UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

Enciclopedia Android para el laboratorio de la asignatura CED de la EPS-UAM

Isabel Gómez Pérez
Tutor: Federico García Salzmann

Julio 2020

Enciclopedia Android para el laboratorio de la asignatura CED de la EPS-UAM

AUTOR: Isabel Gómez Pérez
TUTOR: Federico García Salzmann

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio 2020

Resumen

Este Trabajo de Fin de Grado tiene como objetivo el diseño y desarrollo de una aplicación móvil para el sistema operativo de Android. Esta aplicación está destinada a los alumnos que se encuentran en primer curso de Ingeniería de Tecnologías y Servicios de la Telecomunicación en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid (EPS-UAM). En concreto, se trata de una guía de estudio de las prácticas de la asignatura de Circuitos Electrónicos Digitales.

La aplicación consta de varias secciones. Por una parte, se introducen los conceptos teóricos que son necesarios conocer antes de comenzar a trabajar en el laboratorio. Además, se proporcionan los diagramas y esquemas de los componentes a utilizar durante las prácticas.

Para facilitar y agilizar el trabajo realizado durante las prácticas, la aplicación dispone de un conversor numérico (de decimal a binario y hexadecimal) y una calculadora de checksum. Además, cuenta con una herramienta para calcular la nota obtenida en las pruebas realizadas en clase.

A todo eso, se le suma un apartado en el que se le ofrece al alumno la posibilidad de enviar un correo al profesor de manera rápida y sencilla.

El uso de esta aplicación servirá de ayuda para el aprendizaje de los conocimientos expuestos en la asignatura y su puesta en práctica.

Este trabajo está disponible de manera gratuita para todo aquel que disponga de un dispositivo con sistema operativo Android. Se encuentra en la plataforma digital de descarga de aplicaciones de Google, Play Store.

Palabras clave

Circuitos, componentes, hexadecimal, binario, decimal, checksum, cables, memoria, puertas, formato, recomendaciones, Android.

Abstract

This End of Degree Project aims to design and develop a mobile application for the Android operating system. This application is intended for students in the first year of Engineering in Telecommunications Technologies and Services at the Escuela Politécnica Superior de la Universidad Autónoma de Madrid (EPS-UAM). Specifically, it is a study guide of the practices of the subject of Digital Electronic Circuits.

The application consists of several sections. On the one hand, the theoretical concepts that are necessary to know before starting to work in the laboratory are introduced. In addition, the diagrams and schemes of the components to be used during the practices are provided.

To facilitate and speed up the work done during the practices, the application has a numerical converter (from decimal to binary and hexadecimal) and a checksum calculator. It also has a tool to calculate the grade obtained in the tests carried out in class.

In addition, there is a section that offers the student the possibility of sending an email to the teacher quickly and easily.

The use of this application will help the student to learn the knowledge exposed in the subject and to put it into practice.

This work is available for free to anyone with a mobile device with Android operating system. It can be found in Google's digital application download platform, Play Store.

Keywords

Circuits, components, hexadecimal, binary, decimal, checksum, cables, memory, gates, format, recommendations, Android.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Actualmente, es prácticamente imposible el hecho de que una persona no disponga de un dispositivo móvil. Además, la mayoría de esos dispositivos móviles son ahora *smartphones*, teléfonos móviles inteligentes. Cada vez es más común el uso de estos dispositivos en nuestra vida cotidiana, ya que facilitan muchas labores y almacenan una gran cantidad de información personal de una forma muy práctica y cómoda.

Esto se puede aplicar de manera similar a los estudios, la idea de estudiar o conseguir el material de estudio a través de un teléfono móvil está en aumento desde hace varios años. El motivo es el mismo, resulta más cómodo conseguir toda la información necesaria para estudiar dentro de un dispositivo móvil, el cual llevas siempre contigo, que tener que cargar con diferentes libros y/o apuntes necesarios para el estudio de una asignatura.

Teniendo en cuenta la información anterior, aparece la motivación de realizar este trabajo de fin de grado. Surge la idea de poder facilitar el estudio y la comprensión de las prácticas de una asignatura de primero de Ingeniería de Tecnologías y Servicios de la Telecomunicación a través de algo tan simple como lo es una aplicación móvil para dispositivos Android.

Con este proyecto se pretende acercar a los alumnos todos los aspectos teóricos y prácticos necesarios para poder entender la asignatura de Circuitos Electrónicos Digitales de una manera cómoda y sencilla; pudiéndose utilizar dicha aplicación durante el desarrollo de las prácticas y disponer de las herramientas que ésta dispone.

Las ventajas que se obtienen al realizar este trabajo son las siguientes:

- Comodidad de uso.
- Eliminación del formato físico.
- Reducción del uso de papel.
- Posibilidad de uso en cualquier sitio.
- Posibilidad de uso sin necesidad de internet.
- Corrección automática de errores.
- Comunicación rápida con el profesor.

1.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación móvil destinada a los alumnos de Circuitos Electrónicos Digitales del primer curso de Ingeniería de Tecnologías y Servicios de la Telecomunicación de la Escuela Politécnica Superior de la UAM.

En un principio, se propone consultar y entender los conceptos importantes que aparecen en la primera sección de la aplicación. Dichos conceptos son necesarios para la comprensión y desarrollo de esta asignatura. Seguidamente, se encuentran los componentes a utilizar durante las prácticas, por lo que se propone localizarlos para su posterior uso. Una vez se tiene clara la teoría anterior y se está preparado para desarrollar las prácticas, se puede hacer uso de las siguientes herramientas que incluye, como un conversor de números o una calculadora de checksum.

La información que aparece en la aplicación tiene por objetivo fijar los conceptos básicos sobre los siguientes temas:

- Protoboard.
- Aspecto y uso del entrenador del laboratorio.
- Montaje de circuitos con encapsulados DIP.
- Formato Intel Hex.
- Debugging.
- Material del laboratorio.
- Componentes y sus fichas de datos.

En cuanto a nivel personal y educativo, este trabajo de fin de grado tiene como objetivo el aprendizaje de un nuevo lenguaje de programación como es Java, en un sistema operativo distinto a los aprendidos anteriormente, Android. En ninguna de las asignaturas de esta carrera se ha desarrollado el lenguaje Java en este entorno, por lo que supone un aumento de conocimientos muy útiles para un futuro.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- Capítulo 1: Introducción.
- Capítulo 2: Estado del arte.
- Capítulo 3: Diseño del proyecto.
- Capítulo 4: Desarrollo del proyecto.
- Capítulo 5: Integración, pruebas y resultado.
- Capítulo 6: Conclusiones y trabajo futuro.

2 Estado del arte

Antes de ponerse a diseñar y desarrollar una aplicación, puede ser interesante ver qué aplicaciones similares hay en el mercado. Esto puede resultar útil por dos motivos, tanto para no copiar algo que ya esté hecho de la misma forma, como para coger ideas y utilizarlo de inspiración a la hora de crear la tuya propia.

Por eso, en este apartado se va a realizar un pequeño estudio de las aplicaciones que hay disponibles en el Play Store que traten aspectos sobre Circuitos Electrónicos Digitales.

2.1 Electrónica Digital

Se trata de una aplicación gratuita disponible en Google Play Store. Ofrece un amplio repertorio de información teórica acerca de Electrónica Digital. Todo ese contenido está dividido en diferentes apartados según el tema a tratar. Cuenta con muchos y muy variados esquemas y gráficos explicativos que pueden facilitar la comprensión de los conceptos.

Teóricamente es muy completa, si lo que se busca es únicamente información teórica, puede ser una buena opción.

Sin embargo, esta aplicación no dispone de varias herramientas que pueden resultar útiles a la hora de desarrollar la práctica de esta asignatura, como pueden ser un conversor numérico, una calculadora de checksum o las hojas de datos de los componentes más utilizados.

Tampoco facilita la opción de contactar a través de correo electrónico con el profesor o con algún compañero para compartir o consultar dudas y resultados.

2.2 Arithmetic Circuits

Esta aplicación se encuentra de manera gratuita en Google Play Store. Se trata de una aplicación educativa acerca de circuitos aritméticos.

Cuenta con un apartado a modo de tutorial en el que ofrece teoría sobre el tema a tratar, para, a continuación, proponer un apartado con ejercicios relacionados. Esto resulta muy útil ya que, no solo aprendes la teoría, sino que la pones en práctica con ejercicios propuestos y sirve para fijar los conceptos aprendidos.

Una vez realizados los ejercicios, te indica si se han realizado de manera correcta o no y te da algún consejo y recomendación sobre cómo se debe resolver. Además, tiene la opción de poder enviar esos resultados por correo.

Esta aplicación también carece de las herramientas necesarias para el desarrollo del laboratorio como son el conversor y la calculadora checksum.

2.3 Checksum Calculator.

Esta aplicación se encuentra de manera gratuita en Google Play Store.

Realiza múltiples cálculos de checksum a la vez. Además de realizar el cálculo del checksum, da varias opciones una vez calculado; puedes guardarlo, borrarlo, puedes compararlo con otros resultados anteriores y también da la opción de compartir ese resultado con quien se quiera a través de un correo electrónico.

Muy buena herramienta a la hora de calcular checksum, pero carece de elementos como teoría o un conversor numérico. Se trata de una aplicación muy concreta en desarrollar una única misión, no amplía conocimientos más allá del resultado obtenido.

2.4 Conversor Numérico.

Se trata de una aplicación de pago disponible en Google Play Store. Su precio es de 0.59€.

Esta aplicación realiza varios tipos de conversiones de números. La ventaja que presenta frente al conversor numérico que tiene nuestra aplicación es que te da la opción de trabajar en la base que desees. Tiene 4 bases numéricas disponibles: decimal, binario, hexadecimal y octal. A la hora de realizar una conversión te pide introducir la base del número a convertir y la base del número una vez convertido.

Resulta una aplicación muy útil y rápida, que puede ayudar mucho al desarrollo de ejercicios de electrónica digital de forma muy sencilla. Sin embargo, su precio no está justificado, ya que hay muchas opciones en el mercado que realizan la misma función de manera gratuita.

2.5 Calculadoras de Ingeniería Electrónica.

Esta aplicación se encuentra de manera gratuita en Google Play Store. Contiene una gran cantidad de información y de herramientas útiles sobre Electrónica.

Lo más llamativo de esta aplicación es la enorme cantidad de calculadoras que dispone. Ofrece más de 50 tipos de calculadoras: Ley de Ohm, resistencias, inductores, condensadores, impedancia RLC, reguladores de voltaje, amplificadores operacionales, conversores, diodos, mapas de Karnaugh, etc.

Además de ofrecer esa cantidad de calculadoras, también presenta una sección de teoría donde explica los conceptos importantes acerca de Electrónica. A esta sección de teoría se le añade también la opción de buscar componentes para acceder a sus esquemas internos y a sus tablas de valores.

A pesar de ser una aplicación muy completa, carece de un apartado donde poder hacer ejercicios. Además, dentro de su amplio repertorio de calculadoras, no contiene la del checksum.

Una vez analizado el conjunto de aplicaciones similares que ofrece el mercado actualmente, podemos concluir diciendo que resulta muy complicado encontrar una aplicación que contenga todas las herramientas que ofrece la aplicación desarrollada.

Encontramos cantidad de opciones con teoría y cantidad con ejercicios, muchas calculadoras y muchos conversores, pero ninguna que sea capaz de concentrar todas esas herramientas en una sola aplicación. Por eso resulta muy interesante el disponer de esta aplicación si vas a cursar o decides estudiar una asignatura de Circuitos Electrónicos Digitales.

A continuación, se adjunta una tabla que resume las características buscadas anteriormente de las aplicaciones que se han analizado, junto con la aplicación desarrollada para este trabajo de fin de grado.

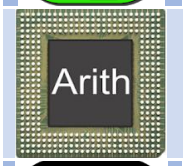
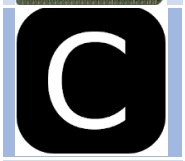
Aplicación	Icono	Gratuita	Teoría	Ejercicios	Conversor	Checksum
Electrónica Digital		Sí	Sí	No	No	No
Arithmetic Circuits		Sí	Sí	Sí	No	No
Checksum Calculator		Sí	No	No	No	Sí
Conversor Numérico		No	No	No	Sí	No
Calculadoras de Ingeniería Electrónica		Sí	Sí	No	Sí	No
CED LAB		Sí	Sí	No	Sí	Sí

Tabla 1: Comparativa aplicaciones similares.

3 Diseño

3.1 Objetivos de diseño

El objetivo de este proyecto es el diseño de una aplicación para dispositivos móviles con sistema operativo Android con la que se pueda aprender de forma rápida y sencilla conocimientos acerca de la asignatura Circuitos Electrónicos Digitales.

En un principio, la aplicación está diseñada para complementar el desarrollo de la asignatura Circuitos Electrónicos Digitales de primero de Ingeniería de Tecnologías y Servicios de la Telecomunicación. Sin embargo, puede ser utilizada para cualquier persona interesada en el aprendizaje de electrónica digital, ya que presenta la teoría y las herramientas necesarias para comenzar el aprendizaje sobre dicho tema.

