UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESIS

PROCESO DE DESARROLLO INDEPENDIENTE DE UNA APLICACIÓN MÓVIL ANDROID

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

GUSTAVO ADOLFO JUÁREZ RODRÍGUEZ

DIRECTOR DE TESIS:
ING. CARLOS ALBERTO ROMÁN ZAMITIZ

CIUDAD UNIVERSITARIA 4/NOVIEMBRE/2013

A mis padres

Y aquellas personas que estuvieron a mi lado en este importante instante de mi vida

ÍNDICE

Capítulo 1
Introducción

Una aplicación móvil, término proveniente del inglés *App,* que es una contracción de *application* con un tono informal, es un software que puede instalarse en un dispositivo móvil con la finalidad de extender sus funcionalidades, al igual que ocurre con la instalación de programas en computadoras de escritorio o portátiles [1].

Las diversas utilidades de las aplicaciones móviles, ya sea para comunicarnos en redes sociales, jugar, comprar artículos, leer noticias, leer libros electrónicos, tomar fotografías, etc., han pasado a formar parte de los hábitos de consumo de los usuarios de teléfonos inteligentes y tabletas. Son además un reflejo de nuestras aficiones y gustos, son un producto que nos hace sentirnos identificados con nuestros conocidos o con alguna marca comercial.

El componente social de una aplicación móvil es muy importante. Los usuarios comparten las aplicaciones de su preferencia con los amigos, pueden compartir contenido con sus conocidos, disfrutando de ellas con los demás. Las tiendas de aplicaciones fomentan el descubrimiento de nuevas aplicaciones a través de listas de popularidad y permitiendo su valoración por aquellos usuarios que las han probado.

Una vez localizada una aplicación se puede descargar directamente desde el dispositivo si se cuenta con una conexión a internet, ya sea vía *Wi-Fi* o por uso de un plan de datos del proveedor de telefonía celular. Se puede instalar también por medio del ordenador, en este caso de debería descargar primero la aplicación y posteriormente sincronizar el dispositivo e instalar en él todas las aplicaciones pendientes.

Sin duda alguna las *apps* móviles son, hoy en día, unas herramientas de comunicación, venta y fidelización de clientes muy importante que muchas empresas no pueden obviar en sus estrategias corporativas y acciones que busquen resultados tanto tangibles como intangibles; tanto a corto como a medio y largo plazo.

1.1. El Mercado de las aplicaciones móviles a nivel mundial

El 10 de julio de 2008 Apple inauguraba su tienda de *apps* para iPhone e iPod Touch con más de 800 aplicaciones. En apenas cuatro días de su estreno se habían alcanzado las 10 millones de descargas. A partir de ese momento se marcaba el modelo a seguir en los próximos años. Se establecía un modelo de negocio que estaba por cambiar el desarrollo para dispositivos móviles y la industria de contenidos.

Las tiendas de aplicaciones se han convertido en un factor determinante a la hora de escoger un dispositivo por parte de los clientes, y no puede desaprovecharse oportunidad alguna en un mercado tan competitivo como este. Es por eso que los fabricantes de dispositivos y de sistemas operativos se han apresurado a poner en marcha sus propias

tiendas cuanto antes. Para que una tienda de *apps* funcione, atraer el interés de los desarrolladores externos es esencial. Los desarrolladores preferirán colocar sus productos en las tiendas que más exposición les den y mayores ingresos les hagan obtener.

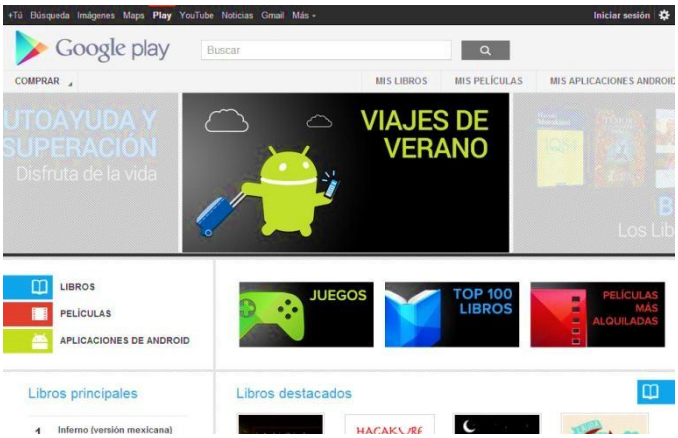

Figura 1.1 Captura de pantalla de Google Play Store, tienda de aplicaciones para el sistema operativo móvil Android. https://play.google.com/

Existe una gran variedad de aplicaciones móviles listas para descarga en las tiendas de *apps*, las más comunes y más descargadas son de los siguientes tipos:

- **Juegos.** Son la categoría más popular, siempre se encuentran entre las más descargadas. Un gran ejemplo de esto es el juego *Angry Birds,* desarrollado en 2009 por *Rovio* para el iPhone. Se ha convertido en el juego para móviles más descargado en toda la historia.
- **Compras.** Algunas tiendas y outlets en línea como *Mercado Libre, Amazon* y *Privalia* utilizan aplicaciones para que sus compradores puedan adquirir sus productos directamente desde el móvil.
- **Redes Sociales.** Las grandes redes sociales han estado siempre entre las listas de lo más descargado en las tiendas de *apps.* Facebook y Twitter brindan una opción móvil para seguir conectados. WhatsApp trajo consigo una nueva forma de enviar mensajes, se vale de la conectividad Wi-Fi o red 3G para ahorrar significativamente el costo de los mensajes.
- **Prensa, radio y televisión.** La información sobre lo que pasa en el mundo ahora se tiene, literalmente, al alcance de la mano. Podemos leer las noticias del día, escuchar noticieros en aplicaciones de radio por internet. De igual forma podemos escuchar música y ver programas con fines de entretenimiento en nuestro móvil.
- **Productividad.** Agendas, software de escritura de notas, listas de cosas por hacer y otras utilidades suelen ocupar los primeros lugares en descargas. Uno de los mejores ejemplos es *DropBox,* un servicio de almacenamiento en la nube.

Actualmente el mercado de aplicaciones móviles ha alcanzado un nivel de ventas significativo. En junio de 2013 Apple publicó un comunicado anunciando que la App Store de Apple había superado los 50 mil millones de descargas efectuadas para dispositivos móviles con sistema operativo iOS [2].

En contraste, Google revelaba en esas mismas fechas en una conferencia, que su tienta de aplicaciones Play Store había alcanzado 48 mil millones de descargas. Una diferencia no muy grande en comparación, tan sólo de 2 mil millones. En el mismo comunicado Apple asegura que los clientes descargan aplicaciones a un ritmo aproximado de 800 por segundo y 2 mil millones por mes.

Figura 1.2. Captura de pantalla de Apple Strore, tienda de dispositivos móviles y aplicaciones de Apple. http://www.store.apple.com/mx

Por su parte Larry Page, director ejecutivo de Google, anunció en marzo de 2013 que Android había alcanzado un total de 750 millones de activaciones, es decir, 750 millones de dispositivos móviles entre smartphones y tabletas habían sido activados con alguna versión del sistema operativo Android [3]. Esta información convierte a Android en el sistema operativo móvil más utilizado en el mundo. En un análisis realizado por Benedict Evans (figura 1.3), FreeLancer en análisis estadísticos, se puede observar el rápido crecimiento y aceptación que ha tenido la plataforma de Android en los últimos 2 años. La cantidad de Activaciones pasó de los 100 millones a 750 millones, incluso se espera que se llegue al billón de activaciones a finales de este año.

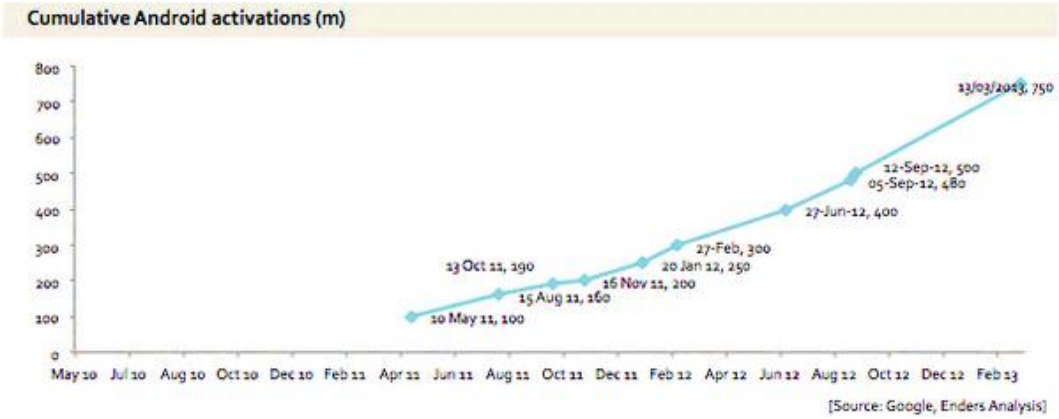

Figura 1.3. Grafica de activaciones de dispositivos Android (en millones) contra tiempo en bimestres.

Estas cifras son de esperarse pues los móviles se están convirtiendo en el principal medio de acceso a internet con un 69% de usuarios, quienes se conectan a internet desde cualquier dispositivo móvil. De entre los usuarios que se conectan por medio de un dispositivo móvil 61% lo hacen a través de un Smartphone, 37% a través de una notebook

y 22% a través de una Tablet [4]. El dato que más interesa es que 71% de estos usuarios han descargado *apps* en sus dispositivos alguna vez. Como vemos, el mercado de aplicaciones móviles abarca una gran cantidad de usuarios. Año con año, al obtener nuevas cifras, la cantidad de usuarios, activaciones de dispositivos y descargas de aplicaciones crecen exponencialmente.

Si bien el mercado de aplicaciones móviles es, hoy por hoy, una opción con muchas oportunidades, lo será más en cuestión de pocos años. Es un mercado que no puede ser desaprovechado por ninguna empresa que desee soluciones innovadoras para mejorar su rendimiento. Inclusive es un mercado que no puede ser desaprovechado por los desarrolladores independientes de aplicaciones.

1.2. La situación en México

En septiembre de 2011 el fundador de la empresa mexicana desarrolladora de Software E-Life, Luis Trillo, declaró en un artículo para la revista InformationWeek que en nuestro país existen excelentes desarrolladores de aplicaciones para dispositivos móviles pero al haber pocos, se contratan a precios altos [5]. En Estados Unidos existe mayor competencia en esta área, tanto que llega a barato contratar a un estadounidense, más barato que a un mexicano. Luis Trillo aseguró también que buscan capacitar desarrolladores mexicanos con el objetivo de hacer crecer este mercado en nuestro país y de esa manera se puedan desarrollar tantas aplicaciones como se hace en otros países como Brasil, India o Estados Unidos.

Ese mismo año México fue el país que presentó el mayor crecimiento del mundo y la región, en acceso a redes sociales desde conexiones móviles [6]. Para éste entonces se tenía conocimiento de cerca de 500 empresas dedicadas al desarrollo de aplicaciones móviles, de las cuales más de la mitad están clasificadas como Pymes. Las soluciones móviles que se pueden desarrollar son prácticamente infinitas y México está frente a una gran oportunidad. Se requieren soluciones móviles que faciliten la interacción de las personas con los distintos dispositivos móviles, aplicaciones que brinden información acerca de promociones y servicios de tiendas departamentales o comercios pequeños. Sin embargo, las empresas mexicanas están trabajando en el desarrollo de aplicaciones de este tipo de una manera algo lenta y con cierto retraso tecnológico [7].

Aunque esta información se tiene desde hace un par de años la situación no ha cambiado mucho, hablando un poco de números la industria del software en México alcanzó en 2012 un valor de 2,300 millones de dólares 11% más que en 2011, se espera incluso que para el presente año se convierta en una oportunidad de mercado para jóvenes emprendedores al tener un crecimiento del 10% [8]. El objetivo es convertirnos en una nación que exporte soluciones en tecnología. Y como se ha estado planteando, una de las tendencias más importantes en la industria del software es la movilidad.

1.3. El desarrollo independiente de aplicaciones.

Después de éste panorama general del mercado de aplicaciones móviles y su impacto tanto social como económico, es preciso hablar del desarrollo de estos “productos de software”. Como se mencionó antes, para 2011 se tenían registradas cerca de 500 empresas desarrolladoras de aplicaciones móviles en el país. Pero también es posible que una sola persona o un grupo de programadores desarrollen una *app*, la publiquen en

cualquiera de las tiendas registradas y obtengan ingresos de ella, sin la necesidad de pertenecer a alguna empresa. Esto es a lo que se le llama “desarrollo independiente”. Como consecuencia, en el desarrollo independiente no se cuenta con el financiamiento de una empresa, es posible que tampoco se tenga la disponibilidad de una infraestructura tecnológica adecuada e incluso es probable que la *app* cuente con campañas publicitarias austeras. Entonces, ¿Por qué es importante para este trabajo de tesis el desarrollo independiente de aplicaciones?

Como declaró Luis Trillo, en nuestro país existen excelentes desarrolladores de aplicaciones para dispositivos móviles pero al haber pocos, se contratan a precios altos. Habiendo tantas áreas de oportunidad en nuestro país en desarrollos móviles, es necesario convertirnos en una nación que explote este mercado y que sobresalga a nivel mundial. Se espera un crecimiento del 10% en la industria del Software en México este año y puede ser aprovechado por los jóvenes emprendedores.

1.3.1 El objetivo de esta tesis.

Considerando el anterior preámbulo el objetivo de esta tesis es realizar un ejercicio de desarrollo independiente de una aplicación móvil con un seguimiento a todo el proceso, desde el nacimiento de la idea hasta la publicación de la aplicación en alguna de las tiendas de *apps.* Se espera con esta actividad, que los emprendedores que se encuentren interesados hallen una guía para llevar a cabo sus primeros desarrollos o mejoren sus prácticas de desarrollo, y que puedan experimentar el ámbito económico y de resultados de sus aplicaciones.

Para comenzar a darle seguimiento al proceso de desarrollo de nuestra aplicación de ejemplo es necesario definir los pasos que se llevarán a cabo a lo largo de esta tesis. Analizar el mercado es sustancial, aunque se han tomado puntos importantes en los subtemas anteriores, falta precisar para qué plataforma se desarrollará nuestra aplicación, cuál es la más conveniente para nuestro ejercicio. Posteriormente se analizarán las opciones de monetización de nuestra *app* y obtendremos la que mejor resultados pueda darnos. Será obligatorio conocer el funcionamiento de la plataforma que hemos elegido y las mejores prácticas de desarrollo de aplicaciones móviles. Consecutivamente se propondrá la aplicación de ejemplo, justificándola (se buscará que aborde temas de la carrera de Ingeniería en Computación) y explicando todos sus elementos. Entonces la aplicación se desarrollará de acuerdo a metodologías de la Ingeniería de Software. Finalmente teniendo la *app* terminada se llevará a cabo el registro como desarrollador en la tienda de aplicaciones y se dará de alta nuestro producto para ser comercializado.

Cabe destacar que no es un proceso corto ni sencillo pero será tan complejo como cada persona desee. La meta es llevar una aplicación móvil a la tienda y monetizarla para obtener algo de la experiencia que exige el mundo laboral en esta área de la Ingeniería de Software.

Para elegir la plataforma móvil en la cual es más conviene desarrollar nuestro ejemplo, se tomó en cuenta la información en la tabla de la figura 1.4. Existen tres principales opciones: iOS, Android y Windows Phone, cada una con su respectiva tienda. Para poder llevar a cabo un proyecto de programación para la plataforma iOS es necesario trabajar en un equipo de cómputo con plataforma Mac; para una aplicación en Windows Phone, es necesario programar con una computadora Windows; pero para desarrollar una aplicación en Android, se puede trabajar con Mac, Windows o Linux. Esta última opción es la más versátil.

Plataforma móvil	iOS	Android	Windows Phone
			
Tienda de aplicaciones	App Store	Google Play	Windows Marketplace
Lenguaje de programación	Objetive C, C++	Java, C++	C#, varios
Número de activaciones 1Q 2013	37.4 millones	162.1 millones	7 millones
Plataforma de desarrollo	Mac	Windows, Mac, Linux	Windows
Costo por publicar	$99 USD / anual	$25 USD una sola vez	$99 USD / anual

Figura 1.4. Tabla comparativa de las tres plataformas móviles de mayor predominio a nivel global. La información de interés incluye el costo de publicación de aplicaciones, la plataforma de desarrollo y el número de dispositivos activados durante el primer trimestre de 2013 [9].

El número de dispositivos activados bajo cada plataforma durante el primer trimestre de 2013, es de 162.1 millones para Android, seguido de iOS con 37.4 millones y después Windows Phone con 7 millones [9]. Esto refleja el alcance que tiene cada una de las opciones, existen más dispositivos con Android en el mundo que con cualquier otra plataforma. Al elegir Android como opción habrá una mayor cantidad de dispositivos que puedan descargar nuestra *app*. En cuanto a cuestión económica, considerando que no se tiene el financiamiento de alguna empresa, la mejor opción es claramente Android. Mientras que las dos opciones restantes tienen una cuota anual como desarrollador de 99 dólares, Android cobra una cuota única de 25 dólares.

Por su versatilidad, su mayor alcance y su bajo costo en la publicación, se eligió a Android como la mejor opción para desarrollar nuestra aplicación móvil de ejemplo. Es la opción que se halla más al alcance de cualquier emprendedor que desee realizar una *app* como la que se desarrollará en esta tesis.

Ya que se ha elegido la plataforma para la cual se ejecutará nuestro ejemplo, en los siguientes capítulos, se presentará la información necesaria acerca de la plataforma móvil y sus mejores prácticas de desarrollo; así como las opciones de monetización disponibles. Para posteriormente, pasar al planteamiento de nuestra aplicación.

Capítulo 2 Monetización de aplicaciones

Se ha hablado brevemente del impacto de las *apps* en la actualidad, y parte de lo que las hace tan importantes es la gran derrama económica que implican. Según un estudio realizado por la empresa Wavefront y la Fundación México Estados Unidos para la Ciencia (FUMEC) se estima que para el 2014, el consumo de aplicaciones alcanzará un valor de 35,000 millones de dólares [7]. En las tiendas de aplicaciones pueden verse un sin número de aplicaciones a bajo costo o en descarga gratuita, aun así, pueden generar gran cantidad de ingresos.

Es importante conocer los medios de obtención de ganancias de un desarrollo móvil, así podremos explorar las opciones de monetización de una aplicación y encontrar la idónea para obtener ingresos de nuestro trabajo. En éste capítulo se expondrán las opciones de monetización más utilizadas y se hablará del modelo de negocios basado en publicidad, ya que nuestra aplicación de ejemplo será distribuida mediante este modelo utilizando la red de publicidad móvil AdMob.

2.1 Opciones de monetización

Una aplicación móvil puede estar basada en una buena idea, funcionar muy bien y ser elegida por muchos usuarios. Pero, en el mercado de los móviles, eso no se traduce automáticamente en ganancias. Una aplicación gratuita y sin publicidad difícilmente generará ingresos, por más usuarios que consiga. Existen diversas opciones de generar ingresos mediante una aplicación móvil a través de las tiendas de aplicaciones. Para obtener una cantidad de ingresos óptima es importante saber elegir la opción de monetización más adecuada de acuerdo a nuestra *app.* Las cuatro opciones de monetización más empleadas, de entre varias existentes, son las siguientes:

- ***Aplicaciones de pago.*** Uno de los caminos para generar ingresos es establecer un precio por la descarga de la aplicación. Por un lado resulta una opción conveniente, cada vez más usuarios están dispuestos a pagar el costo de una *app* si la encuentran realmente útil. Sin embargo, el mercado es altamente competitivo, el problema viene al enfrentarse a apps gratuitas similares. Esta opción es viable para aplicaciones únicas y con una idea original, que brinde servicios o resultados que ninguna otra aplicación pueda lograr. Un ejemplo de esto es el videojuego *Chaos Rings* Ω, el cual no se encuentra en versión gratuita y su precio actual es de 11 Euros.

- ***In app Purchase.*** Una alternativa para ofrecer una aplicación de manera gratuita y aun así ganar dinero es liberar características adicionales a cambio de un pago, vender bienes virtuales dentro de la aplicación o mantener servicios bajo suscripción. Dropbox, que es una aplicación multiplataforma, ofrece un almacenamiento gratuito en la nube de 2 GB, pero la capacidad de almacenamiento puede ser incrementada hasta 500 GB a un costo de 39 Dólares anuales.

- ***Apps gratuitas con In-App Advertising.*** Existe la opción de vender un espacio publicitario. El objetivo es que la aplicación sea descargada y conocida para incrementar el alcance de los anuncios publicitarios, se necesita un gran volumen de usuarios para generar fuertes ganancias. Algunas de las aplicaciones más populares han liberado versiones gratuitas basadas en publicidad para mejorar sus ingresos. El espacio publicitario puede consistir en algún anuncio al abrir la aplicación, en envío de notificaciones periódicas al usuario o colocar un banner publicitario al inferior de la aplicación. Éste banner publicitario cambiará de anuncios cada cierto tiempo y cuando el usuario toque el anuncio será dirigido a la página del anunciante.

- ***Apps para pymes.*** Esta opción no genera ingresos para el desarrollador a través de las tiendas de *apps,* pero si en la venta del producto a empresas. Las aplicaciones móviles juegan un papel fundamental en la estrategia de negocio, a la vez que crece la demanda de los usuarios, esto orilla a las empresas a liberar *apps* de mayor calidad más rápidamente y con mayor frecuencia. Los empresarios entienden la necesidad de desarrollos móviles que les permitan mejorar su rentabilidad operativa. Puede ser que la empresa requiera de alguna aplicación para regalar por medio de alguna de las tiendas y de esta manera ampliar su experiencia con el cliente. También puede ser que necesiten alguna herramienta en dispositivos móviles que les ayude a ejecutar sus procesos.

La opción de monetización que se usará para nuestro ejemplo es *In-App Advertising*. Google cuenta con la herramienta de publicidad móvil AdMob, que facilitará el trabajo de colocación de banners publicitarios dentro de nuestra *app*, y como estaremos trabajando bajo el esquema de publicidad, será preciso aprender un poco más sobre cómo funciona un modelo de negocios basado en publicidad.

2.2 Modelo de negocios basado en publicidad.

Internet ha promovido la aparición de nuevos modelos de negocio y la transformación de los tradicionales en su proceso de adaptación a la red. Por modelo de negocio entendemos la manera en que un negocio genera ventas y beneficios a través de la aportación de valor y la satisfacción de las necesidades de sus clientes. Muchos sitios en internet y muchos de los servicios que disfrutamos en la web son gratuitos y muchos nos hemos preguntado de qué forma perciben sus ganancias todas esas nuevas empresas que incursionan en internet.

El éxito del modelo de negocios basado en publicidad depende de tener un tráfico muy elevado o especializado de visitas, basándose en contenidos atractivos para usuarios en general o para un segmento específico del mercado. Éste modelo es similar al que se utiliza en la radio o en la televisión y es una fuerte fuente de ingresos de grandes portales

como Google, You Tube y Facebook. Estos portales tienen un tráfico inmenso y los ingresos generados por la publicidad junto con otros servicios les permiten ser rentables.

Para explicar cómo funciona exactamente el modelo se tomará un ejemplo, es aquí donde importa el tráfico generado por un portal web o una aplicación móvil.

Supongamos un sitio web que alcanza las 20,000 visitas únicas (cantidad de usuarios reales que ingresan al sitio en un periodo de tiempo) y que en cada una de esas visitas miran tres páginas del sitio (*pageviews*), además, el sitio se actualiza semanalmente, por lo que los visitantes accederán a él cuatro ocasiones en un mes (visitantes que retornan). Esto es lo que definimos como tráfico [10].

La publicidad de sitios web y aplicaciones móviles está dada por espacios publicitarios, los cuales, son ocupados por avisos de publicidad llamados *Banners*. Los banners son imágenes que se ubican, por lo regular, en la parte superior o al costado derecho de una página web. El objetivo de estos avisos es atraer el interés de los visitantes y que al hacer clic en la imagen sean dirigidos a las páginas de los anunciantes.

Éstos espacios publicitarios se pueden contratar de dos principales maneras, el costo por mil CPM, que consiste en que el anunciante pague una cuota para que un mismo anuncio aparezca 1000 veces y el CPC (costo por clic) que consiste en que el anunciante pague determinada cantidad al sitio web cada vez que un visitante de clic en el banner. Supongamos entonces que para el ejemplo que hemos planteado se vende publicidad bajo la modalidad CPM y que las tarifas son las siguientes: banner superior 12 dólares, banner lateral 9 dólares. Además se considerará que cada una de las páginas del sitio cuenta con un banner superior y uno lateral.

Una empresa está administrando el espacio publicitario del sitio y recibe un 35% de comisión. También se considera que no se ha vendido la totalidad de los espacios del sitio, sólo un 50%. En la figura 2.1 se muestran todos los datos he hemos sugerido en este ejemplo.

Visitas únicas	**20,000**
Páginas vistas por visita	3
Veces que retornan	4
Tarifa banner superior	U$ 12
Tarifa banner lateral	U$ 9
Ocupación	50%
Comisión	35%

Figura 2.1 Datos que se utilizarán para calcular el ingreso mensual por concepto de publicidad.

Ahora que se cuenta con todos los datos necesarios se calculará el ingreso mensual por concepto de publicidad para este ejemplo.

Se cuenta con un total de 20,000 visitas únicas y los usuarios retornan 4 veces, visitando en promedio tres páginas.

$$Pageviews = 20{,}000(4)(3) = 240{,}000$$

$$CPM\ por\ página = U\$\ 12 + U\$\ 9 = U\$\ 21$$

$$ganancia\ por\ visita = \frac{U\$\ 21}{1000} = U\$\ 0.021$$

$$Ingreso = (pageviews)(ganancia\ por\ visita) = (240{,}000)(U\$\ 0.021)\ = U\$\ 5040$$

El ingreso máximo por el espacio publicitario considerando el tráfico del sitio sería de 5040 dólares, pero debe considerarse la ocupación de publicidad en el sitio y la comisión por pagar a la empresa administradora. Por lo tanto:

$$Ingreso\ total\ mensual = U\$\ 5040\ (0.5)(0.65) =\ U\$\ 1638$$

El sitio web que hemos planteado genera ingresos mensuales por 1638 dólares únicamente por concepto de publicidad. Si además el sitio realiza ventas u ofrece servicios las ganancias se elevarían mucho más. Los sitios web con enormes volúmenes de tráfico mucho mayores a este ejemplo, obtienen gran cantidad de ingresos en publicidad, aunque no se basan únicamente en ella para ser rentables. Cabe mencionar que la publicidad en sitios web o en aplicaciones móviles es algo que no se puede pasar por alto, se pueden generar ingresos considerables sin que el usuario final de la aplicación o el visitante de algún sitio web tenga que pagar por sus contenidos. La aplicación propuesta en esta tesis se distribuirá utilizando este modelo de negocios, de manera que pueda ser adquirida gratuitamente en la tienda Google Play.

