UNIVERSIDAD
REY JUAN CARLOS

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

Curso Académico 2012/2013

Proyecto de Fin de Carrera

Aplicación de aprendizaje Java para dispositivos móviles Android.

Autor: Juan José Ciudad Arévalo.

Tutores: Raquel Hijón Neira.

Índice

Resumen. 3
CAPÍTULO 1. Descripción y motivación. 5
1.1. Descripción del problema. 6
1.2. Motivación y objetivos. 6
CAPÍTULO 2. Estado del arte. 9
2.1. Descripción de las alternativas encontradas. 10
2.2. Carencias encontradas. 11
2.2.1. JavaTutor. 11
2.2.2. Skill Test. 11
2.2.3. Python programming. 12
2.2.4. TodoTest. 12
2.3. Objetivos. 13
CAPÍTULO 3. Metodología y tecnologías usadas. 15
3.1. Metodología. 16
3.2. Tecnologías. 18
3.2.1. Android. 18
3.2.2. Eclipse. 22
3.2.3. Java. 23
3.2.4. XML. 23
3.2.5. SAX. 23
3.2.6. Twitter4J. 24
3.2.7. Facebook SDK. 24
3.2.8. Draw9-patch. 24
3.2.9. SQLite3. 24
CAPÍTULO 4. Descripción informática. 25
4.1. Requisitos. 26
4.2. Diseño. 28
4.3. Implementación. 31
4.3.1. Las preguntas. 31
4.3.2. Gestos. 34
4.3.3. Redes sociales. 36
4.3.4. Imágenes de apoyo. 39
CAPÍTULO 5. Modo de uso. 41
CAPÍTULO 6. Gestión de la aplicación. 47
CAPÍTULO 7. Conclusión y trabajo futuro. 53
7.2. Trabajos futuros. 54
BIBLIOGRAFÍA. 55
REFERENCIAS. 55

Tabla de Figuras

Figura 1: Esquema del modelo incremental. 17
Figura 2: Modelo incremental aplicado. 17
Figura 3: Logo Android. 18
Figura 4: Arquitectura de Android. 19
Figura 5: Porcentaje de uso de cada S.O. Android. 21
Figura 6: Logo Eclipse. 22
Figura 7: Logo de Java. 23
Figura 8: Jerarquía de la interfaz, ... 28
Figura 9: Ejemplo de diseño UI en XML. 29
Figura 10: Ejemplos de Draw9-patch. 30
Figura 11: Ejemplo pregunta Respuesta Múltiple. 32
Figura 12: Ejemplo pregunta Respuesta Corta. 33
Figura 13: Ejemplo pregunta Verdadero o Falso. 33
Figura 14: Tabla de la base de datos. 33
Figura 15: Ejemplos 1 y 2 de Gestures Builder. 35
Figura 16: Ejemplos 3 y 4 de Gestures Builder. 35
Figura 17: Ejemplo de uso Respuesta V/F. 36
Figura 18: Conexión con Twitter. 37
Figura 19: Resultado Compartir Twitter. 37
Figura 20: Compartir Facebook. 38
Figura 21: Resultado Compartir Facebook. 38
Figura 22 Izq. Pinch Open. Dcha Pinch Close. 39
Figura 23: Ejemplo redimensión Imagen. 40
Figura 24: Ejemplo interfaz principal. 42
Figura 25: Ejemplo interfaz Opciones. 42
Figura 26: Izq. Seleción tipo. Dcha. Selección Tema. 43
Figura 27: Ejemplo interfaz estadísticas. 43
Figura 28: Interfaz Verdadero o Falso. 44
Figura 29: Ejemplo interfaz Respuesta Corta. 44
Figura 30: Ejemplo interfaz Respuesta Múltiple. 45
Figura 31: Izq. Contrarreloj Dcha. Resistencia 45
Figura 32: Eleccion red social. 46
Figura 33: Redimension de imagen 46
Figura 34: MainActivity.java 48
Figura 35:Carpeta Drawable. 48
Figura 36: Tabla excel 49
Figura 37: Vista PsPad 49
Figura 38: Vista comando reemplazar. 50
Figura 39: Ejecución comando reemplazar. 50
Figura 40: Vista reemplazo. 51

Resumen.

En este proyecto se ofrece una aplicación educativa para móviles con sistema operativo Android para estudiantes de programación de titulaciones de informática.

La aplicación tiene como objetivo principal el repaso de la programación en Java y, para ello, consta de diversos tipos de ejercicios basados en responder preguntas sobre los conceptos de un curso de introducción a la programación, que el alumno puede realizar sin necesidad de conectarse a ninguna red inalámbrica (salvo en ocasiones puntuales). El alumno también podrá consultar documentación de cada tema en línea, realizar ejercicios de mayor complejidad a los propuestos con anterioridad, volver a responder las preguntas falladas, ver sus estadísticas y cambiar el idioma de forma manual. Además, la aplicación será capaz de conectar con las principales redes sociales y compartir contenido propio de la aplicación en este tipo de plataformas.

La aplicación también permite realizar exámenes externos (uno al principio y otro al finalizar) que controlarán la evolución, y la participación de los alumnos. Y, desde el punto de vista del personal docente cabe destacar que la aplicación es de fácil mantenimiento ya que una persona con breves nociones de XML podría introducir nuevas preguntas o quitar las que ya estaban.

La aplicación se ha alimentado para la enseñanza de la programación con más de 300 preguntas de repaso divididas en 6 niveles que abarcan las nociones básicas de un curso de programación de un grado en informática.

En resumen se ofrece una aplicación creada para el estudio y el repaso de los alumnos en sus móviles o tablets android, para proporcionar un repaso que “vaya con ellos”. Además, permite compartir sus preguntas con sus contactos en dos redes sociales: facebook y twitter de manera colaborativa. También ofrece la posibilidad de evaluar los conocimientos de dos formas: por un lado los usuarios realizan dos exámenes (inicial y final) online para que puedan ser evaluados por el profesor; por otro lado mediante distintos tipos de exámenes offline para el alumnos (contrarreloj, de resistencia, o preguntas falladas) y todo ello con una interfaz amigable.

De esta forma se intenta conseguir una aplicación de gran diversidad en cuanto a preguntas y tipos de respuesta, con una mayor interactividad con respecto a otras aplicaciones evaluadas y fácil tanto en el uso, como en la gestión de la aplicación.

CAPÍTULO 1. Descripción y motivación.

En las siguientes líneas se hace una breve introducción al proyecto, exponiendo cuál es su motivación y qué objetivos son los que se pretenden conseguir.

1.1. Descripción del problema.

Actualmente las plataformas de distribución de aplicaciones para Smartphones[1] están repletas de programas para el tiempo libre, no siendo igual en el caso de las aplicaciones del sector educativo. Aún así, la mayoría de aplicaciones que se encuentran en esta categoría, están dedicadas al aprendizaje de idiomas y a la obtención del carnet de conducir.

En el caso de los lenguajes de programación encontramos muchas aplicaciones (de casi cualquier lenguaje) con mucha teoría y que permiten al alumno obtener conocimientos sin posibilidad de ponerlos a prueba. También es posible encontrar aplicaciones que incluyan preguntas (muchas menos) pero en estos caso o la interactividad es pobre, o el temario es pobre o ambas cosas.

Por estas razones se crea AprendeJava, una aplicación que concentra teoría y ejercicios, con el propósito de que el usuario pueda aprender de una forma más interactiva, efectiva y amena el lenguaje de programación Java aplicado para alumnos de primer curso de enseñanzas universitarias. Y que añade otras funcionalidades como recuperación de preguntas falladas, conexión con redes sociales, control de estadísticas, cambio manual de idioma, un mayor grado de interactividad incluyendo ejercicios de varios tipos e imágenes de apoyo. Y todo esto sin la más mínima presencia de publicidad. Como ocurre con algunas de las aplicaciones evaluadas.

1.2. Motivación y objetivos.

El aprendizaje electrónico ("e- learning[2]") es aquel que permite una educación a distancia completamente virtualizada. Permite al alumno realizar tareas correspondientes de una asignatura fuera tanto del centro de estudios, como del horario lectivo, dando lugar a un aprendizaje autodidáctico y adaptable al ritmo de cada usuario. Sin olvidarnos de lo infinitamente más atractivo que resulta, para las nuevas generaciones de estudiantes, la enseñanza online frente a enseñanza cara a cara.

Uno de los pocos inconvenientes que tienen este tipo de enseñanza, es que exigen más tiempo y un mayor esfuerzo por parte del personal docente. Es por eso que, con esta aplicación, se intenta reducir en la medida de lo posible ese tiempo y esfuerzo que muchas veces no se tiene.

A la hora de enseñar, motivación del alumno es una parte esencial de la enseñanza, ya que aumenta de forma positiva la implicación y la actividad del mismo dentro de una asignatura. Y como consecuencia, el alumno reforzará sus conocimientos de una forma más rápida y entretenida si lo hace en un entorno cómodo para él como su Smartphone o Tablet

.

Personalmente quería realizar este proyecto por dos motivos. Desde un principio me atraía mucho el hecho de comenzar desde cero en un entorno de desarrollo completamente nuevo para mí, como lo ha sido Android[3]. Aunque es una tecnología relativamente joven, dispone de una gran cantidad de documentación que hace más fácil el desarrollo de aplicaciones, sobre todo cuando se está empezando.

Otro de los motivos por el cual decidí realizar este proyecto es porque quería crear una aplicación que de alguna u otra forma fuese de ayuda para alguien y ¿Qué mejor que ayudar a formarse a futuros compañeros de profesión?

CAPÍTULO 2. Estado del arte.

En este apartado se mostrará otras aplicaciones educativas pero que por sus carencias no cubren nuestros objetivos. Y por otro lado se dará una visión detallada de lo que se pretende realizar con este proyecto.

2.1. Descripción de las alternativas encontradas.

Cómo se ha explicado anteriormente, se va a desarrollar una aplicación educativa sobre la programación en Java adaptada a personas que se estén iniciando en dicho lenguaje de programación.

En Play Store[4] de Android se pueden encontrar infinidad de aplicaciones dentro del ámbito educativo y debido a este gran número de alternativas, la comparativa puede convertirse en una tarea algo complicada de realizar. No obstante, me he centrado en las que he considerado más interesantes teniendo en cuenta los objetivos que se pretenden conseguir con el desarrollo de este proyecto.

Todas las aplicaciones que se han evaluado son relativamente distintas entre sí aunque comparten algunas características. Aún así, con AprendeJava se pretende crear una aplicación educativa mucho más completa que el resto. A continuación se describen las aplicaciones evaluadas junto con las carencias que este tipo de programas contienen.

La primera de las aplicaciones que se presenta es JavaQuiz. Se ha escogido esta aplicación porque es la que más se parece a AprendeJava. JavaQuiz es un juego de preguntas tipo test sobre java y que en principio tiene carácter educativo (más adelante explicaré esto).

Un segundo ejemplo de aplicación de preguntas sobre programación en Java (y otros lenguajes) es “Skill Test”. Esta aplicación es otro juego de preguntas de lenguajes de programación (son varios lenguajes y las preguntas están mezcladas). También se ha escogido por ser tener gran similitud con AprendeJava en cuanto a la idea base (aplicación de preguntas de programación).

Otra aplicación similar a AprendeJava es “python programming”. Esta aplicación toma como referencia el lenguaje de programación python, pero la idea sigue siendo la misma.

La última aplicación es “TodoTest”, una aplicación orientada 100% a aprobar el examen teórico de conducir. Desde mi punto de vista, es una aplicación más interesante que las anteriores (aunque no trate sobre la programación) por la profundidad del temario (algo básico en la formación de un alumno).

2.2. Carencias encontradas.

A continuación se expone un resumen de las aplicaciones y los principales problemas de las aplicaciones anteriormente mencionadas.

2.2.1. JavaTutor.

Java Quiz es una aplicación de preguntas tipo test sobre el lenguaje de programación Java. Aunque es atractivo en cuanto a diseño y funcionamiento, uno de los principales problemas de JavaQuiz es la poca profundidad del temario y su pobre distribución:

- Cuenta con 50 preguntas acerca de cualquier aspecto de java en un único tema sin importar el nivel del usuario, e incluyendo preguntas de carácter general que podrían estar incluidas en un apartado de "curiosidades".
- Otro aspecto negativo de esta aplicación es la cantidad de publicidad, mensajes emergentes y notificaciones que produce sobre otras aplicaciones que incluso no tienen nada que ver con la enseñanza de la programación.
- El último aspecto negativo es que no incluye apenas documentación para que el usuario (alumno) pueda adquirir conocimientos previos a la realización del ejercicio.

