ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERO EN INFORMÁTICA (Plan 97)

GESTOR AUXILIAR CLÍNICA DENTAL EN ANDROID

Realizado por

ALBERTO ALONSO CONESA

Dirigido por

JOSÉ RAMÓN PORTILLO FERNÁNDEZ

Departamento

MATEMÁTICA APLICACA I

RESUMEN

Este proyecto fin de carrera tiene como objetivo desarrollar una aplicación para gestión y administración de clínicas dentales. Mediante ella, el usuario podrá tener actualizada su agenda de citas, la bandeja de entrada de emails de consulta, la lista de contactos, el libro de cuentas diario, etc. sin necesidad de encontrarse físicamente en la clínica. Todo esto podrá hacerlo mediante un dispositivo móvil, ya sea teléfono o tablet, basados en Android y con conexión a Internet (3G, wifi…).

Para su implementación, se ha usado el lenguaje de programación Java, bajo el entorno Eclipse (Helios), con las correspondientes plataformas SDK, librerías de Google, etc.

Todo el proyecto ha sido desarrollado bajo la idea de que el usuario de la aplicación, pueda hacer uso de la funcionalidad completa que ofrece sin la necesidad de salir de ella. Es decir, que todas las secciones se encuentren recogidas en el mismo programa, ofreciendo una mayor comodidad y usabilidad.

A continuación, exponemos todo lo que ha dado de sí este proyecto.

INTRODUCCIÓN

Dado que estamos en los últimos momentos de nuestra vida en la carrera, es de obligado cumplimiento la realización del proyecto. Su realización y desarrollo considero que es una parte fundamental de nuestra formación y en la que vamos a poder mostrar parte de los conocimientos adquiridos, así como, nuestra capacidad para aprender e investigar otros campos por nosotros mismos.

En primer lugar, hay que elegir la temática del proyecto, ya sea uno ofertado por los departamentos o una idea personal aprobada y consensuada con el tutor correspondiente. En mi caso fue esta última opción.

La idea consiste en una aplicación basada en Android, destinada a dispositivos móviles, entendiendo como estos a teléfonos móviles, tablets, portátiles… Intentaremos que su funcionalidad sea muy similar a la de un gestor "auxiliar" de citas para una clínica odontológica. Es decir, algo parecido a lo que las clínicas tienen como programas gestores en sus PCs corporativos. El término de auxiliar es debido a que realizará funciones secundarias y adicionales a las que ya se llevaran a cabo en la propia clínica. Cada una de ellas las explicaremos con detalle más adelante cuando nos adentremos en la sección correspondiente.

Por otra parte, queremos que esta aplicación sea usada por ejemplo, cuando ya estamos en nuestra residencia, de viaje, etc. A modo de consulta y de control del día a día del centro. Puede que alguien se pregunte porque es para una clínica odontológica. La razón es bastante simple. Debido a que en mi familia hay muchos dentistas, surgen muchas conversaciones acerca de dientes, prótesis, tratamientos, etc. Por ello, pensamos en desarrollar algo con lo que poder realizar distintas operaciones sin la necesidad de estar presentes en la clínica.

En un principio, será probada como una versión "beta" y según la aceptación que tenga, se distribuirá libremente en el market de Android, adaptándola para las últimas versiones de software que haya en el momento.

¿QUÉ ES ANDROID?

Android es un sistema operativo móvil basado en Linux, enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes (Smartphone), tabletas, Google TV y otros dispositivos. Ha sido desarrollado por la Open Handset Alliance, la cual es liderada por Google. Este sistema además maneja aplicaciones como Market o su actualización, Google Play.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6% en el tercer trimestre. A nivel mundial alcanzó una cuota de mercado del 50,9% durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de iPhone) con más cuota.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como pueden ser la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung. Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

El anuncio del sistema Android se realizó el 5 de noviembre de 2007 junto con la creación de la Open Handset Alliance, un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Google liberó la mayoría del código de Android bajo la licencia Apache, una licencia libre y de código abierto.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++.

Los componentes principales del sistema operativo de Android son:

- **Aplicaciones**: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

- **Marco de trabajo de aplicaciones**: los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

- **Bibliotecas**: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android; algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.

- **Runtime**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida "dx".

- **Núcleo Linux**: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

ARQUITECTURA DE ANDROID

El corazón de Android es el kernel Linux, donde se encuentran los drivers necesarios para el acceso al hardware, en concreto, para la gestión de la pantalla, el teclado, la cámara, la red Wi-Fi, el audio y la memoria Flash, entre otros.

En principio, el desarrollador no accederá directamente a esta capa, sino que utilizará una serie de librerías que están en un nivel superior y que nos abstraen del hardware.

Estas librerías, entre las que se incluyen la propia libc, están programadas en C. Otras librerías de este nivel son SQLite para la gestión de bases de datos, FreeType para las fuentes de texto, WebKit y SSL para la navegación web y el cifrado de comunicaciones, e incluso OpenGL ES para aprovechar la aceleración gráfica del dispositivo.

Aunque estén escritas en C, el programador accede a ellas desde una API de Java, que es el lenguaje que se usa para desarrollar en Android. Para ello, el sistema incluye una máquina virtual java (JVM). La máquina virtual que incluye Android se llama Dalvik, y ha sido creada por Google para correr en dispositivos con poca memoria y poca capacidad de proceso.

A diferencia de la JVM de SUN, Dalvik ejecuta archivos .dex en lugar de los clásicos archivos .class de Java. Los archivos .dex son más compactos y están más optimizados para el entorno del teléfono. No dispondremos de toda la API de JavaSE o JavaME, sino que se incluye un subconjunto llamado Core Libraries.

Toda la programación del dispositivo se hace usando el Framework de aplicación, que nos ofrece todo lo necesario.

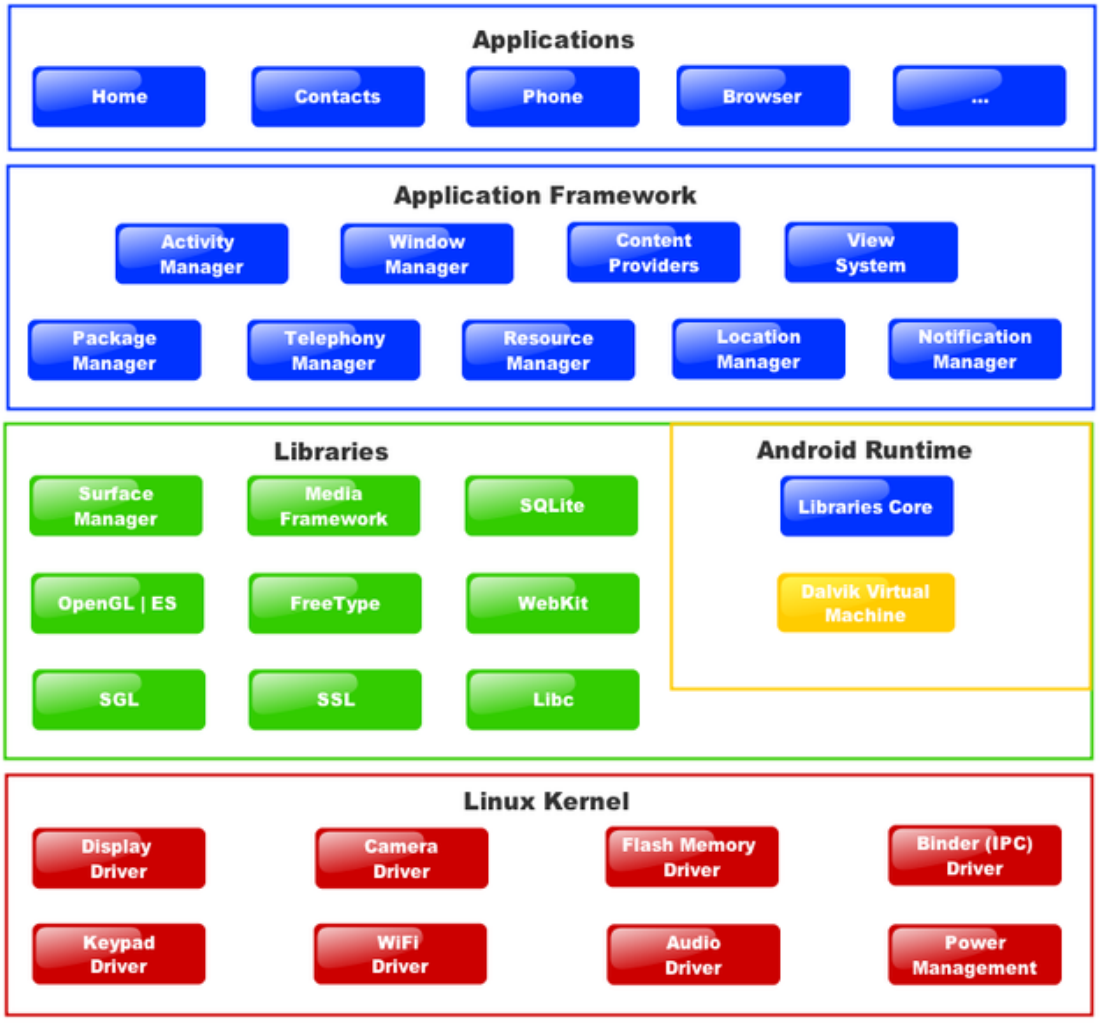


Fig. 1: Arquitectura del S.O. Android.

EL ENTORNO DE DESARROLLO

La última versión del SDK de *Android* es la 4.1. El SDK nos ofrece, además de un emulador, todas las aplicaciones y librerías que vamos a necesitar para desarrollar aplicaciones. El emulador recrea perfectamente un móvil de *Android*, por lo que, en principio, casi todo el desarrollo podremos hacerlo sin usar un terminal real.

A continuación, se exponen los pasos necesarios para su instalación y configuración:

- **Paso 1. Descarga e instalación de Eclipse.**

  Importante, descargar la versión apropiada para tu sistema operativo.

- **Paso 2. Descargar el SDK de Android.**

  Para instalar el SDK sólo hay que descomprimir el archivo que hemos descargado en un directorio y ya podremos empezar a trabajar con el SDK; sin embargo, usarlo directamente puede ser un poco engorroso.

- **Paso 3. Descargar el plugin Android para Eclipse.**

  Google pone a disposición de los desarrolladores un plugin para Eclipse llamado *Android Development Tools* (ADT), que facilita en gran medida el desarrollo de aplicaciones para la plataforma. Para descargarlo mediante las opciones de actualización de Eclipse, se accede al menú "*Help/Install new software…*" y se indica la siguiente url:

  https://dl-ssl.google.com/android/eclipse/

- **Paso 4. Configurar el plugin ADT.**

  En la ventana de configuración de Eclipse, se debe acceder a la sección de *Android* e indicar la ruta en la que se instaló el SDK.

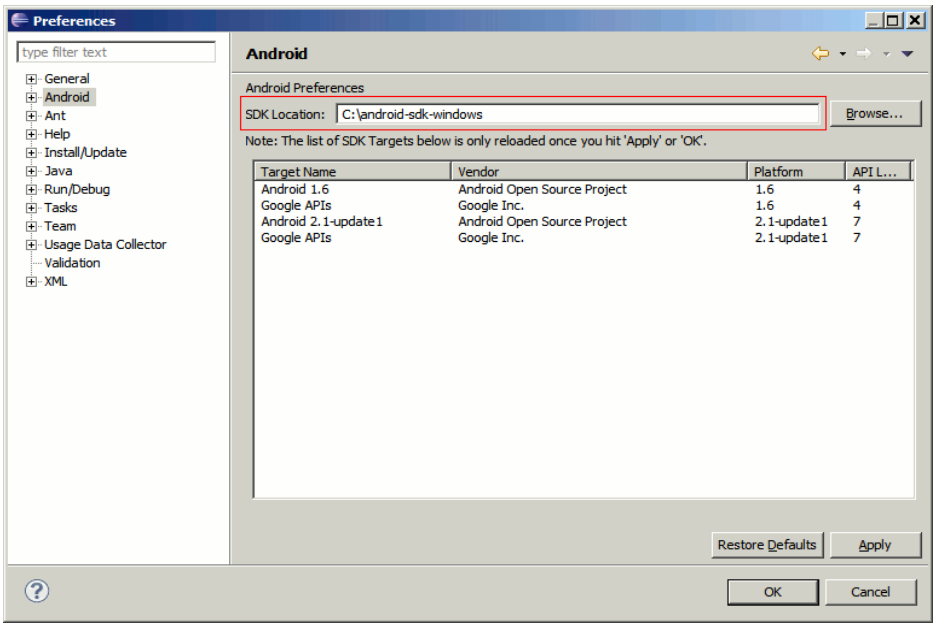


Fig. 2: Configuración ADT.

Paso 5. Descargar los targets necesarios.

También debemos descargar los llamados *SDK Targets* de *Android*, que no son más que las librerías necesarias para desarrollar en cada una de las versiones concretas de *Android*. Para ello, desde Eclipse debemos acceder al menú "*Window/Android SDK and AVD Manager*", y en la sección *Available Packages* seleccionar e instalar todos los paquetes deseados.

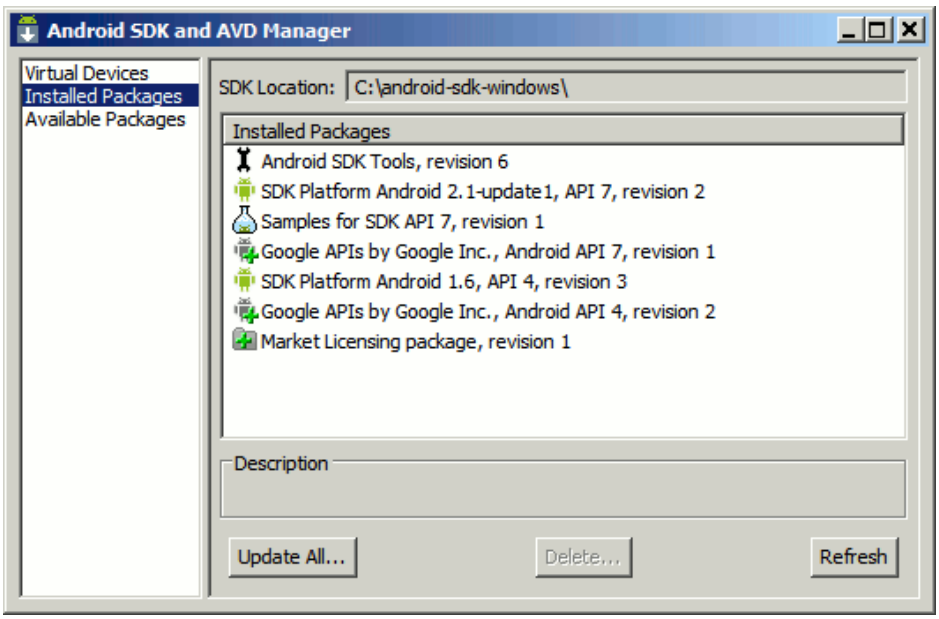


Fig. 3: Descarga targets Android.

Paso 6. Configurar un AVD.

A la hora de probar y depurar aplicaciones *Android*, no tendremos que hacerlo necesariamente sobre un dispositivo físico, sino que podremos configurar un emulador o dispositivo virtual (*Android Virtual Device, AVD*) donde poder realizar fácilmente estas tareas. Para ello, accedemos al *AVD Manager,* y en la sección *Virtual Devices* podremos añadir tantos AVD como necesitemos. Para configurarlo, sólo tendremos que indicar un nombre descriptivo, el target de *Android* que utilizará y las características de hardware del dispositivo virtual (resolución de pantalla, tamaño de la tarjeta SD, disponibilidad de GPS, etc.).