2.3 Google AdMob.

AdMob es una red de publicidad móvil multiplataforma, disponible para Android, iOS y Windows Phone. Cuenta con una amplia gama de anuncios para dispositivos móviles y tabletas. Por sus miles de anunciantes, permite obtener ingresos del tráfico de cualquier aplicación móvil [11].

Una vez desarrollada una aplicación móvil, se deberá instalar el kit de desarrollo *SDK* de AdMob más reciente para implementar los anuncios. Registrarse en AdMob es completamente gratuito y se obtendrán ingresos cada vez que los usuarios hagan clic en los anuncios de la aplicación, por lo que es necesario una cuenta bancaria donde serán depositados dichos ingresos.

2.3.1 Los Anuncios

Los anuncios están agrupados por categorías dependiendo del contenido del servicio o producto ofertado. Se puede filtrar la categoría de los anuncios que se desea aparezcan en la aplicación. Como forma predeterminada la *app* recibe anuncios para el público en general, pero los filtros de los anuncios permiten incluir contenido adecuado para una edad o bloquear anuncios de *URL* específicas, siendo el caso de una *URL* de la competencia. Los tipos de filtros que ofrece AdMob son los siguientes:

- **Filtros de URL:** impiden que los anuncios que enlazan a una determinada *URL* o dominio en internet aparezcan en la aplicación.
- **Filtros de texto:** impiden que se publiquen anuncios que contengan una determinada frase o algún texto en específico.
- **Filtros de categoría:** impiden que se muestren determinadas categorías de anuncios, como anuncios de imagen y anuncios aptos por edad.
- **Filtros de idioma:** impide que los anuncios en determinados idiomas se muestren en la aplicación.

AdMob ofrece a los desarrolladores un total control con respecto al lugar donde se colocarán los anuncios dentro de su aplicación. Para obtener un rendimiento óptimo, es recomendable colocar los anuncios en ubicaciones naturales para el público, como en la parte superior o inferior de la *app,* de manera que no interfieran con la interacción del usuario.

AdMob permite solamente tres tamaños de banner para tabletas, además del tamaño 320 X 50 que se muestra en los teléfonos. En caso de que no exista espacio suficiente en la pantalla del dispositivo para mostrar el anuncio, éste simplemente no se mostrará. En la figura 2.2 se muestran las distintas opciones de tamaño de banners disponibles.

Tamaño en Pixeles (Ancho X Alto)	Tipo de anuncio	Disponibilidad
320 x 50	Banner estándar	Teléfonos y tabletas
300 x 250	Rectángulo mediano	Tabletas
468 x 60	Banner de tamaño completo	Tabletas
728 x 90	Skyscraper horizontal	Tabletas

Figura 2.2. Tamaños de banners disponibles de AdMob y ejemplo del banner estándar para teléfonos y tabletas.

Consciente de la importancia de realizar un seguimiento del rendimiento de las aplicaciones, AdMob genera informes dentro de los cuales se pueden analizar las siguientes métricas para optimizar el rendimiento y tomar decisiones inteligentes que mejoren los ingresos:

- **Solicitudes de red:** número de veces que la aplicación solicita que se muestre un anuncio.
- **Número de impresiones**
- **Total de clics**
- **Porcentaje de clics**
- **Ingresos**
- **eCPM (costo por cada mil impresiones efectivo):** Indicador de los ingresos de desarrolladores o editores por cada mil impresiones del anuncio. El eCPM se calcula como (ingresos/impresiones) x 1,000.

2.3.2 Pagos

AdMob ofrece las siguientes opciones de pago:

- **PayPal:** Es la forma más eficiente de recibir el pago. Basta con registrar dentro de su cuenta de AdMob la dirección de correo electrónico que se utiliza para acceder a PayPal y se enviarán los pagos directamente a esa cuenta.
- **ACH (Exclusivo para editores de EE.UU)**
- **Transferencia bancaria:** Para recibir el pago mediante transferencia bancaria se deben registrar los siguientes datos:
  - Nombre del titular
  - Nombre del banco
  - Dirección del banco
  - Número de cuenta del titular
  - Código SWIFT

Se debe considerar que los pagos de los editores se envían en dólares estadounidenses, y que cada banco podría cobrar un recargo por hacer la conversión a moneda nacional.
Los pagos se comienzan a tramitar un mes después del mes natural en el que se generaron dichos ingresos. Hay un tiempo de tramitación de dos a tres semanas aproximadamente. Para poder iniciar la tramitación se debe alcanzar el límite del método de pago correspondiente. Siendo 20 USD si se ha seleccionado PayPal y 100 USD si se ha seleccionado transferencia bancaria. Para poder recibir entonces el pago, debe alcanzar el límite. Si un mes los ingresos totales por pagar son inferiores al importe del límite de pago, AdMob añade los ingresos pendientes de pago a los del mes siguiente. En la figura 2.3 se muestra un ejemplo de cómo procedería el pago de los ingresos de una *app* teniendo en cuenta que no se alcanzó el límite de la opción de pago durante el primer mes del año.

Figura 2.3. Considerando que se ha registrado la opción de pago por PayPal, suponga que en enero se obtuvieron ingresos de 10 USD y en febrero de 50 USD, se recibiría un pago por valor de 60 USD a mediados de abril.

2.3.3 Directrices y políticas

Para mantener un servicio de calidad, AdMob tiene un conjunto de políticas y directrices que deberán ser cumplidas por los editores. En caso de no cumplir con las políticas, AdMob se reserva el derecho de inhabilitar la publicación de anuncios en la aplicación, así como inhabilitar la cuenta de usuario en cualquier momento. Las políticas más importantes a tener en cuenta como editor, son las siguientes:

- *Clics e impresiones no válidos:* Los editores no pueden hacer clic en sus propios anuncios ni utilizar medios para incrementar de forma artificial el número de impresiones o clics, incluido cualquier tipo de método manual. Google analiza todos los clics y todas las impresiones para determinar si se adecúan a algún patrón de uso que pueda aumentar de forma fraudulenta los costes de un anunciante o los ingresos de un editor.

- *Número de anuncios por página:* El número de anuncios en una sola pantalla no debe ser superior a uno si el anuncio está fijo en la parte superior o inferior de la pantalla. En caso de que la página pudiera desplazarse, únicamente debe poderse ver un anuncio en pantalla a la vez.

- *Anuncios en aplicaciones:* Las aplicaciones sólo deben utilizar el SDK para solicitar anuncios.

- *Botón +1:* Los editores no pueden vender ni ceder a ningún tercero, datos relacionados con la actividad que los usuarios hagan del botón +1 de Google o de funciones similares.

- *Colocación de anuncios:* Los anuncios no se deben colocar ni muy cerca ni debajo de botones u otros objetos en los que los usuarios puedan hacer clic accidentalmente al interactuar con la aplicación. Los anuncios no se deben colocar en una ubicación que cubra u oculte cualquier área en la que pudieran tener interés los usuarios durante la interacción habitual. Los anuncios no se deben colocar en una pantalla “sin salida”. Debe haber una forma de salir de la pantalla sin hacer clic en el anuncio (Por ejemplo, un botón Atrás o Menú). De lo contrario, se debe notificar al usuario de que con el botón inicio se sale de la aplicación.

En este capítulo se ha definido la opción y las herramientas que se utilizarán para monetizar nuestra aplicación de ejemplo. Antes de pasar al planteamiento de la aplicación que se pretende desarrollar es ineludible conocer la plataforma Android y las prácticas más recomendadas para realizar una aplicación interesante y atractiva para los usuarios.

Capítulo 3
Aplicaciones móviles en Android

Una aplicación móvil es un instrumento que cumple con una o varias tareas en específico, y cuando esa aplicación es distribuida a través de una tienda de *apps* es porque está diseñada para un grupo de usuarios. Para Android es importante que las aplicaciones que sean distribuidas a través de la Google Play Store cumplan con tres principales objetivos, los cuales están dirigidos completamente a la satisfacción de sus usuarios [12]:

- *Cautivar:* Las aplicaciones Android deben ser estéticamente atractivas en múltiples niveles, las transiciones, la tipografía, los iconos y el diseño deben combinar simplicidad y belleza.
- *Simplificar la vida:* Las aplicaciones Android deben ser intuitivas y hacer la vida más fácil. Las tareas sencillas nunca requieren procedimientos complejos, y las tareas complejas para el usuario deben ser realizadas de manera sencilla.
- *Motivar:* No es suficiente que una *app* sea fácil de usar, deben motivar a las personas a probar cosas nuevas y a utilizar las aplicaciones de nuevas e imaginativas formas. Además de hacer sentir al usuario que las aplicaciones son personales y parte de su vida.

En este capítulo se explorarán algunas de las prácticas que Android propone que hacen de las aplicaciones móviles productos que logran los objetivos planteados. También se expondrán algunos de los elementos que conforman la plataforma móvil Android, importantes en el diseño de una aplicación.

3.1 Interfaz de usuario

Los elementos de la interfaz de usuario de Android son importantes para mantener una consistente y agradable experiencia de uso. Es útil conocerlos para desarrollar aplicaciones que exploten las capacidades de esta interfaz de la mejor forma, creando sinergia entre la app y el sistema operativo. Los elementos de interés para esta interacción son los siguientes:

- Pantalla *Home*, pantalla de todas las aplicaciones y pantalla de aplicaciones recientes
- Barras del sistema
- Gestos y reacción *Touch*
- Notificaciones
- *Widgets*

La pantalla *Home* es un espacio personalizable que aloja accesos rápidos, carpetas y *Widgets,* se puede navegar entre diferentes paneles de la pantalla *Home.* La barra de favoritos, ubicada al fondo mantiene a la vista los accesos rápidos más comunes sin

importar qué panel se esté mostrando. Desde esta pantalla se puede acceder a otra pantalla con todas las apps instaladas en el dispositivo, tocando el botón *All Apps* ubicado en el centro de la barra de favoritos.

Figura 3.1. Distintas pantallas de la interfaz gráfica de Android.

La pantalla de todas las aplicaciones permite elegir entre las aplicaciones y *widgets* instalados en el dispositivo. Los usuarios pueden arrastrar y soltar íconos de apps o *widgets* y colocarlos en cualquier espacio vacío de la pantalla *Home*.

El botón *Recents* al lado derecho de la barra de navegación despliega las aplicaciones con las que el usuario ha interactuado recientemente. Éstas se despliegan a manera de lista y son organizadas de forma que la app usada más recientemente es colocada al fondo. Se pueden eliminar elementos de la lista al deslizarlos hacia la derecha o hacia la izquierda. En la figura 3.1 se pueden observar las pantallas y los botones que se han mencionado.

Las barras del sistema son áreas de la pantalla dedicadas a desplegar notificaciones e información del estado del dispositivo, también agilizan la navegación entre las *apps* y las pantallas. Las barras del sistema se muestran en pantalla junto con las aplicaciones pero se pueden ocultar temporalmente para permitir al usuario disfrutar diversos contenidos como imágenes y video en pantalla completa. En la figura 3.2 se indica la ubicación en pantalla de las barras del sistema, tanto para teléfonos como para tabletas. A continuación se describirá cada una de ellas.

Barra de estado: Despliega notificaciones pendientes del lado izquierdo, mientras que del lado derecho muestra información del dispositivo como la hora, nivel de batería e intensidad de señal. Se puede deslizar hacia abajo para mostrar detalles sobre las notificaciones.

Barra de navegación: Esta característica es nueva para teléfonos con versiones de Android 4.0, contiene los botones de navegación *Back*, *Home* y *Recents*.

Barra combinada: Para tablets las barras de estado y navegación están combinadas en una sola, ubicada al fondo de la pantalla.

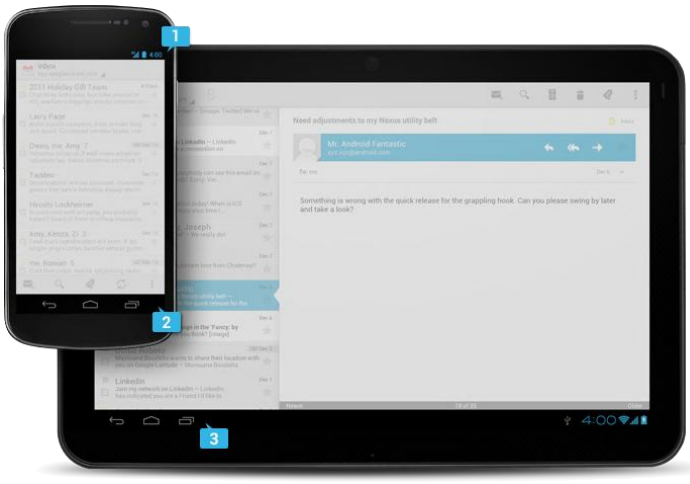

Figura 3.2. Barra de estado (1), barra de navegación (2) y barra combinada (3).

Los Gestos permiten a los usuarios interactuar con las aplicaciones manipulando los objetos en pantalla. Android permite siete principales gestos, descritos a continuación.

Tocar (*Touch*). Ejecuta la funcionalidad por default de un objeto dado. Cómo realizarla: tocar, soltar.
Presión Larga (*Long press*). Permite elegir uno o más objetos en una vista y realizar una acción con ellos usando un menú contextual. Cómo realizarla: tocar, esperar, soltar.
Deslizar (*Swipe*). Se utiliza para navegar entre vistas, paneles y contenido que no puede mostrarse completamente en la pantalla. Cómo realizarla: tocar, mover, soltar.
Arrastrar (*Drag*). Mueve elementos dentro de una vista o información dentro de un contenedor. Cómo realizarla: presión larga, mover, soltar.
Doble toque (*Double Touch*). Realiza un zoom in a imágenes u otros objetos. También usado para selección de texto. Cómo realizarla: Dos toques sucesivos.
Acercar (*Pinch open*). Realiza un *zoom in*. Cómo realizarla: tocar con dos dedos, separarlos, soltar.
Alejar (*Pinch close*). Realiza un *zoom out*. Cómo realizarla: tocar con dos dedos, juntarlos, soltar.

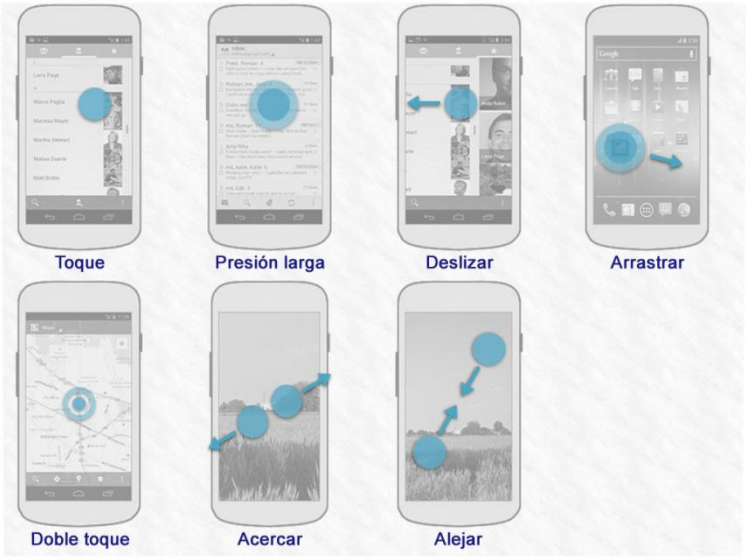

Figura 3.3. Principales gestos disponibles para la plataforma Android.

Para que el usuario perciba el resultado de los gestos es preciso utilizar colores e iluminación. Cuando el usuario toque un área interactiva dentro de la app, sería adecuada una respuesta visual que le permita saber que objeto fue tocado. También se recomienda indicar qué acciones están habilitadas o deshabilitadas. Cuando un objeto es presionado se recomienda iluminarlo de un color distinto, cuando el objeto esta deshabilitado se puede reducir su luminosidad.

Las notificaciones del sistema permiten a las aplicaciones mantener al usuario informado acerca de eventos, como nuevos mensajes de chat o eventos en el calendario. Éstas aparecen en la barra de estado como íconos, pero al desplegarse la barra los detalles importantes de cada notificación aparecerán. Se recomienda que cuando el usuario toque el cuerpo de la notificación se abra la aplicación en el lugar donde se puedan ejecutar acciones referentes a la información enviada por la notificación. En la figura 3.4 se identifican los elementos de una notificación.

Figura 3.4. Las notificaciones contienen un ícono principal que puede ser una fotografía o alguna otra imagen. Se muestra también la hora en la que apareció en la barra de estado y un icono secundario opcional. Este icono puede ser distintivo de la aplicación que manda la notificación.

Las notificaciones pueden activar la vibración del dispositivo, encender el *LED* de notificaciones o usar sonidos de alerta. El usuario debe sentir un control sobre las acciones de la *app*, por lo que es necesario que las notificaciones sean opcionales. También se debe poder elegir el sonido de alerta y el color de la luz del *LED*.

Para que un usuario no se sienta fastidiado por las notificaciones evite interrumpirlo con información o mensajes irrelevantes. Por ejemplo, evite informar al usuario sobre operaciones técnicas como guardar o sincronizar información. Evite interrumpir informando de un error dentro de la app si este se puede solucionar sin la ayuda del usuario. No cree notificaciones que no contengan información valiosa y hagan publicidad de la misma u otras aplicaciones. Para que una notificación sea de interés para el usuario debe considerar información de eventos en el tiempo y que involucren a otras personas. Los recordatorios del calendario son un buen ejemplo ya que en ocasiones implican citas con otras personas o eventos muy importantes que se llevarán a cabo el día en curso. Por ejemplo, el título y mensaje de una notificación de este tipo puede ser: “**Cena con Violeta.** 8:30 Restaurante El Jalisciense”

Los *widgets* son un aspecto esencial en la personalización de la pantalla *Home,* pueden ser acomodados en espacios disponibles dentro de los paneles de esta pantalla. Los *widgets* muestran la información más importante de una *app* o realizan funcionalidades esenciales sin necesidad de pasar a una aplicación. Por esta causa, cuentan con ciertas limitaciones. Dado que residen en la pantalla *Home*, deben respetar su forma de navegación. Esto limita el soporte a gestos de los *widgets,* las únicas acciones disponibles son toque (*Touch*) y deslizamiento vertical (*Vertical Swipe*). Un ejemplo de *widget* se aprecia en la figura 3.5, tiene la funcionalidad de ver distintas fotografías y permite el gesto de deslizamiento hacia abajo para cambiar la imagen que se está observando.

Figura 3.5. *Widget* que muestra distintas imágenes, permite deslizamiento vertical para cambiar de imagen.

Los *widgets* son un gran mecanismo para atraer a un usuario a la aplicación, promoviendo nuevo e interesante contenido disponible. La estrategia es mostrar sólo una parte de la información de interés, brindando la oportunidad de que el usuario utilice la aplicación para enterarse de más detalles. Además de mostrar información, también se pueden proporcionar enlaces a áreas de uso frecuente de una aplicación, esto permite a los usuarios completar tareas más rápido y extiende las funciones de la *app* a la pantalla *Home.*

3.2 Principios de diseño

El equipo de experiencia de usuario Android diseño una serie de principios de diseño de aplicaciones para mantener atraídos a los usuarios. Se recomienda seguirlos y añadir un poco de creatividad, siempre con la finalidad de mejorar la experiencia de quien utilizará las *apps.*

- *Los objetos reales son más divertidos que botones y menús.* Permita a las personas tocar y manipular objetos en su app, esto reduce el esfuerzo cognitivo necesario para realizar una tarea, esto resulta emocionalmente satisfactorio.
- *Permíteme personalizarlo*. Las personas aman añadir toques personales porque esto ayuda a hacerlos sentir cómodos. Provea ajustes personalizados que no interfieran con la finalidad de la aplicación.

- *Conóceme*. Aprenda las preferencias de los usuarios con el tiempo. Por ejemplo, en lugar de hacerlos buscar la misma información una y otra vez, coloque las búsquedas anteriores más recurrentes.
- *Sea conciso.* Use frases cortas con palabras simples, las personas suelen ignorar frases si éstas son largas.
- *Las imágenes son más rápidas que las palabras.* Considere usar imágenes para expresar ideas. Éstas llaman la atención de las personas y pueden ser mucho más eficientes que las palabras.

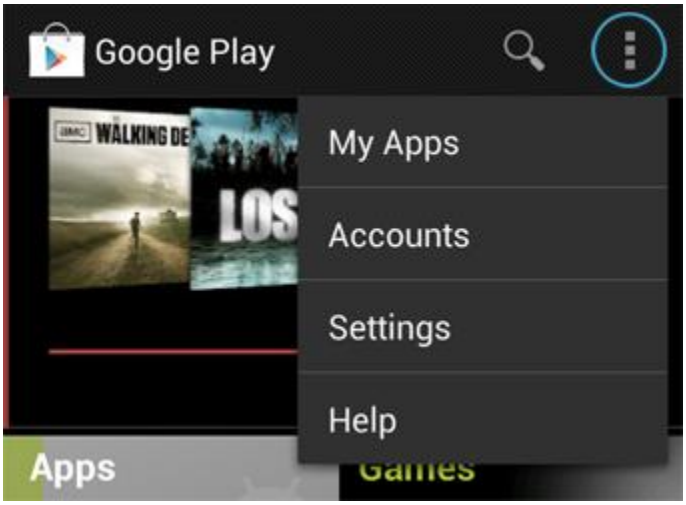

Figura 3.6 En esta aplicación, al tocar el icono con los tres puntos de la parte superior se desplegarán las opciones de mis aplicaciones, cuentas de usuario, ajustes y ayuda. Opciones que permanecerán ocultas para el usuario hasta que requiera de ellas.

- *Decide por mí pero déjame tener la última palabra.* Proponga la mejor opción antes de preguntársela al usuario, demasiadas opciones y decisiones hacen a las personas infelices.
- *Sólo muestra lo que necesito cuando lo necesito.* Las personas se sienten frustradas cuando reciben mucha información a la vez. Oculte las opciones que no son esenciales por el momento y conduzca al usuario por el camino adecuado. En la figura 3.6 se muestra un ejemplo, la opción remarcada con un circulo azul desplegará más opciones al tocarla, en caso de que sean necesarias.
- *Siempre debo saber dónde estoy.* No permita que el usuario se pierda dentro de su aplicación haga que cada lugar luzca distinto y use transiciones para mostrar relaciones entre pantallas.
- *Interrúmpeme sólo cuando sea necesario.* Evite interrumpir al usuario con información de poca importancia, las personas quieren estar concentradas y a menos que sea algo critico o de suma importancia, una interrupción puede ser frustrante.
- *No es mi culpa.* Sea gentil en cómo corrige los erres dentro de la aplicación. Si algo va mal brinde instrucciones claras de cómo resolver el problema explicando los detalles técnicos. Si usted puede corregir el error sin que el usuario lo note, mucho mejor.
- *Haz las cosas importantes con rapidez.* No todas las acciones son iguales, decida lo que es más importante en su app y hágalo fácil de hallar y rápido de usar. Por ejemplo, el botón de capturar la imagen en una cámara o el botón de pausa en un reproductor de música.

El estilo de escritura dentro de una aplicación móvil es muy importante, es la forma en la que habrá comunicación con el usuario. El usuario no debe aburrirse con tanta explicación ni sentir que no se le trata como una persona inteligente. Por lo tanto, se deben tener en cuenta las siguientes recomendaciones.

Sea conciso, simple y preciso. Se propone un límite de 30 caracteres, incluyendo espacios, y no use más a menos que sea absolutamente necesario. Por ejemplo:
Demasiado formal: "Consulte la documentación que viene con su teléfono para más instrucciones"
Adecuado: "Lea las instrucciones que vienen con su teléfono"

Suponga que usted interactúa con una persona inteligente y competente, pero que no conoce de jerga técnica y puede que no entienda el español a la perfección. Deberá usar palabras cortas, verbos sencillos y pronombres comunes. Por ejemplo, de la pantalla de ajustes de ubicación:
Confuso: "**Utilizar satélites GPS** Cuando usa localización precisar a nivel de calle"
***Adecuado: "*GPS** Permitir a las aplicaciones conocer tu ubicación"

Sea amigable. Hable directamente con el lector utilizando segunda persona, no sea abrupto o fastidioso. Por ejemplo, el cuadro de dialogo que aparece cuando una aplicación deja de responder:
Molesto: "Sorry! La Actividad MyAppActivity(in aplication MyApp) no está respondiendo. - Forzar cierre – Esperar – Reportar"
Adecuado: *"*MyApp no está respondiendo ¿Quieres cerrarla? –Esperar – Reportar – Cerrar"

Coloque lo más importante primero. Las primeras dos palabras deben incluir por lo menos una muestra de la información más importante del texto. Ejemplo:
La acción importante al final: "Toque 'siguiente' para completar la configuración de la conexión *Wi-Fi*"
La acción importante primero: "Para finalizar la configuración *Wi-Fi* toque 'siguiente' "

Describa sólo lo que sea necesario y nada más. No trate de explicar a detalle, muchos usuarios podrían confundirse. Ejemplo:
Demasiada información: *"*Conectando… Tu teléfono necesita conectarse con los servidores de Google para conectarse a tu cuenta. Ésta acción puede tomar hasta cinco minutos"
Adecuado: "Conectando… Tu teléfono está contactando con Google. Esto puede tomar hasta 5 minutos"

Evite la repetición. Si un término significativo se repite continuamente encuentre una forma de mostrarlo sólo una vez.

En algunas situaciones, cuando un usuario realiza una acción en una *app,* es buena idea confirmar o informar (*Acknowledge*) de la acción a través de texto. Confirmar es pedir al usuario que verifique si realmente quiere proceder con la acción que pretende realizar. En algunos casos, la confirmación se presenta con alguna advertencia o información crítica acerca de la acción que necesita considerar. Informar es desplegar cierto texto que le permita saber al usuario que la acción que acaba de realizar se ha completado con éxito. Esto elimina la incertidumbre acerca de acciones que el sistema realiza implícitamente.