La aplicación contará con una parte de teoría para conocer los conceptos importantes de la asignatura y una guía de los componentes más utilizados junto con sus esquemas internos. Una vez familiarizado con los conceptos básicos, la aplicación cuenta con las herramientas necesarias para desarrollar ejercicios y prácticas, con el objetivo de asentar la teoría estudiada.

En caso de tener alguna duda durante el proceso de aprendizaje, la aplicación facilita la opción de enviar un correo electrónico al profesor con las preguntas que surjan.

3.2 Requisitos de diseño

3.2.1 Sistema Operativo

El primer punto para estudiar antes de comenzar a diseñar una aplicación es la elección del sistema operativo sobre el que se va a desarrollar. Por ello, se va a realizar un breve estudio sobre los sistemas operativos disponibles más utilizados:

- **Android**
  Se trata de un sistema operativo desarrollado por Google basado en Linux y otros *software* de código abierto. Principalmente, fue diseñado para teléfonos y relojes inteligentes con pantalla táctil, tabletas, automóviles y televisores. Actualmente, es el sistema operativo más usado del mundo con, aproximadamente, el 67% de los dispositivos.

- **IOS**
  Se trata de un sistema operativo desarrollado por la empresa multinacional Apple Inc. Principalmente, fue diseñado para el dispositivo móvil *iPhone,* aunque con el tiempo ha sido utilizado en las tabletas *iPad* y en los relojes inteligentes u otros dispositivos como los *iPod.* Se sitúa como el segundo sistema operativo más usado del mundo con, aproximadamente, el 30% de los dispositivos.

- **Windows *Phone***
  Se trata de un sistema operativo desarrollado por la empresa Microsoft, dirigido para dispositivos móviles. Aproximadamente, el 2% de los dispositivos mundiales están bajo este sistema operativo, colocándolo como el tercer sistema operativo más usado del mundo.

Los nombrados anteriormente son los 3 sistemas operativos más usados mundialmente, pero, aunque en menor cantidad, existen otros como pueden ser *Java ME, Symbian, BlackBerry o Samsung.* A continuación, se muestra un gráfico comparativo con los sistemas operativos mundiales en el año 2018:

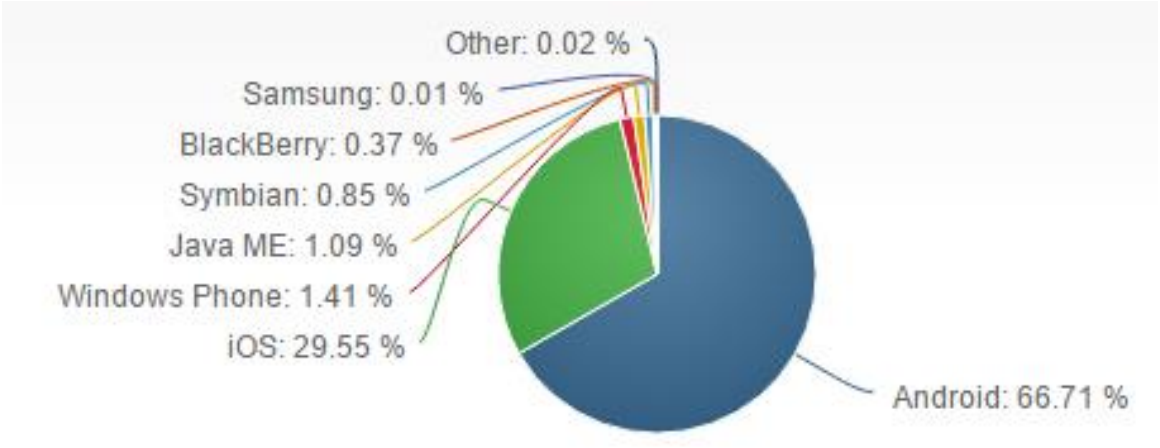


Figura 1: Comparativa sistemas operativos.

Observando los datos obtenidos, se comprueba que Android es el sistema operativo que llega a más población. Aunque, como ya se ha dicho anteriormente, esta aplicación vaya dedicada a una asignatura específica de una universidad concreta, también se busca poder llegar con este trabajo al máximo número de gente posible. Por tanto, esta es uno de los motivos por los que se ha elegido Android como sistema operativo para desarrollar este proyecto.

Otro aspecto a tener en cuenta era las herramientas necesarias para realizar una aplicación móvil. En el caso de Android, se desarrolla en el programa multiplataforma *Android Studio*, el cual se puede encontrar de manera gratuita en la página oficial de Android. En este programa se pueden hacer todo tipo de pruebas en cualquier dispositivo sin necesidad de tener licencia de desarrollador. Sin embargo, en el caso de IOS necesitas acceder a un entorno más específico en el que necesitas obtener una licencia de pago para poder realizar pruebas.

En relación con lo mencionado en el apartado anterior, otra ventaja que presenta Android frente a IOS es que, al ser más utilizado, hay más programadores y, por tanto, resulta más fácil encontrar información y resolver dudas que en el caso de IOS. Durante el proceso de programación de la aplicación aparecen muchos errores difíciles de corregir y resulta muy útil encontrar a gente y páginas con información que pueden ayudarte a solucionarlos.

Una vez finalizada la aplicación, si ha sido realizada con el programa *Android Studio*, al ser multiplataforma, el resultado puede ser utilizado por distintos dispositivos de diferentes marcas, siempre y cuando soporten el sistema operativo de Android. Sin embargo, con IOS, tu aplicación solo podrá ser utilizada en dispositivos exclusivos de Apple, lo que te restringe un gran número de dispositivos móviles.

Teniendo en cuenta las ventajas presentadas anteriormente, se ha decidido realizar este proyecto con el sistema operativo de Android. Se tiene en cuenta también su constante aumento desde hace varios años, como se aprecia en el siguiente gráfico:

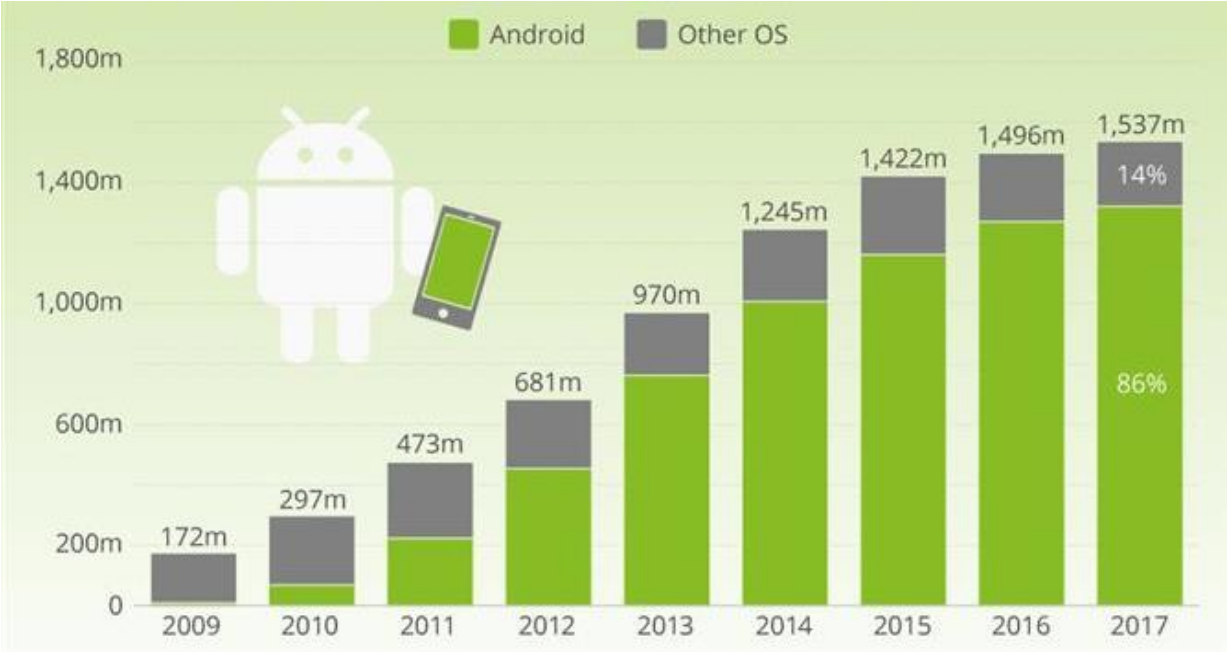


Figura 2: Evolución temporal Android.

3.2.2 Versiones compatibles

Una vez se ha decidido que el sistema operativo es Android, se tiene que decidir qué versión de Android se va a utilizar para la aplicación.

Con el tiempo, se van cambiando las versiones de Android implantando las nuevas actualizaciones disponibles. Al comenzar un nuevo proyecto en el *Android Studio*, se pide decidir la versión deseada, mostrando la siguiente pantalla:

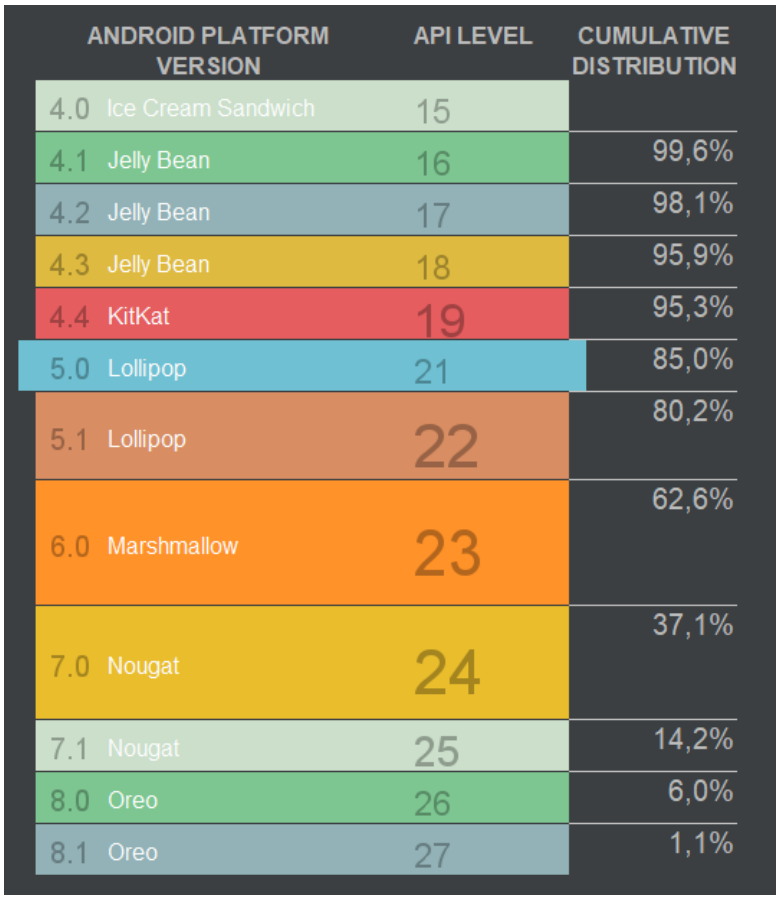


Figura 3: Versiones Android.

En la imagen aparece, a la izquierda, el nombre y el número de la versión, en la columna del medio aparece el nivel que ocupa dentro del programa *Android Studio* y en la última columna aparece un porcentaje. Ese porcentaje representa la cantidad de dispositivos que, actualmente, soportan esa versión; esto quiere decir que cuanto más antigua sea la versión, más dispositivos la soportan. Sin embargo, no conviene escoger una versión muy antigua ya que se puede quedar anticuada. A medida que van saliendo nuevos dispositivos móviles, estos llevan implantada una de las últimas versiones disponibles.

En el caso de esta aplicación, se ha decidido utilizar la versión 5.0 *Lollipop*. Con esta versión se consigue que la aplicación esté disponible para el 85% de dispositivos, que es un número muy elevado, pero sin quedarse anticuado, ya que esta versión ofrece una gran cantidad de actualizaciones.

3.2.3 Interfaz

La interfaz de la aplicación es una de las partes del diseño más importantes, ya que es lo que les llega directamente a los usuarios.

Esta aplicación se ha diseñado dentro de un grupo de trabajo de la Escuela Politécnica Superior de la UAM, DSLab (*Digital System Laboratory*). Al estar dentro de este grupo de trabajo, se tiene el requisito de diseñar todas las interfaces de las aplicaciones de una manera similar.

