UNIVERSIDAD AUTONÓMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA
Ingeniería de Telecomunicación

APLICACIÓN ANDROID PARA RESOLVER PROBLEMAS DE CIRCUITOS DIGITALES SECUENCIALES.

Cristina María Sobrino Hipólito

Diciembre 2015

APLICACIÓN ANDROID PARA RESOLVER PROBLEMAS DE CIRCUITOS DIGITALES SECUENCIALES.

AUTOR: Cristina María Sobrino Hipólito
TUTOR: Federico García Salzmann
PONENTE: Eduardo Boemo

DSLab
Dpto. de Tecnología Electrónica y Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Diciembre de 2015

Resumen

Este proyecto consiste en el diseño y desarrollo de una aplicación móvil dirigida a los alumnos de la asignatura Dispositivos Integrados Especializados de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. Dicha aplicación consiste en una serie catorce problemas de contadores a solucionar por el alumno, un tutorial sobre estos circuitos secuenciales y un test online.

El objetivo principal de la aplicación es ofrecer a los alumnos una mejora de las guías de problemas que se proporcionan en la asignatura, pudiendo éstos resolverse desde cualquier dispositivo Android, además de poder corregir los ejercicios cuando el usuario lo desee y compartir las soluciones por correo electrónico.

Palabras clave

Contadores, Android, circuitos secuenciales, electrónica digital, aplicación, móvil, tableta.

Abstract

This Project involves the design and development of a mobile application for students of the Specialized Integrated Devices subject in Escuela Politécnica Superior of the Universidad Autónoma de Madrid. This application consists in a set of fourteen problems to be solved by the students, a tutorial of these sequential circuits and an online test.

The main objective of the application is offer to students an improvement of the exercises guides provided in the subject so they can resolve the problems from any Android device, correct the exercises when they wish and share solutions by email.

Key words

Counters, Android, sequential circuits, digital electronic, application, mobile, tablet.

Agradecimientos

En primer lugar, me gustaría agradecer a mis padres y a mi hermana por estar siempre ahí, por enseñarme a ser quien soy y por haber hecho posible que llegue hasta aquí.

Agradecer a toda mi familia por ayudarme en todo siempre en especial a mis abuelos, Carmen y Wences, que lo han hecho todo por nosotras. Por los valores que me han inculcado.

A Pablo, uno de los pilares de mi vida. Por su apoyo incondicional, por comprenderme y porque sin él no hubiera llegado al final.

A Antonio y Laura, por acogerme y tratarme como a una más.

Agradecer también a Eduardo, por haberme brindado la oportunidad de realizar este proyecto.

Gracias a todos mis amigos y, en definitiva, a cualquier persona que haya aportado algo en mi vida.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Motivación

Actualmente, las guías de problemas que se facilitan en la asignatura Dispositivos Integrados Especializados de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid se entregan en formato PDF, es decir, en papel. Además, dichas guías no ofrecen soluciones parciales, ni permiten corregir los ejercicios en cualquier momento. Por este motivo, se busca otra forma de proporcionar los ejercicios de la asignatura.

Desde hace unos años hasta ahora, tanto los teléfonos móviles como las tabletas inteligentes se han convertido en algo esencial en nuestro día a día debido a las facilidades que proporcionan a la hora de realizar acciones como buscar en internet o, simplemente entretenernos jugando mientras vamos en el metro. Las aplicaciones móviles cada vez se encuentran más presentes en la rutina diaria tanto de jóvenes como de adultos.

Es por esto que surge la idea de aprovechar las ventajas que ofrecen estos dispositivos inteligentes y realizar una aplicación móvil que puedan aprovechar los estudiantes añadiendo, a la vez, ventajas de las que carece el formato tradicional tales como:

- Incorporación de herramientas de corrección de los ejercicios
- Mayor interacción con el profesor a través del envío de los ejercicios por correo electrónico.
- Eliminación del papel
- Comodidad de uso
- Mayor accesibilidad para los estudiantes y para cualquier persona interesada en la resolución de este tipo de ejercicios.
- Versionar soluciones de manera sencilla.
- Utilización de la aplicación sin necesidad de disponer de internet, exceptuando el test.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación móvil destinada a los alumnos 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 UAM. Dicha aplicación dispondrá de un tutorial, donde se explican los conceptos básicos sobre Contadores, una guía con catorce problemas y un test online.

Los problemas se centrarán en el análisis de contadores. La aplicación permitirá al usuario solucionar los problemas y la posibilidad de corregirlos. Además, los ejercicios podrán guardarse, cargarse o enviarse vía email.

La aplicación debe seguir estrictamente el estilo de las aplicaciones anteriores realizadas para la asignatura de Circuitos Electrónicos Digitales de primer curso del Grado en Ingeniería de

Tecnologías y Servicios de Telecomunicaciones de la EPS. Estas restricciones incluyen específicamente los siguientes aspectos:

- 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

La aplicación se desarrollará para el sistema operativo Android y será ofrecida a los alumnos a través de la plataforma Google Play en la que podrá descargarse gratuitamente. Podrá valorarse y proponer mejoras para un futuro.

Desde el punto de vista educativo, el objetivo del proyecto es adquirir conocimientos de Java, dado que no hay una asignatura obligatoria en la carrera con la que se aprenda dicho leguaje de programación. También es útil para adquirir conocimientos de la plataforma Android y para aprender a subir aplicaciones al Google Play.

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 resultados
- Capítulo 6: Conclusiones y trabajo futuro

2 Estado del arte

En este apartado se va a realizar un breve estudio sobre la evolución de los dispositivos móviles y sobre los principales sistemas operativos activos actualmente, poniendo especial atención en Android, ya que es el sistema operativo elegido para realizar la aplicación.
Además, se realizará un estudio de las aplicaciones similares a la desarrollada que se encuentran en el mercado actualmente.

2.1 Evolución de los smartphones.

Actualmente se denomina smartphone a la familia de teléfonos móviles que disponen de un hardware y un sistema operativo propio capaz de realizar tareas y que, demás, poseen funcionalidades similares a las de un ordenador. Se consideran, por tanto, la evolución tecnológica a los teléfonos móviles clásicos.

Los primeros teléfonos inteligentes se diferenciaron de los móviles clásicos en la incorporación de funcionalidades extras como organizadores personales, es decir, bloc de notas, calendario, gestor de correo electrónico. Además incorporaron un teclado Qwerty que facilitaba la escritura en el teléfono. El objetivo de todas estas funcionalidades no es otro que asemejarse a un ordenador, pero con la ventaja de tenerlas en un dispositivo móvil.

La primera comunicación telefónica se realizó entre dos poblados que se encontraban a diez kilómetros de distancia el 16 de Agosto de 1876 por Graham Bell. En 1973 Martin Cooper realizó la primera llamada telefónica desde un móvil

En 1993, veinte años después de la primera llamada móvil, IBM lanzó al mercado el primer smartphone con el nombre "Simon". Este teléfono incorporaba todas las funcionalidades de un PDA de aquella época con capacidades telefónicas y de SMS. Además, incluía pantalla táctil.

Figura 2-1: Primer Smartphone IBM "Simon"

Durante la década de los 90 y comienzos del 2000, los fabricantes lanzaron varios modelos de teléfonos inteligentes sin obtener éxito debido a que eran dispositivos caros y no tenían mucho valor entre los consumidores.

Sin embargo, en 2007 se revoluciona el concepto de teléfono inteligente con la presentación en el mercado de iPhone por parte de la empresa Apple. Este modelo incluía pantalla táctil a color, una excelente conexión a internet y unas aplicaciones realmente útiles para el usuario. Apple se convirtió en la empresa líder del sector batiendo todos los records de venta pronosticados.

Figura 2-2: Primer iPhone del mercado

En los años sucesivos empresas como Samsung, LG o Nokia han diseñado y fabricado smartphones bajo el sistema operativo Android, dado que es un sistema operativo de código abierto, compitiendo con el iPhone. Esto ha provocado que el mercado actualmente disponga de una amplia de gama de dispositivos inteligentes alcanzables por todos los usuarios.

Figura 2-3: Varios móviles del mercado actual

2.2 Sistemas Operativos para móviles

En este apartado se van a exponer los principales sistemas operativos para smartphones. Se pondrá especial énfasis en el sistema operativo Android ya que es el que ha sido elegido para el desarrollo de la aplicación.

2.2.1 Android

2.2.1.1 Características

Android es un sistema operativo basado en el núcleo de Linux y está enfocado todo tipo de dispositivos móviles y sean teléfonos móviles o tabletas. En un principio fue desarrollado por Android Inc., empresa que fue absorbida por Google.

Se presentó en 2007 para avanzar en el desarrollo de estándares abiertos y en Octubre del año siguiente salió al mercado el primer móvil con este sistema operativo: El HTC Dream.

La versión básica de este sistema operativo se conoce como Android Open Source Project. Android es una plataforma de código abierto, el cual ha sido liberado por Google bajo la licencia de Software Apache (licencia de software libre y de código abierto). Esto quiere decir, que cualquier desarrollador puede crear aplicaciones con diferentes lenguajes y compilarlas a código nativo de ARM.

Algunas de las características más relevantes de Android son las siguientes:

- Framework de aplicaciones: Las partes de una aplicación pueden reutilizarse para otras aplicaciones e incluso se pueden sustituir componentes integrados por otros.

- Navegador integrado: Basado en el motor de proyecto abierto WebKit.

- Base de datos SQlite para el almacenamiento estructurado que se integra con las aplicaciones.

- Gráficos y sonido optimizados: Los gráficos 2D están suministrados por una librería gráficos 2D, mientras que los gráficos 3D están basados en la especificación Open GL ES. Soporta formatos comunes de audio como MP3.

- Android utiliza su propia máquina virtual Dalvik, basada en la de Java.

- Google Play: Android cuenta con un mercado de aplicaciones gratuitas o de pago que pueden ser descargadas al dispositivo móvil sin necesidad de un ordenador.

- Otra característica de Android es que las aplicaciones que no están ejecutándose en primer plano reciben ciclos de reloj.

- Android permite a los dispositivos ser usados como puntos de acceso inalámbricos.

2.2.1.2 Versiones de Android.

A continuación se muestra una tabla con las todas versiones oficiales del este sistema operativo. Todas ellas reciben el nombre de un postre o un dulce y en cada una el nombre del postre empieza con una letra del alfabeto.
Cada nueva versión ha ido mejorando a las funcionalidades y los errores de las anteriores:

VERSION	NOMBRE	API
Android Beta	Beta	-
Android 1.0	Apple Pie	1
Android 1.1	Banana Bread	2
Android 1.5	Cupcake	3
Android 1.6	Donut	4
Android 2.0	Eclair	5
Android 2.0.1	Eclair	6
Android 2.1	Eclair	7
Android 2.2	Froyo	8
Android 2.3,2.3.1-2.3.2	Gingerbread	9
Android 2.3.3-2.3.7	Gingerbread	10
Android 3.0	Honeycomb	11
Android 3.1	Honeycomb	12
Android 3.2	Honeycomb	13
Android 4.0, 4.0.1-4.0.2	Ice Cream Sandwich	14
Android 4.0.3-4.0.4	Ice Cream Sandwich	15
Android 4.1.x	Jelly Bean	16
Android 4.2.x	Jelly Bean	17
Android 4.3	Jelly Bean	18
Android 4.4	Kit Kat	19
Android 5.0	Lollipop	21
Android 5.1	Lollipop	22
Android 6.0	Marshmallow	23

Tabla 2-1: Versiones de Android

2.2.1.3 Arquitectura

La arquitectura de Android está formada por cuatro capas, todas ellas de software libre. Esta arquitectura se conoce como pila, ya que las capas superiores utilizan elementos de las inferiores para realizar sus funciones.

A continuación, se muestran gráficamente estas capas y se explican con más detalle.

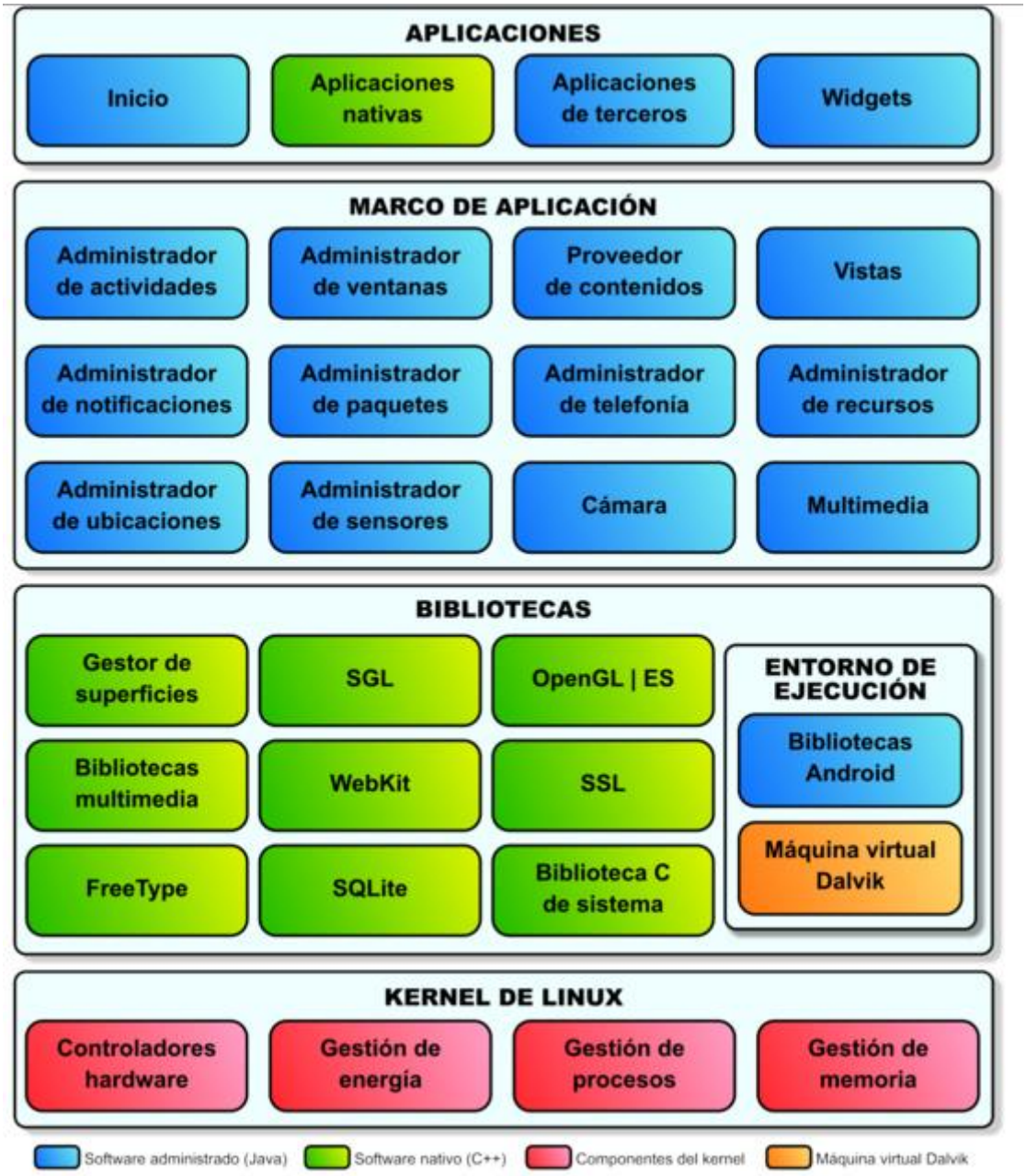


Figura 2-4: Arquitectura de Android

- **Kernel de Linux**: El núcleo de este sistema operativo es Linux versión 2.6. Permite que se pueda acceder a los elementos hardware a los que tienen que acceder las aplicaciones sin necesidad de conocer las características específicas de los elementos que estaban previamente instalados en el teléfono. Para cada elemento hardware existe un driver dentro del núcleo que permite utilizarlo desde el software. El núcleo se encarga también de gestionar recursos del teléfono y del sistema operativo en sí.

- **Bibliotecas**: Esta capa está compuesta por las bibliotecas nativas de Android. Estas bibliotecas están escritas en C o C++ y compiladas para la arquitectura hardware especifica del teléfono. Se encargan de proporcionar funcionalidad a las aplicaciones, para la realización de tareas que se repiten frecuentemente, evitando

su codificación cada vez que se realizan y garantizando que se llevan a cabo de maneras eficiente. Algunas de las bibliotecas que se incluyen habitualmente son: Gestor de superficies, SGL, Open GL ES, Bibliotecas multimedia, WebKit, SSL, FreeType, SQLite o Biblioteca C de sistema.

