Índice de contenidos

Índice de ilustraciones

Índice de tablas

1 Antecedentes y estudio teórico

Una de las circunstancias generadas por la evolución de la telefonía móvil a telefonía móvil inteligente (*Smartphone*) es la aparición de aplicaciones desarrolladas por los propios usuarios y vendidas en "*Market*" o tiendas de venta que ofrecen las distintas plataformas (Android, Apple, Windows Mobile,...). Esto genera un estado de innovación constante ya que cada plataforma pasa a poseer miles de desarrolladores que generan nuevas funcionalidades o aplicaciones a diario.

De estas aplicaciones, una de las áreas en las que más hincapié se hace es en la de educación. Gracias a la gran cantidad de servicios disponibles en Internet para realizar traducciones y a la necesidad cada vez más acuciante de poder comunicarse con gente en otros idiomas, las aplicaciones orientadas a mejorar un idioma o un vocabulario desconocido están a la orden del día. De estas aplicaciones se pueden encontrar principalmente de dos tipos, los diccionarios de traducción y las puramente educativas.

Por lo que hace a las de traducción, aparecen diccionarios online que se basan en servicios gratuitos, diccionarios offline a partir de un diccionario incluido en la propia aplicación y ambas, que obtienen la traducción de las palabras buscadas por el usuario a partir de un servicio de Internet o API. Esta traducción la almacena en la aplicación generando así un diccionario offline a partir de las palabras buscadas.

En cambio en las educativas, la aplicación ofrece al usuario únicamente una serie de herramientas multimedia con las que aprender el idioma o vocabulario. La ventaja de estas aplicaciones es que el hecho de ser interactivas facilitan la memorización del temario, pero la principal desventaja, es que dicho temario en la mayoría de casos no es dinámico ni se ajusta a las necesidades del usuario.

Este proyecto, estaría más cerca del primer grupo que del segundo, ya que lo que se pretende realizar es una aplicación de traducción en el que el usuario realizaría traducciones a través de una servicio de búsqueda incluida en la aplicación.

Dicha traducción se almacenará en la propia aplicación con dos objetivos, el primero, como ya se ha comentado, es ofrecer la posibilidad de tener un diccionario offline y el segundo es a partir de las palabras buscadas por el usuario, ofrecer una serie de juegos con los que memorizar las palabras buscadas, las cuales se podrán considerar un temario dinámico ajustado a las necesidades de cada usuario.

1.1 Diccionarios de traducción en Android Marquet

Como se ha visto en el punto anterior, dentro del campo de los diccionarios de traducción hay principalmente dos tipos: las herramientas de traducción, tales como diccionarios (online y offline), y las herramientas de *e-learning* centradas en aprendizaje de idiomas.

En cuanto a las aplicaciones de traducción, en el momento del estudio se han encontrado cerca de 1000 resultados. Como es inviable realizar un estudio de cada una de ellas y además se aleja del propósito de este proyecto, se han escogido 5 aplicaciones con características diferentes para analizar, a parte de su funcionamiento básico, la puntuación y el número de instalaciones de estas aplicaciones. Con ello se pretende predecir los resultados que se obtendrán en el lanzamiento de la aplicación que se está realizando.

1.1.1 Español Traductor / Diccionario

Número de votos: 32.317 **Puntuación:** 4,5/5 **Instalaciones:** 1-5 M

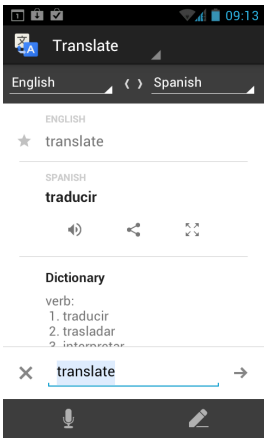
Ilustración 1 - Translator Dictionary

Diccionario traductor de Inglés a Español y de Español a Inglés que permite, además de buscar palabras, traducir los SMS y e-mails entrantes al idioma deseado, copiar y pegar la traducción a otra aplicación y historial de traducción.

Precio: Versión gratuita + publicidad & Pro (2,49€)

1.1.2 Traductor de Google

Número de votos: 168.573 **Puntuación:** 4,6/5 **Instalaciones:** 10-50M



Esta aplicación es la nativa para Android de Google Translate. Tiene capacidad para traducir frases a más de 60 idiomas, traducir por voz, reproducir las traducciones en audio, destacar traducciones favoritas para acceder a ellas rápidamente incluso sin conexión, acceder al historial de traducciones sin conexión, realizar traducciones de idiomas con alfabeto diferente al latino (pinyin, japonés, etc...) para poder leerlos fonéticamente.

Precio: Gratis

Ilustración 2 Google Translate

1.1.3 Translate

Número de votos: 6.511 **Puntuación:** 4,2 **Instalaciones:** 0,5-1M

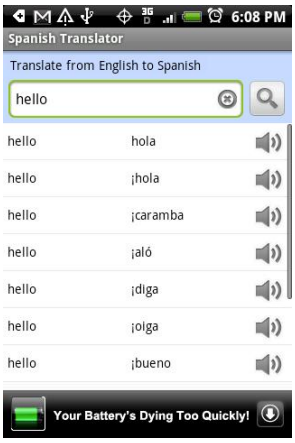
Aplicación que utiliza la Api de Google Translate para traducir entre más de 150 parejas de idiomas. En su última versión requiere el acceso a los contactos del usuario para permitirle enviar por email o SMS las traducciones.

Precio: Gratis

Ilustración 3 Translate

1.1.4 Inglés Traductor

Número de votos: 1.290 **Puntuación:** 4,1 **Instalaciones:** 0,5-1M



Aplicación de diccionario sencilla que permite la traducción de Inglés a Español y viceversa, la entrada por voz, reproducir las traducciones a audio y enviar las traducciones por email, SMS, Twitter y/o Facebook.

Precio: Gratis

Ilustración 4 Ingles Traductor

1.1.5 El traductor de viaje

Número de votos: 129 **Puntuación:**4,1 Instalaciones:5.000-10.000

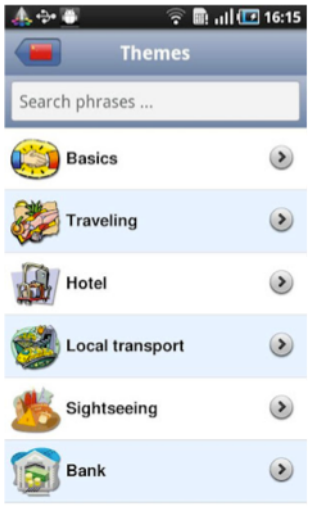


Ilustración 5 - Captura de "el traductor de viaje"

Manual de conversación ilustrado y parlante que traduce palabras y expresiones españolas a 29 idiomas. Los contenidos están divididos en áreas temáticas que permiten al usuario encontrar las frases apropiadas a cada situación de la manera más rápida y eficaz. Además cuenta con un buscador para realizar las consultas rápidamente.

Precio: Gratis

1.2 Servicios de traducción en Internet

Para el desarrollo del proyecto es necesario utilizar un servicio externo que a partir de una palabra dada retorne la traducción de la misma y, a ser posible, cualquier otro tipo de información relacionada con la palabra original o la traducida.

Para ello se han estudiado los principales servicios existentes en Internet y de cada uno se ha extraído su funcionamiento y las condiciones que impone.

1.2.1 Google Translate

Google Translate es quizás la plataforma de traducción más conocida y con más funcionalidades que existe, además del entorno sencillo de traducción en el que el usuario puede traducir textos, sugerir traducciones, reproducir los textos originales y los traducidos en cualquier idioma.

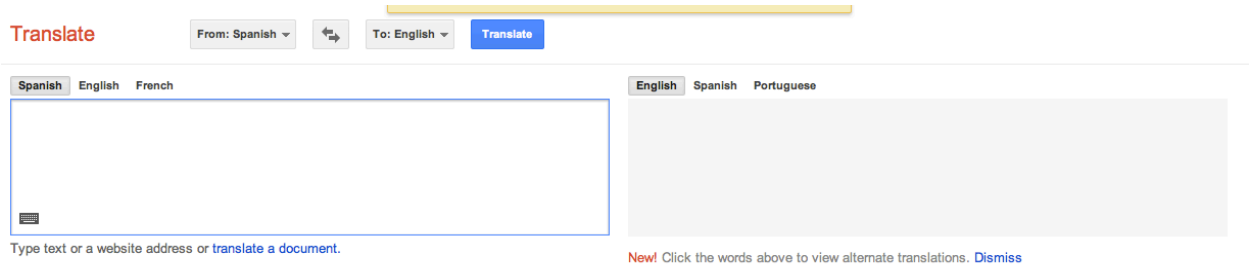


Ilustración 6 Captura de pantalla de Google Translate

Hay que destacar que Google Translate ofrece al usuario otras prestaciones como: la traducción de documentos, la creación de un glosario personalizado, la posibilidad de traducción instantánea a través del chat Google Talk, etc... Todas ellas de uso gratuito inicialmente pero con ciertas limitaciones. Además, ofrece al usuario la posibilidad de modificar la traducción presentada para re-alimentar la base desde la que se generan las traducciones.

En cuanto a APIs de desarrollo, hasta el 26 de mayo de 2011, se podía acceder a una versión gratuita y de fácil uso que permitía, entre otras funcionalidades, convertir la una web en multi-idioma con muy poco trabajo. Esto generó un uso excesivo y quizás incorrecto de esta API lo que provocó que dejara de ofrecerse de manera gratuita.

La versión V2 de la API de Google Translate, sólo está disponible como versión de pago, con un precio de 20$ por cada millón de caracteres traducidos. En este caso, la comunicación entre servidor (API) y aplicación (cliente) se puede realizar con *JSON* y se trabaja sobre protocolo REST[1].

1.2.2 WordReference.com

WordReference es un diccionario de traducción en línea con el que se pueden trabajar los siguientes pares: Inglés-Francés, Inglés-Italiano, Inglés-Español, Francés-Español, Español-Portugués e Inglés-Portugués, y también desde junio de 2009 Inglés-Alemán, Inglés-Ruso, Inglés-Polaco, Inglés-Rumano, Inglés-Checo, Inglés-Griego, Inglés-Turco, Inglés-Chino, Inglés-Japonés, Inglés-Coreano e Inglés-Árabe, a pesar que algunos de estos pares aún se encuentran en proceso de implementación.

A parte del diccionario de traducción, este sitio web cuenta con una sección de foros en la cual el usuario registrado puede realizar preguntas de vocabulario y gramática.

La API que ofrece esta web, se puede realizar a través de una solicitud asíncrona en la que se incluye el par de idiomas con que se quiere trabajar y el texto a traducir, retorna un objeto JSON con los resultados de la petición. O también casi de la misma manera pero con un retorno en formato HTML.

---

[1] **REST** es una técnica de arquitectura del software para sistemas hipermedia distribuidos como Internet. Se usa en el sentido más amplio para describir cualquier interfaz web simple que utiliza XML y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP.

En ambos casos, se ha de generar una URL donde realizar la petición que ha de seguir este patrón:

Para HTML:
http://api.wordreference.com/{version}/{API_key}/{dictionary}/{term}
Para JSON:
http://api.wordreference.com/{version}/{API_key}/json/{dictionary}/{term}

Siendo:
versión: el número de versión con el que se trabaja, actualmente 0.8.

api_key: Es un identificador que proporciona WordReference para poder trabajar con su API.

Para solicitarlo es necesario acceder a la página de registro de WordReference en: (http://www.wordreference.com/docs/APIregistration.aspx) donde se ha de rellenar un formulario y a continuación proporcionan una **api_key** con la que se puede trabajar.

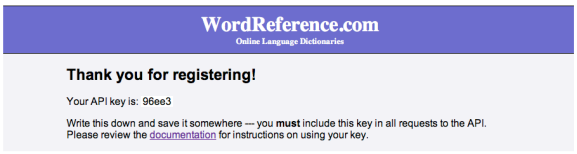


Ilustración 7 Proceso de obtención de la API de WordReference.com

dictionary: Con este parámetro se define el par de idiomas de los comentados anteriormente con el que se quiere trabajar. Dicho par se establece con la abreviatura de los nombres de los dos idiomas en estándar ISO sin espacio de separación y en minúsculas. Así traducir de Español a Inglés sería **"esen"**, de Inglés a Francés **"enfr"**, etc...

Actualmente la API del par Español-Ingles, se encuentra en desarrollo, y Wordreference espera tenerla publicada para primavera de 2012.

term: Este parámetro contiene la palabra a traducir.

Con todo lo anterior, si se quisiera traducir la palabra "dog" de inglés a francés, se generarían las URLs:

http://api.wordreference.com/0.8/96ee3/enfr/dog para HTML y

http://api.wordreference.com/0.8/96ee3/json/enfr/dog para JSON.

<table>
<tr><th>HTML</th><th>JSON</th></tr>
<tr>
<td>
WordReference.com Dictionaries
English-French Go
Principal Translations/Principales traductions
<table>
<tr><td>dog n
(pet)</td><td>chien nm</td></tr>
<tr><td>dog vtr
(hound, harass)</td><td>tracasser⇒ vtr</td></tr>
</table>
Additional Translations/Traductions supplémentaires
<table>
<tr><td>dog n
(canis familiaris)</td><td>chien nm</td></tr>
<tr><td>dog n
(animal)</td><td>canidé nm</td></tr>
<tr><td>dog n
(contemptible person)</td><td>traduction non disponible</td></tr>
<tr><td>dog n
US informal (friend) figuré</td><td>frère nm</td></tr>
<tr><td>dog n
slang (unattractive woman) femme laide : argot</td><td>thon nm</td></tr>
<tr><td>dog n
informal (sthg worthless)</td><td>traduction non disponible</td></tr>
<tr><td>dog n
informal (failure)</td><td>échec nm</td></tr>
<tr><td>familier</td><td>flop nm</td></tr>
<tr><td>dog vtr
(chase with dogs)</td><td>lâcher les chiens vi</td></tr>
<tr><td>dog vtr
(follow) suivre</td><td>filer⇒ vtr</td></tr>
</table>
</td>
<td>
<pre>
{
"term0" : {
"PrincipalTranslations" : {
      "0" :{
            "OriginalTerm" : { "term" :
"dog", "POS" : "n", "sense" : "pet",
"usage" : ""},
            "FirstTranslation" : {"term"
: "chien", "POS" : "nm", "sense" : ""},
"Note" : ""},
      [...]
            "FirstTranslation" : {"term"
: "tracasser", "POS" : "vtr", "sense" :
""}, "Note" : ""}},
"AdditionalTranslations" : {
      "0" :{
            "OriginalTerm" : { "term" :
"dog", "POS" : "n", "sense" : "canis
familiaris", "usage" : ""},
            "FirstTranslation" : {"term"
: "chien", "POS" : "nm", "sense" : ""},
"Note" : ""},
[...]
</pre>
</td>
</tr>
</table>

1.2.3 Apertium

Ilustración 8 Captura pantalla Apertium

http://www.apertium.org/

Apertium, es una plataforma de traducción de código abierto, desarrollada por la empresa **Prompsit** en colaboración con la Universidad de Alicante, la Generalitat de Catalunya, el Ministerio de Asuntos Exteriores de Rumania y el Ministerio de Industria, Turismo y Comercio de España.

En esta plataforma se puede encontrar una sencilla herramienta de traducción online que permite escoger el par de idiomas a traducir (origen-destino) y la caja del texto a traducir.

1.2.4 Bing Translator:

Este es el traductor ofrecido por Microsoft dentro de su plataforma online Bing para hacer competencia al de Google. En su aplicación web se puede encontrar un formulario similar al de Google Translate con el cual, el usuario puede realizar traducciones de manera muy similar al ya comentado.

Además, igual que en el caso de Google Translate, ofrece una serie de herramientas para el desarrollo de aplicaciones y páginas web, aplicaciones móviles, etc…y al igual que el anterior, esta API tiene un coste, que no va en función de caracteres traducidos, sino que se calcula en función de las traducciones realizadas, y se vende por packs:

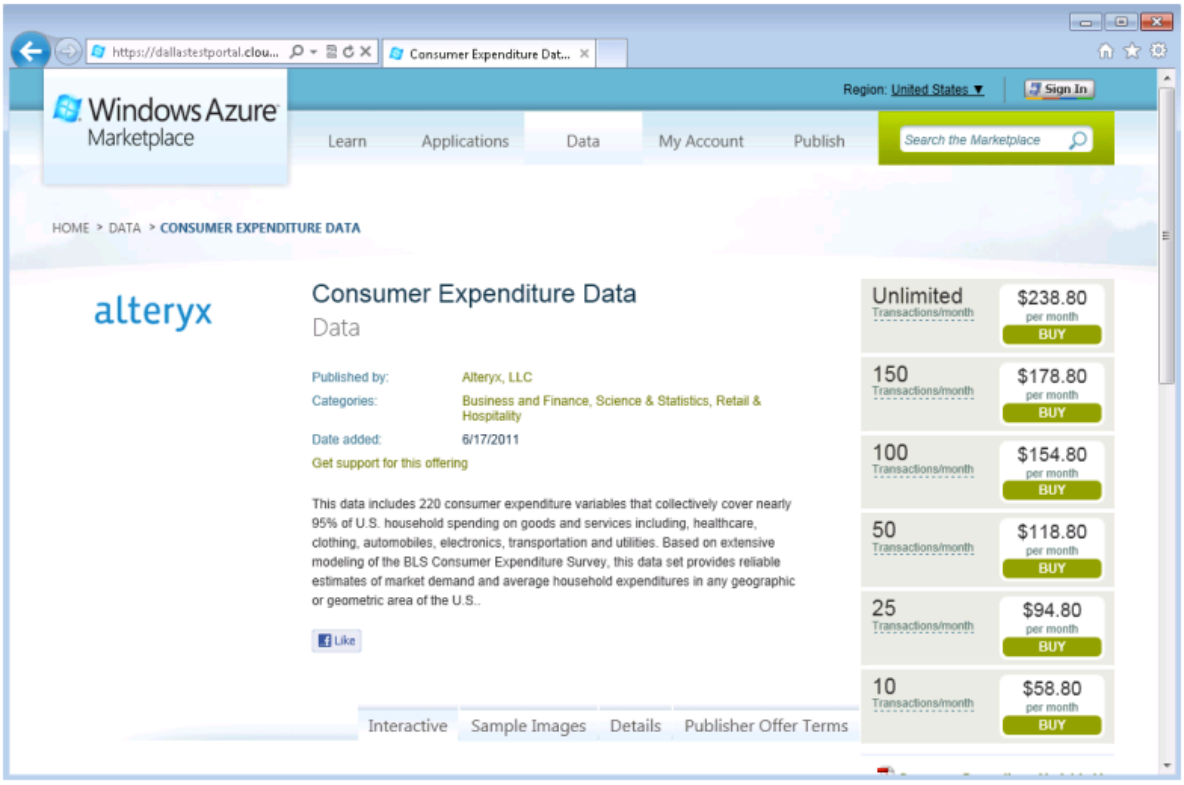


Ilustración 10 Precio Packs API Bing Translator

1.2.5 Yahoo babel fish:

Otro de los servicios de traducción online más importantes que existe, y quizás uno de los más utilizados junto con el de Google, es el servicio de la empresa Yahoo.

Esta empresa en sus inicios se orientó a público americano y latino pero actualmente, es una de las empresas de su sector con más éxito en Asia. Es por ello que este servicio también está orientado a público asiático a pesar de que la plataforma sea multilenguaje.

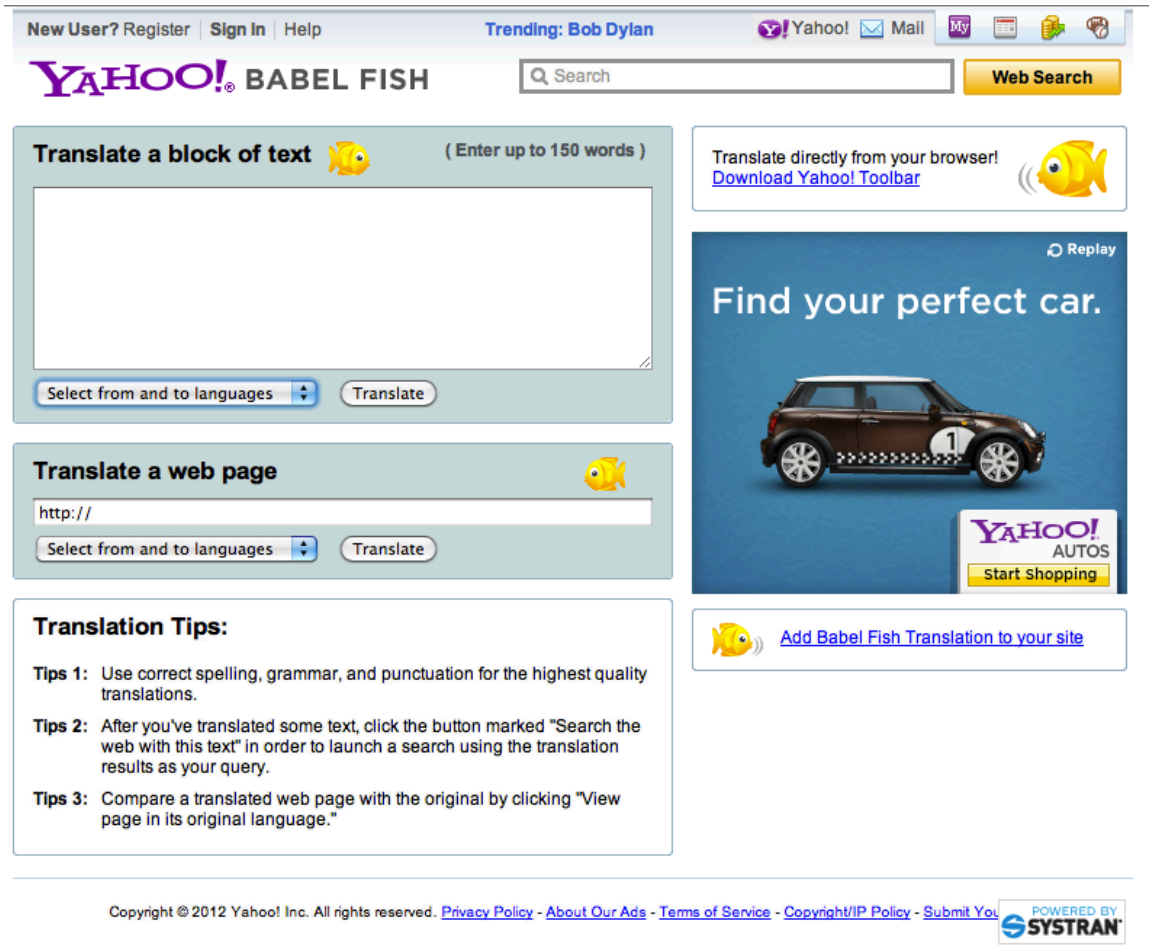


Ilustración 11 - Captura de pantalla de Babel fish

Esta herramienta, a diferencia de las anteriores, no incluye una API para desarrolladores propia, sino que dicha API está incluida dentro del pack que ofrece Yahoo, conocido como **YDN** (Yahoo Developer Network) conocida como **R3**.