De esta forma, la interfaz que se ha elegido es una muy básica y lo más intuitiva posible. No dispone de demasiados colores, fondos ni imágenes. Los botones que aparecen son de color gris, sobre un fondo liso blanco, de forma similar a las demás aplicaciones del grupo DSLab, como aparece en la siguiente imagen:

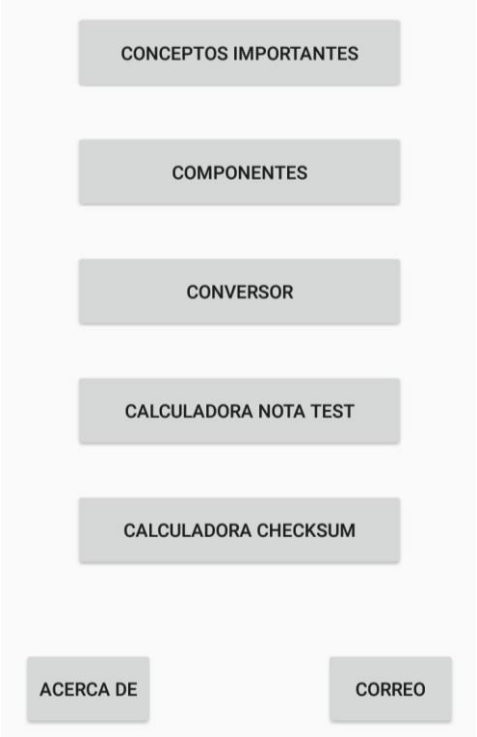


Figura 4: Interfaz menú principal.

Una parte importante de la interfaz de la aplicación es el tipo y el tamaño de la letra utilizada. En cuanto al tipo de letra la elección era sencilla, se utiliza la misma fuente que en las aplicaciones anteriores del grupo DSLab. En cuanto al tamaño de la letra, la cosa cambia.

En pantallas principales, donde aparecen los botones con las opciones que se ofrecen se ha decidido poner la letra mayúscula y con un tamaño lo suficientemente grande como para que se pueda leer sin hacer ningún esfuerzo. En cambio, dentro de la sección de teoría aparece mucho texto y ahí se ha decidido poner un tamaño de letra mayor. La sección de teoría está diseñada con el texto dispuesto en diapositivas que se deslizan de manera horizontal. A pesar de que está diseñado de una forma muy fácil, se facilita aún más la labor de leer la teoría poniendo un tamaño de letra mayor.

Con todos los aspectos mencionados sobre la interfaz se consigue obtener una aplicación con una forma de uso sencilla e intuitiva, para facilitar la labor del desarrollo de la asignatura.

3.2.4 Idioma

Un aspecto importante a tratar para el diseño de la aplicación es el idioma que se va a utilizar.

Aun sabiendo que el idioma más utilizado en dispositivos tecnológicos es el inglés, se ha decidido desarrollar la aplicación en castellano. Esto es así porque se trata de una aplicación para un uso principal muy exclusivo, como es el desarrollo de una asignatura de la Escuela Politécnica Superior de la UAM. Debido a que esta asignatura se imparte en castellano, la aplicación se desarrolla en el mismo idioma.

3.3 Diagrama de bloques

Como se ha mencionado anteriormente, se pretende realizar una aplicación sencilla y lo más intuitiva posible. Por eso, se ha diseñado la distribución de la aplicación en forma de árbol, exponiendo los temas principales en la pantalla de inicio y desglosándolo en temas más específicos a medida que se va entrando en cada uno de ellos.

A continuación, se muestra el diagrama de bloques principal de la aplicación desarrollada:

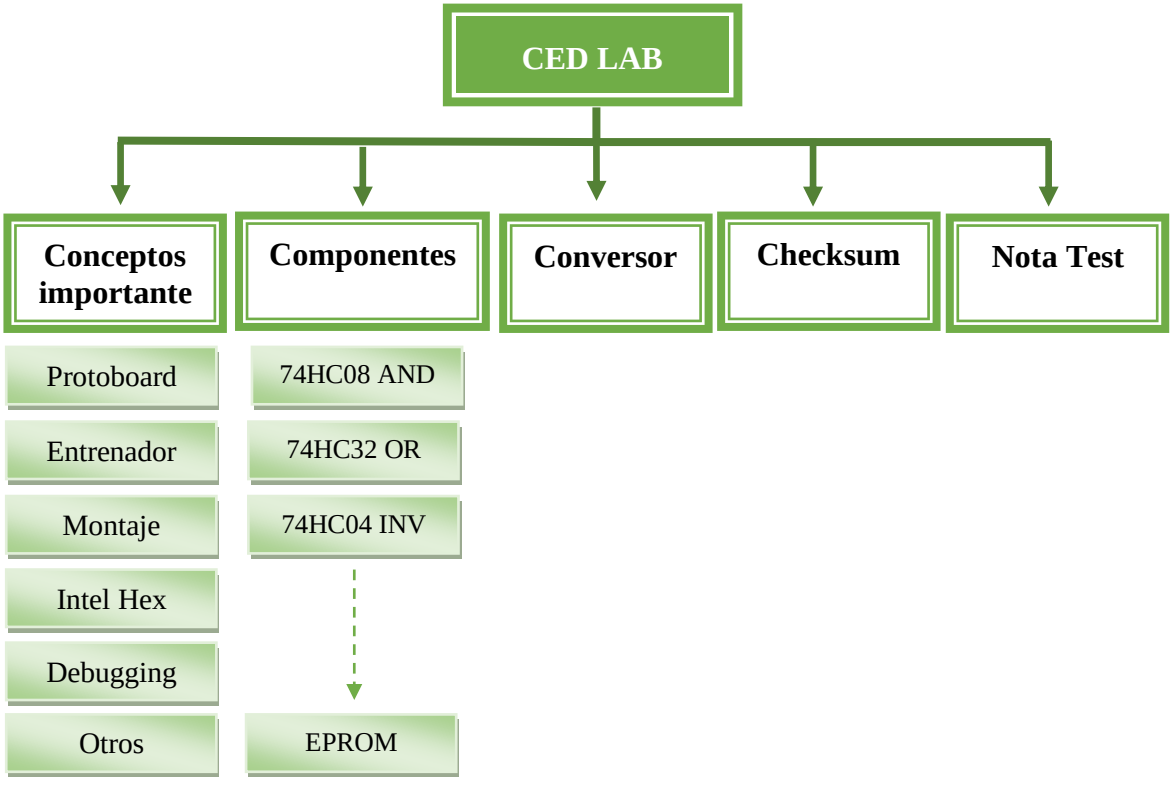


Tabla 2: Diagrama de bloques.

Además de los apartados mostrados en el diagrama de bloques, la aplicación cuenta con dos apartados adicionales de información y comunicación que se explican más adelante.

3.4 Estructura de la aplicación.

Lo primero que aparece al acceder a la aplicación es una pantalla informativa que explica la intención que se tiene con ese trabajo. La pantalla informativa es la siguiente:

Figura 5: Pantalla informativa.

Pasado un breve periodo de tiempo, esa pantalla desaparece y se muestra el menú principal como el que aparece en la Figura 4, que contiene los siguientes apartados:

- Conceptos importantes.
- Componentes.
- Conversor numérico.
- Calculadora checksum.
- Calculadora nota test.
- Acerca de.
- Correo.

A continuación, se van a analizar de manera resumida cada una de las secciones que dispone la aplicación y cada una de las pantallas que aparecen en ellas.

3.4.1 Conceptos importantes

Conceptos importantes es la primera sección que aparece en el menú principal. En ella se concentra la teoría básica y necesaria para seguir el desarrollo de la asignatura. Se resume la teoría y se proponen ejemplos explicativos para comprender cada uno de los temas propuestos.

Si se accede a dicha sección aparece la siguiente pantalla:

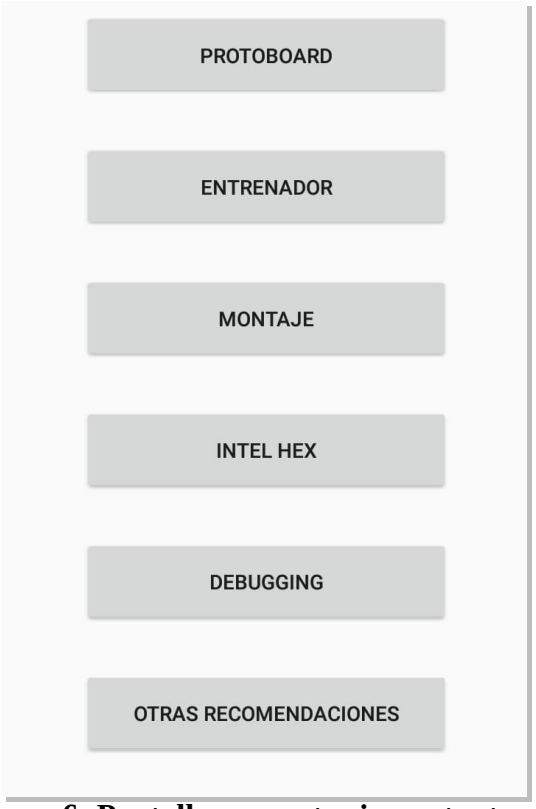


Figura 6: Pantalla conceptos importantes.

Si accedemos al botón de conceptos importantes, aparece otro menú con los siguientes botones:

- **Protoboard**
  En esta sección se explican los conceptos básicos de la Protoboard. Se comienza con una definición, para seguir con el aspecto que tiene. Después, aparece la información de cómo se distribuyen las conexiones dentro de la placa. Esto resulta muy útil a la hora de realizar ejercicios prácticos en el laboratorio. Por último, se hacen varias recomendaciones para ayudar a solucionar posibles situaciones que puedan ocurrir en el laboratorio con la Protoboard.

- **Entrenador**
  Si accedemos al botón de entrenador, encontramos la definición y explicación sobre qué es y el aspecto con el que lo vamos a encontrar en los laboratorios. Debido a que es un elemento que se va a utilizar repetidas veces, seguido a su definición se encuentran las pautas a seguir, paso por paso, para utilizarlo de manera correcta. También se explica detalladamente los interruptores, LEDs y displays del entrenador.

- **Montaje**
  En el apartado de montaje se habla sobre los encapsulados DIP. Se realizan varias recomendaciones sobre el material que es necesario llevar preparado a los montajes del laboratorio. Se habla sobre la nomenclatura utilizada en este tipo de montajes y se finaliza con la secuencia a seguir para trabajar en el laboratorio.

- **Intel Hex**
  En el botón de Intel Hex se encuentra la teoría necesaria sobre el formato numérico Intel Hex. A medida que se avanza con las diapositivas, se va explicando detalladamente cada campo del formato, a lo que se le añade un ejemplo de cada uno para aclarar la información explicada. Una vez explicados los campos del formato, se explica el concepto de checksum y se complementa con un ejercicio en el que se realiza el cálculo numérico del checksum de una memoria.

- **Debugging**
  En esta sección encontramos información sobre el debuguing, sobre lo que es y sobre cómo se realiza en circuitos sencillos, similares a los que se utilizan en las prácticas de la asignatura.

- **Otras recomendaciones**
  En el último apartado, se hacen recomendaciones generales sobre el uso de los alicates y cables en el laboratorio, sobre cómo poder realizar montajes similares a los de las prácticas en casa y sobre el programa necesario.

3.4.2 Componentes

La segunda sección que encontramos en el menú principal es la de componentes. Debido a que en la teoría de la asignatura se explican los esquemas internos de los componentes y que en las sesiones de prácticas es necesario conocer y saber interpretar dichos esquemas, se ha decidido introducir este apartado en la aplicación. Si accedemos a ella encontramos otro menú como el siguiente:

Figura 7: Pantalla componentes.

En la imagen aparecen varios componentes a los que se puede acceder para consultar sus pines y esquemas internos. La lista total de componentes que incorpora la aplicación es la siguiente:

- 74HC08 AND.
- 74HC32 OR.
- 74HC86 XOR.
- 74HC04 INV.
- 74HC00 NAND.
- 74HC02 NOR.
- 74HC74.
- 74HC151.
- 74HC163.
- 74HC175.
- 74HC253.
- EPROM SST39SF040.
- EPROM AM29F010B.

De todos los componentes aparecen sus pines, número y posición. Además, de los componentes necesarios aparece también el esquema interno y sus conexiones. De alguno, también se ha añadido el diagrama de tiempos.