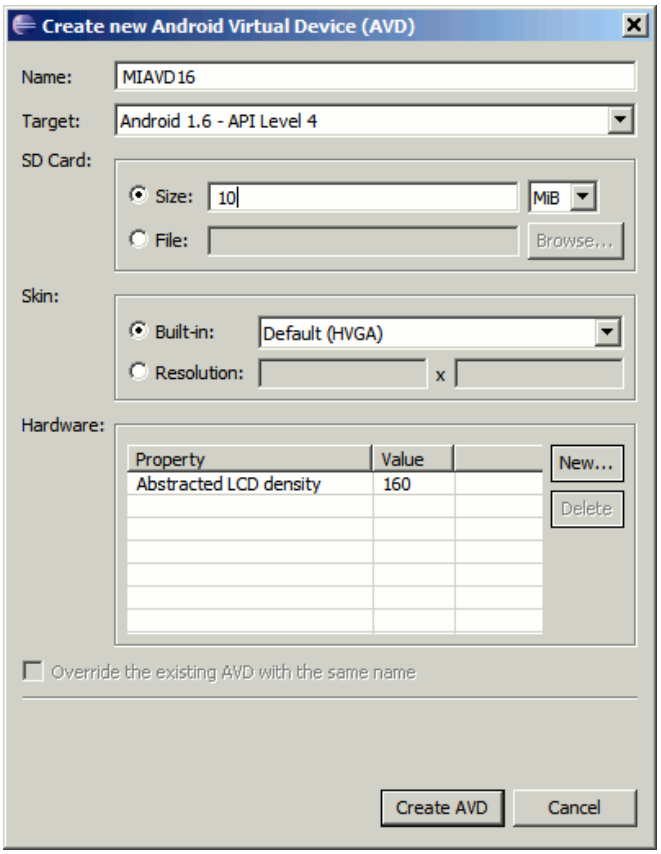


Fig. 4: Configuración emulador virtual.

Paso 7. Hola Mundo en Android.

Ya podemos crear un nuevo proyecto de tipo *Android Project.* Indicamos el nombre, target deseado, el nombre de la aplicación, el paquete Java por defecto para nuestras clases y el nombre de la clase principal (*Activity*). Esta acción creará toda la estructura de carpetas necesaria para compilar un proyecto para *Android*.

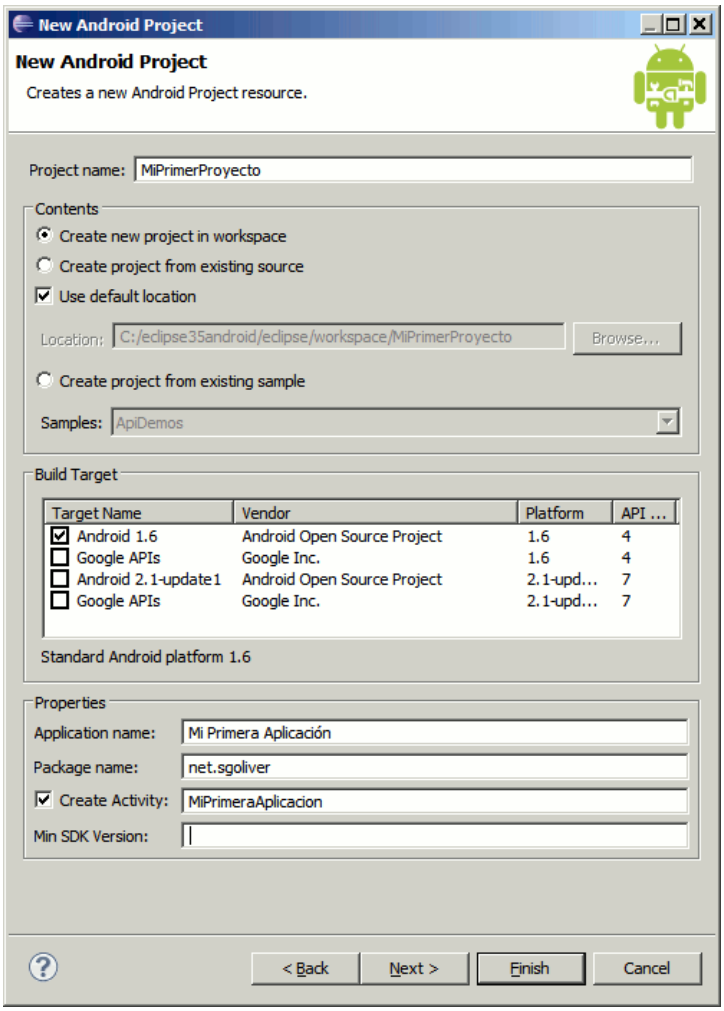


Fig. 5: Proyecto Hola Mundo.

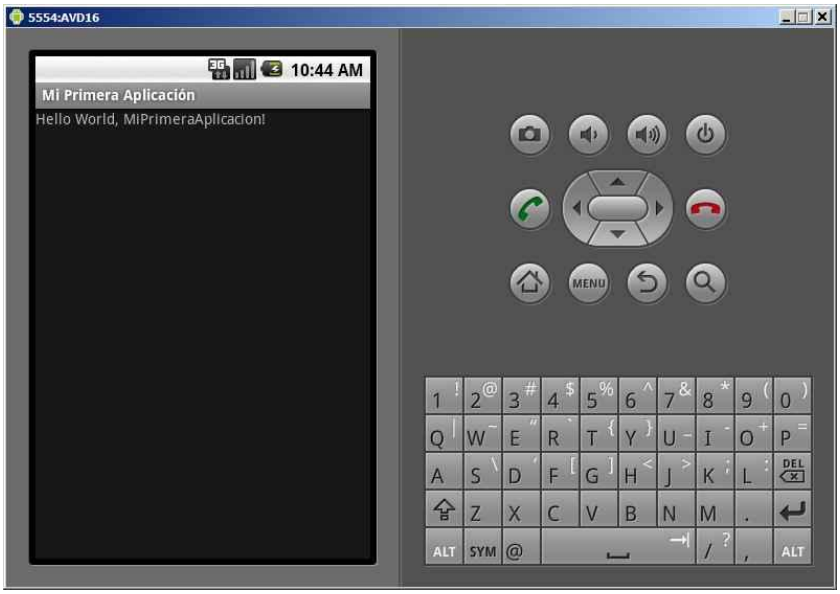


Fig. 6: Primera aplicación compilada vista en el emulador.

HISTORIAL DE ACTUALIZACIONES

Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones. Generalmente cada actualización del sistema operativo *Android* es desarrollada bajo un nombre en código de un elemento relacionado con postres. Versiones:

- ***Cupcake*** (1.5 versión), magdalena glaseada.
- ***Donut*** (1.6 versión), rosquilla.
- ***Éclair*** (2.0/2.1 versión), pastel francés.
- ***Froyo*** (2.2 versión), yogurt helado.
- ***Gingerbread***, (2.3 versión), pan de jengibre.
- ***Honeycomb*** (3.0/3.1/3.2 versión), panal.
- ***Ice Cream Sandwich*** (4.0 versión), sándwich de helado.
- ***Jelly Bean*** (4.1 versión), haba.
- ***Jelly Bean actualización*** (4.2 versión).

Fig. 7: Gráfico distintas versiones (actualizaciones) Android.

MODELO DE PROGRAMACIÓN

1) Activities.

Antes de crear nuestra primera aplicación *Android*, vamos a exponer algunos conceptos que habrá que tener presentes a la hora de crear aplicaciones.

Empecemos con las Actividades (*Activities*). Una Actividad se corresponde con una pantalla de la aplicación, es decir, que tendremos tantas actividades como pantallas tenga nuestro programa.

Cada Actividad es responsable de mantener su estado, de forma que puedan integrarse en el ciclo de vida de la aplicación, que es gestionado por el propio framework de la aplicación. En *Android* podremos crear las interfaces de usuario de dos formas, desde la propia actividad usando código Java, o usando un fichero XML para describirla como si fuera una página HTML.

Un concepto muy importante a la hora de ponernos a desarrollar es el ciclo de vida de las Actividades. En él, una actividad puede atravesar por 7 eventos distintos: *onCreate(), onStart(), onResume(), onStop(), onPause(), onRestart() y onDestroy()*.

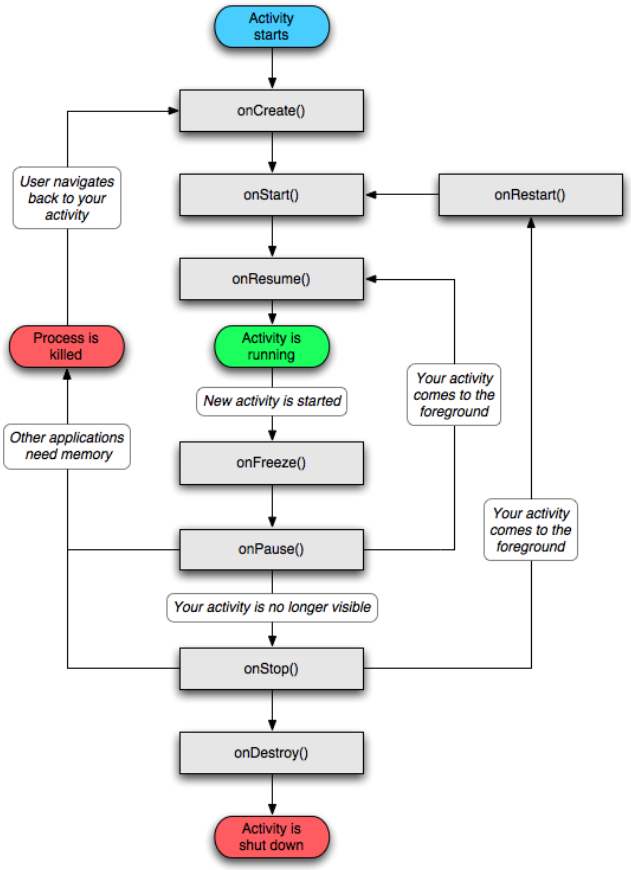


Fig. 8: Ciclo de vida, Activities.

2) Intents.

Otro elemento importante son los *Intents*. Un *Intent* es un mecanismo para describir una acción, por ejemplo, hacer una foto, enviar un email o mensajes SMS, lanzar el navegador web, etc. Todas las acciones que queramos realizar serán mediante *Intents*.

Son un concepto poderoso ya que permiten la comunicación entre cualquiera de los componentes de la aplicación instalados en el dispositivo.

Un objeto *Intent* puede contener información para el receptor y de esta manera, realizar una determinada tarea. Por ejemplo, para lanzar el navegador, hacer falta conocer la URL que queremos mostrar. Por otro lado, un *Intent* también puede contener información para el sistema para que este pueda determinar bajo algunos escenarios qué elemento puede atender la solicitud de la aplicación. Existen *Intents* explícitos o implícitos.

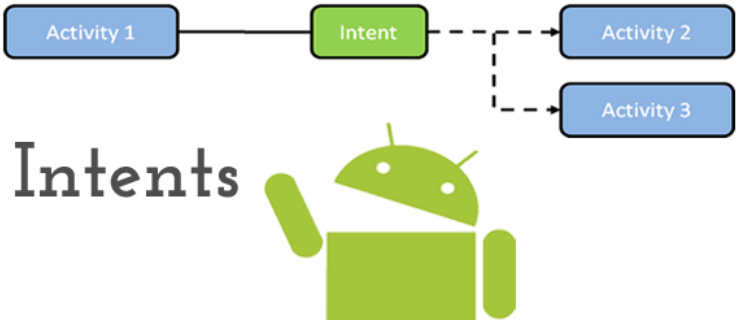


Fig. 9: Intents.

3) Services.

Los Servicios (*Services*) son tareas que corren en segundo plano, como si se tratara de un demonio Unix. Es una aplicación que corre de forma automática, sin interacción con el usuario. Desarrollan tareas importantes para el resto de las aplicaciones o para el sistema. Ejemplo de servicios son: servidores web, sistemas de comunicaciones (GPS), antivirus, etc.

Imaginemos que queremos hacer sonar un MP3, pero mientras escuchamos, queremos poder seguir usando el teléfono. Mediante un servicio, podemos dejar sonando la música de fondo mientras usamos otras aplicaciones. Una actividad puede luego acoplarse a un servicio para, por ejemplo, parar o cambiar de canción.

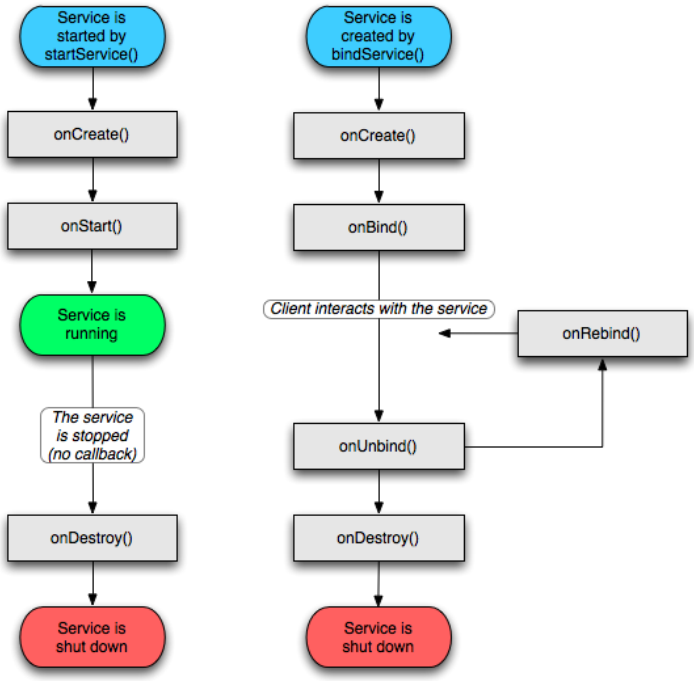


Fig. 10: Ciclo de vida, Services.

4) Contents providers.

Por último, los proveedores de contenido no son más que mecanismos proporcionados por la plataforma *Android* que almacenan datos, para ser compartidos por todas las aplicaciones. Una aplicación que desee que todo o parte de la información que almacena esté disponible de una forma controlada, deberá proporcionar un *content provider* a través del cual se accederá a dicha información. Este mecanismo es utilizado por muchas aplicaciones estándar de un dispositivo *Android*, como por ejemplo la lista de contactos, el envío de mensajes SMS, calendario o agenda, etc.

El acceso a esos datos se hace a través de la API definida para cada proveedor de contenidos.

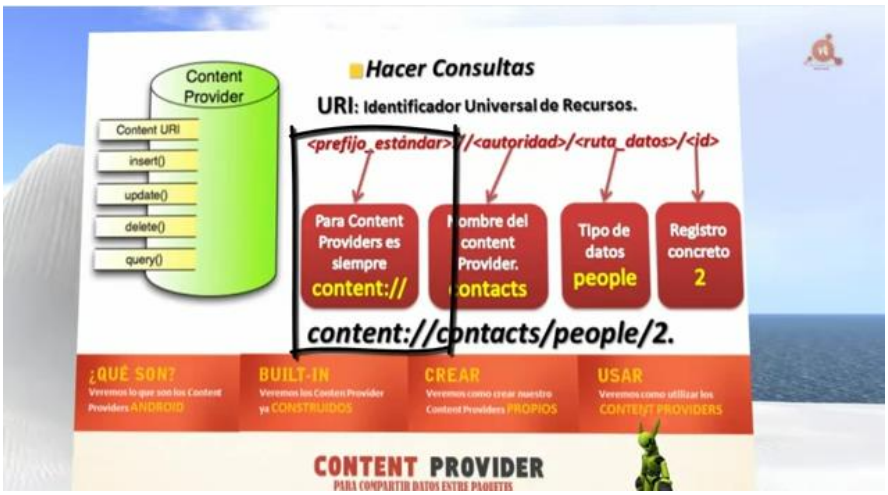


Fig. 11: Esquema Content Provider.

INTERFAZ DE USUARIO

1) Activity.

Pantalla que se muestra al usuario. Las actividades se componen de vistas.