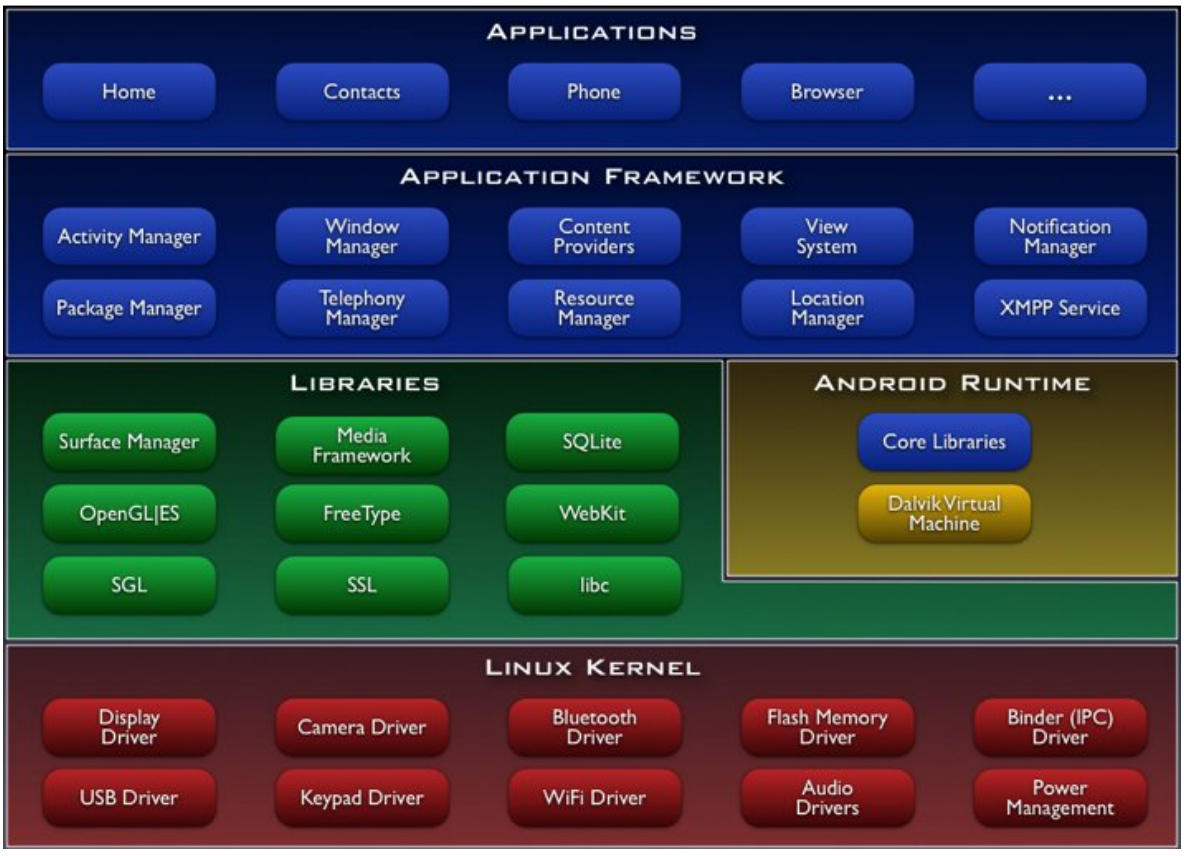
R3, es una herramienta OpenSource, bajo licencia FreeBSD, para mantener y desarrollar variantes de páginas webs creando, gestionando y localizando plantillas y traducciones. Esto se puede hacer utilizando la línea de comandos o entorno web, con lo que se generan archivos de texto para utilizarlos en aplicaciones internacionalizadas.

Al ser una herramienta *OpenSource*, el precio por traducción es gratuito, pero la desventaja de este servicio es que no se puede utilizar en aplicaciones *real time*, en entorno móvil, ya que todas las traducciones dependen unos archivos generados que una vez obtenidos se pueden incluir en la configuración de la aplicación.

1.3 Sistema Operativo Android

Este es un sistema operativo orientado a dispositivos móviles tales como tabletas, *smartphones*... Fue inicialmente desarrollado por la empresa Android Inc, que en 2005 fue comprada por Google y actualmente es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio.

La arquitectura de este sistema operativo, está distribuida en varias capas, cada una de ellas tiene sus propias características y propósito. Estas capas son: Aplicaciones, Marco de trabajo de las aplicaciones, Bibliotecas, *Runtime* de Android y Núcleo Linux.



1.3.1 Kernel de Linux

Linux es un sistema de alta seguridad que ha sido probado y testeado en múltiples entornos desde hace décadas. Es por ello que Android ha escogido Linux como base para su sistema operativo. A pesar de todo, las aplicaciones de Android se ejecutan de manera independiente de los procesos Linux que corren en el sistema.

1.3.2 Librerías

Las librerías nativas de Android, son en su mayoría programadas en C/C++ y a menudo vienen de la comunidad OpenSource con la intención de ofrecer los servicios necesarios para la capa de aplicación de Android. Entre otros, incluye:

- **WebKit**, para el renderizado web que utilizan los navegadores como chrome, Safari,…
- **SQLlite,** un sistema completamente funcional de base de datos SQL.
- **Apache Harmony,** una implementación OpenSource de Java.
- **OpenGL,** un paquete de librerías de gráficos 3D.
- **OpenSSL,** una librería para la conexión y navegación en Internet de manera segura.

1.3.3 Application Framework

El *Application Framework*, es un entorno de desarrollo que provee de numerosos servicios para ayudar al desarrollador. Esta es la parte de la plataforma con mayor y mejor documentación ya que es a partir de las herramientas aquí proporcionadas con las que el

desarrollador puede explotar su creatividad y así lanzar cada día las nuevas funcionalidades o aplicaciones que se han comentado anteriormente.

Precisamente en esta capa se pueden encontrar múltiples librerías en Java especialmente creadas para Android, además se pueden encontrar "*services*" que permiten a la aplicación hacer uso de funcionalidades del *Smartphone*, tales como sensores, WiFi, servicio telefónico, acceso a información de otras aplicaciones,…

1.3.4 Aplicaciones Android

Y por último, la capa de aplicaciones en Android. Estas aplicaciones se desarrollan habitualmente en lenguaje Java junto con *Android Software Development Ki*t (Android SDK), pero es posible desarrollar con otras herramientas como el Kit de Desarrollo Nativo para aplicaciones o extensiones en C y C++ o Google App Inventor.

Las aplicaciones a su vez se encuentran distribuidas en *Activities*, que hacen referencia a los casos de uso de la aplicación. Es decir, se podrían definir las *activities* como cada una de las pantallas de una aplicación.

Para entender la navegación dentro de una aplicación en Android, cabe definir la diferencia entre *Intent* y *Activity*. Las *Activities*, como ya se ha comentado, son básicamente las pantallas de las que se compone una aplicación. Los *Intents*, son la manera de invocar estas *activities*. La definición breve de la documentación es: "*Un intent es la descripción abstracta de una operación que se va a llevar a cabo*". O dicho de otro modo, un *intent* es una clase que permite especificar una *Activity* a ejecutar, llamando a uno de los métodos de la clase *Activity* con ese *Intent* de parámetro.

Cuando una *activity* es ejecutada, dentro de ésta hay un proceso de ejecución que es imprescindible conoce para el desarrollo de cualquier aplicación. Ya que con éste se puede definir que métodos se ejecutarán la primera vez que se instancie una pantalla, cuales se pueden mantener en ejecución en segundo plano cuando una aplicación esté pausada o cuales se ejecutarán cuando el usuario vuelva a una pantalla ya instanciada.

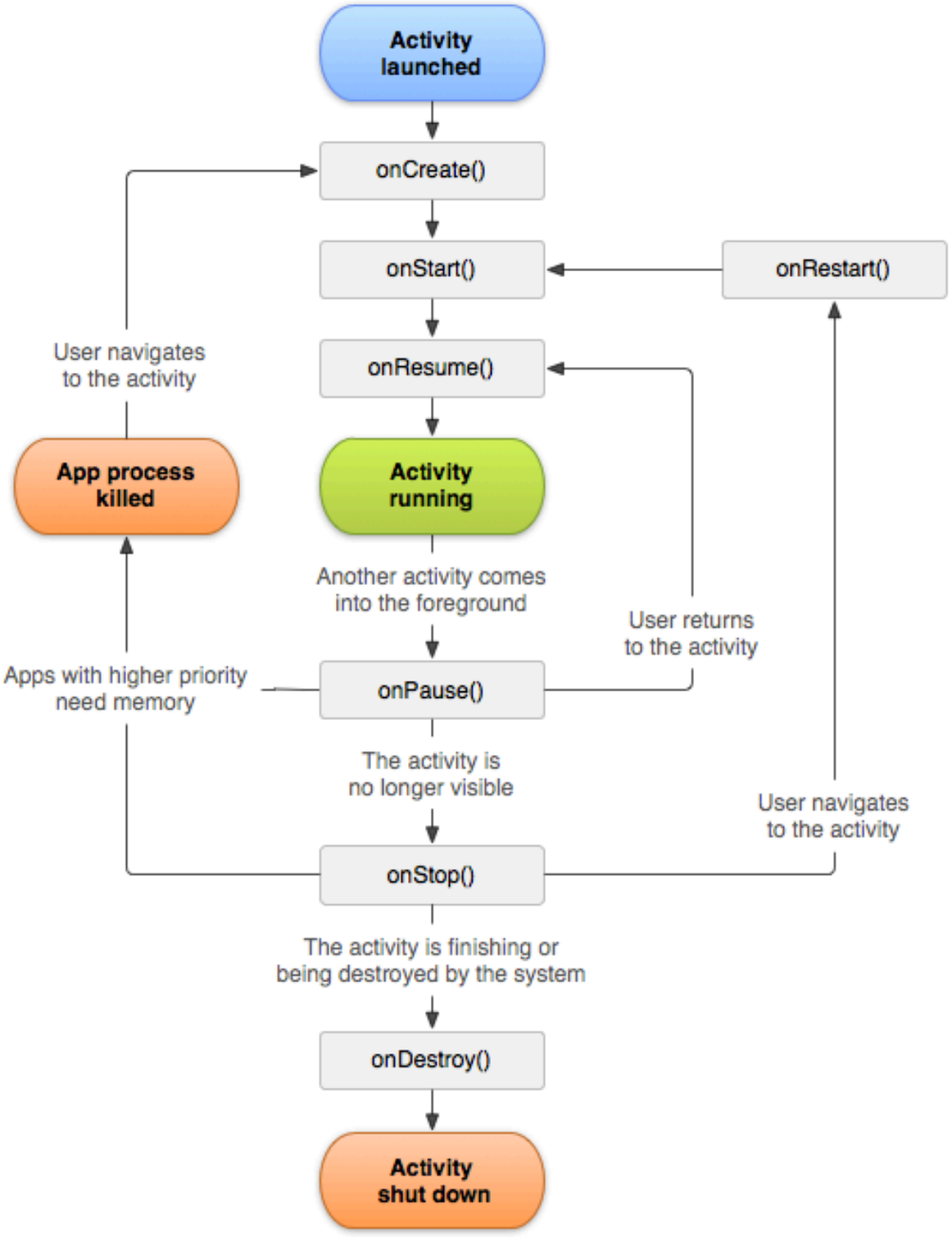


Ilustración 12 - Proceso de ejecución de una aplicación Android

1.4 Modelos de negocio con aplicaciones móviles

En el mundo de las aplicaciones móviles, se pueden encontrar muchas opciones en cuanto a modelos de negocio. Es, por tanto, muy difícil definir todos ellos ya que además algunos se pueden considerar incluso irrelevantes al estar muy ligado a una aplicación en concreto.

Una de las consecuencias de la aparición a posteriori de las otras plataformas de aplicaciones móviles (*Market*), es el hecho de que para ganar público en estos entornos se ha tenido que ofrecer las aplicaciones a precios más bajos que en el caso de App Store e incluso en algunos casos ya no se ofrece la versión de pago fuera de este entorno.

Es por ello que se han seleccionado los cuatro modelos de negocio que generan más ingresos en el mundo de las aplicaciones móviles. Además, en el caso de este proyecto, al ser orientado únicamente al mercado Android, únicamente se van analizar estos modelos dentro de este mercado.

1.4.1 Venta directa en el Market

Este es el modelo de negocio más expandido sobretodo en el mercado de aplicaciones para iPhone ya que es un mercado más acostumbrado a comprar dentro de "*App Store*". A pesar de ello, es un modelo con mucho sentido sobretodo para aplicaciones exclusivas y de difícil competencia.

Una opción común a este modelo, es el hecho de publicar una aplicación limitada con o sin publicidad de manera gratuita. En ésta, si el usuario quiere obtener las funcionalidades extra tiene que comprar la versión de pago.

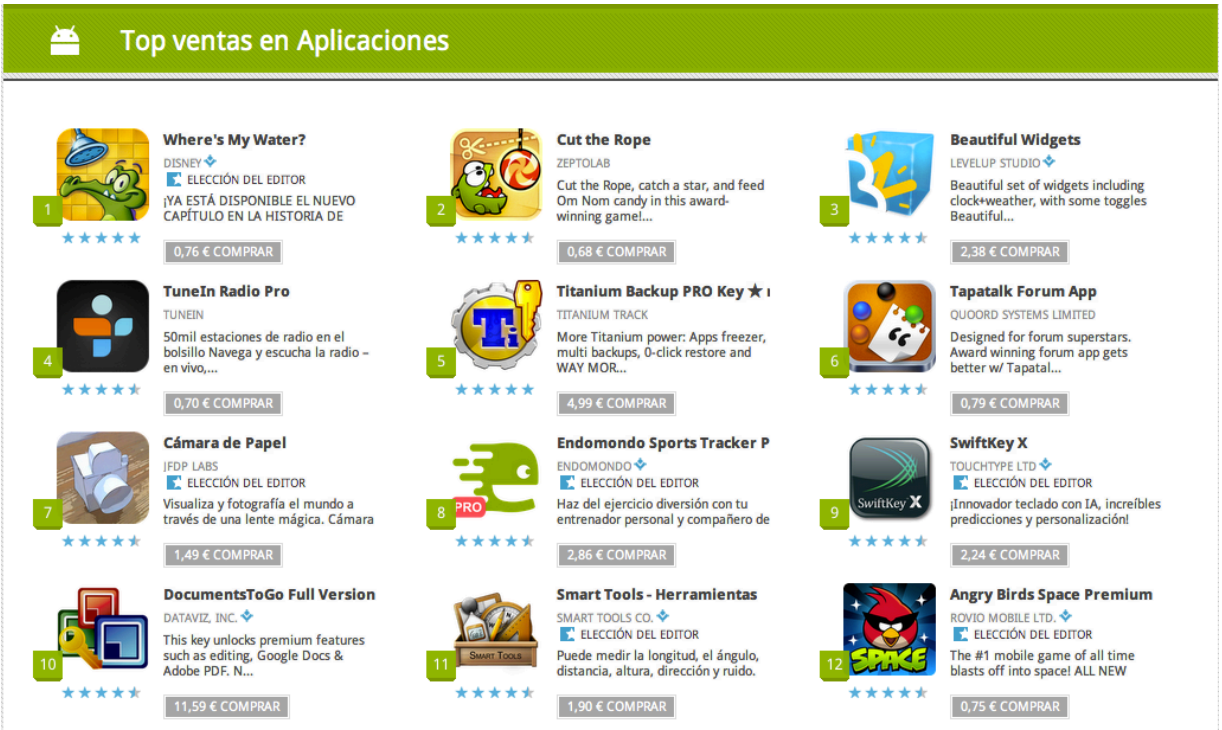


Ilustración 13 Ejemplo de aplicaciones orientadas a venta

Como se puede observar en la imagen anterior, los precios dentro de las aplicaciones de pago, pueden variar de céntimos de euro como la primera (0,76€), hasta por ejemplo la aplicación **DocumentsToGo Full Version** que tiene un precio de 11,59€.

1.4.2 Publicidad

Este modelo, es quizás el más utilizado de los que se van a comentar. En este caso, se trata normalmente de publicar una aplicación de manera gratuita, en la que en diferentes momentos aparece un banner publicitario anunciando normalmente otras aplicaciones o algún producto que por perfil de usuario, a éste le pueda interesar. Es decir, la plataforma intenta recoger cuantos más datos mejor del usuario que está utilizando una aplicación y de este modo ofrecerle productos y/o servicios que le puedan interesar.

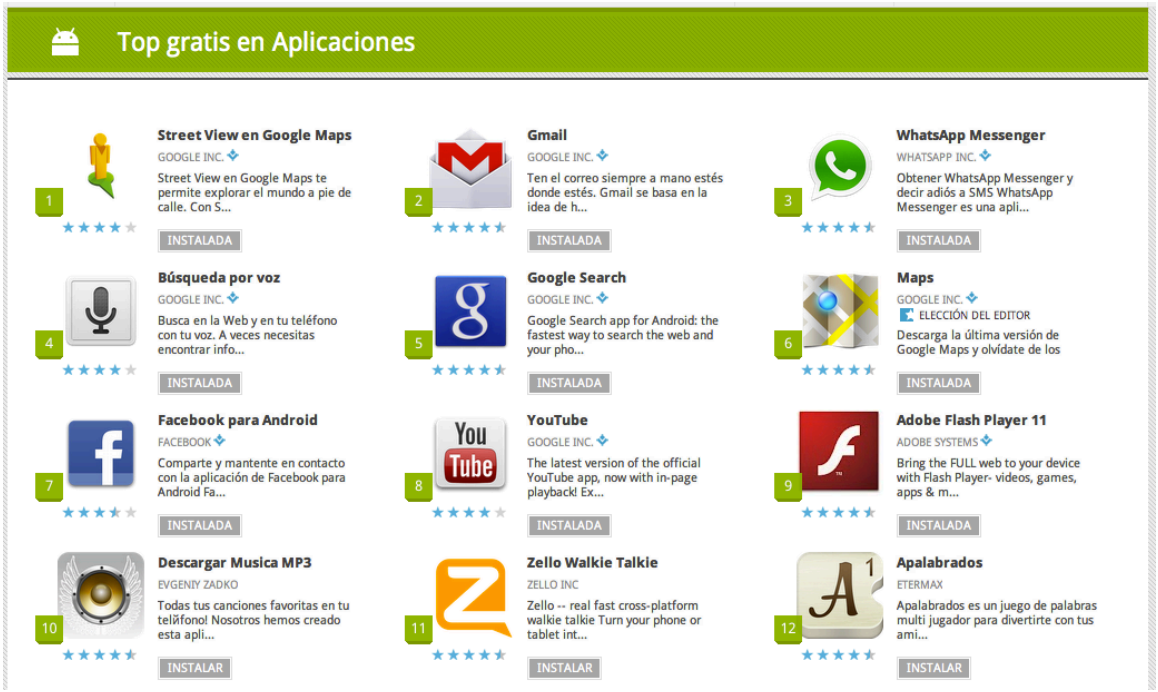


Ilustración 14 Ejemplos de aplicaciones gratuitas

Para poder añadir publicidad a una aplicación, se hace normalmente mediante una plataforma intermedia que se encarga de localizar por un lado a los anunciantes y por el otro a los desarrolladores para poner la publicidad de los primeros en las aplicaciones de los segundos y cobrando un porcentaje por la gestión.