Al informar, el texto puede ir acompañado con la opción de deshacer la acción realizada. Comunicar al usuario con estos métodos previene que las personas cometan errores que puedan lamentar, como la eliminación de archivos o información importantes.

No todas las acciones implican una confirmación o un *acknowledgment*. El diagrama de flujo de la figura 3.7 ayuda a determinar, dependiendo de la situación en la que se encuentre la *app*, si es necesario informar, confirmar o continuar normalmente con el curso de la aplicación.

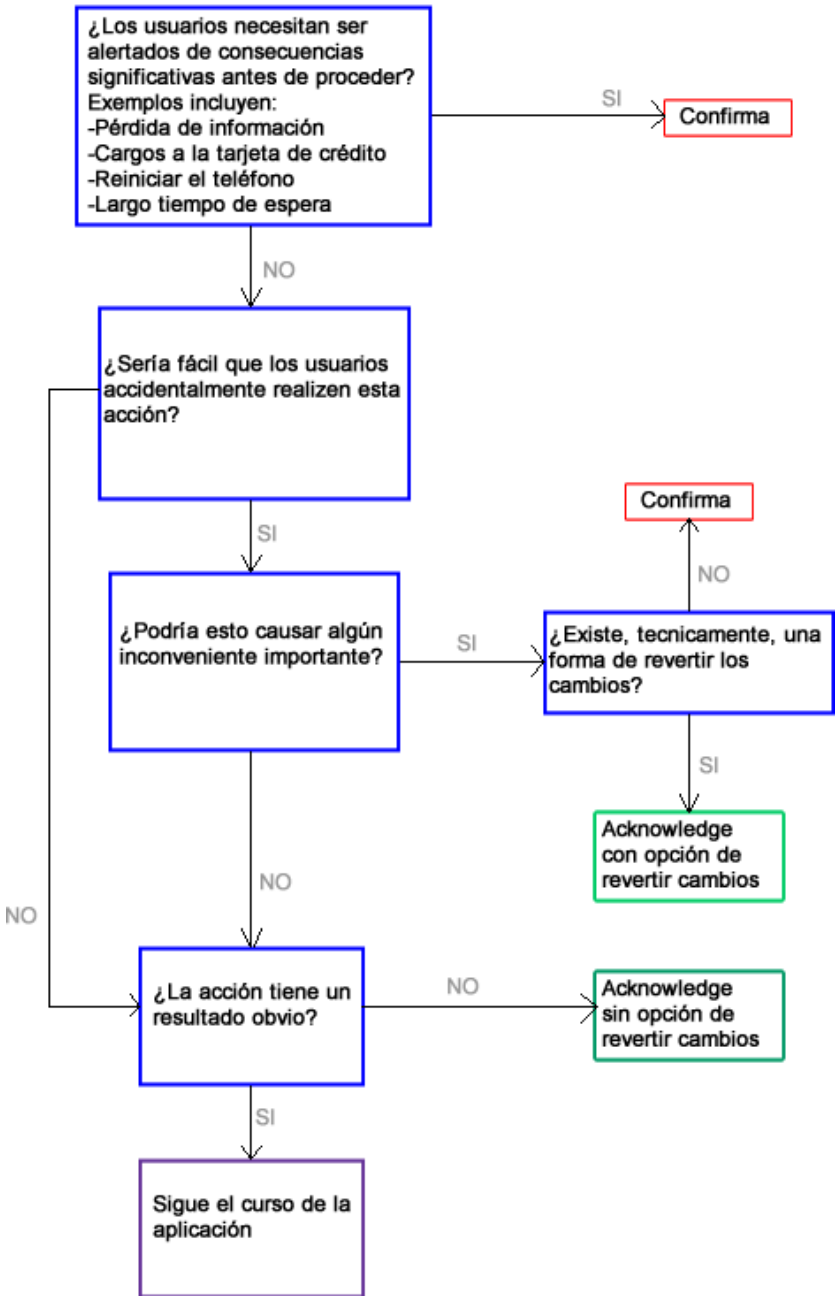

Figura 3.7. Diagrama de flujo que presenta el uso recomendado para confirmación o *acknowledge.*

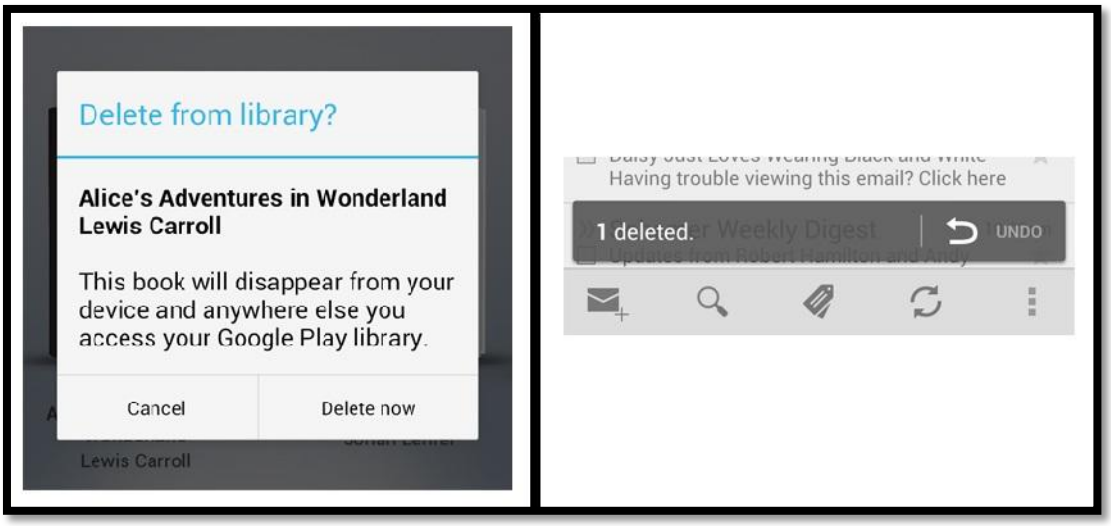

Figura 3.8. Del lado izquierdo un ejemplo de confirmación. Del lado derecho un ejemplo de *acknowledge*. En el primer ejemplo es necesaria una confirmación ya que es importante que el usuario comprenda que al eliminar el libro de la biblioteca éste no volverá a estar disponible. En el segundo ejemplo la aplicación informa que se ha eliminado un mensaje y aparece la opción de revertir esta acción.

3.3 Estructura de una aplicación

Las aplicaciones móviles satisfacen diversas necesidades de los usuarios, que van desde las más sencillas hasta las más técnicamente complejas. Por ejemplo, aplicaciones como la calculadora están enfocadas a una sola tarea que puede ejecutarse en una única pantalla. A diferencia, la *app* PlayStore combina todo un set de pantallas con una navegación profunda. La estructura de su aplicación depende ampliamente del contenido y las tareas que quiere facilitar al usuario.

Una típica aplicación Android consiste en un nivel superior (*Top Level*) y vistas detalladas o de edición (*detail/edit views*). Si la navegación es profunda y compleja, las vistas de categorías conectan el nivel superior y las vistas de detalles. Las vistas de nivel superior pueden mostrar diferentes representaciones de la misma información o exponer distintas funcionalidades de la *app.* Las vistas de categorías permiten adentrarse a las demás vistas de la aplicación, mientras que las vistas de detalles son dónde se colecciona o se crea nueva información (figura 3.9).

El diseño de su vista de nivel superior principal requiere especial atención, es la primera pantalla que las personas verán después de iniciar su aplicación. Pregúntese qué es lo que más planearían hacer sus usuarios con su aplicación, a partir de ello estructure sus contenidos. Es recomendable evitar pantallas sólo con listas de opciones e información, diseñe vistas gráficamente atractivas y apropiadas para el tipo de información que brindará su aplicación.

Use barras de acción, barras de íconos que representan las acciones más importantes que las personas pueden realizar dentro de la *app*. Puede usarlas para desplegar el ícono distintivo o el título de su aplicación. Si su nivel superior consiste de múltiples vistas, asegúrese de facilitar al usuario la navegación. Para una navegación consistente, todas

sus vistas deberían contener barras de acción que permitan recorrer las acciones más importantes que puede realizar el usuario.

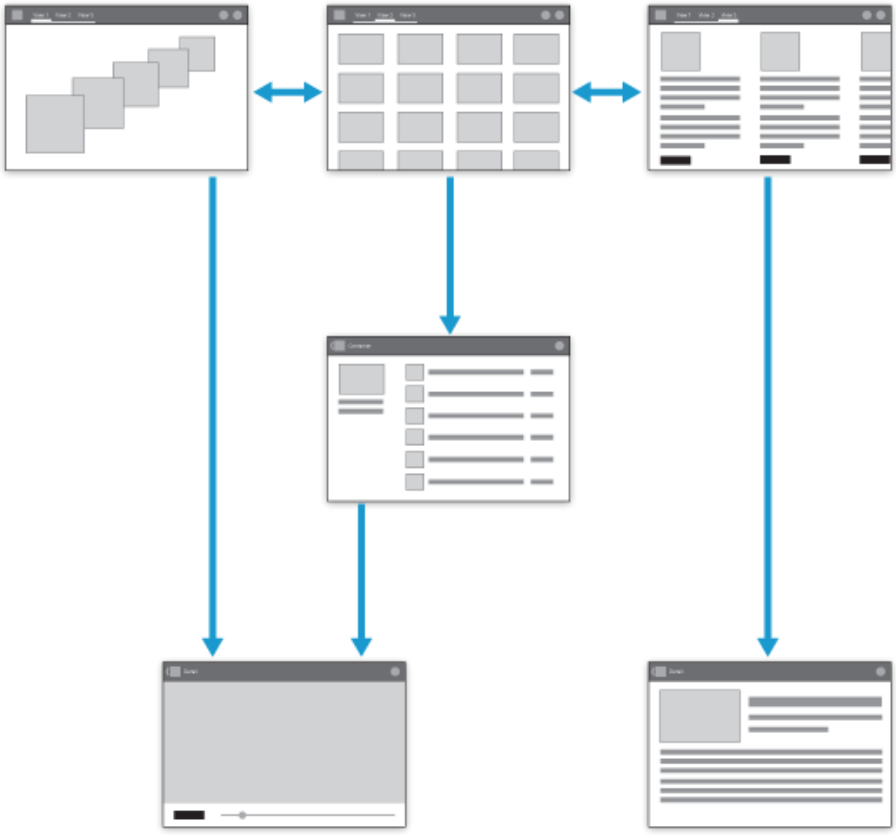

Figura 3.9. Ejemplo de estructura de una aplicación Android. Existen tres vistas de nivel superior, se puede navegar entre ellas a través de pestañas fijas. Cada una de estas vistas lleva a una vista de detalles y permite regresar a las de nivel superior. El icono de la aplicación aparece siempre en la parte superior izquierda de la app.

El nivel superior guía al usuario a través de las principales áreas funcionales de la aplicación. En la mayoría de los casos el nivel superior consistirá de múltiples vistas y es valiosa una navegación eficiente entre ellas. Android maneja controles de vistas para esta tarea. Use el control que más se acople a las necesidades de navegación de su *app.*

Pestañas fijas. Despliegan las vistas de nivel superior concurrentemente y facilitan el intercambio entre ellas. Están siempre visibles en la pantalla, permiten al usuario navegar entre las vistas deslizando hacia la izquierda o la derecha. Su uso es recomendable si se espera que el usuario intercambie las vistas frecuentemente. Soportan un límite de tres vistas de nivel superior. Su uso más frecuente es para crear vistas de categorías. Por ejemplo, las pestañas de categorías en la *app* PlayStore permiten navegar a través del gran número de aplicaciones.

Spiners. Un *spiner* es un menú desplegable hacia abajo. Es recomendable usarlo si el usuario está intercambiando vistas con el mismo tipo de información. Por ejemplo, eventos del calendario agrupados por día, semana o mes.

Paneles de navegación (*Navigation Drawers*). Son menús laterales que pueden desplegar un gran número de elementos. Son accesibles desde cualquier vista de la aplicación. Muestran tanto las vistas de nivel superior, como las de niveles inferiores. Se recomienda su uso cuando existe un gran número de vistas de nivel superior. También para brindar una navegación rápida entre vistas que no tienen relaciones entre sí.

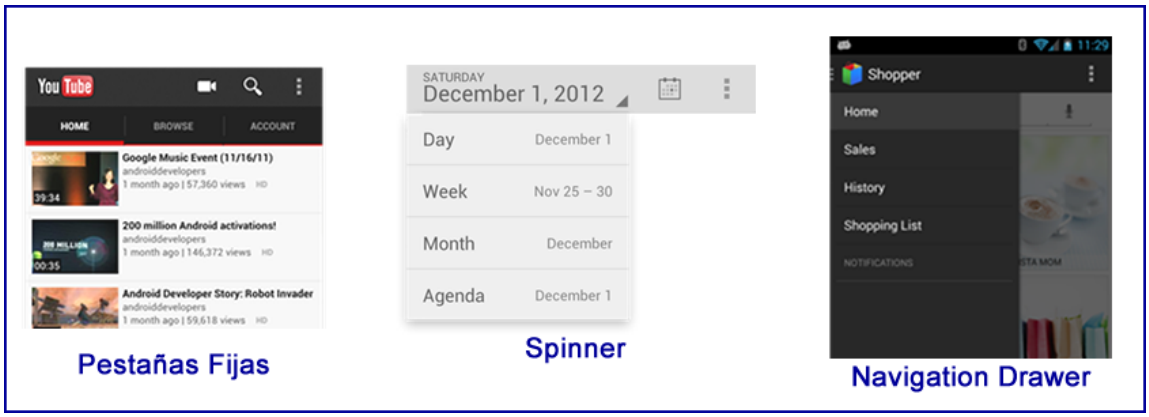

Figura 3.10. Controles de vistas mostrando un ejemplo de su uso.

Finalmente las vistas de detalles permiten observar la información y/o actuar sobre ella. El diseño de esta vista depende del tipo de información que será desplegada y consecuentemente diferirá ampliamente en cada aplicación. Considere las actividades que los usuarios realizarán con el contenido de esta vista y diséñela tomándolas en cuenta. Por ejemplo, en la lista de contactos cada persona tiene asignada una vista donde se despliega su nombre, una fotografía, sus números telefónicos, dirección de e-mail, etc. además de las acciones que se pueden realizar como llamarlo, enviarle un SMS y enviarle un correo electrónico.

Capítulo 4
Aplicación propuesta

Retomando lo que se ha expuesto en capítulos anteriores de esta tesis. Se propone una aplicación a desarrollar, dirigida a dispositivos Android, con un modelo de negocios basado en publicidad. Disponible para su descarga gratuita en la tienda Google Play Store. Se buscará que la aplicación respete las buenas prácticas sugeridas por Android, con el objetivo de cautivar, motivar y simplificar la vida del usuario final. La aplicación se ofrecerá de forma gratuita a través de la tienda de aplicaciones de Android pero con un modelo de negocios basado en publicidad, por lo que será necesario implementar la red publicitaria AdMob dentro de esta *app*. Sabemos que las funciones de AdMob, sólo están disponibles cuando existe una conexión a internet disponible en el dispositivo móvil, así que es esencial que la *app* interactúe con ciertos contenidos en internet. Además, para darle un enfoque más profesional a la *app* se utilizarán algunos temas importantes de la carrera de Ingeniería en Computación, tales como: programación orientada a objetos, arquitecturas cliente/servidor, Ingeniería de Software, bases de datos y cómputo móvil.

Propuesta de aplicación: Desarrollar una aplicación móvil que permita consultar el número de visitas de una determinada página web. El usuario deberá poder registrar las páginas que desee para consultar, por medio de una conexión a internet, las visitas que éstas han recibido. El usuario podrá consultar la cantidad de visitas totales y también las visitas en el último mes y en la última semana.

Notas:
-La aplicación será desarrollada para que cualquier usuario pueda consultar las visitas de un sitio web que él administre, desde la comodidad de su dispositivo móvil.
-Las páginas web que se pueden registrar serán aquellas cuyos usuarios pueden editar y modificar totalmente, es decir, su página personal, la página de su empresa, la página de su negocio, etc. De manera que el usuario sea capaz de añadir cierto fragmento de código a las páginas que haya registrado.
-La cuenta de las visitas se iniciará desde que el usuario añada el código a su página web.

4.1. Arquitectura de la solución.

A partir de la información que nos brinda el enunciado de la propuesta podemos comenzar a diseñar nuestra solución. Conocemos hasta ahora a 3 actores que estarán interactuando entre sí: el usuario, un dispositivo móvil y una página web. Será necesario guardar información acerca de las visitas de cada una de las páginas que haya registrado el usuario y al mismo tiempo que se pueda consultar a través de internet, entonces es necesario un servidor de bases de datos. Para proteger la información de las consultas a la base de datos que se podrían realizar es necesario un intermediario. Este intermediario recibirá las peticiones de la página web y la aplicación móvil para realizar las consultas por medio del servidor de bases de datos, de esta manera se evita enviar información sensible de forma poco segura, tal como el nombre del servidor de bases de datos,

contraseña de usuario de la base y nombres de tablas o procedimientos. Con estos elementos, nuestra solución se puede ajustar a una *Arquitectura Cliente/Servidor en 3 capas.*

El modelo Cliente/Servidor es una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información de forma transparente independientemente de la plataforma de software y hardware empleada. Un servidor es todo proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso o servicio que éste ofrece. Algunas de las principales funciones que lleva a cabo el servidor son procesar requerimientos de bases de datos y formatear datos para transmitirlos a los clientes. El cliente, por lo tanto, es todo proceso que permite al usuario formular las peticiones que se harán al servidor. Las funciones que lleva a cabo este proceso son: administrar la interfaz de usuario, interactuar con el usuario y generar las peticiones al servidor [13]. El esquema de funcionamiento de un modelo Cliente/Servidor resulta de la siguiente manera.

1. El cliente solicita información al servidor
2. El servidor recibe la petición del cliente
3. El servidor procesa la solicitud y obtiene resultados
4. El servidor envía el resultado que se ha obtenido al cliente
5. El cliente recibe el resultado y lo procesa.

El modelo en 3 capas se caracteriza por que el proceso servidor tiene una interacción adicional con un servidor de base de datos. En este esquema el servidor, dependiendo del tipo de solicitud, accede y se conecta con la base de datos. En la figura 4.1 se pueden observar los elementos de esta arquitectura.

Figura 4.1. Arquitectura Cliente/Servidor en 3 capas. El cliente realiza la petición al servidor a través de la red, el servidor procesa la información, hace una consulta al servidor de base de datos y obtiene un resultado. Finalmente el servidor entrega una respuesta al cliente.

De acuerdo a esta propuesta de solución, los clientes serán las páginas web registradas y los usuarios de la aplicación móvil. Estos clientes solicitarán distintas peticiones al servidor, como incrementar el número de visitas, consultar visitas o registrar nuevas páginas. Para ello se necesitará un servidor web, que procese todas esas solicitudes. También se necesitará un servidor de base de datos para almacenar toda la información de los usuarios.

Para un programador independiente que está comenzando, podría ser difícil encontrar un servidor web y de base de datos donde alojar este servicio, afortunadamente existen opciones de *hosting* gratuito que permiten la conexión a una base de datos *MySql* y que están configurados para almacenar y ejecutar código en *PHP* y *JavaScript*. Al ofrecer sus servicios de forma gratuita existen ciertas desventajas como el reducido espacio de almacenamiento, la reducida cantidad de tráfico permitido por mes y la gran cantidad de tiempo en el que el servicio se encuentra suspendido. Aún con estas desventajas, un servicio de *hosting* gratuito ayudará a desarrollar la aplicación propuesta en esta tesis.

Figura 4.2. Implementación de la arquitectura Cliente/Servidor en 3 capas para la aplicación propuesta.

La figura 4.2 muestra un diagrama con la implementación del modelo Cliente/Servidor de 3 capas. Los clientes realizarán una petición al servidor web para incrementar el número de visitas de una página web. El servidor a su vez realizará una conexión con el servidor de base de datos para realizar las consultas necesarias. El usuario final de la aplicación móvil podrá consultar diversa información acerca de las visitas de sus páginas web registradas. En este caso el cliente es la aplicación móvil, que enviará una petición al servidor, quien realizará las consultas pertinentes a través del servidor de base de datos. A pesar de que la *app* en el dispositivo móvil también es un cliente, realiza peticiones distintas al servidor y este a su vez realiza distintos tipos de consultas (*Queries*) a la base de datos. Una característica de este modelo es que es independiente de hardware y software, suponga que en la imagen cada cliente tiene un distinto sistema operativo, además de que se trata de dispositivos distintos, lo que le da mayor flexibilidad a la solución.

4.2. Justificación

¿Por qué hacer una aplicación que debe conectarse a internet para funcionar? La respuesta es muy sencilla. Se está desarrollando una aplicación móvil que será distribuida de forma gratuita con *In-App Advertising* y para que la aplicación cargue los banners publicitarios el dispositivo debe estar conectado a internet. La arquitectura de la solución que se ha planteado previamente obliga al usuario final a mantener una conexión activa a internet, por lo que los banners podrían hacerse ineludibles.

Actualmente se pueden utilizar diversas herramientas para el desarrollo de aplicaciones móviles, existen incluso entornos de desarrollo destinados a personas sin algún conocimiento de programación o de ingeniería de software, tal es el caso del sitio *appsmakerstore,* donde se pueden desarrollar aplicaciones móviles multiplataforma sin algún conocimiento técnico [14]. La complejidad en el desarrollo dependerá de cada persona, sin embargo, esta tesis mostrará el proceso de desarrollo de una aplicación móvil donde intervengan los conocimientos adquiridos en la carrera de Ingeniería en Computación.

El ejemplo propuesto en esta tesis no sólo requiere de una *app* móvil como solución, sino que también una aplicación web alojada en un servidor. Tal servidor hace consultas a una base de datos, utilizando una arquitectura Cliente/Servidor en 3 capas. De esta manera habrá cuatro elementos de software interactuando entre sí: La aplicación móvil, el script en la página web, la aplicación web en el servidor y la base de datos. Todos estos elementos serán desarrollados como un único proyecto de software para dar solución a nuestro ejemplo propuesto.

Se espera que el procedimiento para dar solución al ejemplo de este capítulo demuestre la forma en que los conocimientos adquiridos en la carrera de Ingeniería en Computación son aplicados en conjunto para resolver problemas de la vida diaria. Para el usuario la información se podrá consultar en cuestión de segundos, sin que necesariamente conozca las tecnologías que hay detrás que le permiten obtener esa valiosa información.

Es preciso mencionar que en ningún momento se deja de lado el objetivo de esta tesis, de exponer el proceso de desarrollo de una aplicación móvil y su posterior publicación bajo un modelo de negocios basado en publicidad. La aplicación propuesta y su solución son sólo un ejemplo de aplicación que puede ser llevada a la tienda de *apps*. Cualquier persona que utilice esta tesis como guía para desarrollar una aplicación y convertirse en emprendedor llevándola a la Play Store, debe sentirse libre de publicar cualquier aplicación que sea de su agrado.

En el siguiente capítulo se creará la solución basándose metodologías de Ingeniería de Software, por lo que vale la pena revisarlo. Su solución y su aplicación móvil pueden ser desarrolladas por estas metodologías si usted lo desea, es recomendable hacerlo ya que tienen por objetivo crear un software confiable, eficiente, competente y de calidad.

Capítulo 5
Desarrollo de la solución

Para que un producto de software sea desarrollado se requiere un proceso, tal proceso es llamado por la Ingeniería del Software como *proceso del software*. Existen muchos procesos diferentes de software, pero algunas actividades son fundamentales para todos ellos:

- Especificación del software. Aquí es donde se define la funcionalidad del producto, los requerimientos y las posibles restricciones referentes a su operación.
- Diseño e implementación. Se realizan modelos, o diagramas que faciliten el desarrollo de un software que cumpla con su especificación.
- Validación del software. Es importante que el cliente evalúe si el producto que se desarrolla cumple con sus expectativas.
- Evolución del software. Las necesidades del cliente son siempre cambiantes, por lo que el software deberá también cambiar. Existirán nuevas características que deberá tener el producto y algunas otras que deberán eliminarse.

Un modelo de proceso de software es una representación abstracta de este proceso. Cada modelo de proceso implica una perspectiva particular de los procesos y enfoques para el desarrollo del software.

El modelo en cascada (figura 5.1) fue el primer modelo de proceso de desarrollo de software. Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo y se enumeran a continuación [15]:

1. *Análisis y definición de requerimientos.* La funcionalidad del sistema, así como los servicios que proporciona y sus restricciones se definen a partir de las consultas con los usuarios. Se crea una especificación del sistema.
2. *Diseño del sistema y del software.* Identifica y describe las abstracciones fundamentales del producto y sus relaciones. Se crea una arquitectura completa del sistema. El objetivo es diseñar la solución de cada uno de los requerimientos del usuario y su interacción dentro del sistema.
3. *Implementación y prueba de unidades.* El diseño del software se lleva a cabo como unidades. Las pruebas de unidad implican verificar que cada unidad cumpla con su especificación.
4. *Integración y prueba del sistema.* Durante esta etapa todas las unidades se integran y se prueban como un sistema integral que cumpla con los requerimientos.
5. *Funcionamiento y mantenimiento.* El sistema se instala y es puesto en marcha. Su posterior mantenimiento implica corregir errores que no habían sido descubiertos en la fase de pruebas y añadir nuevas características al sistema cuando nuevos requerimientos sean descubiertos por el cliente.

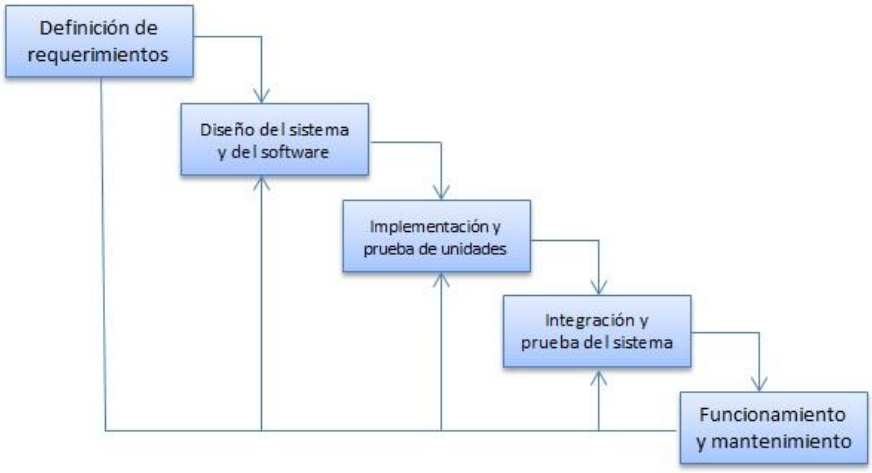

Figura 5.1. Modelo en cascada.