A continuación, se muestra un ejemplo de cómo aparecen los esquemas en esta sección:

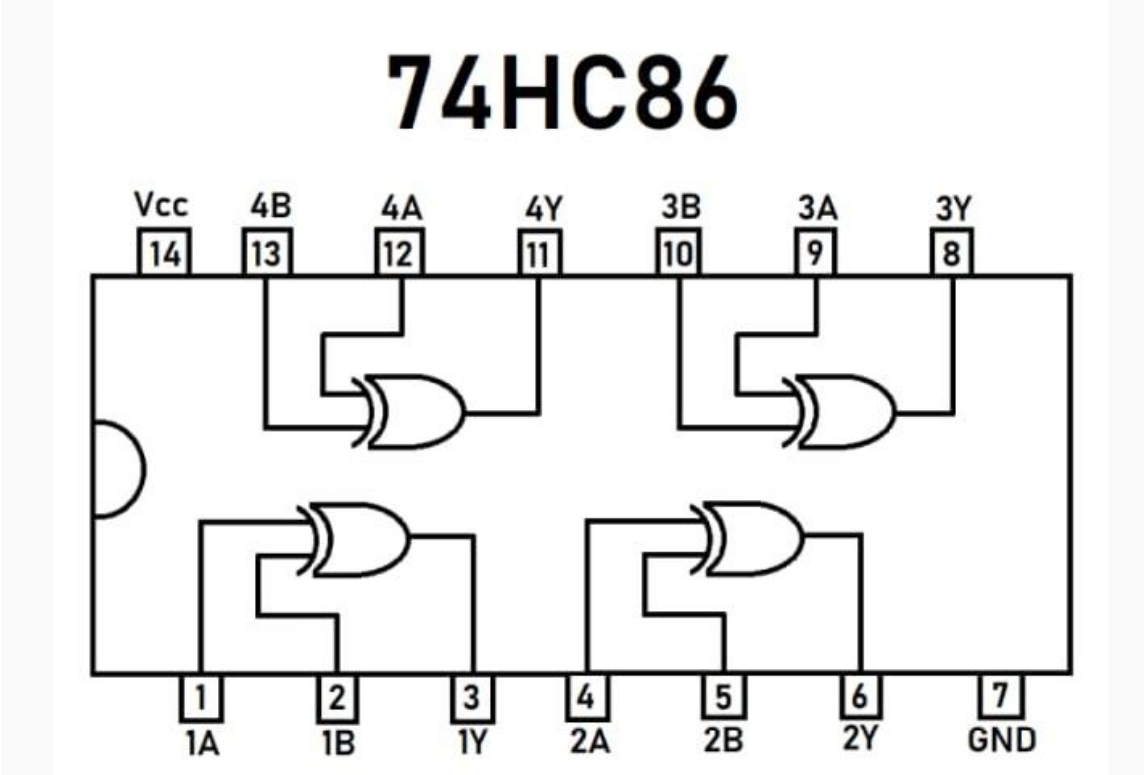


Figura 8: Ejemplo esquema componente.

3.4.3 Conversor

En el tercer botón del menú principal se encuentra un conversor numérico. Durante el transcurso de la asignatura, en muchos de los ejercicios propuestos es necesario convertir un número decimal a binario o a hexadecimal. Esto es lo que se facilita con este conversor numérico.

El funcionamiento es muy sencillo, consiste en introducir el número decimal que deseas convertir y al pulsar el botón “convertir”, te muestra el resultado de la conversión a binario y a hexadecimal.

A la hora de diseñar el conversor se plantea la idea de mostrar también el resultado en octal, pero al ver la poca utilidad que se e iba a dar a este resultado, se decidió no mostrarlo y así simplificar la visión de la solución por pantalla.

A continuación, se adjunta una imagen donde aparece la interfaz que presenta este conversor:

Figura 9: Pantalla conversor.

3.4.4 Calculadora checksum

Una calculadora de checksum es el tercer apartado que ofrece la aplicación en el menú principal. Estudiado el temario de la asignatura y los ejercicios propuestos, resulta muy útil tener una herramienta que te calcule de manera automática el checksum, ya que hacer los cálculos a mano de esta operación resulta muy tedioso.

Para obtener el checksum de la secuencia de números deseada, simplemente hay que meter dicha secuencia en la calculadora y, al darle a calcular, te muestra por pantalla el resultado obtenido, como aparece en la siguiente imagen:

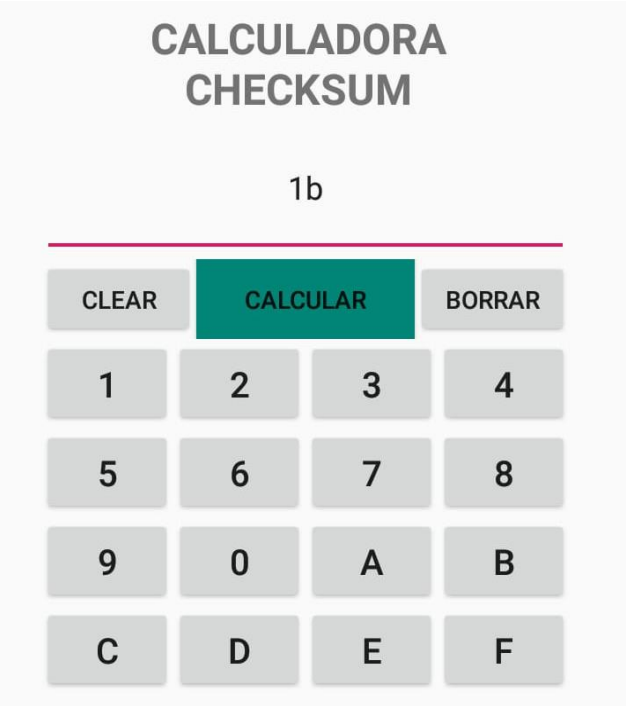


Figura 10: Pantalla calculadora checksum.

El número “1b” que aparece en la imagen es el checksum calculado de la secuencia de números introducida. Esa secuencia de números y letras son el resultado de realizar una serie de cálculos explicados en el apartado de conceptos importantes – Intel Hex, que terminan convirtiéndose en un número hexadecimal apto para introducir en dicha calculadora.

A la hora de calcular dicho número hexadecimal, el resultado debe ser un número par, de no ser así, resulta imposible poder calcular el checksum. Para poder prevenir este error, se ha diseñado un texto informativo momentáneo que aparece por pantalla en caso de que se introduzca un número impar. El mensaje informativo es el siguiente:

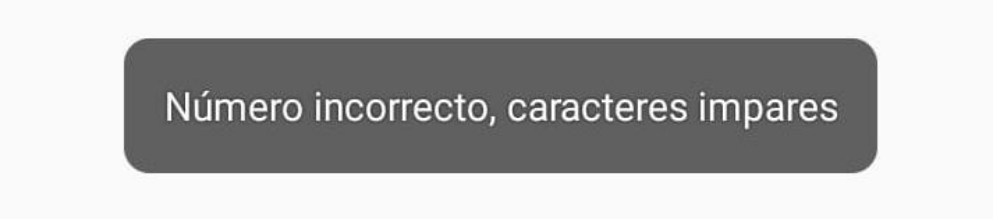


Figura 11: Error números impares.

Además de introducir un número impar de caracteres, pueden surgir otros errores como que le des a "Calcular" sin haber introducido ningún número hexadecimal. En este caso, se vuelve a informar del error con el siguiente mensaje:

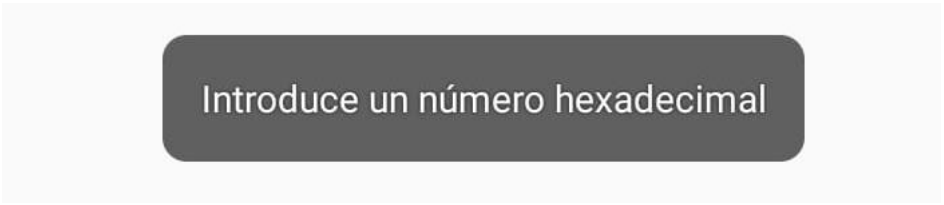


Figura 12: Error introducir número.

Como ya se ha explicado, para calcular el checksum es necesario introducir un número hexadecimal. Como los números hexadecimales están compuestos de números del 0 al 9 y letras de la A a la F, ha sido necesario diseñar un teclado específico.

El teclado diseñado para esta calculadora consta de los números y las letras necesarias para números hexadecimales además de un botón "borrar", para eliminar únicamente un carácter en caso de equivocarse al introducir la secuencia, un botón "clear" que elimina por completo el número que hayas introducido y el botón "Calcular" para que aparezca el resultado por pantalla. Este teclado aparece en la figura 10.

3.4.5 Calculadora nota test

Calculadora nota test es la última opción principal que ofrece el menú inicial. Durante el desarrollo de la asignatura se realizan diversos test para ir evaluando las capacidades que va aprendiendo el alumno. Como el cálculo de la nota de esos test no se hace de una manera convencional, esta aplicación ofrece una calculadora para obtener el resultado de tu prueba de una manera rápida y sencilla, ya que, únicamente se debe introducir los datos necesarios.

Al acceder al botón "Calculadora nota test" aparece una pantalla en la que se te piden los siguientes datos:

- **Nº preguntas:** en este campo se debe introducir el número total de preguntas de la prueba.

- **Nº respuestas:** en este campo se debe introducir el número de respuestas a elegir que hay en cada pregunta.

- **Nº aciertos:** en este campo se debe introducir el número de respuestas correctas que se ha conseguido una vez terminado el test.

- **Nº errores:** en este campo se debe introducir el número de respuestas incorrectas que se ha conseguido una vez terminado el test.

Una vez introducidos los datos anteriores y pulsando el botón "Calcular" aparece por pantalla la nota final que se ha obtenido en la prueba, como se muestra en el siguiente ejemplo:

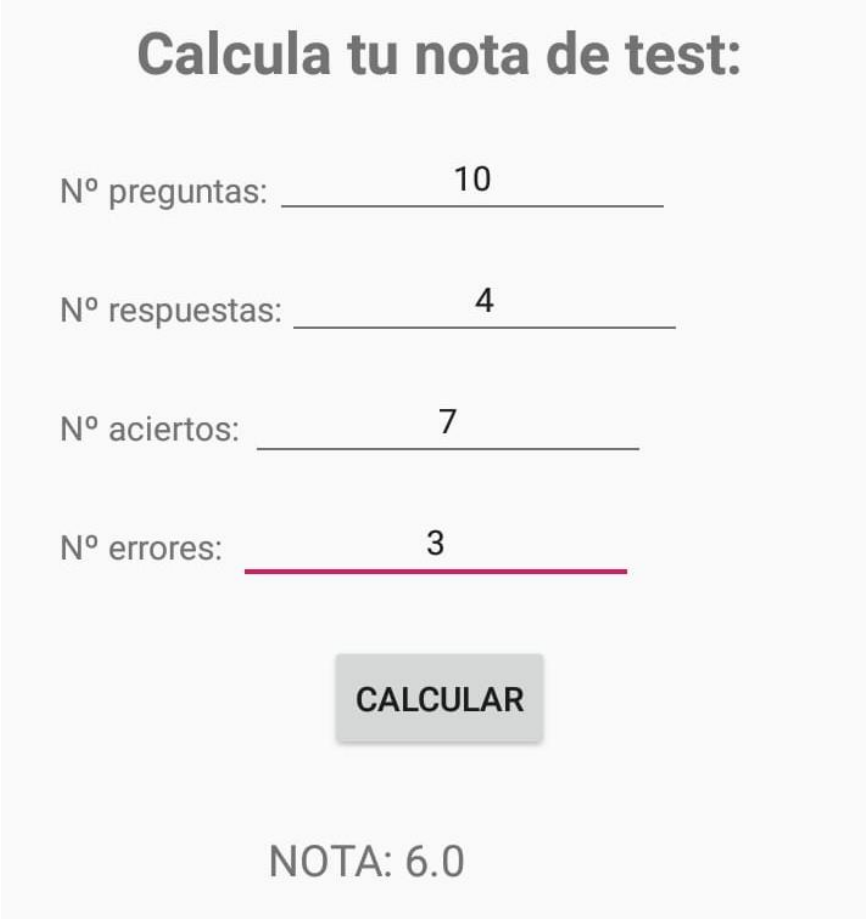


Figura 13: Ejemplo nota test.

3.4.6 Keywords

Con el objetivo de facilitar la búsqueda del contenido necesario en un determinado momento se ha implementado esta sección. Consta de un resumen de todos los contenidos de la aplicación desglosados en los conceptos más importantes. Cuando accedes a alguno de los botones que aparecen, te dirige a la parte de la aplicación donde se encuentra la información mas relevante sobre ese concepto.

El aspecto que tiene esta sección es el siguiente:

Figura 14: Pantalla keywords.

3.4.7 Acerca de

Una vez terminadas las secciones principales del menú de inicio, aparecen dos apartados secundarios y uno de ellos es el botón “Acerca de”.

Si accedes a ese botón, aparece una pantalla que ofrece información sobre el grupo de trabajo en el que se ha realizado este proyecto además de los autores de las figuras de la aplicación y el programador que la ha diseñado.