- **Entorno de ejecución**: Se apoya en las bibliotecas anteriores, pero no se considera una capa como tal, puesto que está formado por bibliotecas también. Éstas son las esenciales de Android, que incluyen la mayoría de funcionalidades de las bibliotecas de Java así como algunas específicas de Android. El componente principal del entorno de ejecución es la máquina virtual Dalvik, que ejecuta todas las aplicaciones no nativas de Android. Está maquina está basada en registros y no en pila para almacenar los datos, lo que provoca que haya menos instrucciones y las ejecuciones sean más rápidas. Los ficheros ".class" de Java se transforman en ficheros ".dex" (Dalvik Executable) antes de ejecutarse en la máquina virtual. La ventaja de estos ficheros es que son más compactos y están preparados para una mayor optimización.

- **Marco de aplicación:** Esta capa está formada por las clases y servicios que utilizan las aplicaciones directamente para realizar sus funciones. Se apoyan en las bibliotecas y en el entorno de ejecución. La mayoría de los componentes de esta capa son bibliotecas Java que acceden a los recursos a través de la máquina virtual Dalvik.

- **Aplicaciones:** Esta capa está formada por las todas aplicaciones que se incluyen en el dispositivo: las que poseen interfaz de usuario, las que no poseen interfaz de usuario, las nativas, las administradas, las que vienen en el dispositivo y las instalas por el usuario. Aquí se encuentra la aplicación de Inicio, que permite ejecutar las otras aplicaciones. Se debe tener en cuenta que todas las aplicaciones utilizan el mismo marco de aplicación para acceder a los servicios que proporciona el sistema operativo, lo que implica que se pueden crear aplicaciones que usen los mismos recursos que usan las aplicaciones nativas y que se puede reemplazar cualquier aplicación del teléfono por otra que desee el usuario.

2.2.2 Otros sistemas operativos

2.2.2.1 iOS

iOS es un sistema operativo móvil desarrollado por Apple Inc. Se desarrolló para el iPhone aunque más adelante se ha usado para el iPad y el iPod. Es un sistema operativo cerrado por lo que Apple no permite que se modifiquen características internas del sistema ni que se instale en hardware de otros dispositivos.

Este sistema operativo está basado en el núcleo de OS X, que a su vez se basa en Darwin BSD, por lo que es un sistema operativo tipo Unix. Su arquitectura está formada por cuatro capas: Núcleo OS, Servicios Principales, Medios y Cocoa Touch.

2.2.2.2 Windows Phone

Es un sistema operativo desarrollado por Microsoft y basado en el núcleo Windows Embedded CE 6.0, que está enfocado en el mercado de consumo en lugar de en el mercado empresarial, como ocurría con Windows Mobile.

Es un sistema operativo cerrado que utiliza como lenguaje de programación C# o Visual Basic .NET para las aplicaciones y XAML para diseñar la interfaz de usuario.

.Entró con retraso frente a Android e iOS, por lo que la cuota de mercado es inferior a la de estos, pero sigue creciendo debido, entre otras cosas, a que ofrece una experiencia de usuario en toda su gama de terminales.

2.2.2.3 Firefox OS

Es un sistema operativo basado en HTML5 con núcleo Linux y código abierto. Fue desarrollado por Mozilla Corporation con apoyo de empresas como Telefónica.

Este sistema operativo ofrece aplicaciones web que pueden ser de servidor o empaquetadas. Las aplicaciones de servidor son páginas web con apariencia de aplicación, por lo que no se puede acceder a ellas si no se tiene conexión a internet. Las aplicaciones empaquetadas necesitan descargar un paquete comprimido.

2.2.2.4 BlackBerry OS

Sistema operativo desarrollado por RIM. Su uso es exclusivo en los dispositivos de marca BlackBerry.

Las aplicaciones se programan normalmente en C o C++, pero el código del SO no está abierto.

2.2.2.5 Ubuntu Touch

Sistema operativo basado en Linux y desarrollado por Canonical. Se presentó en 2013 y actualmente varias empresas están creando terminales para este sistema operativo.
Es de código abierto.

2.2.2.6 Tizen

Es un sistema operativo basado en Linux, patrocinado por Linux Foundation y la Fundación LiMo. Sus interfaces de desarrollo se basan en HTML5 y otros estándares web. Se usarán en tabletas, Smartphones, SmartTV…

En principio se presentó como un sistema operativo de código abierto, pero Tizen 2 funciona con un sistema de licencias no abiertas. El SDK completo se publicó bajo licencia de Samsung.

2.2.2.7 WebOS

Sistema operativo desarrollado por Palm Inc., basado en Linux con una interfaz gráfica basada en HTML5, JavaScript y CSS para evitar el aprendizaje de un nuevo lenguaje de programación por parte de los desarrolladores.

No cosechó éxito, por lo que la compañía fue absorbida por HP al principio haciendo este software libre y en la actualidad el SO pertenece a LG, que lo utiliza como plataforma para sus SmartTV.

2.3 Aplicaciones similares

Cuando se tiene intención de desarrollar una aplicación, es conveniente realizar un estudio en los mercados de aplicaciones de los diferentes sistemas operativos, para ver si existe alguna similar a la que se pretende desarrollar y si es así ofrecer algo distinto que mejore lo existente para poder competir con el mercado.

En este caso, la idea proyecto es, únicamente, poder ayudar a los alumnos de Universidad al aprendizaje de Contadores.

Aun así, se va a realizar la búsqueda de aplicaciones educativas en los mercados de los tres sistemas operativos con más cuota de mercado actualmente: Google Play (Android), Apple Store (iOS) y MarketPlace (Windows Phone).

En los tres mercados se encuentran aplicaciones educativas en las que se explican conceptos de electrónica digital. Sin embargo, la que más se parece a la de este proyecto es: **“I can design a counter”.**

Esta aplicación se ha encontrado en Google Play y está destinada a Estudiantes de Ingeniería Eléctrica, pero únicamente ofrece contadores binarios realizados con máquinas de estados.

2.3.1 Aplicaciones anteriores del DSLab

Existen cinco aplicaciones educativas realizadas como proyecto de fin de carrera por otros compañeros en el DSLab de la UAM. Estas aplicaciones están también destinadas a ayudar a los estudiantes de la asignatura DIE, pero centrándose en otros temas de la asignatura:

- ***Sequential Circuits:*** Ofrece ejercicios de máquinas de estado tipo Moore y Mealy.
- ***Combinational Circuits:*** Ofrece ejercicios de electrónica combinacional.

- ***KARN Map***: Ofrece herramientas para la simplificación de funciones lógicas a través de mapas de Karnaugh

- ***MOS Circuits:*** Ofrece ejercicios de circuitos MOS. Como novedad esta aplicación incluye un test.

- ***STUCK-AT:*** Ofrece ejercicios de detección sobre el modelo de fallos stuck- at. Esta aplicación incluye test.

3 Diseño

3.1 Propósito de la aplicación

El objetivo de este proyecto es desarrollar una aplicación móvil destinada a los alumnos de Dispositivos Integrados Especializados de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid. En ella se pretende facilitar a los estudiantes el estudio y análisis de los contadores.

La idea es mejorar el formato de las guías de problemas que se proporcionan en la asignatura aprovechando el enorme alcance que están teniendo tanto los dispositivos móviles como las tabletas en los alumnos. Las mejoras se verán reflejadas en los siguientes aspectos:

La aplicación ofrecerá una serie de catorce ejercicios de contadores, un tutorial sobre éstos que permita al usuario aprender o afianzar los conceptos básicos de contadores y, como novedad con respecto al set de aplicaciones de resolución de ejercicios para la asignatura DIE, un tutorial online que el profesor podrá modificar cuando lo desee provocando la actualización instantánea en cualquier dispositivo que esté utilizando la aplicación.

Los ejercicios podrán ser guardados, cargados posteriormente y compartidos a través del correo electrónico con cualquier persona. Además, la aplicación dispondrá de una "ayuda" que explica algunos aspectos de la aplicación al usuario.

3.2 Requisitos de la aplicación

En esta sección se explican las características y funcionalidades que deberá cumplir la aplicación. Puesto que ésta pertenece a un conjunto de aplicaciones de carácter educativo ya desarrolladas anteriormente, debe seguir estrictamente ciertas características referentes, sobretodo, a estilo y funcionamiento.

3.2.1 Interfaz grafica

Como se ha mencionado en la introducción a este punto de la memoria, esta aplicación pertenece a un grupo de aplicaciones educativas pertenecientes a los distintos temas de la asignatura Dispositivos Integrados Especializados (DIE) por lo que se seguirán unos patrones establecidos en las mismas tales como:

- Formato de botones
- Tamaño y tipo de letra
- Colores
- Tipo de transición entre actividades
- Formato de ayuda.

3.2.2 Idioma

El contenido de la aplicación se encontrará en inglés. El resto de aplicaciones del conjunto está en este mismo idioma dado que el inglés es el idioma internacional en electrónica y que además, la aplicación se ofrecerá a los alumnos a través de la plataforma Google Play que, es accesible desde cualquier lugar del mundo. Por esto, a pesar de que la aplicación está dirigida a los alumnos de Dispositivos Integrados Especializados de la EPS, se pretende acceder al máximo número de usuarios posible.

No se descarta que en futuras actualizaciones de la aplicación se añadan otros idiomas.

3.2.3 Versiones de Android

Uno de los temas principales que debe abordarse al desarrollar la aplicación es el número de versiones que va a soportar.

Cuando una función se crea en una versión de Android antigua, ésta será soportada por versiones nuevas. Sin embargo, si una función se crea en una versión de Android nueva, luego no servirá en versiones más antiguas.

Para que la aplicación pueda ser usada por el número máximo de usuarios posible, se realizará una estudio de cuáles son las versiones que más se utilizan en el mercado.

*Los datos y porcentajes que se muestran a continuación han sido obtenidos de los publicado por Android en:

http://developer.android.com/intl/es/about/dashboards/index.html

VERSION	NOMBRE	API	DISTRIBUCION
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.8%
4.0.3 - 4.0.4	Ice Cream	15	3.3%
4.1.x	Jelly Bean	16	11.0%
4.2.x		17	13.9%
4.3		18	4.1%
4.4	Kit Kat	19	37.8%
5.0	Lollipop	21	15.5%
5.1		22	10.1%
6.0	Marshmallow	23	0.3%

Tabla 3-1: Versiones de Android en dispositivos a Noviembre de 2015

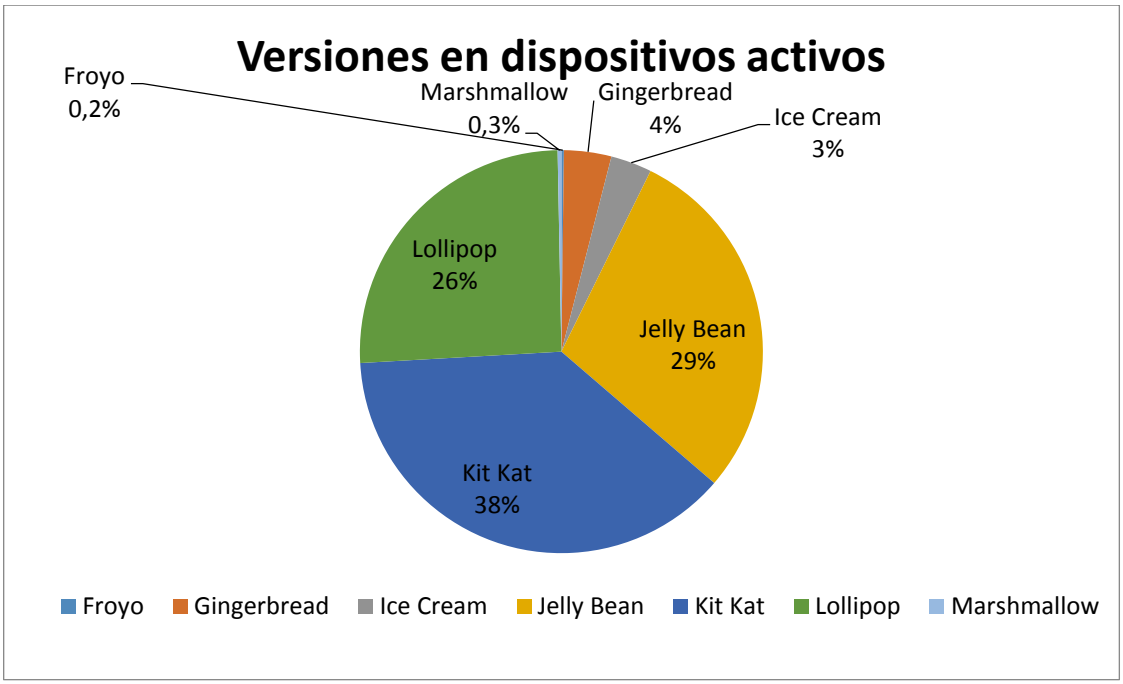


Figura 3-1: Versiones de Android en dispositivos a Noviembre de 2015

Al diseñar la aplicación se pretende que abarque la totalidad de versiones de Android activas actualmente.

Sin embargo, debido a que en el test online que se va a desarrollar se utilizan librerías que pertenecen a versiones de Android a partir de la 4.0.3, se estipula que ésta será la versión mínima que soportará la aplicación.

Como se observa en la tabla 3-1 y en la posterior gráfica, el porcentaje de dispositivos que utilizan versiones anteriores a la estipulada como mínima es alrededor de un 4%, que no se considera un porcentaje muy significativo, por lo que igualmente podrá utilizarla un gran número de usuarios.

3.2.4 Tamaño de las pantallas y densidad

Existen multitud de dispositivos con diferentes configuraciones de pantalla. La configuración de pantalla se define por la combinación del tamaño de la misma y su densidad.

Se entiende por tamaño de la pantalla la medida real de ésta, es decir, la medida de su diagonal en pulgadas. Se distinguen cuatro tipos de pantalla: small, normal, large y xlarge.

Cuando se habla de la densidad se describe la cantidad de pixeles dentro de un área física de la pantalla medida en unidades dpi (puntos por pulgada). Cuanto mayor sea la densidad mayor resolución tendrá la pantalla. Se distinguen seis tipos de densidades de pantalla: low (ldpi), medium (mdpi), high (hdpi), extra-high (xhdpi), extra-extra-high (xxhdpi) y extra-extra-extra-high (xxxhdpi)

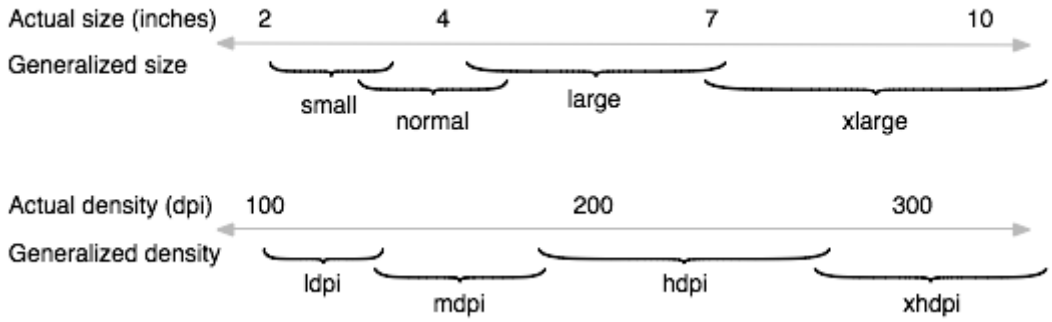


Figura 3-2: Rangos de tamaño y densidad de pantalla

En la siguiente tabla, con datos proporcionados por Android, se muestra la distribución de las diferentes configuraciones de pantalla a Noviembre de 2015:

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	3.1%						**3.1%**
Normal		6.1%	0.1%	41.3%	21.7%	15.3%	**84.5%**
Large	0.3%	4.7%	2.2%	0.6%	0.6%		**8.4%**
Xlarge		3.0%		0.3%	0.7%		**4.0%**
Total	3.4%	13.8%	2.3%	42.2%	23.0%	15.3%	

Tabla 3-2: Configuraciones de pantalla a Noviembre de 2015

Ahora, se mostrarán dos gráficos, uno con la distribución de los distintos tamaños de pantalla y otro con la distribución de densidades, ambos basados en la tabla anterior.