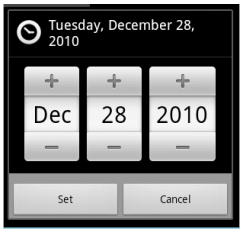
2) View.

Controles o *widgets* de la interfaz de usuario. Elemento básico de la interfaz que permite la interacción con el usuario. Se trata de un área rectangular en la pantalla que gestiona el tamaño, el dibujado, el cambio de foco y los gestos del área que representan.

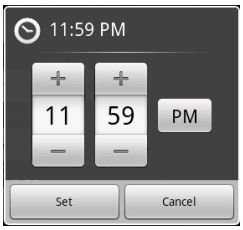
La clase *android.view.View* sirve de clase Base para todos los *widgets*.

Vistas disponibles:

- **TextView**: muestra el texto. No permite editarlo.
- **EditText**: componente de edición de texto, acepta varias líneas.
- **Button**: estándar.
- **ImageButton**: botón con imagen.
- **ToogleButton**: botón de 2 estados, on/off.
- **RadioButton**: estándar, selección múltiple.
- **Checkbox**: estándar, selección múltiple.
- **DatePicker**: cambio de fecha.



- **TimePicker**: cambio de hora.



- **Spinner**: lista desplegable de selección.

Fig. 12: Vistas múltiples.

3) ViewGroup.

Composición de vistas. Los controles complejos y el *Layout* heredan de la clase *ViewGroup*. "Existen muchas vistas y *layout´s* predefinidos aunque puedes crear los tuyos propios".

Se trata de una vista especial que contiene otras vistas hijas. *ViewGroup* es la clase base de los *layouts* y vistas contenedoras. Esta clase también define la clase *ViewGroup.LayoutParams*.
Layouts disponibles:

- **AbsoluteLayout**: dispone los elementos en coordenadas exactas.
- **TableLayout**: dispone los elementos en filas y columnas.
- **LinearLayout**: dispone de los hijos horizontal o verticalmente.
- **RelativeLayout**: dispone unos elementos relativos a otros.
- **FrameLayout**: permite cambiar dinámicamente los controles en el Layout.

ViewGroups: DatePicker, Gallery, GridView, ListView, ScrollView, Spinner, TabWidget…

Las vistas se organizan en estructuras de árbol cuya raíz es un *ViewGroup*. Existe el método *setContentView()*, que nos permite añadir una vista determinada a una actividad.

La plataforma *Android* nos ofrece dos métodos para diseñar la interfaz: procedural (código) o la que nosotros usaremos, declarativa (XML). Con esta última podremos especificar que se quiere ver en la pantalla y no como se tiene que mostrar.

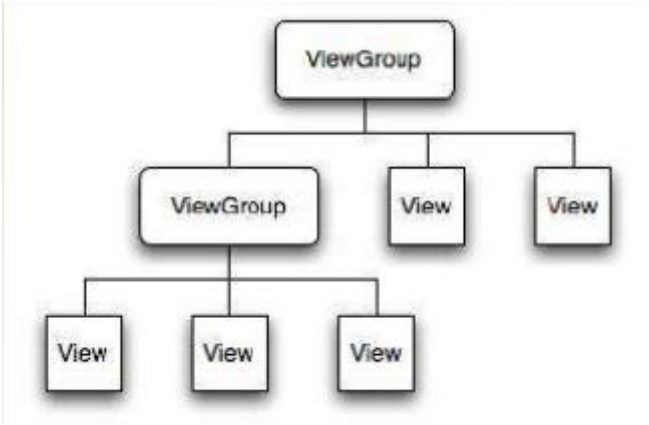


Fig. 13: Árbol jerarquía modelo de vistas.

Procedural	Archivo XML
Archivo Java	Archivo XML

TextView tv= new TextView(this)
tv.setWidth(100); tv.setHeight(60);
tv.setText("phone");
setContentView(tv);
	<TextView
	android:id="@+id/nameLabel"
	android:text="phone:"
	android:layout_width="100"
	android:layout_height="60" />

Fig. 14: Diferencia entre métodos para diseño de interfaz.

El método declarativo permite separar la presentación de la aplicación del código que controla su comportamiento. El tenerlos separados permite modificar la interfaz de la aplicación sin tocar el código fuente. Así, se podría diseñar *layouts* para diferentes orientaciones de la pantalla, diferentes tamaños de pantalla o diferentes idiomas sin tocar el código fuente.

Las vistas heredan atributos de sus clases base y definen sus propios atributos. El atributo *id* identifica a la vista dentro del árbol y permite recuperarla desde la aplicación, mediante el método *findViewByid()*. El símbolo @ indica al *parser* del *.xml* que lo que viene a continuación lo trate como un identificador de recurso. El símbolo + indica que el nombre que viene a continuación es un nombre nuevo y debe ser añadido a la clase de recursos *R.java*.

Cuando se compila la aplicación, se compila también cada archivo *.xml* de presentación y queda accesible desde la clase R generada por *Android*. Esta clase permite acceder a los recursos una vez compilados como atributos estáticos.

GOOGLE PLAY

Google Play (antiguo market) es la tienda en línea de software desarrollado por Google para dispositivos Android. La aplicación está preinstalada en la mayoría de los dispositivos Android y permite a los usuarios navegar y descargar aplicaciones publicadas por los desarrolladores de terceros.

Por otra parte, los usuarios pueden puntuar las aplicaciones e instalarlas desde almacenes de terceros (tales como Amazon Appstore o SlideME) o directamente en el dispositivo si tiene o descarga el fichero APK de la aplicación.

Fig. 15: Android Market.

El 6 de marzo de 2012, esta aplicación ha sido actualizada bajo el nombre “Google Play Store”. En enero de 2012, disponía de más de 500.000 aplicaciones para los usuarios.

La interfaz presenta un acceso fácil y rápido a sus apps. El menú tiene las siguientes opciones:

- **Mostrado**: avanza por los iconos de la parte superior para ver las aplicaciones mostradas.

- **Aplicaciones**: examina todas las aplicaciones o busca aplicaciones por categorías.
- **Juegos**: examina todos los juegos o busca juegos por categorías.
- **Búsqueda**.
- **Mis descargas**: visualiza las aplicaciones que están instaladas en el dispositivo.

Las aplicaciones móviles están especialmente diseñadas para dispositivos móviles y pueden ser gratuitas o de pago. Inicialmente, tenían una función puramente recreativa. Sin embargo, han ido evolucionando a lo que son aplicaciones para el registro de datos, información deportiva, guías de restaurantes, callejeros. Actualmente, las aplicaciones más innovadoras son las llamadas de realidad aumentada que combinan elementos reales y virtuales.

La gran novedad que aporta Google Play hace referencia a los desarrolladores: estos serán capaces de hacer su contenido disponible en un servicio abierto de Google que ofrece retroalimentación y sistema de calificación parecido a Youtube. Los desarrolladores tendrán un entorno abierto y sin obstáculos para hacer su contenido disponible. Podrá subirse al mercado después de tres pasos: registrarse como comerciante, subir y describir su contenido y publicarlo.

Google retribuye a los desarrolladores con el 70% del precio de su aplicación.

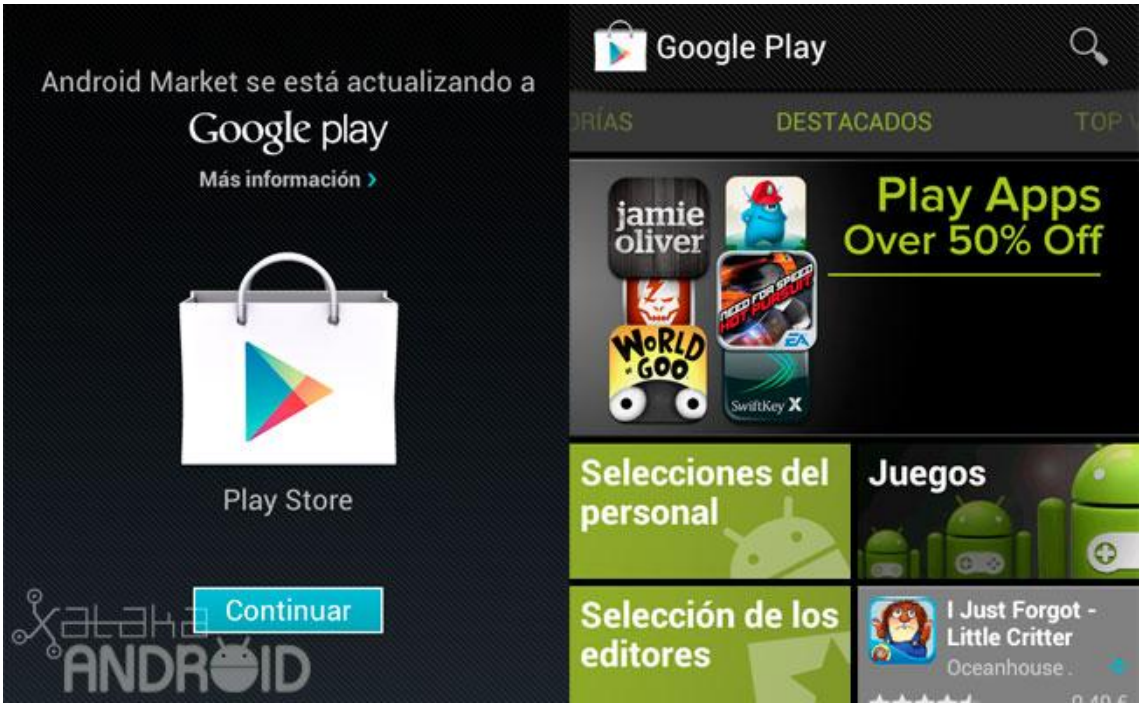


Fig. 16: Android Google Play.

OBJETIVOS

A continuación vamos a mencionar los distintos objetivos hacia los que va dirigida la realización de este proyecto.

- Realizar una aplicación para dispositivos móviles basada en Android.
- Aprender más sobre este nuevo sistema operativo, acerca de su funcionamiento, estructuras, capas, codificación, versiones, etc.
- Desarrollar en el entorno de programación Eclipse.
- Adquirir mayor experiencia en el lenguaje de programación Java.
- Configurar de manera óptima el emulador virtual para Android.
- Instalar aplicación en los dispositivos.
- Liberar versiones en el market de cara al público.
- Mantener la operatividad y portabilidad en distintos terminales.
- Aplicar e integrar los conocimientos, teóricos y técnicos, adquiridos durante la carrera de Ingeniería Informática.
- Ofrecer la propia creatividad y originalidad personal a un proyecto de estas características.

ANÁLISIS TEMPORAL Y COSTES DE DESARROLLO

La planificación efectiva de un proyecto depende del análisis detallado de su avance, anticipando problemas que puedan surgir y preparando soluciones efectivas.

Los objetivos de la estimación de proyectos son reducir los costes e incrementar los niveles de productividad y de servicio. Midiendo determinados aspectos del análisis de objetivos, requisitos y desarrollo de software, se puede tener una visión a mediana escala de lo que sucederá durante todo el proceso.

Nos encargaremos de dividir el ciclo de vida del proyecto en una serie de tareas a las cuales se asignarán las dos estimaciones siguientes:

- **Valor Estimado (E)**, en la fase inicial de planificación. Tiempo que creemos que emplearemos en terminar cada apartado del proyecto. Son los menos exactos.

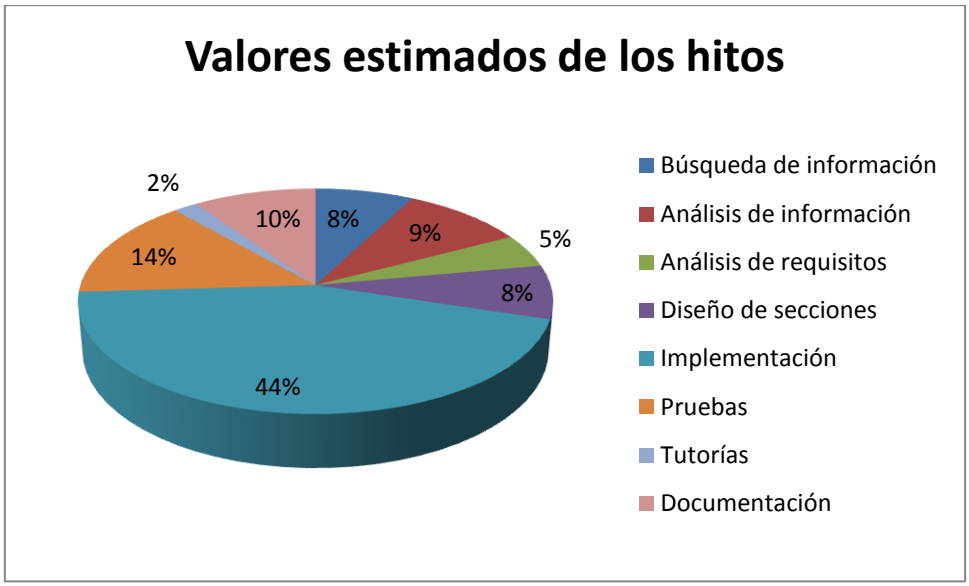
- **Valor Actual (A)**, en la fase de documentación final. Tiempo que finalmente hemos empleado en terminar cada apartado del proyecto. Dichos valores se compararán con los "estimados" para conocer la exactitud de la estimación.

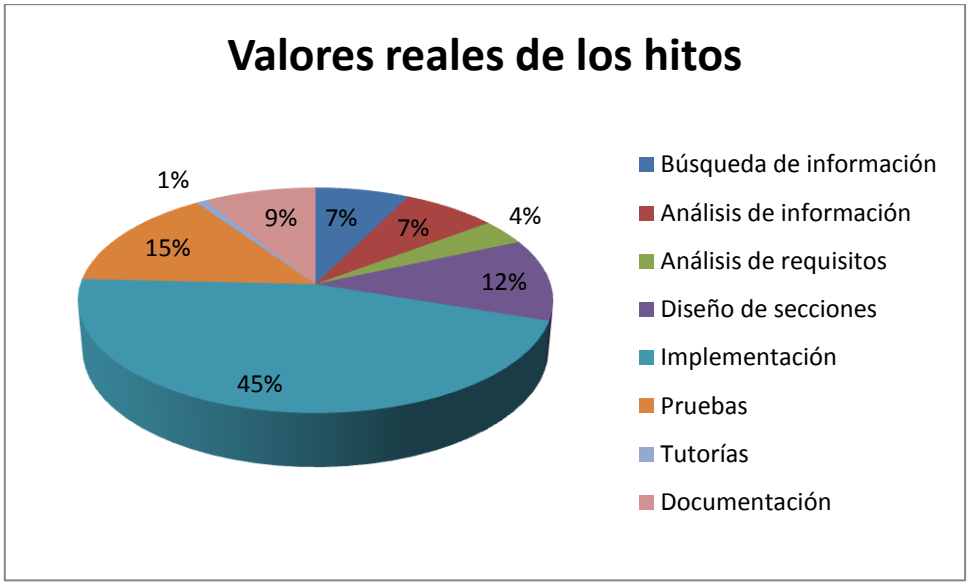
Adicionalmente, se muestra una tabla ilustrativa en la que se relacionan cada una de las tareas asignadas en la planificación con su duración estimada y actual. Tenemos que decir, que según el reglamento oficial para el Proyecto Informático de Ingeniería Superior en Informática, la suma total de horas empleadas en su elaboración debe ser de al menos 520.

TAREA	ESTIMADO (horas)	REAL (horas)
Búsqueda de información	40	50
Análisis de información	50	50
Análisis de requisitos	25	25
Diseño de secciones	40	80
Implementación	230	310
Pruebas	75	100
Tutorías	10	5
Documentación	50	60
TOTAL	520	680

Tal y como se observa, la mayoría de hitos de la evolución del proyecto ha sufrido un incremento en el número de horas de dedicación debido a las situaciones y elementos que hemos encontrado durante el camino y que se han tenido que ir solventando. En nuestra primera planificación a gran escala y esperamos que un futuro la experiencia y dedicación nos lleve a una planificación más precisa.

