**UNIVERSIDAD NACIONAL DEL ESTE**

**FACULTAD POLITÉCNICA**



**ANÁLISIS DE SISTEMAS**

**TRABAJO FINAL DE GRADO**

**UNA APLICACIÓN ANDROID DE GEOLOCALIZACIÓN DE HOTELES Y RESTAURANTES DE CIUDAD DEL ESTE UTILIZANDO GOOGLE MAPS**

**Autores:**

**Oscar Leonardo Duarte Fernández**

**Rocio Elisa Torales Frutos**

**Tutor: Lic. Marcos Adrián Jara**

Febrero, 2014

Ciudad del Este – Paraguay

**Agradecimientos**

La finalización de este Trabajo final de grado es un orgullo para nosotros, representa la culminación de una etapa muy importante de nuestras vidas, que es la de tornarnos excelentes profesionales.

La finalización de dicho trabajo no es posible sin la ayuda de todas las personas que nos rodean, ya sea de forma directa o indirecta para que lleguemos a concluirla. Por, ello agradecemos en primer lugar a Dios y la Virgen, en segundo lugar a nuestra familia, que siempre han velado por nosotros y han sabido inculcarnos los valores que profesamos. También a nuestro tutor Marcos Jara por su orientación y disponibilidad que siempre nos ha dado.

Gracias a todos los profesores que han influenciado en nuestra formación académica y a nuestra querida Facultad Politécnica, una prestigiosa casa de Estudio.

Y por último gracias a todos los compañeros que no han dudado en ningún momento en ayudarnos en lo que fuere posible.

Gracias a todos por haber estado con nosotros en el transcurso de nuestra carrera.

**RESUMEN**

En el presente proyecto se propone el desarrollo de un aplicativo para dispositivos con sistema operativo Android[[1]](#footnote-1) que permita la identificación geolocalizada de Hoteles y Restaurantes de Ciudad del Este, utilizando la API Google Maps.

Así, el sistema de geolocalización de Hoteles y Restaurantes propuesto, permitirá la visualización de la ubicación geográfica de Hoteles y Restaurantes de Ciudad del Este, clasificados por categorías posibilitando la aplicación de filtros de búsqueda y otras características integradas en la API del Google Mapas.

El sistema atenderá a dos clases de usuarios, por un lado, usuarios dueños de establecimientos o administradores del sistema que deseen posicionar locales comerciales en la base de datos y por otro lado, los usuarios del sistema que consultan por informaciones de Georeferenciamiento[[2]](#footnote-2), y visualizan en el Google Maps, el resultado de estas consultas. De esta forma, la API de GoogleMaps permitirá abstraer la capa de presentación, procurándonos apenas en la lógica del negocio.

En líneas generales, la aplicación ofrecerá un catálogo de Hoteles y Restaurantes de la región, a usuarios de dispositivos Android, permitiendo la rápida ubicación de los mismos.

**Palabras Claves**: Aplicación móvil, Geolocalización, Android. Google Maps

Índice de Contenido

[CAPÍTULO I: INTRODUCCIÓN. 8](#_Toc382153759)

[1.1 MOTIVACIÓN. 9](#_Toc382153760)

[1.2 DELIMITACIÓN DEL TEMA. 9](#_Toc382153761)

[1.3 OBJETIVOS OPERACIONALES. 9](#_Toc382153762)

[1.3.1 OBJETIVOS GENERALES 9](#_Toc382153763)

[1.3.2 OBJETIVOS ESPECÍFICOS 9](#_Toc382153764)

[1.4 ORGANIZACIÓN DEL TRABAJO. 10](#_Toc382153765)

[CAPÍTULO II: CONCEPTOS GENERALES 12](#_Toc382153766)

[2.1 Dispositivos Móviles 12](#_Toc382153767)

[2.2 Clasificación de los Dispositivos móviles 12](#_Toc382153768)

[2.3 Sistemas Operativos para dispositivos móviles 13](#_Toc382153769)

[2.3.1 Symbian 13](#_Toc382153770)

[2.3.2 Windows Mobile 14](#_Toc382153771)

[2.3.3 Android 16](#_Toc382153772)

[2.3.4 Blackberry OS 16](#_Toc382153773)

[2.3.5 iOS 17](#_Toc382153774)

[2.4 Geolocalización 18](#_Toc382153775)

[2.4.1 GPS 18](#_Toc382153776)

[2.4.2 A-GPS 19](#_Toc382153777)

[2.5 Aplicaciones tecnológicas 20](#_Toc382153778)

[2.5.1 Sistemas de localización de dispositivos móviles 20](#_Toc382153779)

[2.6 Historia de Google Inc. 21](#_Toc382153780)

[2.6.1 Productos Google 22](#_Toc382153781)

[2.6.2 Google en Google Maps 24](#_Toc382153782)

[2.6.3 Impactó en la sociedad 25](#_Toc382153783)

[2.6.4 Google Maps API Android 25](#_Toc382153784)

