo de clase:** Diálogo.
- **Layout asociado:** Generado dinámicamente
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este diálogo es llamado por la actividad en caso de que el nombre introducido por el usuario coincida con el nombre de alguno de los archivos guardados previamente. Le pide confirmación al usuario de que desea sobrescribir el archivo antiguo con el nuevo.
- **Diseño de la vista:**

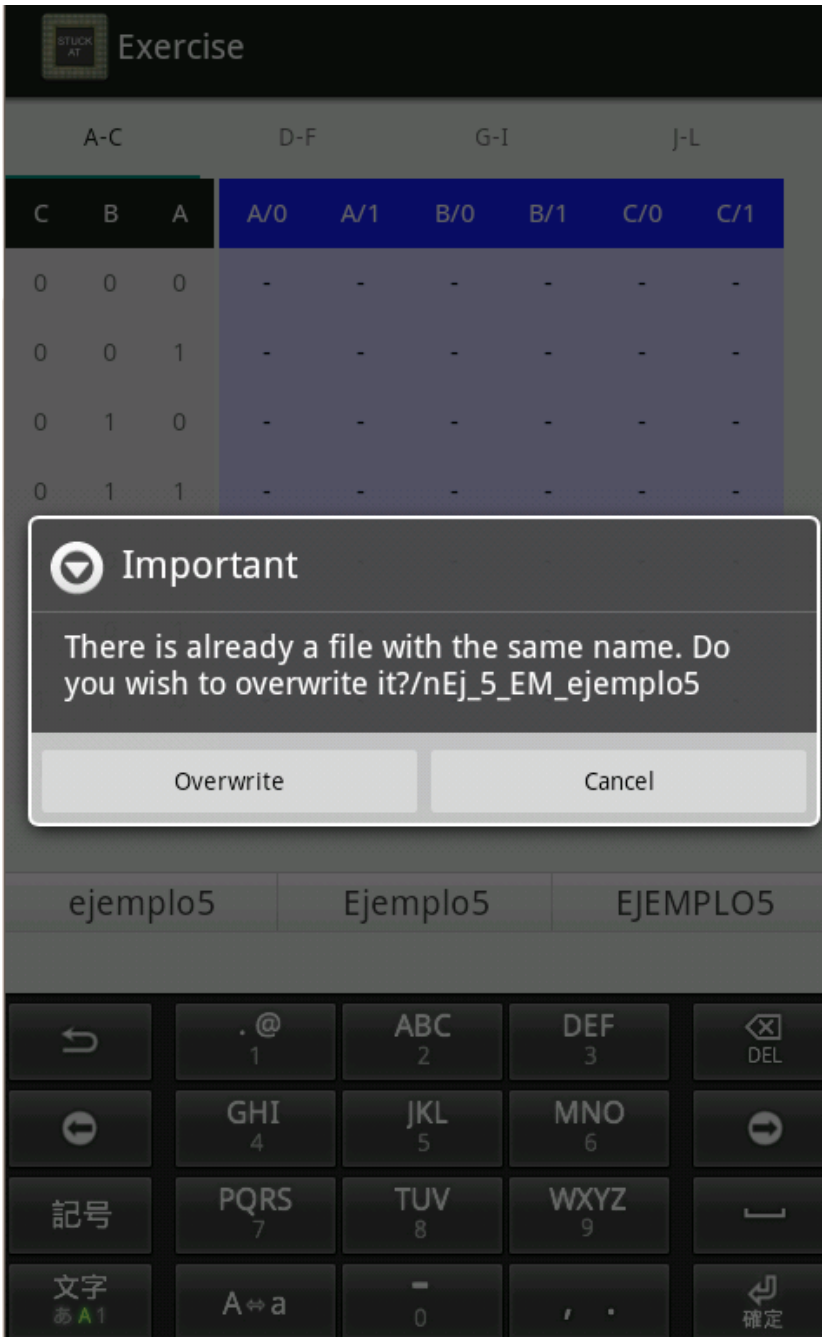


Figura 4-13: Diseño de vista de DialogoSobreescribir

4.4.11 DialogoReset

- **Tipo de clase:** Diálogo.
- **Layout asociado:** Generado dinámicamente.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este diálogo avisa al usuario de que se va a devolver el ejercicio al estado inicial y le pide confirmación de ello.
- **Diseño de la vista:**

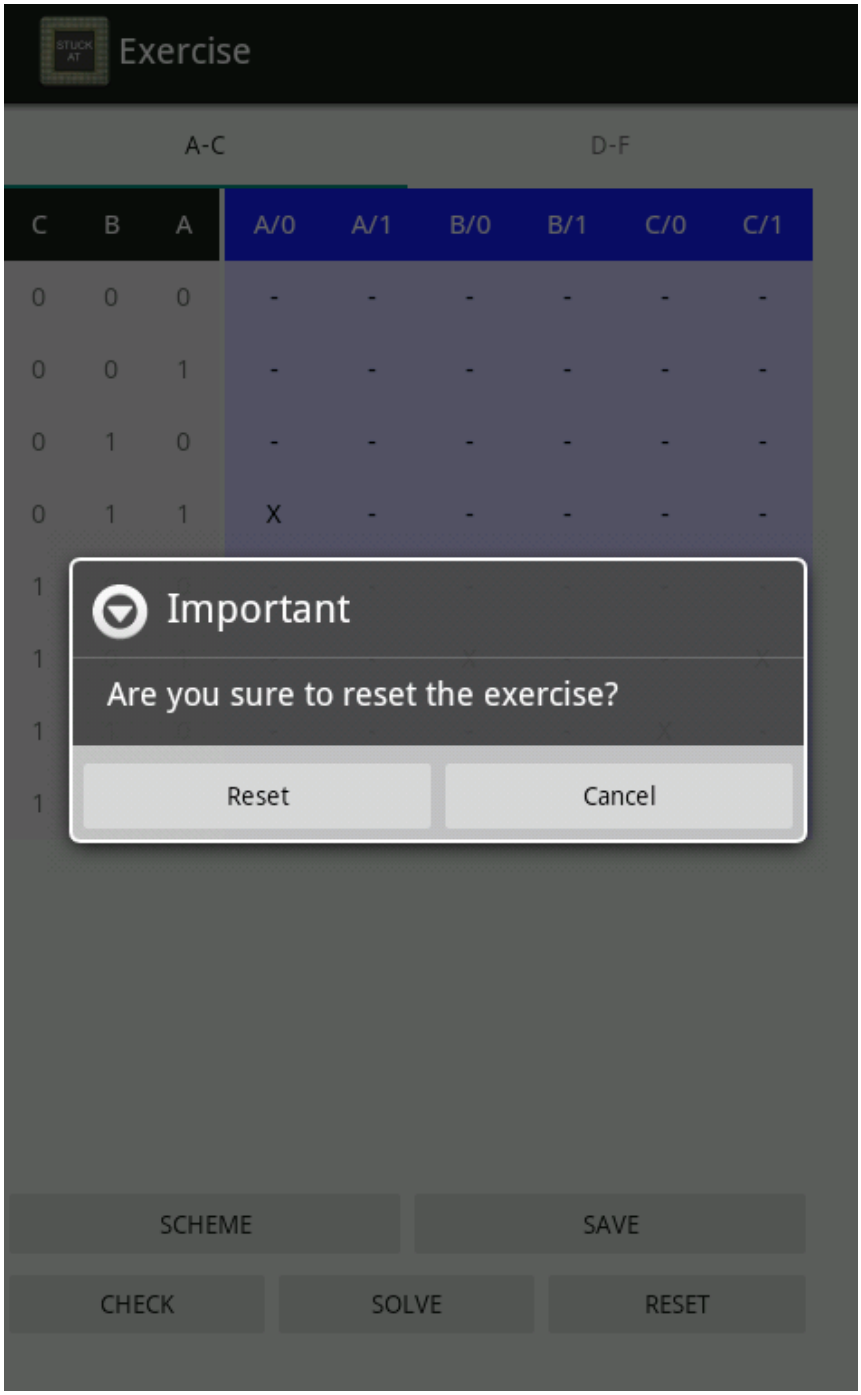


Figura 4-14: Diseño de vista de DialogoReset

4.4.12 DialogoSinCambios

- **Tipo de clase:** Diálogo.
- **Layout asociado:** Generado dinámicamente
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** En caso de que el usuario presione el botón reset de la actividad y ésta no encuentre diferencias en las variables respecto del estado inicial de las mismas, este diálogo le avisa de este hecho.
- **Diseño de la vista:**