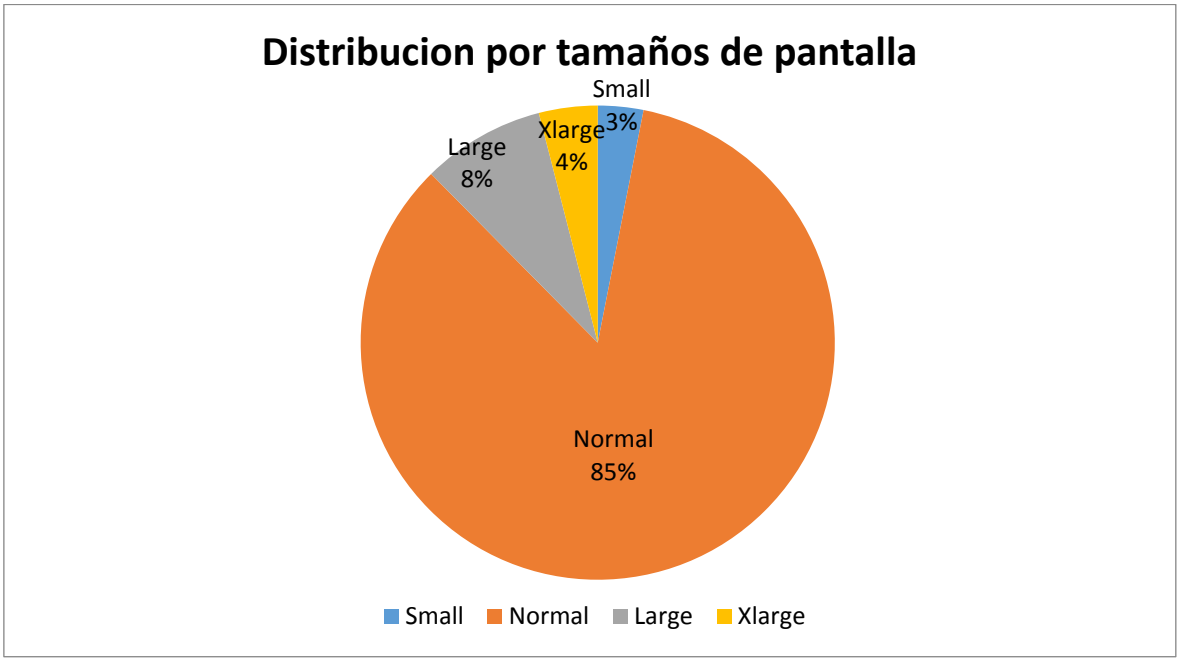


Figura 3-3: Distribución por tamaños de pantalla

Como se puede observar, predomina por mayoría la categoría “normal”, ya que a ella pertenecen los teléfonos convencionales.

La categoría “small” representa un porcentaje pequeño ya que cada vez las pantallas de los teléfonos son más grandes y éstas tienden a desaparecer.

De igual manera, la aplicación se ajustará a todos los tipos de pantallas para abarcar el mayor número de dispositivos posible.

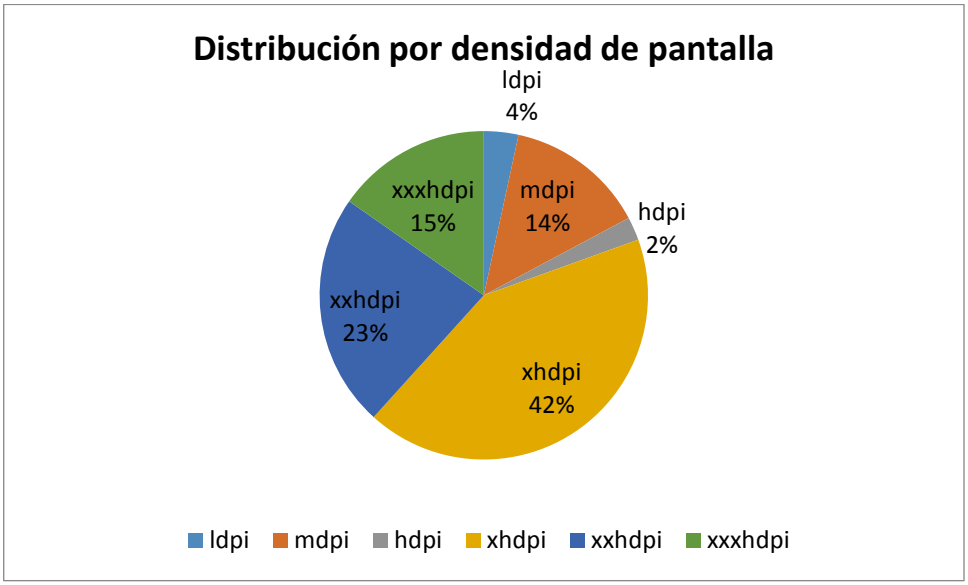


Figura 3-4: Distribución por densidad de pantalla

Se puede observar que predomina “hdpi”, pero hay un reparto más equitativo entre “mdpi”, “xhdpi”, “xxhdpi”. La utilización de pantallas de baja densidad actualmente ha decrecido mucho, mientras que la de pantallas de alta densidad cada vez irá aumentando más.

La densidad “tvpdi” no se tienen cuenta ya que se refiere a pantallas de televisión y para esta aplicación no aplica.

La aplicación se adaptará a las densidades de pantalla “ldpi”, “mdpi”, “hdpi”, “xhdpi” y “xxhdpi”.

3.2.5 Resumen de requisitos para la aplicación

A continuación se muestra una tabla que resume los requisitos de los que va a disponer la aplicación:

REQUISITOS PARA LA APLICACIÓN	
Sistema Operativo	Android
Versiones de Sistema Operativo	4.0.3-6.0
Tipos de dispositivos	Móviles y tabletas
Idioma	Inglés
Tipos de pantallas	Tamaño: small, normal, large, xlarge
	Densidad: ldpi, mdpi, hdpi, xhdpi, xxhdpi

Tabla 3-3: Resumen de requisitos de la aplicación

3.3 Limitaciones de la aplicación

Las limitaciones de la aplicación están referidas a los ejercicios, principalmente, y al tutorial. Dichas limitaciones son:

- En los ejercicios de ondas, por motivos de espacio en pantalla y para poder conservar una visión cómoda del tren de pulsos:
  - El número de pulsos no puede ser muy elevado.
  - El número de bits en cada estado no puede ser muy elevado.
  - No cambian los estados en flanco de subida y de bajada a la vez.
- En los ejercicios de conexiones, éstas no superarán las 140 debido a que los puntos de conexión estarían muy cerca y se complicaría el acierto para pulsar la conexión deseada.
- Las imágenes de la aplicación no superarán los 300 pixel/pulgada ya que el uso de imágenes con mayor densidad elevaría mucho el tamaño de la aplicación.
- A causa de permisos (explicado en el apartado 4.16), no pueden enviarse los ficheros que se almacenen en la memoria interna del dispositivo.

3.4 Módulos de la aplicación

En este apartado se van a resumir los módulos básicos de aplicación. Como la misma pertenece al grupo de aplicaciones de enseñanza de circuitos digitales del DSLab, se va a mantener el patrón estructural de las aplicaciones anteriormente desarrolladas.

3.4.1 Tutorial

La aplicación incluirá un tutorial con la teoría relativa a los contadores, que servirá de ayuda a los alumnos para la resolución de los ejercicios y del test.

Consiste en una sucesión de imágenes realizadas por el profesor Eduardo Boemo basándose en los apuntes de clase. Como en el resto de aplicaciones, las imágenes pasarán hacia delante o hacia atrás deslizando el dedo hacia la izquierda o hacia la derecha.
Cuando el usuario desee volver al menú principal solo tendrá que pulsar el botón atrás.

3.4.2 Ayuda

Se incluirá un manual de ayuda al usuario, en el que se explicará cómo rellenar cada estado en los circuitos de ondas, cómo completar una tabla, cómo realizar una conexión o cómo guardar, cargar o enviar los ejercicios.

Se podrá acceder al menú de ayuda desde cualquier pantalla de la aplicación pulsando sobre el botón de "menu" del dispositivo y seleccionando la opción "Help".

3.4.3 Ejercicios

La aplicación contará con catorce ejercicios de contadores que se dividen en tres grupos según su tipología gráfica:

- Ejercicios de ondas: Estos ejercicios se resolverán rellenando cada estado en el panel.

- Ejercicios de conexiones: Se resolverán realizando las conexiones correspondientes pulsando con el dedo la conexión deseada.

- Ejercicios de tablas: Este tipo de ejercicio se resolverá rellenando la tabla correspondiente.

Cuando se acceda a cada uno de los ejercicios aparecerá en pantalla el enunciado del mismo y, según el tipo de ejercicio que sea, aparecerá un esquema y debajo un tren de pulsos con el panel debajo para rellenar cada estado, un circuito para realizar las conexiones o una tabla.

Además, podrán realizarse las siguientes acciones:

- Resetear el ejercicio: Se vuelve al estado inicial del ejercicio.

- Corregir el ejercicio: Con este botón se indicará si la solución introducida es correcta.

- Guardar la solución actual del ejercicio en un fichero en memoria

- Cargar la solución guardada previamente o borrar dicha solución.

3.4.3.1 Reset

Al pulsar este botón aparecerá un mensaje preguntando si se está seguro de que se desea resetear el ejercicio. Una vez confirmado, se borrarán todos los datos introducidos por el usuario sin posibilidad de volver a recuperarlos.

3.4.3.2 Check

Pulsando este botón en cualquier momento se comprobará si la solución actual es correcta, con lo que aparecerá en pantalla el mensaje "Success!", o es incorrecta, apareciendo el mensaje "Fail".

3.4.3.3 Guardar

La solución actual de cada ejercicio podrá guardarse en cualquier momento pulsando el botón "Save". Aparecerá en pantalla una alerta donde deberá introducirse el nombre con el que se desee guardar el ejercicio precedido por una cabecera fija donde se indica el número de ejercicio que se va a guardar y si se guardará en memoria externa o interna.

En caso de que el dispositivo disponga de memoria externa, el archivo se guardará en una carpeta llamada "Counters" creada por la aplicación en la raíz del dispositivo. Si, por el contrario, el dispositivo no tuviera memoria externa el ejercicio será guardado en memoria interna, en el espacio reservado para los datos privados de la aplicación.

3.4.3.4 Cargar o borrar

Al pulsar este botón podrán cargarse las soluciones guardadas anteriormente. Aparecerá una lista de todas las soluciones que se han guardado del mismo ejercicio tanto en memoria externa como interna y, únicamente, podrá seleccionarse una de ellas.

Al cargar el ejercicio se sobrescribirá todo lo que hubiera antes de la carga.

Además, se proporcionará la opción de borrar las soluciones guardadas. Basta con seleccionar todas las soluciones que se deseen eliminar de la lista de soluciones guardadas y pulsar el botón "Delete". Las soluciones serán borradas sin opción de recuperarlas.

3.4.4 Test Online

Se ofrecerá un test en la aplicación que podrá ser actualizado en cualquier momento por el profesor. El test, que será un fichero de texto plano, se alojará en el mismo espacio web que la asignatura, y la aplicación lo descargará a través de una petición HTTP cada vez que se pulse en la opción "Test".

Será protegido mediante autenticación básica (usuario y contraseña) que conocerá solamente el profesor. Esto es debido a que para facilitar una corrección del test dentro de

la aplicación, el fichero debe incluir las respuestas a las preguntas, lo que añade un grado de seguridad a la finalidad educativa del test.

El formato del fichero que contiene el test será JSON, y éste, utilizando la programación adecuada, será interpretado y mostrado al usuario.

El fichero contendrá las preguntas, respuestas y soluciones del test, todo ello introducido por el profesor. Por lo tanto será necesario realizar y documentar un diseño estándar del fichero JSON para que el profesor pueda modificarlo cuando lo crea conveniente y la aplicación sepa interpretarlo.

La aplicación avisará al usuario que para hacer uso del test es necesaria una conexión a Internet, y se accederá siempre que éste acepte. En caso contrario nunca se accederá al test.

Se habilitará un botón de corrección, que indicará fallo ("Fail") o acierto ("Success!"), y un botón de reseteo completo del test con aviso previo.

3.4.5 Enviar ejercicios

Todas las soluciones guardadas de los ejercicios podrán enviarse por correo electrónico. Al pulsar el botón se mostrará un listado de todas las soluciones de todos los ejercicios.

Se podrán enviar uno o varios ejercicios a la vez. Una vez seleccionados los ejercicios que se desean enviar, se pulsará el botón "Send" y aparecerá en pantalla una alerta con los gestores de correo electrónico que se encuentran disponibles en el dispositivo.

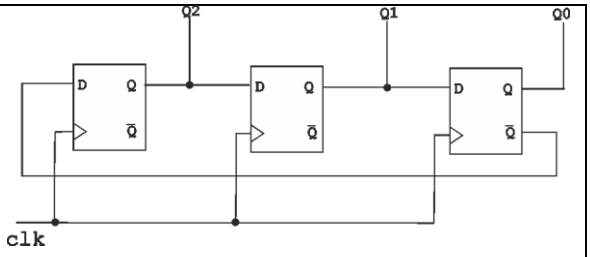
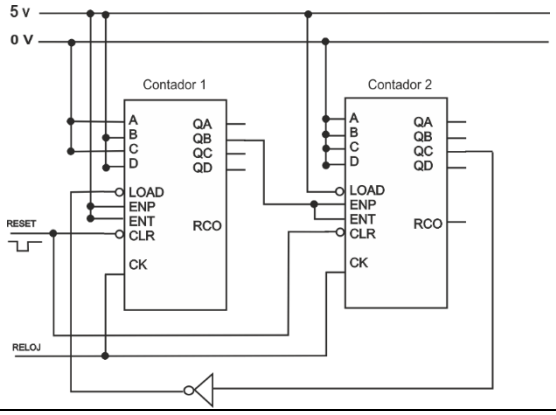
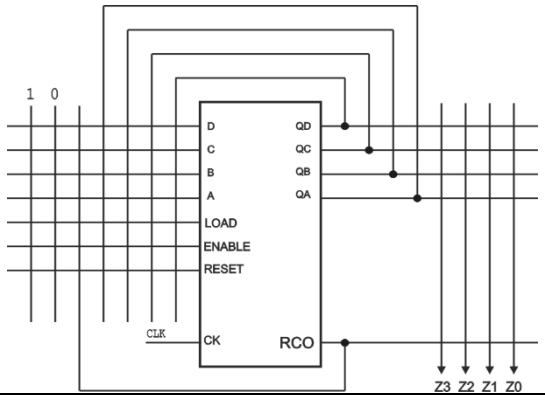
Cuando se elija el gestor de correo electrónico, se creará un mensaje de correo automáticamente con el campo de asunto relleno y las soluciones elegidas adjuntas. Además, en el cuerpo del mensaje habrá una explicación para la persona que lo reciba sobre donde se descargarán las soluciones adjuntas y qué hacer para poder cargar dichas soluciones. El usuario únicamente tendrá que introducir la dirección de correo del destinatario.

Esta funcionalidad es muy útil para compartir las soluciones de los ejercicios entre alumnos o con el profesor de forma rápida.

3.5 Aspectos docentes

La aplicación contendrá la siguiente lista de ejercicios, la cual corresponde a la mayoría de los contenidos que se pretende enseñar:

Ejercicio 1	Conocimientos sobre el contador Johnson
Ejercicio 2	Practicar el análisis (o ingeniería inversa) de contadores. Como ejemplo, el contador Johnson.

	
Ejercicio 3	Concepto de codificación one-hot
Ejercicio 4	Concepto de codificación zero-hot
Ejercicio 5	Concepto de codificación one-hot robusto
Ejercicio 6	Concepto de diseño síncrono. Descubrir que las señales de enable o la de load síncronas actúan en el ciclo siguiente. Además, entender el contador tipo 163. 
Ejercicio 7	Mismo concepto que en el ejercicio 6, pero mostrando que el contador se puede quedar parado en un valor final, del cual no se sale sin hacer un reset.
Ejercicios 8 y 9	Diseño de contadores especiales a partir de la estructura de un 163. Concepto de diseño síncrono. 
Ejercicio 10	Concepto de divisor de frecuencia programable. Concepto de diseño síncrono.
Ejercicio 11	Entender que un divisor programable basado en el 163 tiene un transitorio.

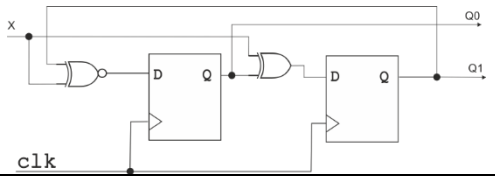
Ejercicio 12	Concepto de codificación one-hot con un 163.
Ejercicio 13	Concepto de eliminación de bits LSBs para conseguir contadores con repetición.
Ejercicio 14	Concepto de contador aleatorio. 

Tabla 3-4: Listado de ejercicios

4 Desarrollo

4.1 Introducción

En este apartado se va a detallar cómo están desarrolladas la mayoría de las partes de la aplicación. Algunas de las partes se detallan más a fondo, con referencias al código, debido a que han sido partidas de cero con respecto al resto del set de aplicaciones de la asignatura. Las partes que se han basado en módulos estas aplicaciones serán explicadas con menor detalle pero comentando las modificaciones hechas.