[Capitulo III: Plataforma Android 28](#_Toc382153785)

[3.1 Arquitectura 28](#_Toc382153786)

[3.2 Componentes de una aplicación Android 32](#_Toc382153787)

[3.2.1 Activity 32](#_Toc382153788)

[3.2.2 Broadcast Intent Receiver 32](#_Toc382153789)

[3.2.3 Service 33](#_Toc382153790)

[3.2.4 Content Provider 33](#_Toc382153791)

[3.3 Ciclo de vida de una aplicación Android 33](#_Toc382153792)

[3.4 Gestión de la información en una aplicación Android 34](#_Toc382153793)

[3.4.1 Preferencias de usuarios 34](#_Toc382153794)

[3.4.2 Ficheros 35](#_Toc382153795)

[3.4.3 Bases de datos 35](#_Toc382153796)

[3.4.4 Acceso por red 35](#_Toc382153797)

[3.4.5 Content Provider 35](#_Toc382153798)

[3.5 Instalación de Eclipse con el SDK de Android 36](#_Toc382153799)

[3.5.1 Descargar el SDK de Android 36](#_Toc382153800)

[3.5.2 Descargar Eclipse IDE 36](#_Toc382153801)

[3.5.3 Instalar el plug-in de Android 36](#_Toc382153802)

[3.5.4 Referenciar el SDK 37](#_Toc382153803)

[3.5.5 Actualizaciones del plug-in 37](#_Toc382153804)

[3.5.6 Creación de una Aplicación de Ejemplo 38](#_Toc382153805)

[3.5.7 Añadir interfaz de usuario 38](#_Toc382153806)

[3.5.8 Ejecutar la aplicación 40](#_Toc382153807)

[Capitulo IV: Desarrollo del Proyecto 43](#_Toc382153808)

[TuristaApp: Aplicativo de Geolocalización de Hoteles y Restaurantes 43](#_Toc382153809)

[4.1 Análisis y Diseño de la Aplicación 43](#_Toc382153810)

[4.2 Caso de Uso 44](#_Toc382153811)

[4.3 Intercambio de información 46](#_Toc382153812)

[4.4 Servidor 49](#_Toc382153813)

[4.5 Arquitectura 53](#_Toc382153814)

[4.6 Desarrollo de la Aplicación Móvil 54](#_Toc382153815)

[4.7 Estructura del Proyecto 54](#_Toc382153816)

[4.8 Conexión de Datos con el Servicio 56](#_Toc382153817)

[4.9 Acceso a Google Maps 56](#_Toc382153818)

[4.9.1 Obtención de la API Key 56](#_Toc382153819)

[4.9.2 Mostrar un mapa 58](#_Toc382153820)

[4.10 Menú de Opciones y Navegación. 59](#_Toc382153821)

[4.11 Integración con funcionalidades del Teléfono 61](#_Toc382153822)

[4.12 Dificultades en el desarrollo del Aplicativo 62](#_Toc382153823)

[CAPÍTULO V: CONCLUSIONES Y SUGERENCIAS 64](#_Toc382153824)

[5.1 Conclusiones del Sistema Android 64](#_Toc382153825)

[Ventaja del sistema Android 64](#_Toc382153826)

[Desventajas del sistema Android 64](#_Toc382153827)

[5.2 Comparaciones con servicios de Búsquedas de hoteles y restaurantes de Ciudad del Este. 65](#_Toc382153828)

[Buscador Google. 65](#_Toc382153829)

[Página especializada de guía: www.guíadeleste.com 65](#_Toc382153830)

[Servicio Guía telefónica 112 Copaco. 65](#_Toc382153831)

[5.3 Conclusiones del Aplicativo Implementado en el proyecto. 65](#_Toc382153832)

[5.4 Beneficios obtenidos 66](#_Toc382153833)

[5.5 Sugerencias para futuros trabajos. 66](#_Toc382153834)

[REFERENCIAS 68](#_Toc382153835)

[Anexos 72](#_Toc382153836)

[ANEXO 1: Tendencias en dispositivos móviles para el año 2014. 72](#_Toc382153837)

[Anexo 2: Términos 73](#_Toc382153838)

**CAPÍTULO I**

**INTRODUCCIÓN.**

***La felicidad es interior, no exterior; por lo tanto, no depende de lo que tenemos, sino de lo que somos****. (Pablo Neruda)*

# CAPÍTULO I: INTRODUCCIÓN.

El crecimiento del mercado de los teléfonos inteligentes ha presentado un enorme salto en cuanto a uso y capacidad de procesamiento de los aparatos telefónicos móviles, esto pone de manifiesto su aplicación como canal de comercialización de productos y servicios.

El mercado de los teléfonos móviles de nueva generación muestra clara tendencias de crecimiento entre los consumidores, sobre todo con el uso de estos dispositivos inteligentes a la hora de consultar información. Esto implica una clara utilidad orientada a satisfacer la necesidad de información rápida en cualquier lugar, además de ofrecer nuevas experiencias y recursos para los consumidores en todos sus procesos.