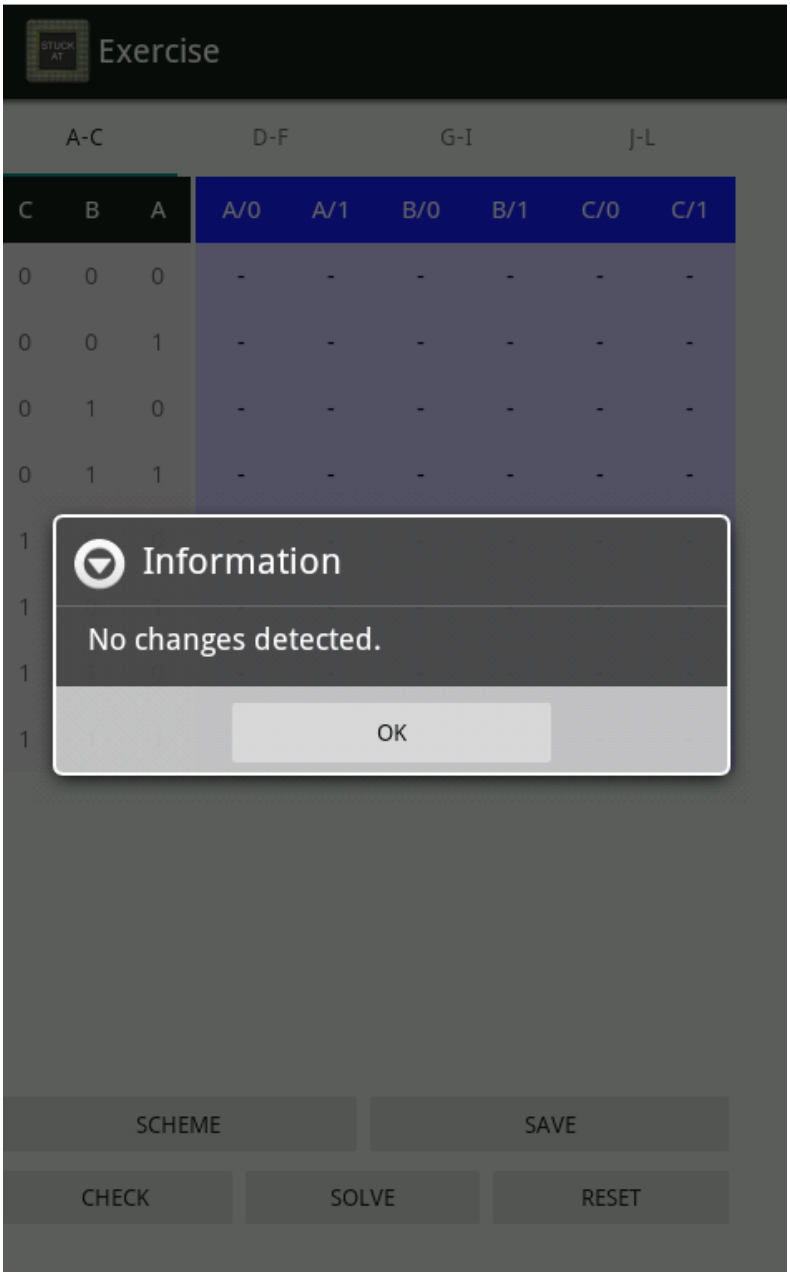


Figura 4-15: Diseño de vista de DialogoSinCambios

4.4.13 Test

- **Tipo de clase:** Activity.
- **Layout asociado:** test.xml
- **Diálogos o fragmentos asociados:** VistaTest.
- **Descripción de la clase:** Esta actividad inicia el test de cuatro opciones de la aplicación. Representa cada pregunta del test a través de fragmentos tipo VistaTest anidados gestionados por un ViewPager Obtiene los datos de las preguntas a través de la clase Partida, que alberga en su interior una matriz con los contenidos de las preguntas correspondientes. Tiene un botón que permite finalizar el test desde cualquiera de las preguntas, que llama a la actividad RecuentoTest, pasándole como parámetros el número de respuestas correctas y erróneas albergadas en la clase Partida.
- **Diseño de la vista:**

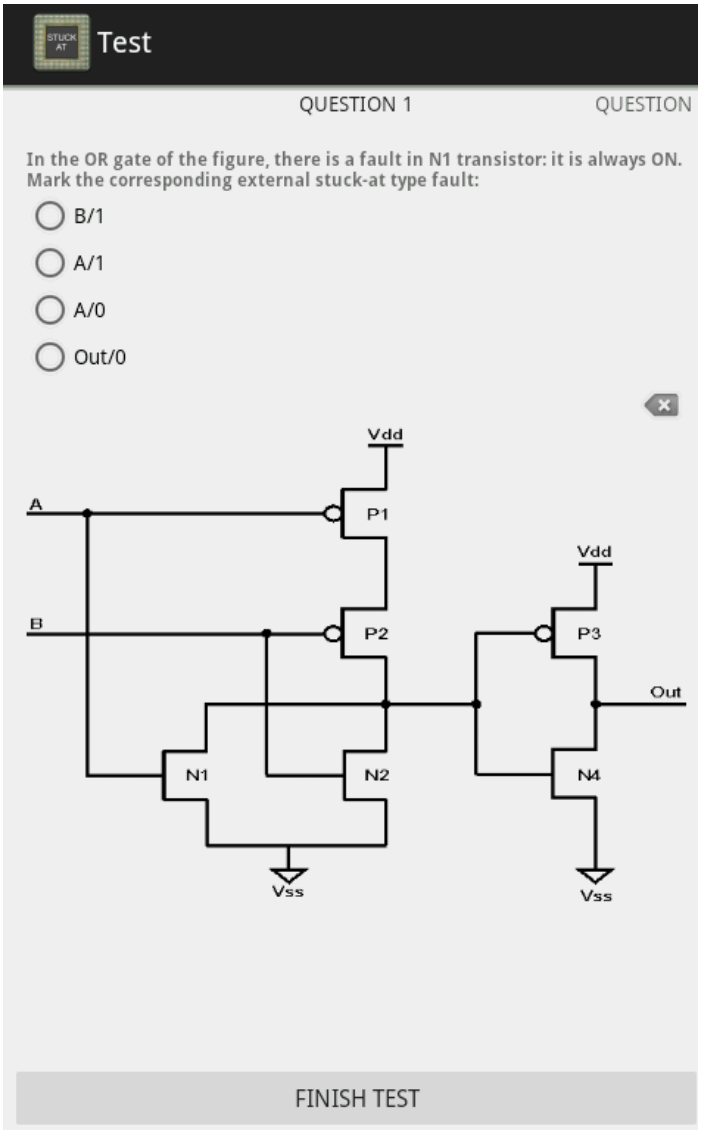


Figura 4-16: Diseño de vista de Test

4.4.14 VistaTest

- **Tipo de clase:** Fragmento.
- **Layout asociado:** vista_test.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este fragmento genera la interfaz de cada pregunta del test, obteniendo sus contenidos de la clase Partida de la actividad que lo aloja. Esta interfaz se compone de un campo de texto donde está el enunciado de la pregunta correspondiente, cuatro RadioButton englobados en un RadioGroup, un botón para borrar la respuesta seleccionada y una imagen con el esquemático del circuito sobre el que se pregunta en caso de que sea necesario. Los RadioButton son un tipo especial de botones que ofrece Android, que son autoexcluyentes si están dentro de un mismo RadioGroup. Dado que el test es de respuesta única resultan muy útiles para representar las respuestas y gestionar las selecciones realizadas por el usuario.
- **Diseño de la vista:**