AdMob

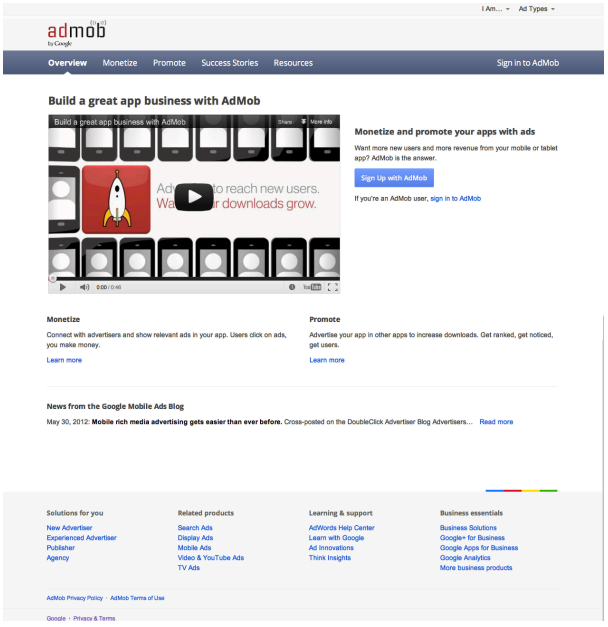
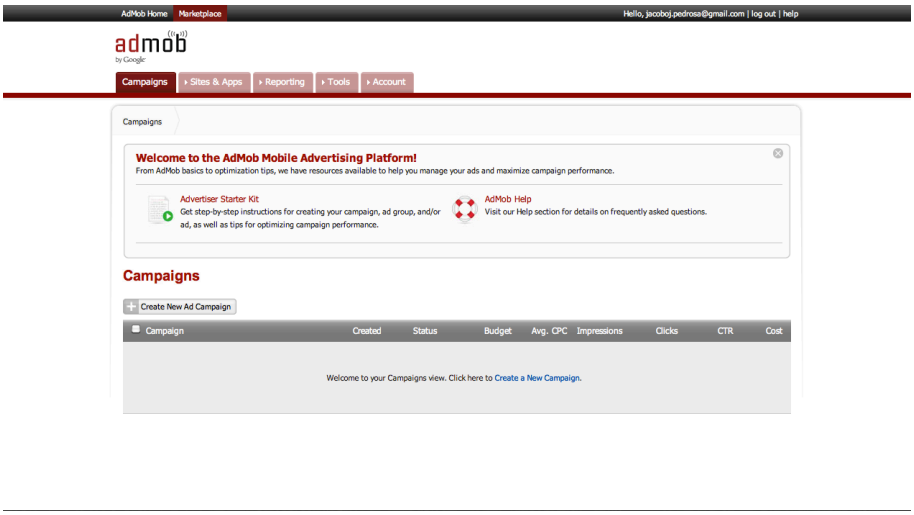


Ilustración 15 Portada de AdMob.com

Esta plataforma es de todas las relacionadas con Android para la gestión de publicidad en aplicaciones móviles, la más conocida. Además, fue una de las primeras en aparecer y ha sido comprada por Google en 2009.

Para gestionar la publicidad con esta plataforma, el desarrollador tiene que incluir una librería dentro de su proyecto y definir con la API que ofrece donde y

cuando va a aparecer la publicidad dentro de la aplicación.



De cara al anunciante, una vez dentro de la plataforma se encuentra en un entorno sencillo con el que dar de alta una campaña y gestionarla.

Ilustración 16 Plataforma AdMob.com

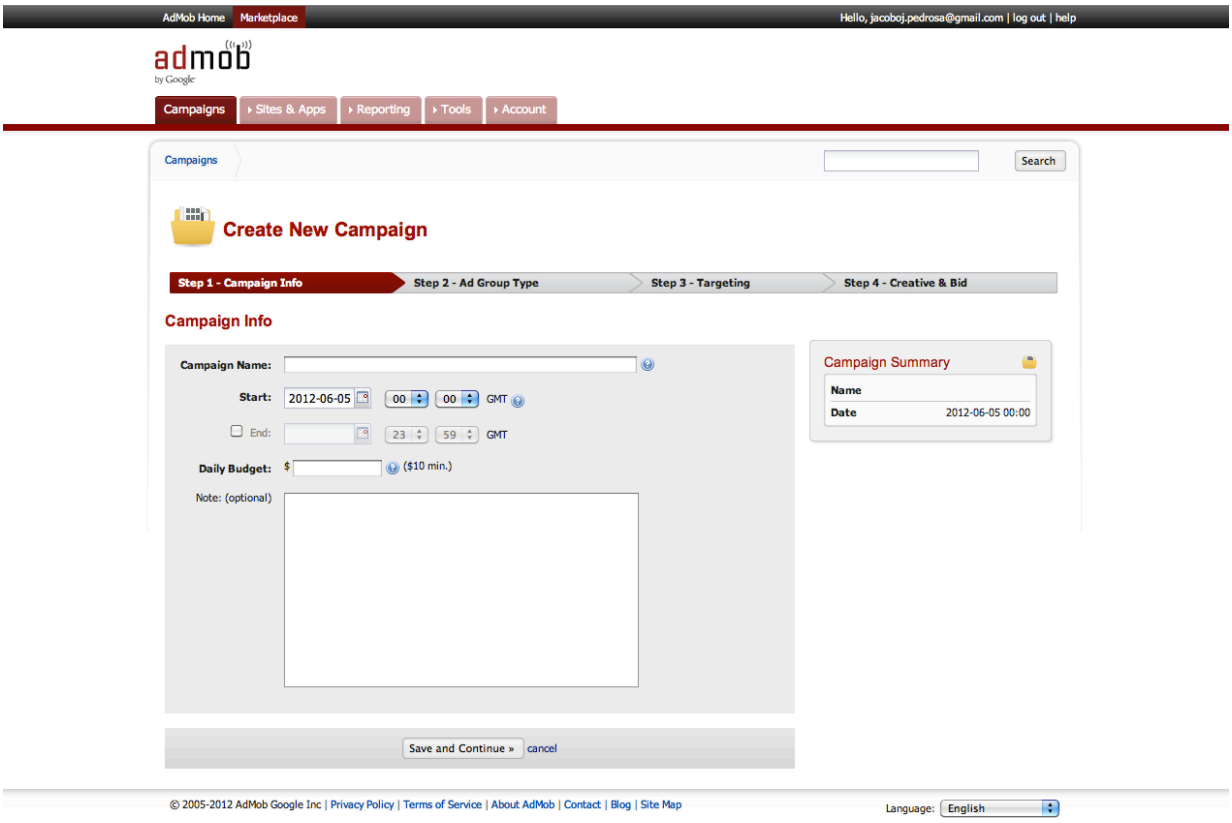
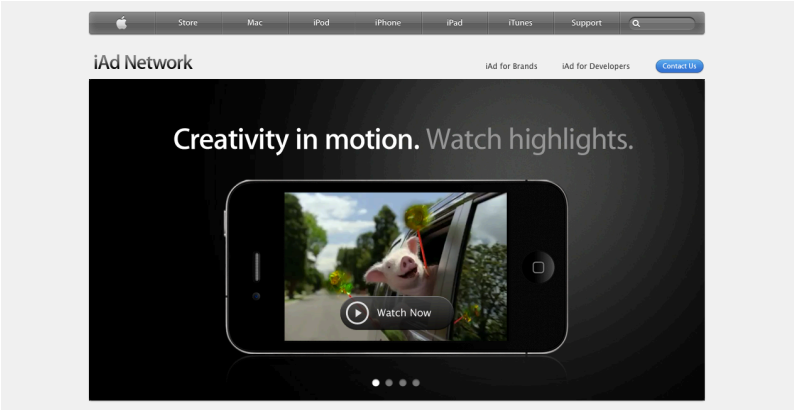


Ilustración 17 Gestión de campañas de publicidad en AdMob

iAd

Otra de las plataformas más reconocidas en cuanto a publicación de anuncios en aplicaciones móviles, se llama iAd.



Esta es la plataforma publicitaria nativa de Apple, en la que de la misma forma que la anterior, se ofrece un entorno para desarrolladores y otro para anunciantes.

La limitación de esta plataforma es el hecho de que es únicamente para aplicaciones iPhone/iPod/iPad, por lo que no va a ser más comentado en este proyecto.

1.4.3 Servicios

Otro de los modelos de negocio alrededor de las aplicaciones móviles, se basa en que la aplicación no sea el centro del propio modelo, sino que sirva como soporte al mismo. Es decir, en este caso, el desarrollador o empresa que crea la aplicación, tiene un modelo de negocio ya montado o lo quiere montar en el cual, la aplicación es simplemente una herramienta de soporte para el mismo.

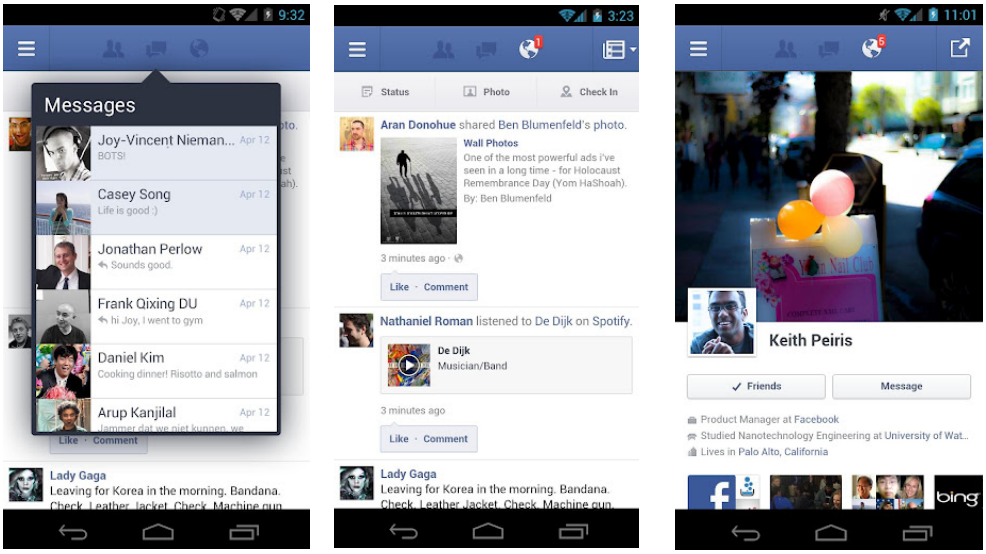


Ilustración 18 - Capturas de la aplicación oficial de Facebook

En este caso, destacan las aplicaciones relacionadas con las redes sociales, como podría ser la aplicación oficial de Facebook, en la que el modelo de negocio no se encuentra ni en la venta de la aplicación, ni en poner publicidad en la misma, sino que es un modelo de negocio externo a la aplicación y ésta únicamente hace de soporte a la plataforma original.

1.4.4 Ventas IN-APP

Por último el modelo de negocio que más ingresos está dando actualmente, y es el de las ventas IN-APP, es decir, dentro de la aplicación.

En este caso, el desarrollador o empresa desarrolladora define, en la mayoría de los casos, una moneda imaginaria u objeto de intercambio que se puede comprar con dinero real. Y a continuación, el usuario puede utilizar este dinero imaginario para comprar objetos o servicios dentro de la aplicación.

Esto ocurre sobretodo en juegos en los que el usuario si quiere puede evolucionar a los personajes más rápidamente de lo normal, puede hacerlo comprando elementos que por la definición del juego tardaría más en obtener.

Ilustración 19 - Captura juego "little empire"

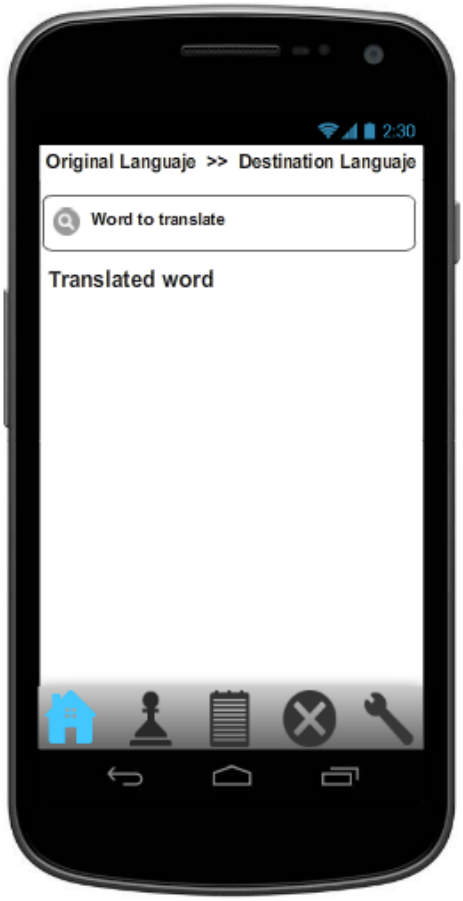
Un claro ejemplo de este modelo de negocio, es el juego "*Little Empire*". Este es un sencillo juego de rol, en el cual el usuario se crea un reino y lo tiene que hacer crecer invadiendo reinos vecinos y protegerse de invasiones vecinas también.

En este juego, el usuario puede comprar unos objetos llamados "mojos" con los que puede comprar armas y armaduras a su "Héroe" dentro del juego.

1.5 Descripción de la aplicación

Previamente al desarrollo o al diseño de la aplicación, se ha decidido realizar una serie de maquetas a modo de boceto (*mockups*) en las que únicamente se tratará la funcionalidad de cada pantalla. De este modo se centra el proyecto en cuales son los elementos mínimos a desarrollar en cada pantalla.

1.5.1 Inicio/Buscador



La pantalla principal de la aplicación se define como el buscador, de manera que el usuario al cargar la aplicación, puede realizar búsquedas de manera rápida y cómoda. De este modo se pretende conseguir que se hagan muchas búsquedas con la aplicación y así poder ofrecer al usuario más variedad en los juegos. Por tanto, en esta pantalla aparecen los elementos:

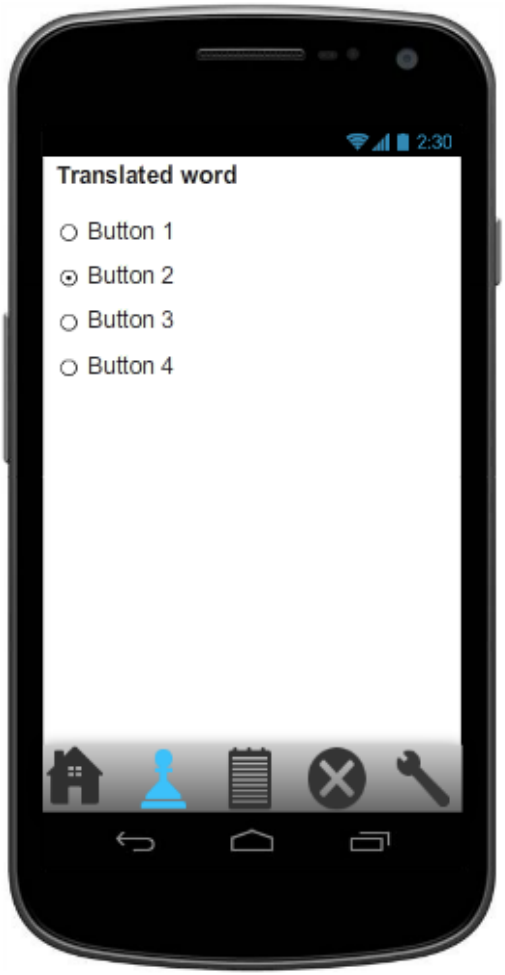
Par de idiomas: Donde aparece el idioma original en que se realiza la búsqueda y el idioma de destino en el que se mostrará la respuesta.

Palabra a buscar: Es un campo de texto en el que el usuario al clicar con el dedo encima le aparecerá el teclado y a continuación podrá escribir una palabra.

Botón buscar: Con este botón, el usuario activa la búsqueda en caso de que haya alguna palabra escrita en el campo anterior.

Resultado: En este campo, se muestra la palabra una vez traducida, si no se encuentra ningún a traducción para dicha palabra, mostraría un mensaje de error.

1.5.2 Juego GameTest



Inicialmente se define un solo juego base para trabajar con las principales características que se quieren tratar con el usuario: Ofrecerle una palabra de las consideradas no conocidas o a memorizar y si se resuelve correctamente el ejercicio se resta 1 en el atributo *numCheck* de la palabra.

En este caso se ofrece una palabra al azar de entre las palabras erróneas y 5 posibles respuestas también de entre las erróneas. El usuario tiene que escoger la respuesta correcta. Cuando una palabra alcanza 0 en el atributo ***numCheck***, aparece un mensaje diciendo que la palabra se considera como aprendida hasta que se vuelva a buscar. Por tanto, en esta pantalla existen los siguientes elementos:

Palabra enunciada: Es la palabra sobre la que se realiza el ejercicio.

Respuestas: Se muestran 5 posibles respuestas a la palabra inicial en forma de test, si el usuario selecciona la respuesta correcta, le aparece un mensaje de "correcto" automáticamente se reinicia el ejercicio con nuevos valores. En caso contrario se le muestra el mensaje de error.

1.5.3 Histrial palabras buscadas

En esta *activity* se ofrece al usuario un historial de palabras buscadas, la idea es que poco a poco pueda hacerse un diccionario personalizado con aquellas palabras con las que ha tenido dudas. Dicho historial estará disponible offline para poder acceder a él en cualquier momento y lugar.

En este caso, solo se muestra una lista de palabras ordenadas alfabéticamente y con 4 datos en cada fila:

Palabra original: Palabra que ha sido introducida en el buscador.

Idioma original: Idioma en que se encuentra la palabra anterior.

Palabra traducida: Es la palabra en el idioma una vez traducida.

Idioma destino: Idioma en el que se obtiene la traducción anterior.

1.5.4 Palabras problemáticas

Esta *activity* es muy similar a la anterior, aunque se filtran únicamente las palabras con las que el usuario tiene problemas. De modo que pueda repasar esta lista y así mejorar su vocabulario.

Por lo que se mantienen los 4 datos en cada fila:

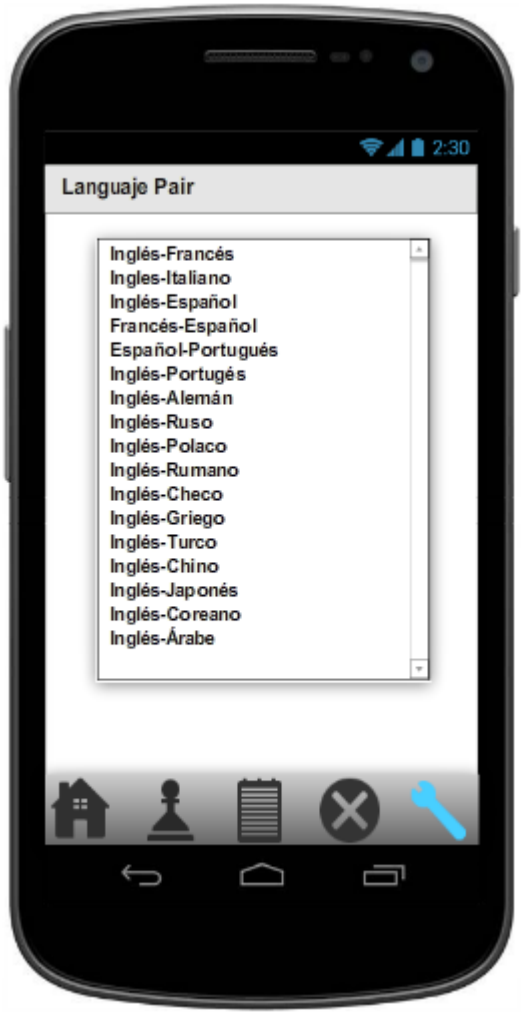
Palabra original: Palabra que ha sido introducida en el buscador.

Idioma original: Idioma en que se encuentra la palabra anterior.

Palabra traducida: Es la palabra en el idioma una vez traducida.

Idioma destino: Idioma en el que se obtiene la traducción anterior.

1.5.5 Configuración



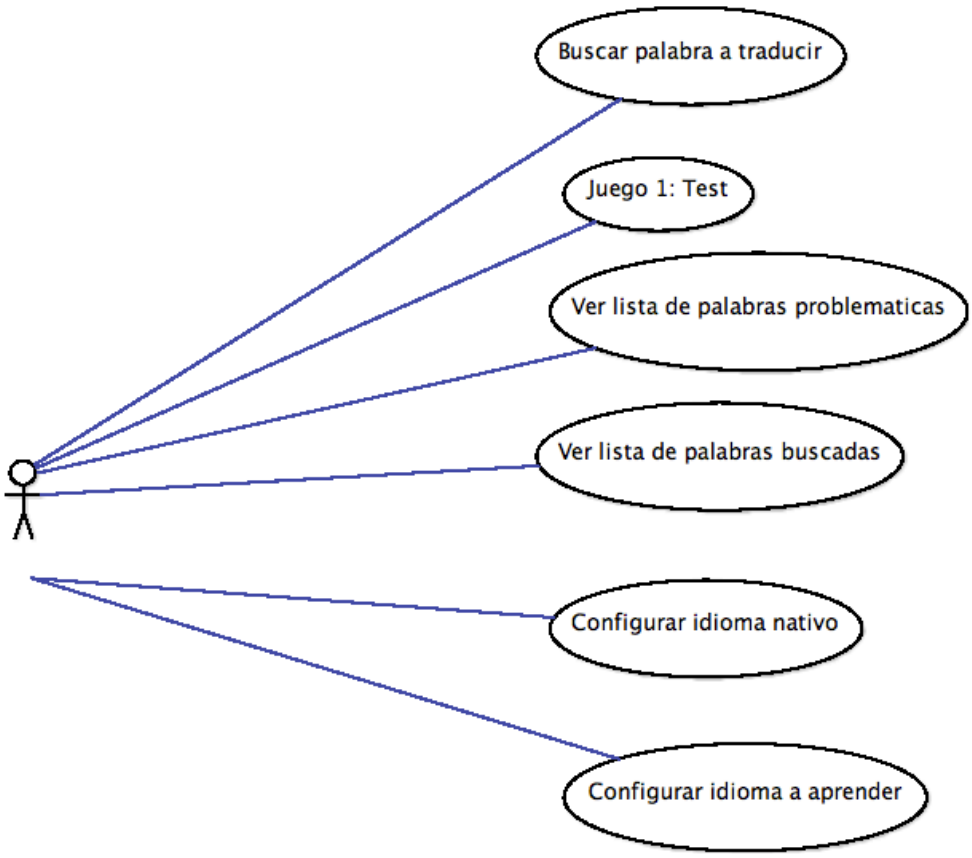
Por último, una pantalla que permita al usuario configurar diferentes opciones dentro de la aplicación. Dada la sencillez del juego y la aplicación en sí, no es necesario por el momento tener una gran configuración, por lo que en esta pantalla y por el momento, solo se puede editar el par de idiomas con los que se puede trabajar.