A continuación se muestran dos gráficos con los porcentajes de tiempo empleados en desarrollar cada una de las tareas del proyecto. Tenemos que decir, que la duración total ha pasado de las 520 horas obligatorias a 680:





En cuanto a la estimación de los costes que supondría la realización de este proyecto para una empresa dentro del mercado laboral, podemos decir que necesitaríamos del trabajo de un analista que se encargase de las tareas de análisis de información, de requisitos, diseño de secciones y documentación. Consideraremos que este ya se encuentra contratado y desempeñaría esta labor dentro de su propio horario de trabajo. Por otra parte, la empresa tendría que contratar a un programador (junior o senior) para las tareas de implementación y pruebas. Dependiendo de la categoría que tenga establecida, su salario mensual sería distinto.

De las 680 horas que dura el proyecto, 410 corresponden a las tareas del programador, el cual dedicaría las 8 horas de la jornada laboral en el proyecto, suponiéndole un esfuerzo del 100%. Aproximadamente, corresponderían a 50 días laborables, por lo que la empresa se decidiría por hacerle un contrato por obras y servicios de 3 meses.

	Salario mensual (€)	**Contrato 3 meses (€)**
Programador junior	1071,45	3214,35
Programador senior	1204,53	3613,59

Por otra parte, el coste de licencias se ve reducido a cero gracias a la decisión de optar por usar software libre en el desarrollo.

El coste total que le supondría realizar este proyecto a una empresa, suponiendo que los salarios se ajustan a los expuestos anteriormente, sería de 3214,35 euros contratando un programador junior o de 3613,59 euros, en caso de ser un programador senior.

ANÁLISIS DE REQUISITOS

1) REQUISITOS FUNCIONALES.

Los sistemas de información no sólo almacenan información, también deben proporcionar servicios usando la información que almacenan. Los requisitos funcionales ayudarán a los clientes y usuarios a responder a la pregunta "¿qué debe hacer el sistema con la información almacenada para alcanzar los objetivos de su negocio?".

- El sistema deberá implementar una sección de "calendario" para la gestión y organización de las citas de los pacientes.

- El sistema deberá implementar una sección de "bandeja de entrada" para la visualización de los correos de su cuenta asociada.

- El sistema deberá implementar una sección de "cuentas" para administrar y contabilizar el plan económico diario y mensual.

- El sistema deberá implementar una sección de "contactos" para tener información almacenada de los pacientes y poder solicitar confirmación de las citas mediante correo electrónico.

- El sistema deberá incorporar una ventana de lector de "feeds" con la que visualizar las principales noticias de páginas webs relacionadas con la odontología, por ejemplo, la web oficial del colegio de odontólogos de Sevilla.

2) REQUISITOS DE USUARIO.

- El usuario deberá disponer de algún dispositivo móvil (teléfono, tablet,…) basado en *Android* para poder disfrutar de la funcionalidad del programa.

- El usuario deberá disponer de ordenador en las instalaciones principales de la clínica con conexión a Internet.

- El usuario deberá disponer de una cuenta de correo de Google (*Gmail*) con la que poder logarse y poder hacer uso de sus servicios.

3) REQUISITOS TÉCNICOS.

Describiremos los distintos aspectos técnicos que deberá cumplir el sistema para su correcto funcionamiento.

- La aplicación deberá ser desarrollada en Java. Se trata de un lenguaje de programación orientado a objetos que implementa el nuevo sistema operativo *Android*, plataforma sobre la que montaremos nuestra aplicación.

- El sistema deberá tener una conexión 3G de datos o inalámbrica (wifi) para poder sincronizar y utilizar las principales funciones de la aplicación.

- El sistema deberá ser desarrollado para la versión 4.0 de *Android*, haciendo el código lo más universal posible para su integración y actualización en futuras versiones.

DISEÑO DE SECCIONES

A la hora de ponernos a pensar en las distintas secciones que nuestra aplicación podrá contemplar, hemos tenido que tener en cuenta el proceso de usuario para poder adaptarlo y dar funcionalidad al programa.

Como ya comentamos, queremos que pueda ser utilizada cuando la dentista ya no se encuentre en la clínica, esté de viaje u otros casos. Partimos de la idea de que vamos a hacer uso de servicios ya implementados por Google y que son de código abierto. Además, los terminales basados en Android que utilizaremos, tienen preinstaladas versiones de estos servicios, como son calendario, email, etc. Hay que tener en cuenta que tanto en el ordenador principal de la clínica y en los terminales que usemos como gestores auxiliares, tienen que estar iniciadas las sesiones para todos los servicios sobre la misma cuenta de correo de "Gmail". Para ello, crearemos una propia dentaldroid@gmail.com. Con esto lo que pretendemos es poder sincronizar los cambios que realicemos en cualquiera de ellos.

Para adaptar cada uno de los servicios ofrecidos por la aplicación, nos basamos en dividir cada una de ellos en distintas mini-aplicaciones (secciones). Entre ellas se encuentran:

- Agenda: permitirá al usuario crear y editar eventos, en este caso, los consideramos como citas para los pacientes. Podrá establecer todos los parámetros que desee y que les sea de utilidad. A su vez, dispondrá de una vista rápida a modo de resumen con los eventos creados y que hay disponibles en su calendario. Dichas citas creadas, quedarán sincronizadas con el servicio de Google Calendar de modo online, asociado a la cuenta de Gmail establecida en la sesión.

- Bandeja de entrada: sección que permitirá al usuario consultar la bandeja de entrada del correo dentro de la aplicación, sin necesidad de salir de ella. Hemos pensado que esta sección sirva como sitio de "Atención al cliente", en la que los pacientes puedan realizar sus consultas y preguntar sus dudas. El personal autorizado empleará parte de su tiempo en su mantenimiento y resolución, dando así una atención más personal a cada solicitud. Por otra parte, este buzón de entrada, también nos servirá para atender las respuestas de los pacientes a los emails de confirmación de citas que se explicarán en el siguiente apartado.

- Contactos: en este apartado, tendremos los datos de cada uno de los pacientes (contactos) y seleccionándolos, podremos enviar un email de confirmación de cita o realizar una llamada telefónica. A su vez, se añadirá un registro en el historial de emails enviados con cada correo que haya sido enviado, de manera que quede reflejado el momento y destinatario de su envío.

- Libro de cuentas: en la siguiente sección, el usuario podrá registrar su actividad diaria según el tipo de tratamiento, número y porcentaje aplicado al precio base de cada uno de ellos. Adicionalmente, se guardará un registro en el historial de cuentas, siendo posible tener una visualización con los beneficios personales de cada día de trabajo.

- Lector de feeds: último apartado en el que tendremos una vista rápida de las noticias de actualidad en el mundo de la odontología.

A continuación pasaremos a detallar cada una de ellas en el manual de usuario, explicando su funcionalidad y principales características.

MANUAL DE USUARIO

Al iniciar la aplicación, observaremos la interfaz principal con las secciones sobre las que podemos interactuar.

1) AGENDA:

Empezaremos con la sección denominada "Calendario de citas". Se trata de una agenda de eventos que se encargará de crear y registrar citas para los pacientes.

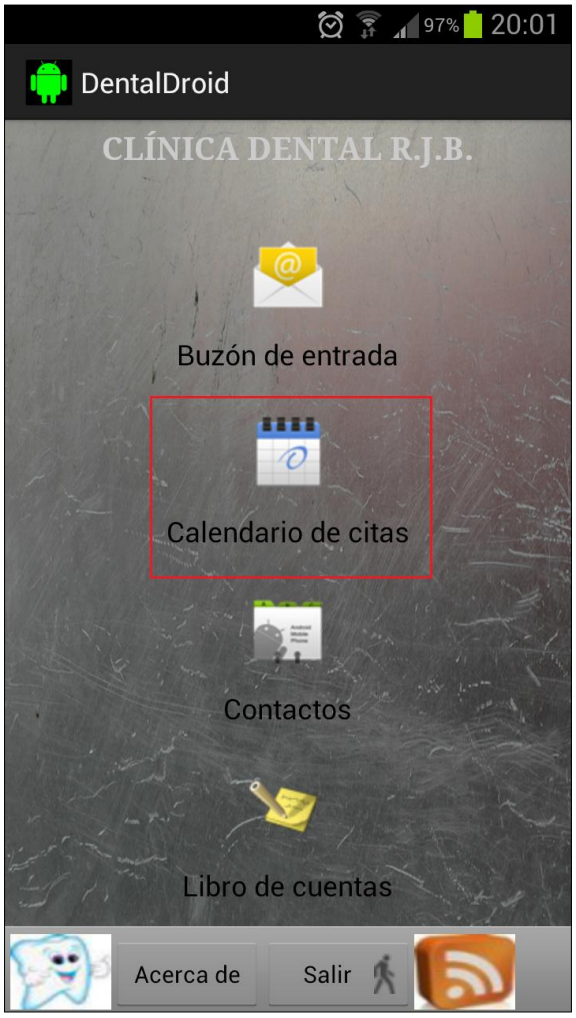


Fig. 17: interfaz principal. Botón encuadrado, sección Calendario de citas.

El usuario podrá gestionar toda la agenda de citas desde su propio domicilio o cualquier otro sitio distinto a la clínica dental. Le permitirá establecer una nueva cita, similar a la de crear un evento, en la que añadirá una serie de datos:

- Título: se indicará un término específico que resuma el objetivo de la cita.

- Día/Hora: comienzo y finalización de la cita.
- Ubicación: lugar.
- Descripción: texto en el que podremos detallar los datos del paciente así como las tareas a realizar en la cita en cuestión.

Una vez que se haya creado correctamente, el evento queda registrado en el calendario que viene instalado por defecto en nuestros dispositivos Android. A su vez, quedará sincronizado con el "Calendar" de nuestra cuenta de Gmail, que es algo que nos interesa mucho. De esta manera, se podrá consultar o editar cualquier cita establecida previamente, ya sea desde el propio dispositivo o desde el pc de la clínica en cuestión gracias a la sincronización del servicio.

Por otra parte, se ha añadido un visor de citas en el que podremos ir navegando y consultando cada uno de los eventos registrados sin tener que acceder al calendario.

A continuación, vamos a mostrar distintas capturas que reflejan el caso de uso que seguirá el usuario a través de la navegación por las distintas funciones.

En la primera imagen, se observa la interfaz principal de la sección, donde apreciamos una breve descripción de su funcionamiento, junto al botón para crear citas y el bloque del visor de eventos.

Fig. 18: vista principal al entrar en la sección.

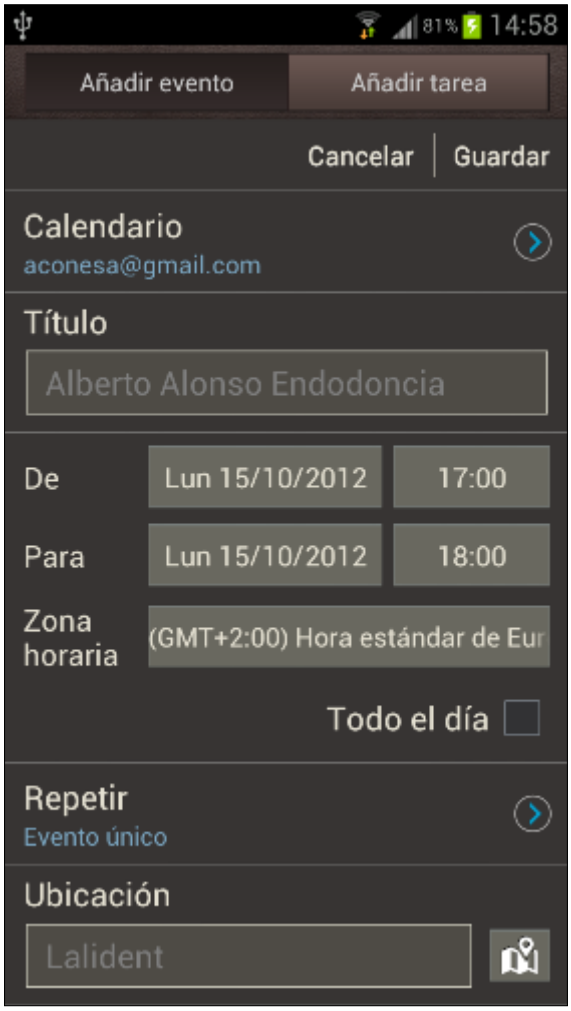


Fig. 19: proceso creación cita. Introducir datos.

Fig. 20: proceso creación cita. Introducir datos.

Fig. 21: registro en el calendario instalado.

Fig. 22: comprobación mediante la vista rápida de su creación.

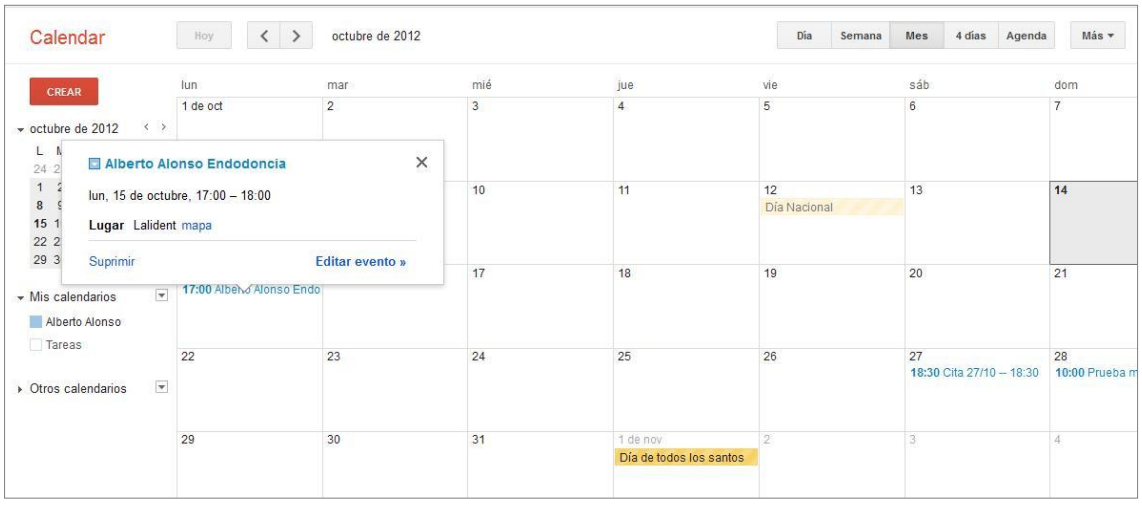


Fig. 23: comprobación en el servicio de Gmail de su actualización.