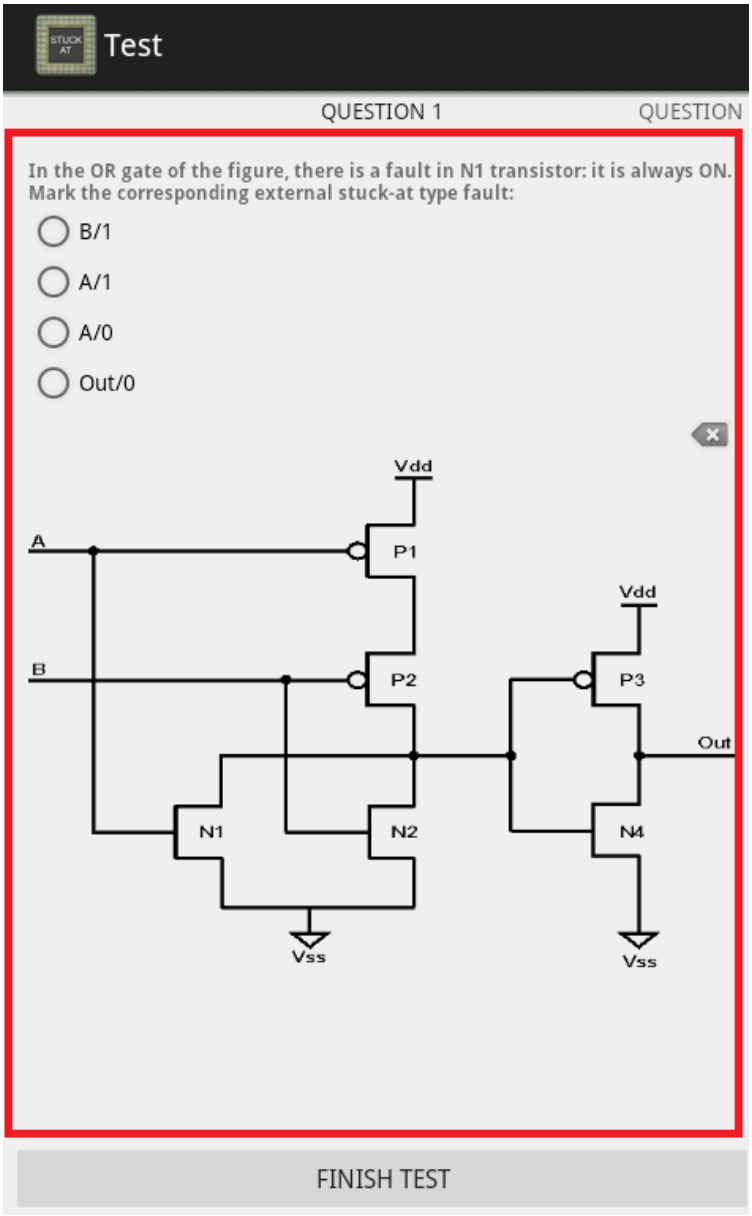


Figura 4-17: Diseño de vista de VistaTest

4.4.15 RecuentoTest

- **Tipo de clase:** Activity.
- **Layout asociado:** recuento_test.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta actividad recibe al ser llamada las respuestas correctas y erróneas del test realizado, así como la cantidad de preguntas con las que contaba el test. Con estos datos, establece el porcentaje total conseguido en el test, dándole a las respuestas correctas 1 punto y a las erróneas -0.5 puntos. Se trata de una actividad meramente informativa y por tanto no ofrece elementos para la interacción con el usuario. En esta actividad se gestiona específicamente el comportamiento del botón "Back" del dispositivo, dado que se desea que el usuario regrese al menú de inicio si lo presiona y no al test.
- **Diseño de la vista:**

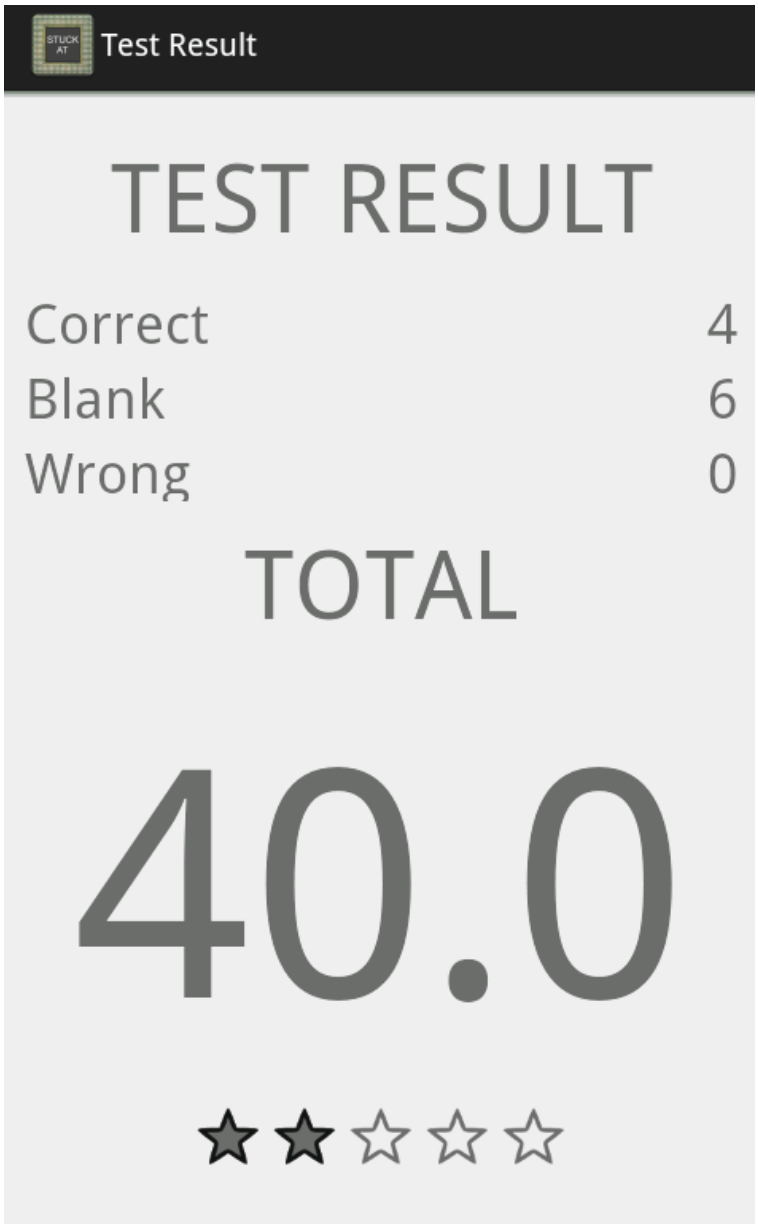


Figura 4-18: Diseño de vista de RecuentoTest

4.4.16 Tutorial

- **Tipo de clase:** Activity.
- **Layout asociado:** pager_tutorial.xml
- **Diálogos o fragmentos asociados:** PaginaTutorial.
- **Descripción de la clase:** Esta actividad se compone de un ViewPager para la gestión de las interacciones del usuario al pasar las páginas del tutorial. Cada página es creada generando un fragmento tipo PaginaTutorial por el ViewPager. Puesto que el escalado de imágenes es muy costoso en memoria para Android, se ha creado un método en la actividad para realizarlo en un proceso aparte de aquél en el que se encuentra la actividad. De este modo se evita que el sistema operativo finalice la aplicación por falta de memoria en los dispositivos más antiguos y con menos capacidad de memoria y procesamiento.
- **Diseño de la vista:**