Esta lista de pares de idiomas viene definida por la capacidad de WordReference en su API, que irá escalando progresivamente y así se podría ofrecer poco a poco más opciones a los usuarios.

2 Análisis

A partir de lo documentado anteriormente, se define el proyecto como una aplicación en sistema operativo Android, en la que el usuario encontrará principalmente 5 funcionalidades a saber: traducir una palabra, jugar una partida a un juego a partir de palabras buscadas, ver lista de palabras buscadas, ver la lista de palabras no aprendidas, y configurar pareja de idiomas a utilizar.

2.1 Casos de Uso



2.1.1 Buscar palabra a traducir

Flujo inicial:

- El usuario entra en la aplicación.
- Presiona en el campo de texto de la pantalla con lo que le aparece un teclado.
- Escribe la palabra a traducir y aprieta en el botón para solicitar la traducción de la palabra introducida.
- La aplicación muestra dicha traducción.

Flujo alternativo:

- El usuario entra en la aplicación.
- Presiona en el campo de texto de la pantalla con lo que le aparece un teclado.
- Escribe la palabra a traducir y aprieta en el botón para solicitar la traducción de la palabra introducida.
- La palabra buscada no existe o no se encuentra en el servicio utilizado.
- La aplicación muestra el texto: "la traducción de la palabra "+palabra+" no está disponible en este momento, verifique que está correctamente escrita".

2.1.2 Juego1 (test)

Flujo inicial:

- El usuario entra en la *Activity* "Juego".
- Se muestra una palabra en idioma nativo o destino y 3 palabras en el idioma contrario.
- El usuario selecciona la traducción correcta de la palabra propuesta.
- Si la opción escogida es la traducción correcta de esta será marcada como acierto en la base de datos. A los 3 aciertos, esta palabra, será considerada como aprendida y no volverá a aparecer en el/los juegos de la aplicación.

Flujo alternativo:

- El usuario entra en la *activity* "Juego".
- Si no se ha realizado ninguna búsqueda o se todas las palabras de la base de datos han sido marcadas como aprendidas, se mostrará el texto "por favor realice alguna búsqueda".

Precondiciones:

- Haber realizado alguna búsqueda en la *activity* de búsqueda.

2.1.3 Ver lista de palabras problemáticas

Flujo inicial:

- El usuario entra en esta *activity* y se le muestra una lista de todas las palabras buscadas que no hayan sido marcadas como aprendidas.
- El usuario selecciona una palabra y aparece la traducción.

Flujo alternativo:

- Si no se ha realizado ninguna búsqueda o se todas las palabras de la base de datos han sido marcadas como aprendidas, se mostrará el texto "por favor realice alguna búsqueda".

Precondiciones:

- Haber realizado alguna búsqueda en la *activity* de búsqueda.

2.1.4 Ver lista de palabras buscadas

Flujo inicial:

- El usuario entra en esta *activity* y se le muestra una lista de todas las palabras buscadas.
- El usuario selecciona una palabra y aparece la traducción.

Flujo alternativo:

- Si no se ha realizado ninguna búsqueda o se todas las palabras de la base de datos han sido marcadas como aprendidas, se mostrará el texto "por favor realice alguna búsqueda".

Precondiciones:

- Haber realizado alguna búsqueda en la activity de búsqueda.

2.1.5 Configurar idioma nativo

Flujo inicial:

- El usuario accede a la *activity* de configuración y selecciona la opción de idioma nativo
- Se muestra un desplegable con las diferentes opciones a elegir.

Flujo alternativo:

- Si se produjese algún error durante el proceso, se mostrará un mensaje especificando el error.

2.1.6 Configurar idioma a aprender

Flujo inicial:

- El usuario accede a la *activity* de configuración y selecciona la opción de idioma a aprender
- Se muestra un desplegable con las diferentes opciones a elegir.

Flujo alternativo:

- Si se produjese algún error durante el proceso, se mostrará un mensaje especificando el error.

2.2 Modelo

En el desarrollo de este proyecto se pretende generar una base en la que a partir de una lista de palabras, que el usuario ha buscado su traducción de un idioma desconocido o en el que quiere ampliar su vocabulario a un idioma conocido, se le ofrezca uno o varios juegos con los que mejorar dicho vocabulario.

2.2.1 Base de datos

Dada la sencillez de la aplicación de este proyecto, su base de datos consta de una única tabla que hace referencia al elemento "palabra". Donde sus registros están formados por: palabra original (*word*), palabra traducida (*translated_word*), idioma original (*orig_lang*), idioma de traducción (*trans_lang*) y número de veces que quedan para considerar una palabra como aprendida (*numChecks*).

2.2.2 Clases del modelo

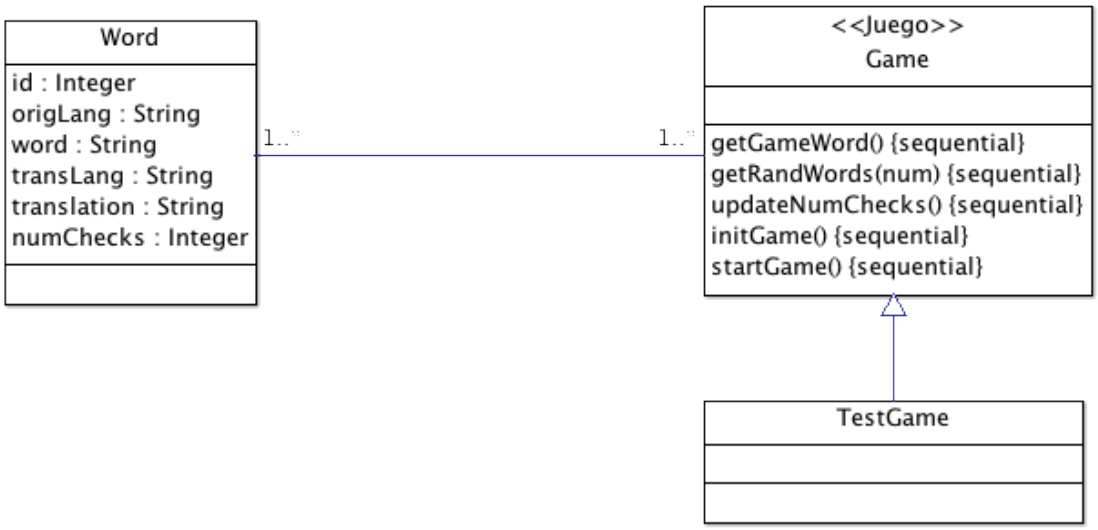


Ilustración 20 Diagrama de clases del modelo

El elemento más básico de la aplicación es una palabra (*word*), que está compuesta por su idioma original, en el que se busca ***origLang***, palabra buscada ***word*** en el idioma original, idioma a traducir o de destino ***transLang***, la palabra traducida ***translation***, y por último el número de veces que le queda a una palabra para ser considerada como aprendida cuando el atributo ***numChecks*** es igual a 0 y se le suma 3 cada vez que una palabra es buscada.

2.2.3 Clases de persistencia

A continuación se definen los dos elementos que hacen posible la gestión de las palabras y sus traducciones de esta aplicación, y son las clases *DatabaseHandler* y *WordReference*.

DataBaseHandler

Esta clase es la que realiza las gestiones relacionadas entre la base de datos y las "palabras":

- Guardar una palabra en la base de datos (*addWord*).
- eliminarla si fuese necesario (*deleteWord*).
- actualizar el campo de *numCkeck* de la palabra.
- (*updateWord*) ya que los otros atributos de la misma se considera han de mantenerse constantes.
- …

También es la encargada de crear las listas:

- todas las palabras buscadas (*getSearchedWords*) .
- todas las palabras problemáticas (getBadWords).

WordReference

Esta clase, se trata de una adaptación exclusiva para este proyecto que permite gestionar las peticiones realizadas a WordReference mediante el protocolo REST que ya se ha definido en el capítulo 1.

En si tiene un método principal *translate()* que dada una palabra sin traducir, realiza una petición y rellena los atributos del objeto *Word* con los datos obtenidos en formato JSON.

2.3 Diagrama de estados

Como se comenta en el punto anterior, la base del proyecto son las palabras buscadas, las cuales pueden no haber sido buscadas nunca, haber sido buscadas y se consideradas como aprendidas o haber sido buscadas y aparecer en los diferentes juegos.

En estos juegos, básicamente se ha de tener en cuenta en cada "partida" que palabra o palabras se van a aprender y el estado de esta palabra en relación al usuario. Si es una palabra que ha sido buscada frecuentemente se tendrá que resolver en más partidas que si es una palabra que ha sido buscada una sola vez. Para ello el atributo ***numChecks*** dentro de la clase **Word.** Este atributo se trata de un descontador que en caso de estar a 0 no aparecerá en los juegos y en caso de ser buscada dicha palabra aumentará en 3. Por ello, si el usuario busca la palabra "*DOG*" tres veces en diferentes momentos, el atributo tendrá el valor de 9. Y por cada vez que haya resuelto correctamente una partida con esta palabra, se descontará 1.

Otro punto a destacar, es que en cada juego ha de tener un número mínimo de palabras para que se pueda jugar, por ejemplo, en el caso del implementado en este proyecto, se ofrece una palabra y 5 opciones como posibles resultados por lo que como mínimo son necesarias 5 palabras para poder jugar. Aquí un ejemplo de funcionamiento básico.

Acción	**situación**
búsqueda "work"	numChecks "work" = 3; total palabras almacenadas = 1;
búsqueda "house"	numChecks "house" = 3; total palabras almacenadas = 2;
búsqueda "work"	numChecks "work" = 6; total palabras almacenadas = 2;
búsqueda "Pastimes"	numChecks "pastimes" = 3;

	total palabras almacenadas = 3;
búsqueda “weather”	numChecks “weather” = 3; total palabras almacenadas = 4
búsqueda “restaurant”	numChecks “restaurant” = 3; total palabras almacenadas = 5; Ya se puede jugar a “GameTest”
búsqueda “relation”	numChecks “relation” = 3; total palabras almacenadas = 6;
búsqueda “work”	numChecks “work” = 9; total palabras almacenadas = 6;
Partida a “GameTest” con la palabra “relation”	numChecks “relation” = 2 total palabras almacenadas = 6;

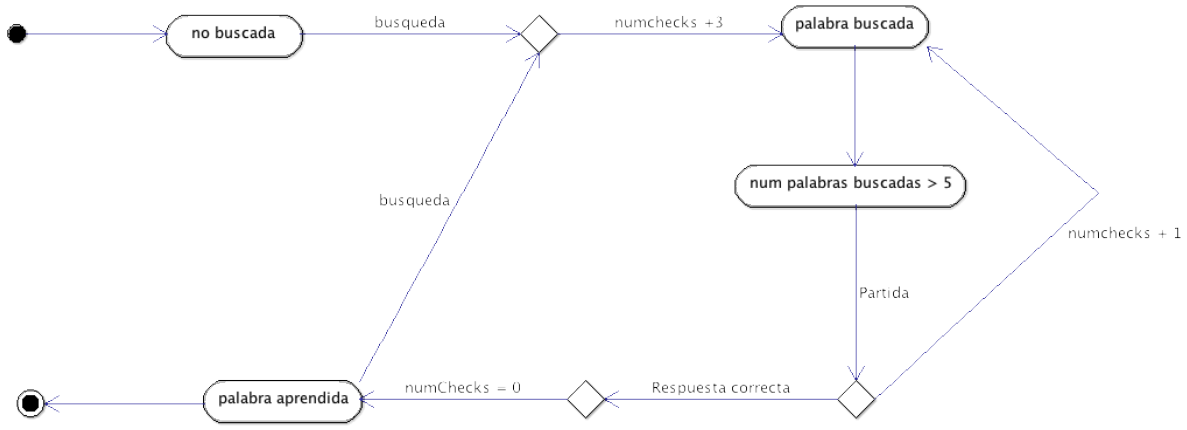


Ilustración 21 Diagrama de estados de una palabra

2.4 Diagramas de secuencia

2.4.1 Home/Search

En el diagrama de secuencia siguiente, se puede observar el proceso de funcionamiento la primera actividad en la que el usuario introduce una palabra en el buscador y obtiene la traducción de dicha palabra.

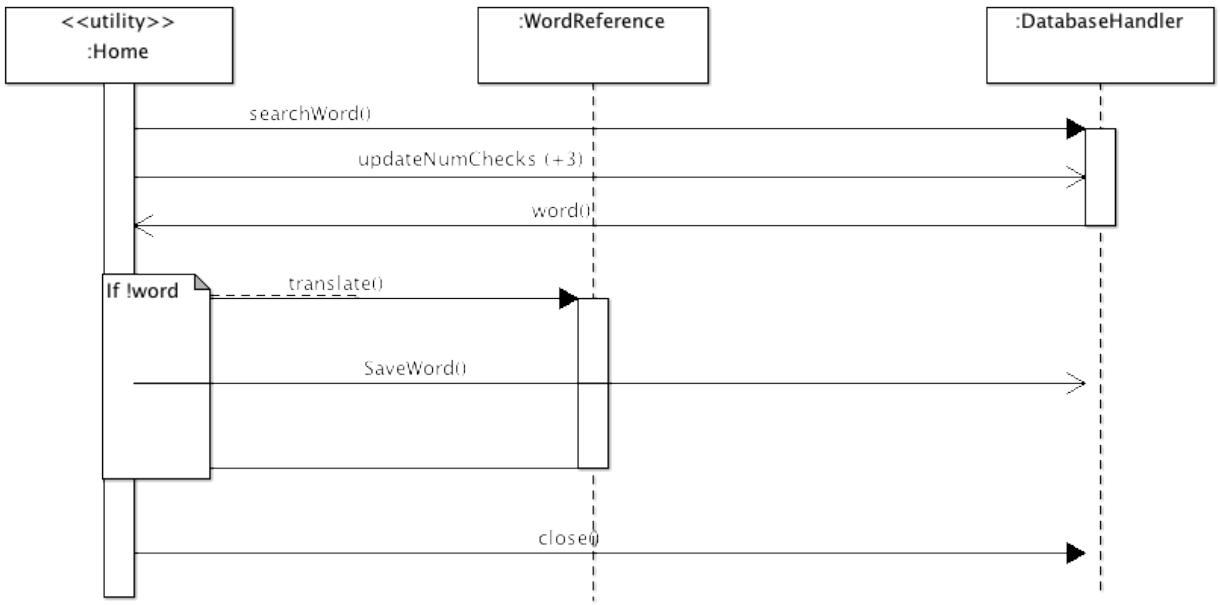


Ilustración 22 Diagrama de secuencia traducción

Lo primero que realiza esta actividad, es una comprobación de que la palabra no haya sido nunca buscada en la base de datos(*searchWord()*). Si ha sido buscada anteriormente, muestra la respuesta y a continuación suma 3 en el campo *numChecks* de la base de datos (*updateNumChecks()*). En caso contrario, a través de la clase WordReference (*translate()*), se hace una petición JSON a la API de WordReference para obtener la traducción y a continuación se guarda en la base de datos con un valor inicial de *numChecks*=3.

2.4.2 Juego 1: GameTest

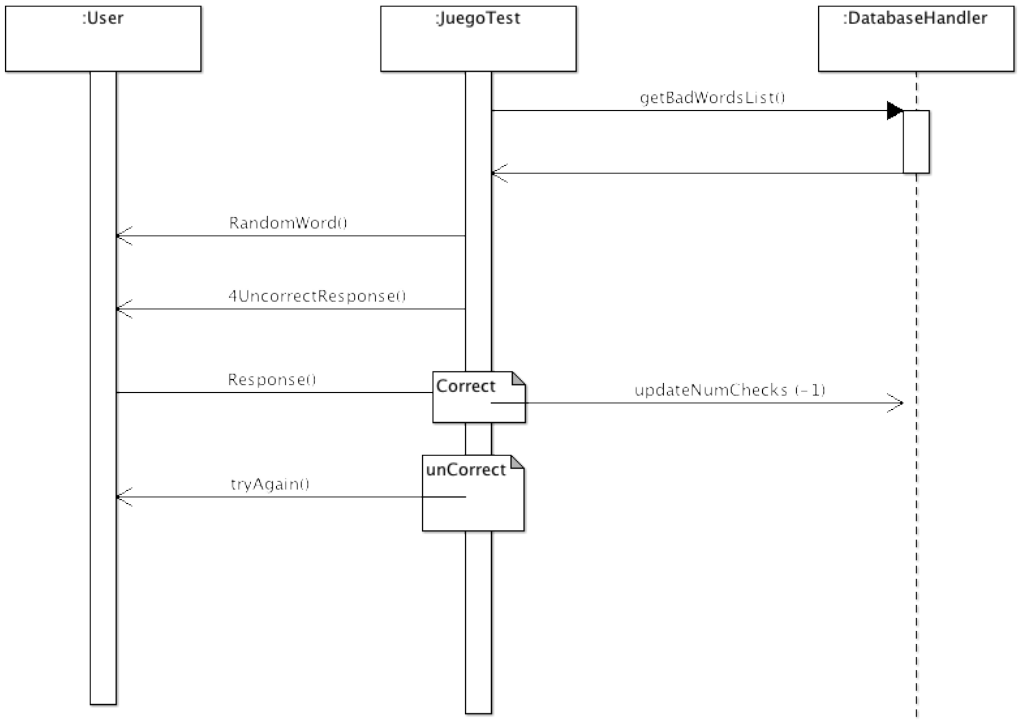


Ilustración 23 Diagrama de secuencia del juego GameTest

Esta actividad de la aplicación, se trata de un primer juego para utilizar todas las funcionalidades que pueden ser necesarias, iniciar ejercicio, comprobar respuesta, ampliar o reducir el atributo ***numChecks*** de una palabra. La actividad se inicia obteniendo la lista de palabras por aprender de la base de datos, y se escoge una aleatoria de entre estas que aparecerá como pregunta en el enunciado del juego, y también se escogen 4 palabras que sean diferentes a la primera que serán mostradas como posibles respuestas.

Cuando el usuario escoge una respuesta, si esta es incorrecta, se suma uno en el atributo *numCheck* de la palabra, en caso contrario se le resta uno y a continuación se reinicia el ejercicio. Todo este proceso hasta que el número de palabras a aprender sean menor que 5.

2.4.3 "BadWordsList" Lista de palabras por aprender

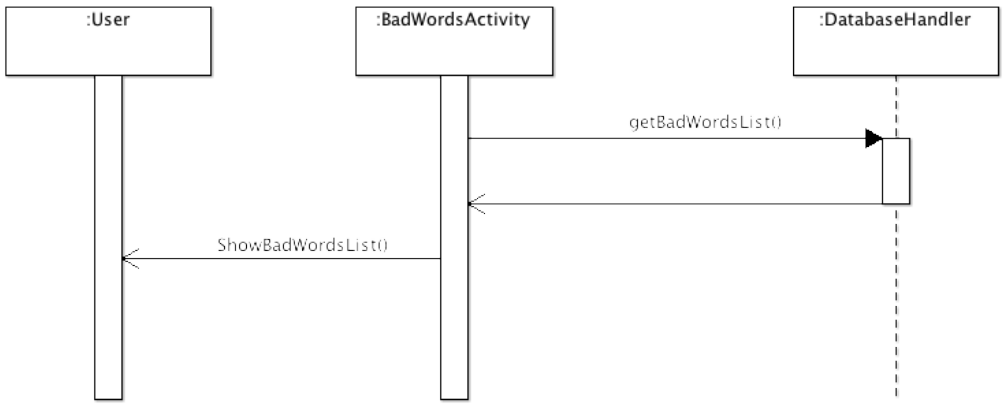


Ilustración 24 Diagrama de secuencia de lista de palabras a aprender

Esta actividad pretende mostrar una lista de las palabras a aprender por el usuario, por lo que al entrar en la actividad se hace una petición a la clase "*DatabaseHandler*" que retorna dicha lista.