4.2 Herramientas utilizadas

Para la realización de este proyecto ha sido necesario:

- Ordenador personal
- SDK de Android (Eclipse + Plugin ADT + APIs Android)
- Java JDK
- Dispositivo móvil con Android (no obligatorio debido a la posible creación de dispositivos móviles virtuales con el SDK de Android)

4.3 Estructura de carpetas de la aplicación

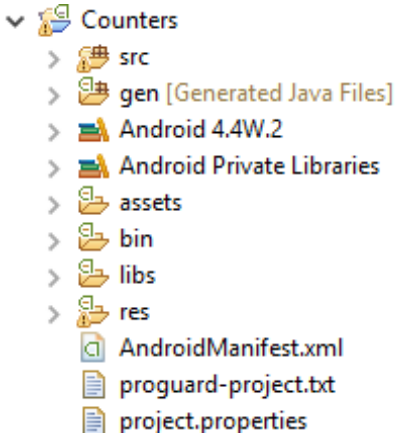


Figura 4-1: Estructura de carpetas proyecto Android

- **src**: en esta carpeta se almacenan los ficheros ".java" de la aplicación

- **gen**: código generado automáticamente por Android. Genera el fichero R.java que contiene los identificadores de cada recurso de la aplicación, para así poder ser accedido desde Java.

- **Android Library**: la API de Android según la versión indicada

- **Android Dependencies**: librerías adicionales de Android. En esta aplicación se ha añadido el fichero android-support-v4.jar, el cual permite añadir clases como ViewPager, Fragments o Navigation Drawable.

- **assets**: carpeta para introducir ficheros accesibles desde la aplicación. Estos ficheros no tienen asociado un identificador y no pueden ser modificados. En esta aplicación se incluirá un fichero cifrado con un usuario y contraseña a utilizar.

- **bin**: código compilado con el fichero ".apk" comprimido, listo para instalarse.

- **libs**: carpeta para añadir librerías adicionales personales. En esta aplicación está vacía.

- **res**: en esta carpeta se almacenan los recursos utilizados por la aplicación como son imágenes y ficheros XML que definen las pantallas.

- **AndroidManifest**: fichero XML en el cual se define la aplicación entera. Se definen parámetros como las actividades que habrá, los permisos que debe tener la aplicación, las versiones que soportará, el icono o la versión.

- **project.properties**: fichero creado por el SDK de Android. Contiene la versión de la API con la que se ha desarrollado la aplicación, además de otras características. En el caso de esta aplicación es la versión 20.

- **proguard-project**: fichero de configuración de la herramienta ProGuard, que te permite optimizar y ofuscar el código generado. Es decir, se obtiene un ".apk" más pequeño y donde resulta más difícil hacer ingeniería inversa. Es creado automáticamente por el SDK, pero no se usa en esta aplicación

4.4 Gestión de los datos con SharedPreferences

La aplicación necesita almacenar datos para que las actividades los compartan o puedan consultarlos en cualquier momento. Son datos de diferentes tipos:

- **Estado de los ejercicios**: es necesario que cuando un usuario quiera salir de un ejercicio, luego pueda volver a retomarlo y partir del último resultado.

- **Variables entre actividades**: son variables que se utilizan para la comunicación entre actividades.

- **Configuración**: mensajes de la aplicación que no se quieran mostrar más.

El método que se ha utilizado para estas funciones es la utilización de la clase SharedPreferences.

Esta clase almacena los datos en un fichero XML de la memoria interna del teléfono en forma etiqueta-valor.

Para hacer referencia a estos valores es necesario instanciar un objeto de esta clase que apunte al fichero almacenado y obtener el valor de cierta etiqueta.

Aquí se muestra un ejemplo de almacenamiento de datos:

```
SharedPreferences preferencias = getSharedPreferences(“Nombre_de_fichero”), MODE_PRIVATE);
SharedPreferences.Editor editor = preferencias.edit();
editor.putBoolean("etiqueta1", true);
```

Y un ejemplo de carga de datos:

```
SharedPreferences preferencias = getSharedPreferences(“Nombre_de_fichero”), MODE_PRIVATE);
preferencias.getBoolean("etiqueta1", false)
```

El fichero se genera en MODE_PRIVATE, es decir, ninguna otra aplicación del dispositivo tiene acceso a este fichero. Esto añade un nivel de seguridad a los datos de la aplicación.

Hay multitud de métodos para obtener y almacenar diferentes tipos de datos como boolean, int, float…

El uso de este método para almacenamiento de datos es usual para los datos de tipo “ajustes” de cierto tipo de aplicación. En este caso, al ser una aplicación que no va almacenar una gran cantidad de datos se ha optado por usar este método, que a su vez sirve como forma de “comunicación” entre actividades, en vez de una base de datos.

4.5 Gestión del ciclo de vida de las actividades

El ciclo de vida de una actividad es el siguiente:

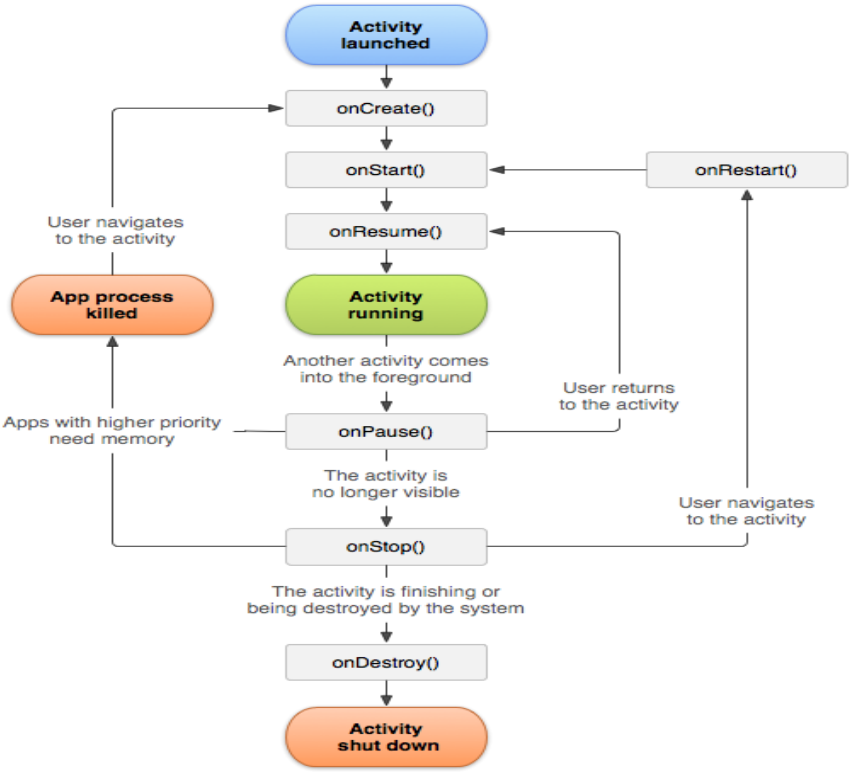


Figura 4-2: Ciclo de vida de una actividad

Este ciclo se resume a los siguientes estados:

- Running: la actividad está activa y es la que se muestra por pantalla.

- Paused: la actividad está activa y visible, pero en segundo plano, con alguna otra vista por delante como por ejemplo una alerta.

- Stopped: la actividad no es visible y está almacenada en una pila para una consulta posterior.

Android gestiona automáticamente el ciclo de vida de las actividades. Debido a la posible alta interacción del usuario con la aplicación (cambios entre ejercicios, salir o entrar, consulta del tutorial, consulta del test, de la ayuda…) no interesa que Android lo gestione todo, ya que pueden quedarse actividades en segundo plano que no interesa que estén.

Por lo tanto, siempre que se lance una actividad desde una primera, se finaliza la que primera. Solamente habrá una actividad siempre visible y ninguna en segundo plano.

Es por ello que se ha tenido que gestionar también el botón físico “back”, presente en todos los dispositivos móviles, para que lance la actividad correspondiente. Este botón se gestiona a través del método onKeyDown, perteneciente a la clase View.

4.6 Unificación de los textos

Para centralizar el control de los textos que aparecen en la aplicación se ha utilizado la herramienta que proporciona Android con el fichero "strings.xml" almacenado en "res/values".

En este fichero se definen variables con formato etiqueta-valor, donde el valor son los textos.

Tiene la siguiente forma:

<string name="etiqueta1">Texto</string>

Cualquier texto de la aplicación está parametrizado en este fichero, por lo que ante cualquier cambio solamente basta con plasmarlo ahí.

Para hacer referencia a estos textos desde cualquier parte del código se utiliza la clase R (dentro de los ficheros generados por Android). Cada uno de los textos tiene un identificador que es accesible de la siguiente forma con el método getString, que pertenece a la clase Context:

getString(R.string.etiqueta1)

4.7 Implementación del tutorial

La implementación del tutorial está basada en los trabajos realizados por la aplicaciones ya desarrolladas de la asignatura (Combinational Circuits, Sequential Circuits, MOS Circuits…)

Se resume brevemente su funcionamiento:

- Al comienzo se muestra un mensaje informando cómo se deben pasar las páginas del tutorial. Esto se ha desarrollado como cualquier diálogo de la aplicación. Además se facilita la opción de no volver a mostrar este mensaje nunca más. Esta opción se implementa mediante una variable almacenada en el fichero de preferencias leída a través de un Checkbox en el diálogo.

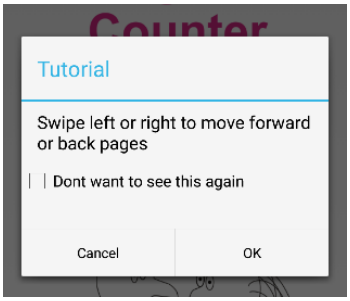


Figura 4-3: Alerta del tutorial

- En el layout XML del tutorial se ha creado un objeto del tipo **ViewPager**

- Desde la actividad Java se obtiene este objeto y se le configura un adaptador de tipo **PagerAdapter**. Este adaptador debe sobrescribir los métodos:

    - **instantiateItem**: método que recibe la View contenedora donde se va a incluir la ImageView a mostrar, es decir, una página del tutorial, y la posición en la que va mostrar la página. Se encarga de instanciar la página del tutorial.

    - **destroyItem**: método que elimina la ImageView de la View contenedora.

    - **getCount**: obtiene el número de imágenes del tutorial

    - **isViewFromObject**: ayuda en la identificación de las páginas del tutorial.

    Por lo tanto, el objeto ViewPager junto con el PagerAdapter automatiza la gestión de las páginas del tutorial, creando un efecto sencillo e intuitivo para el cambio de página.

    Aquí se muestra un ejemplo de la portada del tutorial y de cómo es el efecto de cambio de página.

Figura 4-4: Portada del tutorial

Figura 4-5: Efecto de pasado de página

4.8 Test Online

4.8.1 Comprobación de la conexión a Internet

Antes de acceder al test online, desde la actividad principal (MainActivity) se consulta el estado de la conexión a Internet.

Para ello se ha creado una función llamada isNetworkAvailable, que utiliza las clases ConnectivityManager y NetworkInfo proporcionadas por Android.

Cuando el botón del Test es pulsado, se llama a esta función y se prosigue de la siguiente manera:

- En caso de que no haya conexión a Internet, se comunica mediante una alerta al usuario. No es posible acceder al test.

- En caso de sí haya conexión a Internet, se avisa al usuario que se va a hacer uso de la conexión y se facilita la opción de cancelar en caso de que no quiera. A su vez se facilita un checkbox para guardar (a través del fichero de preferencias) que no se quiere mostrar este mensaje de nuevo.

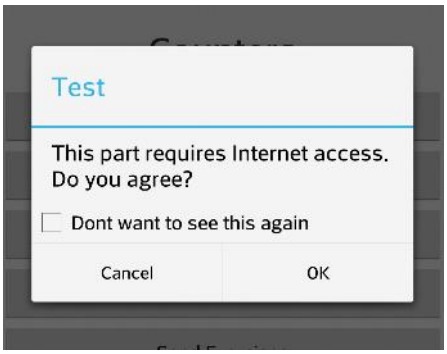


Figura 4-7: Aviso uso Internet

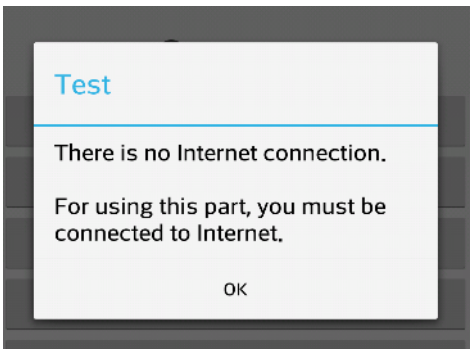


Figura 4-6: Aviso no hay conexión

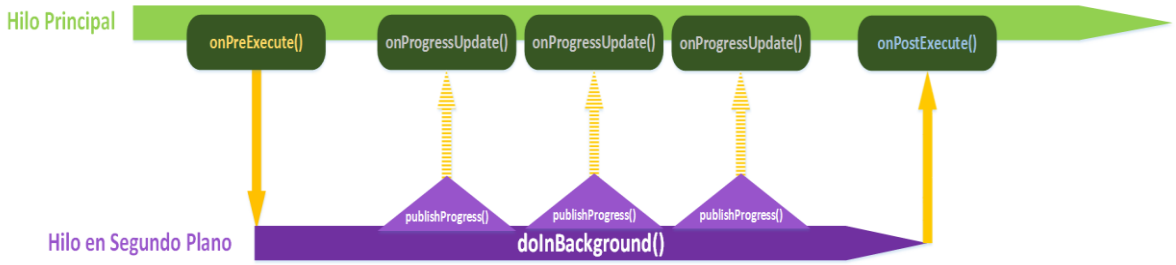
4.8.2 Funcionamiento de la actividad Test

La actividad Test basa su funcionamiento en la clase AsyncTask de Android. Esta clase permite programar en modo multitarea.

Cuando se ejecuta una aplicación Android, se crea solamente un hilo principal o también llamado “hilo de la interfaz gráfica”. Todas las actividades y servicios de la aplicación se ejecutan en este hilo principal.

En esta actividad se ha querido utilizar el modo multitarea para realizar una petición HTTP (junto con otras tareas que se detallan más adelante) a la vez que se muestra un mensaje en la pantalla indicando al usuario que se está procediendo a la descarga de un fichero. Estas dos tareas no podrían realizarse a la vez sin tener dos hilos ejecutándose a la vez.

La clase AsyncTask facilita y optimiza la programación multitarea, y solamente tendremos que sobrescribir algunos métodos propios de la clase. Aquí se muestra un diagrama con el ciclo de vida de la clase AsyncTask.



Como se puede observar, existe el hilo principal (hilo principal de la aplicación) y el hilo secundario creado por esta clase. Se resumen a continuación la función de cada método según qué hilo lo ejecute:

- Hilo principal:
    - **onPreExecute**: método invocado antes de la ejecución de la tarea en segundo plano. Suele utilizarse para configurar la tarea a ejecutar o para lanzar un mensaje con una barra de progreso.
    - **onProgressUpdate**: método invocado siempre después de una llamada a publishProgress por parte de la tarea en segundo plano. Suele utilizarse por ejemplo para mostrar de manera cuantitativa (barra de porcentaje) el estado de la tarea en segundo plano.
    - **onPostExecute**: método ejecutado una vez que finaliza la tarea en segundo plano. Se utiliza para recibir por parámetro el resultado de la tarea en segundo plano.
- Hilo secundario:
    - **doInBackground**: en este método se ejecuta el código principal de la tarea en segundo plano, como puede ser la descarga de imágenes.
    - **publishProgress**: este método puede ser invocado desde doInBackground y sirve para interactuar con el hilo principal a través del método onProgressUpdate.

Solamente es obligatorio sobrescribir el método doInBackground.

Trasladando este diseño a la actividad Test, se han sobrescrito los siguientes métodos:

- **onPreExecute**: se muestra un mensaje diálogo indicando mediante un círculo de progreso que el usuario debe esperar a que se descargue el test.

- **doInBackground**: se realiza la petición HTTP del test en segundo plano. Esto descargará un fichero que contendrá datos en formato JSON. En caso de descargarse con éxito, el JSON resultado es enviado a la función readJson. Si la lectura es correcta, se envía al método onPostExecute todo el test para que lo muestre por pantalla. La petición HTTP necesita una autenticación básica, y para ello se utiliza la función getCredentials.

    - **readJson**: analiza y extrae la respuesta a la petición HTTP, identificando las preguntas, las respuestas y las soluciones. Termina devolviendo al método doInBackground un solo string debidamente definido con todo el test.

    - **getCredentials**: obtiene el usuario y el password de acceso al test, que se encuentran encriptados dentro de la aplicación. Se explica con detalle más adelante.

- **onPostExecute**: recibe un string completo con todo el test. Este string es despedazado para separar preguntas y respuestas. Cada pregunta es mostrada en la pantalla utilizando la clase creada Question, que se detallará más adelante. Se dibujan y definen los dos botones de la actividad (Check y Reset), además de realizar el control de errores por si ha habido algún error en la definición del JSON o simplemente ha ocurrido un error en la comunicación HTTP.