2) BANDEJA DE ENTRADA:

En primer lugar, el usuario deberá acceder a la sección, pulsando el botón seleccionado en la siguiente captura:

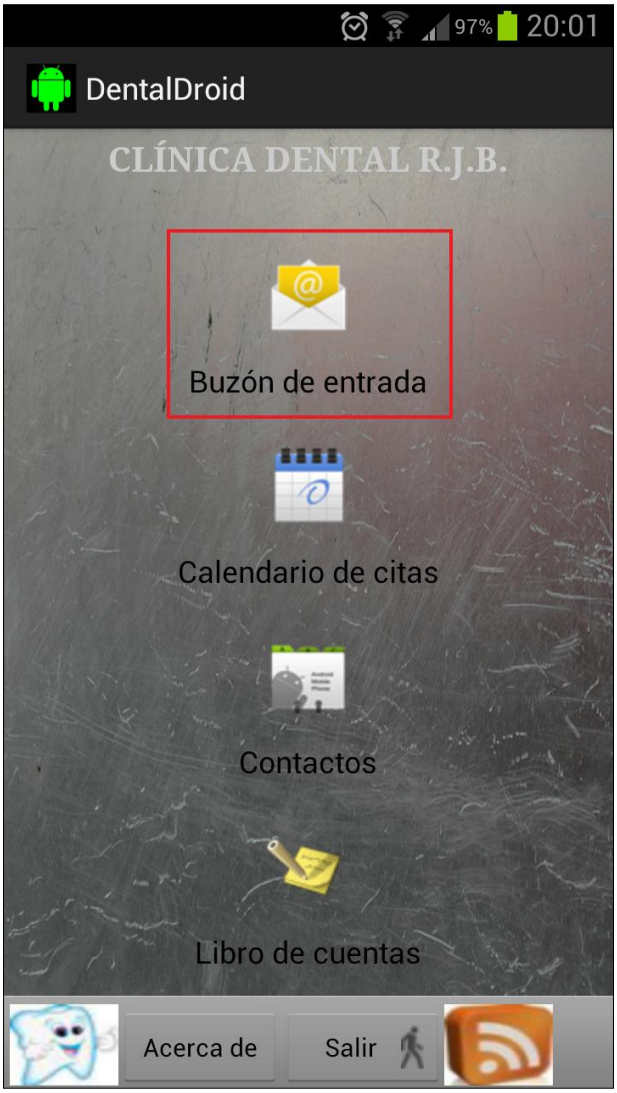


Fig. 24: interfaz principal. Botón encuadrado, sección Bandeja de entrada.

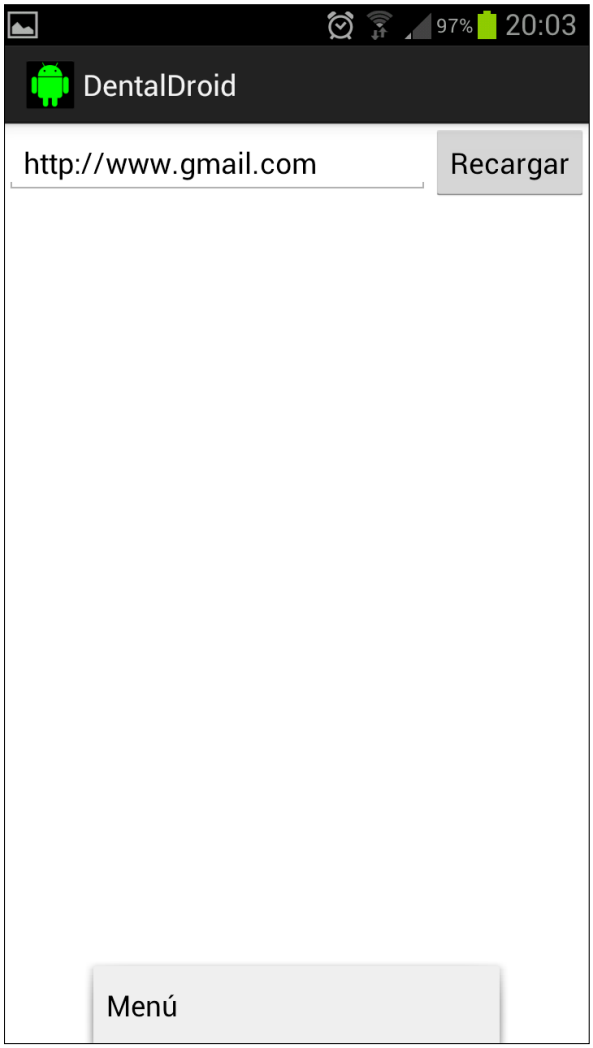


Fig. 25: vista inicial.

En la captura anterior se muestra la vista inicial al arrancar la "Bandeja de entrada". En ella podemos ver la barra de navegación con una url establecida por defecto, en este caso, el correo de Gmail. Junto a ella, se observa el botón para recargar la página. A continuación vamos a mostrar distintas capturas del proceso de usuario, en el que la persona indicada deberá logarse con el usuario&contraseña de la cuenta propiedad de la clínica. Una vez iniciada la sesión, tendremos a nuestra disposición todos los servicios que de la bandeja de correo.

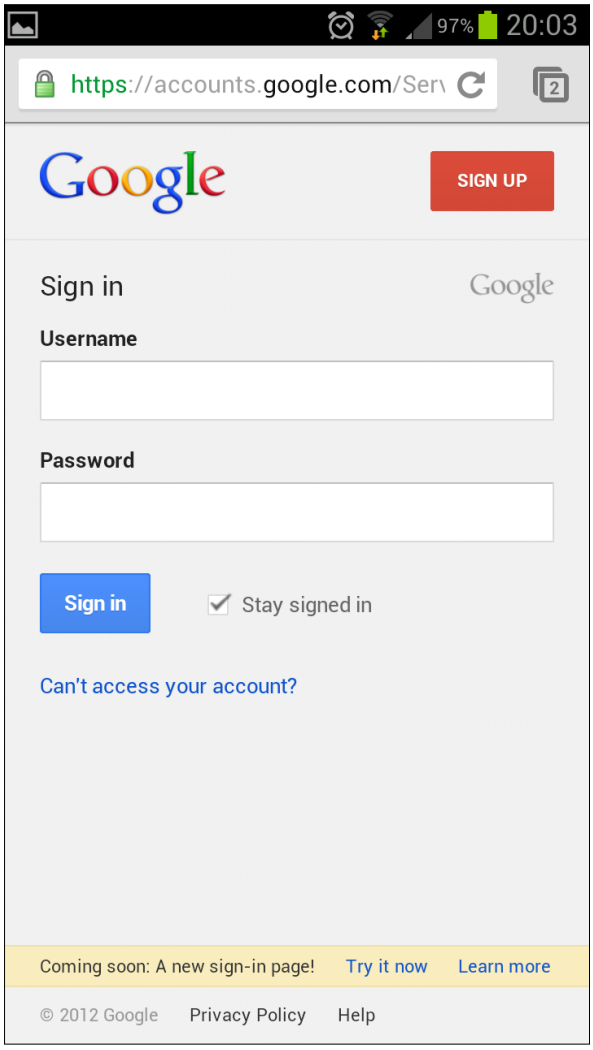


Fig. 26: ventana de login.

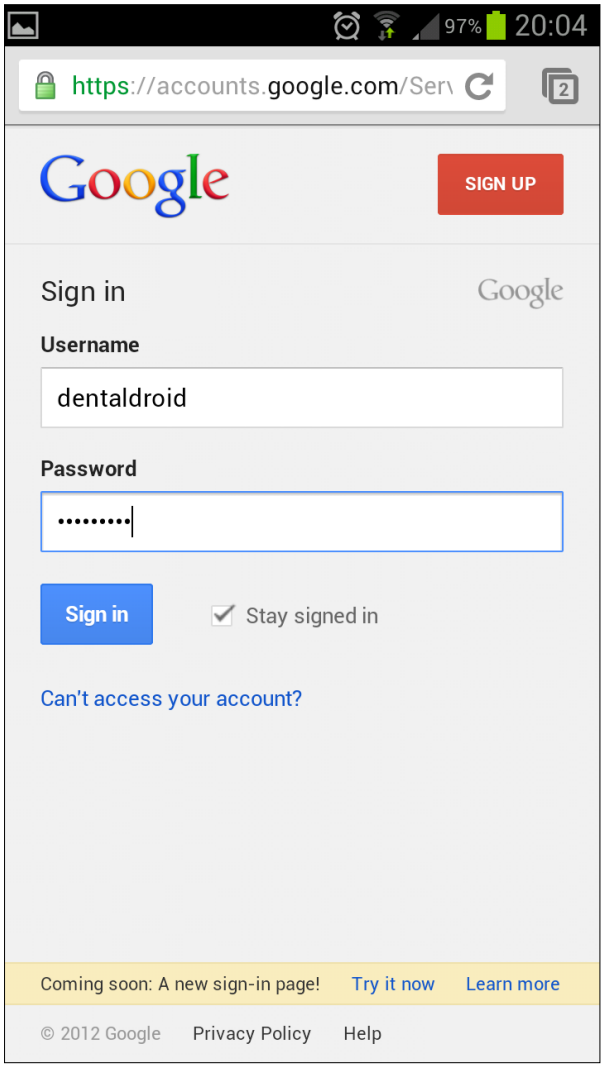


Fig. 27: ventana con usuario y contraseña introducidos.

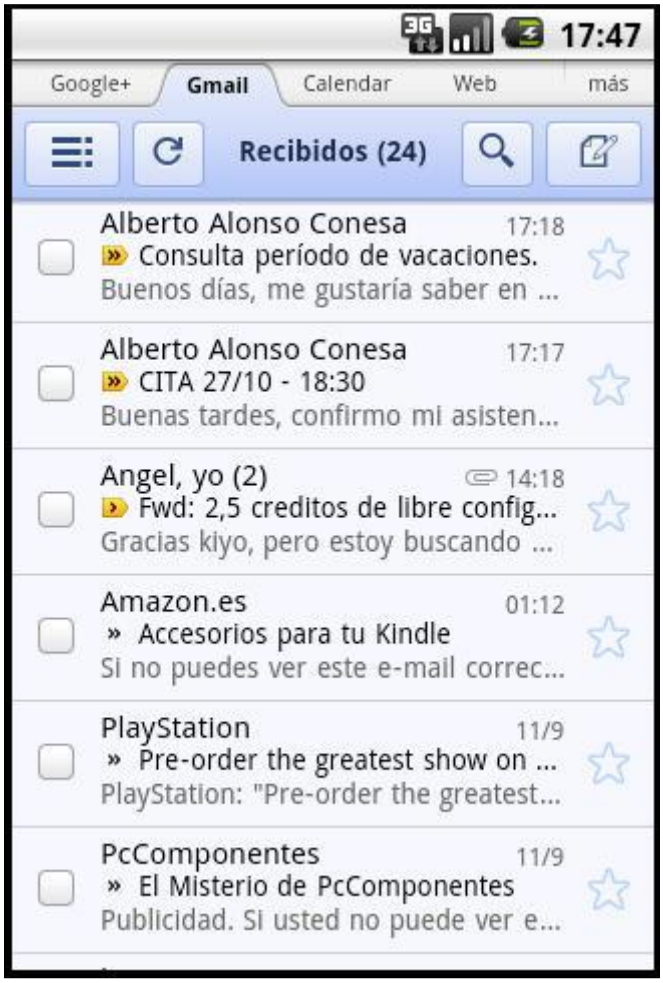


Fig. 28: vista de la bandeja de entrada.

Fig. 29: consulta de un email entrante.

Para la implementación de esta parte de la aplicación, nos hemos basado en el concepto de "Browser View", que permite incrustar un navegador directamente en nuestra propia aplicación.

En un ordenador de escritorio, un navegador web, puede ser una aplicación compleja, que consuma gran cantidad de recursos al cargar funciones como marcadores, plugins, animaciones Flash, pestañas, barras de desplazamiento, etc. En nuestra pequeña aplicación, deseamos que el flujo de funcionamiento sea totalmente estable y por ello, queremos que los distintos componentes utilizados no consuman gran cantidad de recursos. Así, hemos utilizado un contenedor del motor del navegador WebKit proporcionado por Android y que se llama WebView. Este nos permite tener el poder real de un auténtico navegador, consumiendo poco más de 1MB de memoria de nuestro dispositivo portátil.

3) CONTACTOS:

Para iniciar esta sección de nuestro programa, encontramos en la interfaz principal un botón con el nombre “Contactos”.

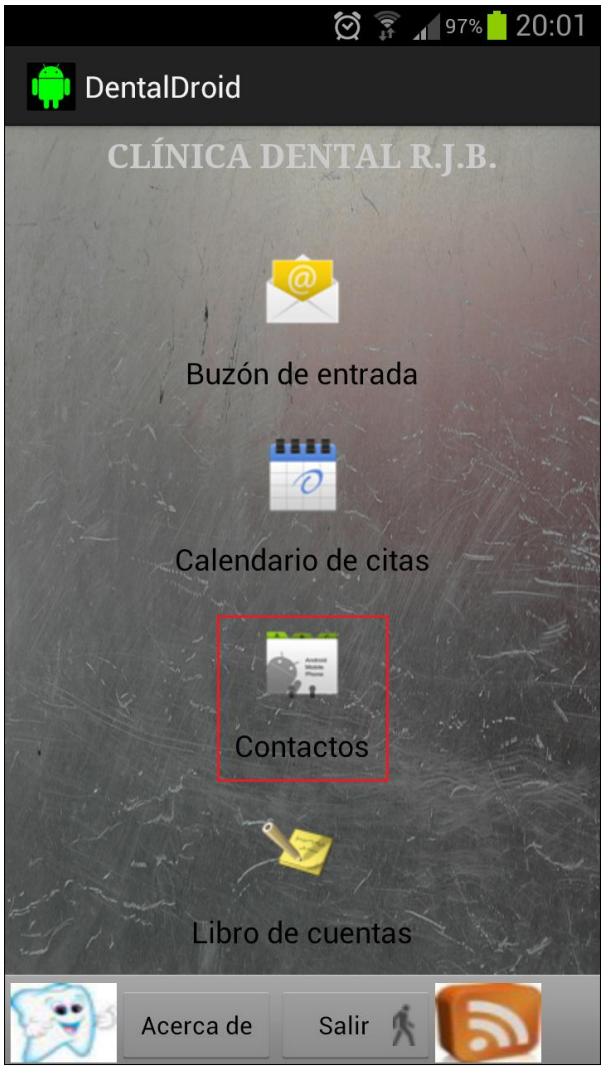


Fig. 30: interfaz principal. Botón encuadrado, sección Contactos.

En esta sección nos hemos planteado dos funciones que podrá realizar el usuario y que estarán separadas en dos pestañas dentro de la misma pantalla.

Por una parte, podrá tener un listado de los contactos (pacientes) con sus datos básicos, como son el nombre, apellidos, teléfono y correo electrónico. Los dispositivos móviles basados en Android existentes en el mercado, disponen de una propia aplicación (Contactos) para administrar la guía telefónica. En ella, se pueden añadir datos adicionales a cada contacto, como son foto, dirección postal, organización o empresa, puesto, notas, seudónimo, sitio web, etc. Además, de poder añadir más números de teléfono o cuentas de correo. La clínica podrá usar la guía para ir registrando a cada uno de los pacientes o, por otra parte, podrá registrarlos y sincronizarlos desde la cuenta de Gmail (dentaldroid@gmail.com) que tenga iniciada la

sesión. Se trata de una nueva funcionalidad ofrecida por Google y que podemos utilizarla en los dispositivos móviles más recientes y novedosos. Nuestra aplicación, tratará de leer de la guía de contactos (previamente introducidos/sincronizados), los datos básicos nombrados anteriormente de cada uno de las personas. Estos serán mostrados en forma de listado en la primera pestaña que vemos al acceder a la sección. Además, podremos recorrer cada uno de los registros a modo de consulta y pinchar sobre el que deseemos, para activar una segunda operación, que se explica a continuación.