2.4.4 Diccionario o lista de palabras buscadas

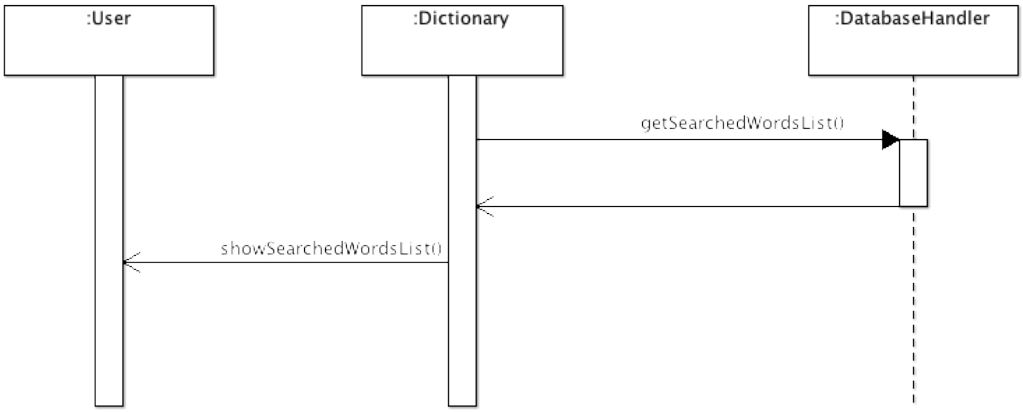


Ilustración 25 Diagrama de secuencia de lista de palabras buscadas

Esta actividad pretende mostrar una lista de las palabras buscadas por el usuario, por lo que al entrar en la actividad se hace una petición a la clase "*DatabaseHandler*" que retorna dicha lista.

3 Desarrollo de la aplicación

El desarrollo de este proyecto se realizará únicamente en el sistema operativo Android, por lo que previamente a explicar el desarrollo de cada pantalla o *activity* del proyecto, es conveniente hablar sobre algunos elementos del *framework* que son imprescindibles en cualquier desarrollo sobre éste y de algunos específicos de este proyecto que su comprensión previa ayudan a comprender los desarrollos particulares de las diferentes *activities*.

3.1 AndroidManifest.xml

Este archivo es el más importante de todo proyecto en Android. En este archivo se definen tanto las *activities* que van a aparecer en el proyecto como los elementos del móvil que se va a necesitar utilizar (permisos), como *bluetooth*, sensores, acceso a Internet…

Además, se definen las versiones de Android que se va a tener en cuenta para el desarrollo, poniendo la versión mínima necesaria para la utilización de la app, que estará definida por el SDK que se utilice y será la que dé límite en cuanto a API para el desarrollo. La máxima se puede limitar manualmente o escoger la opción por defecto, que son todas las superiores a la anterior.

En este proyecto se define:

Versión SDK	8	`<uses-sdk android:minSdkVersion="8" />`
Permisos necesarios	Internet	`<uses-permission android:name="android.permission.INTERNET"></uses-permission>`
Icono de la aplicación		`android:icon="@drawable/ic_launcher"`
Activity	AppDictusActivity	`<activity android:name=".APPdictusActivity" android:label="@string/app_name" >`

		<intent-filter> <action android:name="android.intent.action.MAIN" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity>
Activity	TestGame	<activity android:name=".TestGame" android:label="@string/app_name" android:screenOrientation="portrait"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity>
Activity	BadWords	<activity android:name=".BadWords" android:label="@string/app_name" android:screenOrientation="portrait"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity>
Activity	Searched Words	<activity android:name=".SearchedWords" android:label="@string/app_name" android:screenOrientation="portrait"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.LAUNCHER"

		/> </intent-filter> </activity>
Activity	SettingsActivity	<activity android:name=".SettingsActivity" android:label="@string/app_name" android:screenOrientation="portrait"> <intent-filter> <action android:name="android.intent.action.VIEW" /> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity>

3.2 Gestión de datos

Android proporciona dos herramientas principales para la gestión de datos en una aplicación móvil:

3.2.1 Content Providers

Un *Content Provider* es el mecanismo proporcionado por la plataforma Android para permitir compartir información entre aplicaciones.

Una aplicación que desee que todo o parte de la información que almacena esté disponible de una forma controlada para el resto de aplicaciones del sistema deberá proporcionar un *content provider* a través del cuál se pueda realizar el acceso a dicha información.

Este mecanismo es utilizado por muchas de las aplicaciones estándar de un dispositivo Android, como por ejemplo la lista de contactos, la aplicación de SMS, o el calendario/agenda. Esto quiere decir que permite el acceso a los datos gestionados por

estas aplicaciones desde otras aplicaciones Android haciendo uso de los *content providers* correspondientes.

3.2.2 SQLite - DatabaseHandler.java

SQLite es un motor de bases de datos muy popular en la actualidad por ofrecer características tan interesantes como su pequeño tamaño. Además, no necesita servidor, precisa de poca configuración, es transaccional y de código libre.

Android incorpora de serie todas las herramientas necesarias para la creación y gestión de bases de datos SQLite, y entre ellas una completa API para llevar a cabo de manera sencilla todas las tareas necesarias.

Crear, actualizar, y conectar con una base de datos *SQLite* se hace a través de una clase auxiliar llamada *SQLiteOpenHelper*, o para ser más exactos, de una clase propia que deriva de ella y que se ha personalizado para que se adapte lo máximo posible a las necesidades concretas de nuestro proyecto.

De estas dos herramientas (*content provider* o *SQLite*), se escoge la de trabajar con *SQLite* para el desarrollo del proyecto, dado que es la herramienta de gestión de datos de Android más adecuada para las necesidades del mismo y porque en este no se considera necesaria la cualidad de poder comunicar estos datos con otras aplicaciones del *smartphone*.

addWord:
Este método añade una palabra nueva a la base de datos de la aplicación con los valores de la palabra original buscada, el idioma original en el que se realiza la búsqueda, la palabra traducida, el idioma de la palabra traducida y el atributo *numChecks* que por defecto está definido a 3.

```
public void addWord(Word word) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
```

```
        values.put(KEY_WORD, word.get_word()); // word
        values.put(KEY_TRANSLATION, word.get_translation()); // word traslation
        values.put(KEY_OR_LANG, word.get_languaje_orig()); // word traslation
        values.put(KEY_TR_LANG, word.get_translation_languaje()); // word
traslation
        values.put(KEY_NO_CHECKS, word.get_num_checks()); // word traslation

        // Inserting Row
        db.insert(TABLE_WORDS, null, values);
        db.close(); // Closing database connection
}
```

getWord(int i):

Este método retorna una palabra concreta a partir del identificador de la misma en la base de datos.

```
public Word getWord(int id) {
       Word w = new Word();
       SQLiteDatabase db = this.getReadableDatabase();
              Cursor cursor = db.query(TABLE_WORDS, new String[] { KEY_ID,
                           KEY_TRANSLATION, KEY_NO_CHECKS }, KEY_WORD + "=?",
                           new String[] { String.valueOf(id) }, null, null, null,
null);
                  if (cursor != null && cursor.moveToFirst()){

w.set_id(Integer.parseInt(cursor.getString(cursor.getColumnIndex(KEY_ID))));

w.set_word(cursor.getString(cursor.getColumnIndex(KEY_WORD)));

w.set_orig_languaje(cursor.getString(cursor.getColumnIndex(KEY_OR_LANG)));

w.set_translation_languaje(cursor.getString(cursor.getColumnIndex(KEY_TR_LANG)));

w.set_translation(cursor.getString(cursor.getColumnIndex(KEY_TRANSLATION)));
                     w.set_num_checks(Integer.parseInt(cursor.getString(
```

```
cursor.getColumnIndex(KEY_NO_CHECKS))));
                    return w;
                }
        return null;
}
```

getAllWords:

Este método realiza una consulta a la base de datos para retornar una lista de todas las palabras que se han buscado alguna vez ordenados alfabéticamente. Esto lo retorna en un *ArrayList* de objetos de la clase *Word*.

```
public List<Word> getAllWords() {
        List<Word> wordList = new ArrayList<Word>();
        String selectQuery = "SELECT  * FROM " + TABLE_WORDS + " ORDER BY " +
KEY_WORD;
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);

        if (cursor.moveToFirst()) {
            do {
                Word word = new Word();
                word.set_id(Integer.parseInt(cursor.getString(0)));
                word.set_word(cursor.getString(1));
                word.set_translation(cursor.getString(2));
                word.set_orig_languaje(cursor.getString(3));
                word.set_translation_languaje(cursor.getString(4));
                word.set_num_checks(Integer.parseInt(cursor.getString(5)));

                wordList.add(word);
            } while (cursor.moveToNext());
        }
        return wordList;
}
```

getBadWords:

Este método realiza una búsqueda en la base de datos de aquellas palabras que son consideradas como no aprendidas, es decir, aquellas palabras que tengan en el atributo *numChecks* un valor superior a cero.

```
public List<Word> getBadWords() {
        List<Word> wordList = new ArrayList<Word>();
        String selectQuery = "SELECT  * FROM " + TABLE_WORDS + " WHERE " +
KEY_NO_CHECKS + " > 0" ;
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery(selectQuery, null);
        if (cursor.moveToFirst()) {
            do {
                Word word = new Word();
                word.set_id(Integer.parseInt(cursor.getString(0)));
                word.set_word(cursor.getString(1));
                word.set_translation(cursor.getString(2));
                word.set_orig_languaje(cursor.getString(3));
                word.set_translation_languaje(cursor.getString(4));
                word.set_num_checks(Integer.parseInt(cursor.getString(5)));
                wordList.add(word);
            } while (cursor.moveToNext());
        }
        return wordList;
}
```

updateWord:

Este método sirve para actualizar los valores que puedan variar en una "palabra". Teniendo en cuenta que la traducción de una palabra de un idioma a otro y que la traducción de la misma palabra entre pares de idiomas distintos se consideran elementos diferentes, el único campo que se ha considerado que se va a actualizar, es el de *numChecks*, es por ello que es el único valor que se puede modificar en la base de datos.

```
public int updateWord(Word word) {
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues values = new ContentValues();
```

```
        values.put(KEY_NO_CHECKS, word.get_num_checks());
        return db.update(TABLE_WORDS, values, KEY_ID + " = ?", new String[] {
String.valueOf(word.get_id()) });
}
```

deleteWord(Word w):

Este método permite eliminar una palabra de la base de datos a partir del objeto de la misma para obtener el id.

```
public void deleteWord(Word word) {
        SQLiteDatabase db = this.getWritableDatabase();
        db.delete(TABLE_WORDS, KEY_ID + " = ?", new String[] {
String.valueOf(word.get_id()) });
        db.close();
}
```

find(Word w):

Este método retorna un valor true o false en función de si la palabra se encuentra o no en la base de datos a partir de la palabra original y el idioma. Además si la palabra es encontrada actualiza sus campos con los obtenidos desde la base de datos.

```
public boolean find(Word w) {
       boolean exist = false;
       SQLiteDatabase db = this.getReadableDatabase();
       Cursor cursor = db.query(TABLE_WORDS, new String[] { KEY_ID,
                   KEY_TRANSLATION, KEY_NO_CHECKS }, KEY_WORD + "=?",
                   new String[] { w.get_word() }, null, null, null, null);
       if (cursor != null && cursor.moveToFirst()){

w.set_id(Integer.parseInt(cursor.getString(cursor.getColumnIndex(KEY_ID))));

w.set_translation(cursor.getString(cursor.getColumnIndex(KEY_TRANSLATION)));
```

```
w.set_num_checks(Integer.parseInt(cursor.getString(cursor.getColumnIndex(KEY_NO_
CHECKS))));
            exist = true;
      }
        return exist;
}
```

3.3 Clase WordReference.java

Como se decía anteriormente, de entre las diferentes opciones que existen para generar un diccionario de traducción entre varios idiomas (API de Google Translate, WordReference, Apertium, Bing Translator,...) se ha escogido la API de WordReference sobre todo por el cambio en el formato de utilización de la API de Google Translate, que a partir de ahora tiene un coste por carácter traducido y en comparación a las otras, es mucho más escalable y ofrece información añadida que podría ser interesante para nuevos juegos.

A partir de aquí, se definen los atributos y métodos básicos que son necesarios para sacarle el máximo rendimiento a esta API, y teniendo en cuenta que se hace siempre a partir de peticiones *REST* a una URL con el siguiente formato: **"http://api.wordreference.com/[version]/[API ID]/json/[par de lenguajes]/[Palabra]"** y que retorna objetos JSON, es necesario utilizar la libreria de JAVA ***JSONObject*** específica para este tipo de casos.

3.3.1 Atributos de la clase

APPIKEY:
La *AppiKey* es la clave necesaria para poder utilizar la API en nuestra aplicación. Esta clave además tendrá que estar presente en todas las peticiones que se hagan, es por ello y para que sea más sencillo de modificar, que se crea un atributo estático con la misma.

APILevel:

Otro de los campos que pueden variar de la API ofrecida por WordReference es la versión de la misma, ya que de una a otra pueden haber cambios significativos. Es imprescindible establecer una constante con este valor y se pueda modificar en caso necesario.

APIURL:

La *APIURL* es la URL a la que se hace la petición, en este caso "http://api.wordreference.com/". Este valor no debería cambiar, por lo que se podría utilizar en texto dentro de los métodos, pero se considera una buena práctica definir como atributo este texto por si diese la circunstancia que la URL de la API de WordReference cambiase.

APIFormat:

La API de WordReference ofrece diversos formatos de retorno de la petición: *JSON*, XML, … en un principio se establecen que las peticiones que se van a realizar siempre serán en *JSON* al ser un lenguaje de más sencillo tratamiento. De todos modos y de cara a una posible ampliación de esta clase, se define este atributo para que sea modificable.

sourceWord:

Este atributo es la palabra escrita por el usuario para obtener su traducción.

sourceLanguaje:

Dentro del par de idiomas escogido por el usuario, este atributo haría referencia al idioma original en que el usuario realiza la búsqueda.

destinationLanguaje:

Igual que el anterior, pero en este caso se trata del idioma de destino de la traducción. En ambos casos, se trata del código ISO del lenguaje, es decir ES para Español, EN para Inglés, CA para Catalán, etc...

translation:

En este atributo se puede almacenar el objeto *JSON* una vez convertido por la clase ***JSONObject*** para poder trabajar con los diferentes resultados que ofrece la API de WordReference para cada palabra.

translatedWord:

En este atributo se almacenará la palabra una vez traducida, para poder trabajar con ella a través del método *getTranslatedWord*.

numResults:

En varios puntos de este proyecto, al hablar de la API de WordReference, se ha hecho referencia a que esta ofrece normalmente más de un resultado para una misma petición, es por ello que es interesante tener en algún sitio el número de resultados obtenidos.

translationError:

Como en todas las APIs, siempre hay que tener en cuenta la posibilidad de que haya ocurrido algún error en la petición, por ejemplo que la palabra no exista o no se encuentre, que haya algún error interno en la API de WordReference y no se pueda responder a la petición, que haya una carga muy elevada de peticiones y no se pueda hacer frente,... es por ello que se ha definido un atributo para almacenar el error en caso necesario.

3.3.2 Métodos de la clase WordReference

Constructor WordReference:

Como todo constructor en Java, genera un objeto de la clase, en este caso un objeto WordReference, en el que como mínimo ha de tener un idioma origen y un idioma destino.

```
public WordReference(String sourceLanguaje, String destinationLanguaje) {
	super();
	this.sourceLanguaje = sourceLanguaje;
	this.destinationLanguaje = destinationLanguaje;
}
```

prepareURL:

Lo primero necesario para hacer una petición REST, es preparar la URL a la que se va a hacer dicha petición.

En Java existe un tipo de objeto (URL) que ya está preparado para realizar esta tarea, es por ello, que se prepara un método que a partir de los atributos del punto anterior genera y retorna un objeto URL a la que se va a realizar la petición.

```
private URL prepareURL() throws Exception{
	URL ret;
	ret = new
URL(this.APIURL+"/"+this.APILevel+"/"+this.APIKEY+"/"+this.APIFormat+"/"

	+this.sourceLanguaje+this.destinationLanguaje+"/"+this.sourceWord);
	return ret;
}
```

JSONfromURLRequest:

Este método, es quizás el más importante de toda la clase, dado que realiza una petición Http a la URL que comentada anteriormente y retorna un objeto JSONObject con la respuesta de dicha petición.

Hay que tener en cuenta en este método, que el retorno del servidor al que se hace una petición de este estilo, puede tener diversas codificaciones, en el caso de WordReference, se trata de **UTF-8**[2] y por tanto es esta la codificación que se va a tratar.

[2] **UTF-8** es uno de los formatos de codificación de caracteres más comunes para obtener la información entre cliente y servidor. La codificación de caracteres, define la forma en la que se codifica un carácter dado en un símbolo en otro sistema de representación. En el caso de Internet, esta codificación se representa mediante bits. Por lo que si el codificado entre servidor y cliente no es el mismo, se generan errores de interpretación.

```
public static JSONObject JSONfromURLRequest(String url){
		InputStream is = null;
		String result = "";
		JSONObject jArray = null;

		try{
			HttpClient httpclient = new DefaultHttpClient();
			HttpPost httppost = new HttpPost(url);
			HttpResponse response = httpclient.execute(httppost);
			HttpEntity entity = response.getEntity();
			is = entity.getContent();

		}catch(Exception e){
			Log.e("log_tag", "Error in http connection "+e.toString());
		}
		try{
			BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"UTF-8"),8);
			StringBuilder sb = new StringBuilder();
			String line = null;
			while ((line = reader.readLine()) != null) {
				sb.append(line + "\n");
			}
			is.close();
			result=sb.toString();
		}catch(Exception e){
			Log.e("log_tag", "Error converting result "+e.toString());
		}
		try{
			jArray = new JSONObject(result);
		}catch(JSONException e){
			Log.e("log_tag", "Error parsing data "+e.toString());
		}

		return jArray;
	}
```

Translate:

Por último, y más interesante, es generar el método que se va a utilizar desde fuera del objeto con el fin de obtener una traducción. El método *translate*, realiza una petición con al método anterior y formatear la respuesta obtenida y incluye los resultados en un objeto "*Word*" que le viene dado como parámetro.

Además como se ha comentado en la API de WordReference, la información relativa a una palabra puede ser de tres maneras, la principal (*PrincipalTranslations*) es la traducción directa de una palabra, por ejemplo *Dog-Perro.*

La segunda opción, serían las entradas *Entries*, en las que ofrece frases en las que se puede encontrar la palabra y el significado de la misma en esta frase.

Y por último, se pueden encontrar los compuestos en los que si se combina dicha palabra junto a otra u otras tuviese diferente significado.

Por ejemplo la palabra PERK, no tiene una traducción principal en WordReference pero como entrada ofrece "beneficio" y en la combinación PERK UP, se obtiene "*phrasal verb*" de significado *animarse- espabilarse.*

Este múltiple formato de retorno en la respuesta de la API de WordReference obliga a , dentro de la clase que generada, controlar si alguno de estas maneras de obtener la información no se ha obtenido. Si en una traducción no se ha obtenido traducción primaria, como en el ejemplo anterior, es preciso controlar si tiene entrada o significado en algún compuesto.

```
public void translate(String word){
	try{
		this.sourceWord = word;
		URL url = this.prepareURL();

		JSONObject translation = new JSONObject();
		String urlStr = url.toString();
		urlStr = urlStr+"";
```

```
            Log.i("WordReference.translate", url.toString());
            translation = WordReference.JSONfromURLRequest(url.toString());
            if(!translation.isNull("term0")){
                  this.numResults = translation.length();

      if(!translation.getJSONObject("term0").isNull("PrincipalTranslations")){
                        this.translatedWord =
translation.getJSONObject("term0").

      getJSONObject("PrincipalTranslations").getJSONObject("0").

      getJSONObject("FirstTranslation").getString("term");

      Log.i("WordReference.translate","PrincipalTranslations:"+
                              this.translatedWord);
                  }else
if(!translation.getJSONObject("term0").isNull("Entries")){//Entries
                        this.translatedWord =

      translation.getJSONObject("term0").getJSONObject("Entries").

      getJSONObject("0").getJSONObject("FirstTranslation").
                                    getString("term");
                        Log.i("WordReference.translate", "Entries:" +
this.translatedWord);                        }else{
                        this.translatedWord =

      translation.getJSONObject("term0").getJSONObject("Compounds").

      getJSONObject("0").getJSONObject("FirstTranslation").
                              getString("term");
                        Log.i("WordReference.translate", "Compounds:" +
this.translatedWord);
                  }
            }
            else if(translation.has("Error")){
                  this.translationError = translation.getString("Note");
            }
```

```
            else{
                  this.translationError = "This word has not translation or
does not exist";
            }
      }catch(Exception e){
            Log.e("log_tag", "Error parsing data "+e.toString());
      }
}
```

3.4 Pantallas de la aplicación

Android, para el desarrollo de sus aplicaciones, ofrece la posibilidad de separar el diseño de la programación. Siendo el primero definido dentro de un archivo XML a través de los componentes que ofrece Android para maquetar. Esta maquetación se hace de un modo similar a HTML pero con elementos propios de Android.

Por ejemplo, si se pretende añadir un botón sería con el siguiente código:

```
<Button
            android:id="@+id/search_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/search" />
```

Dónde:

- android:id define el identificador del botón para poder acceder a él desde el código.

- android:layout_width define el ancho del elemento, en este caso con "wrap_content" se definine que el elemento tiene que tener el ancho de su contenido.

- android_layout_height igual que el atributo anterior pero con el alto.

- android:text define el texto que va a contener el elemento, además en este caso, se puede observar que (@string/search) hace referencia a un valor definido en otro sitio. Esto es porque Android permite tener un archivo donde se definan todos los strings de la aplicación, de modo que si se quiere internacionalizar la misma sea mucho más sencillo.

A parte de estos atributos, se pueden encontrar muchos más para este elemento "*Button*" y otros específicos para los otros elementos que se puedan precisar que no se va a incluir en este proyecto dado que no pretende ser una guia para aprender a maquetar en Android.

Además del archivo .xml que genera la parte visual de cada *activity*, es preciso generar una clase extendida de la clase *Activity*, en la que se desarrolla toda la parte lógica de la aplicación. Esta clase al ser herencia de Activity, tiene que tener algunos métodos "sobreescritos" en los que se desarrolle dicha lógica.