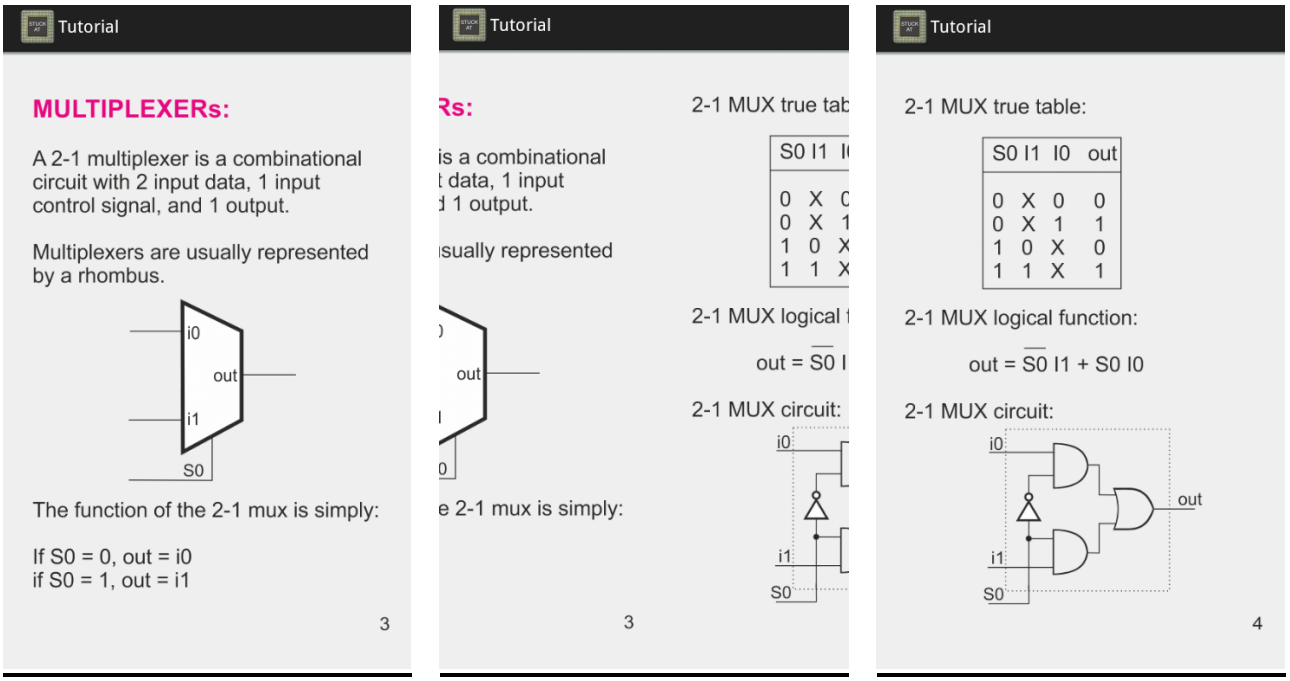


Figura 4-19: Diseño de vista de Tutorial

4.4.17 PaginaTutorial

- **Tipo de clase:** Fragmento.
- **Layout asociado:** pagina_tutorial.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** El fragmento PaginaTutorial se compone simplemente de una vista compuesta por un ImageView que abarca toda la superficie disponible de la pantalla. No dispone de métodos adicionales a los proporcionados por Android, ya que su función es simplemente informativa.
- **Diseño de la vista:**

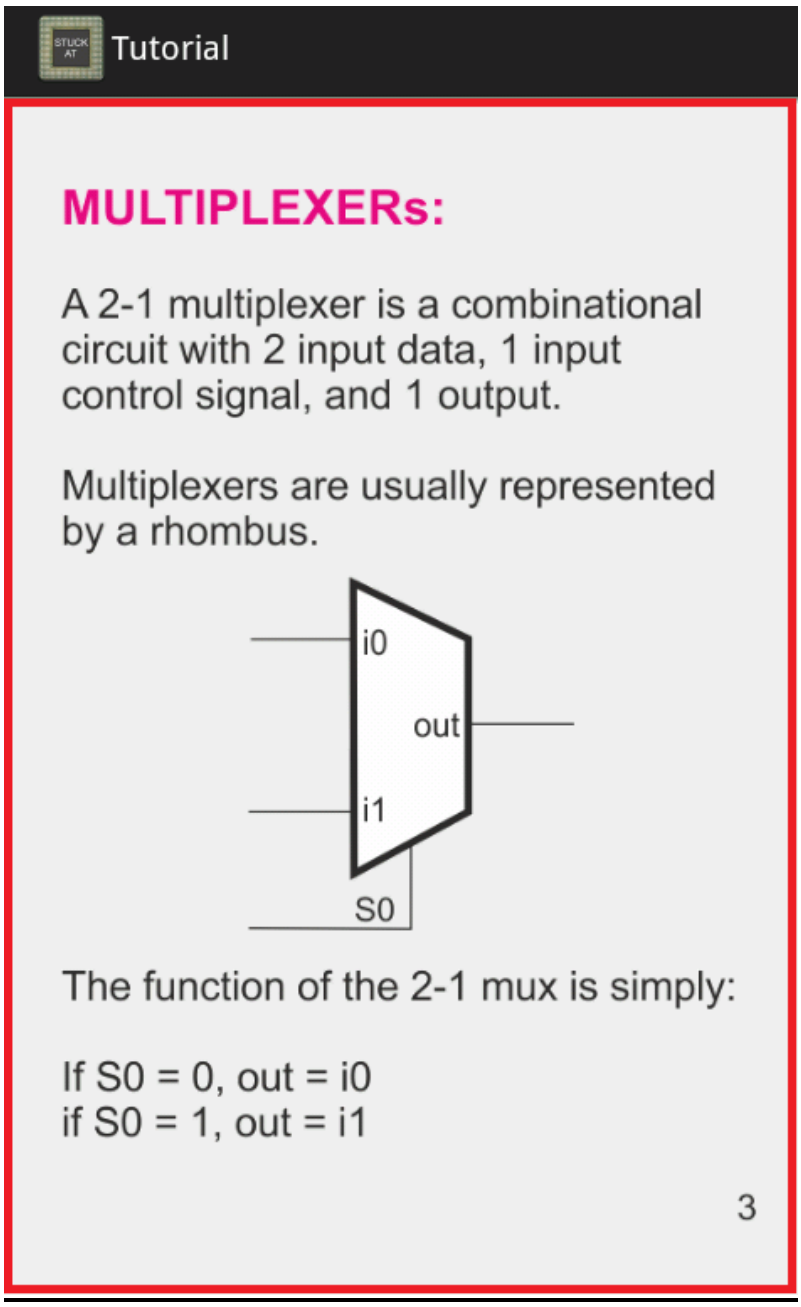


Figura 4-20: Diseño de vista de PaginaTutorial

4.4.18 Ayuda

- **Tipo de clase:** Activity.
- **Layout asociado:** ayuda.xml
- **Diálogos o fragmentos asociados:** ListaAyuda, DetalleAyuda.
- **Descripción de la clase:** Esta actividad incluye y gestiona dos tipos de fragmentos: ListaAyuda y Detalle Ayuda. Se trata de una actividad que tan sólo ofrece un marco donde alojar estos dos fragmentos, ya que ellos son los responsables de gestionar las interacciones del usuario con los elementos que ofrecen.
- **Diseño de la vista:**