En cierto sentido parece que esta aplicación es una lanzadera para otras aplicaciones, ya que la publicidad, las notificaciones, y los mensajes emergentes son un elemento algo más que notorio tanto dentro como fuera de la aplicación.

2.2.2. Skill Test.

Skill Test es una aplicación de preguntas tipo test sobre varios lenguajes de programación. Uno de los grandes problemas es que se mezclan todos temas que alberga y esto, en vez de suponer una mejora, empeora el sistema, ya que provoca confusión y poca claridad. A continuación se citan otras carencias encontradas:

- Desconocimiento del número de preguntas (tanto en cada nivel como en total). En ningún lugar pone un progreso, o cuantas preguntas faltan para finalizar el tema.
- Desconocimiento del lenguaje de programación que se está abordando.
- No aparece teoría. Al menos para saber que temas se están tratando.
- Poca funcionalidad. Sólo se pueden contestar preguntas, nada más.
- Poca interactividad. Sólo se pueden contestar preguntas tipo test.

Esta aplicación no es una buena aplicación educativa ya que provoca mucha confusión e incertidumbre debido a la poca información que muestra al usuario.

2.2.3. Python programming.

Esta aplicación aborda el tema de la programación en python. Seguimos con los mismos problemas:

- Desconocimiento del número de preguntas (tanto en cada nivel como en total).
- No hay progresión en las preguntas. Las preguntas se muestran en pantalla hasta que se aciertan.
- Sólo existe una referencia a teoría. Siendo esta la página oficial de python
- Poca funcionalidad. Sólo se pueden contestar preguntas, nada más.
- Poca interactividad. Sólo se pueden contestar preguntas tipo test.

2.2.4. TodoTest.

"Todotest" es una aplicación que ofrece una gran variedad de exámenes teóricos sacados directamente de la Dirección General de Tráfico. Por su parte, la aplicación "Todotest", aunque mucho más gratificante en cuanto a profundidad del temario y su distribución, también tiene alguna carencia.

- Falta de teoría (o acceso a ella) donde poder consultar las dudas surgidas.
- Ninguna comunicación externa más allá de las actualizaciones.

Llegados a este punto, y habiendo analizado estas aplicaciones, se va a exponer una tabla comparativa de las aplicaciones junto con la aplicación que se pretende crear en este proyecto:

Características	Skill Test	Python	JavaQuiz	TodoTest	**Aprende Java**
Sin publicidad	✓	✓	Mucha	Mínima	**Nada**
Conexión Ext.*	Ninguna	Ninguna	Puntual	Puntual	**Puntual**
Nº Preguntas	**¿?**	**¿?**	Insuficiente	Muy alta	**Media(300)**
Distribución **	Mala	Mala	Mala	Buena	**Buena**
Idioma	Inglés	Inglés	Inglés	Español	**Esp. / Ing.**
Teoría	X	X	X	X	✓
Estadísticas	X	X	✓	X	✓
Repaso errores	X	X	✓	✓	✓
Compartir	X	X	✓	X	✓
Imágenes	X	X	X	✓	✓
Actualización	X	X	X	✓	✓
Interactividad	Pobre	Pobre	Media	Media	**Alta**

*Conexión con internet.
**Se refiere a la distribución (orden) de las preguntas dentro de la aplicación.

Cómo se puede observar, AprendeJava mejora mucho con respecto a las alternativas descritas anteriormente, ya que tiene una mayor interactividad (no se centra solo en los ejercicios tipo test), puede acceder a teoría, está disponible en dos idiomas, se puede consultar estadísticas, contiene imágenes de apoyo, las preguntas están ordenadas por temas, etc.

Con todas estas mejoras AprendeJava se convierte en una opción mucho más que atractiva en lo que se refiere a las aplicaciones educativas. Y si hablamos de aplicaciones educativas que ayudan en la enseñanza de un lenguaje de programación aún más.

2.3. Objetivos.

Se pretende construir una aplicación interactiva para el aprendizaje del lenguaje programación Java que sea atractiva para el estudiante y que se pueda utilizar en cualquier momento y lugar mediante un Smartphone Android. Y que además se pueda compartir el contenido de la aplicación en las redes sociales.

Conociendo esto, se describirán las particularidades que se han desarrollado en esta aplicación:

- Ofrecer una aplicación para el aprendizaje de la programación.
- Realización de exámenes para el control del conocimiento y de la actividad. En este caso estos exámenes están hechos en Moodle, permitiendo acceder vía web a los exámenes propuestos.
- Realización de ejercicios de repaso. Incluyendo no sólo preguntas tipo test (como es habitual), sino también preguntas de verdadero /falso y preguntas de respuesta corta.
- Realización de ejercicios "bajo presión": Se añaden exámenes en los que se combinan los tres tipos de respuesta y además un elemento que proporciona algo de "presión". Esto se realiza mediante el examen de resistencia, en el que no se puede fallar ni una vez para finalizarlo, y el examen contrarreloj , en el que el alumno deberá responder a todas las preguntas antes de que se acabe el tiempo.
- Repaso de las preguntas falladas.
- Posibilidad de consultar teoría.
- Posibilidad de cambiar el idioma de forma manual.

- Posibilidad de consultar las estadísticas.
- Compartir conocimiento a través de las redes sociales: Twitter y Facebook.
- Posibilidad de acompañar las preguntas con imágenes de apoyo.
- Y todo ello con una interfaz sencilla y amigable.

CAPÍTULO 3. Metodología y tecnologías usadas.

En esta Sección hablaremos del proceso de desarrollo del software en cuestión, las decisiones tomadas a lo largo de la implementación y las tecnologías que se han usado para obtener las metas propuestas.

3.1. Metodología.

La ingeniería del software[5] es la disciplina encargada de todo el proceso de planificación, desarrollo y mantenimiento de un software en concreto. Para poder dar "vida" a una aplicación mediante la ingeniería del software, hay que seguir un proceso dividido en cuatro actividades:

- **Especificación software**: En esta etapa se define el problema y las restricciones del mismo. (Análisis).

- **Desarrollo software**: En esta etapa se realiza la implementación del software mediante las herramientas y procesos que se definieron en la etapa anterior (entre otras cosas). (Diseño e implementación).

- **Validación software**: El software es puesto a prueba en esta etapa para comprobar que todo se adecua de acuerdo a la especificación. (Pruebas)

- **Evolución software**: El software se mantiene y se cambia en función de nuevos requisitos solicitados. (Implantación).

Pero con lo descrito anteriormente no es suficiente para desarrollar un software, sino que necesitamos de un modelo de proceso software para que la creación de la nueva aplicación sea más eficiente y se adapte lo mejor posible a los objetivos que se persiguen.

En el caso de este proyecto se ha decidido utilizar el **modelo incremental[6] (Figura1)** ya que es el que mejor se adaptaba a las circunstancias del desarrollo. Este modelo pertenece al grupo de modelos evolutivos, que son aquellos en los que ya sea por cambios de requisitos o por exigencias externas van a cambiar o incrementar sus funciones a lo largo de su vida (tanto en etapas de desarrollo como en etapas de mantenimiento).

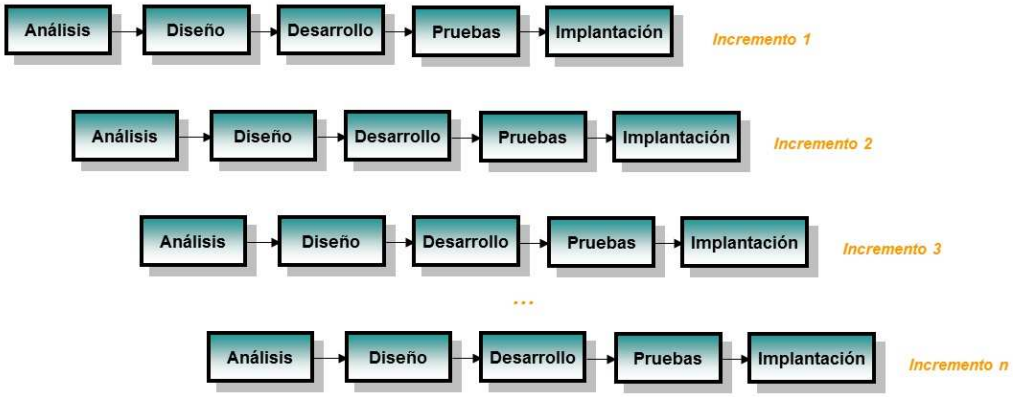


Figura 1: Esquema del modelo incremental.

En el caso de AprendeJava, se puede especificar cada incremento (iteración) con la aparición de nuevas funcionalidades. De esta forma se va ampliando la aplicación con los nuevos requisitos que han ido surgiendo a lo largo del desarrollo **(Figura 2).**

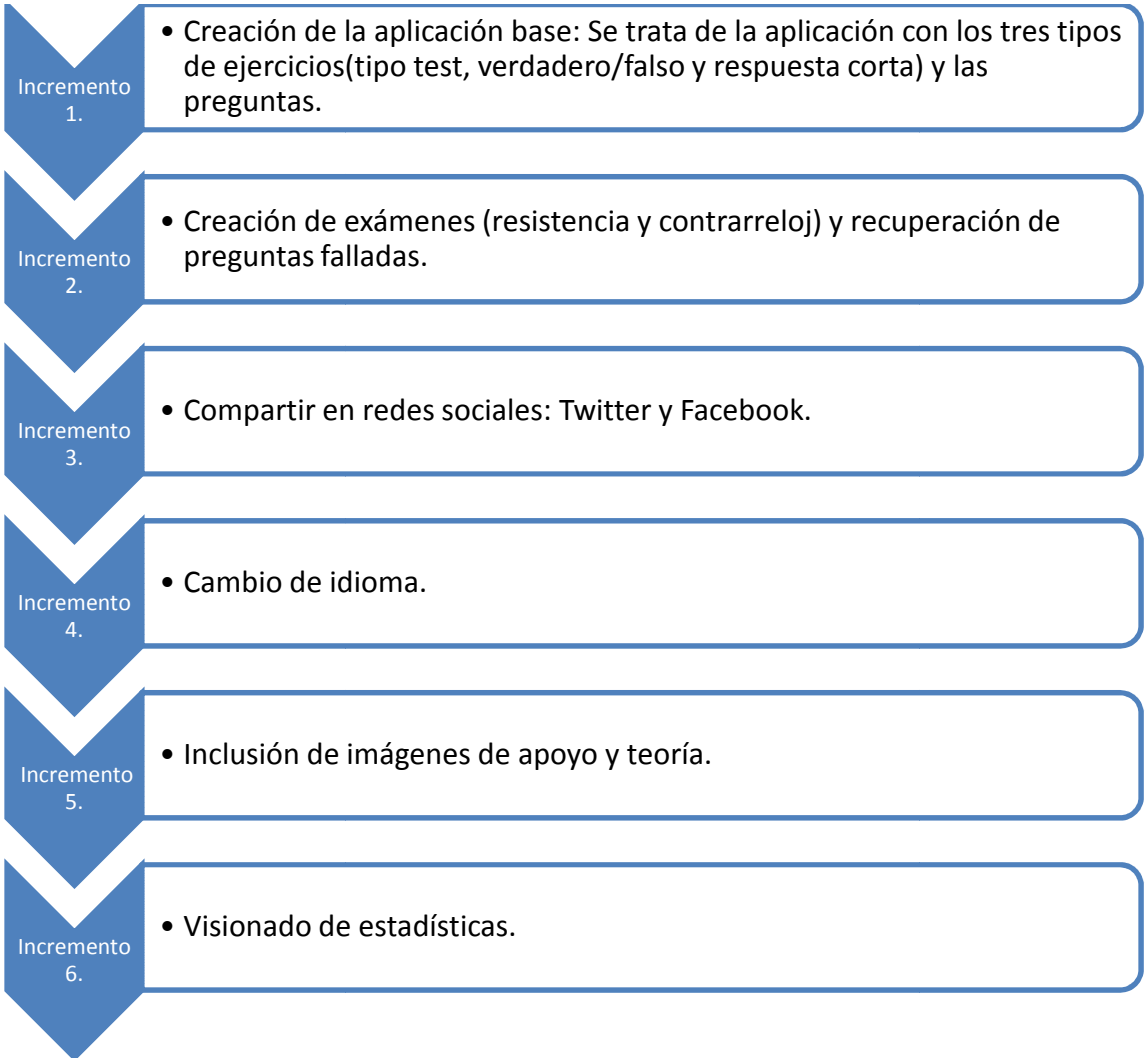


Figura 2: Modelo incremental aplicado.