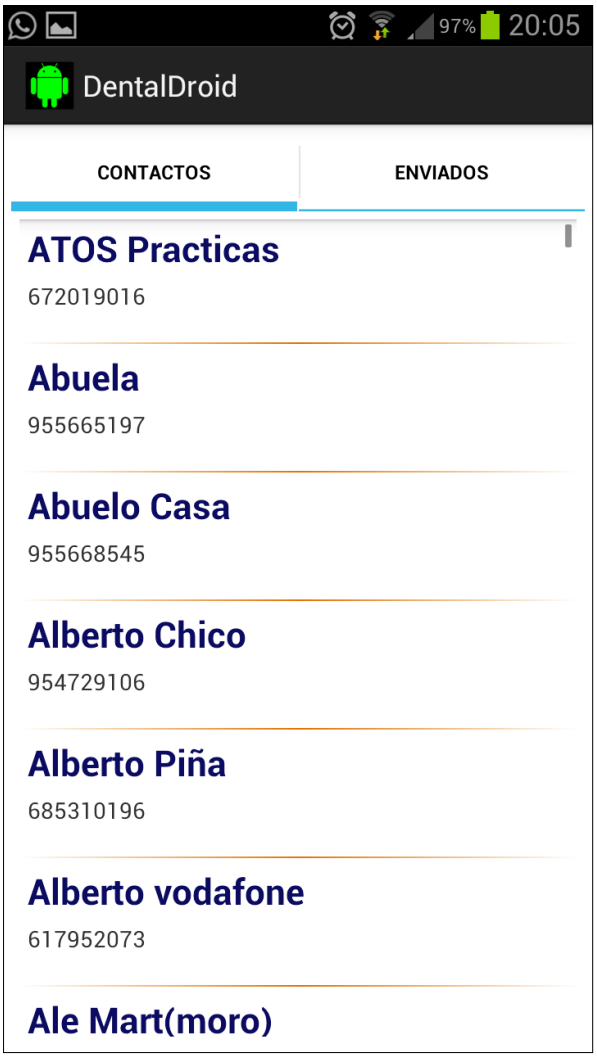


Fig. 31: pestaña 1. Listado de contactos.

La segunda función planteada se trata de poder enviar un email de confirmación de cita, es decir, un correo electrónico a modo de recordatorio a aquellos pacientes que tienen una cita en una fecha próxima. La tarea de recordar una cita la desempeñan prácticamente el 100% de las clínicas para que su agenda quede totalmente confirmada, ya sea en caso de presencia o ausencia del paciente. Hoy en día, la auxiliar o el/la administrativo, se encargan de llamar por teléfono a las personas

para realizar esta función. Incluso, tienen que repetirlo varias veces en caso de que no puedan contactar. Mediante esta propuesta de enviar un email, el tiempo empleado sería mucho menor y nos ahorraríamos los gastos de teléfono, que dependiendo del volumen de pacientes, pueden llegar a ser cuantiosos. Sabemos que se trata de una idea a largo plazo, dado que no todo el mundo maneja o tiene correo electrónico. Por ejemplo, una persona anciana es complicado que tenga incluso email. Pero, cada vez más mayores y por supuesto, los jóvenes, emplean Internet prácticamente a diario. La experiencia y su evolución en la práctica, nos dirá la aceptación o rechazo que puede tener.

Volviendo a nuestro proceso de usuario, si clicamos sobre cualquiera de los registros, nos aparecerá una ventana de diálogo, en la que se nos pregunta si queremos enviar un email de confirmación de citas. Si pulsamos aceptar, accederemos a un formulario de redacción en el que podremos editar el asunto, contenido del mensaje y adjuntar algún archivo. El último paso sería confirmar el envío pulsando "Enviar".

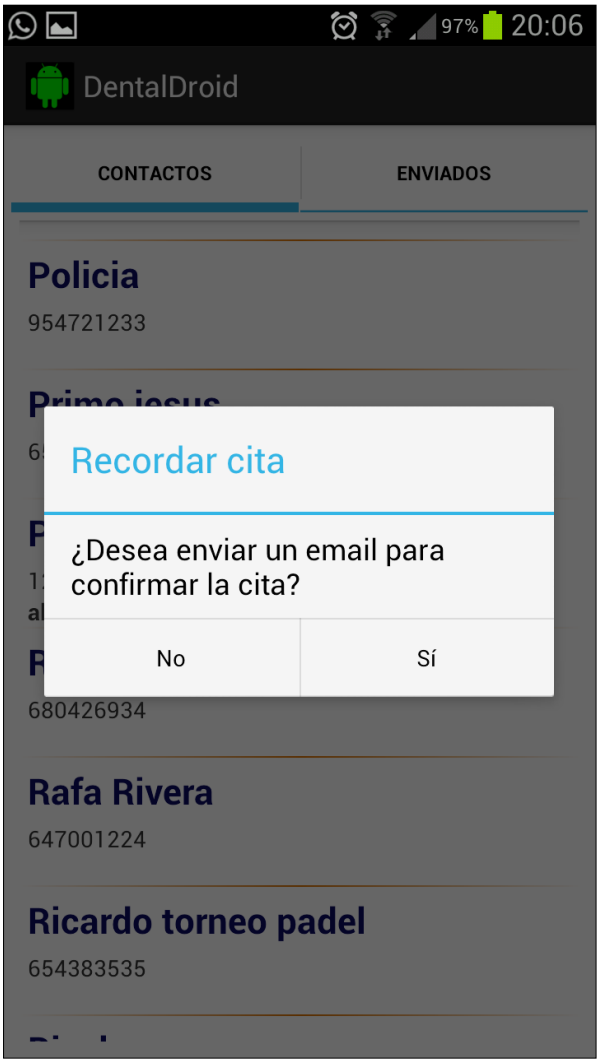


Fig. 32: ventana de diálogo de confirmación.

Tenemos que indicar que el formato a utilizar en estos correos será:

- Asunto: "CITA hora: 20:15 / día: 27/010".
- Mensaje: "*Rogamos su respuesta para confirmar la cita del asunto. Un saludo"* (mensaje introducido por defecto, editable 100%).

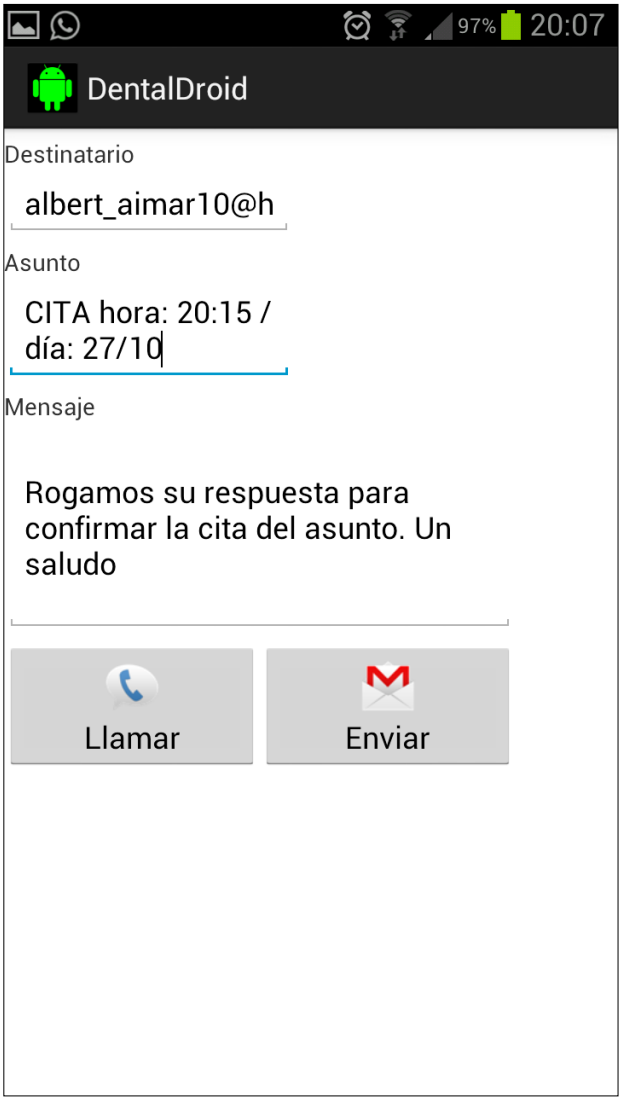


Fig. 33: formulario de envío de email.

Además del botón de envío, se observa otro con el cual podremos llamar al paciente en caso de no poder enviarle el email. Así aseguramos la operatividad de siempre.

Podemos comprobar en nuestra bandeja de correo que el email ha sido recibido correctamente.

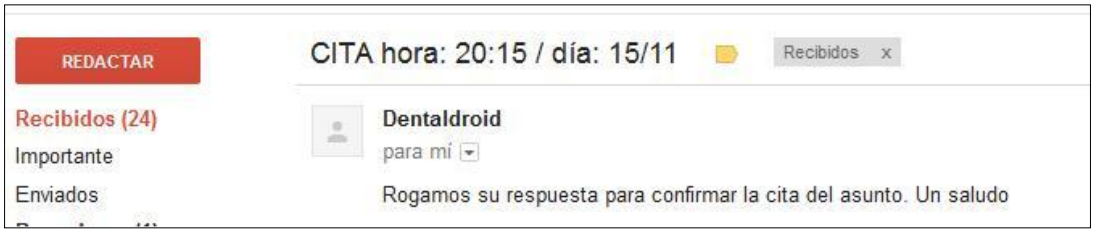


Fig. 34: bandeja de entrada. Confirmar recepción de email.

En la segunda pestaña de esta sección, queremos guardar un historial donde aparezcan registros de cada uno de los correos que hayamos enviado, almacenando la persona destinatario, el asunto, la fecha y la hora. Así, en todo momento sabremos a quien, cuantos fueron y en qué fecha se mandaron los emails de confirmación. Los correos irán organizados por mes.

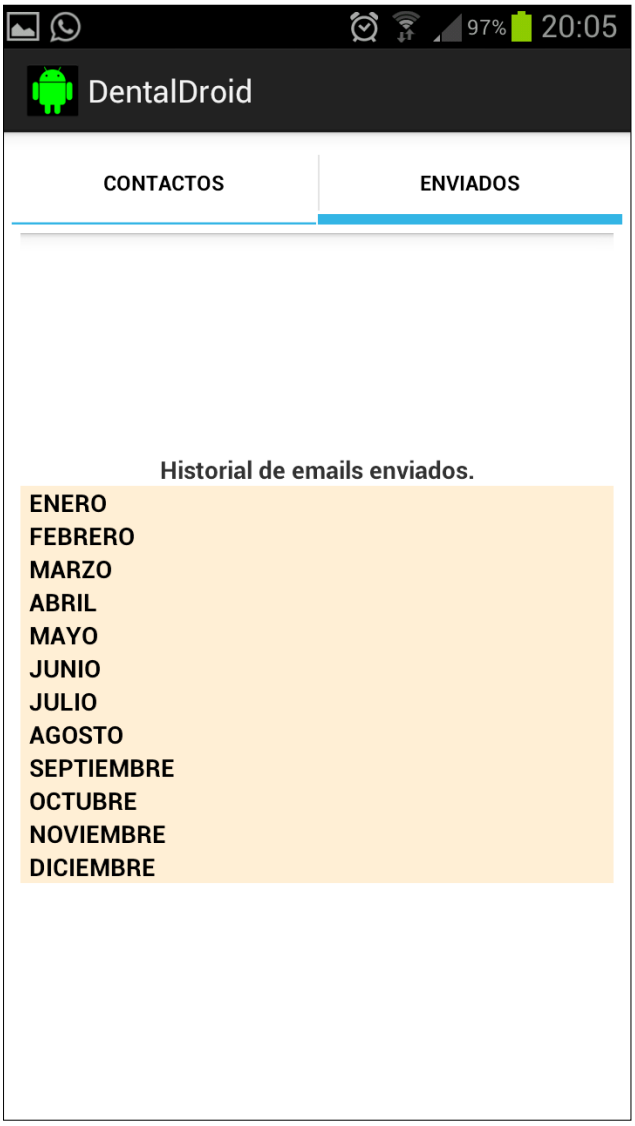


Fig. 35: pestaña 2. Historial organizado por mes.

Después de proceder al envío de los correspondientes emails, podemos ver el historial actualizado.

Fig. 36: historial actualizado.

4) LIBRO DE CUENTAS:

Para entrar en la sección del libro de cuentas, deberemos pulsar el icono correspondiente que hay en la pantalla principal de la aplicación.

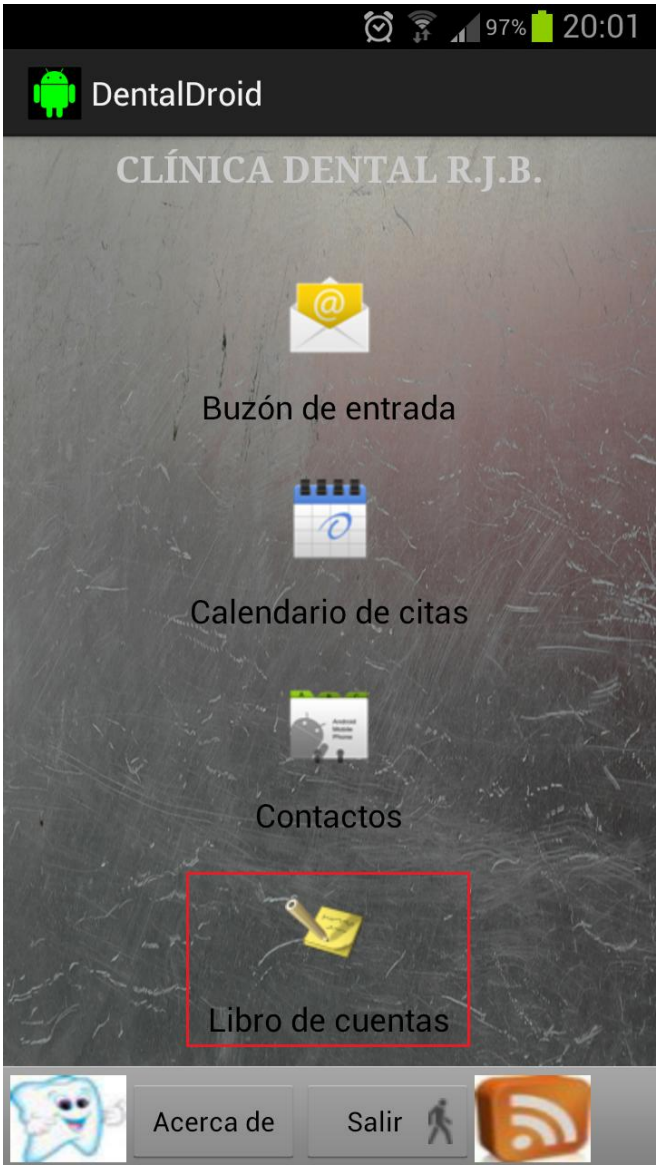


Fig. 37: interfaz principal. Botón enmarcado, sección "Libro de Cuentas".

Una vez dentro, pretendemos que el usuario pueda administrar las tareas que desempeña. Muchos odontólogos son autónomos por lo que sus beneficios son en base a un porcentaje, establecido de acuerdo con la dirección de la clínica, sobre los diagnósticos realizados. Nuestra idea es dividir esta nueva funcionalidad en dos partes, separadas visualmente en pestañas.

En primer lugar, se muestran en la siguiente imagen las distintas opciones de configuración a la hora de realizar los cálculos. Vamos a proceder a comentar cada uno de ellas:

Fig. 38: pestaña de configuración.

- Campo FECHA: mostrará la fecha actual. Nos será de utilidad ya que estamos calculando los beneficios obtenidos en un día en concreto.

- Botón CAMBIAR FECHA: nos servirá para poder realizar los cálculos teniendo en cuenta cualquier otra fecha pasada.

- Botón de IR A HOY: se trata de un acceso inmediato a la fecha actual en caso de haber modificado el campo FECHA.

- Botón de TRATAMIENTOS: este botón nos llevará a una nueva ventana donde nos mostrará cada uno de los posibles tratamientos que se realicen en la clínica.

- Lista PORCENTAJE A APLICAR: tendremos una lista desplegable sobre la que podremos seleccionar el porcentaje específico asignado al profesional en cuestión.

Fig. 39: cuadro para establecer una nueva fecha.