Cada una de las fases debe empezar cuando la fase previa haya finalizado y no antes, aunque en la práctica, estas etapas se superponen. Durante el diseño suelen identificarse problemas con los requerimientos, en la implementación se encuentran problemas de diseño, etc. “El proceso del software no es un modelo lineal simple, sino que implica una serie de iteraciones de las actividades de desarrollo”.

El desarrollo de la solución planteada en el capítulo anterior se llevará a cabo utilizando un modelo en cascada. Se recomienda utilizarlo cuando los requerimientos se han comprendido bien y resulte difícil que éstos cambien radicalmente durante el desarrollo del producto. Para nuestro caso, los requerimientos son fijos y están bien establecidos, el equipo de desarrollo es escaso y la solución a desarrollar no es robusta. Por lo que se opta por este modelo de desarrollo, que si bien es simple, terminará en una solución satisfactoria del ejemplo planteado.

5.1. Especificación de requerimientos

Los requerimientos de un sistema expresan las necesidades del cliente y del usuario final. El *Requerimiento* como tal, es una declaración suficientemente abstracta para establecer, a través de ella, los servicios proporcionados por el sistema o sus restricciones operativas. Para clasificar estos requerimientos, se toma como criterio los diferentes niveles de descripción, es decir, a la descripción detallada de lo que el sistema debe hacer se le conoce como *requerimientos del sistema* y a los requerimientos abstractos de alto nivel se les conoce como *requerimientos del usuario* [15].

También se suelen clasificar en funcionales y no funcionales:

- **Requerimientos funcionales**: Declaran la manera en que el sistema reacciona a entradas de datos y a situaciones específicas. Definen los servicios que proporciona el sistema. De igual forma, también pueden informar de lo que el sistema no debe hacer.
- **Requerimientos no funcionales:** Se refieren a las propiedades que deberá tener el sistema como fiabilidad, tiempo de respuesta, capacidad de almacenamiento y características de la interfaz de usuario. Éstos nacen de las necesidades de los usuarios, debido a diversas restricciones como

presupuesto, interoperabilidad con otros sistemas o políticas de la organización.

La función de los requerimientos del usuario es describir los requerimientos, tanto funcionales como no funcionales, de manera que los usuarios del sistema sin conocimiento técnico puedan comprenderlos. Deben ser redactados en un lenguaje sencillo, por tanto, deben evitar lo más posible utilizar jerga técnica y mencionar las características de diseño del sistema.

Los requerimientos del sistema agregan el detalle necesario para explicar cómo el sistema debe solucionar los requerimientos del usuario. No deben describir el diseño del sistema o su implementación, sólo se deben enfocar en el comportamiento externo del sistema y sus restricciones. Sin embargo, en la práctica, excluir la información de diseño resulta difícil.

A continuación se establecerá el documento de especificación de requerimientos del software, ***SRS*** [16]**,** en el que se declararán de manera formal las características de la solución al ejemplo propuesto en esta tesis. Los requerimientos han sido propuestos basándose en el ejemplo del capítulo 4 y no han sido obtenidos de alguna persona o empresa en particular. Se trata de los requerimientos que se desearía que tenga una aplicación móvil capaz de obtener el resultado del número de visitas a una página web.

/----------\

1. Introducción

Éste documento es una especificación de requerimientos de software para una aplicación móvil que permite consultar las visitas de una página web. Ésta especificación se ha estructurado inspirándose en las directrices dadas por el estándar *IEEE Recommended Practice for Software Requirement Specifications ANSI/IEEE 830 1998.*

1.1 Propósito.

El objeto de la especificación consiste en definir de manera precisa y clara las funcionalidades y restricciones del sistema que se desea desarrollar. El documento va dirigido a cualquier persona interesada en aprender la metodología de especificación de requerimientos.

1.2 Alcance

- Identificación del producto de software *SiteTracker*
- Objetivos del Sistema
    - Creación y eliminación de cuentas de usuario
    - Registro de un nuevo código de rastreo
    - Consulta de visitas asociadas a un código de rastreo

1.4 Definiciones y Acrónimos

- Definiciones
  - Android: Sistema operativo basado en Linux, diseñado para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tabletas.
  - Cadena de conexión: Una cadena de conexión contiene información de inicialización que se transfiere como un parámetro desde un proveedor de datos a un origen de datos.
  - Código de rastreo: Fragmento de código que debe insertarse en la página web de la que se requiera capturar información y estadísticas referentes a sus visitas.
  - Cuenta de usuario: Información que permite al usuario acceder al sistema y registrar nuevos códigos de rastreo. Es personal y le permite tener un control sobre sus códigos registrados.
  - Gesto: Movimiento especifico de uno o varios dedos, detectado por una pantalla táctil.
  - Hosting: servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía web.
  - HTML: Hypertext Transfer Protocol. es el protocolo usado en cada transacción web.
  - In-App Advertising: Publicidad instalada dentro de una aplicación móvil.
  - Javascript: Lenguaje de programación interpretado por un navegador web, que permite mejoras de interfaz de usuario y páginas web dinámicas.
  - jQuery: Biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML
  - Motor de base de datos: El Motor de base de datos es el servicio principal para almacenar, procesar y proteger los datos.
  - PHP: Hypertext Pre-processor. lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico.
  - Vista: Es una presentación visual de los elementos de una aplicación móvil que se encuentran en pantalla en un momento dado.

- Acrónimos
  - RFXX: Estándar seguido para la especificación del identificador de cada requerimiento, en este caso, funcional. Donde XX es una secuencia de dos dígitos utilizada para la enumeración de cada requerimiento.
  - RFNXX: Estándar seguido para la especificación del identificador de cada requerimiento no funcional. Donde XX es una secuencia de dos dígitos utilizada para la enumeración de cada requerimiento.

1.5 Referencias.

IEEE Recommended Practices for Software Requirements Specification ANSI/IEEE 830 1998.

1.6 Visión general del documento

Éste documento está compuesto por tres secciones:

- Introducción: Se detallan los objetivos de este documento y del sistema
- Descripción General: detalla una perspectiva general del producto a desarrollarse, así como las características y limitaciones que el usuario podría tener.
- Requerimientos específicos: Esclarece todos los requerimientos que el usuario desea tenga el producto final.

2. Descripción General

En esta sección se presenta una descripción de alto nivel del sistema. Se expondrán las funciones que el sistema debe realizar, la información utilizada, las restricciones y algunos otros factores que afecten el desarrollo de este sistema.

2.1 Perspectiva del producto.

El sistema a desarrollar no interactuará con algún otro sistema informático. Pero tendrá un diseño en módulos, donde cada uno reside en distintas ubicaciones dentro de internet.

2.2 Funcionalidad del producto

En términos generales, el sistema *SiteTraker* deberá ejecutar cada una de las siguientes tareas, con el fin de permitir al usuario consultar, a través de internet, información referente a las visitas que ha recibido cada una de las páginas web registradas, para las que se ha añadido un código de rastreo:

- Registro y baja de usuarios en el sistema
- Acceso a los usuarios mediante nombre y contraseña
- Listado de todos los códigos registrados por el usuario
- Registro y baja de un código de rastreo
- Consulta de visitas asociadas a un código de rastreo
- Despliegue de anuncios de la red de publicidad móvil AdMob

2.3 Características de los usuarios

Los usuarios de este sistema no tienen una formación profesional en específico, pero cuentan con habilidades de manejo de una computadora, conocimientos básicos sobre programación de páginas web en HTML y JavaScript además, cuentan conocimientos básicos sobre manejo de plataforma móvil Android. Las actividades que realizarán serán: la inserción del código de rastreo en la página web, instalación de la aplicación móvil en sus dispositivos, creación de una cuenta de usuario y consulta de información.
Los usuarios deberán contar con un dispositivo Android versión 2.2.1 (froyo) o superior y una cuenta de GooglePlay Store, desde donde descargarán e instalarán la aplicación.

2.4 Restricciones

Los módulos pertenecientes a la aplicación móvil serán desarrollados para la plataforma Android. Se cuenta con el servicio de *hosting* gratuito *000webhost* ($0.00 webhost), el cual, cuenta con las tecnologías PHP y MySQL. Por tanto, cualquier comunicación entre

el servidor y la aplicación móvil se desarrollará en PHP, asimismo, cualquier consulta a una base de datos se realizará mediante el motor de bases de datos MySQL.

3. Requerimientos

En este apartado se presentan los requerimientos funcionales que deben ser satisfechos por el sistema. Todos los requerimientos aquí expuestos son esenciales, es decir, no sería aceptable un sistema que no satisfaga alguno de los requerimientos aquí expuestos.

3.1 Requerimientos comunes de las interfaces

3.1.1 Interfaces de usuario.

Las interfaces de usuario están relacionadas con las pantallas, vistas, formularios que debe manipular el usuario para realizar una operación determinada. Ésta manipulación será realizada mediante la pantalla táctil del móvil y mediante el método de entrada característico de cada dispositivo. Las interfaces de usuario serán las características del sistema operativo móvil Android, y deberán respetar las recomendaciones de desarrollo de la misma plataforma.

3.1.2 interfaces de hardware

Pantalla del dispositivo.- el software deberá mostrar información al usuario a través de la pantalla del dispositivo. A la vez, será utilizada para navegar a través de la aplicación utilizando los distintos gestos de Android. Dependiendo del dispositivo, esta puede ser también utilizada como método de introducción de texto.

3.1.4 Interfaces de comunicación

La interfaz de comunicación entre la aplicación móvil y el servidor web se realizará mediante peticiones Http. La comunicación entre el servidor web y la base de datos utilizará la extensión mysqli de PHP. La interacción entre la página web con el código de rastreo y el servidor web se realizará mediante una petición Http programada en jQuery.

3.2 Requerimientos funcionales

3.2.1 Requerimiento funcional 1

RF01: Comprobar conexión a internet.

Introducción: Al iniciar la aplicación móvil debe comprobar si existe una conexión a internet disponible en el dispositivo.

Interacción con la base de datos: NO.

Entradas: Ninguna.

Procesos
La aplicación móvil comprobará si existe una conexión a internet, después verificará que esa conexión esté disponible.

Salidas
Mensaje de error en caso de que no exista una conexión a internet.

Mensaje de error en caso de que la conexión a internet no esté disponible.
Proceder a la siguiente vista en caso de éxito.

3.2.2 Requerimiento funcional 2

RF02: Registro de usuarios.

Introducción: La aplicación registrará nuevos usuarios en el sistema.

Interacción con la base de datos: SI.

Entradas: Nombre, Nombre de usuario, contraseña, confirmación de contraseña, dirección de correo electrónico.

Procesos
El usuario llena el formulario y toca el botón registrar. La aplicación comprobará que la contraseña y la confirmación de contraseña coincidan. La aplicación comprobará que el Nombre de usuario esté disponible, es decir, que no exista un registro de este nombre dentro de la base de datos. La aplicación comprobará que el correo electrónico no haya sido usado previamente por otro usuario, en otras palabras, que no exista un registro de este correo dentro de la base de datos. Si todas estas condiciones se cumplen se procederá a escribir un nuevo registro en la base de datos con la información solicitada.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Mensaje de error en caso de que la contraseña no coincida con la confirmación de contraseña.
Mensaje de error en caso de que el nombre de usuario no esté disponible.
Mensaje de error en caso de que la dirección de correo electrónico ya se encuentre registrada.
Mensaje de error en caso de que no pueda escribirse el nuevo registro.
Proceder a la siguiente vista en caso de éxito.

3.2.3 Requerimiento funcional 3

RF03: Acceso al sistema.

Introducción: El usuario ingresará su nombre de usuario y contraseña para acceder al sistema.

Interacción con la base de datos: SI

Entradas: nombre de usuario, contraseña.

Procesos: El usuario llena el formulario y toca el botón acceder. La aplicación móvil verifica que el nombre de usuario exista dentro de la base de datos. La aplicación verifica que la contraseña ingresada coincida con la contraseña asociada al nombre de usuario. La aplicación obtiene el identificador de nombre de usuario y lo almacena.

Salidas
Mensaje de error en caso de que falle la conexión con la base de datos.
Mensaje de error en caso de que el nombre de usuario no exista dentro de la base de datos.
Mensaje de error en caso de que la contraseña ingresada no coincida con la contraseña asociada al nombre de usuario.
Proceder a la siguiente vista en caso de éxito.

3.2.4 Requerimiento funcional 4

RF04: Recuperación de contraseña

Introducción: El usuario podrá recuperar su contraseña en caso de haberla olvidado, esta será enviada a la dirección de correo electrónico que proporcionó al momento de su registro.

Interacción con la base de datos: SI

Entradas: Nombre de usuario

Procesos
El usuario llena el formulario y toca el botón recuperar. La aplicación comprobará que el nombre de usuario esté registrado en la base de datos. La aplicación obtendrá la dirección de correo electrónico y contraseña asociada a ese nombre de usuario. La aplicación móvil se conecta con el servidor web y le envía la dirección de correo electrónico junto con la contraseña. El servidor web envía la contraseña por correo electrónico a la dirección proporcionada. El servidor web regresa un mensaje de éxito o fallo según sea el caso.

Salidas
Mensaje de error en caso de que falle la conexión con la base de datos.
Mensaje de error en caso de que el nombre de usuario no esté registrado en la base de datos.
Mensaje de error en caso de que falle la conexión con el servidor web.
Mensaje de éxito o fallo devuelto por el servidor web.
Proceder a la siguiente vista en caso de éxito.

3.2.5 Requerimiento funcional 5

RF05: Listado de códigos de rastreo

Introducción: La aplicación móvil desplegará un listado de todos los códigos de rastreo registrados por el usuario. Al tocar un elemento de la lista podrá acceder a una vista detallada del código.

Interacción con la base de datos: SI

Entradas: Identificador de nombre de usuario.

Procesos
La aplicación móvil obtendrá un listado de todos los códigos de rastreo registrados en la base de datos, asociados con el identificador de nombre de usuario. La aplicación móvil desplegará el listado de los códigos de rastreo obtenidos. Al tocar un elemento de la lista, la aplicación móvil procederá a otra vista.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Despliegue del listado de los códigos de rastreo.
Proceder a la vista correspondiente en caso de que el usuario toque un elemento del listado.

3.2.6 Requerimiento funcional 6

RF06: Registro de nuevo código de rastreo.

Introducción: La aplicación móvil permitirá al usuario registrar un nuevo código de rastreo para alguna de sus páginas web. Al registrar un nuevo código bajo un nombre de página, se obtendrá también el fragmento de código a insertar en el código de la página web.

Interacción con la base de datos: SI

Entradas: nombre de página, identificador de nombre de usuario.

Procesos
El usuario llena el formulario y toca el botón registrar. La aplicación móvil verifica que no exista previamente el nombre de página dentro de la base de datos para ese identificador de usuario. La aplicación móvil inserta un nuevo registro en la base de datos con el identificador de nombre de usuario y nombre de página. En caso de que lo anterior se cumpla la aplicación muestra el fragmento de código que debe insertarse en el código fuente de la página web.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Mensaje de error en caso de que el nombre de la página ingresado por el usuario coincida con uno registrado previamente por ese mismo usuario.
Mensaje de error en caso de que no se pueda añadir el nuevo registro.
En caso de éxito la aplicación procederá a una vista con el código que deberá añadirse al código de la página web de la que se desee empezar a contar visitas.

3.2.7 Requerimiento funcional 7

RF07: Consulta de visitas.

Introducción: El usuario podrá consultar algunos detalles de su código de rastreo tales como: fecha de creación, número de visitas totales, número de visitas en la semana en curso y en el mes en curso. El usuario tendrá la opción de refrescar la información.

Interacción con la base de datos: SI

Entradas: código de rastreo

Procesos
La aplicación obtiene la fecha de creación del código de rastreo mediante una consulta a la base de datos y la despliega en pantalla. La aplicación calcula el número total de visitas, número de visitas de la semana en curso y número de visitas del mes en curso, asociadas al código de rastreo, mediante consultas a la base de datos y lo despliega en pantalla. El usuario toca el botón refrescar y se repite todo el proceso.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Fecha de creación del código de rastreo.
Número de visitas totales.
Número de visitas en la última semana.
Número de visitas en el último mes.

3.2.8 Requerimiento funcional 8

RF08: Despliegue de anuncios publicitarios

Introducción: La aplicación desplegará los anuncios de la red de publicidad móvil AdMob en las vista de despliegue de todos los códigos de rastreo y en la vista detallada de cada código de rastreo.

Interacción con la base de datos: NO

Entradas: Ninguna

Procesos
La aplicación cargará los anuncios a través de la red de publicidad AdMob y los mostrará en pantalla.

Salidas
Banner publicitario con los anuncios de AdMob.

3.2.9 Requerimiento funcional 9

RF09: Baja de un código de rastreo

Introducción: El código de rastreo será eliminado del registro de códigos y todas las visitas asociadas a ese código serán también eliminadas.

Interacción con la base de datos: SI

Entradas: código de rastreo

Procesos
El usuario tocará la opción dar de baja este código. El usuario confirmará la opción. La aplicación eliminará el registro de la base de datos perteneciente al

código de rastreo. La aplicación eliminará mediante una consulta a la base de datos todos los registros de las visitas pertenecientes a ese código de rastreo.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Mensaje de éxito en caso de completar correctamente la acción de eliminar registros.

3.2.10 Requerimiento funcional 10

RF10: Baja de un usuario

Introducción: El usuario podrá darse de baja del sistema y toda la información relacionada a su identificador de nombre de usuario será eliminada.

Interacción con la base de datos: SI

Entradas: identificador de nombre de usuario.

Procesos
El usuario tocará la opción de eliminar cuenta. El usuario confirmará la eliminación de su cuenta. La aplicación realizará una consulta a la base de datos para obtener todos los códigos de rastreo asociados a ese identificador de nombre de usuario. La aplicación eliminará todos los registros de las visitas asociadas a cada uno de los códigos de rastreo obtenidos. La aplicación eliminará todos los registros de la base de datos pertenecientes a los códigos de rastreo obtenidos. La aplicación eliminará el registro de la base de datos perteneciente al identificador de nombre de usuario. La aplicación eliminará el valor almacenado del identificador de nombre de usuario. La aplicación regresa a la vista inicial.

Salidas
Mensaje de error en caso de que falle la conexión a la base de datos.
Mensaje de éxito en caso de completar correctamente la acción de eliminar la información del usuario.
En caso de éxito la aplicación procede a la vista inicial.

3.3 Requerimientos no funcionales

3.3.2. Seguridad

Cuando se realiza una consulta a una base de datos, es substancial utilizar una cadena de conexión, que incluye la dirección IP del servidor de base de datos, usuario y contraseña. Para evitar que esta información sea obtenida por cualquier usuario de la aplicación será necesario que toda conexión a la base de datos se realice mediante un intermediario, en este caso el servidor web. El servidor web será el encargado de recibir información acerca de las consultas necesarias y al mismo tiempo entregará los resultados solicitados por la aplicación. El servidor web poseerá la información de la cadena de conexión y realizará toda consulta a la base de datos.

3.3.3 Fiabilidad

El servicio de almacenamiento y de servidor web pertenece a 000webhost. Industria líder en servicios de web hosting. Lo que garantiza una mayor seguridad y disponibilidad de la información.

3.3.4 Disponibilidad

La disponibilidad depende enteramente del servicio 000webhost. Para usuarios con cuenta gratuita el servicio se encuentra disponible en un 99% del tiempo.

3.3.5 Mantenibilidad

El servicio gratuito de 000webhost permite completo control sobre la información y los archivos que residen en el servidor web. Además admite respaldos de información y respaldos automatizados. Se podrá dar mantenimiento a la base de datos y al servicio web. De igual forma, la tienda Google Play Store permite actualizaciones de aplicación. En cuanto surjan nuevas versiones del sistema podrán ser proporcionadas al usuario.

3.3.6 Portabilidad

Cualquier versión de la aplicación en cualquier dispositivo será capaz de mostrar la información de códigos de rastreo de todo usuario. Bastará con que inicie una nueva sesión con su cuenta de usuario dentro de la aplicación.

3.4 Otros requisitos

La aplicación móvil será distribuida de manera gratuita a través de la tienda Google Play Store, bajo la modalidad de *In-App Advertising.* Por lo que debe instalarse la red de publicidad móvil AdMob.
/----------\

5.2. Diseño del software.

Cuando se han establecido y analizado los requerimientos del software, el diseño del software es la primera de las tres actividades técnicas que se requieren para la construcción del producto de software, posteriormente se generará el código para después realizar pruebas al sistema. La tarea de diseño del software produce un diseño de datos, un diseño arquitectónico, un diseño de interfaz y un diseño de componentes [17].
El *diseño de datos* se apoya de un diagrama de entidad relación para transformarlo en las estructuras de datos que serán necesarias para implementar el software.
El *diseño arquitectónico* define la relación entre los elementos estructurales principales del software. En el capítulo anterior se ha definido este diseño.
El *diseño de la interfaz* define el flujo de la información y su comportamiento, describe la forma en que se comunicará el software con otros sistemas y con las personas que lo operan.
El *diseño a nivel de componentes* convierte aquellos elementos de la arquitectura del software en una descripción de procedimientos de los componentes del software.

"El diseño del software es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado".

Diseño de datos. Para realizar el diseño de datos se elaborará un diagrama entidad relación. Éste diagrama será útil para familiarizarse con la información que va a manipular el sistema, a su vez, ayudará a la creación de la base de datos. De la especificación de requerimientos tomamos la siguiente información:

Al momento de registro el USUARIO deberá proporcionar su nombre de usuario, una contraseña y una dirección de correo electrónico. Si el registro es exitoso, el usuario obtendrá un identificador de nombre de usuario. El nombre de usuario y la dirección de correo electrónico deben ser únicas y no pueden repetirse. Cada usuario puede registrar varios códigos de rastreo, los cuales contienen un alias de la PAGINA (nombre de la página), su fecha de creación y el código en sí, entonces cada página será asociada a un único código de rastreo. Cada una de las páginas registradas podrá recibir visitas y registrarlas en el sistema, de cada VISITA se desea saber el código de rastreo que la está generando y la fecha en que esta se realizó. Un código de rastreo únicamente puede pertenecer a un usuario así como una visita solamente puede pertenecer a una página.

El diagrama entidad relación que se puede obtener de esta observación se muestra en la figura 5.2. La relación entre usuario y página es de 1 a muchos, es decir que un usuario podrá tener muchas páginas registradas y que una página sólo pertenece a un usuario. La relación entre una página y una visita es similar, puede notarse que un usuario puede o no tener alguna página registrada, de igual forma, una página puede o no tener visitas. Las fechas se obtienen automáticamente del sistema al momento de insertar un nuevo registro. Los atributos subrayados indican que su valor será utilizado como identificador llave principal en la base de datos.

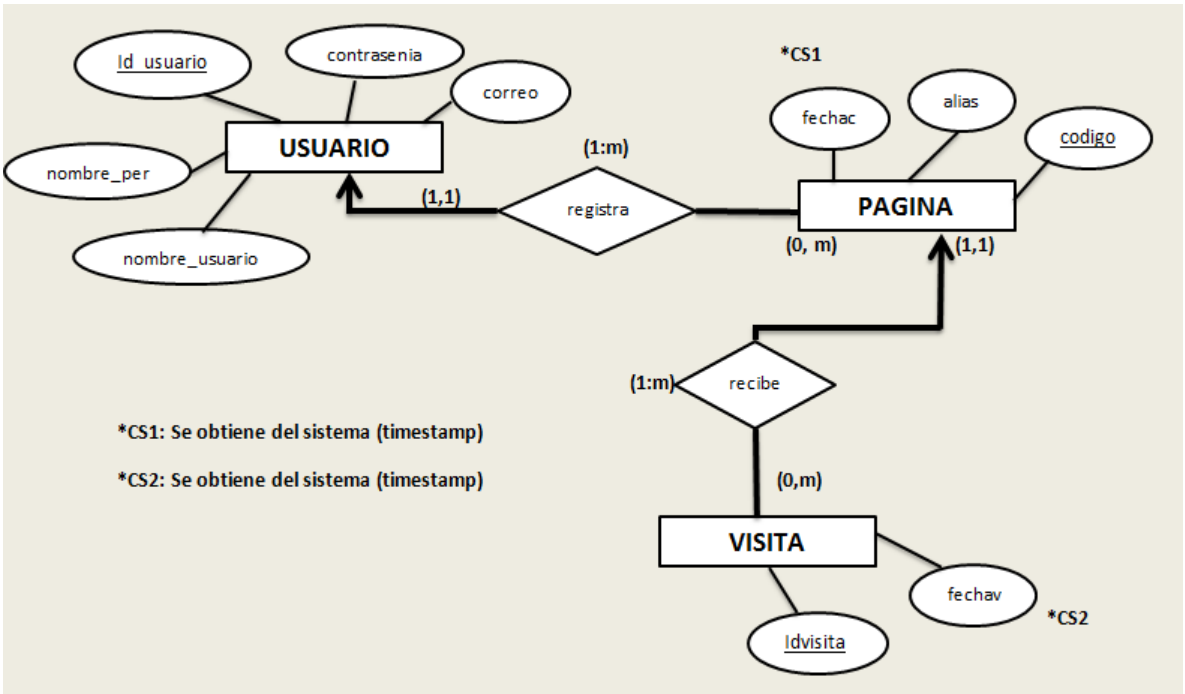

Figura 5.2. Diagrama entidad relación obtenido del documento de especificación de requerimientos.