3.2. Tecnologías.

Cuando hablamos de Smartphone nos referimos a los sucesores de los teléfonos móviles sencillos que han ido evolucionando hasta "ordenadores que caben en la palma de la mano" y que incluyen las funciones básicas de sus antecesores como llamadas, envío y recibo de sms, mms, etc…

A partir de 2007, los Smartphones empezaron a cobrar importancia debido a su relación entre la potencia (como ordenador) y su reducido tamaño. Llegando incluso a superar en ventas a los móviles "no inteligentes" y a ser indispensables en ciertos sectores de la sociedad.

Junto con la aparición de los teléfonos inteligentes se presentaron, a su vez, los sistemas operativos que éstos necesitan llevar integrados: Android, IOS, Symbian... Cada uno con características propias, ofreciendo así, un entorno de fácil manejo para el usuario final y un sinfín de aplicaciones desarrolladas tanto por empresas como particulares.

3.2.1. Android.

Fue desarrollado inicialmente por Android Inc **(Figura 3).** Que posteriormente fue comprada por Google en 2005. Ese mismo año se desarrollaba la máquina virtual para dispositivos móviles (Dalvik VM[7]), optimizada para requerir poca memoria y diseñada para poder ejecutar varias máquinas al mismo tiempo sin que el sistema anfitrión gestione los procesos, la memoria y los hilos.

En 2007 se creó "Open Handset Alliance[8]", un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles, en el que figuran nombres como Google, Samsung, Toshiba y Vodafone entre otros. El cinco de noviembre de ese mismo año se lanza la versión beta del hoy tan popular sistema operativo, llegando la primera versión comercial (conocida como 1.0 ó Apple pie) el veintitrés de septiembre del año siguiente. A partir del 2008 y hasta la actualidad se han desarrollado 10 versiones nuevas, cada cual más potente que la anterior y adaptadas también a los avances en el campo del hardware móvil.

Figura 3: Logo Android.

Gracias a la rápida actualización de las versiones (ya sea incorporando características o mejorando las presentes), al software libre, y a la gran abanico de dispositivos móviles que existen en el mercado, "asequibles para todos los bolsillos", han hecho que este Sistema Operativo tenga una cuota de mercado de más del 75% en 2012, desbancando así a sus principales competidores, IOS (iphone) y Blackberry OS (Blackberry).

La arquitectura de Android[9] **(Figura 4)** está formada por cuatro capas que facilitan la creación de aplicaciones. Estas capas están organizadas en forma de pila para que cada capa pueda usar elementos de la capa inferior. Además, ésta arquitectura, ofrece al desarrollador un conjunto de librerías (de software libre) que permiten acceder y hacer uso de los componentes hardware del Smartphone. Para entender mejor esta arquitectura se muestra el siguiente gráfico y se detallará cada capa:

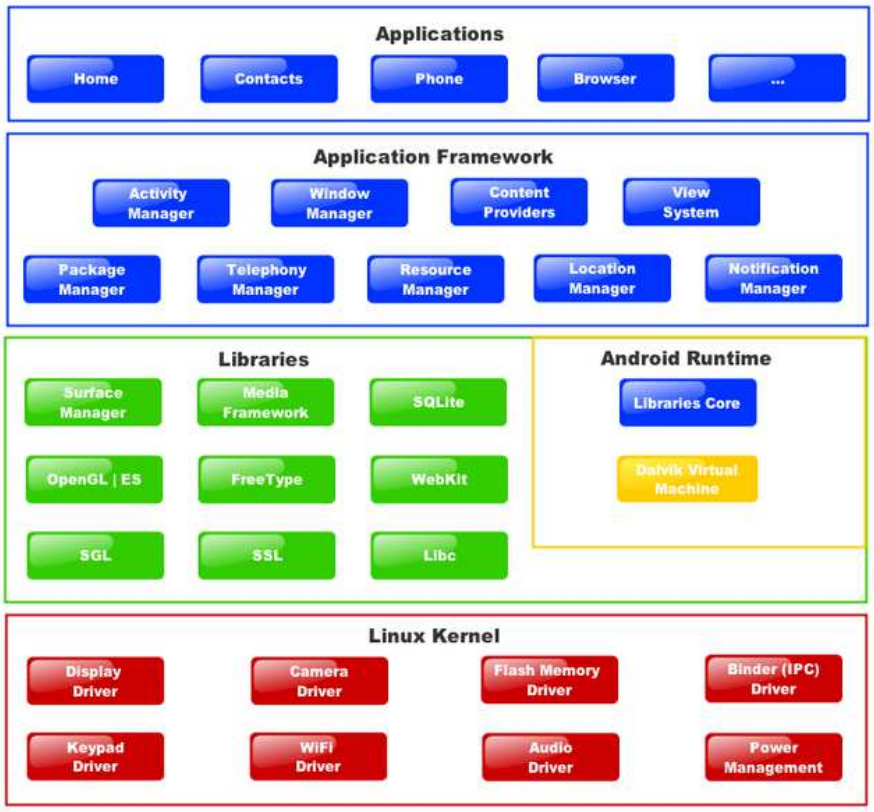


Figura 4: Arquitectura de Android.

- **Aplicaciones:** Este nivel está formado por el conjunto de aplicaciones nativas (sin interfaz) del dispositivo Android (desarrolladas en c/c++) , las administradas (con interfaz y desarrolladas en Java), y las que el usuario ha instalado.

- **Framework de aplicaciones:** Esta capa está formada él conjunto de herramientas de desarrollo de cualquier aplicación. La mayoría de componentes de esta capa son librerías que se usan para el desarrollo de aplicaciones (nivel superior), es decir, todas las aplicaciones desarrolladas en Android usarían este Framework y el mismo conjunto de APIS de este nivel. Entre las API más importante se encuentran las siguientes: Activity Manager, Window Manager, Content Provider, Telephone Manager, Location Manager, Notification Manager….

- **Librerías:** Este nivel se corresponde con las librerías que usa Android para su funcionamiento. Están compiladas en el código nativo del procesador para que las funciones que se repiten con frecuencia no necesiten ser codificadas cada vez que se ejecutan. Han sido escritas en C/C++ y junto con el kernel constituyen el núcleo de Android. Algunas de estas librerías son: OpenGl, Surface, Manager, SQLite…

- **Runtime de Android:** Este nivel se refiere a la máquina virtual (Dalvik).Las aplicaciones se codifican en Java y son compiladas en un formato especñifico para que esta máquina virtual las ejecute. De esta forma tenemos la certeza que la aplicación, una vez distribuida, funcionará en los dispositivos que tengan la versión mínima del sistema operativo que requiera la aplicación.

- **Kernel de Linux:** El núcleo del sistema operativo Android está basado en el kernel de Linux 2.6, similar al que puede utilizar cualquier distribución Linux (como Ubuntu). Esta capa actúa como capa de abstracción entre el hardware y el resto de la pila. Por lo tanto, es la única que es dependiente del hardware.

Como podemos ver, Android es un sistema Operativo sumamente rico y potente en cuanto a opciones a desarrollar ya que no hay elemento, función, o característica a la que no podamos acceder.

Para este proyecto se eligió este sistema operativo por los siguientes motivos:

- **Gran aceptación entre los usuarios**: la posibilidad de tener un Smartphone conforme a las necesidades de cada uno hacen de este sistema operativo una opción muy atractiva.

- **Gran cuota de mercado**: Es básicamente una consecuencia del punto anterior. Android cuenta con más de un 75% de los Smartphones que actualmente están en circulación.

- **Es un sistema operativo de código libre**: Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto.

- **Facilidades en el desarrollo:** la página de desarrolladores de Android dispone de muchísimas páginas de documentación en el que se explica desde la implementación de interfaces gráficas , toda la interfaz de programación de aplicaciones, y sobre el uso de herramientas como Sqlite3, 9patch...etc.

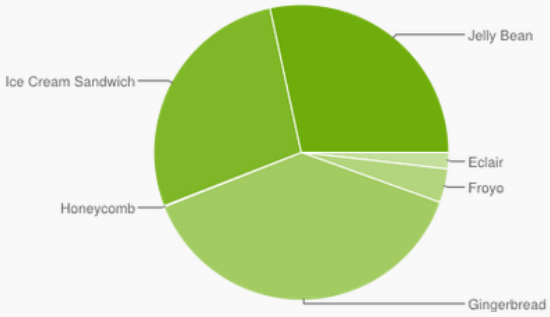
- **Baja inversión:** Si queremos dedicarnos al desarrollo de aplicaciones Android de forma profesional la inversión que tenemos que hacer al principio es muy baja si disponemos de Smartphone y de un ordenador de gama media. Por el contrario, en el caso de plataformas como IOS (su principal competidor) tendríamos una inversión inicial algo superior si no se dispone de ordenador MAC(son sustancialmente más caros que el resto).

El proyecto se ha desarrollado para la versión 4.2 de Android(Jelly Bean) pero con compatibilidad desde la versión 2.2 (Froyo). De esta forma, y según la página de desarrolladores de Android , con esta configuración se puede llegar a un 98.2% de los usuarios de la plataforma a día de hoy **(Figura 5).**

This section provides data about the relative number of devices running a given version of the Android platform.

For information about how to target your application to devices based on platform version, read Supporting Different Platform Versions.

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	3.7%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	38.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1.x	Jelly Bean	16	26.1%
4.2.x		17	2.3%



Data collected during a 14-day period ending on May 1, 2013.
Any versions with less than 0.1% distribution are not shown.

Figura 5: Porcentaje de uso de cada S.O. Android.

La arquitectura utilizada para el desarrollo del proyecto ha sido el llamado **Modelo Vista Controlador (MVC[10])**. A continuación se presentan unas definiciones:

- **Modelo**: Cuando hablamos de modelo, nos referimos a los datos (o recursos) que van ser consumidos por nuestra aplicación. Dependiendo que tipo de datos sean y de las necesidades que se tengan los datos se presentarán de una forma u otra (en bases de datos ya sean externas o internas, a través de webServices….).

- **Vista:** La vista es la interfaz gráfica que se ofrece al usuario. En el caso de Android, éstas se construyen a partir de archivos XML en los que se da

forma, color y formato a los elementos que se desplegaran en la pantalla de nuestro dispositivo.

- **Controlador:** Es el conjunto de clases que darán sentido a lo que ya definimos previamente en la vista y que nos ayudarán a consumir los recursos y a desplegar información. En el caso de Android el controlador ésta basado en un lenguaje java con las particularidades del sistema Android.

Con estas definiciones, es evidente que una de las características principales del modelo vista controlador es la separación de los recursos que va a consumir la aplicación, de la lógica (controlador) que esta lleve y de la propia interfaz gráfica (Vista).

De esta forma podemos seccionar el trabajo de una forma muy eficiente (y mucho más cuando se cuenta con un equipo de desarrollo). Otra ventaja del MVC es que al separar el código de la interfaz del código de lógica, se consigue una mayor claridad en el mismo, y, por tanto, un mejor mantenimiento, reutilización, y corrección.

3.2.2. Eclipse.

Es un entorno de desarrollo integrado (IDE, Integrated Development Enviroment). Un IDE es un programa informático compuesto por un conjunto de herramientas de código libre que facilitan el desarrollo de aplicaciones.

Eclipse **(Figura 6)**, aunque principalmente se usa para el desarrollo de aplicaciones Java, es un IDE con soporte para una amplia gama de lenguajes: C, C++, python, ETC…. Y cuenta a su vez, con una serie muy grande de complementos (plugin) que amplían su funcionalidad, como en el caso del plugin ADT(Android development Toolkit).

Figura 6: Logo Eclipse.

ADT es un plugin que sirve para extender las capacidades de eclipse y te permite configurar de forma rápida nuevos proyectos, crear interfaces gráficas y probar tu software mediante usando el Software Development Kit (SDK[11]) de Android.

Dentro de la página de desarrolladores de android explican, paso a paso, como instalar el plugin ADT en eclipse.

3.2.3. Java.

El lenguaje de programación Java **(Figura 7)** es utilizado en la implementación del **controlador.** Es un lenguaje orientado a objetos desarrollado por Sun Microsystems y publicado en 1995. Las principales características de este lenguaje son:

Figura 7: Logo de Java.

- **Simple**: La estructura es muy parecida a la de C/C++ pero sin incluir las herramientas de bajo nivel.

- **Orientado a objetos**: Aunque no es en su totalidad orientado a objetos, la mayoría de la programación si lo está, facilitando la manipulación de las estructuras encapsuladas.

- **Distribuido:** Permite crear y manipular conexiones con servidores o clientes remotos.

- **Indiferente de la arquitectura:** Java es una combinación de lenguaje interpretado y lenguaje compilado. El código es interpretado a un código que conoce la máquina virtual de Java y a continuación se compila al código máquina del sistema anfitrión. Es por este motivo que un programa java es portable e igual independientemente de la plataforma en la que se ejecute.