Esa pantalla tiene el siguiente aspecto:

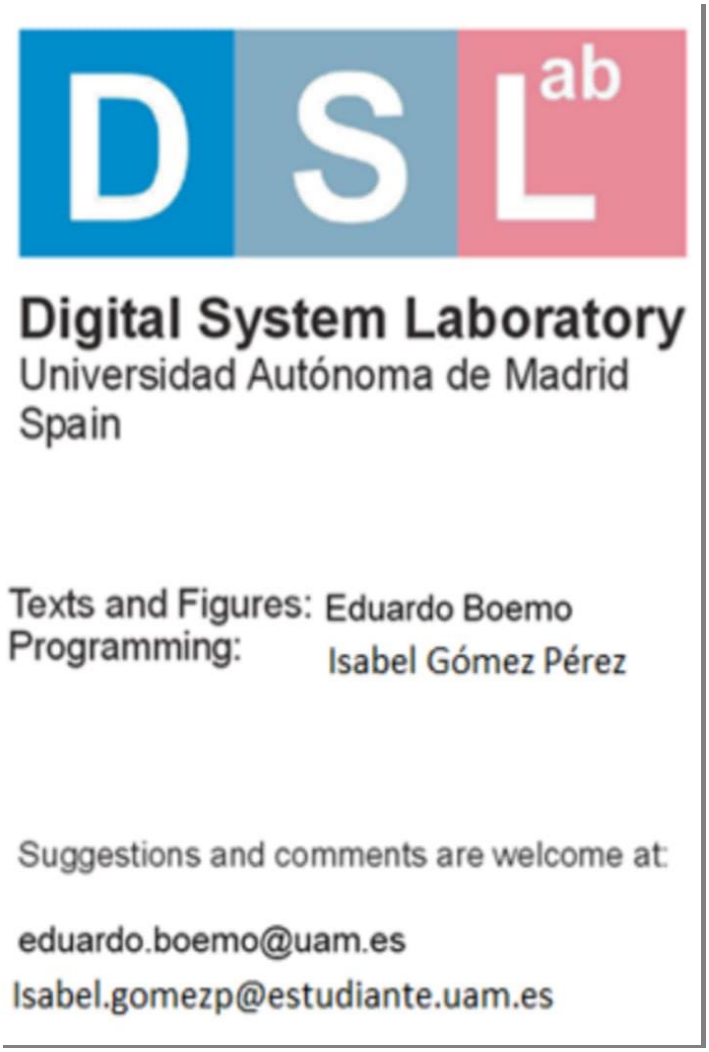


Figura 15: Pantalla Acerca de.

3.4.8 Correo

El segundo apartado secundario que aparece en el menú inicial es el botón de “Correo”. La función de este botón es poder acceder a una pantalla que te facilita el envío de correos al profesor de la asignatura.

La pantalla que aparece al acceder a dicho botón es la siguiente:

Figura 16: Pantalla enviar correo.

Como se observa en la Figura 16, esta sección de la aplicación está programada con los destinatarios necesarios, que en este caso son el profesor y la programadora de esta aplicación. De esta manera lo único que se debe escribir es el asunto del correo, el cuerpo del mensaje y darle al botón “Enviar”. Con esta herramienta, contactar con el profesor para aclarar cualquier duda que pueda surgir se convierte en algo muy rápido y sencillo.

3.5 Limitaciones

Como en todos los proyectos, a pesar de tener una idea de diseño, aparecen una serie de limitaciones que se deben tener en cuenta para diseñar la aplicación.

En este proyecto, las principales limitaciones se han dado a la hora de mostrar los aspectos teóricos por pantalla.

Una de las limitaciones impuesta para la aplicación era la necesidad de presentar la teoría a modo de diapositivas que se deslizan de manera horizontal. Esta limitación nos lleva a otra, que es la de que, al tener la información en diapositivas debes tener tanto el texto como las imágenes necesarias a un tamaño que sea visible pero que quepan en la pantalla del dispositivo.

Con muchas de las imágenes que aparecen en la aplicación, sobre todo en la sección de componentes, se ha tenido que introducir el uso del zoom. Con el uso del zoom lo que se consigue es, introducir la imagen entera en la pantalla del dispositivo móvil, pero, a la vez, se ofrece la posibilidad de ampliar la imagen para poder apreciar de forma más clara los detalles.

4 Desarrollo

En este punto se va a analizar de una manera más técnica el procedimiento seguido durante la creación de la aplicación. Se explicará cómo se han programado los aspectos más importantes, los errores que han surgido y las soluciones que se han propuesto.

En este apartado no aparecerá el código completo de la aplicación, pero sí se pondrán las partes que sean necesarias para la explicación del proyecto y las que resulten más interesantes.

4.1 Pasos previos

Debido a la falta de experiencia en trabajos relacionado con Android, antes de comenzar a programar la aplicación, se realizó una labor de investigación acerca del diseño de aplicaciones en Android y sobre programación en Java. Para ello, se han realizado las siguientes tareas:

- Consulta de apuntes sobre la introducción a la programación en Java.
- Búsqueda de ayuda en personas especializadas en Android.
- Búsqueda de aplicaciones similares con código abierto.
- Consulta de las aplicaciones anteriores realizadas en DSLab.

4.2 Herramientas

Las herramientas necesarias para el desarrollo de este proyecto son las siguientes:

- Ordenador personal.
- Software de desarrollo de aplicaciones para Android (*Android Studio*).
- Emulador de aplicaciones dentro del *Android Studio*.
- Dispositivo móvil o tableta con sistema operativo Android.
- Editor de imágenes.

4.3 Elementos principales de la aplicación

4.3.1 Actividades

Una actividad en Android es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar. Dentro del programa *Android Studio* se refiere a una actividad como "*Activity*". Se puede decir que una aplicación es un conjunto de actividades.

Cada actividad está formada por dos partes, una parte gráfica (XML) en la que se manejan los aspectos más visuales de la actividad con los que el usuario va a poder interactuar directamente. En ella se colocan los botones, los accesos para que en usuario introduzca elementos por pantalla, imágenes, etc.

La segunda parte que compone la actividad (java) es la encargada de hacer funcionar todos los elementos que se introduzcan en la parte gráfica. Esta parte es programada en el lenguaje de programación java.

Cada actividad puede estar en tres estados diferentes:

- **Activa**: se dice que una actividad está activa cuando se encuentra en primer plano. Es decir, la acción que realiza la aplicación es la suya.
- **Pausada**: se dice que una actividad está pausada cuando se encuentra en segundo plano. Es decir, aunque la aplicación esté realizando otra acción, la suya no se ha cerrado.
- **Parada**: se dice que una actividad está parada cuando otra actividad ocupa su acción por completo. Es decir, la acción de la actividad no realiza nada.

Cuando una aplicación inicia una nueva acción, detiene la anterior pero la guarda en una "pila" conservando su actividad. Una vez que el usuario termina de interactuar con la actividad y sale de ella, la saca de la pila y la destruye, dejando libre la siguiente actividad.

Todas las actividades tienen lo que se llama un ciclo de vida que se basa en el cambio de los estados anteriores. Esos cambios de estado se realizan mediante métodos *callback*, que son los siguientes:

- ***onCreate():*** se utiliza para crear una actividad y la interfaz del usuario.
- ***onStart():*** se utiliza para hacer visible la actividad al usuario.
- ***onResume():*** se utiliza para hacer posible la interacción con el usuario.
- ***onPause():*** se utiliza para pasar la actividad a segundo plano.
- ***onStop():*** se utiliza para dejar de hacer visible la actividad al usuario.
- ***onDestroy():*** se utiliza para destruir la actividad.

A continuación, se resume el ciclo de vida de una actividad en un gráfico:

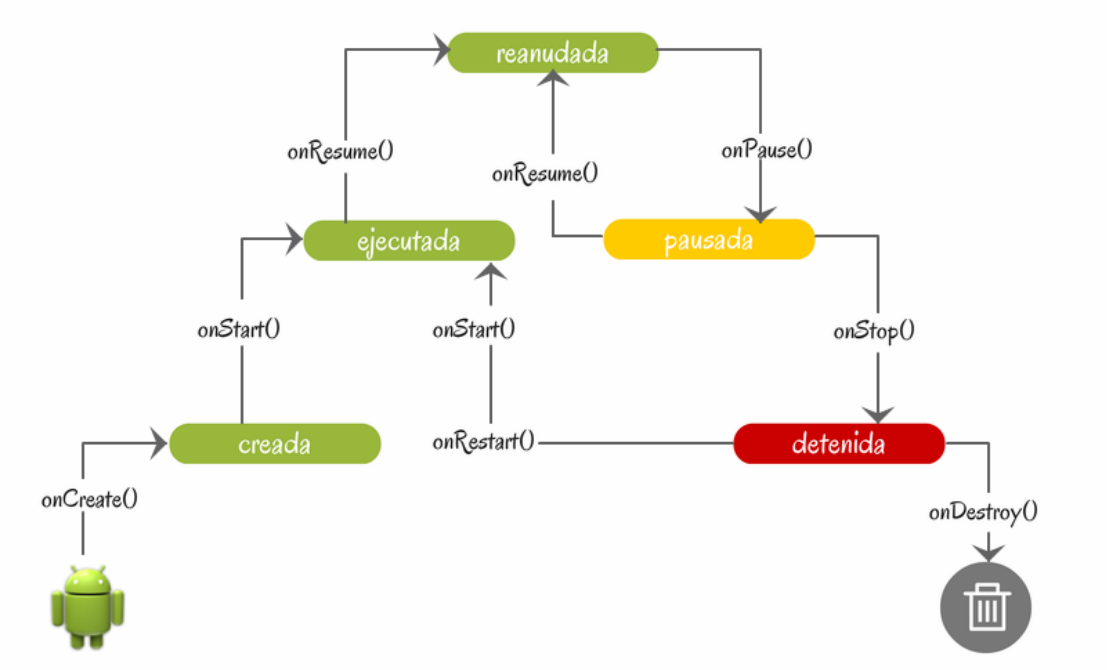


Figura 17: Ciclo de vida actividad.

4.3.2 Vistas

La vista de una actividad es el componente básico para la interfaz de usuario. La vista ocupa una parte rectangular de la pantalla. Es en esa parte donde se incluyen los elementos visuales con los que interactúa el usuario, como pueden ser botones, imágenes, campos de texto, etc. Además, dentro de una vista se pueden introducir distintos *layouts* predeterminados para disponer los elementos necesarios de manera ordenada.

Además de las vistas, existen otro tipo llamadas "vistas de grupo" o *viewgroup*. Estas son un tipo especial de vista que se caracteriza porque puede contener más vistas dentro.

4.3.3 Manifiesto

Cada aplicación tiene un archivo manifiesto (AndroidManifest.xml) en el directorio raíz. Este archivo presenta información esencial al sistema de Android, la cual tiene que estar organizada de manera correcta para poder ejecutar el código.

El documento completo del manifiesto se puede ver en la sección de anexos.

4.4 Diseño y desarrollo

Dentro del apartado de diseño y desarrollo se va a explicar de manera más detallada el código correspondiente a cada una de las partes que componen la aplicación. Además, se explicarán los conceptos teóricos que han sido necesarios estudiar para poder desarrollar cada sección.

4.4.1 Botones

De manera general y una parte común a todos los apartados, es que la aplicación está compuesta por diferentes pantallas con sus correspondientes botones que te dirigen a la sección correspondiente. Para ello, es necesario relacionar la parte XML y la parte java de la activity.

La parte XML de la activity es la que se encarga del diseño y de la apariencia de la pantalla. Lo primero que se declara es el layout, que se encarga de organizar en el espacio los componentes que se encuentran en esa pantalla. En el caso de esta aplicación, se ha utilizado el mismo layout para todas las activitys y es el siguiente:

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context=".MainActivity">
```

Aparece definido el ancho y el alto como “match parent”. Esto significa que adapta el tamaño del contenido al tamaño del layout principal, consiguiendo una adaptación total en todas las dimensiones de los dispositivos. La otra alternativa al “match parent” es el “wrap content”. Esta opción utiliza el tamaño mínimo del contenido, por lo que no garantiza la adaptación total a todos los tamaños de pantalla. Como se buscaba una buena adaptación, se ha decidido elegir el match parent.

Justo debajo se declara el tamaño de los márgenes. Para ello se utiliza el “paddingBottom/Left/Right/Top”. Se ha definido una distancia de 16dp para todos los márgenes, respecto a la parte alta de la página, a la parte baja, a la derecha y a la izquierda.

A continuación, se declara el “ScrollView”. Con esto se consigue el desplazamiento vertical y lateral por la pantalla para poder acceder a todo el contenido. En este caso, se utiliza también el “match parent” y se decide ajustar la gravedad para que quede centrado:

```
<ScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center">
```

Se define un layout dentro del anterior para determinar la orientación de los componentes que vayan dentro de forma vertical:

```
<LinearLayout
    android:id="@+id/LayoutPrincipal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
```

Con todo esto definido, simplemente se incorporan los componentes que se necesitan y automáticamente los coloca de la manera que hemos diseñado. En este caso, se ha hecho uso de los botones. Para ello se introduce, dentro de los tres apartados anteriores el siguiente código:

```
    <Button
        android:id="@+id/botonComponentes"
        android:layout_width="250dp"
        android:layout_height="60dp"
        android:layout_centerInParent="true"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"
        android:gravity="center"
        android:text="@string/componentes" />
```

Aparece primero el tipo de componente que has elegido, en este caso tipo “Button”. Para cada elemento es necesario asociarle un id. Se le asocia un ancho, un alto que determina el tamaño del botón, en este caso todos los botones son del mismo tamaño (250x60dp). Para su colocación se pone una distancia de 30dp con el elemento que tiene encima y se centra. La última línea se utiliza para definir el texto que aparecerá dentro del botón.

Una vez definidos todos los elementos que aparecerán en la pantalla, es necesario cerrar los layouts y el scroll donde se han metido:

```
</LinearLayout>
   </ScrollView>

</RelativeLayout>
```

Con la apariencia de los botones definida, hay que relacionarla con la parte java de la activity, para que, al pulsar el botón, te dirija a la pantalla que decidas. Para ello hay que comenzar importando la librería que reconoce el widget del botón:

```
import android.widget.Button;
```

Se declara la variable de tipo Button:

```
    Button botonAND;
```

Se busca la variable a través de su id con la función "findViewById" y se recoge la actividad que presenta, en este caso, cuando se pulse el botón:

```
        botonAND = findViewById(R.id.botonAND);
        botonAND.setOnClickListener(this);
```

Dentro del "Oneclick", cuando registra que se ha pulsado el botón, se lanza la función que ejecuta el cambio de activity:

```
        if (v.getId() == R.id.botonAND)
         lanzarActivityAND();
```

Al haber nombrado la función "lanzarActivity" hay que crearla para que ejecute el cambio de la activity actual a la activity que se desea. Para ello:

```
      public void lanzarActivityAND(){
              Intent i = new Intent(this,activity_AND.class);
              startActivity(i);
          }
```

4.4.2 Conceptos importantes

En este apartado aparecen 6 secciones: Protoboard, Entrenador, Montaje, Intel Hex, Debugging y Otras recomendaciones. En cada una aparece la información necesaria en formato de diapositivas.

4.4.2.1 Diapositivas

Las diapositivas correspondientes a cada sección se deslizan hacia ambos lados para moverse dentro del contenido; avanzando (hacia la derecha) o retrocediendo (hacia la izquierda).

Para ello, se ha creado una *activity* por cada página de los documentos. En esas *activitys* se ha añadido en forma de texto plano el contenido.

Para conseguir relacionar todas las *activitys* en forma de diapositivas laterales se crea una *activity* general por cada sección que se encarga de relacionarlas todas de la forma deseada. Para ello, es necesario añadir una librería que permita esta opción:

```
import android.support.v4.view.ViewPager;
```

Con esa librería añadida, se crean dos variables de tipo privado "*ViewPager*" y "*MpagerAdapter*":

```
private ViewPager mPager;
private MpagerAdapter mpagerAdapter;
```

Se añaden todas las *activitys* que se quieren relacionar dentro de una variable array de tipo "int[]" de la siguiente forma:

```
    private int[] layouts =
{R.layout.activity_protoboard1,R.layout.activity_protoboard2,R.layout.act
ivity_protoboard3,R.layout.activity_protoboard4};
```

Por último, se relacionan todas las variables creadas consiguiendo el resultado deseado:

```
mPager = (ViewPager)findViewById(R.id.viewPager);
mpagerAdapter = new MpagerAdapter(layouts,this);
mPager.setAdapter(mpagerAdapter);
```

El formato final que tienen las *activitys* de este apartado es el siguiente:

```
package es.uam.isabelgomez.lab_ced;

import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.github.barteksc.pdfviewer.PDFView;

public class activity_protoboard extends AppCompatActivity {

    private ViewPager mPager;
```

```
    private int[] layouts =
{R.layout.activity_protoboard1,R.layout.activity_protoboard2,R.layo
ut.activity_protoboard3,R.layout.activity_protoboard4};
    private MpagerAdapter mpagerAdapter;

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

        mPager = (ViewPager)findViewById(R.id.viewPager);
        mpagerAdapter = new MpagerAdapter(layouts,this);
        mPager.setAdapter(mpagerAdapter);
    }
}
```

4.4.3 Componentes

Para el apartado de Componentes hay 13 secciones disponibles. Cada una, muestra el esquema interior de cada componente.

Para ello, se ha introducido un PDF por cada componente con la información que debe aparecer en cada una. Estos documentos se agregan a la carpeta de "Assets". Una vez se tienen los documentos en el programa, se hace referencia a ellos para que aparezcan en el lugar necesario.

Para poder mostrar la información que hay en el PDF se necesita importar una librería que contenga el tipo de variable que necesitamos:

```
import com.github.barteksc.pdfviewer.PDFView;
```

Se crea la variable tipo pdfView (reflejada en la librería anterior):

```
PDFView pdfView;
```

Esa variable se relaciona con el documento correspondiente. Se busca el PDF con la función que ofrece el Android Studio a través de su ID "findViewById" y a continuación se localiza el documento en la carpeta Asset con la función "fromAsset". En este ejemplo de muestra el del del componente NAND:

```
pdfView = (PDFView)findViewById(R.id.PDFnand);
pdfView.fromAsset("74HC00 (NAND).pdf").load();
```

El código final que tienen las *activitys* de los componentes tiene el siguiente aspecto:

```
package es.uam.isabelgomez.lab_ced;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.github.barteksc.pdfviewer.PDFView;

public class activity_NAND extends AppCompatActivity {
```

```
    PDFView pdfView;

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

        pdfView = (PDFView)findViewById(R.id.PDFnand);
        pdfView.fromAsset("74HC00 (NAND).pdf").load();
    }
}
```

4.4.4 Conversor

En este apartado de la aplicación, se realiza un conversor numérico en el que se introduce un número en decimal y devuelve la conversión en binario y en hexadecimal.

Para ello, se declaran tres variables:

- Una variable de tipo “EditText”, que se encarga de recoger el número introducido.

```
EditText numero;
```

- Una variable de tipo “Button” para el botón que hay que pulsar para que se haga la conversión.

```
Button botonConvertir;
```

- Una variable de tipo “TextView” que muestra el resultado final.

```
TextView resultado;
```

Lo siguiente que se hace es recoger el número introducido en el campo “Número” y definir la configuración del botón y del texto del resultado:

```
numero = (EditText) findViewById(R.id.numero);
botonConvertir = (Button) findViewById(R.id.botonConvertir);
resultado = (TextView) findViewById(R.id.resultado);
```

A continuación, se convierte el número introducido a binario y a hexadecimal. Para ello, se hace uso de las funciones que ofrece el *Android Studio* para convertir. En nuestro caso, usamos la función “toBinaryString” y “toHexString” de la siguiente manera:

```
String bin = Integer.toBinaryString(dec);
String hex = Integer.toHexString(dec);
```

Por último, se muestra el resultado en los tres tipos de datos en el campo correspondiente:

```
resultado.setText("Decimal: " + dec + "\n\n" +
                  "Binario: " + bin + "\n\n" +
                  "Hexadecimal: " + hex);
```

El resultado final de la activity correspondiente al apartado del conversor es el siguiente:

```
package es.uam.isabelgomez.lab_ced;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class activity_conversor extends AppCompatActivity {

    EditText numero;
    Button botonConvertir;
    TextView resultado;

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

        numero = (EditText) findViewById(R.id.numero);
        botonConvertir = (Button) findViewById(R.id.botonConvertir);
        resultado = (TextView) findViewById(R.id.resultado);

        botonConvertir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                resultado.setText("Decimal: " + dec + "\n\n" +
                        "Binario: " + bin + "\n\n" +
                        "Octal: " + oct + "\n\n" +
                        "Hexadecimal: " + hex);
            }
        });}}
```

4.4.5 Checksum

El checksum es una función de redundancia que localiza cambios accidentales en una secuencia de datos y se utiliza para proteger la integridad de estos.

En este apartado se implementa una calculadora que realiza esa suma de manera automática. Se debe introducir la secuencia de datos sobra la que se quiere saber el checksum. A continuación, se detallarán los pasos a seguir para realizar esta operación.

Al tratarse de números hexadecimales, se utilizan números del 0 al 9 y letras de la A hasta la F. Es por eso por lo que se han creado botones con todos esos valores y se han añadido tres nuevos para mandar las ordenes de operar, borrar o resetear:

```
    private Button boton0, boton1, boton2, boton3, boton4, boton5,
boton6, boton7,
            boton8, boton9, botonA, botonB, botonC, botonD, botonE,
botonF, enter, borrar, borrar1char;
```

Con las variables creadas, se relacionan con su Id correspondiente para que introduzcan el valor adecuado. Por ejemplo, para el botón del 0:

```
        boton0 = findViewById(R.id.boton0);
        boton0.setOnClickListener(this);

        if (v.getId() == R.id.boton0) {
            resultado += "0";
            TextResultado.setText(resultado);
        }
```

Las tres opciones que se ofrecen en los botones de la calculadora son las siguientes:

Borrar:
Esta opción permite borrar un solo carácter de las secuencias de datos que se introducen. Para conseguir esto se implementa la siguiente función:

```
            if (v.getId() == R.id.botonBorrarUnNumero) {
                        if (resultado.length() == 0) {
                            //Nothing
                        } else if (resultado.length() == 1) {
                            resultado = "";
                            TextResultado.setText(resultado);
                        } else {
                            resultado = resultado.substring(0,
            resultado.length() - 1);
                            TextResultado.setText(resultado);
                        }
                    }
```

Clear:
Esta opción resetea todos los caracteres que están escritos en la pantalla, tanto los que se introducen, como el resultado de haber alguno. Eso se consigue con la siguiente función:

```
borrar = (Button) findViewById(R.id.botonBorrar);

borrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TextResultado.setText("");
                hexa = "";
                resultado = "";
            }
```

Calcular:
Este es el botón que se pulsa cuando has terminado de introducir la secuencia para que saque por pantalla el resultado del checksum. En esta opción también se contemplan todos los errores posibles a la hora de introducir los datos.

En caso de no introducir ningún dato y pulsar "Calcular", aparece un mensaje momentáneo diciendo "Introduce un número hexadecimal". Eso se consigue con la función "Toast":

```
if (hexa.equals("")) {
            Toast t = Toast.makeText(getApplicationContext(),
"Introduce un número hexadecimal", Toast.LENGTH_SHORT);
            t.show();
```

Un aspecto a tener en cuenta a la hora de introducir el número hexadecimal es que los caracteres deben ser pares. Si son impares se trataría de un error en la secuencia de datos. Esto también se contempla en la calculadora desarrollada. Si se da ese caso, imprime por pantalla un mensaje momentáneo diciendo "Número incorrecto, caracteres impares". Esta función continua la anterior:

```
} else if ((hexa.length() % 2) != 0) {
            Toast t = Toast.makeText(getApplicationContext(), "Número
incorrecto, caracteres impares", Toast.LENGTH_SHORT);
            t.show();
            suma_hexas_string = "";
            hexa = "";
            resultado = "";
            TextResultado.setText("");
```

En caso de introducir la secuencia de números correcta y pulsar "Calcular" se realizará la operación de checksum.
El checksum son los dos últimos caracteres del fichero de texto que define un dispositivo. Se calcula de manera que la suma de todos los bytes del campo, más el propio checksum, sea 00 (en hexadecimal). Sólo se considera el byte menos significativo de la suma. Para calcularlo se siguen los siguientes pasos:

- Se suman todos los bytes del record (en hexadecimal) de dos en dos.
- Se trunca el resultado y se usa solo el byte menos significativo.
- A lo anterior se le halla el complemento a2 y se obtiene el resultado.

Todo ese procedimiento se implementa seguido de los dos errores anteriores a través de un bucle while de la siguiente manera:

```
} else {
    int iteraciones = (hexa.length() / 2) - 2;

    String dos_ultimos = hexa.substring(hexa.length() - 2, hexa.length()); //2 ultimos HEXA
    String anteriores = hexa.substring(hexa.length() - 4, hexa.length() - 2); //2 siguientes HEXA
    int primeros = Integer.parseInt(dos_ultimos, 16);
    int segundos = Integer.parseInt(anteriores, 16);
    int suma_hexas = primeros + segundos; //suma - ultimos 4 HEXAs
    String primera_suma = Integer.toHexString(suma_hexas);
    suma_hexas_string = primera_suma.substring(primera_suma.length() - 2, primera_suma.length());
    hexa = hexa.substring(0, hexa.length() - 4);

    while (iteraciones >= 1) {
        //Transformo a decimal los siguientes 2 HEXAs
        String siguienteHexa = hexa.substring(hexa.length() - 2, hexa.length());
        int siguientesHexa = Integer.parseInt(siguienteHexa, 16);

        //Transformo a decimal la suma guardada de los anteriores HEXAs
        int suma_hexa_dec = Integer.parseInt(suma_hexas_string, 16);

        //Realizo la suma decimal
        int suma_prov = siguientesHexa + suma_hexa_dec;

        //Paso la suma a HEXA
        String prov = Integer.toHexString(suma_prov);

        //Trunco el resultado y lo guardo en la variable global
        suma_hexas_string = prov.substring(prov.length() - 2, prov.length());

        //Preparo las variables para la siguiente iteracion de bucle
        if (hexa.length() > 2)
            hexa = hexa.substring(0, hexa.length() - 2);
        iteraciones--;
    }
}
//suma_hexas_string -> resultado suma en string
//Le resto a 100 la suma total de los HEXAs y trunco el resultado
int resultado_final = BASE - Integer.parseInt(suma_hexas_string, 16);
suma_hexas_string = Integer.toHexString(resultado_final);
```

Como se realizan numerosos pasos a lo largo del código, se han ido añadiendo comentarios para indicar qué se esta haciendo en cada momento.

Por último, simplemente se muestra el resultado final del checksm por pantalla a través de un texto plano:

```
TextResultado.setText(suma_hexas_string);
```

En vista de que se pueda volver a realizar otra operación con una secuencia distinta, se resetean todos los valores para poder introducir unos nuevos:

```
suma_hexas_string = "";
hexa = "";
resultado = "";
```

4.4.6 Nota test

Durante la asignatura se harán diferentes pruebas para evaluar los conocimientos adquiridos por los alumnos. Estas pruebas serán tipo test. La ponderación de los test siempre varía y resulta tedioso calcular la nota final. Es por eso que se ha incorporado una calculadora de nota test.

Para obtener la nota del examen únicamente se debe introducir los siguientes datos:

- Número de preguntas que hay en total.
- Número de respuestas entre las que se puede responder.
- Número de aciertos.
- Número de fallos.

Para recoger estos datos, se declaran cuatro variables de texto editable:

```
EditText preguntas, respuestas, aciertos, errores;
```

En esas variables se recoge el dato introducido por el alumno de la siguiente manera:

```
preguntas = (EditText)findViewById(R.id.preguntas);
respuestas = (EditText)findViewById(R.id.respuestas);
aciertos = (EditText)findViewById(R.id.aciertos);
errores = (EditText)findViewById(R.id.errores);

int preg = Integer.parseInt(preguntas.getText().toString());
int res = Integer.parseInt(respuestas.getText().toString());
int ac = Integer.parseInt(aciertos.getText().toString());
int err = Integer.parseInt(errores.getText().toString());
```

Sabiendo esos datos se aplica la fórmula correspondiente. Se asume que las preguntas sin contestar ni suman ni restan, pero las preguntas falladas conllevan una penalización en la nota. Esto es así para evitar las preguntas respondidas al azar. La nota final es el resultado de las notas individuales en cada pregunta. En cambio, la nota esperada del test se obtiene mediante la esperanza.

La nota de cada pregunta que se ha respondido puede tener dos valores: 1 si es correcta y p si es incorrecta, donde p representa el valor que penalizan las preguntas falladas. Cada pregunta tiene k posibles respuestas. La probabilidad de acertar al responder al azar es de 1/k mientras que la probabilidad de fallar es de 1-1/k. La nota n de cada pregunta i viene dada por la siguiente función:

$$n_i = \begin{cases} 1, & con \quad probabilidad \quad \frac{1}{k} \\ -p, & con \quad probabilidad \quad 1 - \frac{1}{k} \end{cases}$$

La esperanza es el resultado de la suma de cada multiplicación entre el valor y la probabilidad de cada pregunta:

$$E[n_i] = 1 \cdot \frac{1}{k} + (-p) \cdot \left(1 - \frac{1}{k}\right) = \frac{1}{k} - p \cdot \frac{k-1}{k} \Rightarrow$$

$$\Rightarrow E[n_i] = \frac{1 - p(k-1)}{k}$$

Para la nota final del test (NOTA) se calcula la esperanza. Para ello se hace uso de la linealidad, que determina que la esperanza de la suma corresponde con la suma de las esperanzas. La siguiente expresión muestra el resultado de responder todas las preguntas de manera aleatoria:

$$E[NOTA] = E\left[\sum_{i=1}^{N} n_i\right] = \sum_{i=1}^{N} E[n_i] =$$

$$= \sum_{i=1}^{N} \frac{1 - p(k-1)}{k} = N \cdot \frac{1 - p(k-1)}{k}$$

El resultado de responder al azar debe corresponder con un 0 en la nota final, evitando así notas negativas. Para conseguir eso se iguala la expresión a 0 y se despeja el valor de p:

$$E[NOTA] = 0 \Rightarrow N \cdot \frac{1 - p(k-1)}{k} = 0 \Rightarrow$$

$$\Rightarrow 1 - p(k-1) = 0 \Rightarrow p = \frac{1}{k-1}$$

Se obtiene, finalmente, la ecuación que representa la nota de los test:

$$\boxed{Nota = Aciertos - \frac{Errores}{k-1}}$$

Esta fórmula se introduce en el código de la siguiente manera, mostrando el resultado final en un cuadro de texto:

```
double  preg_d = preg*1.0;
double  res_d = res*1.0;
double  ac_d = ac*1.0;
double  err_d = err*1.0;
double  notafinal = ((ac_d-(err_d/(res_d-1)))/preg_d)*10;

        nota.setText("NOTA: " + notafinal );
```

4.4.7 Keywords

A modo de buscador y con el fin de agilizar la búsqueda de un elemento concreto, se ha implementado esta sección.

Aparece un botón con cada concepto básico, si pulsas en alguno de ello, la app te dirige a la parte exacta donde se encuentra la información necesaria. Puede dirigirte a una página en concreto del apartado de conceptos importantes, a algún componente, incluso a alguna de las herramientas prácticas.

Para conseguir esto se ha tenido que crear una activity que contenga un botón con cada concepto. Ese botón se declara en el fichero java y se relaciona con la activity, ya creada, correspondiente.

El proceso es el mismo que se ha explicado en la sección 4.4.1 Botones.

4.4.8 Acerca de

La interfaz de este apartado simplemente consta de una imagen con la información sobre el desarrollo de la aplicación.

Dicha imagen se ha introducido en formato PDF, de la misma manera que se ha hecho en el apartado de “Conceptos importantes” y “Componentes”.

El código resultante de esta *activity* es el siguiente:

```
package es.uam.isabelgomez.lab_ced;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class activity_info extends AppCompatActivity {

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

4.4.9 Correo

Se desarrolla este apartado con el fin de facilitar la comunicación con el profesor de la asignatura para resolver cualquier tipo de consulta o de duda. La manera más sencilla es a través del correo electrónico, por lo que se implementa la opción de mandar ese correo de forma rápida y simple.

Una de las mejores ventajas que presenta esta sección es que únicamente se tiene que introducir el asunto del correo y el cuerpo del mensaje, el resto de las opciones están configuradas previamente con los datos del profesor correspondiente.

Para ello, se introducen en el código los datos necesarios para enviar un email. Se utilizará la aplicación que disponga el dispositivo para enviar el correo. A continuación, se recogen los datos que introduce el alumno en el campo de asunto y cuerpo de mensaje y, una vez pulsado el botón de enviar, se realiza la acción de enviar lo escrito.

Para todo eso, se utiliza el siguiente código:

```
protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_correo);

        mSubjectEt = findViewById(R.id.subjectEt);
        mMessageEt = findViewById(R.id.messageEt);
        mSendEmailBtn = findViewById(R.id.sendEmailBtn);

        mSendEmailBtn.setOnClickListener(new View.OnClickListener(){

            @Override
            public void onClick(View v){

                String subject = mSubjectEt.getText().toString().trim();
                String message = mMessageEt.getText().toString().trim();
                String subjectComplete = "App CED LAB: " + subject;
                String[] to = { "isabel.gomezp@estudiante.uam.es" };
                String[] cc = { "eduardo.boemo@uam.es" };
                sendEmail(to,cc,subjectComplete,message);

            }        });
```

Una vez que el correo ha sido enviado, la aplicación vuelve a la pantalla principal y muestra un mensaje informando que el correo ha sido enviado. Esto se implementa de la siguiente manera:

```
    private void sendEmail(String[] to, String[] cc,String asunto, String
mensaje) {

        Intent emailIntent = new Intent(Intent.ACTION_SEND);
        emailIntent.setData(Uri.parse("mailto:"));
        emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
        emailIntent.putExtra(Intent.EXTRA_CC, cc);
        emailIntent.putExtra(Intent.EXTRA_SUBJECT, asunto);
        emailIntent.putExtra(Intent.EXTRA_TEXT, mensaje);
        emailIntent.setType("message/rfc822");

        //Vuelve a la pantalla de inicio cuando se pulsa el boton de
enviar
        Intent inicio = new Intent(this, MainActivity.class);
        startActivity(inicio);

        try{
            startActivity(Intent.createChooser(emailIntent,"Send by:"));

        }catch (Exception e){

Toast.makeText(this,e.getMessage(),Toast.LENGTH_SHORT).show();
        }
    }
```

5 Integración, pruebas y resultados

5.1 Publicación de la aplicación

Una vez terminada la aplicación y lista para su uso, se decide publicarla para que sea accesible a toda persona que disponga de un dispositivo Android. La plataforma elegida para la publicación es la de Google Play Store, debido a su enorme cantidad de usuarios.

El primer paso que hay que dar es generar el fichero APK de la aplicación. Se trata de un fichero que contiene toda la información necesaria para que funcione como se ha diseñado.

Esto se hace desde el Android Studio. En la pestaña "Build" se selecciona la opción "Generate Signed Bundle/APK". Aparece una ventana en la que se selecciona la opción de "APK" seguido de la opción "Next".

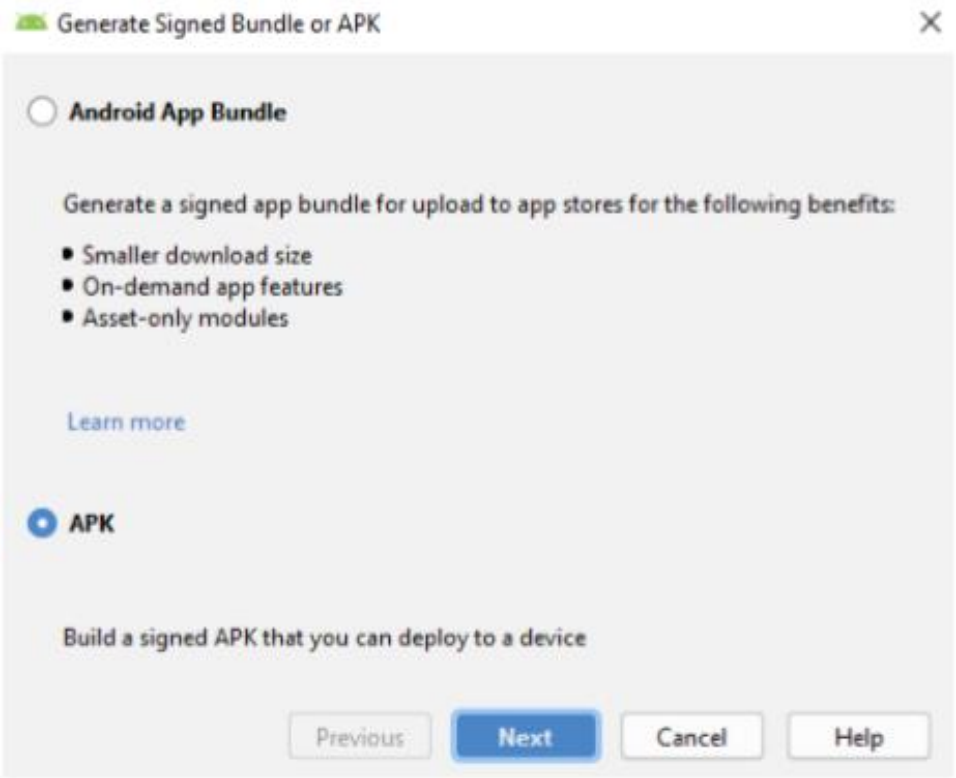


Figura 18: Generar APK 1.

Cómo lo que se pretende es lanzar la aplicación, se selecciona “reléase”, la otra opción “debug” se utiliza para probarla. Se termina este proceso saliendo con la opción “Finish”.

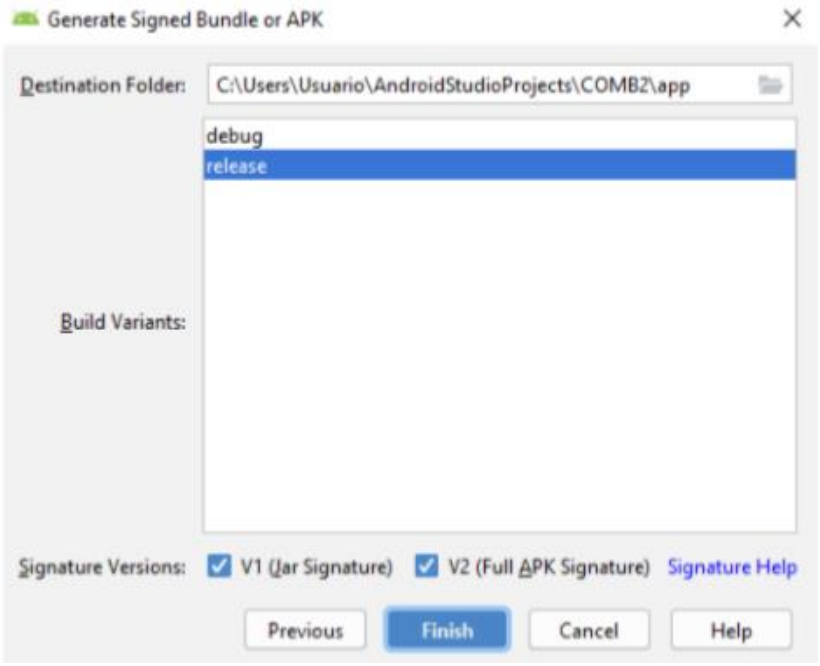


Figura 19: Generar APK 2.