Diseño arquitectónico. En el capítulo anterior se ha mostrado un diseño arquitectónico del sistema. Ahora que se conocen la especificación de requerimientos se puede replantear el diseño arquitectónico de la aplicación, a una forma más detallada. La figura 5.3 muestra este diseño de arquitectura de la solución con las tecnologías y servicios que se plantearon en el documento de requerimientos. Las páginas de internet con el fragmento de código añadido, programado en JavaScript usando JQuery, incrementarán las visitas mediante una petición el servidor web de modificar la base de datos. El servidor web 000webhost es el intermediario entre la aplicación móvil y el servidor de base de datos MySQL. Todas las consultas serán realizadas por el servidor web en código PHP. La aplicación móvil Android intercambiará información con el servidor web, enviará datos usando peticiones http y recibirá información codificada en JSON. Ésta explicación incluye varios nombres de tecnologías de programación que serán abarcadas más a detalle en partes posteriores de esta tesis.

Figura 5.3. Arquitectura de la solución. Versión detallada de la figura 4.2, después de analizar el documento de especificación de requerimientos

Diseño de interfaz. A continuación se mostrará un diagrama de estados basado en el diagrama de estados UML pero considerando que cada estado es una vista de la aplicación Android. Un estado identifica una condición o una situación en la vida de un objeto durante la cual satisface alguna condición, ejecuta alguna actividad o espera a que suceda algún evento. En este caso se considera que toda vista de Android es un objeto que, efectivamente esperará a que suceda alguna actividad mientras se muestra al usuario. Un estado se representa gráficamente por medio de un rectángulo con tres divisiones internas, estas divisiones alojan el nombre del estado, los atributos del objeto en ese estado y las acciones que realiza, respectivamente. En algunos diagramas se suelen omitir los dos apartados inferiores, pero para el diseño de este sistema serán necesarios. Los diagramas de estado tienen un punto de comienzo, que se dibuja mediante un circulo sólido relleno y uno o varios puntos de finalización, que se dibuja por medio de un circulo conteniendo otro más pequeño, relleno. Para pasar de un estado a otro se usa una transición, que es una relación entre dos estados, representado gráficamente como una línea continua dirigida desde el estado origen hasta el estado destino [18].

El diagrama de estados/vistas de la aplicación móvil Android (figura 5.4) ejecuta las actividades recopiladas del documento de especificación de requerimientos, en un orden intuitivo y funcional. Como se observa, no existen puntos de terminación, tampoco existen transiciones de regreso en cada vista o estado. La interfaz de Android permite que el usuario termine las aplicaciones en cualquier momento de su ejecución, también permite un regreso a la vista anterior. Algunos estados cuentan con transición de retorno, esto indica que ese retorno debe ser obligatorio. Por ejemplo cuando se da de baja un código de rastreo es obligatorio regresar a la lista de códigos de rastreo, no tiene razón de existir esa vista cuando el código ya ha sido eliminado.

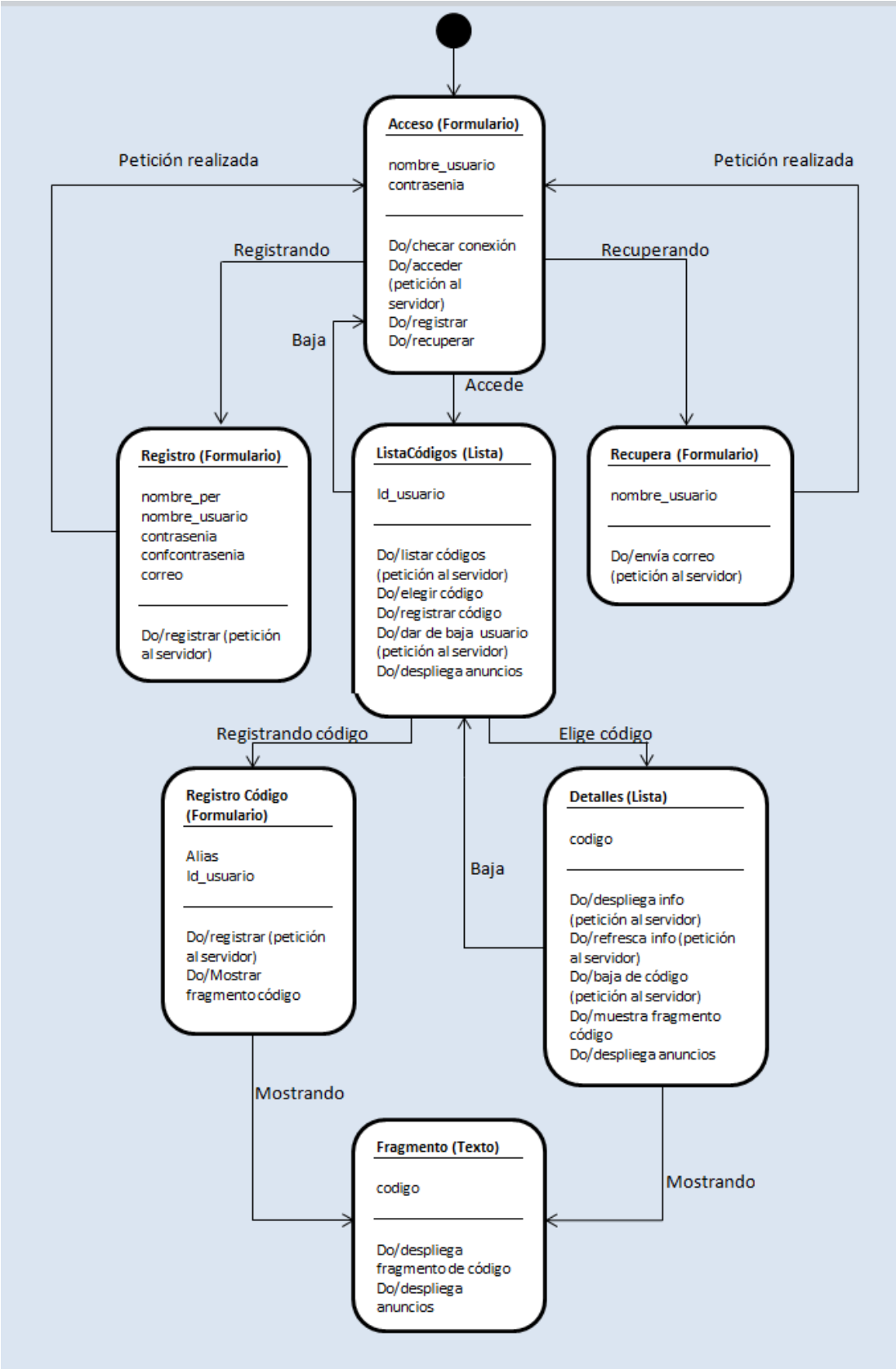

Figura 5.4. Diagrama de estados basado en vistas de la aplicación Android (SiteTracker.apk).

Diseño a nivel de componentes. Con los diagramas elaborados hasta ahora se tiene una vista más clara de cómo será implementado el sistema, en el diagrama de la figura 5.4 tenemos “peticiones al servidor”, en este apartado se utilizará el *diagrama de distribución* UML para describir la disposición física del sistema del lado del servidor y los componentes que lo forman.

Los diagramas de distribución muestran la distribución física de los distintos nodos que componen un sistema y el reparto de los componentes software sobre éstos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, con memoria y capacidad de procesamiento, es un dispositivo sobre el que se pueden desplegar los componentes. La relación entre un nodo y el componente que despliega puede mostrarse con una relación de dependencia, o listando los componentes desplegados en un compartimiento adicional dentro del nodo [19].

El diagrama de distribución correspondiente se muestra en la figura 5.5. Los nodos son: el dispositivo Android, un dispositivo que cuenta con navegador web, el servidor web y el servidor de base de datos. El dispositivo o los distintos dispositivos Android ejecutan el componente SiteTracker.apk (Diagrama de estados figura 5.4) y sus peticiones al servidor serán atendidas por los distintos componentes que ejecuta el servidor web. La petición que se realiza desde un dispositivo con un navegador web para incrementar el número de visitas se ejecuta con un componente de software que consiste en un fragmento de código JavaScript insertado en una página HTML. Ésta petición es atendida por el componente contador.php ejecutado por el nodo servidor web. Como lo indica el documento de requerimientos, y como ha sido mencionado antes, el servidor web atenderá todas estas peticiones siendo intermediario entre la aplicación móvil y la base de datos.

Los múltiples niveles de diseño que se han abarcado hasta ahora brindan una amplia ayuda para comenzar con la siguiente actividad técnica para la construcción del producto de software que se ha planteado. Hasta este momento conocemos la organización de los datos, las vistas de la aplicación y los componentes de software. También conocemos la arquitectura bajo la que estarán trabajando estos elementos.

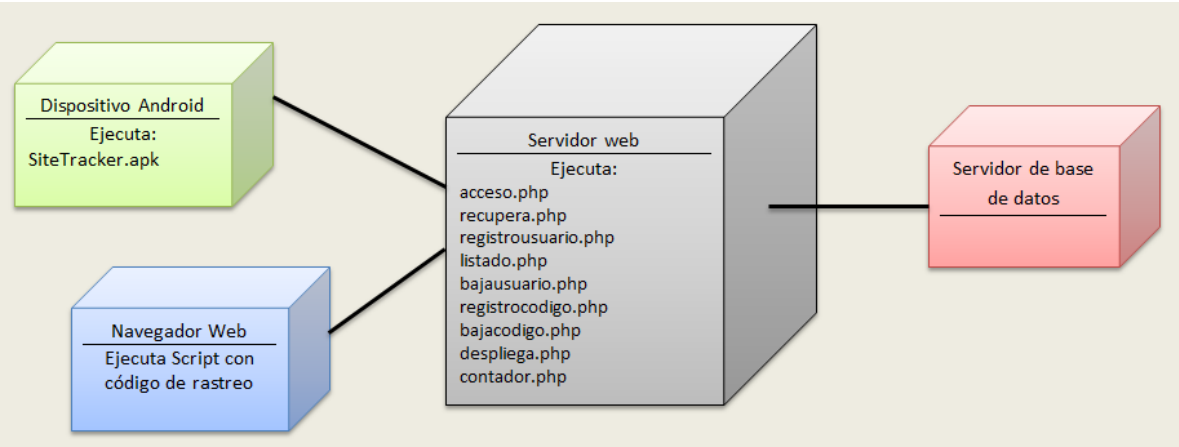

Figura 5.5. Diagrama de distribución. Diseño a nivel de componentes la solución propuesta.

5.3. Construcción o implementación del software.

La amplia visión que nos ha dado la especificación de requerimientos y el diseño sobre el sistema mismo permiten continuar con la implementación del software. En este punto del desarrollo podemos darnos cuenta lo importantes que han sido los pasos anteriores, la implementación del sistema está orientada a cumplir los requerimientos del cliente y a respetar una arquitectura definida. Esto se refleja en la construcción de un producto de software eficaz, que resuelva la problemática del cliente y que brinde una verdadera *solución.*

El servicio de *Hosting* gratuito de 000webhost proporciona una cuenta de acceso a un servidor de base de datos MySQL. Para la implementación de la Base de datos se utilizará el lenguaje de definición de datos (DDL). Éste lenguaje permite realizar las tareas de creación, eliminación y modificación de objetos de la base de datos, en este caso se utilizará para mudar el diagrama entidad relación de la figura 5.2 a las tablas que se van a necesitar.

El código se muestra a continuación:

//----------

```
CREATE TABLE usuario(
id_usuario int AUTO_INCREMENT PRIMARY KEY,
nombre_usuario VARCHAR(30) NOT NULL UNIQUE,
nombre_per VARCHAR(50) NOT NULL,
contrasenia VARCHAR(20) NOT NULL,
correo VARCHAR(40) NOT NULL UNIQUE);

CREATE TABLE pagina(
codigo int AUTO_INCREMENT PRIMARY KEY,
alias VARCHAR(30) NOT NULL,
fechac timestamp,
id_usuario int NOT NULL,
FOREIGN KEY(id_usuario) REFERENCES usuario(id_usuario));

CREATE TABLE visita(
idvisita int AUTO_INCREMENT PRIMARY KEY,
fechav timestamp,
codigo int NOT NULL,
FOREIGN KEY(codigo) REFERENCES pagina(codigo));
```

//----------

Cada entidad pasa a ser una tabla, con sus atributos correspondientes. Las relaciones entre una entidad y otra se reflejan en aquellos atributos FOREIGN KEY, la llave primaria (PRIMARY KEY) de una entidad fuerte pasa a ser llave foránea de la entidad débil. Los atributos con característica AUTO_INCREMENT tomarán un valor de forma automática desde 1 e irá incrementando. Los atributos con la característica UNIQUE, no podrán tener un valor repetido en la tabla. Finalmente, los atributos con característica timestamp

tomarán su valor automáticamente del sistema y será la hora en que el registro sea añadido o modificado.

El servidor web ejecutará componentes programados en PHP, que recibirán los datos de la aplicación móvil, realizarán una consulta a la base de datos y entregarán de vuelta el resultado. Las consultas que son realizadas por estos componentes están programadas con el lenguaje de manipulación de datos (DML), que permite acceder, crear, modificar o eliminar los datos de un esquema de base de datos. Las ordenes que se utilizan son insertar (INSERT), borrar (DELETE) y seleccionar (SELECT) los datos. Fragmentos de código de los componentes se muestran a continuación:

----------acceso.php----------

```
<?php
/* Recibiendo variables de entrada nombre de usuario y contraseña*/
$nick = $_POST['nombre_usuario'];
$pass = $_POST['contrasenia'];

/*Definiendo paquete de salida*/
$return = array();

/* Consulta, Si existe un registro con el nombre de usuario compara si la contraseña es correcta*/
if ($result = $mysqli->query("SELECT contrasenia,id_usuario FROM usuario WHERE nombre_usuario='$nick'")) {
        if($result->num_rows==0)
                $return["resultado"] = 2; /*resultado = 2 El usuario no existe*/
        else{
                $row = mysqli_fetch_array($result, MYSQLI_BOTH);
                $contrasenia = $row[0];
                $user = $row[1];
                if($contrasenia!=$pass)
                        $return["resultado"] = 3; /*resultado = 3 Contraseña incorrecta*/
                else{
                        $return["resultado"] = 0; /*Resultado = 0 acceso válido*/
                        $return["id_usuario"] = $user; /*Devuelve el identificador de usuario*/
                }
        }
$result->close();
 }
else
        $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/

$mysqli->close();
/*Devolviendo código de terminación*/
echo json_encode($return);
?>
```

/----------

acceso.php devuelve en notación *JSON* (función json_encode) el resultado de la petición y el identificador de usuario en caso de que el usuario y contraseña coincidan con un registro en la base de datos. Las consultas se realizan con la extensión para PHP,

MySQLi. Es necesario definir la cadena de conexión, previo a la consulta. Dicha cadena debe incluir el URL del servidor de base de datos, nombre de usuario de la base, contraseña y nombre de la base de datos [20].

----------recupera.php----------

```
/* Recibiendo variable de entrada nombre de usuario*/
$nick = $_POST['nombre_usuario'];

/* Consulta, Obtiene nombre personal, correo y contraseña del usuario dado, después le envía un correo electrónico con su contraseña*/
if ($result = $mysqli->query("SELECT nombre_per,correo,contrasenia FROM usuario WHERE nombre_usuario='$nick'")) {
        $row = mysqli_fetch_array($result, MYSQLI_BOTH);
        $nombre = $row[0];
        $correo = $row[1];
        $contrasenia = $row[2];
        /*Funcion que envía el correo electrónico*/
        $mensaje = "Estimado/a $nombre\r\nSu contraseña es: $contrasenia\r\n Gracias por utilizar SiteTracker";
        mail('$correo', 'Recuperación de contraseña SiteTracker', $mensaje);
        /*Codigo de terminación 0 para exitoso 1 para erroneo*/
        $return["resultado"] = 0;
        $result->close();
}
else
        $return["resultado"] = 1;
```

/----------

El Script recupera.php utiliza la función mail de PHP para enviar un correo electrónico con la recuperación de la contraseña de un usuario. Cabe recalcar que se presentan sólo fragmentos de código de los componentes en PHP, esencialmente las consultas a la base de datos.

----------registrousuario.php----------

```
/* Recibiendo variables de entrada nombre de la persona, nombre de usuario, contraseña y correo*/
$nombre = $_POST['nombre_per'];
$nick = $_POST['nombre_usuario'];
$pass = $_POST['contrasenia'];
$email = $_POST['correo'];

/* Consulta, los valores nombre_usuario y correo son UNIQUE, por lo que si existen previamente no se registrará el
usuario*/
if ($result = $mysqli->query("INSERT INTO usuario(nombre_per,nombre_usuario,contrasenia,correo)
VALUES('$nombre','$nick','$pass','$email');")) {
$return["resultado"] = 0; /*resultado = 0 Registro exitoso*/
 }
else
        $return["resultado"] = 2; /*resultado = 2 Nombre de usuario o correo existentes*/
```

/----------

----------listado.php----------

```
/* Recibiendo variable de entrada identificador de usuario*/
$user = $_POST['id_usuario'];

/*Creando una matriz de elementos página y una variable con el número de páginas registradas por el usuario*/
$return["paginas"] = array();

/* Consulta, Obtiene el codigo y alias de todas las páginas registradas por el usuario y las guarda en una matriz*/
if ($result = $mysqli->query("SELECT codigo,alias FROM pagina WHERE id_usuario='$user' ORDER BY codigo")) {
        $return["numero"]=$result->num_rows;
        while($row = mysqli_fetch_array($result, MYSQLI_BOTH)){
                $pagina= array();
                $pagina["codigo"] = $row[0];
                $pagina["alias"] = $row[1];
                array_push($return["paginas"], $pagina);
        }
        $return["resultado"] = 0; /*resultado = 0 Listado exitoso*/
        $result->close();
        }
else
        $return["resultado"] = 1; /*resultado = 2 error en la consulta a la base de datos*/
```

/----------

Listado.php devuelve una matriz en notación *JSON* que almacena varios elementos página y cada uno de ellos contiene un código de rastreo y un alias.

----------bajacodigo.php----------

```
/* Recibiendo variable de entrada codigo de rastreo de la página*/
$codigo = $_POST['codigo'];

/* Consulta, elimina todos los registros asociados a un $codigo en las tablas visita y pagina*/
if ($result = $mysqli->query("DELETE FROM visita WHERE codigo = '$codigo';")) {
        if($result2 = $mysqli->query("DELETE FROM pagina WHERE codigo = '$codigo';")) {
        $return["resultado"] = 0; /*resultado = 0 baja de codigo exitosa*/
        }
        else
        $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
 }
else
        $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
```

/----------

Al declarar una tabla se puede agregar una validación o *constraint* para eliminar todos los registros existentes en las tablas dependientes con llaves foráneas. Sin embargo, el servicio 000webhost de base de datos no permite declaración de este tipo de validaciones. Por este motivo, bajausuario.php realiza una acción similar a

bajacodigo.php. Con la diferencia que primero debe identificar todos los códigos registrados por un usuario, después eliminará todos ellos.

----------bajausuario.php----------

```
/* Recibiendo variable de entrada identificador de usuario*/
$usuario = $_POST['id_usuario'];

/* Consulta  Busca y elimina todas las visitas de todas las paginas registradas por dicho usuario, despues elimina al usuario de la base de datos*/
if($result = $mysqli->query("SELECT codigo FROM pagina WHERE id_usuario = '$usuario';")) {
  while($row = mysqli_fetch_array($result, MYSQLI_BOTH)){
            $codigo = $row[0];
            if ($result2 = $mysqli->query("DELETE FROM visita WHERE codigo = '$codigo';")) {
                        if($result3 = $mysqli->query("DELETE FROM pagina WHERE codigo = '$codigo';")) {
                                    $return["resultado"] = 0; /*resultado = 0 eliminación de visitas y páginas completa*/
                        }
                        else
                                    $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
            }
            else
                        $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
  }
  if($result4 = $mysqli->query("delete from usuario where id_usuario = '$usuario';")) {
            $return["resultado"] = 0; /*resultado = 0 baja de usuario exitosa*/
            }
  else
            $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
}
else
            $return["resultado"] = 2; /*resultado = 2 error en base de datos*/
```

/----------

Para registrar un nuevo código de rastreo se proporciona el alias (que no se debe repetir entre los ya registrados por un usuario) y el identificador de usuario. registrocodigo.php realiza 3 acciones: primero, debe revisar que el alias no se encuentre ya dentro de los registros de “pagina” asociados a ese identificador; si no es un valor repetido, inserta un nuevo registro en la tabla “pagina”; finalmente, realiza una consulta para obtener el código de rastreo que se generó automáticamente por su característica AUTO_INCREMENT.

----------registrocodigo.php----------

```
$usuario = $_POST['id_usuario'];
$alias = $_POST['alias'];

/* Consulta multiple si no existe el registro lo inserta, después de insertarlo obtiene su código de rastreo*/
if ($result = $mysqli->query("select * from pagina where id_usuario='$usuario' AND alias='$alias';")) {
            if($result->num_rows==0){
                        if($result2 = $mysqli->query("INSERT INTO pagina(id_usuario,alias) VALUES('$usuario','$alias');")) {
                                    if($result3 = $mysqli->query("SELECT codigo FROM pagina WHERE id_usuario='$usuario' AND alias='$alias';")) {
                                                $row = mysqli_fetch_array($result3, MYSQLI_BOTH);
```

```
                                        $return["resultado"] = 0; /*resultado = 0 registro exitoso*/
                                        $return["codigo"] = $row[0];
                                }
                                else
                                        $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
                        }
                        else
                                $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
                }
                else
                        $return["resultado"] = 2; /*resultado = 2 pagina con alias ya existente*/
        }
else
        $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
```

/----------

El script despliega.php es el que se encargará de desplegar la información referente a un código de rastreo. Realizará cuatro consultas, la primera de ellas será obtener la fecha de su registro, la cual se obtendrá de forma que se pueda presentar al usuario en el siguiente formato: dd month yy. Para que sea posible se utilizarán funciones propias de MySQL como MONTH() que extrae el nombre del mes de un atributo de tipo fecha.

Las tres consultas restantes contarán el número de visitas que ha tenido la página a la que pertenece el código de rastreo, primero las visitas totales, luego las visitas en el mes en curso y finalmente las vistitas en la semana en curso. Para que las últimas dos consultas obtengan la información requerida se acotarán las fechas con la función DATE_SUB(). La función COUNT() devuelve entonces el número de registros que cumplen la condición de la operación SELECT [21].

----------despliega.php----------

```
/* Recibiendo variable de entrada codigo de rastreo*/
$codigo= $_POST['codigo'];

/*Variables para definir el dia de la semana y el dia del mes en el momento de la consulta*/
$semana = date('N');
$mes = date('j');

/* Consulta multiple Obtendrá fecha de creación(dia,mes,año), visitas totales, visitas en el mes y visitas en la semana en curso*/
if ($result = $mysqli->query("SELECT DAY(fechac), MONTHNAME(fechac), YEAR(fechac) FROM pagina WHERE codigo
='$codigo';")) {
        if($result->num_rows>0){
                $row = mysqli_fetch_array($result, MYSQLI_BOTH);
                $return["dia"] = $row [0];
                $return["mes"] = $row[1];
                $return["anio"] = $row[2];
                if($result2 = $mysqli->query("SELECT COUNT(*) FROM visita WHERE codigo='$codigo';")) {
                        $row2 = mysqli_fetch_array($result2, MYSQLI_BOTH);
                        $return["visitast"] = $row2 [0];
                        if($result3 = $mysqli->query("SELECT COUNT(*) FROM visita WHERE codigo ='$codigo' AND
DATE_SUB(CURDATE(),INTERVAL $mes DAY)<= fechav;
")) {
                                $row3 = mysqli_fetch_array($result3, MYSQLI_BOTH);
                                $return["visitasm"] = $row3 [0];
                                if($result4 = $mysqli->query("SELECT COUNT(*) FROM visita WHERE codigo
='$codigo' AND DATE_SUB(CURDATE(),INTERVAL $semana DAY)<= fechav;
")) {
                                        $row4 = mysqli_fetch_array($result4, MYSQLI_BOTH);
                                        $return["visitass"] = $row4 [0];
                                        $return["resultado"] = 0; /*resultado = 0 Consulta multiple realizada con exito*/
                        }
                        else
                                $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
                 }
                        else
                                $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
                }
                else
                        $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
        }
        else
                $return["resultado"] = 2; /*resultado = 2 Codigo no registrado*/
 }
else
        $return["resultado"] = 1; /*resultado = 1 error en la base de datos*/
```

/----------

El script que acumulará estas visitas es contador.php. La interacción de este script es directamente con el navegador web y no con la aplicación móvil. El nagevador, a su vez llamará a este script con un elemento jQuery para realizar la petición de añadir una nueva vistia.