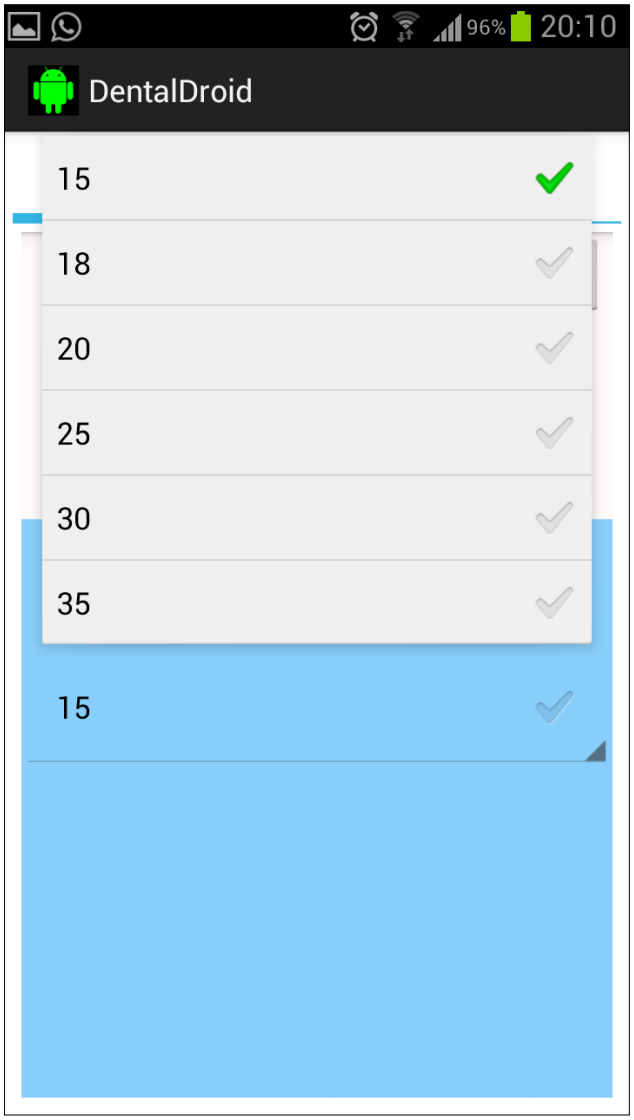


Fig. 40: lista desplegable de selección de porcentajes.

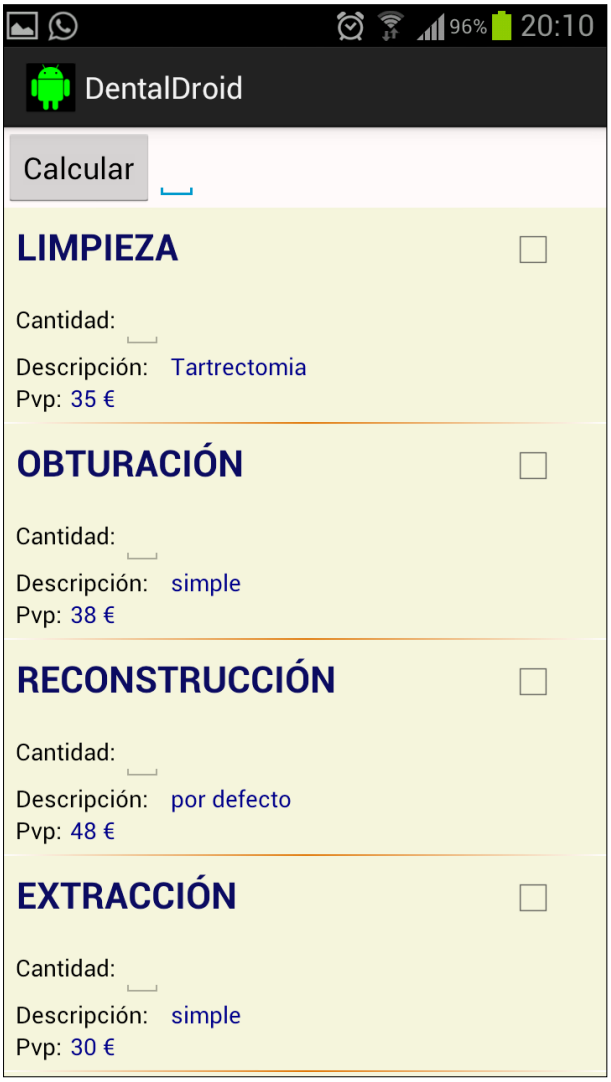


Fig. 41: lista de tratamientos seleccionables.

En la imagen anterior, se muestran los tratamientos que podremos seleccionar según las labores realizadas en la fecha en cuestión. Tal y como vemos, cada tratamiento estará compuesto del título, una casilla para la cantidad junto con una opción de marcado (checkbox) que se "chequea" automáticamente cuando añadimos una cantidad mayor que cero. Junto a ellos, también observamos una breve descripción y el precio establecido. El usuario deberá seleccionar aquellos tratamientos realizados, junto con la cantidad correcta, y pulsar el botón de "Calcular". En este, se mostrará el importe total deducido, con el porcentaje ya aplicado.

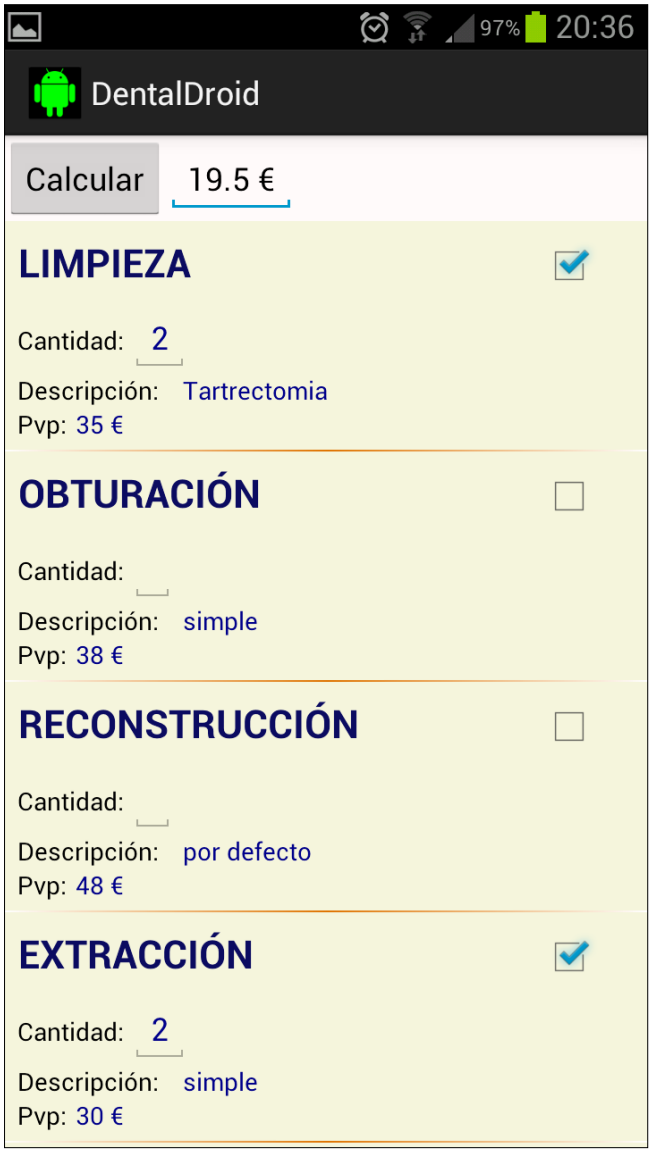


Fig. 42: lista de tratamientos seleccionados. Importe calculado.

La segunda parte de esta nueva sección, consiste en almacenar en un histórico, cada uno de los beneficios obtenidos por fechas, y así, a final de mes, poder obtener un extracto de las cuentas. Procedemos a mostrar dos capturas en la que se observan el diseño en sí de la pestaña y los registros ya almacenados después de realizar todas las acciones comentadas en este apartado.

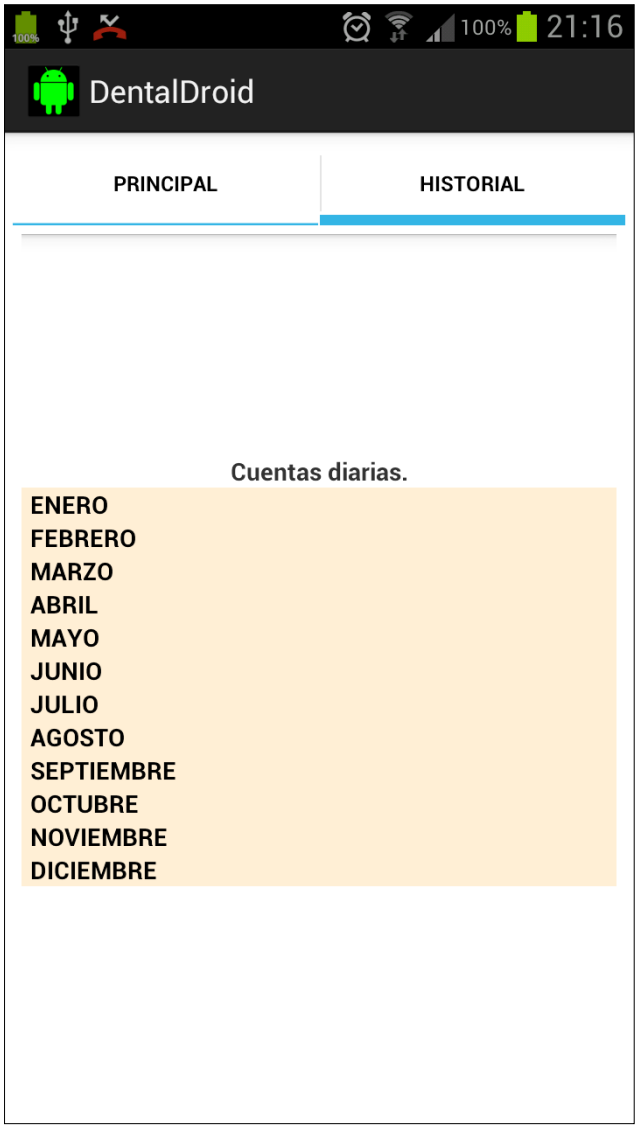


Fig. 43: pestaña Historial de cuentas diarias.

Fig. 44: histórico con registros añadidos.

5) LECTOR DE FEEDS:

En la interfaz principal hemos incorporado un botón con el que accederemos a una sección en la que se muestran las noticias actualizadas de distintas páginas webs. En concreto para nuestro caso, consultaremos webs dedicadas o relacionadas con el mundo de la odontología. Un ejemplo de ellas es la página oficial del colegio de odontólogos de Sevilla, http://www.dentistassevilla.com/.

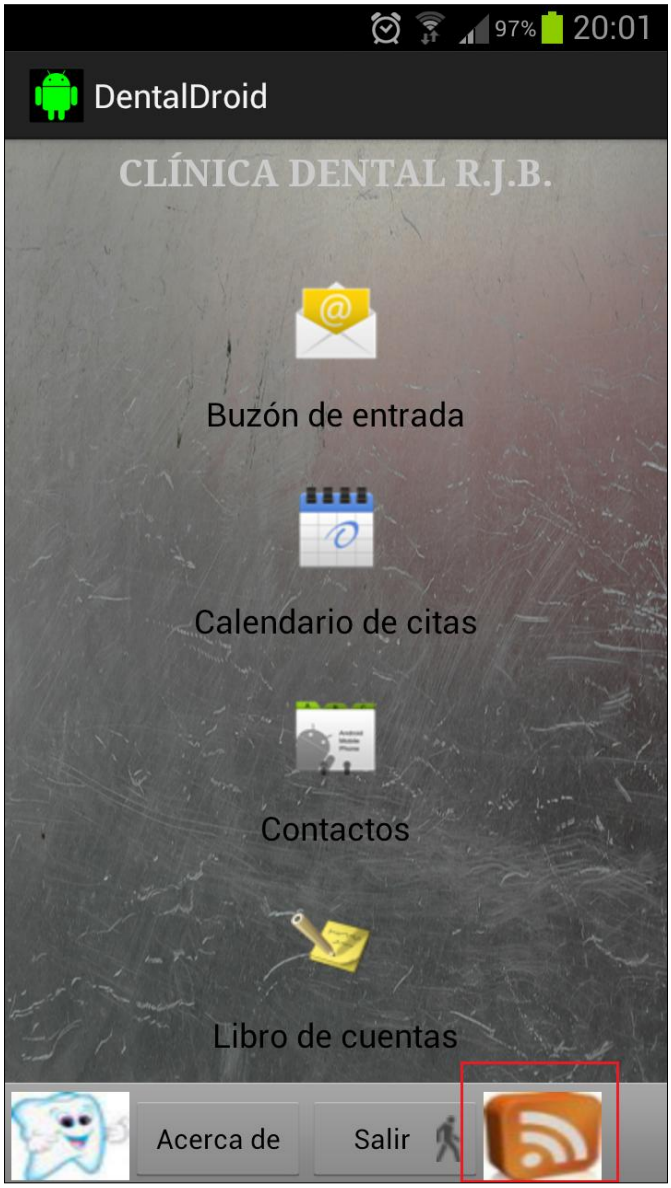


Fig 45: interfaz principal. Botón encuadrado, lector de RSS.

Una vez dentro de esta sección, nos muestra un listado con el resumen, título y fecha de cada uno de los artículos de la sección de noticias. Podremos actualizar la información cada vez que queramos pulsando un botón situado en la parte superior con la etiqueta “Recargar datos”. Con esta información estaremos al día en las

novedades que vayan apareciendo en aquellas webs que nos interesan. Tenemos que añadir que los datos mostrados y las páginas de las que se extraen, pueden modificarse en todo momento a gusto del usuario.

Hubo un momento en el que se pensó añadir una nueva funcionalidad a esta sección. Se trataba de poder lanzar el navegador y visualizar la noticia completa de la web origen acerca del artículo que previamente se había seleccionado. Después del análisis y reflexión, se descartó debido a que las webs que nos interesan controlar nos redirigen a artículos de "Twitter" que no nos aportan ninguna información extra, más que el propio título. Si quisiéramos leer los RSS de otra página web, por ejemplo http://www.elmundo.com, sí podríamos haber añadido esta funcionalidad.

A continuación se muestran diversas capturas durante el proceso del usuario en esta sección:

Fig. 46: vista de la sección con los datos cargados.

Fig. 47: vista de la sección durante el proceso de carga/actualización de datos.

Fig. 48: vista extra de la sección con datos cargados desde la página web de http://www.elmundo.com.

A continuación, vamos a explicar más técnicamente que es un feed RSS y un lector.

Un feed es un archivo generado por algunos sitios web que contiene una versión específica de la información publicada en esa web. Cada elemento de información contenido dentro de un archivo RSS se llama “item”. Cada ítem consta normalmente de un título, un resumen y un enlace a la web de origen que contiene el texto completo. Además puede mostrar información adicional como la fecha de publicación o nombre del autor. El archivo RSS se reescribe automáticamente cuando se produce alguna actualización en los contenidos del sitio web.

Cada feed dispone de su propia dirección de Internet o URL del mismo modo que las páginas HTML convencionales. Sin embargo, a diferencia de éstas, no se pueden visualizar directamente con el navegador, de modo que es necesario utilizar un lector de feeds. Estos lectores funcionan de forma similar a otros programas de

correo electrónico. En este caso, consultan de forma periódica las direcciones de los feeds para obtener la última versión disponible de su archivo RSS.

Por último, los lectores de feeds pueden ser aplicaciones web o aplicaciones locales que se instalan en el ordenador del usuario.

PRUEBAS

A lo largo de la realización del proyecto, hemos ido realizando pruebas integradas o unitarias sobre cada una de las secciones implementadas de manera individual, y finalmente, sobre todo el software ya integrado en la propia aplicación. Tal y como he visto anteriormente en el apartado de manual de usuario, se detallan cada uno de los pasos a seguir en las distintas secciones (podemos llamar pruebas) junto con los resultados obtenidos (capturas).