- **Multihilo:** Java también es capaz de trabajar la programación concurrente.

3.2.4. XML.

Es un lenguaje de marcas desarrolado por World Wide Web Consorcium que permite estructurar documentos grandes para que sean fáciles de entender y que propone un estándar para el intercambio de información estructurada.

En este caso concreto XML implementa la **vista** de la aplicación y se utiliza para guardar las preguntas de forma estructurada.

3.2.5. SAX.

SAX[12] es un estándar para el tratamiento de documentos XML. En este caso, se utiliza para la lectura de los ficheros de preguntas.

3.2.6. Twitter4J.

Twitter4J[13] es una librería Java no oficial para la interfaz de programación de aplicaciones de Twitter. Con esta librería podemos integrar fácilmente nuestra aplicación con el servicio ofrecido por Twitter. (Aunque no es una librería oficial es aceptada por Twitter).

3.2.7. Facebook SDK.

Facebook SDK[14] es el kit de desarrollo que ofrece Facebook para integrar de la forma más sencilla posible nuestra aplicación (en este caso de Android) con su plataforma.

3.2.8. Draw9-patch.

Draw9-patch[15] es una herramienta desarrollada para crear gráficos "ninePAtch".Este tipo de gráficos están diseñados para que Android redimensione el tamaño automáticamente y acomodar el contenido de la vista en la que se ha colocado el gráfico.

3.2.9. SQLite3.

Esta herramienta[16] sirve para visualizar (y también modificar) desde la consola del sistema anfitrión (en mi caso Windows) las bases de datos que se hayan creado dentro de la máquina virtual.

CAPÍTULO 4. Descripción informática.

En este capítulo se dará una visión más detallada de las funcionalidades que se van a desarrollar en la aplicación, las posibilidades que se planteaban a la hora de desarrollar y el porqué se ha elegido una alternativa y no otra.

4.1. Requisitos.

Como ya se ha explicado anteriormente, este proyecto nace como alternativa a las opciones que actualmente están en el mercado y que sirven para repasar conocimientos sobre un tema específico.

El resultado obtenido ha sido AprendeJava, una aplicación de idea muy simple pero enriquecedora tanto en el desarrollo, como en el uso.

A continuación se va a listar la toma de requisitos

1) Una aplicación en la que se puedan contestar preguntas que el personal docente crea oportunas.

2) El nivel de mantenimiento de la aplicación, y por tanto, de las preguntas, debe ser mínimo.

3) La existencia de más de una forma de contestar a las preguntas.

4) La existencia de elementos que pongan a prueba la rapidez o la resistencia a la hora de contestar

5) La existencia de niveles con nota de corte (Si no se ha superado el nivel anterior, no podemos acceder al nuevo).

6) La posibilidad de consultar documentación específica.

7) La posibilidad de añadir imágenes de apoyo.

8) La posibilidad de compartir las preguntas.

9) El guardado de las preguntas falladas para un posterior repaso.

10) La existencia de un examen previo a las preguntas y uno posterior para analizar la diferencia al pasar por la aplicación.

11) Posibilidad de ver estadísticas.

12) Cambio de idioma en tiempo de ejecución.

13) Alimentar la aplicación con contenidos.

A partir de la toma de requisitos se puede establecer una aplicación “base” que constaría del desarrollo del sistema de preguntas completo, es decir, la implementación de la aplicación con los requisitos del 1) al 5).

Y luego por cada apartado que se menciona en la lista anterior una nueva característica, obteniendo así la funcionalidad que a continuación se describe:

- Para conseguir una mayor interactividad se han realizado:
  - Preguntas tipo test: Con cuatro posibles opciones.
  - Preguntas Verdadero/ Falso: Se dibuja en la pantalla la respuesta y automáticamente se valida.
  - Preguntas de respuesta corta: La respuesta debe ser escrita (da igual si es en mayúsculas o minúsculas.
  - Exámenes contrarreloj: Se selecciona un número de preguntas de entre todas y se da un tiempo para contestarlas.
  - Exámenes de resistencia: Se usan todas las preguntas del sistema y la aplicación sólo se continúa si se ha acertado la respuesta.
  - Repetir las preguntas falladas: Se pueden volver a contestar las preguntas que se han fallado a lo largo de la aplicación.

  Con estos ejercicios conseguimos que la tarea del alumno no se vuelva tan repetitiva.

- Realización de exámenes externos a la aplicación: De esta forma el personal docente puede obtener una visión global de la utilidad de la aplicación y la evolución que ha tenido cada alumno.

- Posibilidad de compartir preguntas en redes sociales. Ya que las personas que pertenecen a nuestra red también pueden ser parte de nuestra formación aportando sus conocimientos.

- La creación de una aplicación que puede cambiar de idioma en tiempo de ejecución: Consiguiendo así llegar a un mayor número de personas.

- La posibilidad de consultar teoría en el momento que no sepamos la pregunta nos sirve para afianzar conocimientos y familiarizarse con conceptos nuevos. Esta consulta se realiza a través de una página web embebida. Para lo que hace falta tener conexión a internet.

- El visionado de estadísticas sirve al alumno para intentar superarse hasta conseguir el máximo nivel.

4.2. Diseño.

En este apartado es necesario que recordemos que el **modelo vista controlador** se basa en la separación de Java por parte del controlador, XML para la vista y los datos (organizados de alguna forma) que se van a consumir en la aplicación.

A la hora de hablar de la vista tenemos que saber que todos los componentes de la interfaz de usuario de Android descienden de la clase ***View.*** Dichos objetos están organizados en forma de árbol **(Figura 8)** y pueden contener nuevos objetos View, permitiendo crear interfaces muy completas.

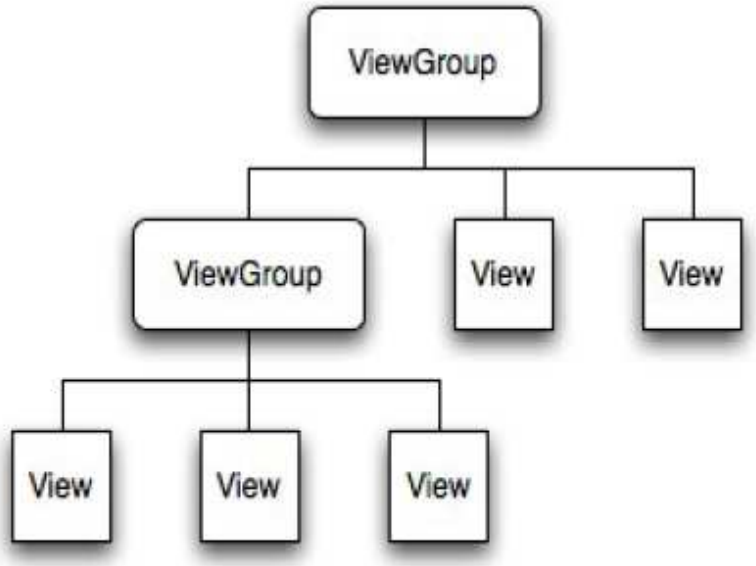


Figura 8: Jerarquía de la interfaz,

Un View como los que se muestran en la imagen se encargan de diseño de la pantalla y gestionan las interacciones con su área rectangular asociada. Y los ViewGroup son un conjunto de Views "hijas" dispuestos en la pantalla mediante el uso de layouts.

Existen varios tipos de layout, y dependiendo de nuestras necesidades, podremos hacer uso de uno o varios dentro de una misma vista. Algunos ejemplos de layout son: linear layout, relative layout, table layout, absolute layout….

Estas vistas y conjunto de vistas se pueden definir de dos formas:

- Mediante ficheros XML guardados en el directorio del proyecto "res/layout.
- Directamente en el código.

Es evidente que si estamos hablando del modelo vista controlador, nos decantemos por la primera opción, ya que facilita la depuración y el mantenimiento de la propia interfaz. El siguiente, es un ejemplo de viewGroup (en este caso un RelativeLayout) y algunos de los componentes que contiene. Cómo podemos observar **(Figura 9)**, cada componente

tiene atributos propios que definen desde su tamaño, el color del texto (si tiene) , hasta si va a tener una imagen definiendo su “background”. Los atributos más importantes y que siempre hay que definir dentro de una vista son el ID para poder referenciar los componentes en un futuro y las dimensiones (layout_heigh y layout_width) que harán que se defina mejor el componente.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity"
    android:background="@drawable/pizarra_001">

    <TextView
        android:id="@+id/TV1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textColor="@color/blanco"
        />

    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/TV1"
        android:layout_centerHorizontal="true"
        android:background="@drawable/tiza_larga_001" />

    <TextView
        android:id="@+id/TV2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="15dp"
        android:layout_below="@id/spinner1"
        android:textColor="@color/blanco"
        />

    <Spinner
        android:id="@+id/spinner2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/TV2"
        android:layout_centerHorizontal="true"
        android:background="@drawable/tiza_larga_001"
         />

        <Button
            android:id="@+id/btnCont"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/spinner2"
            android:layout_marginTop="23dp"
            android:layout_marginLeft="10dp"
            android:background="@drawable/tiza_001"
            android:textColor="@color/blanco"
            android:typeface="serif" />
```

Figura 9: Ejemplo de diseño UI en XML.

El plugin ADT[17] ofrece también la posibilidad de crear la interfaz con un editor gráfico para poder ver la organización de los componentes a medida que se van guardando los cambios. De esta forma, una persona que no se encuentre a gusto con el código XML puede crear una interfaz e incluso definir eventos que el usuario dispare sin necesidad de escribir ni una sola línea de código.

Y como podemos deducir no es lo mismo hacer una interfaz para una pantalla de 3.2’’ (por ejemplo Samsung Galaxy mini 2) que para una pantalla de 5.5’’ (Samsung

Galaxy note 2). Para solventar este problema, Android nos recomienda encarecidamente el uso del layout Relativo para la composición de las vistas, el uso de “Draw-9patch” para el correcto redimensionamiento de los recursos (tipo background) y la creación de recursos de distintos tamaños para las distintas densidades de pantalla (ldpi,mdpi,hdpi,xhdpi).

La herramienta Draw -9patch permite crear gráficos redimensionables sin perder las proporciones del recurso en cuestión. La forma de crear un gráfico ninePatch es con el programa del mismo nombre que viene instalado dentro del SDK (~\Android\android-sdk\tools). Para ello tenemos que cargar el recurso que queremos convertir , y una vez que se haya cargado marcar en la zona añadida extra (del tamaño de 1 pixel) las zonas que se pueden estrechar y las que se pueden ampliar sin que afecte al contenido de la imagen**(Figura 10)**.

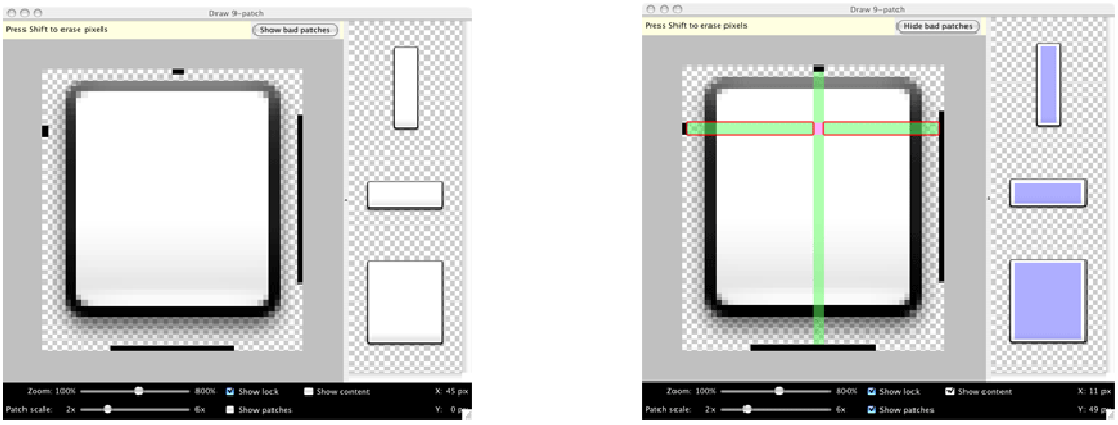


Figura 10: Ejemplos de Draw9-patch.

Las zonas superior e izquierda sirven para marcar las zonas que se pueden ampliar y las zonas inferior y derecha sirven para marcar las zonas que se pueden estrechar.

Aplicando a este proyecto estas técnicas y estos recursos obtenemos las interfaces para los distintas partes de la aplicación (que más adelante se mostrarán junto con un breve tutorial sobre su utilización).

4.3. Implementación.

Para hablar de la implementación tenemos que volver a las funciones que va a tener la aplicación y comentar las particularidades

4.3.1. Las preguntas.

A la hora de analizar los diferentes métodos existentes para incluir las preguntas se presentaban 3 opciones:

- **Base de datos externa:** Es la opción más potente de las que se van a describir ya que permite tanto una mejor actualización como visualización de las preguntas. Otra ventaja es que podemos usar las tecnologías que mejor conozcamos y que nos sintamos más a gusto, pero exige tener un servidor activo en todo momento con todos los problemas que ello conlleva (caídas, perdidas de conexión, mantenimiento).

- **Base de datos interna:** Esta opción se desarrolla con una base de datos ligera que incorpora el SDK de Android (SQLite). Esta opción aunque también bastante potente es de peor mantenimiento porque se necesita hacerlo a través de código, o bien, usando herramientas externas tipo SQLite3 (a partir de la consola del sistema).

- **Ficheros XML:** Aunque es menos potente que las otras opciones, su sencillez y su fácil mantenimiento hacen de ella la opción finalmente escogida .

Android, en su API, nos ofrece unas librerías que vienen de Java y que sirven para tratar ficheros XML: DOM y SAX.

- DOM[18] (*Document Object Model*) nos proporciona un estándar para representar documentos HTML y XML. La estructura que obtenemos a la hora de analizar el documento XML es la de un árbol jerárquico en memoria. Esto puede suponer un problema si el documento es excesivamente grande, ya que los móviles no suelen contar con mucha memoria.

- SAX (*Simple Api for XML*): nos provee un mecanismo para leer datos de un documento XML. La forma de operar de SAX es secuencial, ya que analiza el documento de principio a fin leyendo las etiquetas que se encuentra, y lanzando un evento cada vez que detecta un elemento que nosotros consideremos “de interés”.

El motivo por el cual se ha elegido SAX para resolver este problema es que presenta interesantes ventajas con respecto a DOM en el desarrollo de aplicaciones móviles. A priori no conocemos el número de preguntas, pero si sabemos que en un momento determinado vamos a usar todas al mismo tiempo, y también sabemos que SAX se comporta mejor con archivos grandes ya que no usa tanta memoria para ello (solo estaría en memoria la parte que se está analizando).

Una vez elegido el método en el que vamos a leer las preguntas, debemos definir la estructura XML del fichero, ya que cada una va a contener toda la información necesaria para alimentar la aplicación. Esta información es la siguiente:

- Tipo: En el caso de los exámenes (contrarreloj y resistencia) y la recuperación de las preguntas falladas se usa para saber que interfaz hay que usar, es decir, la interfaz en estos casos se establece de forma dinámica.

- Pregunta: Contiene la pregunta.

- Resp[1,2,3,4,V]: Contienen las respuestas disponibles y "respV " la correcta de las anteriores. En el caso de los ejercicios de respuesta corta o Verdadero y falso, estos valores (exceptuando respV) se pondrían a null.

- Teoría: Ciertas preguntas una url con teoría asociada a la pregunta. En el caso de no contener se pone a null.

- Imagen: Ciertas preguntas contienen una Imagen asociada que sirve para mostrar código de una forma más cómoda. Para incluir una imagen en una pregunta sólo tendremos que incluir el nombre de la imagen en el apartado "imagen" de la pregunta asociada dentro del fichero XML. (como se muestra en el primer ejemplo). Si no se tiene imagen se escribe null en ese campo.

A continuación se muestran unos ejemplos de cada tipo de preguntas **(Figuras 11, 12,13)**:

```
<objeto>
    <tipo>respMultiple</tipo>
    <pregunta>Dado el siguiente fragmento de código, sup
    <resp1>n=1 c='2'</resp1>
    <resp2>n=12 c='_'</resp2>
    <resp3>n=12 c='a'</resp3>
    <resp4>n=12 y tiene que esperar a que el usuario int
    <respV>n=12 c='a'</respV>
    <teoria>null</teoria>
    <imagen>preg93</imagen>
</objeto>
```

Figura 11: Ejemplo pregunta Respuesta Múltiple.

```
<objeto>
  <tipo>resCorta</tipo>
  <pregunta>¿Cómo accedemos al séptimo elemento de un ar:
  <resp1>null</resp1>
  <resp2>null</resp2>
  <resp3>null</resp3>
  <resp4>null</resp4>
  <respV>p[7]</respV>
  <teoria>www.codexion.com/tutorialesjava/java/nutsandbo
  <imagen>null</imagen>
</objeto>
```

Figura 12: Ejemplo pregunta Respuesta Corta.

```
<objeto>
    <tipo>vof</tipo>
    <pregunta>Las instrucciones if..else y switch son eq
    <resp1>null</resp1>
    <resp2>null</resp2>
    <resp3>null</resp3>
    <resp4>null</resp4>
    <respV>VERDADERO</respV>
    <teoria>www.codexion.com/tutorialesjava/java/nutsand
    <imagen>null</imagen>
</objeto>
```

Figura 13: Ejemplo pregunta Verdadero o Falso.

Hasta aquí hemos visto como se almacenan los datos dentro de nuestra aplicación, pero también hay que mencionar que en el caso en el que fallemos alguna pregunta, ésta también se guarda para su posterior repaso. El método usado esta vez es una base de datos interna SQLite ,ya que la gestión de la misma es automática (depende de si fallamos o acertamos la pregunta) y no es necesaria la conexión con un servidor.

PreguntasFalladas		
ID	integer(10)	
Tipo	varchar(255)	N
Pregunta	varchar(255)	N
resp1	varchar(255)	N
resp2	varchar(255)	N
resp3	varchar(255)	N
resp4	varchar(255)	N
respV	varchar(255)	N
teoria	varchar(255)	N
imagen	varchar(255)	N
idioma	varchar(255)	N

Figura 14: Tabla de la base de datos.

Como se puede observar **(Figura 14)**, con ésta única tabla se gestiona todo el sistema. Se ha incluido también el idioma de la pregunta para después seleccionar solo las del idioma que estemos usando. Los atributos de esta tabla son:

- Tipo: Se corresponde con el tipo de pregunta que va a estar almacenada (Respuesta múltiple, Verdadero o falso, o Respuesta corta).
- Pregunta: Es la pregunta que se ha fallado con anterioridad.
- Resp1,2,3,4: Son las distintas opciones que se dan. (en el caso de Verdadero o falso y Respuesta corta estos valores se guardarán como "null").
- RespV: Es la respuesta verdadera.
- Teoría: En caso de tener teoría, se guarda en este campo.
- Imagen: El nombre de la imagen asociado a esta pregunta.
- Idioma: El idioma en el que está la aplicación en ese momento.

Todos estos campos son Textos, que en base de datos son varchar(255).

4.3.2. Gestos.

Continuando con el tema anterior, no podemos pasar por alto las preguntas de tipo Verdadero o falso. Para su realización, se integró el uso de "gestos" (Gestures) en la pantalla táctil del teléfono.

Los gestures son movimientos pregrabados sobre la pantalla táctil que la aplicación puede reconocer. De esta forma, la aplicación podrá realizar acciones función del gesto introducido por el usuario. Los gestos se crean a partir de la aplicación del emulador "gesture builder"[19] capaz de asociar el gesto a una cadena de texto para su posterior uso. Una vez realizados los gestos, para poder utilizarlos tenemos que guardarlos en la carpeta ~/res/raw del directorio del proyecto.

En el caso de la aplicación sólo hay dos tipos de gesture, uno para representar una respuesta verdadera y otro para la respuesta falsa y para un reconocimiento óptimo se han introducido cinco muestras de cada gesto (porque cada persona escribe de forma distinta).

A continuación **(Figuras 14 y 15)** se muestran unos ejemplos de los gestos introducidos dentro de la aplicación a través de gestures builder:

Figura 15: Ejemplos 1 y 2 de Gestures Builder.

Figura 16: Ejemplos 3 y 4 de Gestures Builder.

A continuación se muestran en la **Figura 17** un ejemplo del uso de los gestos dentro de la aplicación. Una vez introducido el patrón, si es reconocido por la aplicación, es decir, está contemplado en el archivo de gestos generado con Gestures Builder, Muestra si la respuesta es correcta o no.

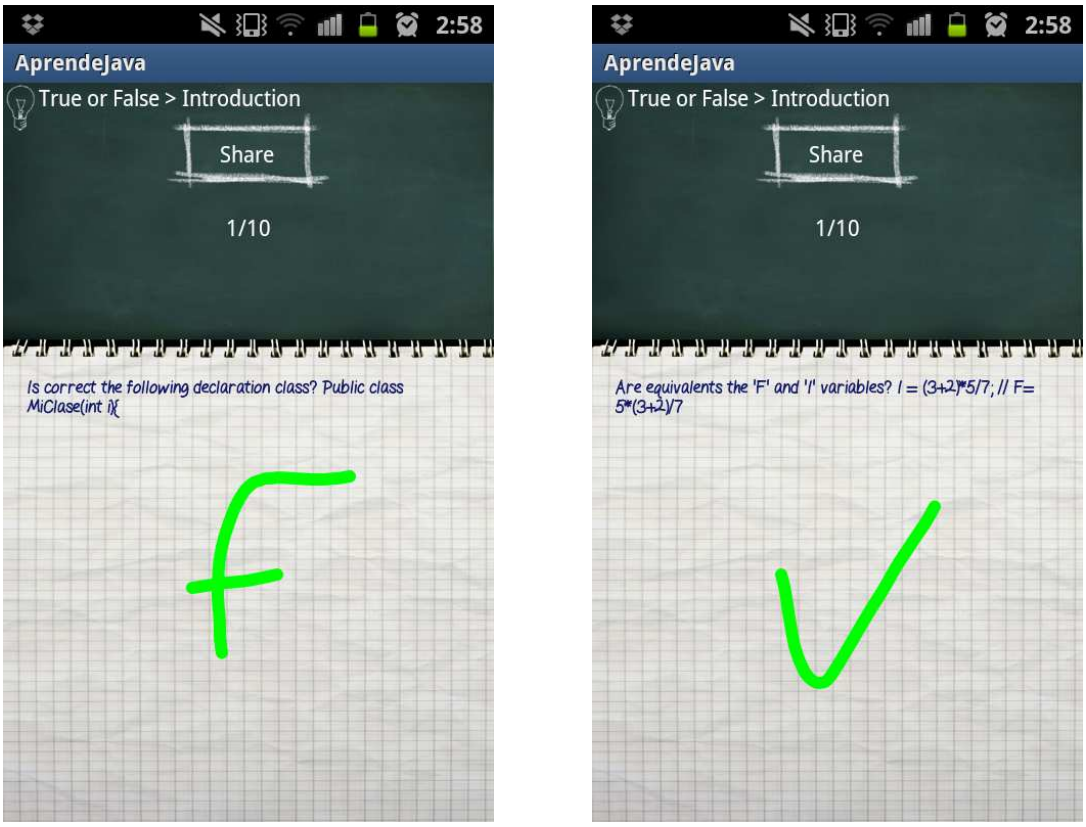


Figura 17: Ejemplo de uso Respuesta V/F.

4.3.3. Redes sociales.

Las redes sociales son plataformas que sirven para conectar personas que tienen algún tipo de relación entre sí. A día de hoy es raro encontrar todavía a alguien joven que no use alguna de estas aplicaciones de forma diaria.

En este proyecto se ha permitido compartir las preguntas en las redes sociales con la intención de que un contacto de nuestra red social pueda ayudarnos a dar con su solución. Las redes sociales que se han usado para ello son Facebook y Twitter.

Es importante entender la forma en la que cada red social conecta con su servidor e intercambia información sensible del usuario bajo un estricto control de seguridad.

En el caso de Twitter la aplicación ataca directamente a su servidor mediante una página Web embebida **(Figura 18)** dentro de un Activity (WebActivity) ,y en la que se carga una página para iniciar sesion. El servidor, cuando el usuario introduce sus datos de inicio de sesión, devuelve dos tokens que se guardarán en un archivo que está oculto para el resto de aplicaciones del móvil, creando así, una relación entre esos tokens y la aplicación. Estos tokens sirven para que en las posteriores conexiones no se vuelva a pedir otra vez los datos. Y, aunque pudiésemos acceder a esos tokens, no podríamos hacer nada ya que solo funcionan si se conecta con ellos a través de la aplicación.

Figura 18: Conexión con Twitter.

El resultado de compartir la pregunta en la red social Twitter se muestra en la siguiente Figura.

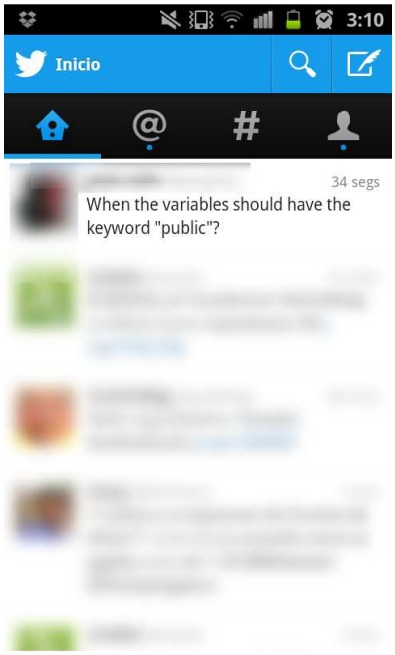


Figura 19: Resultado Compartir Twitter.

Por otra parte, Facebook, primero analiza la memoria del terminal para ver si encuentra la aplicación oficial instalada. En caso de estar en la memoria, le pediría los

datos de sesión a la aplicación y de esta forma no habría necesidad de que el usuario intervenga (en caso de no ser así sería muy parecida a twitter). Otro aspecto que hay que tener en cuenta es que las políticas de facebook obligan a las aplicaciones a dejarle el control total de lo que se comparte al usuario, evitando así publicar datos comprometidos o comportamientos no deseados. En la **Figura 20** se muestra como el usuario puede modificar los datos que se van a compartirá la hora de pedir ayuda en una pregunta. Y en la **Figura 21** se muestra el muro de facebook después de compartir la pregunta

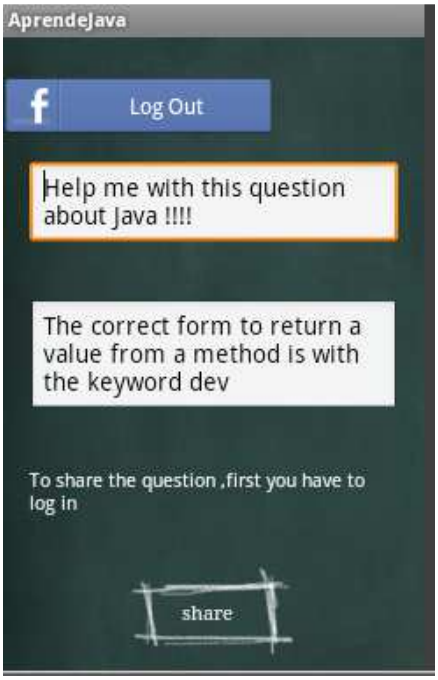


Figura 20: Compartir Facebook.

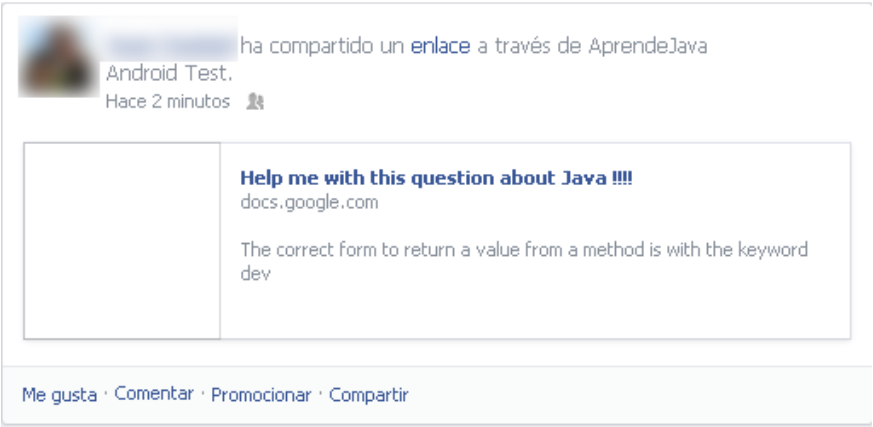


Figura 21: Resultado Compartir Facebook.

4.3.4. Imágenes de apoyo.