----------contador.php----------

```
<?php

/*Recibiendo variable de entrada código de rastreo*/
$codigo = $_POST['codigo'];

/*Cadena de conexión*/
…

/*Consulta Insertando una nueva visita,*/
if ($result = $mysqli->query("INSERT INTO visita(codigo) VALUES('$codigo');")){
printf("Operacion realizada con exito");
}
   /* Liberando memoria */
   $result->close();
$mysqli->close();
?>
```

/----------

El fragmento de código HTML que deberá agregarse en el navegador web es el siguiente:

/----------

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
</script>
<script>
function Contador(){
jQuery.ajax({
      type: "POST",
      url: 'http://www.tesispdma.webuda.com/componentes/contador.php',
      data: {codigo: 'COLOCA_AQUI_TU_CODIGO'},
       success:function(data) {
      alert(data);
       }
   });
}
onload=Contador;
</script>
```

/----------

Deberá colocarse entre las etiquetas <head> y </head>. El texto que indica COLOCA_AQUI_TU_CODIGO deberá ser sustituido por el código de rastreo correspondiente, sin remover las comillas simples.

La aplicación móvil se puede implementar, entre tantas opciones, con las herramientas que proporciona Android en su sitio oficial para desarrolladores [22]. El complemento ADT (Android Developer Tools) permite programar aplicaciones para la plataforma Android utilizando lenguaje de programación *Java,* combinado con *XML* para desplegar las vistas de la aplicación.

Cada clase Java puede ser asociada a un documento XML, el documento que almacena los elementos de la interfaz gráfica como imágenes, botones y cajas de texto está programado en este lenguaje, y las acciones que podrá realizar el usuario sobre los elementos de la interfaz gráfica están definidas en la clase Java. Una aplicación Android está organizada por varias clases Java que muestran elementos gráficos al usuario e interactúan entre sí, a cada una de estas clases se les llama *Activity.* El diagrama de estados mostrado en la figura 5.4 fue realizado para que cada estado de la aplicación se convierta en una Activity. A continuación se mostrarán los principales fragmentos de código que hacen funcionar la aplicación SiteTracker.

Estado: Acceso. Éste es el código XML completo de la vista principal de la aplicación. La etiqueta ScrollView indica que si el contenido de la vista no se alcanza a mostrar por completo en pantalla, el usuario podrá usar el gesto “deslizar” para apreciar el contenido completo. La etiqueta ImageView se utiliza para mostrar imágenes, TextView para mostrar cadenas de texto, EditText es un campo de entrada de texto. El elemento EditText con identificador "@+id/pass" contiene el atributo android:password="true", todo texto que se introduzca en ese campo aparecerá con asteriscos “******”. El elemento TextView con identificador "@+id/txtresult" será utilizado para mostrar texto en caso de que exista algún error en el acceso cuando el usuario toque el botón "@+id/btnLogin". Como puede notarse, el texto es una cadena vacía android:text="", pero la clase Java podrá modificar el valor de ese atributo, dependiendo de la situación de error. La figura 5.6 presenta el resultado de este archivo.

/---------- activity_main.xml ----------\

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/grad2">

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <!--imagen de cabecera-->
    <ImageView android:src="@drawable/header2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
```

```
    android:layout_marginLeft="2dip"/>

<!--  nombre usuario -->
<TextView android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18dip"
    android:textColor="#0040FF"
    android:text="USUARIO"/>

<EditText android:id="@+id/user"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dip"
    android:layout_marginBottom="20dip"
    android:singleLine="true"/>

<!--  contraseña -->
<TextView android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18dip"
    android:textColor="#0040FF"
    android:text="CONTRASEÑA"/>

<EditText android:id="@+id/pass"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dip"
    android:singleLine="true"
    android:password="true"/>

<!--Testo de respuesta en caso de error -->
<TextView android:id="@+id/txtresult"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="12dip"
    android:textColor="#FF0000"
    android:text=""/>

<!-- boton de acceso -->
<Button android:id="@+id/btnLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:layout_marginBottom="30dp"
    android:text="Login"/>

<!--texto de registro-->
<TextView android:id="@+id/register"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="#0040FF"
    android:textSize="20dip"
    android:layout_marginBottom="20dip"
    android:text="Nuevo en SiteTracker? Registrate"/>
```

```
    <!--texto de recuperación de contraseña-->
    <TextView android:id="@+id/recover"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textColor="#0040FF"
        android:textSize="20dip"
        android:text="Olvidé mi contraseña"/>

    <!--imagen de pie-->
    <ImageView android:src="@drawable/footer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dip"
        android:layout_marginTop="20dip"/>

</LinearLayout>
    </ScrollView>
```

/----------\

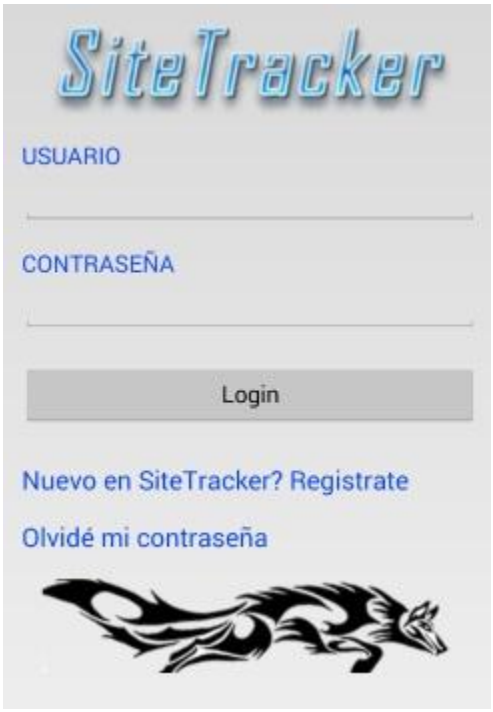

Figura 5.6. activity_main.xml

El estado Acceso obtiene la información de id/pass e id/user de la interfaz de usuario. Realiza las acciones checar conexión y permitir acceso. También conduce a la activity Registro y Recupera.
El método verificaInternet de MainActivity.java se encarga de revisar el estado de red del dispositivo, en caso de que no exista una conexión a internet disponible se mostrará un cuadro de dialogó pidiendo al usuario que verifique su conexión a internet. Entonces, la aplicación se cerrará.

/---------- MainActivity.java ----------\

```
package tesispdma.fi.sitetracker;

        //Se impirtan las bibliotecas de funciones necesarias
import …

public class MainActivity extends Activity {

   // url para la petición al servidor.
   private static String url = "http://tesispdma.webuda.com/componentes/acceso.php";
   // tags JSON que recibe del servidor
   private static final String TAG_RESULTADO = "resultado";
   private static final String TAG_IDUSUARIO = "id_usuario";

   //Se declaran los elementos de la interfaz gráfica que se van a utilizar
   EditText nombre_usuario;
   EditText contrasenia;
   Button acceso;
   TextView registro;
   TextView recuperar;
   TextView txt_result;

  //El siguiente método se ejecuta cuando se inicia la Activity
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      //Asociando la activity con el archivo activity_main.xml, despliega en pantalla el resultado del archivo.
      setContentView(R.layout.activity_main);

      //método que verifica la conexión a internet al iniciar Activity
      verificaInternet();

      //Verificando un log in previo
      SharedPreferences prefs1 = getSharedPreferences("Preferencias",Context.MODE_PRIVATE);
      boolean login = prefs1.getBoolean("logged",false);
      if(login){
         Intent myIntent = new Intent(MainActivity.this, ListaCodigos.class);
         MainActivity.this.finish();
         startActivity(myIntent);
      }

      /*Identificando los elementos de la interfaz gráfica a utilizarse, provenientes de activity_main.xml*/
      nombre_usuario = (EditText) findViewById(R.id.user);
      contrasenia = (EditText) findViewById(R.id.pass);
      acceso = (Button) findViewById(R.id.btnLogin);
      registro = (TextView) findViewById(R.id.register);
      recuperar = (TextView) findViewById(R.id.recover);
      txt_result = (TextView) findViewById(R.id.txtresult);

      //Abre una activity para registro de un nuevo usuario, se ejecuta cuando el usuario toca el texto de registro
      registro.setOnClickListener(new View.OnClickListener(){
         @Override
         public void onClick(View view){
```

```
        Intent myIntent = new Intent(MainActivity.this, Registro.class);
        startActivity(myIntent);
    }
});

//Abre una nueva Activity para recuperación de contraseña
recuperar.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        Intent myIntent = new Intent(MainActivity.this, Recupera.class);
        startActivity(myIntent);
    }
});

//Accion al presionar el botón log in
acceso.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view){
        try {
            //Tomando el valor de usuario y contraseña
            String usuario = nombre_usuario.getText().toString();
            String contrasena = contrasenia.getText().toString();

            //El siguiente if verifica que el usuario haya ingresado algún valor a los campos de usuario y contraseña
            if(!(usuario.equals("")||contrasena.equals(""))){
                // Creating JSON Parser instance
                JSONParser jParser = new JSONParser();
                // Cargando los valores que se enviarán al archivo en PHP que reside en el servidor
                List<NameValuePair> params = new ArrayList<NameValuePair>();
                params.add(new BasicNameValuePair("nombre_usuario", usuario));
                params.add(new BasicNameValuePair("contrasenia", contrasena));

                // getting JSON Object
                // Realizando una petición Http a la dirección URL especificada antes y con los parámetros anteriores.
                JSONObject json = jParser.makeHttpRequest(url, "POST", params);
                Log.d("Create Response", json.toString());
                //Recopilando los resultados obtenidos.
                int result = json.getInt(TAG_RESULTADO);
                switch(result){
                    case 0:
                        int uid = json.getInt(TAG_IDUSUARIO);
                        //Guardando el identificador de usuario en la memoria de la aplicación.
                        SharedPreferences prefs = getSharedPreferences("Preferencias",Context.MODE_PRIVATE);
                        SharedPreferences.Editor editor = prefs.edit();
                        editor.putBoolean("logged",true);
                        editor.putInt("user_id",uid);
                        editor.putString("user_name",usuario);
                        editor.commit();
                        txt_result.setText("");
                        contrasenia.setText("");
                        //Llamando a una nueva activity
                        Intent myIntent = new Intent(MainActivity.this, ListaCodigos.class);
                        MainActivity.this.finish();
                        startActivity(myIntent);
                        break;
```

```
                    case 1:
                      //Colocando el texto en caso de error
                      txt_result.setText("Servicio no disponible, intente más tarde");
                      break;
                    case 2:
                      txt_result.setText("El usuario ingresado no existe");
                      break;
                    case 3:
                      txt_result.setText(R.string.contraseniaincorrecta);
                      break;
                  }
                }
                else
                  txt_result.setText("Por favor ingresa todos los campos");
              }
              catch (JSONException e) {
                e.printStackTrace();
              }
            }
          });
        }

  //El método verifica internet obtiene información acerca del estado de la red del dispositivo, si no existe una conexión a internet activa
//y disponible la aplicación muestra un cuadro de dialogo, pidiendo al usuario que verifique su conexión a internet.
    public void verificaInternet(){
    //Verificando conexión a internet
    ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo ni = cm.getActiveNetworkInfo();
    if(!(ni!=null && ni.isConnectedOrConnecting())){
      AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
      alertDialogBuilder.setTitle(R.string.nohayinternet);
      alertDialogBuilder.setMessage(R.string.verificainternet).setCancelable(false).setPositiveButton("OK",new
DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog,int id) {
          // if this button is clicked, close
          // current activity
          MainActivity.this.finish();
        }
      });
      // create alert dialog
      AlertDialog alertDialog = alertDialogBuilder.create();
      // show it
      alertDialog.show();
      }
    }
}
```

/----------\

El script acceso.php recibe los datos a través de una petición http, que se realiza con el método "*makeHttpRequest"* del objeto JSONParser. Los parámetros que recibe este método son: la dirección URL a la que se hace la petición; el método que se utilizará, en este caso POST, que impone valores; y una lista de parámetros en pares, variable y valor.

Esta lista de parámetros se prepara con el objeto *params* que es una matriz de pares de valores.

List<NameValuePair> params = new ArrayList<NameValuePair>();
params.add(new BasicNameValuePair("nombre_usuario", usuario));

El script lo recibe como: *$nick = $_POST['nombre_usuario'];*

Los datos que devuelve el Script se decodifican y se almacenan en un objeto de la clase JSONObject. Se puede acceder a ellos con los métodos getInt, getString, getBoolean, etc. Introduciendo como parámetro el nombre de la variable dentro del Script en PHP. Por ejemplo, en este caso:

acceso.php obtiene: *$return["id_usuario"] = $user;*
Después lo devuelve: *json_encode($return);*

Después de la petición http de la clase MainActivity.java se obtiene el valor como un entero: *int uid = json.getInt(TAG_IDUSUARIO);*
Donde TAG_IDUSUARIO = "id_usuario"

Antes de conceder el acceso por parte de la aplicación, la variable *uid* se almacena en la memoria del dispositivo móvil, para que las demás Activity puedan hacer uso de ella. Esto se realiza con un objeto de la clase *SharedPreferences.*

La estructura de los demás estados de la aplicación, es específico de los que son formularios, es muy similar. Lo mismo sucede con los archivos XML correspondientes. A continuación se mostrarán fragmentos de código de los estados que hacen de formulario.

/---------- Registro.java ----------\

```
public class Registro extends Activity {

    // url para la petición al servidor
    private static String url = "http://tesispdma.webuda.com/componentes/registrousuario.php";
    // tags JSON que recibe del servidor
    private static final String TAG_RESULTADO = "resultado";
    //Elementos de la interfaz gráfica
    EditText nombre_per;
    EditText nombre_usuario;
    EditText mail;
    EditText contrasenia;
    EditText confcontrasenia;
    Button registro;
    TextView txt_result;

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

        registro.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
        public void onClick(View view) {
            try {
                String nom_per = nombre_per.getText().toString();
                String nom_usr = nombre_usuario.getText().toString();
                String email = mail.getText().toString();
                String stpass = contrasenia.getText().toString();
                String ndpass = confcontrasenia.getText().toString();
               if(!(nom_per.equals("")||nom_usr.equals("")||email.equals("")||stpass.equals(""))){
                if(stpass.equals(ndpass)){
                    // Creating JSON Parser instance
                    JSONParser jParser = new JSONParser();
                    // Building Parameters
                    List<NameValuePair> params = new ArrayList<NameValuePair>();
                    params.add(new BasicNameValuePair("nombre_per", nom_per));
                    params.add(new BasicNameValuePair("nombre_usuario", nom_usr));
                    params.add(new BasicNameValuePair("contrasenia", stpass));
                    params.add(new BasicNameValuePair("correo", email));

                    // getting JSON Object
                    // Note that create product url accepts POST method
                    JSONObject json = jParser.makeHttpRequest(url, "POST", params);
                    // check log cat fro response
                    Log.d("Create Response", json.toString());
                    int result = json.getInt(TAG_RESULTADO);
                    switch (result) {
                        case 0:
                            Context context = getApplicationContext();
                            String text = nom_usr +" Registrado exitosamente";
                            int duration = Toast.LENGTH_SHORT;
                            assert context != null;
                            Toast toast = Toast.makeText(context, text, duration);
                            toast.show();
                            Registro.this.finish();
                            break;
                        case 1:
                            txt_result.setText("Servicio no disponible, intente más tarde");
                            break;
                        case 2:
                            txt_result.setText("El nombre de usuario o correo ya se encuentran registrados");
                            break;
                    }
                }
                else
                    txt_result.setText(R.string.repeatedpasword);
               }
                else
                  txt_result.setText("Por favor ingrese todos los campos");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
  }
/----------------------------------------------------------------------------------------------------------------------\
```

/---------- Recupera.java ----------\

```
public class Recupera extends Activity {

   // url para la petición al servidor
   private static String url = "http://tesispdma.webuda.com/componentes/recupera.php";
   // tags JSON que recibe del servidor
   private static final String TAG_RESULTADO = "resultado";

   //Elementos de la interfaz gráfica
   EditText nombre_de_usuario;
   Button enviar;
   TextView txt_result;

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

enviar.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            try {
               String usuario = nombre_de_usuario.getText().toString();
              if (!usuario.equals("")){
               // Creating JSON Parser instance
               JSONParser jParser = new JSONParser();
               // Building Parameters
               List<NameValuePair> params = new ArrayList<NameValuePair>();
               params.add(new BasicNameValuePair("nombre_usuario", usuario));
               // getting JSON Object
               // Note that create product url accepts POST method
               JSONObject json = jParser.makeHttpRequest(url, "POST", params);
               // check log cat fro response
               Log.d("Create Response", json.toString());
               int result = json.getInt(TAG_RESULTADO);
               switch (result) {
                  case 0:
                     Context context = getApplicationContext();
                     String text = "Correo enviado exitosamente";
                     int duration = Toast.LENGTH_SHORT;
                     assert context != null;
                     Toast toast = Toast.makeText(context, text, duration);
                     toast.show();
                     Recupera.this.finish();
                     break;
                  case 1:
                     txt_result.setText("Servicio no disponible, intente más tarde");
                     break;
                  case 2:
                     txt_result.setText("El usuario ingresado no existe");
                     break;
               }
```

```
            }
              else
                txt_result.setText("Por favor ingrese todos los campos");
            } catch (JSONException e) {
              e.printStackTrace();
            }
          }
      });

  }
```

/----------\

/---------- NuevoCodigo.java ----------\

```
public class NuevoCodigo extends Activity {

   // url para la petición al servidor
   private static String url = "http://tesispdma.webuda.com/componentes/registrocodigo.php";

   //Cadenas para recepcion de informacion JSON
   private static final String TAG_RESULTADO = "resultado";
   private static final String TAG_CODIGO= "codigo";

   EditText alias;
   TextView respuesta;
   Button registro;

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

      registro.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            try {
               //Tomando el valor del usuario registrado en la aplicacion
               SharedPreferences prefs = getSharedPreferences("Preferencias", Context.MODE_PRIVATE);
               String usuario = Integer.toString(prefs.getInt("user_id", 0));
               //Tomando el alias de la interfaz de usuario
               String alias_pag = alias.getText().toString();
               if(!alias_pag.equals(""))
               {
                  // Creating JSON Parser instance
                  JSONParser jParser = new JSONParser();
                  // Building Parameters
                  List<NameValuePair> params = new ArrayList<NameValuePair>();
                  params.add(new BasicNameValuePair("id_usuario", usuario));
                  params.add(new BasicNameValuePair("alias", alias_pag));
                  // getting JSON Object
                  // Note that create product url accepts POST method
                  JSONObject json = jParser.makeHttpRequest(url, "POST", params);
                  // check log cat fro response
                  Log.d("Create Response", json.toString());
```

```
                int result = json.getInt(TAG_RESULTADO);
                switch (result) {
                    case 0:
                        String code = json.getString(TAG_CODIGO);
                        //Guardando el reciente código en las preferencias
                        SharedPreferences.Editor editor = prefs.edit();
                        editor.putString("current_code",code);
                        editor.commit();
                        respuesta.setText("");
                        Context context = getApplicationContext();
                        String text = "Registro exitoso";
                        int duration = Toast.LENGTH_SHORT;
                        assert context != null;
                        Toast toast = Toast.makeText(context, text, duration);
                        toast.show();
                        Intent myIntent = new Intent(NuevoCodigo.this, Fragmento.class);
                        startActivity(myIntent);
                        break;
                    case 1:
                        respuesta.setText("Servicio no disponible, intente más tarde");
                        break;
                    case 2:
                        respuesta.setText(R.string.yatiene);
                        break;
                }
            }
            else
                respuesta.setText("Ingrese un alias");
          } catch (JSONException e) {
                e.printStackTrace();
          }
        }
    });
  }
/----------------------------------------------------------------------------------------------------------------------\
```

Figura 5.7. Resultado de los archivos XML correspondientes a Registro.java, Recupera.java y NuevoCodigo.java, respectivamente.

Estas tres clases java tienen un nuevo elemento en común. En el capítulo tres de este documento de habló del Acknowledge y es en estas tres vistas donde se aplica. Utilizando el criterio de la figura 3.7 se utiliza el Acknowledge al tocar el botón que realiza la petición al servidor. Los usuarios no necesitan ser alertados antes de que se realice la acción, tampoco es fácil que los usuarios envíen datos accidentalmente pero la acción no tiene un resultado obvio, el usuario necesita ser informado del resultado de su petición. Para informar un resultado de este tipo se utiliza la clase Toast, de Android. Las siguientes líneas de código preparan y muestran un mensaje de acknowledge en la pantalla del dispositivo, que permanecerá ahí un cierto tiempo independientemente de la activity que se esté ejecutando.

```
Context context = getApplicationContext();
String text = "Registro exitoso";
int duration = Toast.LENGTH_SHORT;
Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Estado: ListaCódigos. En este estado se implementa una lista de elementos cuya dimensión puede variar dependiendo del usuario. Si los elementos no se pueden listar en la totalidad de la pantalla, la lista podrá deslizarse. Para evitar muchos elementos en pantalla a la vez también se implementa un menú desplegable. Estas entidades se implementan más por parte del lenguaje XML que por las clases en Java.

El menú es un archivo en lenguaje XML cuyo producto se despliega al presionar el botón menú del dispositivo. Las acciones de cada opción del menú deben ser programadas dentro de la clase Java que lo invoca.

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Busca el archivo XML que desplegará con el menú
        getMenuInflater().inflate(R.menu.lista_codigos, menu);
        return true;
    }

@Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.id1:
                    //Acciones a realizar
                    return true;
             case R.id.id2:
                    //Acciones a realizer
                    return true;
             default:
             }
}
```

El código del documento en XML para el menú de ListaCodigos.java es el siguiente:

/---------- lista_codigos.xml ----------\

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/nuevo_codigo" android:title="@string/nuevocode"
        android:icon="@drawable/ic_action_new"/>
    <item android:id="@+id/cierra_sesion" android:title="@string/cierrasesion"
        android:icon="@drawable/ic_action_accounts"/>
    <item android:id="@+id/elimina_cuenta" android:title="Eliminar Mi Cuenta"
        android:icon="@drawable/ic_action_discard"/>
</menu>
```

/----------\

A cada elemento del menú se le asigna un identificador, una cadena de texto con la acción que va a realizar y un ícono.

Para implementar la lista de páginas que ha registrado el usuario se agregó una etiqueta ListView al documento XML asociado a ListaCodigos.java.

```
    <ListView
        android:id="@android:id/list"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"/>
```

La lista cuenta con un identificador, y cada elemento de la lista es una instancia de list_item.xml.

/---------- list_item.xml ----------\

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:orientation="horizontal">
   <TextView
      android:id="@+id/pid"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:paddingLeft="6dip"
      android:textSize="20dip"
      android:textColor="#909090"
      android:paddingRight="10dip"
      android:layout_marginTop="10dip"
      android:text="CODIGO DE RASTREO”
      />

   <!-- Name Label -->
   <TextView
      android:id="@+id/name"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:paddingTop="3dip"
      android:paddingLeft="6dip"
      android:textSize="22dip"
      android:textStyle="bold"
      android:textColor="#787878"
      android:paddingBottom="10dip"
      android:layout_marginTop="10dip"
      android:text="NOMBRE DE LA PAGINA"/>

</LinearLayout>
```

/----------\

Mediante un ciclo for se rellenará la lista con cada uno de los elementos recibidos de la matriz *codigos* de tipo JSONArray

```
for (int i = 0; i < codigos.length(); i++) {
                    JSONObject c = codigos.getJSONObject(i);

                    // Storing each json item in variable
                    String id = c.getString(TAG_CODIGO);
                    String name = c.getString(TAG_ALIAS);

                    // creating new HashMap
                    HashMap<String, String> map = new HashMap<String, String>();

                    // adding each child node to HashMap key => value
                    map.put(TAG_CODIGO, id);
                    map.put(TAG_ALIAS, name);

                    // adding HashList to ArrayList
                    codeList.add(map);
                }
```

Después la despliega en pantalla con un objeto de la clase ListAdapter.

```
ListAdapter adapter = new SimpleAdapter(
    ListaCodigos.this, codeList,
    R.layout.list_item, new String[] { TAG_CODIGO,
    TAG_ALIAS},
    new int[] { R.id.pid, R.id.name });
setListAdapter(adapter);
```

Las partes esenciales de código de las clases Java restantes se muestra a continuación. Nótese que dentro de las acciones del menú desplegable existe la opción de eliminar código de rastreo o eliminar la cuenta del usuario. Ambas opciones utilizan de nuevo el criterio de la figura 3.7 para utilizar, en este caso, una confirmación previa antes de eliminar información importante.