Estos métodos son interesantes tenerlos en cuenta en su concepto y sobretodo dentro del ciclo de vida de la Activity:

3.4.1 Inicio / Buscador

Ilustración 26 Pantalla inicial de la aplicación

Esta es la pantalla o *Activity* inicial de la aplicación, y también la que hará de buscador. En ella el usuario se puede encontrar el área de texto donde introducir la palabra a traducir, el idioma de origen y el idioma de destino. Y, por último, el botón que realiza la petición para obtener la traducción de la palabra a traducir.

A partir de todo lo anterior, dentro del método sobreescrito *onCreate*, se inician al principio todos los elementos que están definidos en el XML para poder interactuar con ellos desde Java.

A continuación se crea un *listener* para este botón que al ser clickado realiza una petición a la clase *DababaseHelper*.

En caso de no encontrar la palabra buscada en la base de datos se realiza una petición a WordReference para obtener dicha palabra así y a continuación la guarda.

```
this.search_button.setOnClickListener(new View.OnClickListener() {
                @Override
       public void onClick(View v) {
                    if(!search_word.getText().toString().equals("")){
                     String orig_word = search_word.getText().toString();
                     Word w = new Word(orig_word,
AppDictus.getInstance().getOriginalLanguajeCode(), "",
AppDictus.getInstance().getTraslationLanguajeCode());
                     if(!db.find(w)){
                           WordReference wr = new
WordReference(AppDictus.getInstance().getOriginalLanguajeCode(),
AppDictus.getInstance().getTraslationLanguajeCode());
                           wr.translate(orig_word);
                           if(!wr.getError().isEmpty()){
       Toast.makeText(APPdictusActivity.this.getApplicationContext(),
wr.getError(), 10000).show();
                           }
                           else{
                                 String translation = wr.getTranslatedWord();
                                 w.set_translation(translation);
                                 db.addWord(w);
                           }
                     }else{
                           w.set_num_checks(w.get_num_checks()+3);
                           db.updateWord(w);
                     }
                     search_results.setText(w.get_translation());
              }else{
                     search_results.setText("you must write any word");
              }
             }
        });
    }
```

3.4.2 Juego 1: GameTest

Ilustración 27 Pantalla GameTest

En esta aplicación, se ha definido un único juego inicial que utilice todos los elementos básicos que pueda necesitar cualquier otro juego: solicitar una palabra a aprender de manera aleatoria, una lista de todas ellas, comprobar si la respuesta del usuario es correcta, marcar una palabra como aprendida cuando se considere necesario,...

En este caso, se puede ver que la maquetación creada es un poco más compleja que la de la *activity* anterior, ya que en este caso aparece la palabra de pregunta, y con 5 campos de *RadioButton* que corresponden a las 5 posibles respuestas que se van a ofrecer.

```
<TextView
        android:id="@+id/question1"
        android:background="#ffffff"
        android:layout_height="70dip"
        android:layout_width="fill_parent"
        android:textSize="15dip"
        android:textColor="#000000" />
```

```
<RadioGroup
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
      <RadioButton android:id="@+id/option1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:textColor="#000"
          android:onClick="onRadioButtonClicked"/>
```

```
    <RadioButton android:id="@+id/option2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/option3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/option4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/option5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#000"
        android:onClick="onRadioButtonClicked"/>
  </RadioGroup>
```

A continuación se muestran los métodos que dan vida a esta maquetación:

initGame():

Primero de todo se inicia el juego con un método *initGame()* que obtiene de la base de datos todas las palabras a aprender y comprueba que la longitud de está lista sea mayor que 5 y lanza el método *showTestQuestion()*, en caso contrario muestra un mensaje de error.

```
private void initGame(){
            this.wordList = db.getBadWords();
            if(this.wordList.size() > 5){
                  this.showTestQuestion();
            }else{
                  this.question.setText("You don't have actually enough words
```

```
searched, please find translations to play.");
                    this.option1.setVisibility(View.INVISIBLE);
                    this.option2.setVisibility(View.INVISIBLE);
                    this.option3.setVisibility(View.INVISIBLE);
                    this.option4.setVisibility(View.INVISIBLE);
                    this.option5.setVisibility(View.INVISIBLE);
            }
      }
```

showTestQuestion():

Este método escoge una de las palabras a aprender al azar de las obtenidas en el anterior, que será la considerada como palabra de partida.

A continuación, en el mismo método se escogen también de manera aleatoria otras 5 palabras aleatorias de la misma lista y se ponen en otra lista diferente denominada **lista de opciones de partida**, además con el fin de que no haya repeticiones, cada palabra escogida es extraída de la lista original.

Luego se sustituye la palabra de partida por una de las que aparecen en la lista de opciones de partida, también de manera aleatoria.

Por último, en este método se cogen las 5 palabras de las lista de opciones de partida y se ponen en los 5 elementos *RadioButton*.

```
private void showTestQuestion(){
        for(int i=0; i<wordList.size(); i++){
             wordsNames.add(wordList.get(i).get_word());
        }

      this.options = new ArrayList<Word>();
      this.qWord = wordList.remove((int)(Math.random()*this.wordList.size()));
      this.question.setText(qWord.get_word());
```

```
        //option 1
        int val = (int)(Math.random()*this.wordList.size());
        this.options.add(this.wordList.remove(val));
        //option 2
        val = (int)(Math.random()*this.wordList.size());
        this.options.add(this.wordList.remove(val));
        //option 3
        val = (int)(Math.random()*this.wordList.size());
        this.options.add(this.wordList.remove(val));
        //option 4
        val = (int)(Math.random()*this.wordList.size());
        this.options.add(this.wordList.remove(val));
        //option 5
        val = (int)(Math.random()*this.wordList.size());
        this.options.add(this.wordList.remove(val));

        this.options.add((int)(Math.random()*this.options.size()), this.qWord);

        this.option1.setText(this.options.get(0).get_translation());
        this.option2.setText(this.options.get(1).get_translation());
        this.option3.setText(this.options.get(2).get_translation());
        this.option4.setText(this.options.get(3).get_translation());
        this.option5.setText(this.options.get(4).get_translation());
}
```

onRadioButtonClicked():

Una vez han aparecido la pregunta y las 5 posibles opciones, se tiene que evaluar la respuesta del usuario, esto ocurre cuando selecciona alguno de los 5 campos *RadioButton* de los comentados.

Si el contenido de este elemento es igual a la traducción de la palabra de partida, se establece como correcto, se muestra por pantalla el resultado y se carga una nueva partida.

En caso contrario, se muestra un texto indicando al usuario que se ha equivocado.

```
public void onRadioButtonClicked(View v) {
           RadioButton rb = (RadioButton) v;
           String response = "";
           if(rb.getText().equals(this.qWord.get_translation())){
              response = "Correct";
              this.qWord.set_num_checks(this.qWord.get_num_checks()-1);
              this.db.updateWord(this.qWord);
              if(qWord.get_num_checks() ==
0)Toast.makeText(getApplicationContext(), qWord.get_word() + " aprendida!",
5000).show();
              rb.setChecked(false);
                  this.initGame();
           }
           else response = "Not correct";
           Toast.makeText(getApplicationContext(), response, 5000).show();
      }
```

3.4.3 Historial y lista palabras a aprender

Ilustración 28 Pantallas: Historial y lista de palabras a aprender

El historial de palabras buscadas o diccionario, junto con la pantalla de palabras a aprender, son quizás los elementos más sencillos del proyecto, teniendo en cuenta que actualmente la interacción que tienen es mínima y que únicamente tienen que mostrar una lista u otra de palabras.

Además, toda la inteligencia de estas dos pantallas, la tiene la clase ya explicada anteriormente *DataBaseHandler* en sus métodos *getAllWords()* y *getBadWords()*. Es por ello que se ha decidido aunar la explicación de estas dos Activities en este punto.

Lo primero que se va a tratar, es el hecho de mostrar una

lista de elementos en Android a la que se ha añadido la posibilidad de interactuar con el usuario mínimamente, es decir, cuando se selecciona una palabra ofrece más información sobre esta en formato de popup o diálogo.

Para mostrar las palabras buscada por el usuario en una lista, y poder interactuar con ella es necesario hacer como mínimo tres layouts o archivos XML donde se maqueta el diseño:

searched_words_list.xml:
Este es un archivo en el que se define una lista mediante el elemento *ListView* de Android en el que aparecerán todas las palabras en una lista.

Además se define un elemento *TextView* para mostrar un texto por defecto en el caso de que la lista esté vacía.

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffff">
     <ListView
         android:id="@android:id/list"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:cacheColorHint="#00000000"
         />
     <TextView android:id="@android:id/empty"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="You do not have actually any words"
         android:textColor="#000000"
         android:textSize="15dp"
         android:textStyle="bold"
         />
 </LinearLayout>
```

Como se puede observar en el código anterior, a parte de los atributos generales de un *layout* en Android (*layout_width, layout_height*,...) en el elemento *ListView* identificado como ***list,*** se encuentra el atributo *cacheColorHint*, este atributo hace referencia al color de fondo del layout cuando este es activado, por ejemplo cuando el usuario selecciona una palabra o cuando se hace *scroll*. Para evitar que dé ningún problema, se establece un color transparente en RGB. Es por ello que es de 8 dígitos en lugar de 6 como sería lo normal.

list_words.xml:

A continuación se define el *layout* para cada una de las palabras que aparecen en la lista, ya que de este modo es posible personalizar cada uno de los elementos por separado, color de fondo, tipo de letra... como el diseño de esta aplicación está lejos de los oscuros que utiliza Android por defecto, se ha de intentar en todo momento obtener los máximos grados de personalización posible.

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="10dp"
    android:textSize="16sp"
    android:textColor="#000000">
</TextView>
```

word_info_dialog.xml:

Cuando el usuario quiere obtener más información sobre una palabra que aparezca en cualquiera de estas dos listas, se podría haber realizado de dos maneras. Mostrado esta información en un nuevo *layout* incluso con una nueva *activity* o en un dialogo a modo de *popup*.

La principal ventaja del segundo formato, es la interacción con el usuario es mucho más ágil y rápida que en el primero, y teniendo en cuenta que la información a buscar no es excesiva como para ocupar toda la pantalla, se ha considerado que este es el mejor formato para hacerlo.

Para hacerlo, se ha definido el *layout* ***word_info_dialog.xml*** con 3 elementos, un *TextView* donde aparecerá la traducción, un segundo *TextView* donde aparecerá un comentario en función del número de *numChecks* que tenga esa palabra, si es 0 indicará al usuario que ya se la ha aprendido (solo en el caso del historial) y si es mayor a 0, muestra la frase *"You need to hit **X** times this word to regard it as learned*". Además, se hace uso del título de diálogo para poner la palabra de origen, de modo que sea más usable y estético.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/layout_root"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="10dp"
android:background="#cfcfcf"
>
<TextView android:id="@+id/translated_word"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:textColor="#222"
              android:textStyle="bold"
              android:textSize="16dp"
                    android:layout_alignParentLeft="true" />
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:paddingTop="20dp">
        <TextView android:id="@+id/numChecks"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="0.62"
            android:textColor="#222"
            android:ems="10">
        </TextView>

        <Button android:id="@+id/delete_word"
            android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:text="@string/delete"
            android:textColor="#AA1010" />
    </LinearLayout>
</RelativeLayout>
```

Además, como se puede observar en el código anterior, dentro del dialogo que muestra la información de la palabra, se encuentra un botón *delete* que permite al usuario eliminar una palabra de la base de datos.

En cuanto al código de estas dos *Activities*, como se ha comentado, las únicas diferencias son el hecho de que el historial solicita a *DataBaseHandler* una lista de todas las palabras buscadas y la *Activity* de palabras a aprender solo una lista de palabras que tengan un valor en *numCheck* mayor a 0.

El hecho de que en el caso del historial se muestra un texto diferente si la palabra tiene un valor en *numCheck* mayor a 0 en el diálogo:

Historial:

```
private List<Word> wordList= null;
...
...
final DatabaseHandler db = new DatabaseHandler(this);
this.wordList= db.getAllWords();
...
...
if(w.get_num_checks()>0)numChecks.setText("You need to hit " +
w.get_num_checks() + " times this word to regard it as learned");
else numChecks.setText("Congratulations! you have already learned this word");
```

Lista de palabras a aprender:

```
private List<Word> wordList= null;
...
final DatabaseHandler db = new DatabaseHandler(this);
this.wordList= db.getBadWords();
...
...
...
numChecks.setText("You need to hit " + w.get_num_checks() + " times this word
to regard it as learned");
```

Una vez obtenidas las diferentes listas de palabras, estas se han de mostrar en el layout *searched_words_list* que ya se ha explicado. Esto se hace recorriendo todo el array de objetos que retorna la clase *DataBaseHandler*, y guardando sólo el *String* de la palabra original buscada en un nuevo *ArrayList*, y a continuación se asigna este al *layout list_words* que también ha sido comentado.

```
List<String>wordsNames = new ArrayList<String>();//List<Word> wordList = new
ArrayList<Word>();
        for(int i=0; i<this.wordList.size(); i++){
             //wordsNames.add(wordList.get(i).get_word() + "(" +
wordList.get(i).get_languaje_orig() + ") : " +
wordList.get(i).get_translation() + "(" +
wordList.get(i).get_translation_languaje() + ") ["+
wordList.get(i).get_num_checks() + "]");
             wordsNames.add(wordList.get(i).get_word());
        }
        setListAdapter(new ArrayAdapter(this, R.layout.list_words,
wordsNames));
```

Con esto, se ha conseguido crear una lista de palabras que al ser pulsadas pueden interactuar de alguna manera, por lo que faltaría crear un *listener* que responda a esta acción. Para ello Android ofrece un método que se puede sobreescribir *onListItemClick,* de este modo se puede definir que muestre la información de la palabra que el usuario ha pulsado a partir de la posición de esta en la lista. Es decir, la lista de palabras que se ha

obtenido inicialmente que se guarda en un *ArrayList* de objetos de tipo *Word* **(*wordList*)** y la lista de *Strings* que genera para mostrarla en el *layout* **(*wordNames*)** tienen el mismo orden, por lo que cuando el usuario selecciona una palabra que se encuentra en la posición 3 del *layout*, se puede obtener todo el objeto de esta palabra en la posición 3 del primer *ArrayList*.

Una vez obtenida la palabra solicitada por el usuario, es preciso enviarla al *layout word_info_dialog.xml* y mostrarlo, esto se realiza con el método *showWordInfo* dentro del *listener* que que ya se ha comentado:

```
@Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);
        this.showWordInfo(position);
    }
```

Este método es el encargado de crear un diálogo con el layout *word_info_dialog* e introduce en éste los datos relativos al objeto *Word* que ha seleccionado el usuario, la palabra buscada en el título, la palabra traducida dentro de una frase y el *numChecks* que le quedan al usuario para que su palabra quede considerada como aprendida.

```
public void showWordInfo(int p){
        Word w = this.wordList.get(p);
        final Dialog d= new Dialog(this);
        d.setContentView(R.layout.word_info_dialog);
        d.setTitle(w.get_word());

        TextView translated_word = (TextView)
d.findViewById(R.id.translated_word);
        translated_word.setText("Translation to " +
w.get_translation_languaje() + ": " + w.get_translation());
        TextView numChecks = (TextView) d.findViewById(R.id.numChecks);
        if(w.get_num_checks()>0)numChecks.setText("You need to hit " +
w.get_num_checks() + " times this word to regard it as learned");
```

```
        else numChecks.setText("Congratulations! you have already learned this
word");
        d.show();
        Button closeButton = (Button) d.findViewById(R.id.close_word_dialog);
        closeButton.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View view) {
            d.dismiss();
            }
    });}
```

3.4.4 Configuración

Ilustración 29 Pantalla de configuración

A pesar de la sencillez de la aplicación desarrollada para la realización de este proyecto, se ha incluido en ella una pantalla donde el usuario puede personalizar algunos elementos de la misma.

Actualmente, solo se ha considerado imprescindible la posibilidad de escoger entre el par de idiomas que quiera aprender. Además, a raíz del cierre de la versión 1 de la API de Google Translate, se ha tenido que restringir en pares de idiomas en lugar de escoger ambos idiomas. Pero ésta es considerada una de las funcionalidades que se tendrían que resolver a más corto plazo.

Para la generación de una pantalla de configuración, Android ya ofrece por defecto un sistema sencillo con el que hacerlo.

Este se basa en la creación de un archivo XML en el que se definen los elementos que se van a poder configurar de la aplicación y a continuación se utiliza dicho archivo como

layout para una *Activity* que en lugar de ser herencia de la clase *Activity* extiende de *PreferenceActivity*.

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
  xmlns:android="http://schemas.android.com/apk/res/android">
      <PreferenceCategory android:title="Languaje settings">
             <ListPreference
                       android:title="Pair Languajes"
                       android:key="lang_pair"
                       android:entries="@array/listArray"
                       android:entryValues="@array/listValues"
                       android:defaultValue="English - Spanish"/>
      </PreferenceCategory>
</PreferenceScreen>
```

Como se puede observar, en el código anterior que hace referencia al archivo *preferences.xml* no se incluyen todos los valores de los pares de idiomas ofrecidos al usuario. Esto se ha realizado de esta manera para que sea más escalar al tener estos pares de idiomas centralizados en un único archivo *arrays.xml*.

Además, para mostrar los idiomas correctamente y almacenarlos sin que haya ningún problema con la codificación de carácteres, se han definido en el mismo archivo dos arrays, uno con los pares de idiomas y el otro con los valores de dichos pares. Se consideran valores a los códigos ISO de cada idioma, de modo que el para Inglés-Español sería EN-ES.

```
<string-array name="listArray">
   <item>English - French</item>
   <item>English - Italian</item>
   <item>English - Spanish</item>
   <item>Français - Espagnol</item>
   <item>Español - Portugués</item>
   <item>English - Portuguese</item>
```

```
<string-array name="listValues">
   <item>EN-FR</item>
   <item>EN-IT</item>
   <item>EN-ES</item>
   <item>FR-ES</item>
   <item>ES-PT</item>
   <item>EN-PT</item>
```

```
   <item>English - German</item>
   <item>English - Russian</item>
   <item>English - Polish</item>
   <item>English - Romanian</item>
   <item>English - Czech</item>
   <item>English - Greek</item>
   <item>English - Turkish</item>
   <item>English - Chinese</item>
   <item>English - Japanese</item>
   <item>English - Korean</item>
   <item>English - Arabic</item>
</string-array>
```

```
   <item>EN-GE</item>
   <item>EN-RU</item>
   <item>EN-PO</item>
   <item>EN-RO</item>
   <item>EN-CH</item>
   <item>EN-GR</item>
   <item>EN-TR</item>
   <item>EN-CN</item>
   <item>EN-JP</item>
   <item>EN-KO</item>
   <item>EN-AR</item>
</string-array>
```

3.5 Menú y navegación de la aplicación

En todas las pantallas que se han explicado en el punto anterior, no se ha incluido uno de los elementos más importantes: la navegación entre pantallas.

Esto se puede realizar mediante un elemento que ofrece la API de Android, en el que se define el menú en un archivo XML dentro de la carpeta *menu.*

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/goHome"
          android:icon="@drawable/ic_launcher" />
    <item android:id="@+id/goGame"
          android:icon="@drawable/icon_juego"/>
    <item android:id="@+id/goWords"
          android:icon="@drawable/icon_badwords"/>
    <item android:id="@+id/goDiccionary"
          android:icon="@drawable/icon_listwords"/>
    <item android:id="@+id/goSettings"
          android:icon="@drawable/icon_settings"/>
</menu>
```

De este modo, además, se pueden definir menús variables en función de la pantalla en que se ubique cada menú. Esto se realiza con el método sobrescrito *onCreateOptionsMenu.*

```
@Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.main_menu, menu);
        return true;
    }
```

Además, como se puede observar en el archivo *main_menu.xml*, este menú está compuesto de cinco elementos, en los que cada uno está compuesto a su vez de un identificador de elemento de menú y una imagen.

La imagen es la que se muestra en la aplicación y el identificador es el que se puede utilizar para averiguar la opción escogida por el usuario y así lanzar el "*Intent*" de esta *Activity*.