Antes de generar el fichero APK, es importante firmar la aplicación con un certificado digital. Esto permite la opción de modificarla una vez publicada, de poder lanzar actualizaciones sobre la misma aplicación. Para ello, se debe crear un *Key Store* de la siguiente manera:

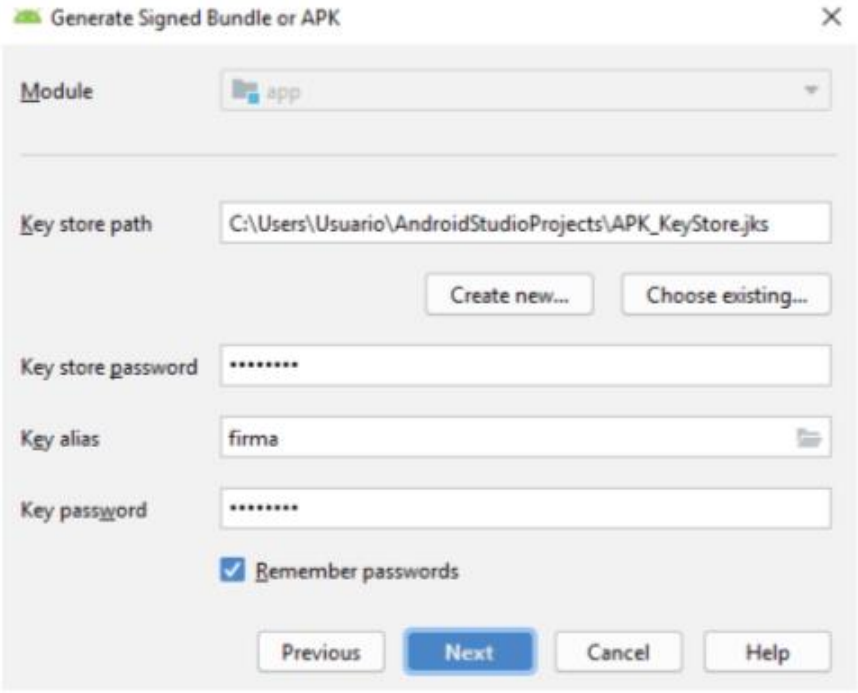


Figura 20: Generar firma.

Con el archivo firmado de esta manera, se genera el fichero APK necesario. El lugar donde se guarda es en la carpeta app\release del proyecto.

El segundo paso que hay que dar para la publicación, es el de conectarse a la plataforma de Google Play Store. Para ello es necesario registrarse con un correo y acceder así a la Consola de Android Developers.

En este caso se ha utilizado la cuenta que nos ha facilitado el laboratorio DSLab de la universidad.

Antes de proceder con la subida de la aplicación, se debe preparar el material necesario para ello. Con ese material disponible, se añade la aplicación y se rellena la ficha que la acompaña. En ella te pide información básica, como puede ser el logo, una breve descripción, imágenes significativas y adjetivos que la describan. Todos esos datos son utilizados para crear el formato que se ve a la hora de descargarla.

Todos los datos introducidos deben cumplir con unos requisitos de calidad mínimos. Además, aparecen otros requisitos en cuanto al contenido. No puede aparecer materia inadecuada (violenta, inadecuada, etc). Para asegurarse de que se cumple es necesario rellenar un test que evalúa el contenido de la app. Con ese test se obtienen una serie de etiquetas de clasificación. En nuestro caso, al tratarse de una aplicación educativa, es apta para todos los públicos y obtiene las siguientes etiquetas:

Figura 21: Clasificación app.

Con la aplicación publicada en esta plataforma, se estudiará el impacto y las descargas que tiene cuando se utilice durante el siguiente curso.

5.1 Pruebas

Se han realizado pruebas de sistema y aceptación. Estas consisten en comprobar que la aplicación funciona en el soporte al que va dirigido, en este caso, dispositivos Android y que reacciona de la manera esperada. Para ello, se ha instalado la app en diferentes dispositivos (móviles y tabletas) con sistema operativo Android. En ellos, se han verificado los siguientes puntos:

- Descarga e instalado. Se comprueba el tiempo de descarga y que el aparato no presenta ningún error a la hora de instalarlo, que sigue funcionando correctamente.
- Acceso. Se comprueba que todos los apartados desarrollados son accesibles sin inconvenientes. Se prueba para ello, todas las opciones y botones posibles.
- Contenido. Se comprueba que el contenido que aparece es el correcto, que corresponde con la última versión realizada.
- Diseño. Se comprueba que la interfaz tiene el diseño creado en el Android, ya que este puede variar en función de los tamaños de pantalla. Se verifica que aparece todo con la misma forma con la que se diseñó.
- Tiempos. Se comprueba que los tiempos de acceso y carga se mueven en valores normales y aceptables para los usuarios.
- Entrada y salida. Se comprueba que entra de manera correcta a la aplicación y que no presenta errores al cerrarla y liberar su memoria.

Tras realizar las pruebas, se deduce un resultado positivo en todas, por lo que no será necesario hacer ninguna modificación ni actualización del proyecto.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El objetivo principal de este proyecto era facilitar el estudio de la asignatura de Circuitos Electrónicos Digitales, tanto la teoría como la práctica, a los alumnos. El hacerlo a través de una aplicación para móvil tenía como propósito aprovechar el auge que las tecnologías están teniendo.

Con el proyecto terminado, se puede decir que se ha cumplido ese objetivo y, en muchos aspectos, se ha superado. A medida que se avanzaba en el desarrollo de la aplicación surgieron muchos problemas y muchas posibles mejoras que se han implementado de manera satisfactoria.

Se han estudiado las necesidades básicas para estudiar la asignatura de primero de carrera de la manera más eficaz posible. Sabiendo estas necesidades, se han desarrollado las herramientas óptimas para cubrirlas. El resultado final ha sido una aplicación muy completa que cumple con los requisitos propuestos.

Los apartados que comprende la aplicación son tan útiles como diversos y son los siguientes:

- Conceptos importantes.
- Componentes.
- Conversor numérico.
- Calculadora nota test.
- Calculadora Checksum.
- Información acerca de la aplicación.
- Correo.

Implementando todos esos apartados se ha conseguido aprender y formarse en muchos campos que antes se desconocían, como pueden ser los siguientes:

- Lenguaje de programación java.
- Lenguaje de programación xml.
- Desarrollo de aplicaciones Android.
- Uso de la herramienta Android Studio.
- Publicación de aplicaciones en Play Store.

Se puede concluir diciendo que se trata de un trabajo útil, tanto para aprender al desarrollarlo como para ayudar a entender la asignatura de Circuitos Electrónicos Digitales.

6.2 Trabajo futuro

A pesar de que se trata de una aplicación para una asignatura específica, es muy susceptible a muchas modificaciones para ampliar su contenido y hacerla más general. El hecho de que se haya programado de forma sencilla y simplificada facilita estas modificaciones. No se han planteado para este proyecto, pero cabe la posibilidad de realizar muchas.

Una modificación importante, podría ser la de diseñar la aplicación en varios idiomas. Adaptar el contenido a distintos idiomas y que se ofrezca la posibilidad de elegir el idioma deseado antes de comenzar.

En cuanto al contenido, se puede ampliar orientado a muchos aspectos, consiguiendo así una aplicación más general y completa. Algunas opciones pueden ser las desarrolladas en otros proyectos, como añadir un método de calificación que evalúe los conocimientos adquiridos. Otra opción podría ser la de añadir la posibilidad de realizar ejercicios que ayuden a completar los conocimientos relacionados con la teoría.

Respecto al diseño de la aplicación, hay muchas mejoras posibles. Dentro de la herramienta de Android Studio hay muchas opciones que añadir al diseño. Se pueden cambiar formas, tamaños y colores al gusto para hacerla más visible e intuitiva.

A la hora de subir la aplicación a la plataforma de descarga, se ofrece la posibilidad de añadir publicidad y monetizar así, la descarga de la aplicación. Podría ser otra posible mejora en actualizaciones futuras.

Referencias

Las referencias utilizadas para el desarrollo de este proyecto se encuentran, mayoritariamente *online*. Las más importantes se muestran a continuación:

[1] http://arantxa.ii.uam.es/~cedeps/ced-comp.htm

[2] https://www.youtube.com/watch?v=mMQZt7wLLfc&t=1s

[3] Calidad y Auditoria Software, Universidad de Cantabria, 2017.

[4] https://developer.android.com/studio

[5] https://stackoverflow.com/

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

[7] Miguel Ángel Morales. "Las matemáticas de la fórmula de puntuación de exámenes test", El País, Febrero 2017

[8] J.L González-Santander y G. Martín. "Análisis para la calificación de pruebas tipo test multi-respuesta", Nereis, Noviembre 2010.

[9] https://play.google.com/store/apps

[10] Fco Javier Ceballos, "Java 2 Curso de Programación", 4º edición, 2010.

[11] Jesús Tomás Gironés, "El gran libro de Android", MARCOMBO, S.A.

[12] Material "Desarrollo de aplicaciones para entornos móviles". EPS UAM.

[13] http://es.wikipedia.org/wiki/Android

[14] http://www.androidcurso.com

[15] Apuntes de clase de Circuitos Electrónicos Digitales, Escuela Politécnica Superior – UAM, 2015.

[16] http://codictados.com/

Glosario

ADT	Android Development Tools.
API	Application Programming Interface.
APK	Android Application Package.
AVD	Android Virtual Device.
CED	Circuitos Electrónicos Digitales.
DIE	Dispositivos Integrados Especializados.
DSLab	Digital System Laboratory.
EPS	Escuela Politécnica Superior.
IDE	Integrated Development Enviornment.
PDF	Portable Document Format.
SDK	Software Development Kit.
SO	Sistema Operativo.
TFG	Trabajo de Fin de Grado.
UAM	Universidad Autónoma de Madrid.
UI	Interfaz de Usuario.
XML	eXtensible Markup Language.

Anexos

A Documento manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="es.uam.isabelgomez.lab_ced">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".activity_correo"></activity>
        <activity android:name=".activity_info" />
        <activity android:name=".activity_memoria2" />
        <activity android:name=".activity_memoria1" />
        <activity android:name=".activity_debugging3" />
        <activity android:name=".activity_debugging2" />
        <activity android:name=".activity_debugging1" />
        <activity android:name=".activity_encapsulados6" />
        <activity android:name=".activity_encapsulados5" />
        <activity android:name=".activity_encapsulados4" />
        <activity android:name=".activity_encapsulados3" />
        <activity android:name=".activity_encapsulados2" />
        <activity android:name=".activity_encapsulados1" />
        <activity android:name=".activity_entrenador5" />
        <activity android:name=".activity_entrenador4" />
        <activity android:name=".activity_entrenador3" />
        <activity android:name=".activity_entrenador2" />
        <activity android:name=".activity_entrenador1" />
        <activity android:name=".activity_protoboard4" />
        <activity android:name=".activity_protoboard3" />
        <activity android:name=".activity_protoboard2" />
        <activity android:name=".activity_protoboard1" />
        <activity android:name=".activity_recomendaciones3" />
        <activity android:name=".activity_recomendaciones2" />
        <activity android:name=".activity_recomendaciones1" />
        <activity android:name=".activity_intel11" />
        <activity android:name=".activity_intel1" />
        <activity android:name=".Main2Activity" />
        <activity android:name=".activity_intel10" />
        <activity android:name=".activity_intel9" />
        <activity android:name=".activity_intel8" />
        <activity android:name=".activity_intel7" />
        <activity android:name=".activity_intel6" />
```

```
        <activity android:name=".activity_intel5" />
        <activity android:name=".activity_intel4" />
        <activity android:name=".activity_intel3" />
        <activity android:name=".activity_intel2" />
        <activity android:name=".activity_Intel1" />
        <activity android:name=".activity_Intel" />
        <activity android:name=".activity_253" />
        <activity android:name=".activity_175" />
        <activity android:name=".activity_163" />
        <activity android:name=".activity_151" />
        <activity android:name=".activity_74" />
        <activity android:name=".activity_recomendaciones" />
        <activity android:name=".activity_entrenador" />
        <activity android:name=".activity_encapsulados" />
        <activity android:name=".activity_debugging" />
        <activity android:name=".activity_checksum" />
        <activity android:name=".activity_calculadora" />
        <activity android:name=".activity_conversor" />
        <activity android:name=".activity_NOR" />
        <activity android:name=".activity_xor" />
        <activity android:name=".activity_OR" />
        <activity android:name=".activity_NAND" />
        <activity android:name=".activity_INV" />
        <activity android:name=".activity_AND" />
        <activity android:name=".MainActivity" />
        <activity android:name=".activity_splash">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"
/>
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".activity_componentes" />
        <activity android:name=".activity_protoboard" />
        <activity android:name=".activity_reglas" />
    </application>

</manifest>
```