/---------- ListaCodigos.java ----------\

```
public class ListaCodigos extends ListActivity {

   // url para la petición al servidor
   private static String url = "http://tesispdma.webuda.com/componentes/listado.php";
   private static String url2 = "http://tesispdma.webuda.com/componentes/bajausuario.php";

   //Creando parser de JSON
   JSONParser jParser = new JSONParser();
   // codigos JSONArray
   JSONArray codigos = null;

   //Elementos de la interfaz grafica
   ArrayList<HashMap<String, String>> codeList;
   TextView showuser;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.lista_codigos);
      //Recibiendo elementos de la lista
      List<NameValuePair> params = new ArrayList<NameValuePair>();
      params.add(new BasicNameValuePair("id_usuario", usuario));
      JSONObject json = jParser.makeHttpRequest(url, "POST", params);
      Log.d("Create Response", json.toString());
      try{
         int result = json.getInt(TAG_RESULTADO);
         switch (result) {
            case 0:
              int numcods = json.getInt(TAG_NUMERO);
              codigos = json.getJSONArray(TAG_PAGINAS);
              if(codigos.length()!=0){
                for (int i = 0; i < codigos.length(); i++) {
                   JSONObject c = codigos.getJSONObject(i);

                   // Storing each json item in variable
                   String id = c.getString(TAG_CODIGO);
                   String name = c.getString(TAG_ALIAS);
```

```
            // creating new HashMap
            HashMap<String, String> map = new HashMap<String, String>();

            // adding each child node to HashMap key => value
            map.put(TAG_CODIGO, id);
            map.put(TAG_ALIAS, name);

            // adding HashList to ArrayList
            codeList.add(map);
          }
           //Saludo al usuario
           showuser.setText("Bienvenido "+nombre+" usted tiene "+numcods+" codigos registrados");
           //reimprimiendo lista
           ListAdapter adapter = new SimpleAdapter(
                ListaCodigos.this, codeList,
                R.layout.list_item, new String[] { TAG_CODIGO,
                TAG_ALIAS},
                new int[] { R.id.pid, R.id.name });
            setListAdapter(adapter);
        }
          else{
          showuser.setText(R.string.invite);
        }
          break;
      case 1:
          showuser.setText("Servicio no disponible, intente más tarde");
          break;
      case 2:
          showuser.setText("Error en la base de datos, intente más tarde");
          break;
    }
}
catch (JSONException e){
    e.printStackTrace();
}

ListView lv = getListView();

lv.setOnItemClickListener(new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view,
                        int position, long id) {
        // getting values from selected ListItem
        String pid = ((TextView) view.findViewById(R.id.pid)).getText()
                .toString();
        SharedPreferences prefs = getSharedPreferences("Preferencias",Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.putString("current_code",pid);
        editor.commit();
        //Starting new intent
        Intent in = new Intent(getApplicationContext(),Detalles.class);
        ListaCodigos.this.finish();
        startActivity(in);
    }
```

```
    });
  }

  @Override
  public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.lista_codigos, menu);
    return true;
  }

  @Override
  public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
      case R.id.nuevo_codigo:
        Intent in = new Intent(getApplicationContext(),NuevoCodigo.class);
        ListaCodigos.this.finish();
        startActivity(in);
        return true;
      case R.id.cierra_sesion:
        SharedPreferences prefs = getSharedPreferences("Preferencias",Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = prefs.edit();
        editor.clear();
        editor.commit();
        Intent in2 = new Intent(getApplicationContext(),MainActivity.class);
        ListaCodigos.this.finish();
        startActivity(in2);
        return true;
      case R.id.elimina_cuenta:
        //Confirmación previa al eliminado de cuenta
        AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
        alertDialogBuilder.setTitle("Eliminar su cuenta");
        alertDialogBuilder.setMessage(R.string.eliminandocuenta).setCancelable(true).setPositiveButton("Aceptar",new
DialogInterface.OnClickListener() {
          public void onClick(DialogInterface dialog,int id) {
            // if this button is clicked, close
            // current activity
            //Tomando el valor del usuario registrado en la aplicacion
            SharedPreferences prefs = getSharedPreferences("Preferencias", Context.MODE_PRIVATE);
            String usuario2nd = Integer.toString(prefs.getInt("user_id", 0));
            //Realizando la petición al servidor
            List<NameValuePair> params2 = new ArrayList<NameValuePair>();
            params2.add(new BasicNameValuePair("id_usuario", usuario2nd));
            JSONObject json2 = jParser.makeHttpRequest(url2, "POST", params2);
            //Contexto para mostrar acknowledge
            Context context = getApplicationContext();
            int duration = Toast.LENGTH_SHORT;
            //Ejecutando peticion
            Log.d("Create Response", json2.toString());
            try{
              int result = json2.getInt(TAG_RESULTADO);
              switch (result) {
                case 0:
                  //Eliminando información de preferencias
                  SharedPreferences.Editor editor = prefs.edit();
                  editor.clear();
```

```
                        editor.commit();
                        //Acknowledge de cuenta eliminada
                        String text = "Su cuenta ha sido eliminada";
                        assert context != null;
                        Toast toast = Toast.makeText(context, text, duration);
                        toast.show();
                        //Regresando a la pantalla de acceso
                        Intent in2 = new Intent(getApplicationContext(),MainActivity.class);
                        ListaCodigos.this.finish();
                        startActivity(in2);
                        break;
                    case 1:
                        //Acknowledge de error
                        String text2 = "Error en el servicio";
                        assert context != null;
                        Toast toast2 = Toast.makeText(context, text2, duration);
                        toast2.show();
                        break;
                    case 2:
                        //Acknowledge de cuenta eliminada
                        String text3 = "Error en el servicio";
                        assert context != null;
                        Toast toast3 = Toast.makeText(context, text3, duration);
                        toast3.show();
                        break;
                }
            }
            catch (JSONException e){
                e.printStackTrace();
            }
        }
    });
    alertDialogBuilder.setNegativeButton("Cancelar",new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog,int id) {
        }
    });
    // create alert dialog
    AlertDialog alertDialog = alertDialogBuilder.create();
    // show it
    alertDialog.show();
    return true;
  default:
    return super.onOptionsItemSelected(item);
  }
}
```

/----------\

/---------- Detalles.java ----------\

```
public class Detalles extends Activity {

    TextView pagename;
    TextView fechac;
    TextView rstcode;
```

```
TextView visitastot;
TextView visitasmes;
TextView visitassem;

@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_detalles);
   //Tomando el valor de código seleccionado
   SharedPreferences prefs = getSharedPreferences("Preferencias", Context.MODE_PRIVATE);
   String code = prefs.getString("current_code", "");

   //Peticion al servidor de desplegar información
   List<NameValuePair> params = new ArrayList<NameValuePair>();
   params.add(new BasicNameValuePair("codigo", code));
   JSONObject json = jParser.makeHttpRequest(url, "POST", params);
   Log.d("Create Response", json.toString());
   try{
      int result = json.getInt(TAG_RESULTADO);
      switch (result) {
         case 0:
            String alias = json.getString(TAG_ALIAS);
            String dia = json.getString(TAG_DIA);
            String mes = json.getString(TAG_MES);
            String anio = json.getString(TAG_ANIO);
            String visitast = json.getString(TAG_VISITAST);
            String visitass = json.getString(TAG_VISITASS);
            String visitasm = json.getString(TAG_VISITASM);
            pagename.setText(alias);
            rstcode.setText(R.string.codrast);
            rstcode.append(" "+code);
            fechac.setText("Creado el "+dia+" / "+mes+" / "+anio);
            visitastot.setText(visitast);
            visitassem.setText(visitass);
            visitasmes.setText(visitasm);
            break;
         case 1:
            pagename.setText("Servicio no disponible, intente más tarde");
            break;
      }
   }
   catch (JSONException e){
      e.printStackTrace();
   }
}


@Override
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it is present.
   getMenuInflater().inflate(R.menu.detalles, menu);
   return true;
}

@Override
```

```
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.muestra_frag:
                Intent in = new Intent(getApplicationContext(),Fragmento.class);
                startActivity(in);
                return true;
            case R.id.actualiza:
                Intent reload = new Intent(getApplicationContext(),Detalles.class);
                this.finish();
                startActivity(reload);
                return true;
            case R.id.elimina_codigo:
                //Confirmación previa al eliminado de cuenta
                AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
                alertDialogBuilder.setTitle(R.string.eliminacodigo);
                alertDialogBuilder.setMessage(R.string.eliminandocodigo).setCancelable(true).setPositiveButton("Aceptar",new
DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog,int id) {
                        //Tomando el valor de código seleccionado
                        SharedPreferences prefs = getSharedPreferences("Preferencias", Context.MODE_PRIVATE);
                        String code = prefs.getString("current_code", "");
                        List<NameValuePair> params2 = new ArrayList<NameValuePair>();
                        params2.add(new BasicNameValuePair("codigo", code));
                        JSONObject json2 = jParser.makeHttpRequest(url2, "POST", params2);
                        //Contexto para mostrar acknowledge
                        Context context = getApplicationContext();
                        int duration = Toast.LENGTH_SHORT;
                        //Ejecutando peticion
                        Log.d("Create Response", json2.toString());
                        try{
                            int result = json2.getInt(TAG_RESULTADO);
                            switch (result) {
                                case 0:
                                    assert context != null;
                                    Toast toast = Toast.makeText(context, R.string.codigoeliminado, duration);
                                    toast.show();
                                    //Regresando a la pantalla de listado de códigos
                                    Intent in = new Intent(getApplicationContext(),ListaCodigos.class);
                                    Detalles.this.finish();
                                    startActivity(in);
                                    break;
                                case 1:
                                    //Acknowledge de error
                                    String text2 = "Error en el servicio";
                                    assert context != null;
                                    Toast toast2 = Toast.makeText(context, text2, duration);
                                    toast2.show();
                                    break;
                                case 2:
                                    //Acknowledge de cuenta eliminada
                                    String text3 = "Error en el servicio";
                                    assert context != null;
                                    Toast toast3 = Toast.makeText(context, text3, duration);
                                    toast3.show();
                                    break;
```

```
                    }
                }
                catch (JSONException e){
                    e.printStackTrace();
                }
                catch (NullPointerException n){
                    n.printStackTrace();
                }
            }
        });
        alertDialogBuilder.setNegativeButton("Cancelar",new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int id) {
            }
        });
        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();
        // show it
        alertDialog.show();
        return true;
      default:
        return super.onOptionsItemSelected(item);
    }
  }

  @Override
  public void onBackPressed() {
    Intent in = new Intent(getApplicationContext(),ListaCodigos.class);
    Detalles.this.finish();
    startActivity(in);
  }
/----------------------------------------------------------------------------------------------------------\
```

Figura 5.8 Resultado de Detalles.java y activity_detalles.xml

Implementación de anuncios publicitarios de AdMob. El medio de monetización de esta aplicación, que se estableció desde el capítulo 2, se implementa con la biblioteca de AdMob. Es necesario descargar el SDK desde la página de Google Developers [23] y agregarlo a su proyecto como una librería externa, dependiendo del entorno de desarrollo serán las acciones que deba realizar.

Para poder añadir un anuncio y cobrar los ingresos generados por las impresiones de éste, se debe enlazar una cuenta de google al servicio de AdMob (http://es.AdMob.com, evite confundirla con https://apps.AdMob.com/) y contar con una cuenta de *PayPal* [24]. Después de dar de alta el servicio se puede registrar un nuevo bloque de anuncios. Al registrarlo se obtiene como resultado un identificador llamado *Publisher ID*, que se utilizará al momento de realizar la petición de impresión del anuncio dentro de la aplicación.

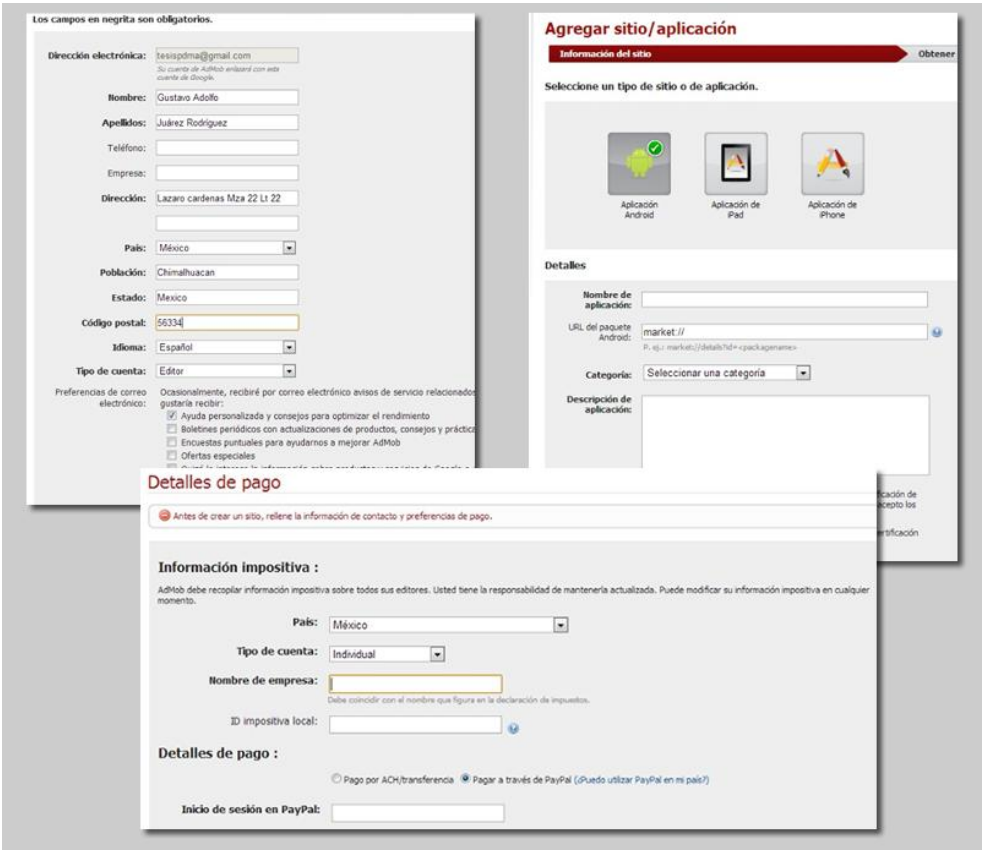

Figura 5.9. Registro usuario y un nuevo bloque de anuncios en AdMob. http://es.AdMob.com

Los anuncios pueden ser añadidos desde el archivo XML de cualquier vista o mediante código en Java. En la aplicación de ejemplo se añaden desde la interfaz gráfica debido a que no es necesario tener control sobre ellos mediante código. Cuando se carguen las vistas ListaCódigos, Detalles y Fragmento (figura 5.4) se cargarán los bloques publicitarios inmediatamente, respetando las directrices y políticas de AdMob.

La etiqueta XML que carga estos bloques es denominada *AdView* y debe tener la siguiente estructura mínima:

```
<com.google.ads.AdView
ads:adSize="BANNER"
ads:adUnitId="a15253a1e66dac2"
ads:testDevices="TEST_EMULATOR"
ads:loadAdOnCreate="true" ></com.google.ads.AdView>
```

Donde *adUnitId* es la clave obtenida *Publisher ID* y testDevices es un atributo que se incluye al momento de ejecutar la aplicación en modo de depuración o dentro de un emulador, antes de publicar la *app* se debe remover este atributo. Se debe tener en cuenta que la etiqueta que contenga el *AdView* debe incluir el siguiente atributo:

```
xmlns:ads="http://schemas.android.com/apk/lib/com.google.ads"
```

A continuación se muestra una parte del código de *activity_fragmento.xml* donde se implementa un adview.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   xmlns:ads="http://schemas.android.com/apk/lib/com.google.ads"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

<LinearLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:background="@drawable/grad2"
   tools:context=".Fragmento">