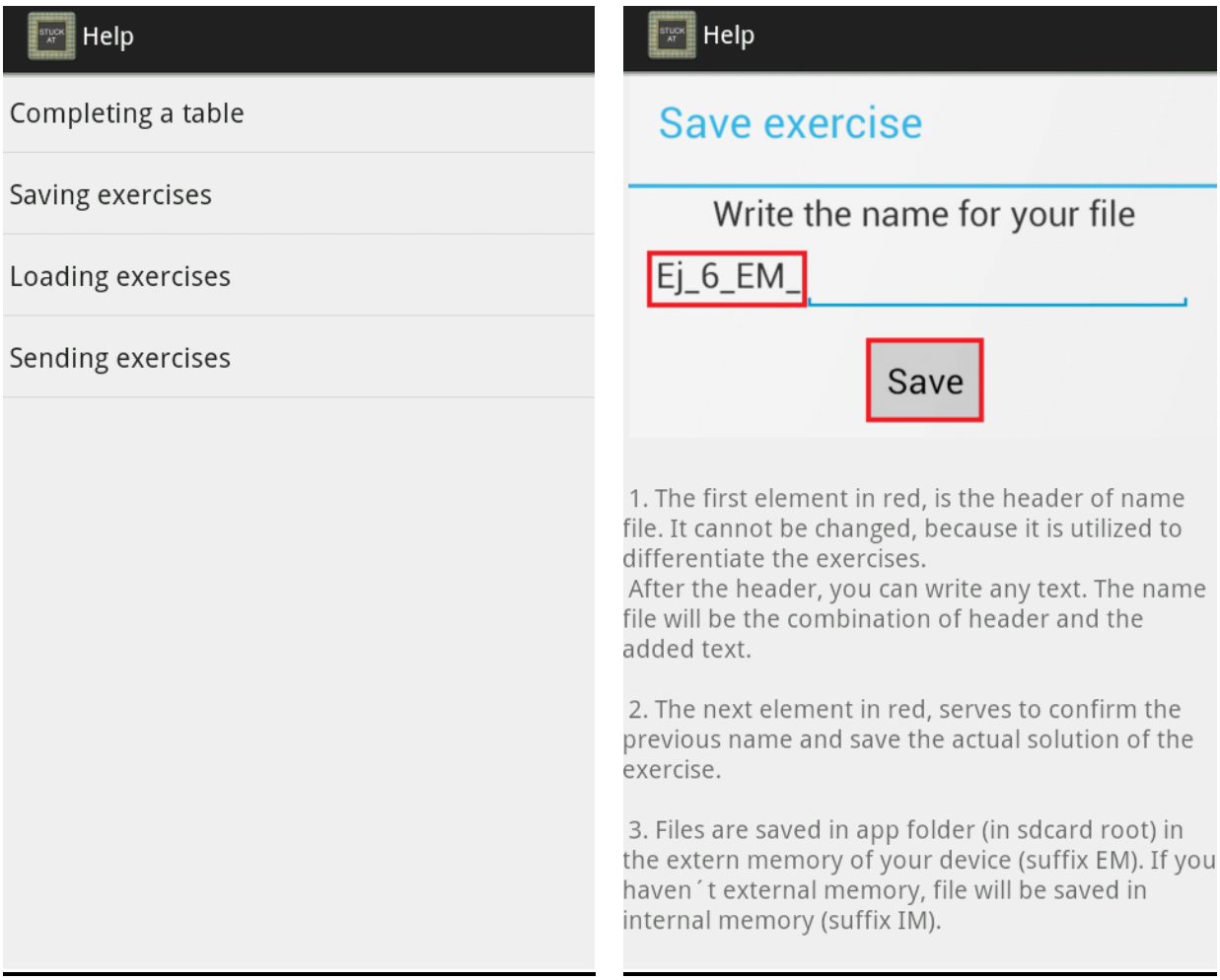


Figura 4-21: Diseño de vista de Ayuda

4.4.19 ListaAyuda

- **Tipo de clase:** Fragmento.
- **Layout asociado:** lista_ayuda.xml
- **Diálogos o fragmentos asociados:** DetalleAyuda.
- **Descripción de la clase:** Este fragmento genera una lista con las secciones correspondientes del menú de ayuda de la aplicación. Al presionar el usuario la sección deseada, genera un fragmento tipo DetalleAyuda, que ocupa todo el espacio disponible de la pantalla.
- **Diseño de la vista:**

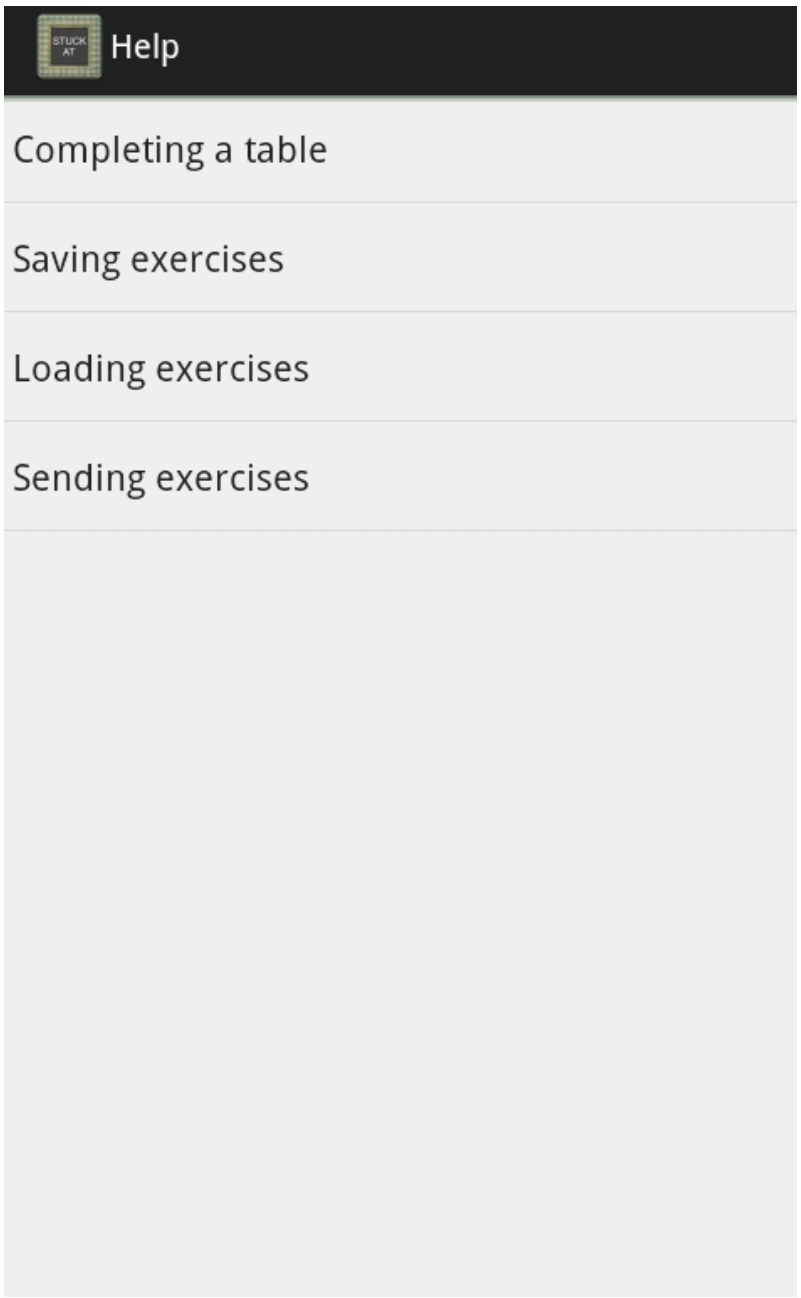


Figura 4-22: Diseño de vista de ListaAyuda

4.4.20 DetalleAyuda

- **Tipo de clase:** Fragmento.
- **Layout asociado:** ayuda_cargar.xml, ayuda_guardar.xml, ayuda_enviar.xml, ayuda_tabla.xml.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Este fragmento muestra a pantalla completa la vista correspondiente de la sección del menú de ayuda seleccionada por el usuario. Recibe del fragmento ListaAyuda el índice del elemento seleccionado y genera su vista escogiendo el layout correspondiente de una matriz interna donde se encuentran referencias a todos ellos.
- **Diseño de la vista:**

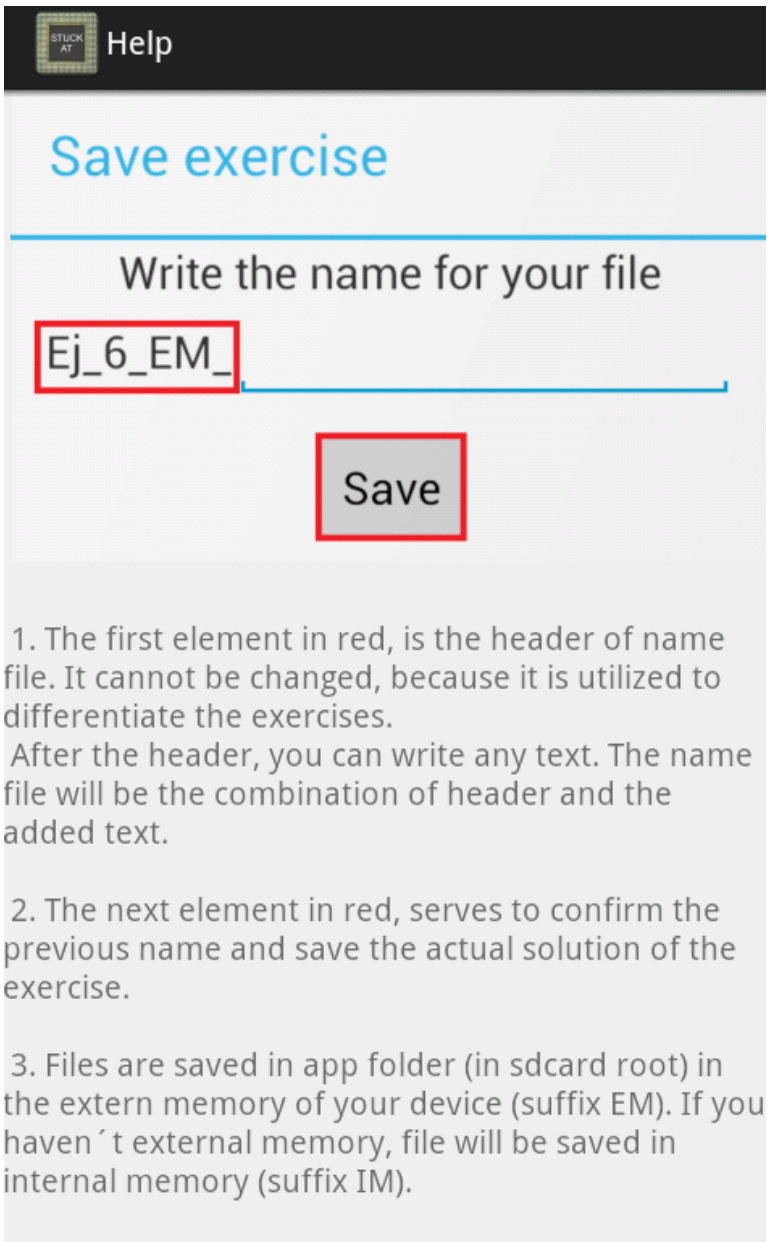


Figura 4-23: Diseño de vista de DetalleAyuda

4.4.21 AcercaDe

- **Tipo de clase:** Activity.
- **Layout asociado:** acerca_de.xml
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta actividad muestra la vista de los datos del departamento responsable de la creación de la aplicación, su desarrollador y demás personas implicadas en el proyecto, su versión y el año en el cual fue desarrollada.
- **Diseño de la vista:**