El sistema de preguntas necesita tener código (en este caso Java) para poder ampliar el número de preguntas posibles (ya que de otra forma sólo se podría preguntar sobre definiciones o teoría). Este problema se podría haber resuelto introduciendo el código directamente en el fichero XML, pero, o la pregunta se visualizaba sin formato (el propio del código) o la estructura del fichero XML perdía su legibilidad. Es por ello que se optó por meter imágenes dentro de los recursos del sistema, que mostraran este código java a incluir en la pregunta, y así poder recuperarlas en un momento determinado. Para asociar una imagen y una pregunta sólo es necesario escribir el nombre de la imagen en el campo "imagen" de la pregunta (dentro del fichero XML correspondiente).

Las imágenes tienen el formato png (*portable network graphics*) es un formato de archivos de gráficos de mapa de bits basado en una compresión sin perdidas. Este formato mejora en compresión respecto a JPG y GIF (la imagen que más ocupa actualmente dentro del sistema son 11KB) y pierde menos calidad en imágenes pequeñas (en este caso las imágenes contienen pocos colores y se puede reducir un poco más con una buena configuración).

También cabe destacar que cada imagen puede tener un tamaño diferente al resto y que cada una se ve de forma óptima a una determinada escala, es por eso que no se redimensionan a una misma escala las fotos, sino que se deja al usuario el control de la escala mediante el "pinch open" y "pinch close" o lo que es lo mismo pinchando con dos dedos en la pantalla y alejándolos (para ampliar) o acercándolos (para acercar)**(Figura 22)**.

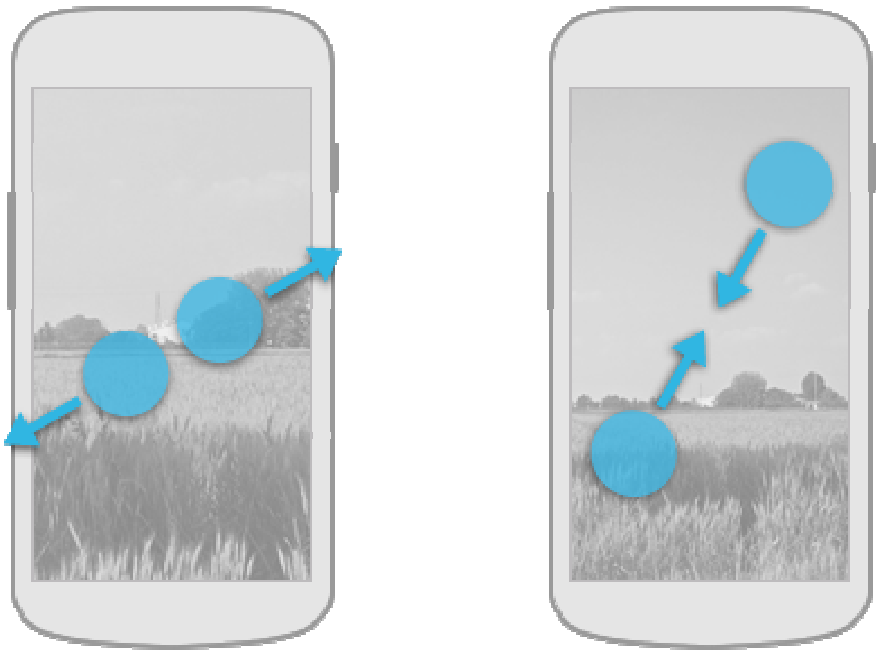


Figura 22 Izq. Pinch Open. Dcha Pinch Close.

La figura que se muestra a continuación **(Figura 23)** representa un ejemplo real de lo que se puede hacer con las imágenes. Se observa cómo a partir de la figura de la izquierda se ha conseguido agrandar la imagen hasta visualizarla como se observa en la parte derecha

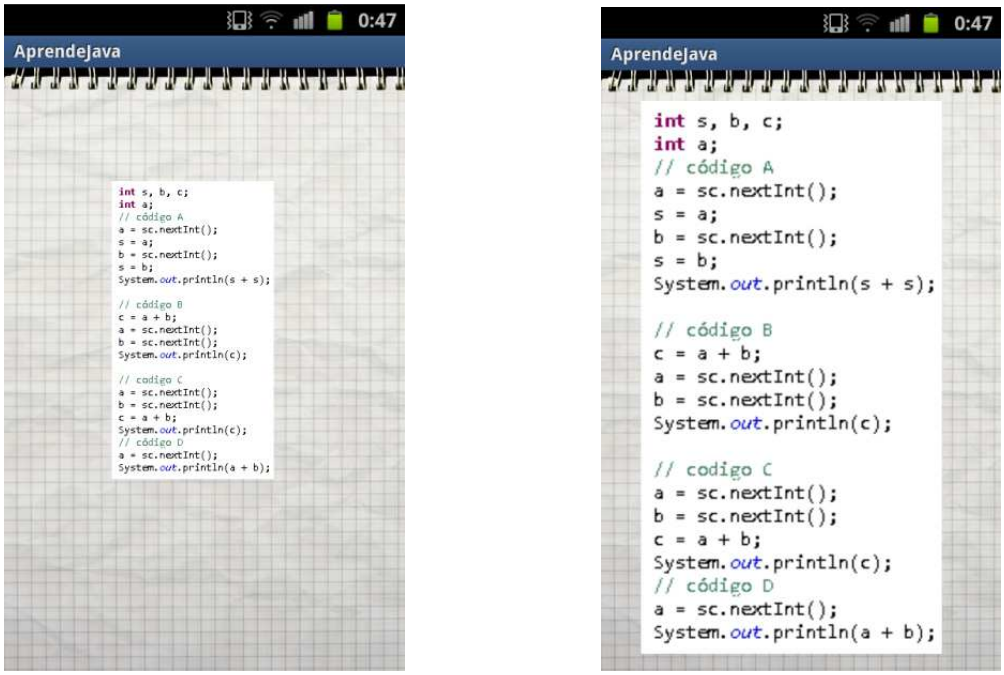


Figura 23: Ejemplo redimensión Imagen.

CAPÍTULO 5. Modo de uso.

A continuación se explicará cómo se usa la aplicación desde la posición del alumno y se mostrarán las pantallas (tal y como se dijo en el apartado de diseño).

- **Pantalla principal: (Figura 24)** Es el primer activity que se abre y sirve esencialmente para elegir que es lo que queremos hacer.

1) **Menús desplegables (Spinners):** Sirven para seleccionar el tema a estudiar y el tipo de preguntas que queremos usar. (Si se elige alguno de los exámenes o las preguntas falladas no es necesario elegir tipo).
2) **Botón continuar** Una vez seleccionados el tipo y el tema usamos el botón continuar para empezar la actividad.
3) **Botón estadísticas:** En este apartado podemos ver las estadísticas del usuario.
4) **Botón salir**: Con este botón saldremos de la aplicación.
5) **Botón opciones:** Sirve para abrir las opciones y cambiarlas.

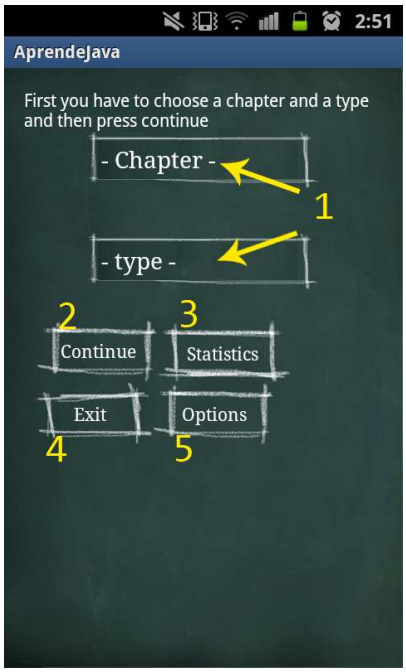


Figura 24: Ejemplo interfaz principal.

- **Pantalla de opciones: (Figura 25)** Sirve para cambiar el idioma de la aplicación y para activar y desactivar el control de vibración (En caso que el alumno falle la pregunta).

1) **checkBox de vibración:** Activa o desactiva el control de la vibración del terminal.
2) **ChecckBox alerta:** Activa o desactiva las instrucciones de Verdadero o falso.
3) **Toggle Button de idioma:** Es un botón biestado que se utiliza para cambiar entre los diferentes idiomas (en este caso español e inglés).

4) **Botón de guardado:** Guarda las opciones establecidas en los apartados 1 y 2.

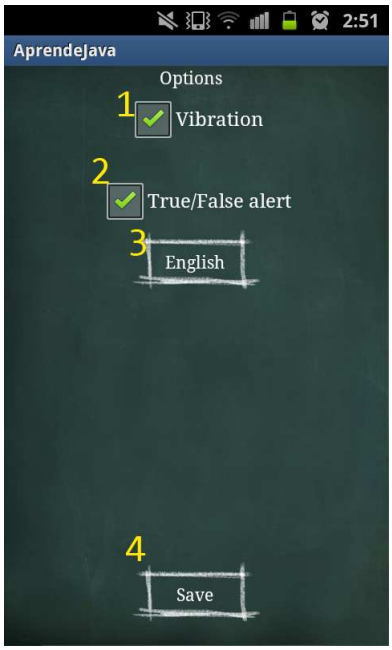


Figura 25: Ejemplo interfaz Opciones.

- **Selección de tema y tipo: (Figura 26)** como

se ha mencionado anteriormente, sólo en los casos que corresponden con cada tema (se excluyen exámenes, preguntas falladas y formularios) es necesario seleccionar un tipo de pregunta.

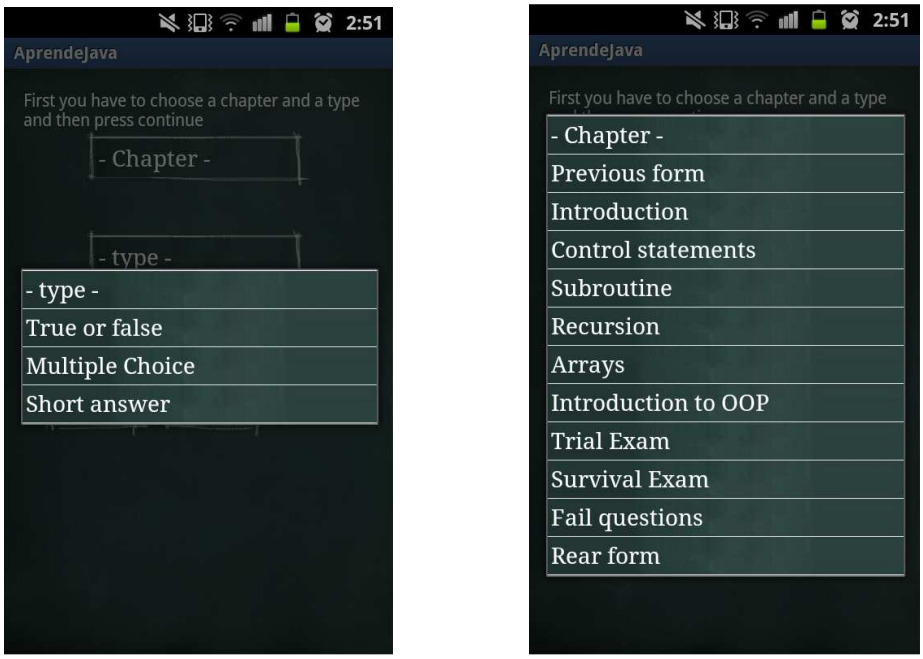


Figura 26: Izq. Seleción tipo. Dcha. Selección Tema.

- **Pantalla de estadísticas: (Figura 27)** En esta pantalla podemos consultar el resultado obtenido dentro de cada tema y tenemos la opción de resetear todas las estadísticas del sistema.

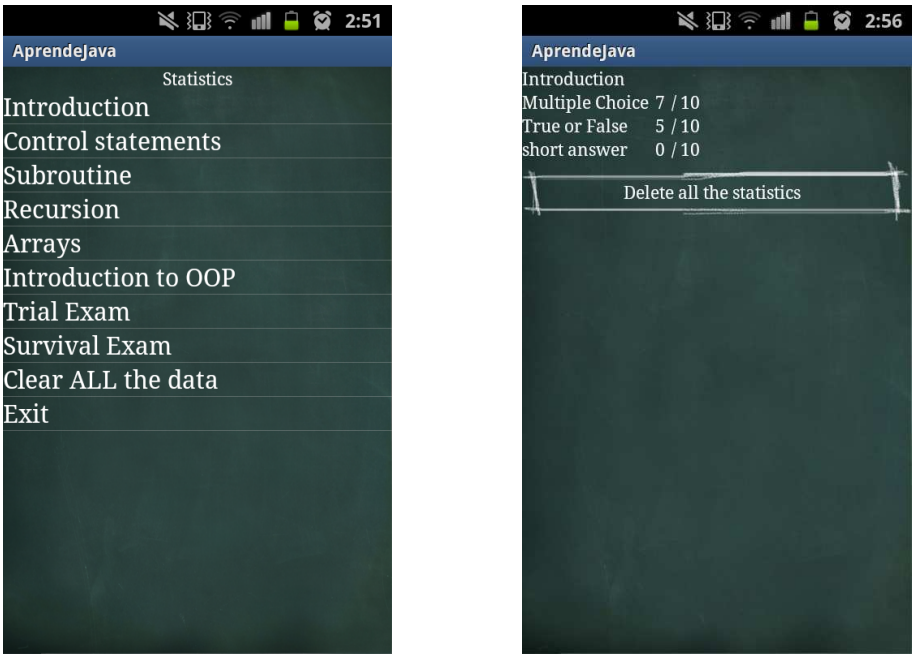


Figura 27: Ejemplo interfaz estadísticas

- **Preguntas Verdadero o falso: (Figura 28)** Como se puede apreciar, estas preguntas se responden mediante gestos (trazos en verde en forma de F).

1) **Pista:** es un botón con el que podemos interactuar, al pinchar nos redirige a una página de internet con documentación.

2) **Botón Compartir:** Pinchando en este botón compartimos los datos en una de las redes sociales.

3) **Progreso:** muestra la pregunta por la que vamos y las que hay en ese tema.

4) **Pregunta:** Al principio de lo que sería el "cuaderno" se muestra la pregunta.

5) **Zona de gestos:** Esta zona es el lugar en el que pintaremos los gestos.

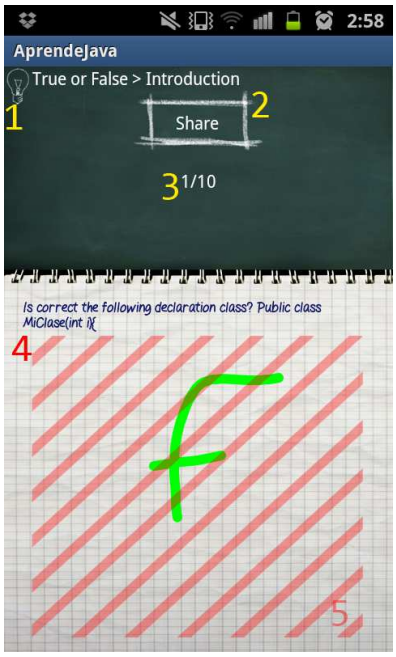


Figura 28: Interfaz Verdadero o Falso.

- **Preguntas Respuesta Corta: (Figura 29)** En este caso la pregunta se responde escribiendo la solución con el teclado.

1) **EditText:** Escribiremos la respuesta en esta área.
2) **Botón continuar:** Una vez escrita la respuesta pasaremos a la siguiente.

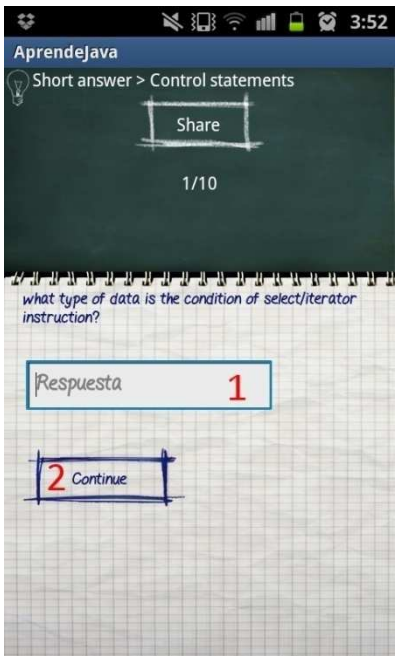


Figura 29: Ejemplo interfaz Respuesta Corta.

- **Pregunta Respuesta Múltiple: (Figura 30)** Son preguntas tipo Test.

1) **Botón Mostrar:** Accederíamos para ver la imagen con código (en caso de que haya).
2) **Respuestas posibles:** Sólo se puede seleccionar una de ellas.

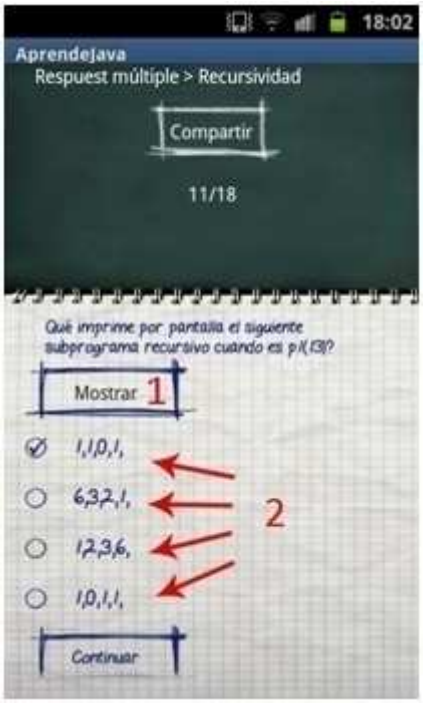


Figura 30: Ejemplo interfaz Respuesta Múltiple.

- **Examen de Resistencia, contrarreloj y preguntas falladas: (Figura 31)** los tipos de ejercicio son iguales, la única diferencia son los retos que se plantean.

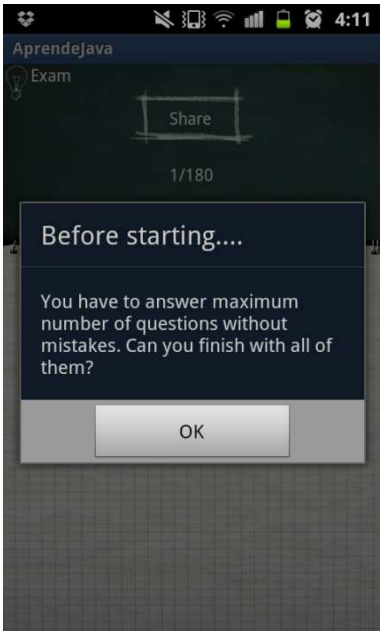


Figura 31: Izq. Contrarreloj Dcha. Resistencia

- **Las redes sociales:** Explicadas con anterioridad en el apartado 4.3.3. En twitter conecta automáticamente, así que no hace falta nada más que pulsar el botón enviar una vez nos devuelva el control la aplicación.

  En el caso de Facebook, tendríamos que iniciar sesión primero y después enviar el post. En los dos casos se accede a través del botón "share" o "compartir" que hay en cada tipo de ejercicio, el cual, nos lleva a la pantalla de selección de red social.

Figura 32: Eleccion red social.

- **Redimensión de imagen.** Explicado con anterioridad en el apartado 4.3.4. Para usarlo hay que pinchar con dos dedos en la pantalla y separarlos (como redimensionaríamos elementos en otras aplicaciones de la terminal).

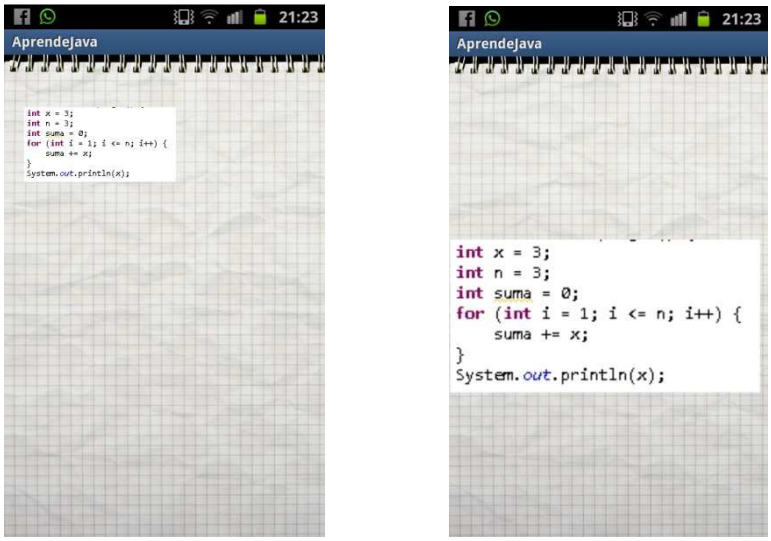


Figura 33: Redimension de imagen

CAPÍTULO 6. Gestión de la aplicación.

Desde el punto de vista del personal docente, la aplicación se gestionaría de la siguiente forma.**(Figura 34).**

Para poder cambiar los exámenes (inicial y final) y para cambiar las notas de corte es necesario ir a la clase MainActivity.java (dentro del paquete com.aprendejava).