  Aquí se muestra un ejemplo del test:

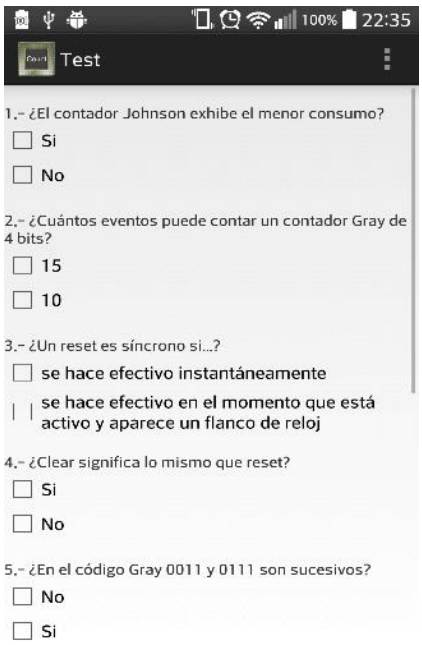


Figura 4-8: Ejemplo de test

4.8.3 Diseño estándar del fichero JSON

El fichero donde se encuentra el test, que es texto plano, contiene los datos en formato JSON (JavaScript Object Notation).

Un ejemplo de dato en formato JSON:

```
{
 “Campo1”: valor,
 “Campo2”: {“campo21”: valor, “campo22”: valor},
 “Campo3”: valor
}
```

Se ha definido un formato estándar para realizar el test, y lo deberá cumplir al completo para una correcta visualización en la pantalla. El formato es el siguiente:

```
[
 {
  “Pregunta”: “Pregunta1”,
  “Respuestas”: [“Respuesta1”, “Respuesta2”, “Respuesta3”],
  “Soluciones”: [boolean_Respuesta1, boolean_Respuesta2, boolean_Respuesta3]
 },
 {
  “Pregunta”: “Pregunta2”,
  “Respuestas”: [“Respuesta1”, “Respuesta2”],
  “Soluciones”: [boolean_Respuesta1, boolean_Respuesta2]
 }
]
```

- Los primeros corchetes indican que empieza el test, y que será un array de preguntas. Cada pregunta se separa por una coma.

- Las siguientes llaves (tipo de dato complejo) indican que comienza una pregunta, y este valor se compondrá de pregunta, respuestas y soluciones. Cada uno de los 3 campos es separado por comas también.

    - La pregunta será simplemente un texto.

    - La respuesta será un array con los textos de todas las que se van a mostrar.

    - La solución será un array del mismo tamaño que las respuestas pero con un valor booleano dependiendo de si su respuesta homóloga es correcta (True) o incorrecta (False). Esto permite la opción de varias respuestas válidas.

4.8.4 Creación de la clase Question

Para facilitar y reducir el código se ha diseñado una clase llamada Question. Se crea un objeto de esta clase cada vez que se extrae una pregunta del JSON recibido.

La clase simplemente define un LinearLayout con el texto de la pregunta y los textos de las respuestas. Junto con cada respuesta define un checkbox, el cual permite la elección de la/s respuesta/s.

Este LinearLayout es añadido al layout principal de la actividad Test ("activity_test.xml").

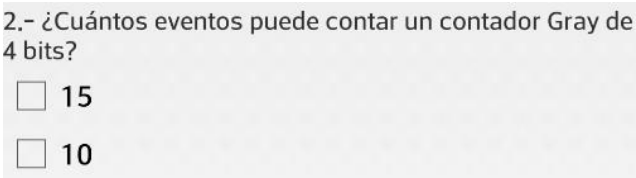


Figura 4-9: Objeto de tipo Question

La clase además define dos métodos:

- **getEstadoRespuestas**: para obtener qué respuestas ha seleccionado el usuario para esa pregunta.

- **resetEstadoRespuestas**: como su nombre indica, para resetear las respuestas seleccionadas por el usuario en esa pregunta.

Por lo tanto, cada vez que el método onPostExecute extrae una pregunta del objeto string con todo el test completo, crea un objeto Question.

4.8.5 Seguridad

El test consiste en un fichero de texto plano ".txt", que será alojado en el espacio web de la asignatura, es decir, en uno de los servidores de la UAM.

La URL del test no es pública, es decir, no hay ningún enlace publicado en ningún sitio, nada más que dentro del código de la aplicación.

Esto es debido a que como el fichero contiene las soluciones del test, no interesa que la URL sea pública y cualquier alumno pueda acceder a ella para visualizar las respuestas. No tendría sentido educativo que fuera así.

Por lo tanto, es necesario añadir seguridad y "trabas" a una posible visualización de las respuestas.

Esta seguridad se puede dividir en varios niveles:

- Seguridad del servidor de la UAM: la propia seguridad de una web dentro de un servidor de la UAM.

- Autenticación: usuario y contraseña una vez que se conoce la URL.

- Encriptación de la contraseña una vez que se tiene el código de la aplicación.

En el momento que alguien posea el código de la aplicación, aunque la contraseña esté encriptada en un fichero, es posible utilizar la función que la descifra. Existe otra contraseña que sirve para cifrar y descifrar la contraseña de la web, y se encuentra en texto plano dentro de la aplicación.

Por lo tanto, un posible agujero de seguridad del test es que el código de la aplicación sea accesible de alguna manera.

4.8.6 Protección con autenticación básica

Éstos han sido los pasos seguidos para proteger la web con usuario y contraseña:

- Lado del servidor:

  - **Fichero .htaccess**: En la carpeta del servidor donde se va a almacenar la web, tener un fichero llamado .htaccess con el siguiente contenido:

  - **Fichero .htpasswd**: En la misma carpeta o en otra cualquiera del servidor crear un fichero llamado .htpasswd, donde se va almacenar el usuario y la contraseña cifrada (configurando apropiadamente el fichero .htaccess para que apunte a la ruta de éste fichero).

    Para la creación de este fichero se ha utilizado la siguiente herramienta web: http://www.htaccesstools.com/htpasswd-generator/

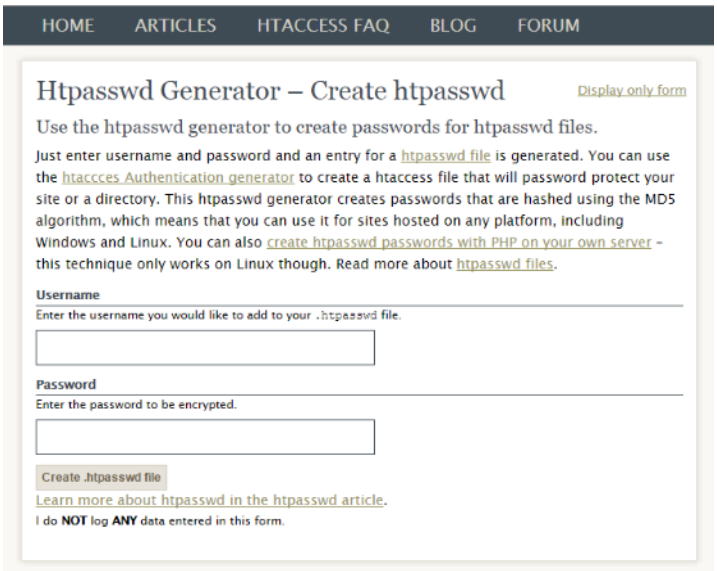


Figura 4-10: Herramienta para cifrar contraseña

Solamente introduces el usuario y la contraseña que necesites y pinchando en el botón “Create .htpasswd file” te crea el fichero.

Utiliza el algoritmo MD5 para escribir en el fichero el hash de la contraseña.

Una vez que estos pasos se han realizado, el acceso a la web queda protegido:

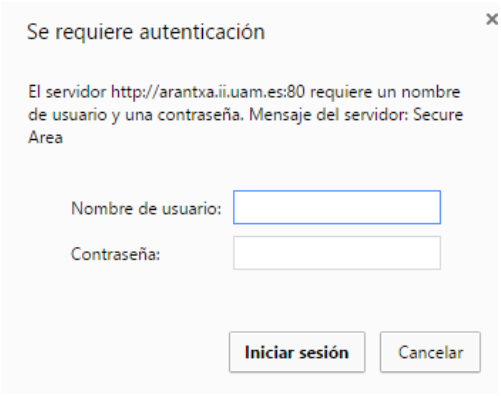


Figura 4-11: Protección de la web

- Lado de la aplicación:

Al hacer la petición HTTP desde la aplicación es necesario el usuario y la contraseña. Por motivos de seguridad no es conveniente tener la contraseña escrita en texto plano en el código, por lo que es necesario tenerla cifrada.

Para ello se ha almacenado en el directorio "assets" de la aplicación un fichero con el usuario y la contraseña cifrados (ambos).

El procedimiento para la generación del fichero cifrado es el siguiente:

- Se crea un fichero de texto plano con el formato:

  Usuario
  Contraseña

- Se ha desarrollado un pequeño programa Java (independiente de la aplicación Android) que obtiene este fichero de texto y lo cifra. Para ello captura los bytes de las cadenas de texto y los transforma aplicando el algoritmo AES.

Una vez que el fichero está cifrado dentro de la aplicación, es necesario implementar una función para descifrarlo. La función que realiza esta tarea se ha llamado getCredentials, y hace la operación inversa al cifrado explicado anteriormente, obteniendo el usuario y la contraseña en texto plano para poder utilizarlas en la autenticación de la petición HTTP.

Como se ha comentado anteriormente, es necesaria una contraseña para implementar el cifrado y el descifrado del fichero. Es imprescindible utilizar la misma contraseña en ambos procesos. Esta contraseña sí que va en texto plano en el código de la aplicación, y por lo tanto establece el límite de la seguridad a la obtención del código fuente de la aplicación, que no será publicado en ningún sitio.

4.8.1 Permisos necesarios en el dispositivo para la utilización del Test

Para hacer uso de internet desde el dispositivo y consultar el estado de la conexión es necesario añadir a la aplicación los siguientes permisos:

- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE

4.9 Implementación de las clases Ejercicio_X

En la aplicación hay tres tipos de ejercicios con contadores:

- Rellenar un panel con los estados de un determinado contador en función de un tren de impulsos.

- Implementación de puntos de conexión entre cables para la creación de un determinado contador

- Rellenar una tabla con los estados de un determinado contador.

Estos tipos de ejercicios son los que aparecen en la guía de problemas en formato papel.

Para la creación de estos ejercicios en la aplicación, se han desarrollado tres tipos de clases.

- Ejercicio_ondas

- Ejercicio_conexiones

- Ejercicio_tabla

4.9.1 Clase Ejercicio_ondas

Esta clase tiene el siguiente formato:

- **onCreate**: método ejecutado al inicio en el que se muestra el layout asociado al ejercicio (texto del enunciado e imagen), se crea y se inserta el/los objetos PulseView necesarios y se añaden desde el código los botones de reseteo, corrección, guardado y cargado.

- **checkIfCount**: método que indica si el panel de estados está vacío. Utilizado en la lógica del botón de reset.

- **guardarPreferencias**: método que almacena los datos necesarios del ejercicio en el fichero de preferencias. Es llamado siempre que se añada algo al panel de estados del contador o se salga del ejercicio. De esta forma los datos siempre se mantienen actualizados. Los datos que almacena son:

- Estados de los contadores (el texto).
- Coordenadas (x,y) para escribir el texto en las casillas correspondientes del panel de estados.

- **cargarPreferencias**: método que obtiene los datos del fichero de preferencias. Es llamado siempre que se entra al ejercicio para mostrar el estado en el que lo dejó el usuario al salir del mismo.

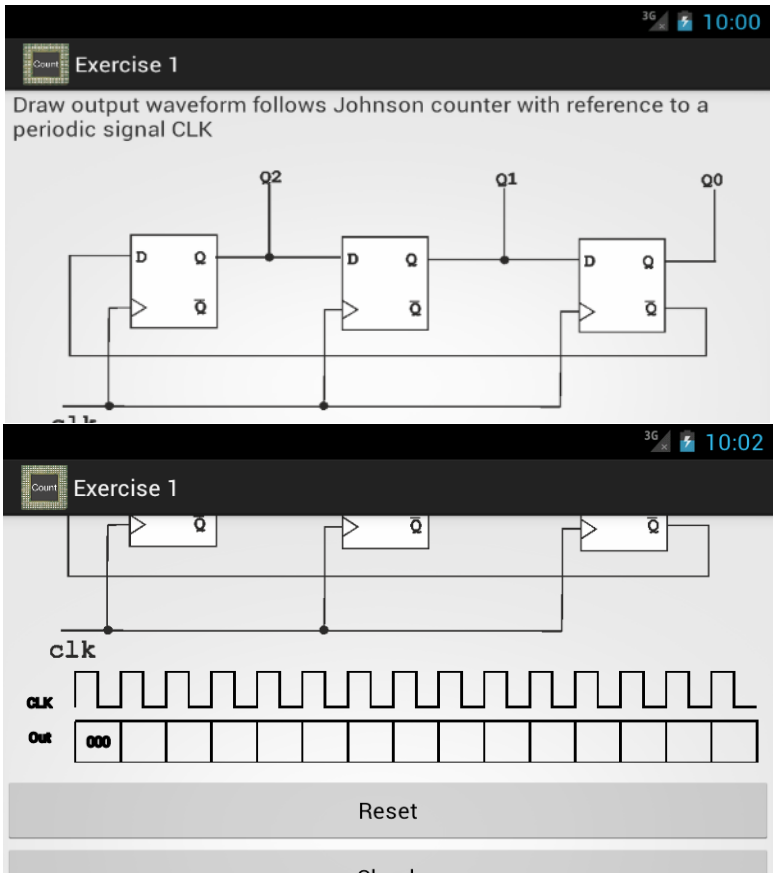


Figura 4-12: Ejercicio de ondas

4.9.2 Clase Ejercicio_conexiones

Esta clase tiene el siguiente formato:

- **onCreate**: método que se ejecuta al inicio y que muestra el layout asociado al ejercicio (texto del enunciado). Después se inserta la imagen del enunciado correspondiente, adaptada según el tamaño de pantalla utilizando la clase Bitmap de Android. También se insertan los botones y el código de reseteo, corrección, guardado y cargado de soluciones.

- **Dibujo**: clase de tipo View, similar a PulseView (clase creada en la aplicación). Sobrescribe los métodos:

  - **onDraw**: se encarga de dibujar los puntos de conexión entre cables.

- **onTouchEvent**: detectar la interacción del usuario con la pantalla cuando pulsa para establecer una conexión entre cables.

- **onMeasure**: para redimensionar el dibujo ya que el objeto contenedor es de tipo ScrollView y puede no caber en pantallas de algunos dispositivos pequeños.

- **checkIfConections**: método que comprueba si hay conexiones en el ejercicio. Utilizado en el botón de reset.

- **guardarPreferencias**: para almacenar los datos necesarios del ejercicio en el fichero de preferencias. El método es llamado siempre que se realiza una conexión o si se sale del ejercicio. De esta forma siempre los datos se mantienen actualizados. Los datos almacenados son:

    - Array de booleans que indica si hay conexión (true) o no la hay (false) en todos las conexiones posibles del ejercicio.

- **cargarPreferencias**: método que obtiene los datos del fichero de preferencias. Es llamado siempre que se entra al ejercicio para mostrar el estado en el que lo dejó el usuario al salir del mismo.

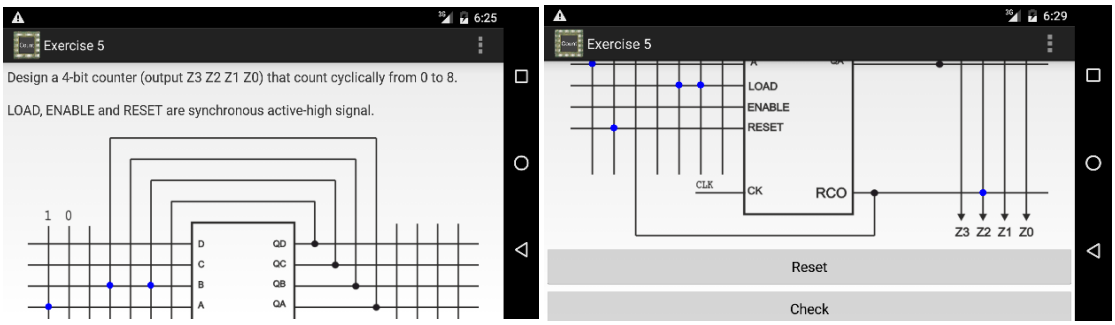


Figura 4-13: Ejercicio de conexiones

4.9.3 Clase Ejercicio_tabla

Esta clase tiene un formato algo diferente a los dos anteriores. Toda la parte gráfica está definida en el layout asociado.

- **onCreate**: método ejecutado al inicio. Muestra el layout asociado al ejercicio, es decir la tabla y los botones de reseteo, corrección, guardar y cargar.

- **guardar_TB**: método llamado siempre que se exista algún cambio en la tabla. Obtiene el estado de la tabla y llama a guardarPreferencias para reflejar los cambios.