Figura 4-24: Diseño de vista de AcercaDe

4.4.22 DBHelper

- **Tipo de clase:** Clase estándar.
- **Layout asociado:** Ninguno.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta clase se encarga de crear la base de datos interna en el dispositivo utilizando el archivo con los datos de la misma alojado en la carpeta assets del proyecto. Posee también métodos para recorrer la base de datos y entregar a las clases Pregunta y Ejercicio los contenidos de la base de datos.
- **Diseño de la vista:** No dispone de vista.

4.4.23 Ejercicio

- **Tipo de clase:** Clase estándar.
- **Layout asociado:** Ninguno.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** La clase Ejercicio, encapsula todos los contenidos necesarios para la representación y gestión de los ejercicios de la aplicación. Almacena campos como el enunciado, el número de variables, el nombre de las mismas, la solución del ejercicio y el nombre de la imagen del esquemático asociada. Obtiene estos contenidos de la base de datos a través de la clase DBHelper, de manera que tan sólo es necesaria una consulta a la base de datos por cada ejercicio, reduciendo considerablemente el gasto de memoria y optimizando la posible actualización de la aplicación.
- **Diseño de la vista:** No dispone de vista.

4.4.24 Pregunta

- **Tipo de clase:** Clase estándar.
- **Layout asociado:** Ninguno.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** La clase Pregunta es el equivalente a la clase Ejercicio, pero en este caso para las preguntas del test. Almacena el enunciado, la respuesta correcta, las respuestas a la pregunta y la imagen del esquemático del circuito asociada a la pregunta en caso de haberla. Obtiene estos campos de la base de datos a través de la clase DBHelper, como en el caso de la clase Ejercicio y por consiguiente ofrece las mismas ventajas que ésta. Para añadir dificultad al test y conseguir forzar al usuario a demostrar su conocimiento acerca de la materia sobre la que se le pregunta, se mezcla el orden de las opciones de respuesta de cada pregunta al obtenerla de la base de datos, por lo que cada vez que se inicie el test, la respuesta correcta estará situada en distinta posición.
- **Diseño de la vista:** No dispone de vista.

4.4.25 Partida

- **Tipo de clase:** Clase estándar.
- **Layout asociado:** Ninguno.
- **Diálogos o fragmentos asociados:** Ninguno.
- **Descripción de la clase:** Esta clase sirve de marco para los contenidos del test. Alberga una matriz con las preguntas correspondientes y además gestiona la cuenta de las preguntas correctas o erróneas del test en proceso. Se ha diseñado con intención de poder ser reutilizable en aplicaciones futuras de la asignatura que también incluyan test.
- **Diseño de la vista:** No dispone de vista.

4.5 Estructura de la base de datos

La base de datos de la aplicación alberga algunos contenidos de los ejercicios y problemas de la aplicación, así como ciertos parámetros necesarios para el correcto funcionamiento de los métodos utilizados en las actividades y fragmentos que gestionan el test y los problemas. Se ha diseñado con una tabla para cada tipo, esto es, una tabla para los problemas y una tabla para las preguntas.

4.5.1 Tabla de preguntas

Id	Enunciado	Opcion1	Opcion2	Opcion3	Opcion4	Imagen

Tabla 4.1: Estructura de la tabla para las preguntas

La tabla de las preguntas del test consta de siete columnas para albergar los datos necesarios.

La primera columna, "id", es obligatoria para correcto funcionamiento de la base de datos, pero dado que las consultas realizadas en la misma no son especialmente complejas, no se utiliza en el código de la aplicación.

La segunda columna, "Enunciado", alberga datos de tipo string (cadenas de caracteres). En esta columna está el enunciado de las preguntas.

La tercera columna, "Opcion1", alberga la respuesta correcta a la pregunta de test. Se ha diseñado la clase Pregunta de manera que utilice este dato para almacenar la respuesta correcta y posteriormente mezcla esta columna junto con las tres siguientes, de manera que cada vez que se genere una pregunta, las opciones de respuesta aparezcan en distintas posiciones.

Por último la columna "Imagen" contiene el nombre del recurso de imagen asociado a la pregunta en cuestión, en caso de que lo haya, o la palabra "nada", como indicativo de que la pregunta no necesita una imagen.

4.5.2 Tabla de ejercicios

Id	Número de ejercicio	Número de variables	Enunciado	Nombre de la Imagen Asociada	Solución	Nombre Variables

Tabla 4.2: Estructura de la tabla para los ejercicios

Como en el caso del test, la tabla con los contenidos de los ejercicios consta de siete columnas.

La primera columna es obligatoria para el correcto funcionamiento de la base de datos, pero igual que en el caso del test, no se utiliza en la aplicación.

La segunda columna almacena el número del ejercicio en esa fila. Este es el campo que se utiliza en la creación de una instancia de la clase Ejercicio ya que es único para cada ejercicio.

La tercera columna alberga el número de variables que aparecen en el ejercicio en cuestión. Es utilizada en la actividad EjercicioTabla durante la creación de las vistas, para poder generar la cantidad necesaria de fragmentos VistaEjercicioTabla y destruir las columnas correspondientes en el caso de que el número de variables no sea múltiplo de seis, la anchura escogida para la parte de la tabla representada por cada fragmento VistaEjercicioTabla.

El cuarto campo de la base de datos contiene el enunciado del ejercicio y es usado por la actividad Enunciado para obtenerlo.

La quinta columna alberga el nombre de la imagen asociada al ejercicio en cuestión. Este dato de tipo string es utilizado también en la actividad Enunciado y en el dialogo Esquema, para obtener a través de una instancia de la clase de Android "Resources" la imagen .png almacenada en la carpeta drawables correspondiente.

En la sexta columna está la solución del ejercicio. Se trata de una cadena de caracteres con unos y ceros representando los estados booleanos true y false respectivamente, que es convertida en la clase Ejercicio en una matriz de valores booleanos. Esta matriz se utiliza en los métodos de comprobación y resolución de los ejercicios.

En la última columna se almacena el nombre de las variables, para poder así asignar los títulos de las columnas de manera correcta, ya que deben corresponder con los nombres utilizados en la imagen con el circuito a resolver.

5 Integración, pruebas y resultados

5.1 Pruebas realizadas

Durante el desarrollo de la aplicación y una vez finalizada la misma se han realizado pruebas constantes de su funcionamiento. Los principales parámetros que influyen en el correcto funcionamiento de la aplicación son los expuestos en la sección 3: el tamaño de pantalla, la densidad de pantalla y la versión de Android. Ya que resultaría increíblemente costoso disponer de un terminal con cada posible configuración de estos parámetros, se ha utilizado una herramienta muy útil que proporciona Android Studio.

5.1.1 AVD Manager

Esta herramienta, el Android Virtual Device Manager (AVD Manager), permite crear una máquina virtual en la que instalar un emulador de un dispositivo Android. Gracias al AVD Manager, resulta muy sencillo crear distintos perfiles de emulación, pudiendo almacenar distintos perfiles de configuración, para así poder comprobar el correcto funcionamiento de la aplicación en cualquier dispositivo.

5.1.1.1 Creación de un dispositivo virtual

Presionando el icono del AVD Manager en Android Studio, aparece la pantalla principal con un listado de los dispositivos creados, si los hubiese, y el botón para la creación de un nuevo dispositivo virtual.