```
MainActivity.java
1  package com.aprendejava;
2
3  import android.app.Activity;
22
23 public class MainActivity extends Activity {
24
25     private Button btnSalir, btnCont, btnEstadisticas, btnOpciones;
26     private Spinner spinTema, spinTipo;
27     private final String FORM_INICIAL =
28     private final String FORM_FINAL =
29     private boolean idiomaOn;
30     private SharedPreferences notas, preferencias;
31     private final int NOTA_DE_APROBADO_NORMAL =
32     private final int NOTA_DE_APROBADO_CONTRARRELOJ =
33     private final int NOTA_DE_APROBADO_RESISTENCIA =
34
```

Figura 34: MainActivity.java

Y al principio, cambiar los valores de esas constantes. Es **MUY IMPORTANTE** que las url se construyan sin " http:// "por ejemplo: Si quisiésemos conectar con Google, tendríamos que escribir www.google.es pero no http://www.google.es ya que se incluye de forma automática.

Para introducir las imágenes asociadas a las preguntas, hay que importarlas (basta con arrastrarlas) a la carpeta drawable (cualquiera de ellas) dentro del directorio ~/res/.**(Figura 35).**

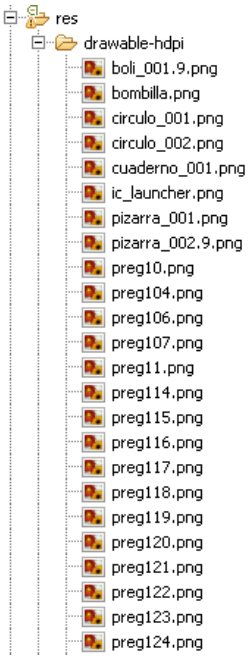


Figura 35:Carpeta Drawable.

En el caso de querer cambiar las preguntas, es necesario que nos dirijamos a la carpeta ~/res/raw/ en el que se encuentran todas las preguntas en formato xml.

Se pueden cambiar a mano uno a uno todos los campos, pero la forma más rápida de cambiarlas es siguiendo estos pasos (ejemplo sencillo incluido):

1) Disponemos los datos en un documento Excel **(Figura 36)** : De esta forma obtenemos una estructura mejor formada.

Nombre	Apellido 1	Apellido 2	Edad
Pepe	Martín	de la Rosa	25
Marina	Bascé	Ferrero	19
Juan	Nerón	Clavo	33
Dolores	Landa	Merindo	50

Figura 36: Tabla excel

En este caso, por simplificar el ejemplo y que sea legible, vamos a usar personas ficticias para representarlo.

2) Abrimos un editor de texto como PsPad (no sirve el bloc de notas de Windows ni wordpad ya que no usan expresiones regulares) y copiamos el contenido del Excel sin hacer ninguna modificación **(Figura 37)**.