- **save**: método que gestiona el guardado del ejercicio cada vez que se pulsa el botón de guardar.

- **load**: método que gestiona el cargado de una solución guardada cada vez que se pulsa el botón cargar.

- **resetear**: método que resetea todos los valores de la tabla al valor 0. Es ejecutado cada vez que se pulsa el botón reset.

- **check**: método que comprueba si la solución es correcta o no. Es ejecutado cada vez que se pulsa el botón de check.

- **guardarPreferencias:** método para guardar los datos del ejercicio en el fichero de preferencias. Guarda un dato de tipo boolean por cada bit de la tabla, true si es 1 y false si es 0. Es llamado siempre que hay algún cambio en la tabla por parte del usuario o siempre que se sale del ejercicio. De esta forma se mantiene actualizada la tabla si el usuario decide salir y después volver a entrar.

- **cargarPreferencias**: método para cargar los datos del ejercicio desde el fichero de preferencias. Es llamado desde el método onCreate cada vez que se entra al ejercicio.

La diferencia de esta clase con las dos anteriores es que los botones de opción (reseteo, corrección, guardar y cargar) han sido creados a través del fichero XML asociado. En las dos clases anteriores, estos botones han sido creados desde código java.

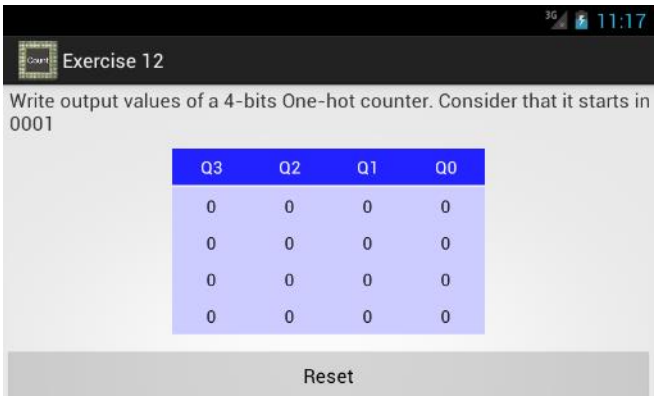
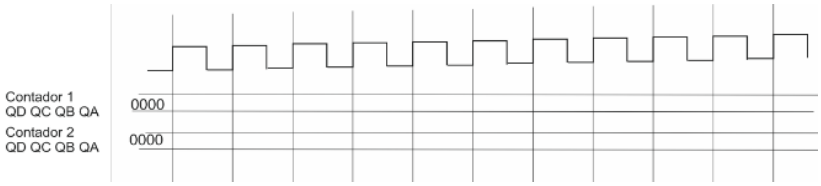


Figura 4-14: Ejercicio tabla

4.10 Creación de la clase PulseView

Esta clase permite crear un objeto en el que se visualiza un tren de impulsos y debajo un panel donde se va a anotar el estado de los contadores para cada pulso. De esta forma se simula este tipo de ejercicios de la guía de contadores.



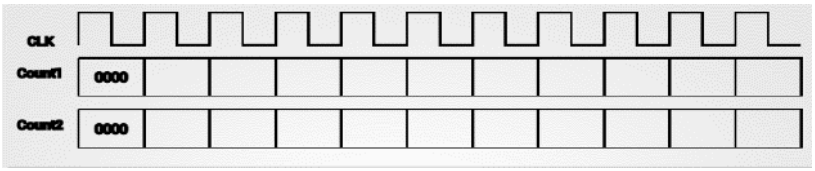


Figura 4-15: Comparación entre vista de papel (arriba) y objeto PulseView (abajo)

Tanto el tren de impulsos como el panel serán totalmente parametrizables dependiendo del tipo de ejercicio que se necesite crear, aunque vendrá con algunas limitaciones (explicadas en el apartado de limitaciones) debidas principalmente al tamaño de los dispositivos. Además, la clase se ha diseñado pensada para adaptarse automáticamente a cualquier tamaño y densidad de pantalla (explicado en el apartado de adaptación a todas las pantallas).

La clase funciona de la siguiente manera:

- PulseView hereda de la clase **View** de Android.
- Tiene un **constructor** en el cual se pueden elegir las siguientes opciones:
    - Número de pulsos a mostrar
    - Número de bits del contador
    - La coordenada superior de la pantalla (en píxeles) donde se quiere empezar a mostrar a mostrar el objeto
    - Si quieres dibujar el tren de impulsos o no, y su nombre escrito (CLK por ejemplo). Esto se ha implementado para los ejercicios en los que hay que mostrar a salidas de diferentes contadores referidas al mismo tren de impulsos.
    - El nombre de la señal de la que vas a escribir su estado en el panel
    - Si quieres que aparezcan siempre algunos estados fijos (que no se puedan modificar por ser parte del enunciado) y cuáles son.
    - Si quieres que el estado cambie en flanco de subida o en flanco de bajada (nunca ocurrirá a la vez).
- Método **onDraw**: este método pertenece a la clase View y se encarga de dibujar todo lo que diseñes en el objeto gráficamente (líneas, puntos, bitmaps, texto…). Es decir, es la función encargada de dibujar el tren de impulsos, el panel y los estados de los contadores que introduce el usuario. Esta función es ejecutada una vez al principio siempre que se implemente un objeto de tipo PulseView. Después es necesario llamarla cada vez que se quiera modificar algo del dibujo (a través de la función invalidate). Cada vez que el usuario añade o quita un estado del contador en el panel, onDraw es ejecutada para que actualice los cambios.

- Método **onTouchEvent**: este método pertenece a la clase View y responde cuando el usuario interactúa con la pantalla. En esta aplicación se utiliza para mostrar un listado de posibles valores de un estado del contador una vez que el usuario ha pulsado una casilla del panel. Una vez que se ha elegido el valor, se llama a la función onDraw para que actualice el cambio.

- Método **onMeasure**: este método pertenece a la clase View y se encarga de redimensionar el objeto siempre que el "objeto padre" (en este caso el layout donde se introduce el objeto PulseView) cambie de tamaño. Este método es útil en la aplicación ya que el "objeto padre" donde se introducen los objetos PulseView se declaran como ScrollView. Este tipo de vista se implementa cuando todo tu diseño gráfico no cabe en la pantalla de un teléfono y se necesita hacer "scroll" (vertical en este caso).

- Métodos **getResultados**, **getPosicionX**, **getPosicionY**, **reset** y **setResultados**: son métodos que interactúan con el objeto solicitando, reseteando o estableciendo los valores del panel.

Por lo tanto, se puede crear un objeto de tipo PulseView desde cualquier actividad que permite simular el mismo tipo de ejercicio que existe en la guía de problemas.

A diferencia de algunas aplicaciones de la asignatura creadas anteriormente, se conseguido separar el contenedor y el objeto que permite dibujar en la pantalla e interactuar con ella. En las aplicaciones anteriores todo estaba desarrollado dentro de la misma clase.

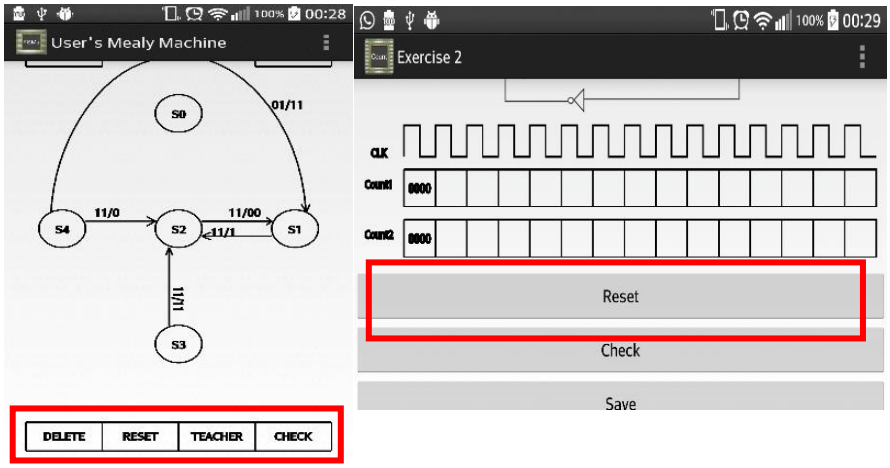


Figura 4-16: Comparación botones y dibujo con aplicación anterior

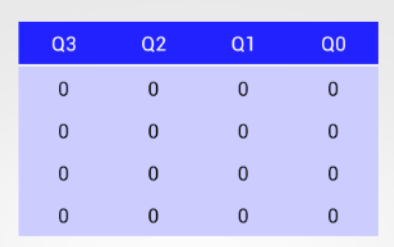
4.11 Implementación de tablas

El desarrollo de las tablas hereda la base desarrollada en las aplicaciones anteriores de la asignatura.

- Layout asociado

En el fichero XML se ha desarrollado al completo la interfaz gráfica de este tipo de ejercicio. Utilizando un contenedor de tipo ScrollView, se ha incluido una tabla de tipo TableLayout en la que cada fila es un objeto de tipo TableRow. Los estados de la tabla son botones de tipo ToggleButton que permiten cambiar solamente entre dos estados.

Todas son clases pertenecientes a la API de Android.



Q3	Q2	Q1	Q0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Figura 4-17: Tabla de ejercicio

4.12 Implementación de conexiones

La base del desarrollo de las conexiones entre cables también ha sido heredada de aplicaciones anteriores. Está implementada de la siguiente forma:

1- Diseño de la imagen

Todos los ejercicios de conexiones entre cables están basados en el mismo circuito.

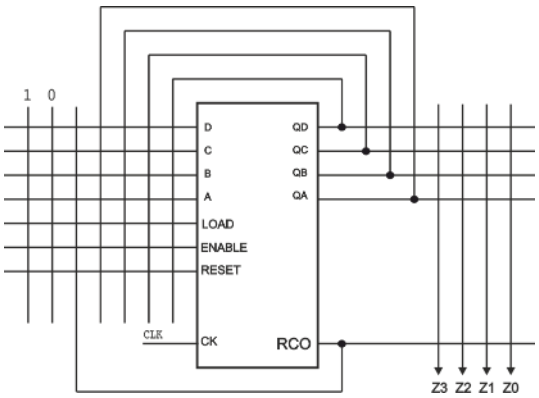


Figura 4-18: Circuito del ejercicio de conexiones

Se ha tenido que diseñar esta imagen con las siguientes características:

- Ancho: 22.01 cm
- Alto: 16,09 cm
- Separación entre conexiones: 1 cm
- Resolución: 300 ppi (píxeles por pulgada)

- Toda línea está en proporción a 1 cm de cualquier punto. Esto nos servirá a la hora de movernos por la imagen en el código.

Para exportar la imagen se ha tenido que reducir el tamaño proporcionalmente en ambas dimensiones debido a que ocupa un gran espacio en disco. Por lo tanto ha quedado:

- Ancho: 7 cm
- Alto: 5,11 cm
- Separación entre conexiones: 0,318037255792821444797819173103 13 cm
- Resolución: 300 ppi (píxeles por pulgada)

2- Inserción de la imagen en el código

Para insertar la imagen en el código se ha utilizado la clase Bitmaps de Android. Esta clase permite redimensionar una imagen almacenada en la aplicación en píxeles. Esto está contemplado porque se quiere controlar el tamaño de la imagen en píxeles en la pantalla. También se ha utilizado la clase DisplayMetrics de Android que permite detectar el ancho y el alto en píxeles del dispositivo que ejecuta la aplicación.

Por lo tanto, en píxeles:

- Ancho: 90% del ancho del dispositivo.
- Alto: la relación de imagen (alto_cm/ancho_cm) multiplicada por el ancho de la imagen en píxeles.

3- Creación de conexiones

Para trabajar con la imagen en el código es necesario trabajar con píxeles.

Como sabemos el ancho en centímetros del dispositivo que ejecuta la aplicación, y además sabemos el ancho en píxeles de la imagen (siempre mostraremos la imagen al 90% del ancho del dispositivo en píxeles) podemos lograr una relación con los datos anteriores en centímetros.

Por lo tanto:

$$relacion\_px\_cm = \frac{ancho\_imagen\_px}{ancho\_pantalla\_cm}$$

Lo único que se necesita saber es la separación entre conexiones en píxeles para así poder dibujar las mismas en el código. Como conocemos la separación en centímetros y la relación con los píxeles:

$$separacion\_conexiones\_px = separacion\_conexiones\_cm \times relacion\_px\_cm$$

Ya es posible moverse entre las conexiones desde el código.

Se definen tres áreas de conexiones correspondientes al dibujo. Y por lo tanto se definen tres vectores de tipo Boolean donde se almacenará si hay conexión (true) o si no la hay (false), y tres vectores de tipo entero donde se almacenarán las coordenadas (x, y) en píxeles de cada conexión.

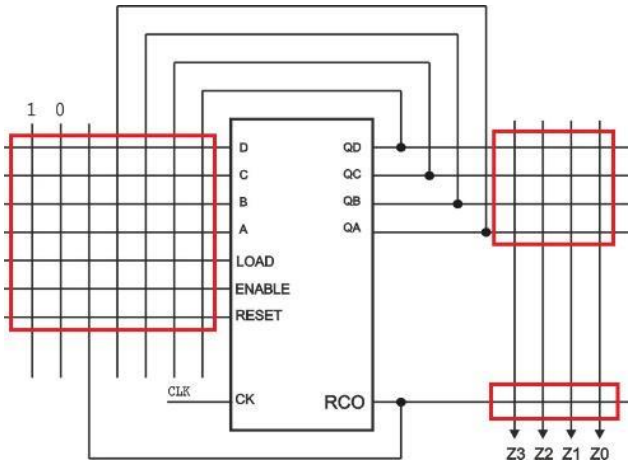


Figura 4-19: Las tres áreas de conexiones

La creación de este vector requiere saber un punto inicial en píxeles, que es donde comienza la imagen. Este punto se establece donde se requiera colocar la imagen.

Una vez que tenemos ese punto de referencia y sabiendo que toda distancia está proporcionada al valor "separación_conexiones_px" (se ha diseñado con respecto a esto), podemos almacenar las coordenadas en los vectores correspondientes.

4- Definición de áreas para la pulsación

Al ser muy difícil que el usuario acierte exactamente con el dedo en la coordenada de la conexión, se definen áreas cuadradas para cada conexión. De esta forma si se pulsa en cualquier punto de esa área se mostrará o borrará la conexión.

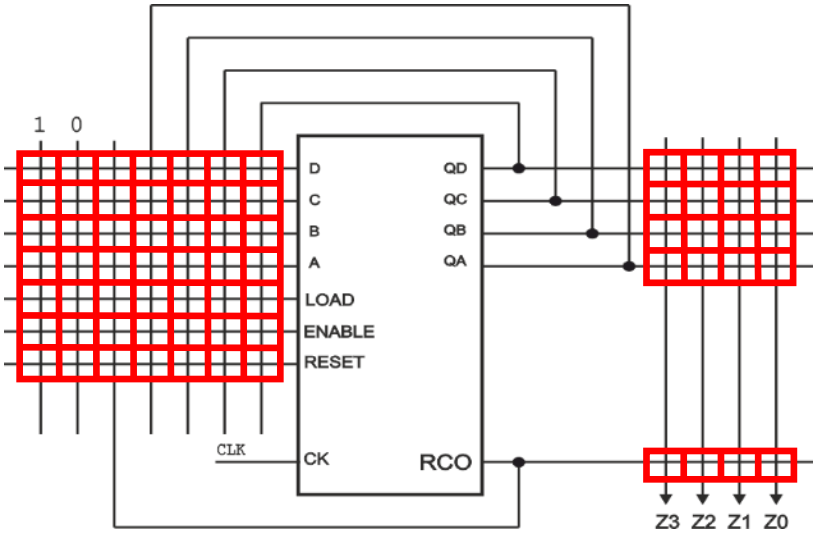


Figura 4-20: Áreas de margen para conexiones

Para resumir de forma entendible el funcionamiento, primero se inserta la imagen como bitmap y después se “coloca” por encima (que coincida exactamente) una malla de conexiones utilizando la clase Dibujo.

4.13 Gestión de las soluciones

Para la gestión de las soluciones se ha creado una clase sencilla llamada Soluciones. Esta clase ofrece un método público para que cada ejercicio (en su actividad correspondiente) pueda consultarlo.

Por lo tanto solamente es necesario crear un objeto de tipo Soluciones en cada ejercicio y llamar a su método getSolucionEjercicioX correspondiente pasándole por parámetro la solución implementada por el usuario.

Estos métodos comparan la solución del usuario con la correcta (presente en la clase Soluciones) y devuelven 1 o 0 dependiendo de si la solución es correcta o incorrecta respectivamente.

A diferencia de otras aplicaciones anteriores de la asignatura, se ha centralizado la gestión de las soluciones.

4.14 Guardar ejercicios

El método de guardar los ejercicios es común a todas las aplicaciones de la asignatura.