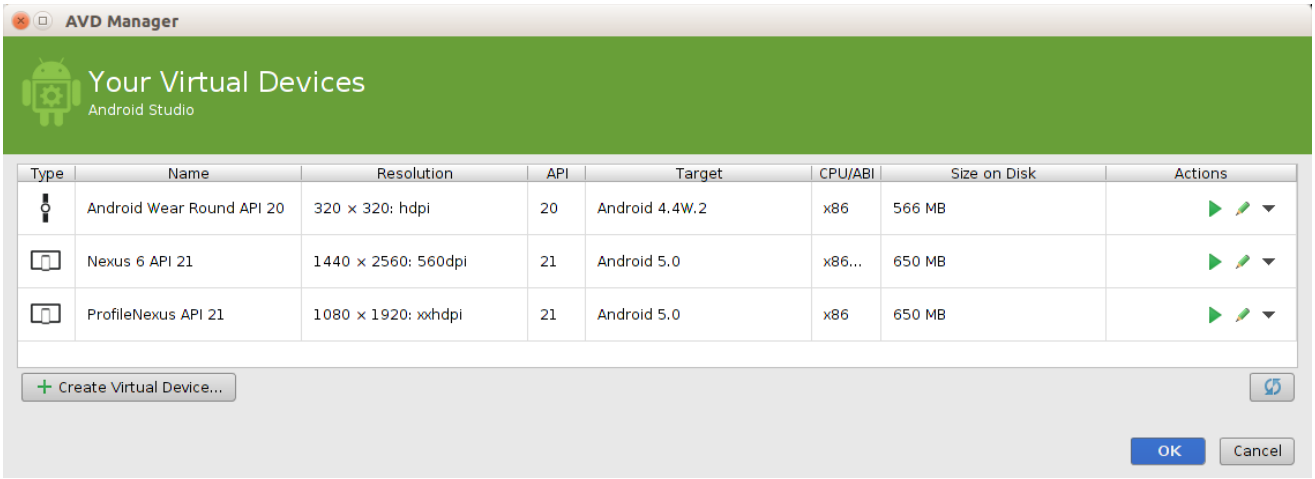


Figura 5-1: AVD Manager

Al presionar sobre “Create Virtual Device”, se abre la pantalla de configuración. En ella podemos escoger entre varias configuraciones preestablecidas, o crear nuestra propia configuración. Dado que las configuraciones ofrecidas abarcan la mayoría de combinaciones

de tamaño y densidad de pantalla existentes en dispositivos Android reales, se ha optado por escoger siempre entre ellas para la creación de nuestro dispositivo virtual.

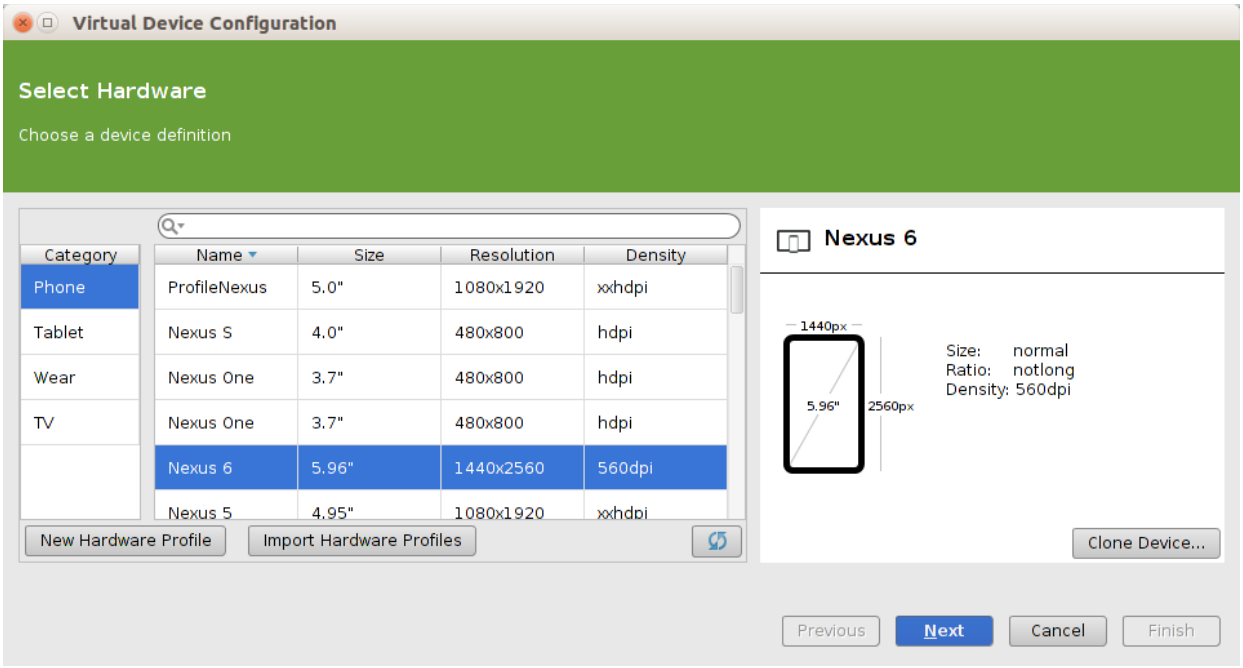


Figura 5-2: Configuración de pantalla

Una vez seleccionada la configuración deseada, clickamos en "Next" para acceder a la selección de la versión de Android que deseamos que tenga nuestro dispositivo. Ya que la aplicación debe soportar versiones de Android desde la 2.2 (API 8) hasta la 5.0.1 (API 21), se han creado dispositivos con las mismas características físicas, pero distinta versión de sistema operativo, para comprobar que no ocurren errores en las versiones más antiguas.

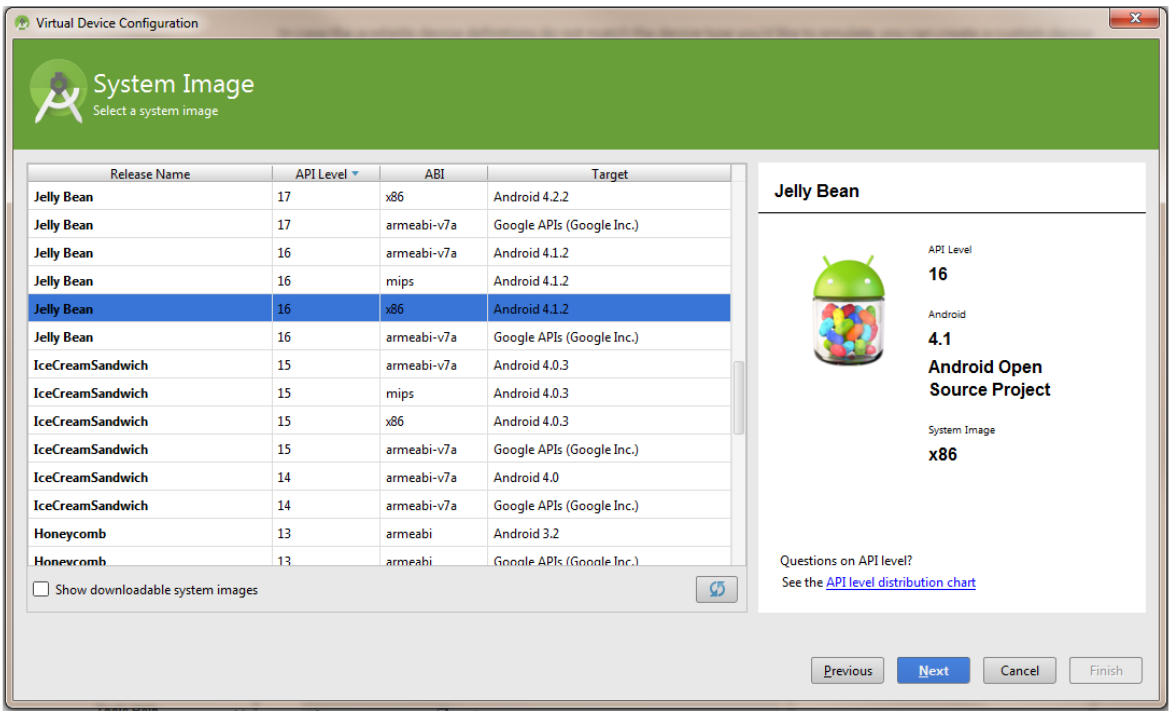


Figura 5-3: Selección de API

Por último, debemos darle un nombre a nuestro dispositivo y podemos seleccionar algunos parámetros más, como la existencia de cámara tanto trasera como delantera, la orientación de la pantalla al iniciarse el dispositivo o la escala con la que aparecerá en la pantalla. Estos parámetros no resultan de interés en nuestro caso, ya que la aplicación no hace uso de tales elementos por lo que la configuración preestablecida por Android resulta conveniente.