COMPARATIVA CON OTRAS ALTERNATIVAS

Durante muchos años, la ya conocida fábrica de Apple, ha copado las ventas en todo el mundo en lo que refiere al mercado tecnológico. Cabe destacar su óptimo y eficiente sistema operativo (iOs), además de todo tipo de dispositivos móviles tales como iPhone, iPad, iMac, etc. Sin embargo, en los últimos años le ha salido un duro competidor, Android, que se está expandiendo a gran ritmo en todos los frentes gracias a su espectacular evolución y funcionalidades que ofrece.

En el momento de decidir que plataforma usar para nuestro proyecto, nos decantamos por Android debido a las siguientes características que vamos explicar y comparar con el gigante iOs.

En primer lugar, Android es de código abierto y está diseñado para distintos fabricantes mientras que iOs es un sistema operativo cerrado y específico para los dispositivos de la marca Apple. El código abierto supone unos costes técnicos iniciales muy altos, mientras que la opción cerrada supone unos fuertes costes sociales a largo plazo.

En cuanto a seguridad, el gran problema de la apertura de código es la mayor vulnerabilidad ante virus que supone poner al servicio de los programadores su código. Además, Android deja a elección del usuario el poder instalar aplicaciones que no provengan de la tienda oficial, que puede suponer un claro riesgo. Siempre ha existido la creencia de que iOs era invulnerable frente a los virus (falso) pero ciertamente se comporta de una manera más robusta frente a los ataques dada la dificultad de acceder al sistema de archivos. Esa dificultad para acceder al sistema de archivos es una ventaja de seguridad pero también supone una restricción al hacer que no se pueda sacar el máximo rendimiento a sus dispositivos.

Hablando de la rapidez siempre se ha dicho que iOs es más fluido que Android. Con las primeras versiones del código libre dicha afirmación sí podía ser más apreciable. Android se basa en Java y por ello, necesita cargar inicialmente una máquina virtual y gestionar su memoria y caché de una manera menos efectiva. Pero gracias a las últimas mejoras de las versiones 4.x, podemos decir que el rendimiento que le han sacado al sistema operativo le hace comportarse con la misma fluidez (incluso mejor) que iOs.

En cuanto a la experiencia de usuario, Android se encuentra un paso por delante ofreciendo:

- Posibilidades de personalización de la pantalla, añadiendo widgets que conceden al usuario la posibilidad de tener sus múltiples escritorios aplicaciones abiertas con información e interfaces atractivas.
- Instalación de aplicaciones de manera remota, mientras que en iOs es estrictamente necesario conectar el dispositivo para tener la aplicación en él.

Si comparamos las últimas versiones de ambos, iOS5 y Ice Cream Sandwich&JellyBean, encontramos mejoras muy significativas que vamos a analizar:

- iOs5, además de incorporar el sistema de notificaciones, han añadido muchos servicios basados en la nube (iCloud). Se trata de una aplicación de almacenamiento con sincronización instantánea en todos los sistemas bajo una misma cuenta de iTunes.
- La respuesta de Google fue sacar Google Drive, su aplicación de almacenamiento en la nube que viene incorporada por defecto en los nuevos terminales.
- Con Android Ice Cream Sandwich, Google ha apostado fuertemente por la tecnología NFC, otorgándole una integración absoluta.

Por último, desde un punto más técnico, se muestran las especificaciones más detalladas de cada plataforma:

iOS5 Android 4.0

	iOS 5	Android 4.0
Kernel	OS X	Linux
Estándares soportados	GSM y CDMA	GSM y CDMA
Multitarea	Si	Si
Cortar/Copiar/Pegar	Si	Si
Hardware soportado	iPhone, iPad y iPod touch	Amplia variedad de dispositivos
Compatibilidad entre modelos	Compatibilidad total	No hay compatibilidad
Seguridad	Muy seguro	Suscetible al malware
Tienda de películas	iTunes	Si (desde el Market)
Integración Social Media	Twitter	Facebook y Twitter
Music Store	iTunes	No
Juegos	Amplia variedad	Poca variedad
Social Gaming	Game Center	No
Navegador	Safari Móvil	Basado en Chrome
Soporte Flash	No	Si
Crashes (Cuelgues)	Muy poco frecuentes	Frecuentes
Motor de búsqueda por defecto	Google	Google
Pantalla de Incio	Iconos	Iconos y Widgets
Suite de productividad	iWork	Google Docs
Reconocimiento de voz	Si	Si
Asistente por Voz	Siri	No
Sincronización Wi-Fi	Si	No por defecto
Soporte para tablets	Si	Si
Actualizaciones	Si	Si
Actualizaciones aéreas	Si	Si
Soporte en la nube (Cloud)	iCloud	Google Sync
Personalización	Si (con jailbreak)	Si
Aplicaciones	Más de 500,000	Más de 250,000
Calidad de las aplicaciones	Buena calidad	Mediana calidad
Notificaciones	Pull-down	Pull-down
Aceleración de hardware	Si	Si
Impresión de pantalla	Si	Si
Tienda de libros	iBooks	Google Books
Respaldo inalámbrico en nube	Si	No
Interfaz de Usuario	Fácil e Intuitiva	Poco Intuitiva

FUTURAS MEJORAS

Las futuras mejoras y actualizaciones que se pueden desarrollar sobre la aplicación en base a su liberación y difusión en el market (Google Play) de Android son:

- Adaptar funcionalidad y optimización para versiones de firmware superiores a la versión 4.0.
- Añadir base de datos SQLITE de respaldo.
- Incorporar banners* de publicidad.

Banner: http://es.wikipedia.org/wiki/Banner

BIBLIOGRAFÍA

A continuación se van a mostrar cada uno de los recursos consultados, tanto enlaces a páginas web en Internet, tutoriales… como libros de programación en Android:

- Enlaces de búsqueda "Integrate your Android application with google calendar". http://www.google.es/search?q=integrate+your+Android+application+with+google+calendar&hl=es&rlz=1G1ACALCESES392&prmd=imvnsfd&ei=G55CT5usDcS6-Ab9seTPBQ&start=10&sa=N&biw=1280&bih=664

- Google Code. "How to get Google Calendar on your mobile device". https://support.google.com/calendar/bin/answer.py?hl=en&answer=65928

- StackOverflow. "Unable to integrate Google Calendar API in Android app". http://stackoverflow.com/questions/4774160/unable-to-integrate-google-calendar-api-in-android-app

- IBM, DeveloperWorks. "Integrate your PHP application with Google Calendar". http://www.ibm.com/developerworks/xml/library/x-googleclndr/index.html?ca=drs-tp2808

- Google Code. "Autorización aplicaciones". http://code.google.com/intl/es/apis/calendar/v3/using.html#auth

- Google Code. "Video Google I/O 2011 Best Practices". http://code.google.com/p/google-api-java-client/wiki/Android

- Maestros del web. "Libro Curso Android, Desarrollo de aplicaciones móviles". http://www.maestrosdelweb.com/descargas/

- Google Code. "Welcome to Google on Android". http://code.google.com/intl/es-ES/android/

- Google Code. "Google APIs Add-On". http://code.google.com/intl/es-ES/android/add-ons/google-apis/index.html

- Maestros del web. "Curso Android: nociones lector de feeds". http://www.maestrosdelweb.com/editorial/curso-android-construir-lector-de-feeds/

- Android Developers. "Services". http://developer.android.com/guide/topics/fundamentals/services.html

- Jim Blackler blog. "Accessing the internal calendar database inside Android apps". http://jimblackler.net/blog/?p=151

- Google Code. “Get started with the Google Calendar API”. https://developers.google.com/google-apps/calendar/get_started

- Proyecto Ajpdsoft. “Desarrollar aplicación Android con acceso a base de datos SQLite con Eclipse”. http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=537

- Francho(LAB). “Usando ContactsContract para sacar el listado de teléfonos de la agenda”. http://francho.org/2011/01/11/android-usando-contactscontract-para-sacar-el-listado-de-telefonos-de-la-ageda/

- Android-spa. “Acceso a contactos Android 2.0 y superiores”. http://www.android-spa.com/viewtopic.php?t=14320&sid=1017c0510355cbf6d49fc6eefeca13b6

- Anddev.org. “How to get emails from contacts”. http://www.anddev.org/how_to_get_emails_from_contacts-t719.html

- HigherPass. “Working with Android Contacts”. http://www.higherpass.com/Android/Tutorials/Working-With-Android-Contacts/

- Tutorial Android. “Como pasar datos a otra Activity”. http://www.tutorialandroid.com/medio/como-pasar-datos-a-otra-activity/

- Android-spa. “Como pasar datos de la Activity principal a la secundaria”. http://www.android-spa.com/viewtopic.php?t=1004&sid=96f6942d0f9893ed9b2d7ec42c99f3de

- AndroidIdeity. “Intents explícitos y paso de información entre actividades”. http://androideity.com/2011/10/23/intents-explicitos-y-paso-de-informacion-entre-actividades/

- JRamirez Blog. “Android III: diseño de pestañas”. http://jramirez-favoritos.blogspot.com.es/2010/02/android-iii-ejemplo-de-diseno-con.html

- DCSAPP Mobiles. “Android Tutorial – Creación de Tabs”. http://dcsappmobiles.wikispaces.com/Android+Tutorial+-+Crear+Tabs

- AndroidIdeity. “Pestañas usando Eclipse”. http://androideity.com/2012/02/13/pestanas-en-android-usando-eclipse/

- MyBringBack.com. “Canal Android Development For Absolute Beginners”. http://www.youtube.com/playlist?list=PLB03EA9545DD188C3&feature=plcp

- TheNewBoston. “Canal Android Application Development Tutorials”. http://www.youtube.com/playlist?list=PL2F07DBCDCC01493A&feature=plcp

- Javielinux Blog. “Creando una pantalla de preferencias en Android”. http://www.javielinux.com/232-Creando_una_pantalla_de_preferencias_en_Android.htm

- Developer Android. “Samples”. http://developer.android.com/resources/samples/ApiDemos/res/xml/preferences.html

- Dipler. “Solucionando el problema de los ListView con CheckBox en Android”. http://www.dipler.org/2011/02/solucionando-el-problema-de-los-listview-con-checkbox-en-android/

- StackOverflow. “Android Calendar View for Date Picker”. http://stackoverflow.com/questions/3712710/android-calendar-view-for-date-picker

- Slideshare. “Android Application Development, User Interface”. http://www.slideshare.net/androidstream/android-user-interface-tutorial-datepicker-timepicker-spinner

- Developer Android. “CalendarView”. http://developer.android.com/reference/android/widget/CalendarView.html

- Libre y extremo. “Widget DatePicker en Android”. http://libreyextremo.blogspot.com.es/2012/03/tutorial-android-parte-15-widget.html

- Developer Android. “Pickers”. http://developer.android.com/resources/tutorials/views/hello-datepicker.html

- Sanathnandasiri Blog. “How to work with Android spinner widget”. http://sanathnandasiri.blogspot.com.es/2011/11/today-im-going-to-show-how-to-work-with.html

- Slideshare. “Desarrollo de aplicaciones móviles con acceso a datos remotos”. http://www.slideshare.net/istmosoft/android-presentacion

- Android-spa. “Obtener ítems seleccionados en ListView con CheckBox”. http://www.android-spa.com/viewtopic.php?t=12197&start=0&postdays=0&postorder=asc&highlight=&sid=40e6ebbc524052aa950595b3e075a139

- ForumSEE. “Recorrer un ListView con dato e inserta a la BBDD”. http://telefonia.forumsee.com/a/m/s/p12-4072-0586430--recorrer-listview-con-dato-insertar-ala.html

- Enlaces de búsqueda. “CheckBox is checked listview in android”. https://www.google.es/search?hl=es&rlz=1G1ACALCESES392&sa=X&ei=mhu1T8i-DejX0QWRjfHaDw&ved=0CAgQvwUoAQ&q=checkbox+is+checked+listview+android&spell=1&biw=1280&bih=635

- Google Groups. “How to use checkbox in listview”. http://groups.google.com/group/android-developers/browse_thread/thread/ed4d5de0b9a9afd?pli=1

- StackOverflow. “Intent URI to launch GMAIL app”. http://stackoverflow.com/questions/3470042/intent-uri-to-launch-gmail-app

- AndroidOverflow. “Trabajar con WebViews”. http://www.androidoverflow.com/63/trabajar-con-webviews

- Adictos al trabajo. “Leer correos de Gmail”. http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=android-gmail-content-provider

- Fernando F. Gallego Blog. “Android dev: abrir Google Navigation mediante Intent”. http://www.forgottenprojects.com/android-dev-abrir-google-navigation-mediante-intent/

- NosinmiUbuntu. “Abrir aplicaciones en Android: Intents”. http://www.nosinmiubuntu.com/2012/05/como-abrir-aplicaciones-en-android.html

- Android-spa. “Abrir una aplicación desde otra”. http://www.android-spa.com/viewtopic.php?t=14990&sid=4917f6a9a65f5ad76bc3dd39b404edb7

- Developer Android. “Intents and Intent Filters”. http://developer.android.com/guide/components/intents-filters.html

- R Android Blog. “How to create a View on a map button”. http://randroidblog.blogspot.com.es/2012/01/check-calendar-events.html

- Developer Android. “View your Calendar & Events”. http://support.google.com/android/bin/answer.py?hl=en&answer=1715294

- TechRepublic. “Programming with the Android 4.0 Calendar API”. http://www.techrepublic.com/blog/app-builder/programming-with-the-android-40-calendar-api-the-good-the-bad-and-the-ugly/825

- Archivo .PDF Programación en Android v2. http://books.openlibra.com/pdf/Manual-Programacion-Android-v2.pdf

- StackOverflow. “Create calendar event in the default Android calendar”. http://stackoverflow.com/questions/11999004/create-calendar-event-in-the-default-android-calendar

- Vogella. “Android Calendar API - Tutorial”. http://www.vogella.com/articles/AndroidCalendar/article.html

- StackOverflow. “Add a calendar to Android app, marking certain events”. http://stackoverflow.com/questions/6667095/add-a-calendar-to-android-app-marking-certain-events&usg=ALkJrhg6JDSCyMWVVTjLi5f3rETggH3EyQ

- DreamInCode. “Calendar Viewer Application”. http://www.dreamincode.net/forums/topic/25042-creating-a-calendar-viewer-application/

- StackOverflow. “Add events to Android calendar using Java”. http://stackoverflow.com/questions/10361831/add-android-calendar-events-using-java&usg=ALkJrhjpQdKLAbLj14ntwNilKvLKMLHHUA

- Android Tutorial. “Android SDK Development & Programming”. http://www.edumobile.org/android/android-development/calendar-application/

- Roman10. “Programming with Calendar”. http://www.roman10.net/android-tutorialprogramming-with-calendar/

- Wagied David Blog. “Android Simple calendar”. http://w2davids.wordpress.com/android-simple-calendar/

- Developer.com. “Android Calendaring 4.0”. http://www.developer.com/ws/android/programming/android-calendaring-tutorial-android-4.0.html

- StackOverflow. “Google API Java client for Google Calendar on Android Infinite loop”. http://stackoverflow.com/questions/6403293/google-api-java-client-for-google-calendar-on-android-infinite-loop&usg=ALkJrhjB-4DOXhu8FxdKac6XyJZszHUGpA

- StackOverflow. “Android Calendar API insert recurring event”. http://stackoverflow.com/questions/8856468/android-calendar-api-insert-recurring-event&usg=ALkJrhjddcVI3102KaT1tnAnDb4vwj2QZg

- DiMarzio, Jerome. Android, a programmer´s guide. 1998.

- Ableson, W. Frank. Android, a developer´s guide. 1997.

- Burnette, Ed. Hello, Android. Introducing Google´s Mobile Development Platform. 2nd edition. 2009.

- Burnette, Ed. Hello, Android. Introducing Google´s Mobile Development Platform. 3rd edition. 2010.

- Rogers, Rick. Lombardo, John. Mednieks, Zigurd. Meike, Blake. Android, Application Development. Programming with the Google SDK.

- Hashimi, Sayed. Komatineni, Satya. Pro Android & Pro Android2. 2009 – 2010.