Consiste en una clase de tipo Activity a la que se llama cuando un usuario pulsa el botón de guardar el estado del ejercicio.

Dentro de la actividad, dependiendo de qué ejercicio la ha lanzado, se ejecuta un trozo de código que guarda los datos necesarios de ese tipo de ejercicio.

La clase detecta el ejercicio desde el que se ha llamado utilizando el fichero de preferencias, previamente asignando un identificador a cada ejercicio.

Los datos (que son los mismos que se almacenan en el fichero de preferencias) se almacenan en un fichero de texto plano. El nombre del fichero, que se establece por parte del usuario, contiene un prefijo establecido por la aplicación para poder identificar a qué ejercicio se referencia y en qué memoria se almacena.

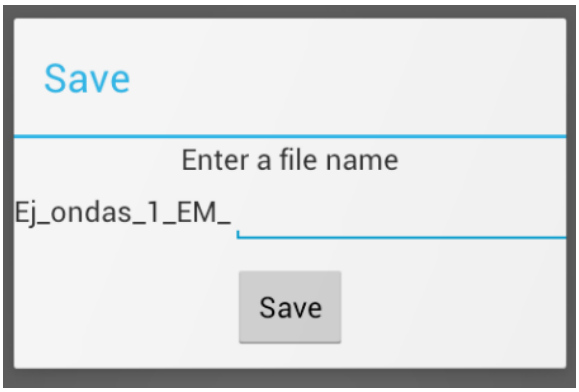


Figura 4-21: Guardar ejercicio

Los datos se almacenan de la siguiente forma para cada tipo de ejercicio:

- Ejercicio ondas (ejemplo)

  0010,125,250
  0011,150,250
  1000,175,250

  Donde cada línea corresponde con un estado del contador, el primer campo es el estado del contador, y los dos siguientes son las coordenadas x e y del panel.

  En el caso de que el ejercicio contenga varios paneles, se seguirán añadiendo líneas en el fichero.

- Ejercicio conexiones (ejemplo)

  True,False,True,False,False,False,True
  False,True,False,False,False
  False,True,False,False

  Donde cada línea es un área de conexiones (hay siempre tres). Cada valor indica si hay conexión (True) o no la hay (False).

- Ejercicio tablas (ejemplo)

  True, False, True, False, True, False

  Donde solamente hay una línea con todos los valores de la tabla seguidos y separados por comas.

La clase identifica, a través de la clase Environment de Android si el dispositivo contiene memoria externa o no.

La memoria externa puede estar presente en varias formas como es una tarjeta SD o una partición de la memoria interna (la mayoría de móviles actuales funcionan con esta forma).

En caso de que no exista o no sea accesible esa memoria externa, el ejercicio se almacenará en la memoria interna del dispositivo. Esta memoria interna es una partición dedicada a la aplicación e inaccesible por ninguna otra aplicación.

4.15 Cargar o borrar ejercicios

El método de cargar ejercicios es común a todas las aplicaciones de la asignatura.

Funciona de manera inversa al guardar ejercicios.

Se trata de una clase de tipo Activity que se lanza cuando el usuario pulsa el botón de cargar o borrar ejercicios.

La clase identifica (a través del fichero de preferencias) qué ejercicio la ha lanzado. Dependiendo de qué ejercicio la ha llamado (utilizando el prefijo comentado anteriormente) muestra por pantalla todos los ficheros almacenados de ese ejercicio.

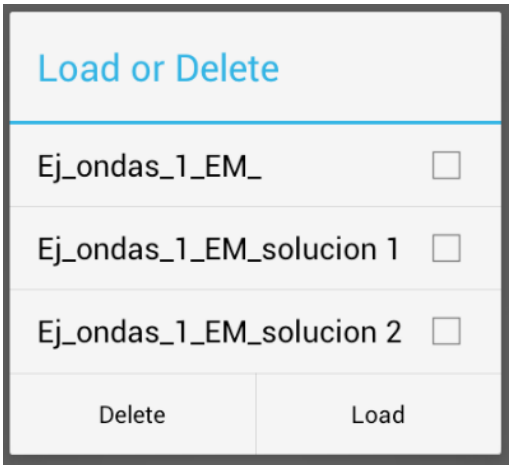


Figura 4-22: Cargar o borrar ejercicio

Utilizando también la identificación del fichero se ejecuta un trozo de código adaptado a cada tipo de ejercicio (ondas, conexiones, tablas) que lee ficheros de texto con el formato anteriormente mostrado.

Los valores leídos son almacenados en el fichero de preferencias. De esta manera, se vuelve al ejercicio, lanzándolo de nuevo y por lo tanto cargando los nuevos valores deseados.

Esta opción permite también el borrado de los ejercicios que se desee.

4.16 Enviar ejercicios

Otra de las funcionalidades comunes a todas las aplicaciones de la asignatura es la de enviar ejercicios.

Este código es ejecutado desde la clase MainActivity cuando se pulsa el botón de enviar.

Lee todos los ficheros almacenados en la memoria externa y los muestra por pantalla, permitiendo elegir todos los que se quieran enviar. Una vez que se acepta el envío de los ficheros se lanza una actividad propia de Android que crea un "chooser" identificando y mostrando todas las aplicaciones con las que se puede enviar los ficheros.

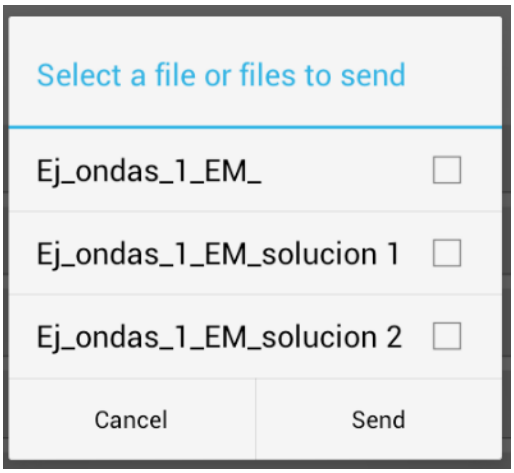


Figura 4-23: Enviar ejercicios

A partir de este punto, la aplicación que elija el usuario para enviar los ficheros es la encargada del envío de los ficheros.

Existe una limitación, y es que debido a que los ficheros de la memoria interna solo pueden ser accedidos por la aplicación Counters, cualquier otra aplicación que gestione el envío no tiene permisos de obtener estos ficheros. Es por esto que sólo se pueden enviar las soluciones almacenadas en memoria externa.

4.17 Implementación de la ayuda

La implementación de la ayuda consiste una clase (Ayuda1) que contiene un objeto de tipo ListView que muestra un listado de las opciones que se han considerado más importantes para facilitar una ayuda.
Cuando se pulsa uno de los elementos de la lista, se lanza otra actividad (Ayuda2) que obtiene el layout XML correspondiente con toda la ayuda necesaria.

Reseñar que la actividad Ayuda1 no es finalizada cuando se lanza Ayuda2, tal y como se ha ido realizando en toda la aplicación. Esto es debido a que desde la actividad Ayuda2 no es posible acceder a ningún otro sitio (salvo a la página About), sólo se puede volver hacia atrás, es decir, a a la Ayuda1. Por lo tanto, Ayuda1 puede mantenerse en la pila mientras se muestra Ayuda2.

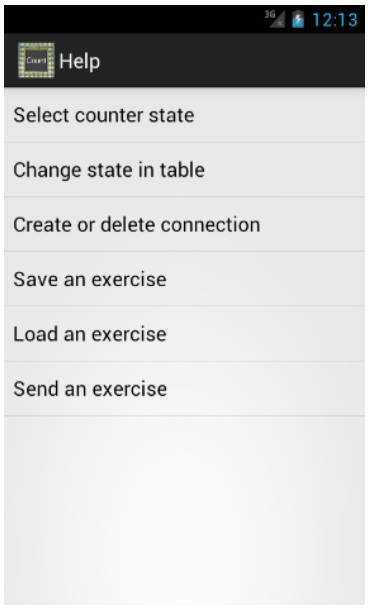


Figura 4-24: Menú de ayuda de la aplicación

4.18 Implementación del menú siempre disponible

Desde cualquier pantalla se ha habilitado que el usuario pueda pulsar el botón físico de menú siempre presente en los dispositivos y pueda acceder a la ayuda y la página de about.

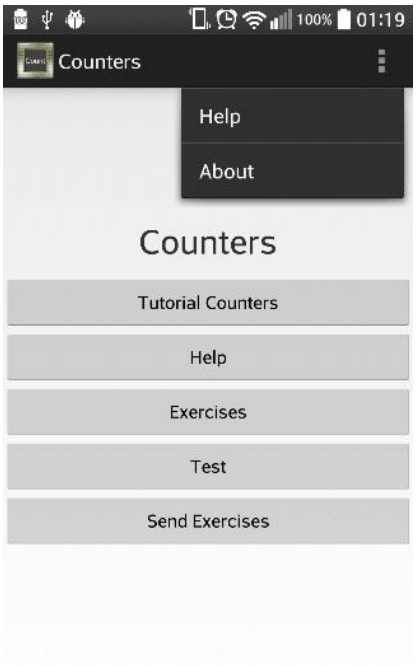


Figura 4-25: Menú siempre disponible

Android proporciona unos métodos para esto:

- **onCreateOptionsMenu**: método que obtiene el fichero XML de menú que está almacenado en la carpeta res/menú. Aquí se muestran las opciones ayuda o about.

- **onOptionsItemSelected**: método que ejecuta una acción dependiendo qué opción de las anteriores se ha elegido. Es decir, si se pulsa ayuda se lanza la actividad Ayuda1 y si se pulsa la opción about la actividad About.

4.19 Adaptación a todas las pantallas

Como se indica en el punto 3, se ha diseñado la aplicación que sirva en cualquier tipo de pantalla.

Hay una gran variedad de tipos de pantalla en los dispositivos que corren un sistema operativo Android. Esta variedad se corresponde con tamaño y densidad.

Android gestiona esta parte y facilita su diseño, pero algunas partes de la aplicación han tenido que ser adaptadas.

Por lo tanto, podemos dividir en dos las partes que ha habido que desarrollar: los layouts y la clase PulseView.

Para las pruebas se han utilizado las herramientas de simulación de dispositivos móviles que tiene el SDK de Android.

4.19.1 Adaptación de los layouts

Android facilita un método en el que dentro de la carpeta res se puede crear varias carpetas donde almacenar los layouts dependiendo del tamaño o la densidad de la pantalla.

Basta con crear una carpeta que se llame layout-TIPO, donde TIPO puede ser uno de los tipos de densidad o tamaño establecidos por Android. Por ejemplo layout-xlarge para pantallas con tamaño xlarge, o layout-xdpi para pantallas con densidad xdpi. Existen también multitud de combinaciones con otros sufijos que no entran en el contexto de esta aplicación.

En esta aplicación, la clasificación ha sido según el tamaño de pantalla. Se han creado las siguientes carpetas:

- layout
- layout-large
- layout-small
- layout-xlarge

En cada una de las carpetas existen todos los layouts de la aplicación, pero diseñados específicamente para esos tamaños de pantalla. El cambio principal que hay en cada una de las carpetas es el tamaño de la letra de los textos.

Todos los ejercicios se muestran en modo “portrait”. Esto se debe a que las imágenes son anchas y es necesario poder verlas lo más completas posibles. Para ello sólo basta modificar una propiedad de cada actividad, declarada en el fichero Manifest.xml, con la cual fuerzas a que la vista de esa actividad sea horizontal siempre.

4.19.2 Adaptación de la clase PulseView

El método de adaptación de dibujos con Canvas se basa en el desarrollo realizado en algunas aplicaciones anteriores de la asignatura.

Estas clases dibujan sobre un lienzo llamado Canvas, que no se puede incluir en ninguna carpeta. Es diseño total sobre una página en blanco y con valores en píxeles.

Pero no se puede dibujar las coas con valores constantes sino que es necesario que las medidas se autoajusten.

Para resolver este problema se han utilizado medidas de referencia, para después crear constantes que permitan adaptar las medidas a cualquier variabilidad. Siempre contando que Android permite la obtención del ancho y el alto del dispositivo donde se ejecuta la apliación.

El proceso es el siguiente:

- Se diseña el dibujo en un dispositivo de referencia, hasta que tenga una visión buena. Se anotan todas las medidas constantes en píxeles utilizadas.

- Se crean las constantes que van a permitir adaptar a cualquier pantalla cualquier medida anotada. Para medidas horizontales o verticales

$$cte_{horizontal} = \frac{medida_{referencia}}{ancho\_pantalla_{referencia}}$$

$$cte_{vertical} = \frac{medida_{referencia}}{alto\_pantalla_{referencia}}$$

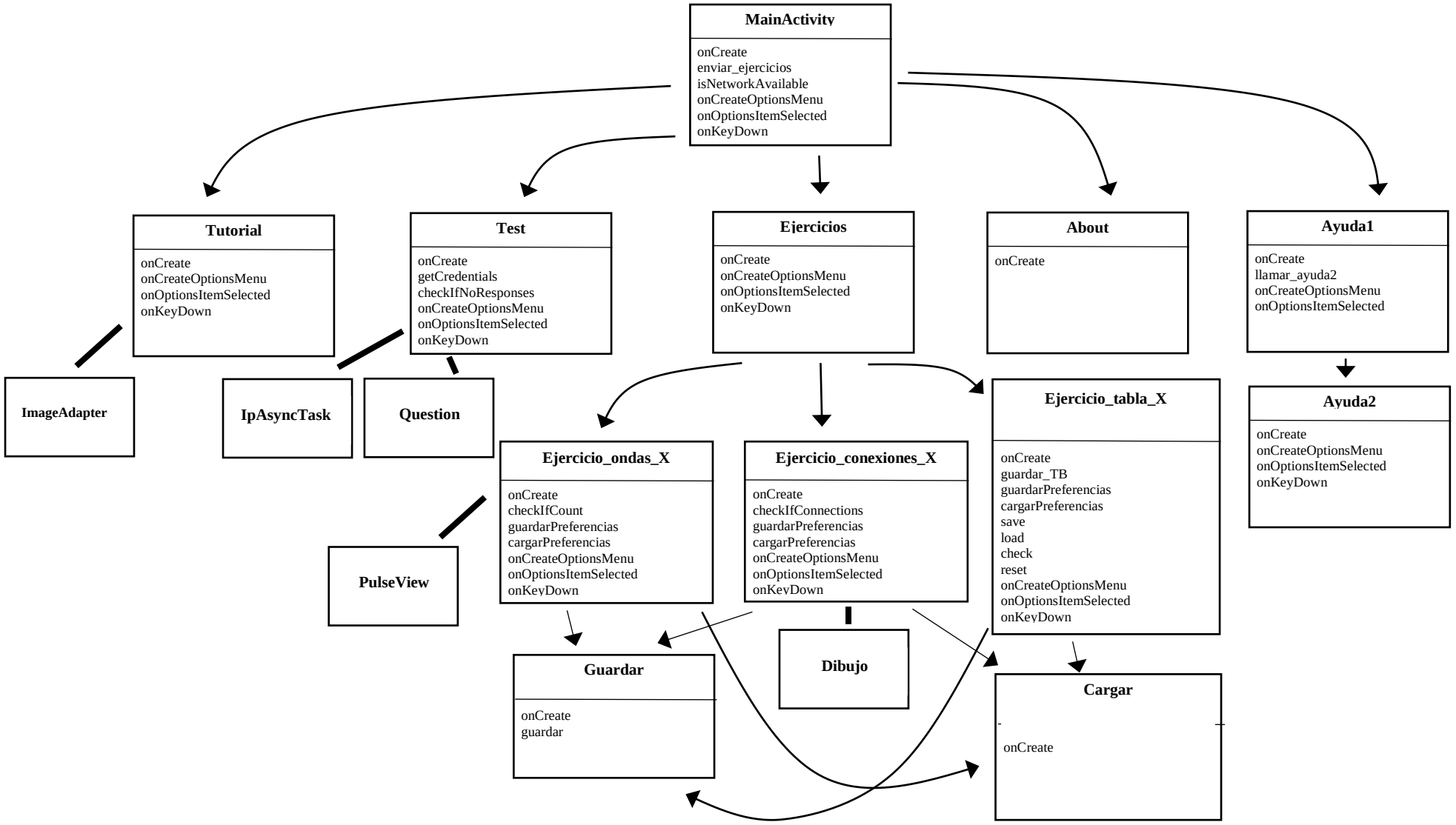
- Finalmente para cualquier medida que se quiera dibujar, basta con calcularla así:

$$medida\_horizontal_{dispositivo} = cte_{horiziontal} \times ancho\_pantalla_{dispositivo}$$

$$medida\_vertical_{dispositivo} = cte_{vertical} \times alto\_pantalla_{dispositivo}$$

Con esta solución, cualquier tipo de pantalla que ejecute la aplicación verá estos dibujos adaptados a la misma.