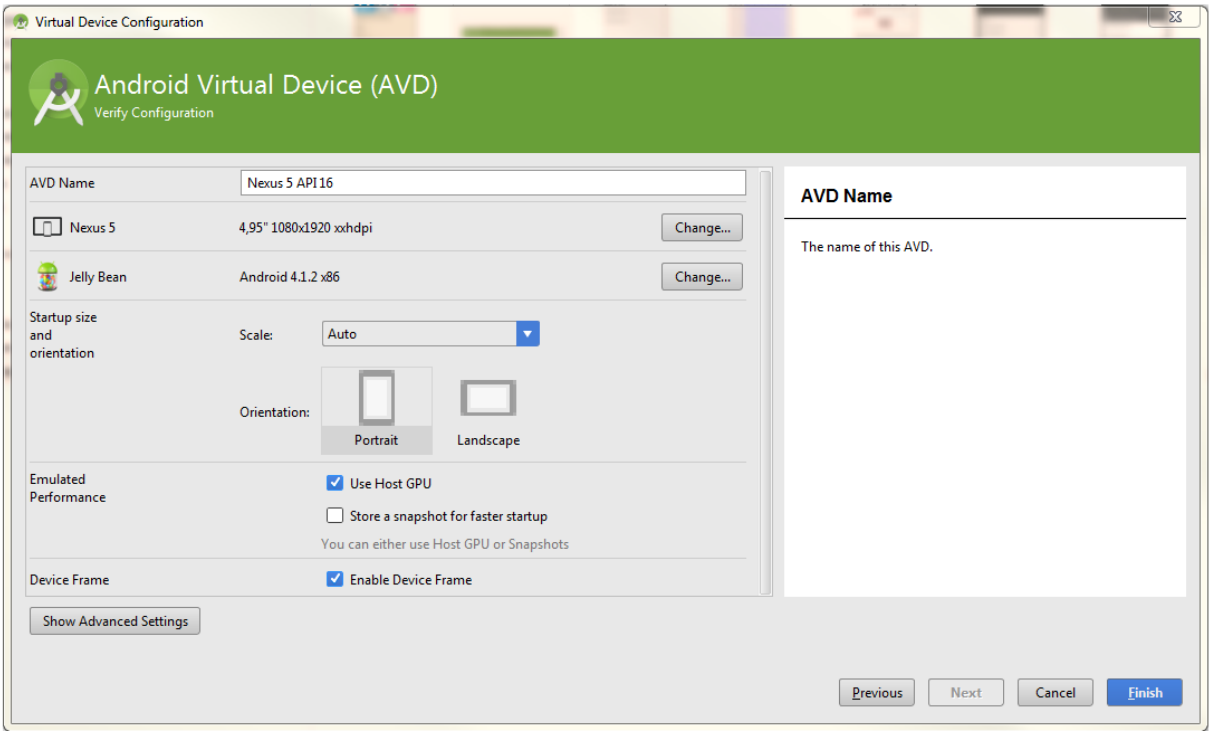


Figura 5-4: Configuración de dispositivo virtual

Una vez creado el dispositivo virtual, podemos acceder a él a través de la pantalla inicial de la figura 5.1, o, al ejecutar el comando "Run" desde Android Studio, se nos abrirá una ventana con los dispositivos iniciados y la opción de iniciar cualquiera de los dispositivos que tengamos guardados en memoria. En esta ventana también aparece cualquier dispositivo físico que esté conectado al pc en ese momento, y permite instalar la aplicación en el mismo.

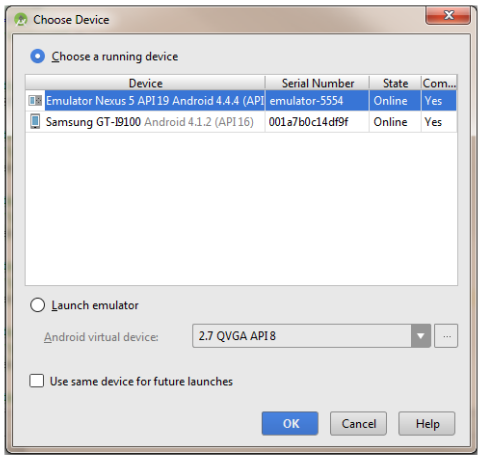


Figura 5-5: Selección de dispositivo

Tras seleccionar el dispositivo, virtual en este caso, en el que ejecutar la aplicación, Android Studio la instala y nos es posible comprobar su funcionamiento.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Tras la finalización del proyecto se han llegado a las siguientes conclusiones:

Los objetivos marcados inicialmente han sido conseguidos en su totalidad, mejorando en algún caso los métodos utilizados en las aplicaciones anteriores para alcanzarlos.

Se ha aprendido lenguaje de programación orientada a objetos, más específicamente Java, y lenguaje XML.

Se ha conseguido reducir considerablemente la cantidad de código de la aplicación utilizando clases y métodos autónomos, que permiten su reutilización en cualquiera de las aplicaciones de la asignatura futuras de una manera cómoda y clara.

Se ha estructurado la adquisición de datos de una manera más ordenada y con mayor potencia a la hora de actualizar los contenidos de la aplicación.

Se han creado estilos para los componentes de la aplicación que permiten su uso a través de referencias a los mismos, estandarizando su diseño para futuras aplicaciones.

6.2 Trabajo futuro

Como posibles mejoras a la aplicación o complementos a la misma se pueden incluir:

- Creación de layouts y adaptación de las clases para funcionamiento en orientación horizontal.
- Desarrollo de una aplicación que aúne las ya desarrolladas en una sola, englobando así todos los contenidos de la asignatura.
- Desarrollo de un método para lectura de los contenidos de la aplicación de una base de datos remota, alojada en servidor, para reducir el peso de la aplicación y ofrecer siempre contenidos actualizados.
- Optimización del código para la reducción de consumo de energía del dispositivo.
- Posibilidad de cambio de tamaño de letra para personas con problemas visuales.
- Traducción de la aplicación a varios idiomas, principalmente castellano, para poder acceder a mayor cantidad de usuarios.
- Creación de login con password, para acceder a funcionalidades específicas, sólo accesibles por alumnos de la asignatura. Un ejemplo de estas funcionalidades sería la realización de exámenes con cronómetro, y que los resultados fuesen enviados automáticamente al profesor de la asignatura una vez se acabase el tiempo o el alumno finalice la prueba.

Referencias

[1] M.J.S. Smith, “Application-Specific Integrated Circuits”, Addison-Wesley VLSI Design Series, Addison-Wesley VLSI Design Series, June 1997.

[2] http://www.signal.army.mil/OLD/ocos/museum/AMC/talk.asp

[3] https://www2.deloitte.com/content/dam/Deloitte/es/Documents/tecnologia-media-telecomunicaciones/Deloitte_ES_TMT_Consumo-movil-espana-2014-def.pdf

[4] http://stackoverflow.com

[5] https://developer.android.com

Glosario

API	Application Programming Interface
AVD	Android Virtual Device
DFT	Design For Test
ADT	Android Development Tools
DIE	Dispositivos Integrados Especializados
SDK	Software Development Kit
UAM	Universidad Autónoma de Madrid

Anexos

A PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 1.000 €
- Alquiler de impresora láser durante 6 meses .....50 €
- Smartphone .....300€
- Tablet .....400€
- Material de oficina .....150 €
- Total de ejecución material ..... 1.900 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 304 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 114 €

4) Honorarios Proyecto

- 1200 horas a 15 € / hora ..... 18000 €

5) Material fungible

- Gastos de impresión ..... 60 €
- Encuadernación..... 200 €

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 20318 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto..... 4266,78 €

8) Total presupuesto

- Total Presupuesto.....24584,78 €

Madrid, Septiembre de 2015

El Ingeniero Jefe de Proyecto

Fdo.: Juan Burgos Abadie
Ingeniero de Telecomunicación

B PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una APLICACIÓN DE PROBLEMAS RESUELTOS DE CIRCUITOS DIGITALES COMBINACIONALES BAJO ANDROID. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.