      <com.google.ads.AdView
       ads:adSize="BANNER"
       ads:adUnitId="a15253a1e66dac2"
       ads:testDevices="TEST_EMULATOR"
       ads:loadAdOnCreate="true" ></com.google.ads.AdView>
```

5.4. Integración y pruebas

Para integrar el sistema los componentes fueron almacenados en el sitio personal en 000webhost (tesispdma.webuda.com/componentes), también se programó la base de datos, perteneciente al mismo servicio, mediante el código mostrado en este capítulo y se cargó la aplicación en modo de prueba a un emulador propio de *Android Developer Tools.* La siguiente actividad para el desarrollo de la solución es el plan de pruebas.

Una vez que el software se ha construido, es necesario hacerlo pasar por una serie de pruebas antes de y entrar a la fase de publicación. Mediante dichas pruebas, se medirá su reacción integral frente a diversas acciones que realizarán los usuarios. El objetivo de probar un software es precisamente encontrar errores, entre más errores se encuentren en la prueba, mas correcciones podrán hacerse para entregar un producto de eficacia.

Para nuestra aplicación de ejemplo se realizarán inicialmente pruebas de caja negra. Las pruebas de caja negra reciben este nombre por el hecho de probar un programa sin saber que hay dentro y sin saber cómo funciona. Estas pruebas se desarrollan considerando únicamente los requisitos del sistema, que se han definido previamente en este mismo capítulo. Las primeras pruebas se harán en un emulador de la plataforma Android, después se realizará la integración del sistema en un dispositivo Android real.

Características a probarse:

- **Requerimientos funcionales.** Se probará que la aplicación móvil pueda cumplir con cada uno de los requerimientos que se han especificado en el documento SRS.
- **Integridad de la aplicación.** Se probará que la forma en que se ha programado la aplicación permita una navegación adecuada a través de todas sus funcionalidades.
- **Integridad de la base de datos.** Se probará que la forma en que se ha programado la base de datos y las peticiones que se ejecutan por el servidor preserven la integridad de la base de datos, es decir, que al realizar consultas se evite la perdida de información y la existencia de datos que jamás volverán a ser utilizados.

Categorías de resultados de la prueba:

- **Éxito:** El resultado de la prueba es conforme al resultado esperado.
- **Aceptable.** No se llega al resultado esperado, pero es aceptable y no interfiere con las funcionalidades del sistema.
- **Error.** El resultado de la prueba es incorrecto, y la falla debe ser corregida antes de concluir la fase de prueba.

Casos de prueba:

Caso de prueba 1: Comprobar Conexión a internet.

Prueba 1.
Objetivo: Probar el requerimiento RF01.
Descripción: Ingresar a la aplicación sin disponer de una conexión a internet.
Resultado esperado: La aplicación muestra mensaje de error.

Prueba 2.
Objetivo: Probar el requerimiento RF01.
Descripción: Ingresar a la aplicación disponiendo de una conexión a internet activa y disponible.
Resultado esperado: La aplicación continúa con su curso.

Caso de prueba 2: Registro de usuarios.

Prueba 1.
Objetivo: Probar el requerimiento RF02.
Descripción: Intentar registrar un nuevo usuario.
Resultado esperado: Visualizar el acknowlwdge y volver a la vista anterior.

Prueba 2.
Objetivo: Probar el requerimiento RF02.
Descripción: Intentar registrar un nuevo usuario cuyo nombre de usuario exista previamente en la base de datos.
Resultado esperado: Visualizar mensaje de error correspondiente a nombre de usuario no disponible.

Prueba 3.
Objetivo: Probar el requerimiento RF02.
Descripción: Intentar registrar un nuevo usuario cuyo correo exista previamente en la base de datos.
Resultado esperado: Visualizar mensaje de error correspondiente a dirección de correo ya registrada.

Caso de prueba 3: Acceso al sistema.

Prueba 1
Objetivo: Probar el requerimiento RF03.
Descripción: Intentar acceder con un nombre de usuario que no exista en la base de datos.
Resultado esperado: Visualizar mensaje de error correspondiente a usuario no registrado.

Prueba 2
Objetivo: Probar el requerimiento RF03.
Descripción: Intentar acceder con un nombre de usuario válido pero con una contraseña errónea.
Resultado esperado: Visualizar mensaje de error correspondiente a contraseña errónea.

Prueba 3
Objetivo: Probar el requerimiento RF03.
Descripción: Intentar acceder con un nombre de usuario y contraseña válidos.
Resultado esperado: Acceso al sistema, cambio a la vista ListaCódigos.

Caso de prueba 4: Recuperación de contraseña.

Prueba 1
Objetivo: Probar el requerimiento RF04.
Descripción: Intentar recuperación de contraseña de un nombre de usuario no existente en la base de datos.
Resultado esperado. Visualizar mensaje de error correspondiente a usuario no registrado.

Prueba 2
Objetivo: Probar el requerimiento RF04.
Descripción: Intentar recuperación de contraseña de un nombre de usuario válido.
Resultado esperado. Visualizar acknowledge de correo enviado con éxito y volver a vista Acceso.

Caso de prueba 5: Registro de nuevo código de rastreo

Prueba 1
Objetivo: Probar el requerimiento RF06.
Descripción: Ingresar un nombre de página que no exista en la base de datos para el usuario inscrito.
Resultado esperado: Visualizar el acknowledge de registro exitoso y proceder a la vista Fragmento.

Prueba 2
Objetivo: Probar el requerimiento RF06.
Descripción: Ingresar un nombre de página registrado previamente por el usuario inscrito.
Resultado esperado: Visualizar mensaje de error correspondiente a nombre de página no válido.

Caso de prueba 6: Listado de códigos de rastreo.

Prueba 1
Objetivo: Probar el requerimiento RF05
Descripción: Verificar el despliegue de códigos de rastreo.
Resultado esperado: Correcto despliegue de listado de códigos de rastreo.

Prueba 2
Objetivo: Probar el requerimiento RF05
Descripción: Tocar elemento del listado.
Resultado esperado: Proceder a la vista Detalles con la información del código de rastreo tocado.

Caso de prueba 7: Consulta de visitas

Prueba 1
Objetivo: Probar el requerimiento RF07
Descripción: Verificar el despliegue de información de la vista Detalles.
Resultado esperado: Correcto despliegue de información correspondiente al código de rastreo.

Prueba 2
Objetivo: Probar el requerimiento RF07
Descripción: Añadir visitas artificiales a la base de datos para verificar la certeza de la información desplegada.
Resultado esperado: Despliegue de información acertada correspondiente al código de rastreo.

Caso de prueba 8: Despliegue de anuncios publicitarios.

Prueba 1
Objetivo: Probar el requerimiento RF08.
Descripción: Realizar impresiones en modo de prueba del bloque de anuncios de AdMob.
Resultado esperado: Correcto despliegue de bloque de anuncios.

Prueba 2
Objetivo: Asegurar el cálculo de impresiones de la aplicación.
Descripción: Verificar registro de impresiones fuera del modo de prueba, dentro de la plataforma AdMob.
Resultado esperado: Adecuada suma de impresiones del bloque de anuncios.

Caso de prueba 9: Baja de código de rastreo

Prueba 1
Objetivo: Probar el requerimiento RF09
Descripción: Eliminar un código de rastreo mediante la opción del menú de la vista Detalles.
Resultado esperado: Visualización de *acknowledge* correspondiente a eliminación de código y retorno a vista ListaCódigos.

Caso de prueba 10: Baja de usuario

Prueba 1
Objetivo: Probar el requerimiento RF09
Descripción: Eliminar a un usuario mediante la opción del menú de la vista ListaCódigos.
Resultado esperado: Visualización de *acknowledge* correspondiente a baja de usuario y retorno a vista Acceso.

Prueba 2
Objetivo: Probar el requerimiento RF09
Descripción: Intentar acceder al sistema utilizando un nombre de usuario y contraseña recién eliminados.
Resultado esperado: Visualizar mensaje de error correspondiente a usuario no registrado.

Caso de prueba 11: Navegación dentro de la aplicación.

Prueba 1
Objetivo: Verificar la integridad de la aplicación.
Descripción: Navegar por todas las vistas de la aplicación accediendo y saliendo de ellas de todas las formas posibles.
Resultado esperado: Posibilidad de navegar por todas las vistas de la aplicación.

Caso de prueba 12: Integridad de la base de datos.

Prueba 1
Objetivo: Verificar la integridad de la base de datos.
Descripción: Realizar operaciones en la base de datos mediante las funcionalidades de la aplicación.
Resultado esperado: Inexistencia de pérdida de información e inexistencia de registros basura.

Caso de prueba 13: Integración del sistema

Prueba 1
Objetivo: Verificar la integración del sistema en dispositivos reales.
Descripción: Instalar la aplicación en dispositivos reales con diferentes versiones del sistema operativo Android.
Resultado esperado: Correcta instalación y funcionamiento de la aplicación.

Reporte de resultados.

Pruebas con categoría Aceptable:

- Caso de prueba 12, prueba 1. Al eliminar un código de rastreo de la base de datos es posible seguir añadiendo visitas correspondientes a este código.

  Corrección aplicable: Modificación de código de contador.php, para que verifique que el código de rastreo para el que se hace la petición esté registrado previamente.

- Caso de prueba 5, prueba 1. Es posible agregar un nuevo código de rastreo sin ingresar un "alias", es decir, de nombre nulo.

  Corrección aplicable: Modificar el código de todas las clases que sean de tipo formulario para evitar ingreso de valores nulos.

- Caso de prueba 11, prueba 1. La vista Detalles no mostraba el alias de la página registrada.

  Corrección aplicable: Modificación de clase Detalles.java, activity_detalles.xml y despliega.php para recoger e imprimir en pantalla el alias de la página.

- Caso de prueba 4, prueba 2. El correo a enviarse puede tardar hasta días en llegar a destino.

  Corrección aplicable: Se podría utilizar otro método de envío de correos alterno a *mail* de PHP.

Pruebas con categoría Error:

- Caso de prueba 11, prueba 1. Al volver a la vista ListaCódigos después de eliminar o agregar un nuevo elemento, la información de la lista no se actualizaba, permitiendo elegir elementos que ya no existían e impidiendo elegir aquellos elementos recién añadidos.

  Corrección aplicable: Modificación de código de la clase ListaCodigos.java, para actualizar la información que contiene cada vez que sea necesario.

- Caso de prueba 8, prueba 1. Los bloques de anuncios de AdMob no se mostraban. El depurador mostró que el tamaño para los banners no era el adecuado.

Corrección aplicable: Modificación de código de todos los archivos XML que requirieran la impresión del bloque de anuncios, asignar mediante código un tamaño fijo para los banners, en este caso: 320X50 dp.

- Caso de prueba 13, prueba 1. La aplicación termina de ejecutarse al hacer peticiones al servidor, en versiones de Android 4 o superior.

  Corrección aplicable: Re compilación de la aplicación móvil utilizando la interfaz de programación (API) correspondiente a Android 4

- Caso de Prueba 4, prueba 1. Podían hacerse peticiones de recuperación de contraseña con usuarios no existentes.

  Corrección aplicable. Modificación de código de recupera.php para verificar la existencia del usuario antes de realizar el envío de correo.

Pruebas con categoría Éxito:

- Todas las demás pruebas.

Las correcciones fueron aplicadas, dejando la aplicación lista para el siguiente paso: funcionamiento y mantenimiento. El funcionamiento de la aplicación iniciará a partir de que ésta sea publicada en la tienda Google Play, donde cualquier persona interesada en el servicio que proporciona SiteTracker pueda descargarla en su dispositivo y utilizarla. El mantenimiento se realizará conforme a los errores reportados por los usuarios, tratando de corregir lo más pronto posible aquellos errores que impidan el uso adecuado de la aplicación.

Capítulo 6
Distribución

Antes de poder publicar una aplicación móvil en la tienda Google Play Store se deben considerar algunos requisitos, que no son complicados y no requieren gran inversión de tiempo. Antes de tener su aplicación móvil lista para subirla a la tienda considere lo siguiente:

- Todos los servicios de Google requieren una cuenta de correo, preferentemente de g-mail. En el capítulo anterior se requirió para dar de alta la aplicación móvil en el servicio de la red de publicidad móvil AdMob.
- Será necesario realizar el pago de la licencia de desarrollador de Android, esta tiene un costo de $25 USD (costo en dólares estadounidenses a octubre de 2013). El único método de pago que recibe Google por este concepto es tarjeta de crédito o débito. Preferentemente de crédito.
- El pago de la licencia de desarrollador se realiza a través de una cuenta de Google Wallet, deberá asociarse una cuenta de gmail a este servicio.
- Al publicar una aplicación móvil, no se olvide de sus futuros usuarios, reciba sus críticas y realice actualizaciones a la aplicación, corrigiendo aquellas fallas que le sean reportadas.
- Todas las aplicaciones móviles deben pasar por un proceso de firma digital. No se requiere una autoridad de certificación, usted puede realizar su propia firma digital.

Si es posible cumplir con lo anterior, en especial la parte del pago de licencia con tarjeta de crédito, entonces se puede comenzar con el proceso de distribución de la aplicación móvil.

6.1. Proceso de firma digital

El sistema Android requiere que todas las aplicaciones a instalarse sean firmadas digitalmente con un certificado cuya llave privada es propiedad del desarrollador de la aplicación. La firma es utilizada para identificar al autor de una aplicación y establecer relaciones confiables entre funcionalidades de distintas aplicaciones [25].

Lo más importante que se debe comprender sobre la firma digital es que todas las aplicaciones deben ser firmadas. La plataforma Android no instalará ninguna aplicación si no cuenta con una firma digital. Las herramientas de compilación generan una llave especial para modo de prueba, así que no se pueden publicar aplicaciones en modo de prueba o depuración. Es posible generar un certificado digital propio, no necesariamente de una autoridad en certificados digitales.

Almacene su llave privada en un lugar seguro para posteriores actualizaciones o aplicaciones que puedan interactuar con las que ya ha programado. Cuando se publica una actualización de aplicación Android compara su certificado con el de la aplicación ya

instalada y deberán coincidir para que proceda la actualización. Además la firma digital tiene un periodo de validez y Google Play requiere que ese periodo sea después del día 22 de Octubre de 2033.

En la figura 6.1 se observan capturas de pantalla del procedimiento de firma digital usando las ADT *(Android Developer Tools)* para Eclipse.

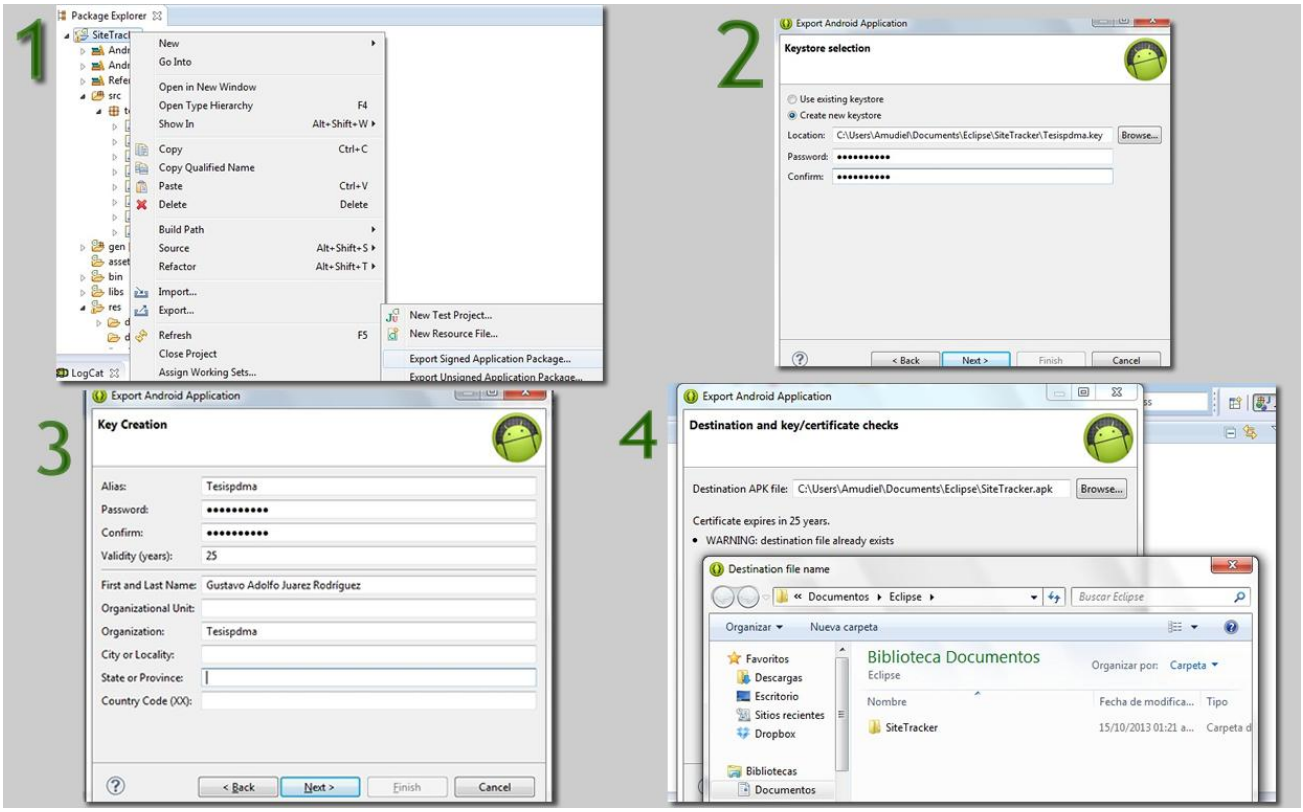

Figura 6.1. Procedimiento de firma digital usando las ADT (Android Developer Tools) para Eclipse.

En la carpeta del proyecto en el menú de opciones (click derecho del ratón) se encuentra la opción *AndroidTools* que despliega un submenú donde se encuentra la opción *Export Signed Application Package.* El asistente le brindará la opción de usar una llave privada existente o crear una nueva. Si se crea una nueva llave privada se deberá especificar la ubicación del archivo a generar, puede llevar cualquier nombre y cualquier extensión, posteriormente deberá asignar una nueva contraseña para esa llave. Después, deberá editar la información de la nueva llave privada, los atributos mínimos obligatorios son: alias, contraseña, validez en años, nombre completo y nombre de la organización. Se consideró la fecha mínima por Google Play para la validez de la aplicación, SiteTracker será válida hasta el año 2038. El último paso es generar la aplicación móvil con la firma digital, completamente lista para llevarla a la tienda de aplicaciones. El asistente le pedirá especificar la ruta del archivo de la aplicación, de extensión *.apk*.

Si ha desarrollado su aplicación móvil desde una plataforma distinta a Eclipse, será necesario que investigue el proceso de firma digital correspondiente. El cual, como puede notar, no es complicado ni laborioso.

6.2. Registro como desarrollador.

Para acceder a la consola de desarrollador de *Google Play* y poder subir aplicaciones a la tienda, es necesario pagar la cuota de registro. De no pagar la cuota, no es posible acceder a ninguna función de este servicio. Para realizar el registro como desarrollador se siguen cuatro pasos: acceder con una cuenta de Google, aceptar el acuerdo para desarrolladores, pagar la cuota de registro y rellenar la información de la cuenta, como nombre de desarrollador, correo electrónico de soporte para los usuarios, etc.

Figura 6.2. Interfaz de registro como desarrollador.

Después de acceder con una cuenta de Google, deberá aceptar el acuerdo de distribución para desarrolladores. Debe también consultar si en su país de residencia está permitido distribuir, vender aplicaciones y tener una cuenta de comerciante. En México estas tres opciones están permitidas. En cuanto al acuerdo de distribución para desarrolladores, básicamente trata sobre precios y pagos, los términos de uso de la tienda, los derechos que se otorgan a Google sobre sus aplicaciones, privacidad de la información y derechos de autor [26]. Es recomendable que lo lea y lo entienda, evite cometer acciones no permitidas para evitar una cancelación de su cuenta de desarrollador.

El siguiente paso es realizar el pago a través de una cuenta de Google Wallet, si no se cuenta con un medio de pago previamente registrado dentro de *Wallet* se deberá especificar uno nuevo. Si tampoco ha creado una cuenta de Google Wallet anteriormente, ésta se asociará inmediatamente a su cuenta de g-mail proporcionada en el primer paso. Deberá entonces facilitar la información de su tarjeta de crédito o débito (preferentemente de crédito). Ingresará el número de la tarjeta, la fecha de expiración y el código de seguridad. También será necesario que brinde la información sobre “dirección de facturación”, que consta de su domicilio, código postal, nombre completo y teléfono. Esta información deberá ser capturada tal cual como la proporciona el banco en su estado de cuenta. Si usted no recibe su estado de cuenta en su domicilio consulte con su banco esta información. Algún error en la captura de esta información podría ocasionar que el pago sea rechazado.

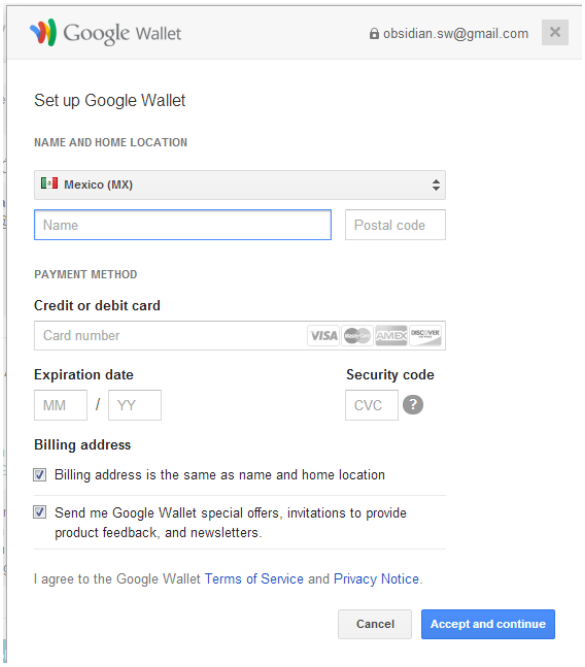

Figura 6.3. Formulario de Google Wallet para realizar el pago de registro.

Si los datos de pago son correctos se le concederá el acceso a la consola de desarrollador, sin antes proveer información para crear su perfil como desarrollador. Los datos obligatorios que deberá proporcionar son: Nombre de desarrollador, es el nombre que se mostrará a los usuarios debajo del nombre de su aplicación; dirección de correo electrónico, para ser contactado por sus usuarios y un número telefónico, únicamente para que Google pueda ponerse en contacto con usted en caso de que surja algún problema con el contenido que usted ha publicado.

6.3. Consola de desarrollador de Google Play

Una vez completo el registro, tendrá acceso inmediato a la consola de desarrollador, aunque no podrá subir aplicaciones a la tienda hasta que el proceso del pago sea confirmado, lo cual, dependiendo del banco, no puede tardar más de 1 semana.

La interfaz de esta *developer console* es amigable e intuitiva, es muy sencillo llevar una aplicación a la tienda siguiendo los pasos por los que el sistema lo conduce. Los pasos se pueden cumplir en cualquier orden y se puede guardar su progreso para continuar en otra ocasión. Se puede editar la información de una nueva aplicación móvil a publicar mientras el pago es confirmado. Las acciones que debe realizar como requisito para poder publicar su aplicación son las siguientes:

Subir el archivo APK. Elija la ruta del archivo APK de su aplicación móvil, recuerde que el archivo debió pasar previamente por el proceso de firma digital.

Ingrese la información de su producto. Deberá ingresar la información que se mostrará en la página de su aplicación dentro de la *Play Store* (figura 6.4). Se incluye el título de la aplicación, una descripción de las principales funciones que realiza, los cambios recientes (en caso de que sea una nueva versión), capturas de pantalla, un ícono de alta

resolución, la categoría a la que pertenece la *app,* los datos de contacto para ofrecer soporte y la política de privacidad si es que cuenta con alguna.

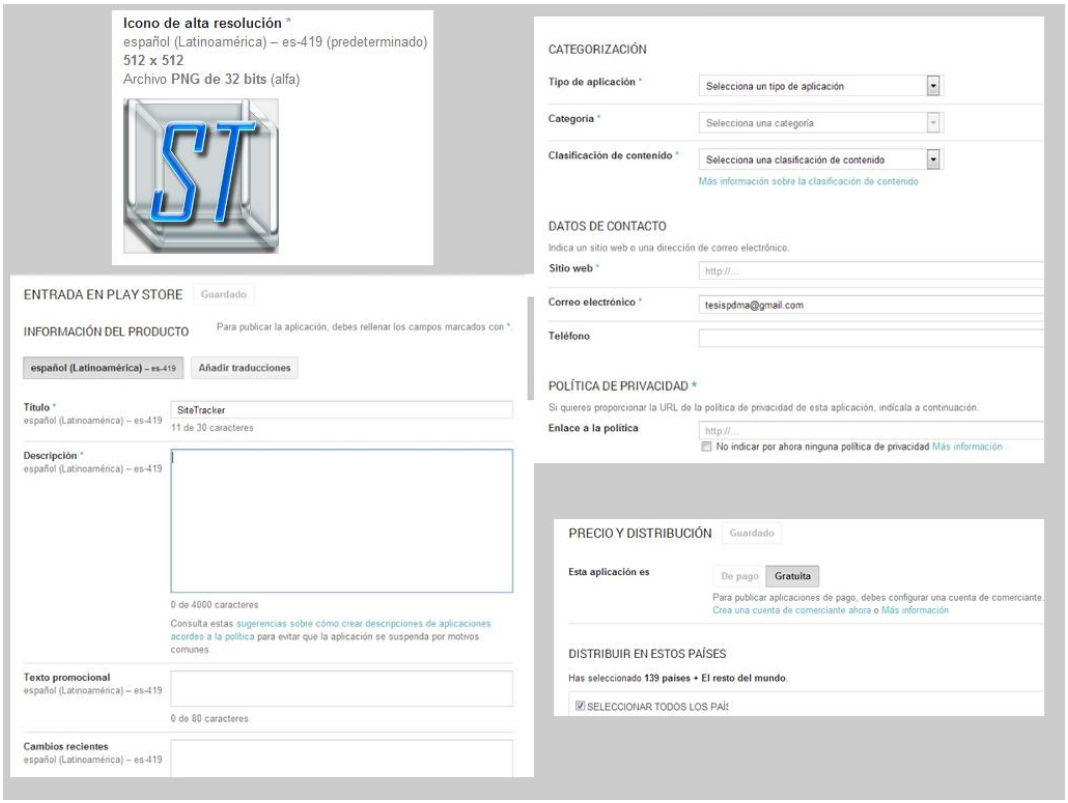

Figura 6.4. Se muestran algunos de los campos obligatorios que deben rellenarse para poder publicar la aplicación.

El icono de alta resolución es el que aparecerá en la página de su aplicación dentro de la tienda y en el listado de aplicaciones. También debe subir capturas de pantalla mostrando las vistas más importantes de su aplicación o mostrando su *app* en acción. Google recomienda subir capturas de pantalla para dispositivos de 7" y 10", de esta manera su aplicación también podrá ser recomendada para tabletas.

Defina el precio de su aplicación. En esta pestaña definirá el precio de la aplicación y los países en los que será distribuida. Para poder establecer un precio a la aplicación y venderla se debe configurar una cuenta de comerciante de Google Wallet. La aplicación de ejemplo SiteTracker será distribuida de forma gratuita con *In-App Advertising*. En este caso no será necesario configurar alguna cuenta de este tipo.

La consola de desarrollador le indicará cuando se hayan cumplido los requisitos necesarios para subir la aplicación a la tienda. Entonces, se podrá elegir la opción "Publicar esta aplicación" (figura 6.5). Compruebe que toda la información que ingresó sea la adecuada antes de publicar. En ese momento ha finalizado el proceso de publicación de su aplicación móvil.

Figura 6.5. Los indicadores en verde muestran que la aplicación esta lista para publicarse. Éste pasa a ser el último paso para la publicación de una *app* usando la consola de desarrollador de Google Play Store.

No olvide considerar que esto es apenas el principio, la aplicación móvil requerirá que se corrijan los errores reportados por los usuarios y que se les ofrezcan actualizaciones y nuevas características.

SiteTracker, nuestra aplicación de ejemplo, fue publicada siguiendo todos estos pasos, fue desarrollada con el apoyo de algunas metodologías de la Ingeniería de Software, se le realizaron pruebas y se corrigieron algunas de sus fallas, se le implementó la red publicitaria AdMob para ofrecerla de manera gratuita con *In-App Advertising.* Se firmó digitalmente y se creó una cuenta de Google Wallet para pagar la cuota de registro con tarjeta de crédito. Se proporcionó la información requerida en la consola de desarrollador de Google Play y su primera versión finalmente se publicó el 15 de octubre de 2013. SiteTracker llegó a su destino final, como se planteó en el objetivo de esta tesis, con la finalidad de dar a conocer el proceso de desarrollo de una aplicación móvil de manera independiente, sin grandes recursos económicos y haciendo uso de los conocimientos de la carrera de Ingeniería en Computación.

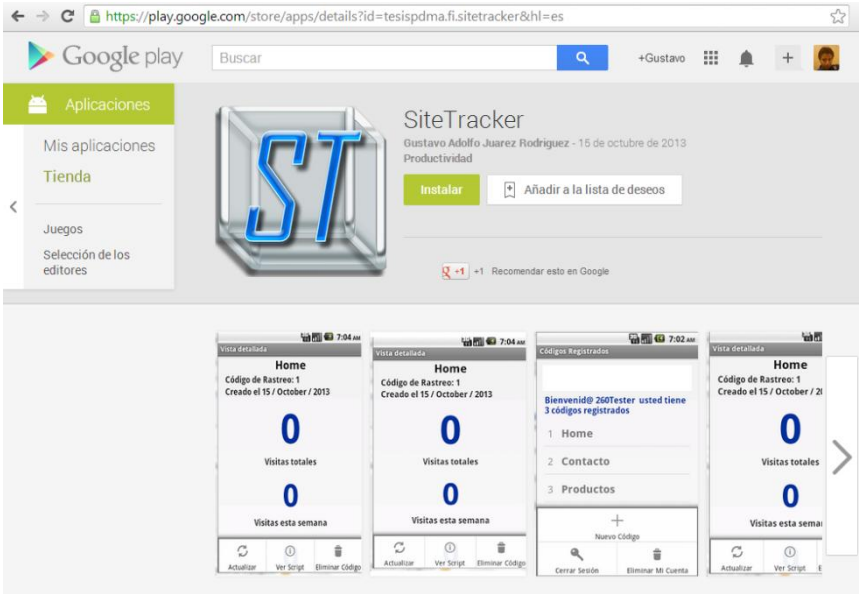

Figura 6.6. Página de SiteTracker en Google Play Store
https://play.google.com/store/apps/details?id=tesispdma.fi.sitetracker&hl=es

Capítulo 7
Conclusiones

La realización de este trabajo conduce a varias conclusiones relacionadas al desarrollo de aplicaciones móviles. El proceso de desarrollo puede llegar a ser tan extenso como se desee, dependiendo de la aplicación que se piensa crear. Como hemos visto existe todo tipo de aplicaciones, desde aquellas que tienen una navegación muy sencilla hasta aquellas otras que realizan muchas actividades y cuentan con una compleja navegación. La programación de la aplicación es también muy variada, existe una gran cantidad de herramientas, desde las que no requieren ningún tipo de conocimiento técnico hasta las más completas, como fue el caso de *Android Developer Tools,* con la cual se administran a detalle los recursos de la aplicación y en la que se utiliza un lenguaje orientado a objetos de alto nivel (Java), combinado con un lenguaje de tipo interprete (XML) que presenta los elementos gráficos en pantalla. Es sencillo encontrar un problema a resolver y llevar a cabo el desarrollo de una aplicación móvil que lo resuelva, en esta tesis se buscó la solución a un problema, que si bien no es muy común en toda la población de usuarios, puede llegar a ser muy práctica. Conocer la cantidad de visitas de una página web en todo momento desde su móvil es importante cuando se desea conocer el impacto que está teniendo el contenido de una página. Ayuda a la toma de decisiones y puede llegar a cambiar el modelo de negocio de una empresa. Se optó por un proceso de desarrollo que involucrara algunos de los conocimientos de la carrera de Ingeniería en Computación con el fin de demostrar el tipo de aplicaciones que pueden llegar desarrollarse. No se trata de una calculadora o un calendario, se trata de un servicio completo que incluye interacción con un servidor y una base de datos, es la implementación de un modelo cliente/servidor en tres capas.

Desafortunadamente la aplicación fue publicada en modo de prueba debido a las limitantes que ofrecen los servicios de *hosting* gratuitos. Aunque son de gran ayuda, sus limitantes impiden ofrecer un servicio de alta calidad. El tiempo en que el servidor web se encuentra fuera de servicio es muy alto, es mucho mayor al 0.1% que prometen a la hora de contratar estos servicios gratuitos, además su cuota de transferencia mensual es aparentemente alta, pero con una gran cantidad de tráfico llega a su límite en relativamente poco tiempo, dejando el servicio inactivo por varios días. La configuración del servidor es también muy limitada, así como la configuración de los archivos para ejecutar scripts en PHP. El envío de correos por medio de PHP es una tarea casi imposible. Algo similar sucede con la base de datos, no es posible programar *triggers* ni modificar los *constraints*. Los procedimientos almacenados y la eliminación en cascada son un sueño que sólo es posible con un servidor de bases de datos propio o de pago. Aún con estas limitantes, este tipo de servicios permitieron la creación de la aplicación móvil de ejemplo, 000webhost permitió que SiteTracker pudiera ser llevada a la tienda y que llevara a cabo su función, contar visitas.

Cuando una persona está iniciando en el desarrollo independiente, lo más probable es que no tenga intención de invertir grandes recursos económicos. La propuesta que se desarrolla en este documento, es el ejemplo perfecto de esto, se utilizó un servicio de *hosting* gratuito, que permitía a la aplicación realizar sus funciones fundamentales. Pensando en este motivo se decidió realizar la *app* bajo la plataforma Android, que cuenta con la cuota de registro más económica y que además es vitalicia. Cualquier persona puede entonces, acercarse e intentar subir una *app* de cualquier tipo a la tienda, si requiere almacenamiento en base de datos o procesos externos realizados por un servidor web, no existe limitante. A final de cuentas este puede ser el principio de algo más grande, si la aplicación tiene éxito y está siendo demandada por una gran cantidad de usuarios puede salir de un modo de prueba y asignársele un precio, o bien pueden realizarse ventas dentro de la aplicación. Si de esta forma llega a ser rentable, se puede optar por migrar los servicios de la aplicación a un servidor propio o de pago que elimine las limitantes de un *hosting* gratuito, además de que se puede pensar en llevar la aplicación a otras plataformas como iOS y Windows Phone. En el mejor de los casos este Desarrollo Independiente de aplicaciones puede llevar a una persona con una buena idea a ser rentable por sí mismo, sin depender de ninguna empresa.

Sin importar el costo de la cuota de registro, la plataforma Android es una muy buena opción para desarrollar. Para marzo de 2013 existían 750 millones de dispositivos con Android, la aplicación puede tener un gran alcance. Es muy fácil desarrollar una *app* usando *Android Developer Tools*, la documentación oficial de las clases Android en Java es de fácil acceso y existen foros de desarrolladores en internet que brindan demasiada información de cómo utilizar estos recursos de programación, incluso sin tener grandes conocimientos de programación en este lenguaje es posible programar una *app* como el ejemplo de esta tesis.

La publicación de aplicaciones es también una tarea fácil, si se distribuye un trabajo de manera gratuita no es necesaria la creación de una cuenta de comerciante de Google Wallet, que requiere cuestiones legales, bancarias y de impuestos. La única dificultad de publicar una *app* a través de Google Play es el pago de registro, el pago con tarjeta de crédito podría no estar al alcance de cualquier persona que intenta publicar su primera *app*. Wallet permite pagar con cualquier tarjeta de débito o crédito, incluso si no es propia, así que se puede pedir una prestada con algún conocido. El pago de registro es único y con el mismo se pueden publicar un sinfín de aplicaciones. El procedimiento de publicación es sencillo y rápido, la consola de desarrollador es intuitiva y amigable, lo que permite publicar en cuestión de minutos.

La importancia de este seguimiento al desarrollo de una aplicación móvil para Android reside en la manera con la que se facilita y se da a conocer el procedimiento a seguir para llevar una idea a la tienda de aplicaciones, de manera independiente y con un posible retorno de inversión por concepto de publicidad. El mercado de aplicaciones móviles es un mercado que no puede ser desaprovechado, los móviles se están convirtiendo en el principal medio de acceso a internet y año con año las activaciones de dispositivos y descargas de aplicaciones crecen exponencialmente. La industria del software en México está creciendo y se espera que se convierta en una oportunidad de mercado para jóvenes emprendedores. El objetivo es convertir a México en una nación que exporte soluciones en tecnología, en especial en cuestión de movilidad, esto es trabajo de todos.

Esta tesis ayuda a aquellas personas que tengan en mente una aplicación móvil y que crean no tener todos los elementos para desarrollarla. Con mi trabajo les digo y les muestro que se equivocan, ser emprendedor en este rubro de la tecnología es más fácil de lo que parece, se requieren pocos recursos económicos y de tiempo, los resultados finales son de gran provecho. Se puede aprender mucho y adentrarse al mundo de la movilidad.

Bibliografía

[1] Natalia Arroyo Vázquez. Información en el móvil. Editorial UOC, 2011.

[2] Lara Lazo. App Store bate récords: 50.000 millones de descargas. http://computerhoy.com/noticias/apps/app-store-bate-records-50000-millones-descargas-3767, 5 2013

[3] Ingrid Lunden. Larry Page Says There Have Now Been 750M Android Activations. http://techcrunch.com/2013/03/13/larry-page-says-there-have-now-been-750m-android-activations/, 3 2013

[4] Accenture. Mobile Web Watch 2012. http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Mobile-Web-Watch-Internet-Usage-Survey-2012.pdf.

[5] Angel Álvarez. Desarrolladores mexicanos de apps, pocos y caros. http://www.informationweek.com.mx/movilidad/desarrolladores-mexicanos-de-apps-pocos-y-caros/, 9 2011

[6] Beatriz Arias. México podría desarrollar más aplicaciones móviles. http://www.informationweek.com.mx/movilidad/mexico-aplicaciones-moviles/, 7 2011

[7] Fabiola Naranjo. Aplicaciones móviles, un nicho para pymes. http://elempresario.mx/actualidad/aplicaciones-moviles-nicho-pymes, 7 2011

[8] Marisela Delgado. Software, sector con gran potencial. http://elempresario.mx/actualidad/software-sector-gran-potencial, 6 2013

[9] John Koetsier. Windows Phone jumps to third in global smartphone market share — and could be second faster than you think. http://venturebeat.com/2013/05/16/windows-phone-jumps-to-third-in-global-smartphone-market-share-and-could-be-second-faster-than-you-think/, 5 2013

[10] Carlos López. Modelos de negocio en internet. Generación de ingresos a través de la publicidad. http://www.gestiopolis.com/canales/emprendedora/articulos/no%205/modelosdenegocioeninternet1.htm, 1 2001

[11] Página de ayuda y soporte de Google AdMob. https://support.google.com/AdMob/

[12] Página de desarrolladores Android, sección de diseño. http://developer.android.com/design/index.html

[13] Paul E. Renaud. Introduction to client/server systems: a practical guide for systems professionals. Wiley, 1993.

[14] Página de appsmakerstore para desarrollo de aplicaciones móviles.
http://appsmakerstore.com/

[15] Ian Sommerville. Ingeniería del software, Pearson Education, 2005

[16] IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998.
http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf

[17] Roger S. Pressman. Ingeniería del software un enfoque práctico, McGrawHill, 2002

[18] Martin L. Shoemaker. UML Applied, Apress, 2004

[19] Pierre-Alain Muller. Instant UML, Springer-Verlag New York Incorporated, 1997

[20] Página de documentación oficial PHP. http://php.net/docs.php

[21] Página de documentación oficial MySQL.
http://dev.mysql.com/doc/refman/5.1/en/index.html

[22] Página de herramientas de desarrollo Android.
http://developer.android.com/tools/index.html

[23] Descarga de AdMob SDK. https://developers.google.com/mobile-ads-sdk/download

[24] PayPal, servicio de pagos y cobros por internet.
https://www.paypal.com/mx/webapps/mpp/home

[25] Página de firma de aplicaciones Android. http://developer.android.com/tools/publishing/app-signing.html

[26] Acuerdo de distribución para desarrolladores Google Play Store.
https://play.google.com/about/developer-distribution-agreement.html