4.20 Diagrama de clases de la aplicación



4.21 Instalación en un dispositivo

En el fichero Manifest.xml existe una propiedad llamada “android:installLocation”, la cual está establecida al valor “auto”.

Esto significa que Android gestiona automáticamente el lugar de instalación. Valorando diferentes aspectos del dispositivo, decide si lo hace en la memoria interna o en la memoria externa.

Con esta propiedad además el usuario podrá mover la aplicación entre memoria interna y externa cuando lo desee.

5 Integración, pruebas y resultados

5.1 Publicación en Google Play

Para la publicación de la aplicación en Google Play se ha utilizado la cuenta ya registrada del laboratorio DSLab.

Estos han sido los pasos realizados para la publicación:

- Creación de fichero ".apk" utilizando la plataforma Eclipse. Este fichero ".apk" va firmado con una clave previamente creada.

- Acceder a la consola del desarrollador de Google Play con las credenciales de la cuenta del laboratorio y acceder al menú para una nueva publicación.

- Rellenar la ficha del Play Store.

  https://play.google.com/store/apps/details?id=com.Counters

- Subir el fichero ".apk" a producción en versión 1.0. Esta versión no contiene todavía el tutorial definitivo. Esta versión ocupa 4,7 MB en el dispositivo. En el momento que se actualice con el tutorial ocupará más.

- Test para clasificación del contenido de la aplicación. Preguntas sobre el contenido y uso de la aplicación.

- Publicar la aplicación. Acción que tarda unas 2 horas.

- La publicación de la primera versión se ha hecho efectiva el día 30 de Noviembre de 2015.

La aplicación, aun habiendo sido exhaustivamente comprobada, puede contener algunas erratas de texto o pequeños errores que corregir, por lo que no es probable que vaya sufriendo varias actualizaciones en un periodo corto de tiempo.

5.2 Pruebas en diferentes dispositivos

Para probar la aplicación en diferentes dispositivos se ha utilizado la herramienta de creación de dispositivos virtuales del SDK de Android.

Con esta herramienta puedes diseñar cualquier tipo de dispositivo que se te ocurra, modificando y combinando tipos de densidad y tamaño de pantalla, versión de Android, memoria, procesador...

Se muestran a continuación capturas de pantalla de la aplicación en dispositivos con diferentes modalidades de pantalla. Como se aprecia, la gestión de Android junto con las adaptaciones necesarias realizadas en alguna clase, se consigue que la aplicación tenga un aspecto muy similar en los diferentes dispositivos.
Para diferentes tamaños: small, normal, large y xlarge

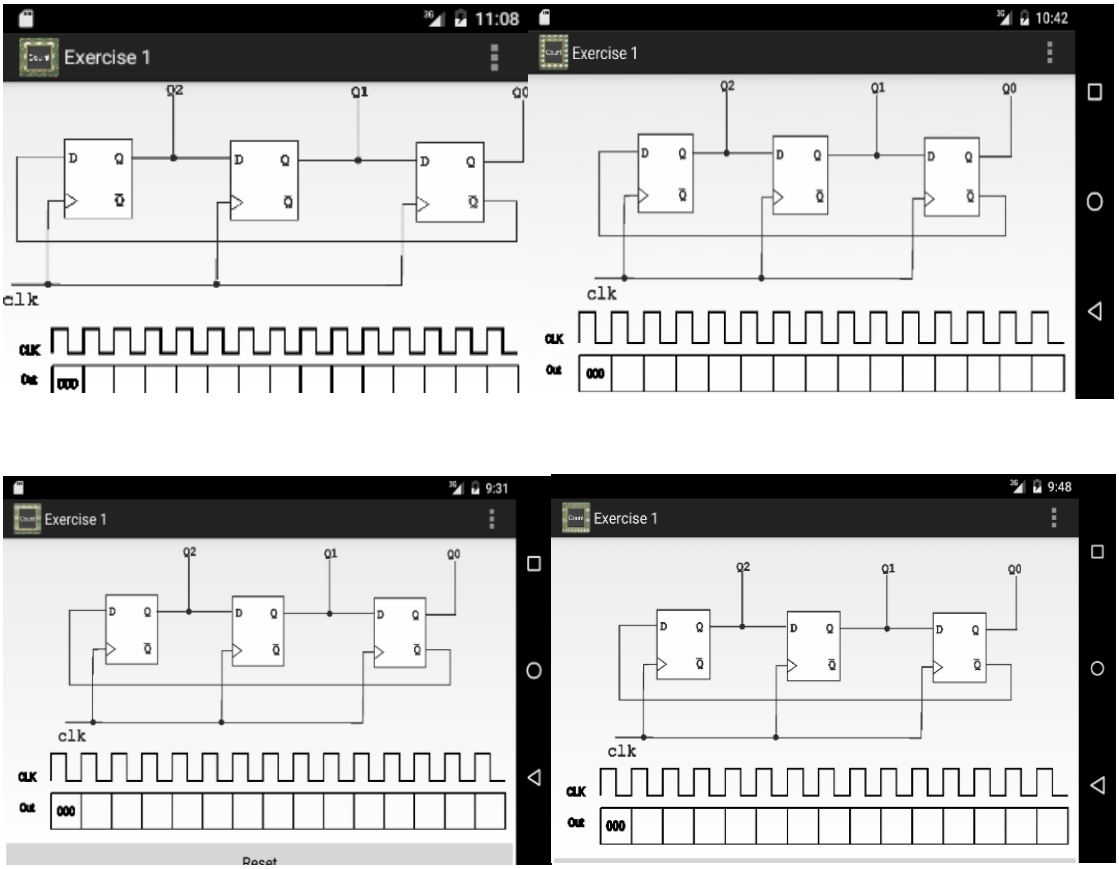


Figura 5-1: Tamaños de pantalla small, normal (arriba de izquierdada a derecha), large y xlarge (debajo de izquierda a derecha).

Y para las diferentes densidades: ldpi, mdpi, hdpi, xhdpi y xxhdpi

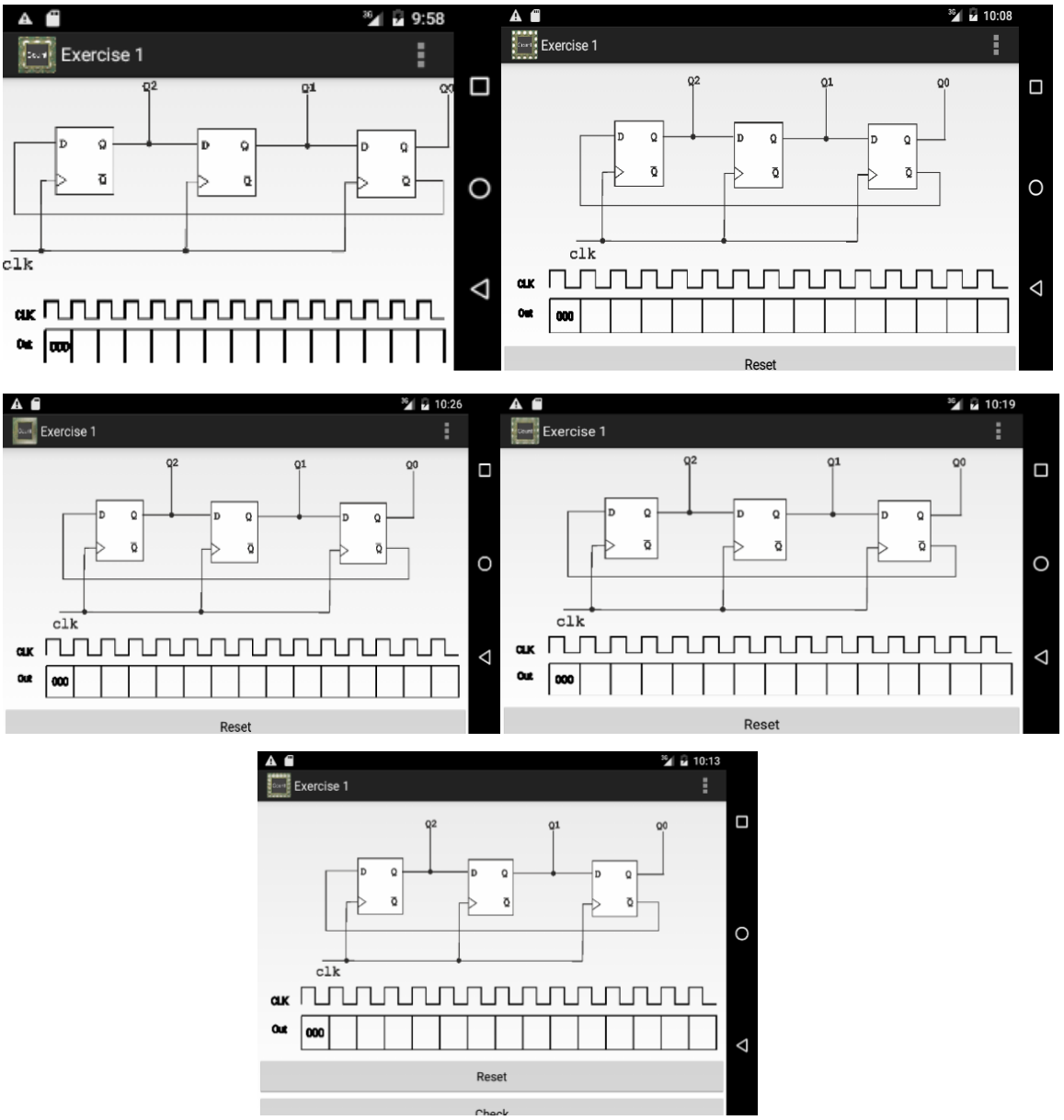


Figura 5-2: Densidades de pantalla ldpi, mdpi, hdpi, xhdpi y xhdpi en orden de arriba abajo y de izquierda a derecha

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Las conclusiones de este proyecto son:

- Se han cumplido los objetivos del proyecto, que era el desarrollo de una aplicación móvil para resolver problemas de circuitos secuenciales como son los contadores.

- Se ha realizado una aplicación que pretende reforzar o incluso sustituir a la guía de problemas en papel o PDF sobre contadores que se entrega en la asignatura Dispositivos Integrados Especializados de la Escuela Politécnica Superior de la UAM.

- Dicha aplicación replica tipos de ejercicios de la guía en papel y la forma en la que se resuelven, lo que refuerza la convicción de poder sustituir en un futuro próximo al papel.

- Con respecto a labores de programación:

    - Se han desarrollado clases como Question o Pulseview que pueden ser referenciadas en cualquier aplicación futura que las necesite.

    - Desarrollo de clase PulseView que simula un tipo de ejercicio de la guía de problemas de contadores.

    - Se ha desarrollado un test online el cual puede ser una base para futuros "controles" o "exámenes" de la asignatura.

    - El test online puede ser actualizado por el profesor sin necesidad de tocar la aplicación ni de contactar con el desarrollador.

    - Se ha conseguido mejorar el desarrollo de los ejercicios que contienen dibujos con Canvas y botones con respecto a aplicaciones anteriores.

    - Se han utilizado y mejorado utilidades de aplicaciones anteriores como son las de guardar, cargar o enviar ejercicios, el diseño gráfico, mensajes de error, o la adaptación a todas las pantallas, adaptándolas a los requerimientos de esta aplicación.

    - Se han utilizado herramientas de generación de ejercicios de tablas o conexiones de aplicaciones anteriores, adaptándolas a los requerimientos de esta aplicación.

    - Se ha utilizado la herramienta de generación de un tutorial utilizando el objeto ViewPager de aplicaciones anteriores.

    - Se ha mejorado el código consiguiendo un grado mayor de abstracción del código en caso de cambios futuros en la aplicación.

    - Se ha conseguido una aplicación estable y rápida, requisitos indispensables para el grupo de aplicaciones de la asignatura.

    - Se ha conseguido que la aplicación sea válida para la mayoría de los diferentes tipos de pantalla que existen en los dispositivos móviles con Android, lo que hace que llegue a un mayor número de usuarios.

- Con respecto a aspectos docentes, se ha aprendido:

    - Lenguaje de programación Java, no aprendido en la carrera.

    - Lenguaje de programación XML.

    - Arquitectura Android.

    - Entorno de desarrollo Eclipse.

    - Creación de página web que sólo contiene un objeto JSON.

    - Protección de página web con autenticación básica.

    - Publicación de la aplicación en el mercado Google Play.

    - Utilización de herramientas de diseño gráfico para la creación de los circuitos que aparecen en la aplicación.

- La aplicación ha sido puesta a disposición de los alumnos a través del mercado de aplicaciones de Google Play.

- Se continúa mejorando e innovando el aprendizaje de esta asignatura.

- La aplicación comenzará a ser utilizada por los alumnos a partir de Enero de 2016, cuando dé comienzo la asignatura.

- Se ha tardado un total de 720 horas para el desarrollo de la aplicación, repartidas entre las siguientes tareas:

    - Aprendizaje para desarrollo en Android y Java (Aprendizaje de forma autónoma y asistencia a curso básico de Android impartido por el laboratorio DSLAB): 120 horas

    - Desarrollo de la aplicación: 600 horas

6.2 Trabajo futuro

Existen varias mejoras para esta aplicación que pueden desarrollarse en un futuro:

Con respecto a la programación:

- Solucionar todas las limitaciones de espacio de la clase PulseView permitiendo un scroll horizontal para el panel de estados.
- Añadir un sistema de puntuación al test online.
- Mejorar el cifrado de las credenciales, ocultando la clave que se utiliza para cifrar.
- Herramienta web para el profesor con el fin de generar el test con mayor comodidad, sin tener que escribirlo a bajo nivel.
- Traducción al español
- Análisis y optimización de consumo de batería.
- Adaptar a Apple iOS
- Nuevas formas de compartir soluciones (redes sociales, servidor web…)
- Optimización general del código

Con respecto a la docencia:

- Añadir más tipos de ejercicios de contadores
- Nuevas formas de trasladar el diseño en papel al dispositivo móvil.
- Añadir más tutorial explicativos y detallados.

Con respecto a lo comercial de la aplicación:

- Añadir publicidad para generación de ingresos.
- Posicionamiento en redes sociales, para conseguir un mayor número de usuarios.
- Añadir más tipos de problemas en un apartado de pago.

Referencias

[1]. “El gran libro de Android”, Jesús Tomás Girones, 2011.
[2]. “Java 2 Curso de Programación”,
[3]. Fco Javier Ceballos, 4º edición, 2010.
[4]. “Android”, Ed Burnette.
[5]. “Desarrollo de aplicaciones para dispositivos móviles”, EPS-UAM.
[6]. http://developer.android.com/
[7]. http://jarroba.com
[8]. http://www.cristalab.com/tutoriales/proteger-carpetas-con-.htaccess-y-.htpasswd-c213l/
[9]. http://www.htaccesstools.com/htpasswd-generator/
[10]. http://stackoverflow.com/
[11]. http://blogthinkbig.com/sistemas-operativos-moviles/
[12]. https://columna80.wordpress.com/2011/02/17/arquitectura-de-android/
[13]. http://www.mundomanuales.com/telefonia/telefonos-moviles/que-es-android-caracteristicas-y-aplicaciones-4110.html
[14]. “Contadores”, Apuntes de clase de Circuitos Electrónicos Digitales, Escuela Politécnica Superior - UAM.
[15]. "Introducción al Diseño Lógico Digital", Hayes J., Addison-Wesley, 1996. ISBN: 0201625903.
[16]. “Fundamentos de Diseño Lógico”, Charles Roth Jr., Thomson 2005, ISBN: 9706863737
[17]. "Circutos Lógicos Programables", Tavernier, C. Editorial Paraninfo.
[18]. “Electrónica digital”, Tomás Pollán Santamaría, Zaragoza Prensas Universitarias de Zaragoza 2007. ISBN: 9788477339786.

Anexos

A. PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... ..... 700 €
- Alquiler de impresora láser durante 6 meses ..... 50 €
- Teléfono móvil ..... 300 €
- Material de oficina ..... 50 €
- Total de ejecución material ..... 1.100 €

2) Gastos generales

- 16 % sobre Ejecución Material ..... 176 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material ..... 66 €

4) Honorarios Proyecto

- 720 horas a 15 € / hora ..... 10800 €

5) Material fungible

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

6) Subtotal del presupuesto

- Subtotal Presupuesto ..... 12402 €

7) I.V.A. aplicable

- 21% Subtotal Presupuesto ..... 2604.42 €

8) Total presupuesto

- Total Presupuesto ..... 15006,42 €

Madrid, Diciembre de 2015

El Ingeniero Jefe de Proyecto

Fdo.: Cristina María Sobrino Hipólito
Ingeniero de Telecomunicación

B. PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de una APLICACIÓN ANDROID PARA RESOLVER PROBLEMAS DE CIRCUITOS DIGITALES SECUENCIALES. 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.