```
<Nombre>$1</Nombre><Apellido1>$2</Apellido1><Apellido2>$3</Apellido2><Edad>$4</Edad>


Pepe  Martín  de la Rosa  25
Marina  Bascé Ferrero 19
Juan  Nerón Clavo 33
Dolores Landa Merindo 50
```

Figura 37: Vista PsPad

La cadena que está en la primera línea se corresponde con el patrón que vamos a reemplazar y cada $ es una variable que representa a cada elemento que vamos encontrar con la expresión regular. En el caso de ser $1, será el primer elemento, y en caso de ser $4 será el 4º.

3) Usamos el comando reemplazar y marcamos la opción "usar expresiones regulares" **(Figura 38).**

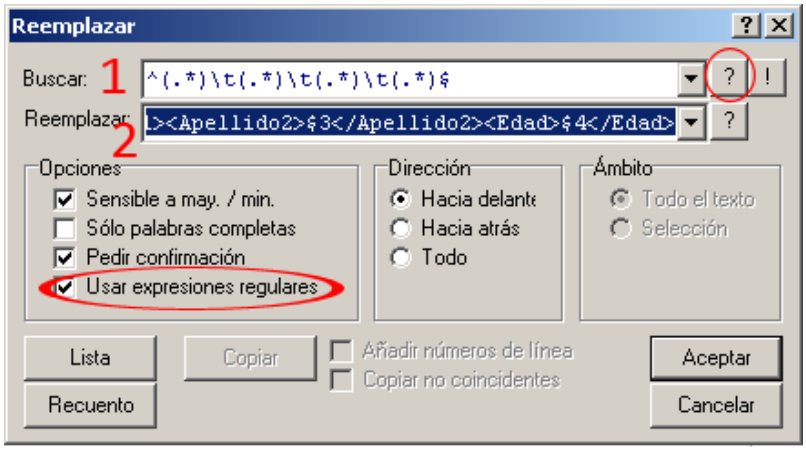


Figura 38: Vista comando reemplazar.

En este momento tenemos que construir la expresión regular en el campo Buscar (1) y reemplazar por el patrón (2) que se mostraba en la anterior ilustración.(Con la interrogación podemos ver una ayuda con los comandos).

- ^ es el inicio de línea.
- **(.*)** significa que busca cualquier cosa (en nuestro caso es texto).
- **\t** significa que tiene que encontrar una tabulación (al realizar el paso 2 se crea automáticamente).

Cómo tenemos 4 elementos tenemos que repetir los comandos hasta llegar a todos los elementos que tenemos que usar.

- **$** En el caso de la expresión regular es un fin de línea.

Una vez que esté todo correcto pulsamos aceptar.

4) Reemplazamos y comprobamos.**(Figura 39).**

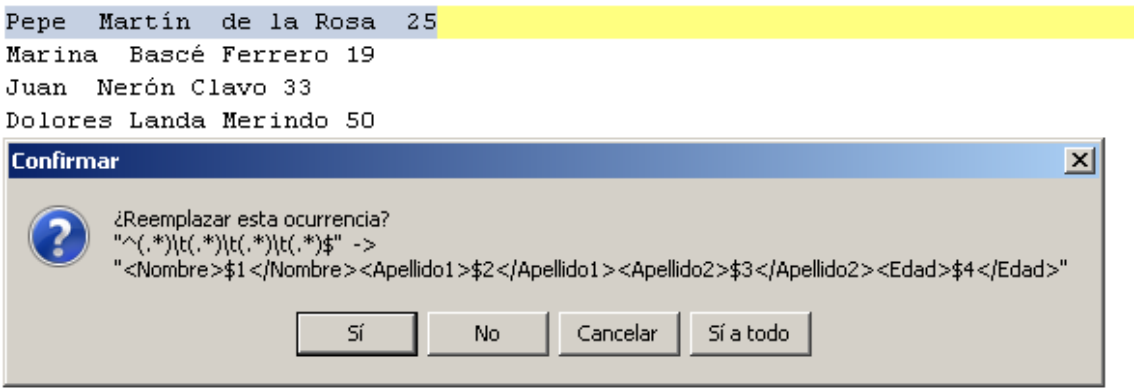


Figura 39: Ejecución comando reemplazar.

Si no estamos seguros de si lo que estamos haciendo es correcto o no, lo mejor es ir poco a poco viendo como se transforman las líneas **(Figura 40)**.

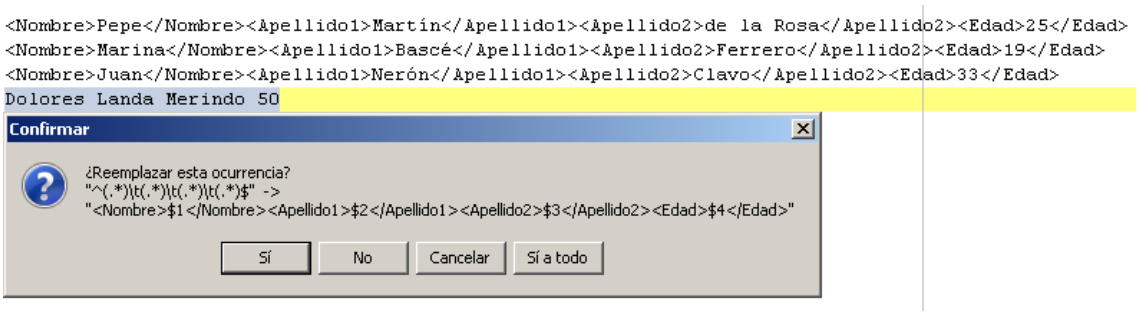


Figura 40: Vista reemplazo.

Y a partir de aquí, ya es copiarlas dentro de los ficheros XML que se han mencionado anteriormente.

Cabe mencionar que, dependiendo del procesador de texto que se use, los comandos pueden variar. No obstante, suele estar bien explicado en los archivos ReadME.

CAPÍTULO 7. Conclusión y trabajo futuro.

7.1. Conclusiones.

Se ha conseguido crear una aplicación entretenida a la vez que didáctica que cumple con todos los objetivos que nos propusimos:

- La realización de exámenes externos se ha conseguido conectando a través de internet con exámenes creados en Moodle.
- Realización de ejercicios de repaso y de ejercicios "bajo presión": desarrollado mediante el uso de la programación en Java y las características que nos ofrece el SDK de Android.
- Repaso de las preguntas falladas: Desarrollado con las técnicas ofrecidas por android para el uso de bases de datos inherentes al programa (SQLite).
- Posibilidad de consultar teoría a través de internet y particularizando la teoría para cada pregunta.
- Posibilidad de cambiar el idioma de forma manual mediante el uso de preferencias compartidas.
- Posibilidad de consultar las estadísticas mediante el uso de SharedPreferences.
- Compartir conocimiento a través de las redes sociales: Twitter y Facebook Desarrollado con sus respectivas librerías.
- Posibilidad de acompañar las preguntas con imágenes de apoyo mediante el uso de las librerías que ofrece Android.
- Y todo ello con una interfaz sencilla y amigable manipulando ficheros XML para su composición.
- Alimentación de la aplicación.

7.2. Trabajos futuros.

A continuación se expone una posible evolución dentro de la vida útil del software desarrollado:

- En lo que se refiere a la aplicación en sí, se pueden inventar nuevos tipos de preguntas para una mayor interactividad del usuario con la aplicación.
- Las preguntas podrían guardarse en una base de datos externa (si ya se dispone de un servidor gestionado).
- La base de datos externa mencionada en el punto anterior podría estar conectada con Moodle 2.0 (ya que ofrece servicios web) para que a la vez que recibe las preguntas, pueda guardar el progreso del usuario y las notas del usuario.
- Añadir nuevos idiomas.

BIBLIOGRAFÍA.

- Meier, Reto. Professional android application development. Wiley,2009.
- Murphy, Mark L. Beginning android 2. Apress. 2010.
- Tomás Gironés, Jesús. El gran libro de android. Marcombo 2013.
- Desconocido, Javaya http://www.javaya.com.ar/androidya/
- Página de desarrolladorres de Android http://developer.android.com/index.html
- Página de desarrolladores de Facebook. https://developers.facebook.com/
- Páginas de desarrolladores de Twitter. https://dev.twitter.com/
- Desconicido. Vidas Concurrentes. http://blog.vidasconcurrentes.com/
- Salvador Gómez. Sgoliver. http://www.sgoliver.net/
- Universidad Politécnica de Valencia http://www.androidcurso.com/index.php/recursos-didacticos/tutoriales-android/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android

REFERENCIAS.

- [1] Smartphones https://es.wikipedia.org/wiki/Tel%C3%A9fono_inteligente
- [2] E-learning http://es.wikipedia.org/wiki/Aprendizaje_electr%C3%B3nico
- [3] Android http://www.android.com/about/
- [4] Play Store https://play.google.com/store?hl=es&tab=w8
- [5] Ingeniería del SW. http://definicion.de/ingenieria-de-software/
- [6] Modelo incremental http://www.slideshare.net/AvelinoFelipePolicarpio/modelo-incremental-12729581
- [7] Dalvik VM http://es.wikipedia.org/wiki/Dalvik
- [8] Open Handset Alliance http://www.openhandsetalliance.com/
- [9] Arquitectura Android http://androideity.com/2011/07/04/arquitectura-de-android/
- [10] MVC http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html
- [11] SDK http://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software
- [12] SAX http://www.saxproject.org/
- [13] Twitter4J http://twitter4j.org/en/index.html
- [14] SDK Facebook https://developers.facebook.com/android/

- [15] Draw9-patch http://developer.android.com/tools/help/draw9patch.html
- [16] SQLite3 http://developer.android.com/tools/help/sqlite3.html
- [17] Plugin ADT http://developer.android.com/tools/sdk/eclipse-adt.html
- [18] DOM http://www.w3schools.com/dom/
- [19] Gestures Builder http://android-developers.blogspot.com.es/2009/10/gestures-on-android-16.html