```
@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle item selection
       switch (item.getItemId()) {
        case R.id.goHome:
             Intent home = new Intent(getApplicationContext(),
APPdictusActivity.class);
                   startActivity(home);
                   return true;
        case R.id.goGame:
             Intent testGame = new Intent(getApplicationContext(),
TestGame.class);
                   startActivity(testGame);
                   return true;
        case R.id.goWords://Bad words activity
             Intent badWords = new Intent(getApplicationContext(),
BadWords.class);
```

```
                    startActivity(badWords);
                    return true;
        case R.id.goDiccionary:
             Intent searchedWords = new Intent(getApplicationContext(),
SearchedWords.class);
                    startActivity(searchedWords);
                    return true;
        case R.id.goSettings:
             Intent settings = new Intent(getApplicationContext(),
SettingsActivity.class);
                    startActivity(settings);
                    return true;
        default:
            return super.onOptionsItemSelected(item);
        }
    }
```

4 Verificación y pruebas

En el momento en que se tiene la aplicación desarrollada y se ha comprobado unilateralmente que no existen problemas en su utilización. Se ha de proceder a realizar una serie de gestiones para, primero ratificar su correcto funcionamiento y a continuación hacerla accesible al público.

4.1 Protocolo de pruebas de la aplicación

A pesar de que ya se considere que la aplicación no tiene problemas de uso ni "*bugs*" o errores de codificación, es imprescindible generar un juego de pruebas en las que se compruebe el correcto funcionamiento de todos los elementos de los que está compuesta.

Para este caso se han escogido como grupo de control a 5 usuarios con perfiles diferentes y se les ha asignado a cada uno un libro en un idioma conocido del que tenga un nivel bajo. El resultado esperado es que después de realizar la prueba, cada usuario demuestre un incremento en el vocabulario de dicho idioma.

4.1.1 Definición de la prueba

La prueba a realizar consiste en escoger una página al azar de un libro en el idioma a mejorar y interpretar su significado con la única ayuda de la aplicación. La intención de esta parte, es que el usuario haya obtenido una lista de cómo mínimo 10 palabras a aprender. En caso contrario, el usuario debería continuar con la página siguiente del libro hasta obtener dicha cantidad de palabras.

A continuación una vez comprobado que se ha entendido el correcto significado de la página escogida, el usuario ha de jugar al juego propuesto hasta que se hayan agotado las palabras a aprender.

4.1.2 Resultados obtenidos

Una vez realizada la prueba anterior, se revisan las palabras buscadas por todos los usuarios con el fin de comprobar si estos han mejorado el idioma propuesto a cada uno y si han memorizado dichas palabras.

Usuario	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
Dispositivo	Nexus S	Galaxy Nexus	Samsung Infuse	Samsung Galaxy Tab	HTC Desire
Par de idiomas	EN-PT	EN-ES	DE-EN	FR-EN	EN-IT
Palabras buscadas	15	12	18	16	23
Palabras memorizadas	12	10	7	6	12
% palabras memorizadas	80%	83.3%	38.8%	37.5%	52.17

Tabla 1: Tabla resultados prueba aplicación

El dato más relevante de la tabla anterior, es obviamente la cantidad de palabras memorizadas por cada uno de los cinco usuarios. Además, se puede observar que este dato varia notablemente de un usuario a otro.

A partir de estos resultados, se obtiene la conclusión de que cada usuario precisa de una cantidad diferente de apariciones de una palabra en los juegos por lo que se ha añadido a la pantalla de opciones la posibilidad de modificar la cantidad de palabras a buscar.

4.2 Google Analytics para Android

El SDK de Google Analytics para Android permite a los desarrolladores de Android Apps obtener información sobre la actividad de los usuarios dentro de sus aplicaciones, ofreciendo las herramientas para monitorizar las campañas de marketing que se quieran realizar sobre las aplicaciones.

De esta manera se pueden analizar:

- El número de usuarios que está usando las aplicación.
- Desde donde se usan la aplicación en el mundo.
- Adopción y uso de determinadas funcionalidades.
- Compras y transacciones dentro de la aplicación.
- Analizar los resultados de un anuncio.
- ETC...

4.2.1 Instalación

1- Alta de proyecto en Google Analytics: Lo primero que hay que hacer para instalar Google Analytics en Android es dar de alta la aplicación en analytics de manera similar a si de una web se tratase, pero en el punto en el que pregunta si el protocolo va a ser http o https, hay un tercera opción que dice "**Not a website**" al seleccionarla y darle a siguiente ya ofrece la descarga de la librería tanto para Android como para iPhone.

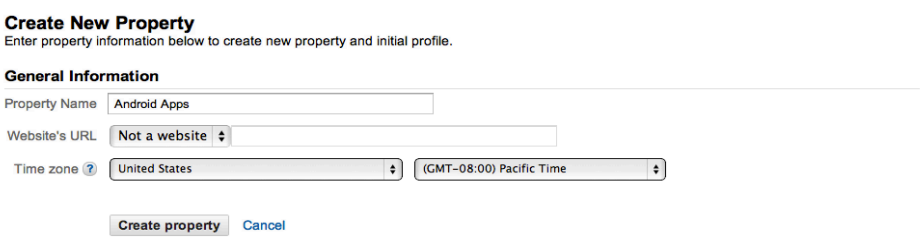


Ilustración 30 Proceso instalación Google Analytics -Alta

2- Inclusión de la librería en el proyecto: A continuación, el siguiente paso después de descargarse la librería, es importarla al proyecto de modo que se pueda trabajar con ella. Para ello, en Eclipse, dentro de las propiedades del proyecto, se pueden añadir librerías de manera muy sencilla.

Ya se ha visto anteriormente, que para todo proyecto en Android, es preciso definir qué elementos del teléfono se van a necesitar y que esto se define en el **AndroidManifest.xml**.

En el caso de la libreria de Google Analytics, hay que incluir los permisos de conexión a Internet y acceso al estado de la red.

INTERNET
ACCESS_NETWORK_STATE

Como el primero ya ha sido incluido para el funcionamiento de la aplicación solo hay que añadir el segundo, de modo que el archivo AndroidManifest.xml queda:

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

3- Definición de elementos a analizar: En este punto, la libreria de Google Analytics ya se encuentra instalada en el proyecto, por lo que el siguiente paso a realizar será definir que elementos del mismo se van a analizar.

Este es uno de los pasos más importantes de este punto ya que se han de tener en cuenta todas aquellas funcionalidades del proyecto como acciones pueda realizar el usuario y además se han de definir en cada una de las actividades del proyecto.

Además en este punto, cabe comentar que la API de Google Analytics ofrece algunos frameworks precocinados en los que de manera muy sencilla se pueda analizar diferentes activities. A pesar de las ventajas que ofrecen, estos frameworks, en general tienen un inconveniente en cuanto a determinadas funcionalidades de Google Analytics que son consideradas muy interesantes. Por ello se ha decidido inicialmente prescindir de ellos y utilizar la API básica de Google Analytics tal y como se explica a continuación.

Home/Search:
Esta actividad del proyecto es la encargada de realizar traducciones de palabras para el usuario así como de añadir contenido a la base de datos de la aplicación. Por ello, en este

caso, se analizarán las búsquedas realizadas tanto el número de las mismas como los pares de idiomas en los que se realicen.

Es por todo lo anterior, que en esta pantalla o actividad se añade un elemento de análisis del tipo de búsqueda que hace el usuario, en el cual se utiliza además una de las últimas funcionalidades creadas por Google Analytics: **Variables personalizadas**[3], de manera que se pueda analizar el par de idiomas que utilicen los usuarios para ver cual(es) son los más utilizados y ofrecer nuevas funcionalidades o ventajas a estos si se diese el caso de tener que centrarnos.

De manera que para poder incluir Google Analytics en la *Activity* inicial o de buscador es necesario añadir el siguiente código:

```
...
GoogleAnalyticsTracker tracker;
…
tracker = GoogleAnalyticsTracker.getInstance();
tracker.startNewSession("UA-31977959-1", this);
tracker.trackPageView("/HomeSearchActivity");
…
```

Y dentro del listener del botón de buscar:

```
tracker.setCustomVar(1, "Languaje pair",

AppDictus.getInstance().getOriginalLanguajeCode()+

AppDictus.getInstance().getTraslationLanguajeCode(), 2);
```

[3] Las variables personalizadas es una funcionalidad en la que se puede enviar el par nombre-valor de la variable y de este modo escalar los datos que se pueden analizar de tanto una web como una aplicación.

Game - Test:

Esta es junto con la anterior la *Activity* con más información que pueda interesar obtener. Por ejemplo, la cantidad de aciertos o fallos que suelen tener los usuarios y en qué idioma, o si se considera que está habiendo un progreso en su aprendizaje. Todo esto junto a mucha más información del usuario se puede obtener mediante las mentadas variables personalizables, pero en una primera etapa se ha decidido analizar únicamente el progreso del usuario, de modo que se analizan las palabras memorizadas por el usuario.

```
...
GoogleAnalyticsTracker tracker;
…
tracker = GoogleAnalyticsTracker.getInstance();
tracker.startNewSession("UA-31977959-1", this);
tracker.trackPageView("/game-test");
…
tracker.setCustomVar(2, "word learned", qWord.get_word(), 2);
```

Otras activities:

En cuanto a las otras tres *activities* del proyecto, como ya se ha explicado en puntos anteriores, son simplemente pantallas para mostrar información, a excepción de la activity de configuración, pero el actual dato de la misma (el par de idioma utilizado) ya se recoge en la primera *activity*.

Es por ello que en estas tres, por el momento, únicamente se va a analizar la información de que el usuario ha entrado en la misma y la referente a esta acción:

Tiempo de duración de su visita, visitas únicas, rebote,...

```
...
GoogleAnalyticsTracker tracker;
…
```

```
tracker = GoogleAnalyticsTracker.getInstance();
tracker.startNewSession("UA-31977959-1", this);
tracker.trackPageView("/....");
```

4.2.2 Resultados obtenidos

Una vez instalado y configurado Google Analytics para Android en la aplicación de este proyecto, se ha dejado pasar un periodo de tiempo prudente en el que diversas personas han probado la aplicación, lo que permite acceder a una pequeña estadística sobre el uso que se ha hecho de ella.

Al ser Google Analytics una plataforma orientada al análisis de visitas en páginas web, los resultados que se muestran, están inicialmente orientados a este tipo de análisis. A pesar de ello sigue siendo muy efectiva la información proporcionada.

A continuación se van a detallar algunos de los aspectos que Google Analytics para Android permite analizar, aquellos que se consideran más relevantes. Para ello se accede a la URL **http://analytics.google.com** y se introduce el nombre de usuario y contraseña utilizados en el momento de dar de alta el proyecto.

Resumen:

En la primera pantalla de la plataforma, se puede ver las visitas obtenidas en el último mes, en el caso de una aplicación móvil, serían accesos a la aplicación.

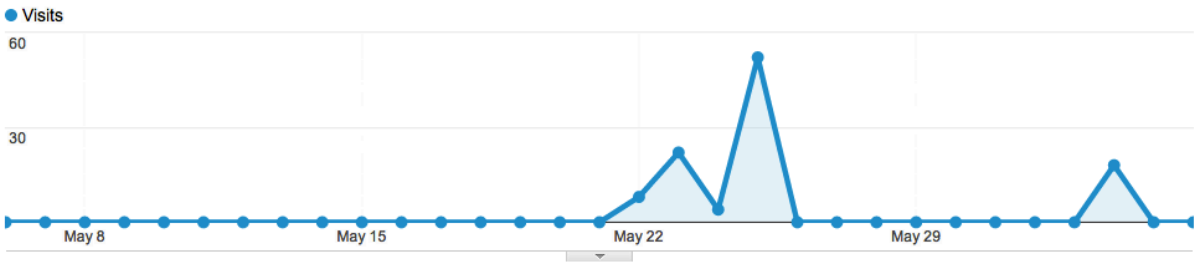


Ilustración 31 Google Analytics - Visitas/mes

Y de estos accesos, se puede obtener un pequeño análisis inicial, accesos totales en el periodo, usuarios únicos, pantallas visitadas, tiempo medio de visita,… Toda esta información, ha de ser traducida desde el lenguaje orientado a análisis web a lenguaje orientado a análisis de aplicaciones móviles: visita-acceso, visitante-usuario, pagina vista-pantalla vista,…

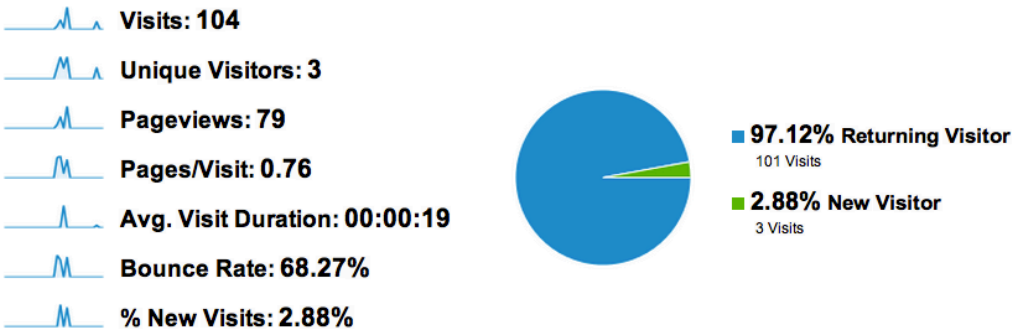


Ilustración 32 Google Analytics - Resumen mes

Detalle pantallas vistas:

Tanto en análisis web como en análisis de aplicaciones, uno de los aspectos que se consideran más relevantes por no decir el más importante, es que pantallas/páginas ha visitado el usuario ya que esta factor puede proporcionar al desarrollador o empresa información sobre los intereses de los usuarios.

	Page	Pageviews	% Pageviews
1.	/HomeSearchActivity	32	40.51%
2.	/SearchedWords	18	22.78%
3.	/testApplicationHomeScreen	13	16.46%
4.	/BadWords	7	8.86%
5.	/Game/Test	6	7.59%
6.	/TestGame	2	2.53%
7.	/Settings	1	1.27%

Ilustración 33 Google Analytics - Pantallas visitadas

Como ya se ha comentado, la imagen anterior se basa en un periodo de prueba, por lo que la información obtenida no es para nada concluyente.

Se puede observar que de todas las visitas obtenidas, un 40% ha accedido a la pantalla de portada, y el siguiente elemento más visitado es la pantalla de palabras buscadas. Y únicamente un 2.53% de las visitas ha accedido al juego.

Análisis de las variables personalizadas:

Se ha comentado en un punto anterior la posibilidad que ofrece Google Analytics para crear variables personalizadas con las que analizar conceptos específicos de un proyecto. En este caso, se selecciona como variable personalizada a analizar el par de idiomas escogido por los usuarios en el momento de hacer una búsqueda.

4.3 Puesta producción

Cuando un proyecto en Android se considera como finalizado desde el punto de vista de desarrollo y se quiere ofrecer a otros usuarios, es necesario firmar la aplicación para que el teléfono del usuario pueda mostrar información del proveedor al usuario. Además, de esta manera se consigue un control por parte de Google de los desarrolladores con el fin de garantizar cierta seguridad. Si se puede saber quien ha creado que aplicación y esta aplicación es nociva, se puede impedir que ese usuario suba más aplicaciones al Market.

4.3.1 Creación del archivo APK firmado

El proceso de firmado de una aplicación es, en general, muy sencillo, y se puede realizar de dos maneras. A través de la línea de comandos de cualquier sistema operativo o a través de eclipse con el ADK de Android. Para este caso, se ha utilizado la segunda opción por ser más visual y sencilla.

Paso 1:

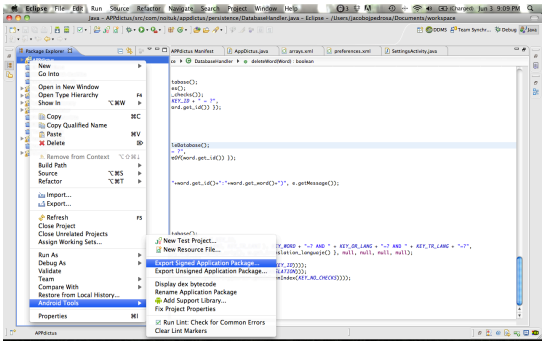


Ilustración 34 Creación .apk firmado 1

En la barra lateral de Eclipse, donde se listan los proyectos, se selecciona el proyecto que se quiere empaquetar, y con el botón derecho del ratón se abre el desplegable de opciones y dentro de **Herramientas de Android**, se selecciona **"Export Signed Application Package…"**.

Paso 2:

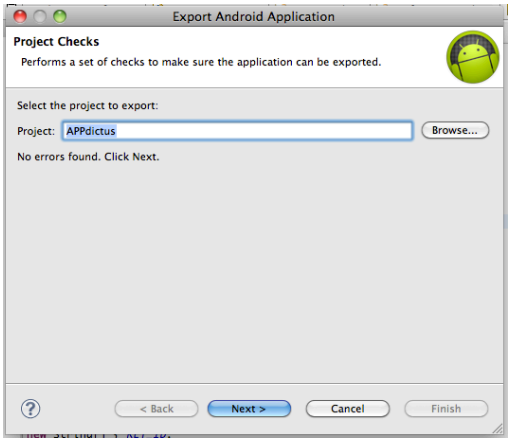


Ilustración 35 Creación .apk firmado 2

En este paso, simplemente se ha de ratificar que el proyecto es el seleccionado. En caso contrario, mediante el botón "*browse…*" se puede seleccionar otro proyecto a firmar.

Paso 3:

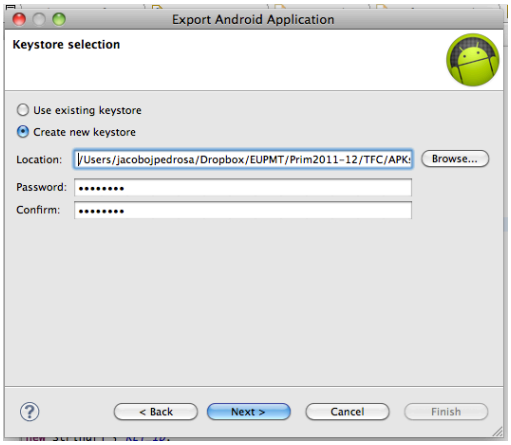


Ilustración 36 Creación .apk firmado 3

Si alguna vez se ha firmado una aplicación Android, es posible que se posea ya una firma y no sea necesario crearla.

En caso de no tenerla, en este paso, se selecciona la opción "**Create new keystore**", se selecciona la ubicación que va a tener y se le asigna un nombre. Por último se ha de definir una contraseña con la que validar dicha firma.

Paso 4:

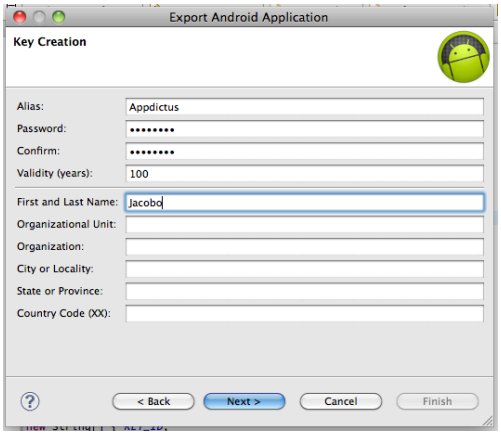


Ilustración 37 Creación .apk firmado 4

Este es el paso en el que el desarrollador ha de proporcionar su información y/o de su empresa para ligarla a la aplicación empaquetada.

Cabe destacar, que como mínimo se ha de proporcionar un dato acerca del autor.

Paso 5:

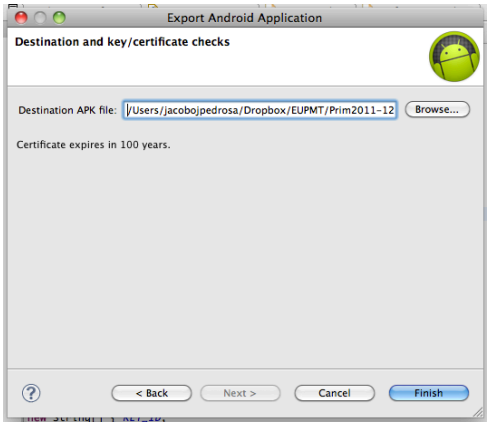


Ilustración 38 Creación .apk firmado 5

Este es el último paso a realizar, en este se define el directorio donde se va a ubicar la aplicación una vez firmada y empaquetada en formato .APK.

Una vez obtenido el archivo empaquetado, se puede instalar fácilmente en cualquier dispositivo con sistema operativo Android que cumpla los requisitos definidos en el AndroidManifest.xml que ya se ha comentado.

A continuación, este archivo empaquetado, se puede enviar de cualquier manera: mail, mensaje instantáneo, por almacenamiento usb,… y una vez dentro del teléfono móvil, instalarlo es cuestión de ejecutar el archivo y aceptar los permisos solicitados.

En este punto del proyecto, se considera que la aplicación ya está desarrollada y ha superado todas las pruebas. A demás, ya se ha explicado como empaquetar la aplicación en un formato que permita compartirla con otros usuarios de manera sencilla.

4.3.2 Subida al Market (Google Play)

Una manera de compartir una aplicación con los usuarios de Android en general, es la plataforma ligada a este Sistema Operativo llama Google Play Market. A ésta se puede acceder desde web o desde el propio móvil, escoger que aplicación se quiere instalar y instalarla. Además permite desde web realizar el proceso de instalación sin tener que interactuar con el dispositivo móvil, e incluso para usuarios que tienen varios dispositivos móviles gestionar que aplicaciones se tienen instaladas en que dispositivo.

Para poder publicar una aplicación en esta plataforma, esta requiere de cierta cantidad de información con el fin de poder ofrecerla a su público más objetivo. Lo primero que solicita es todo el material gráfico que se pueda facilitar para promocionar la aplicación: capturas de pantalla, logotipo y/o icono, gráfico promocional, capturas de pantalla, etc…

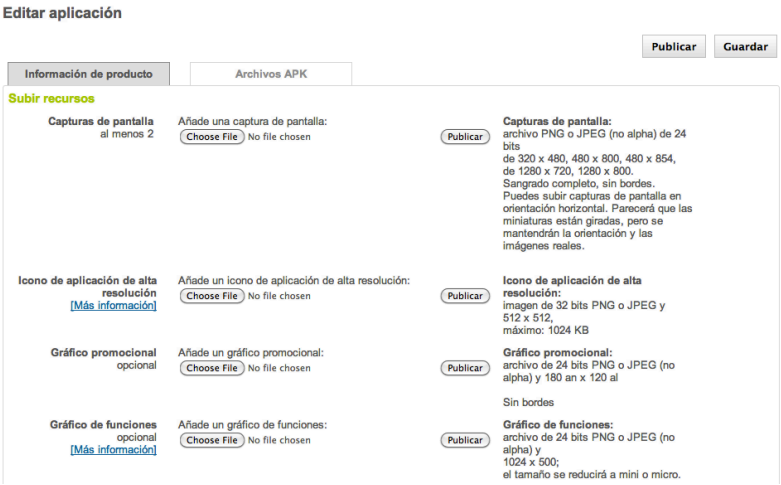


Ilustración 39 Alta en Market 1

A continuación, también solicita información por escrito de la aplicación a publicar, es decir, título, descripción, categoría, etc… Toda esta información, además ha de estar en cada idioma de los que se quiera publicar la aplicación. En el caso del presente proyecto, se ha escogido el inglés y el español, en ambos casos, la decisión ha sido tomada por el gran alcance

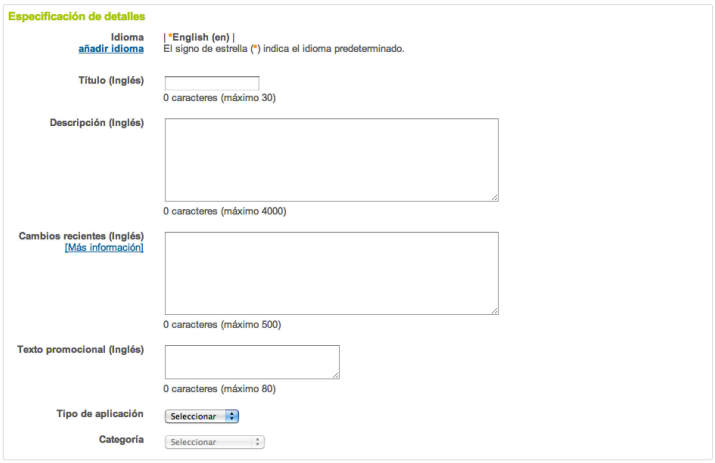


Ilustración 40 Alta en Market 2

en cuanto a público que tienen ya que entre los dos idiomas se abarca el 60% del público Android.

El siguiente grupo de peticiones del formulario, solicita información sobre el público al que va dirigido, tipo de licencia, grado de madurez de la aplicación, si va a ser gratuita o de pago y los países en los que se va a publicar la aplicación.

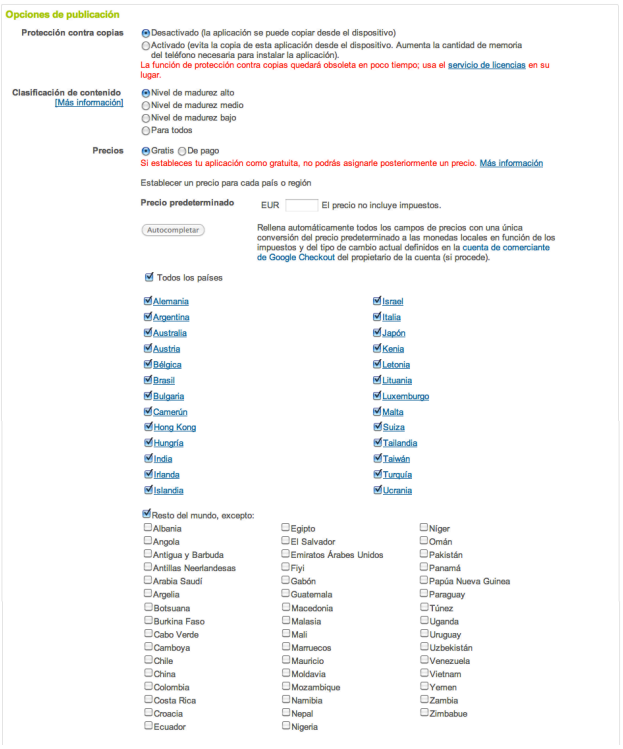


Ilustración 41 Alta en Market 3

Hay que tener en cuenta que de los países en los que se publique la aplicación, los idiomas en los que se habla en ellos. Es decir, no tiene sentido publicar una aplicación en Japón si los textos creados anteriormente son únicamente en castellano. Para este proyecto, se han escogido el Castellano y el Inglés como idiomas en los que publicar ya que con ellos se abarca aproximadamente un 60% del mercado en la mayoría de los países.

Por último, el formulario para publicar una aplicación, solicita información directa sobre el desarrollador para que los usuarios puedan ponerse en contacto con el así como su consentimiento para publicar la aplicación.

Ilustración 42 Alta en Market 4

5 Estudio económico

A continuación se procede a detallar el coste que ha tenido el desarrollo de este proyecto. Este coste está dividido entre los materiales precisados tanto para su desarrollo como para su testeo, las horas de ingeniería dedicadas para la definición del proyecto y por último horas de desarrollo invertidas.

5.1 Materiales

En cuanto a los materiales utilizados, como en la mayoría de proyectos de software, éste ha consistido en un equipo para poder realizar el desarrollo, 2 dispositivos móviles, un Smartphone y una Tablet sin modificación alguna por parte de la operadora en su sistema operativo y, por último, material gráfico para realización del diseño.

El valor a amortizar mensualmente para todos estos dispositivos, se ha calculado dividiendo su coste inicial entre los meses de uso esperado. A continuación se ha multiplicado este valor por 5 meses que se ha considerado como duración del proyecto.

Por otro lado, las licencias de software, al ser un proyecto desarrollado con aplicaciones que poseen licencia OpenSource no han comportado coste alguno.

Material	Precio(€)
Licencias de software	0
Equipo informático	225
Dispositivo móvil para testeo	137.5
Tablet móvil para testeo	208.34
Material gráfico	65
Total	**570.84€**

Tabla 2 - Materiales necesitados para el desarrollo del proyecto

Equipo informático : 1600€/36meses = 44.4€/mes. 44,4€/m*5meses = 225€

Smartphone: 300€/12meses = 27.5€/mes. 27.5€/mes*5meses = 137.5€

Tablet: 500€/12*5=208.34

5.2 Horas de ingeniería

Se definen como horas de ingeniería aquellas que han sido necesarias para la definición del proyecto: definición de la aplicación, fuentes de información que se van a utilizar (API), …

En cuanto al precio por hora de ingeniería, se ha realizado una búsqueda de los valores más comunes en presupuestos y se ha definido a partir de los resultados obtenidos quedando en 50€/h.

Actividad	Horas	Coste
Análisis de aplicaciones existentes	8	400€
Definición del proyecto	5	250€
Diseño del proyecto	25	1250€
Estudio de las APIs de traducción necesarias para el proyecto	60	3000€
Estudio de servicios de soporte	30	1500€
Total	**128**	**6400€**

Tabla 3 - Detalle horas de ingeniería dedicadas

5.3 Horas de desarrollo

En cuanto a las horas de desarrollo, igual que en el caso anterior, se ha hecho una búsqueda de los presupuestos más comunes para un desarrollados JAVA/Android y se ha obtenido una estimación de 20€/h

Actividad	Horas	Coste
Creación de la estructura de pantallas en Android	15	300€
Creación del proxy para la API de WordReference.com	30	600€
Creación de la base de datos SQLite	25	400€
Implementación de Google Analytics	20	300€
Maquetación del diseño	60	900€
Creación del Juego GameTest	30	200€
Total	**180**	**3600€**

Tabla 4 - Definición horas de desarrollo

5.4 Rentabilidad

Al inicio de este proyecto, se han comentado posibles modelos económicos que existen alrededor de las aplicaciones móviles:

- Venta directa
- Publicidad
- Servicios añadidos
- Ventas in-app

Para la explotación de este proyecto, se ha decidido hacerla mediante las dos primeras opciones, ya que actualmente, no se puede ofrecer ningún servicio añadido y la infraestructura necesaria para realizar ventas IN-APP está fuera de alcance.

Para hacer un análisis de rentabilidad de una aplicación Android mediante publicidad, se han utilizado los datos ofrecidos por la web http:// androidappprofits.com que hacen un estudio mensual de 6 aplicaciones (Air Horn Pro, Tazer Pro, Mutt Trainer Pro, Buzz Cut, Text Bomb, Impossiballz).
A continuación, se puede observar un resumen de ingresos del mes de abril de 2012 del conjunto de las aplicaciones anteriores:

- **Descargas totales:** 289,202
- **Descargas activas:** 48,990
- **Admob Banner Ads:** 136,556 Impresiones
- **Ingresos por mil impresiones (eCPM): $**1.05
- **Ingresos totales:** $142.76

Resumen:

- **142.76/289,202 = 0,0005centimos de dólar por descarga**
- **142.76/48,990 = 0,003 céntimos de dólar por descarga activa**

5.4.1 Escenarios de rentabilidad por publicidad:

Para realizar un cálculo estadístico de la rentabilidad de la aplicación desarrollada en este proyecto, se han planteado 2 escenarios. En ellos, se calcula la cantidad de usuarios a partir de una variable denominada **factor viral.** Este factor es un elemento muy utilizado para el cálculo viral en proyectos relacionados con redes sociales y viene a ser la cantidad de usuarios que se generan a partir de los usuarios ya existentes.

Por ejemplo, si en una aplicación se obtienen cada mes 1 usuario por cada 4 de los que ya existen, se considera que hay un factor viral de **1,25**. Si Cada mes duplica al anterior, su factor viral sería de **2**.

Escenario 1 (factor viral 1.25):

Para este primer escenario, se ha partido de la base que se poseen inicialmente 500 usuarios con la aplicación ya que en los días es sencillo captar usuarios al estar al principio de las listas.

mes	descargas	Descargas nuevas	Ingresos	Ingreso acumulado total
1	625	625	$1.88	$1.88
2	781	156	$0.47	$2.34
3	977	195	$0.59	$2.93
4	1,221	244	$0.73	$3.66
5	1,526	305	$0.92	$4.58
6	1,907	381	$1.14	$5.72
7	2,384	477	$1.43	$8.04
8	2,980	596	$1.79	$8.94
9	3,725	745	$2.24	$11.18
10	4,657	931	$2.79	$13.97
11	5,821	1,164	$3.49	$17.46
12	7,276	1,455	$4.37	$21.83
13	9,095	1,819	$5.46	$27.28
14	11,369	2,274	$6.82	$34.11
15	14,211	2,842	$8.53	$42.63
16	17,764	3,553	$10.66	$53.29
17	22,204	4,441	$13.32	$66.61
18	27,756	5,551	$16.65	$83.27
19	34,694	6,939	$20.82	$104.08
20	43,368	8,674	$26.02	$130.10
21	54,210	10,842	$32.53	$162.63
22	67,763	13,553	$40.66	$203.29
23	84,703	16,941	$50.82	$254.11
24	105,879	21,176	$63.53	$317.64

Tabla 5 - Escenario publicidad 1

Como se puede observar a partir de los resultados de la tabla anterior, por mucho que la aplicación tenga éxito los primeros días, con un factor viral bajo, la aplicación no será rentable con el modelo de publicidad.

Escenario 2 (factor viral 1.5):

En este segundo caso, se supone un factor viral más elevado, de cada 3 usuarios uno nuevo (1.5). Este es un factor elevado y requiere de gran comunicación de los usuarios a sus conocidos sobre el proyecto.

mes	descargas	Descargas nuevas	Ingresos	Ingreso acumulado total
1	750	750	$2.25	$2.25
2	1,125	375	$1.13	$3.38
3	1,688	563	$1.69	$5.06
4	2,531	844	$2.53	$7.59
5	3,797	1,266	$3.80	$11.39
6	5,695	1,898	$5.70	$17.09
7	8,543	2,848	$8.54	$8.04
8	12,814	4,271	$12.81	$38.44
9	19,222	6,407	$19.22	$57.67
10	28,833	9,611	$28.83	$86.50
11	43,249	14,416	$43.25	$129.75
12	64,873	21,624	$64.87	$194.62
13	97,310	32,437	$97.31	$291.93
14	145,965	48,655	$145.96	$437.89
15	218,947	72,982	$218.95	$656.84
16	328,420	109,473	$328.42	$985.26
17	492,631	164,210	$492.63	$1,477.89
18	738,946	246,315	$738.95	$2,216.84
19	1,108,419	369,473	$1,108.42	$3,325.26
20	1,662,628	554,209	$1,662.63	$4,987.89
21	2,493,943	831,314	$2,493.94	$7,481.83
22	3,740,914	1,246,971	$3,740.91	$11,222.74
23	5,611,371	1,870,457	$5,611.37	$16,834.11
24	8,417,056	2,805,685	$8,417.06	$25,251.17

Tabla 6 - Escenario de publicidad 2

Como se puede observar, a partir de los 2 años de lanzamiento, la aplicación comienza a ser rentable. Se ha de tener en cuenta el valor de casi 8,5 millones de usuarios que han de utilizar la aplicación para obtener dicha rentabilidad y que el factor viral especificado (cada dos usuarios generan uno nuevo) es totalmente utópico para una aplicación de este estilo.

5.4.2 Escenarios de rentabilidad de venta:

Como ya se ha comentado, para este proyecto, se ha decidido analizar el marco económico de publicidad en una aplicación, y el marco de venta de una versión profesional de la misma. Para ello, se ha analizado la proporción de descargas gratuitas frente a descargas de pago de una aplicación del mismo autor de este proyecto sobre las que se han realizado los cálculos.

Estas descargas vienen a ser, después de un año, de 4825 en la versión gratuita y 327 en la versión profesional. Por lo que la proporción gratuito-venta es de un 7,6%. A partir de este dato y de los factores virales del punto anterior, tenemos:

mes	descargas	Descargas nuevas	Ingresos	Ingreso acumulado total
1	538	538	$533.31	$533.31
2	579	41	$40.70	$574.01
3	623	44	$43.80	$617.81
4	671	48	$47.15	$664.96
5	722	51	$50.74	$715.70
6	777	55	$54.62	$770.31
7	837	59	$58.78	$8.04
8	900	64	$63.27	$892.37
9	969	69	$68.10	$960.46
10	1043	74	$73.29	$1,033.75
11	1123	80	$78.89	$1,112.64
12	1208	86	$84.91	$1,197.55
13	1301	92	$91.38	$1,288.93
14	1400	99	$98.36	$1,387.29
15	1507	107	$105.86	$1,493.15
16	1622	115	$113.94	$1,607.10
17	1745	124	$122.64	$1,729.73
18	1879	133	$132.00	$1,861.73
19	2022	143	$142.07	$2,003.80
20	2176	154	$152.91	$2,156.71
21	2342	166	$164.58	$2,321.29
22	2521	179	$177.14	$2,498.42
23	2713	192	$190.65	$2,689.08
24	2921	207	$205.20	$2,894.28

Tabla 7 - Escenario de rentabilidad de venta

Como se puede observar en la tabla anterior, el modelo de venta es también muy poco rentable, pero a partir del doceavo mes desde su publicación, genera unos ingresos cercanos a los 100€/mes.

5.5 Resumen y conclusión económica

Concepto	Horas	Valor/hora	Coste
Materiales	--	--	570.84€
Horas de ingeniería	128	50	6400€
Horas de desarrollo	180	20	3600€
Total	**303**		**10.570,84€**

Tabla 8 - Resumen costes proyecto

A partir de estos resultados, está claro que es muy difícil que la aplicación sea rentable, o por lo menos los ingresos a dos años deberían ser bastante elevados para llegar al punto de inflexión en menos de 3 años.

Como ya se ha visto, con el fin de calcular la rentabilidad del proyecto se han realizado varias pruebas en las que se han modificado la viralidad del proyecto para, a partir de unos ingresos medios, saber que condiciones serían optimas. El resultado obtenido, es que requeriría de gran cantidad de descargas con el fin de que sea rentable.

6 Conclusiones

El objetivo inicial de este proyecto viene a partir de la idea de generar una aplicación móvil en plataforma Android y de este modo poder realizar un estudio en profundidad de este sector tanto a nivel técnico como a nivel económico.

La aplicación, que como ya se ha visto, se trata de una herramienta de traducción que ofrece a su vez un sencillo ejercicio al usuario que le permite memorizar las palabras buscadas y de este modo mejorar su vocabulario.

En referencia a la parte técnica del proyecto, cabe destacar los problemas que de trabajar con servicios externos como la API de Google Translate o cualquier otra. De este modo, el desarrollador tiene que estar pendiente de cambios en el protocolo del servicio para que la aplicación funcione correctamente. Además, el servicio puede cambiar sus condiciones, lo cual afectaría a la base del funcionamiento del proyecto. En el caso actual, el servicio de Google Translate cambió de ser gratuito a de pago en poco mas de 6 meses, por lo que se han tenido que buscar alternativas gratuitas que permitan alcanzar el objetivo del proyecto, como WordReference, que a pesar de ofrecer más información por traducción, es muy limitado en cuanto a pares de idiomas se refiere.

Por otro lado, la plataforma Android y su marco de trabajo, se han mostrado como una buena herramienta, bastante sencilla de utilizar y de escalar. Las mayores dificultades encontradas, se han dado, sobretodo, en el área visual de la aplicación. En un desarrollo Android, se ha de tener en cuenta que el mismo desarrollo va a aparecer en multitud de dispositivos y con diferentes tamaños de pantallas, desde las 3” de un teléfono hasta las 10,1” de una tablet, por lo que los diseños han de ser adaptativos.

En el marco económico del proyecto, se ha realizado un pequeño estudio de la rentabilidad del mismo, y se ha demostrado que es muy difícil obtener gran rentabilidad económica de una aplicación media. Para solventar esta circunstancia, es preciso generar acciones que por un lado incrementen notablemente el número de descargas, ya sean de publicidad o por venta, y por otro incrementar los ingresos producidos en cada modelo.

7 Ampliaciones y mejoras

Como ya se ha visto, la aplicación desarrollada en este proyecto se trata de una versión de estudio, más teórica que práctica en el marco de trabajo de las aplicaciones móviles. A pesar de ello, los resultados obtenidos durante las pruebas realizadas con usuarios reales, indica gran *engagement* de cara a los usuarios, ya que el *feedback* obtenido por los mismos ha sido bastante positivo. Por tanto se ha decidido hacer ciertas ampliaciones:

Mejorar y ampliar los juegos

Uno de los elementos más interesantes del proyecto es precisamente el y los posibles juegos que se puedan realizar con las palabras buscadas. El juego ofrecido (*GameTest*), es un juego totalmente educativo y no se puede considerar precisamente como "entretenido". Es por ello que la ampliación más importante a realizar, es la generación de cómo mínimo 2 o 3 juegos más que ofrezcan esta característica.

Mejorar y ampliar funcionalidades de traducción:

Por otro lado, se ha considerado como muy interesante el análisis de los pares de idiomas utilizados por los usuarios para ofrecerles mejoras a partir de la información obtenida. Es decir, si el par lingüístico más utilizado, es el de inglés-español, se pueden generar u obtener librerías y funcionalidades exclusivas para este par de idiomas.

Social:

En el punto de análisis económico, se ha demostrado que para obtener mayores ingresos en la aplicación y para que esta sea lo más rentable posible, es necesaria gran viralidad por parte de la misma. Para ello, una de las herramientas más eficientes hoy en día, son las redes sociales. Por todo esto, se pretende añadir funcionalidades dentro de la aplicación que permitan al usuario compartir con su circulo sus logros dentro de la aplicación: Cuanto ha mejorado su vocabulario en un mes, o cuantas palabras ha buscado, ….

8 Bibliografía

[1] Zigurd Mednieks, Laird Dornin, G.Blake Meike & Masumi Nakamura - ***Programming Android*** - O'REILLY

[2] Marko Gargenta - ***Learning Android*** - O'REILLY

[3] Lauren Darcey, Shane Conder – ***Android Application Development*** - SAMS

[4] http://market.android.com - ***Android Market*** – mayo 2012

[5] http://bit.ly/Lgj84N - ***Español Traductor / Diccionari*** - mayo 2012

[6] http://bit.ly/ug29NA - ***Traductor de Google*** - mayo 2012

[7] http://bit.ly/LvVVvY - ***Translate*** - mayo 2012

[8] http://bit.ly/Korlrd - ***Inglés Traductor*** – mayo 2012

[9] http://bit.ly/L0e4EM - ***El Traductor de viaje*** – mayo 2012

[10] http://code.google.com/apis/language/translate/overview.html - ***Google Translate*** - marzo 2012

[11] http://code.google.com/apis/language/translate/v2/getting_started.html – ***Google Translate*** - febrero 2012

[12] http://en.wikipedia.org/wiki/Representational_State_Transfer – ***Wikipedia*** - mayo 2012

[13] http://code.google.com/p/google-api-translate-java/ - ***Google Translate*** – enero 2012

[14] http://www.wordreference.com/docs/api.aspx – ***WordReference*** - enero 2012

[15] http://blog.mobclix.com/index/PDF/120112_mobclixx_holiday_shopping_infographic.pdf - ***Mobclix*** - Junio 2012

[16] http://blog.mobclix.com/2011/02/10/mobclix-index-monthly-value-of-an-app-user/
http://bit.ly/gAShXx – Mobclix - Mayo 2012

[17] http://androidappprofits.com/ - ***Android APP profits*** – Junio 2012

9 Contenido del CD

- Portada del proyecto
- Memoria del proyecto en formato PDF
- Diseños
    - Diseño global en PSD
    - Pantalla inicio en PNG
    - Pantalla Juego *GameTest* en PNG
    - Pantalla todas las palabras buscadas en PNG
    - Pantalla palabras no aprendidas en PNG
    - Pantalla *settings* en PNG
- Imágenes
    - Imágenes del menú
    - Botones de aplicación
    - Fondos
- Código de la aplicación
- Aplicación empaquetada en formato APK