Prueba de ello es sin duda el rápido crecimiento de las tiendas de aplicaciones móviles que aumentan a medida que madura este mercado y como consecuencia, muchos desarrolladores de aplicaciones tendrán la necesidad muy cercana de desarrollar aplicaciones móviles.

En el campo de la información georeferenciada, la utilización de API de mapas de uso gratuito como Google Maps y la capacidad de conectividad de los teléfonos inteligentes permite el desarrollo de aplicaciones que permita rápidamente ubicar un punto en la superficie terrestre, conociendo los parámetros de longitud[[3]](#footnote-3) y latitud[[4]](#footnote-4)

El SPG o GPS (Global Positioning System: sistema de posicionamiento global), el cual es un [sistema global de navegación por satélite](http://es.wikipedia.org/wiki/Sistema_global_de_navegaci%C3%B3n_por_sat%C3%A9lite)(GNSS) permite determinar en todo el mundo la posición de un objeto, una persona o un vehículo con una precisión exacta. Los receptores GPS Sistema de Posicionamiento Global, son dispositivos electrónicos capaces de determinar su posición en la superficie terrestre con bastante exactitud valiéndose de varios satélites espaciales y cuyo bajo costo de adquisición ha permitido su rápida popularización como sistema de negación en vehículos e incluso teléfonos celulares.

Así mismo actualmente se puede contar con potentes teléfonos móviles con alta conectividad de datos, lo cual permite disponer de la información en cualquier lugar. En ese sentido se dispone de la Plataforma de Desarrollo para Aplicaciones de Teléfonos Móviles ANDROID, el cual puede ser fácilmente integrado al Servicio Google Maps, a través de la API[[5]](#footnote-5) Google Maps For Android.

## 1.1 MOTIVACIÓN.

El desarrollo de la electrónica, la informática y las redes de comunicación han permitido actualmente contar con potentes teléfonos móviles de alta conectividad, además la integración con otros servicios permite disponer de la información necesaria en cualquier lugar e incluso ubicarla geográficamente.

Para desarrollar una aplicación en la plataforma Android es necesario entender el funcionamiento de la API Google Maps, las características de una aplicación Android, así como conocer las buenas prácticas para la creación de una aplicación móvil.

## 1.2 DELIMITACIÓN DEL TEMA.

Se propone desarrollar un aplicativo que permita obtener los datos de hoteles y restaurantes de Ciudad del Este previamente registrados, y disponibilizarlos como servicio para ser visualizados en el Google Maps desde un teléfono Android.

Los usuarios y visitantes de los grandes centros urbanos, como Ciudad del Este pueden beneficiarse con una aplicación que les permita obtener la información precisa sobre la ubicación de Hoteles y Restaurantes de la ciudad a través de sus teléfonos celulares con sistema operativo Android.

También los propietarios de hoteles y restaurantes podrán beneficiarse con la posibilidad de dar a conocer su ubicación y servicios a través de una funcionalidad de añadir ubicación.

## 1.3 OBJETIVOS OPERACIONALES.

### 1.3.1 OBJETIVOS GENERALES

* Desarrollar una aplicación en la plataforma Android que permita interactuar con la API de Google Maps.
* Ofrecer el servicio de catastro y consulta de Hoteles y Restaurantes en Ciudad del Este.

### 1.3.2 OBJETIVOS ESPECÍFICOS

* Estudiar el desarrollo de aplicativos en la plataforma Android.
* Estudiar e Implementar la API Google Maps para la ubicación de puntos geográficos en la plataforma Android.
* Implementar un servidor web para actualización de los dispositivos clientes.
* Documentar todas las etapas de realización.

## 1.4 ORGANIZACIÓN DEL TRABAJO.

El desarrollo del contenido de este trabajo se extiende a lo largo de cinco capítulos, más los anexos.

**El primer capítulo**, introduce los temas tratados, como ser la motivación, la delimitación del tema, los objetivos y así como la visión general del contenido de este trabajo.

**El segundo capítulo,** expone las bases teóricas generales de algunos de los aspectos relacionados con el proyecto: como los dispositivos móviles, características, clasificación y los sistemas operativos que utilizan, también se expone un poco el concepto de Geolocalización, y sus aplicaciones tecnológicas, y finalizando un breve repaso respecto a Google y el servicio de Google Maps.

**El tercer capítulo**, explica al lector un estudio sobre la plataforma móvil Android, características, arquitectura, componentes de una aplicación Android y ambiente de desarrollo.

**En el cuarto capítulo,** se expone el desarrollo del aplicativo, el diseño de la aplicación desde las especificaciones de las funcionalidades, la metodología de desarrollo hasta la implementación de la misma.

**En el quinto capítulo,** se presentan los resultados obtenidos tras la finalización del proyecto, además de exponer las sugerencias para futuras ampliaciones del trabajo.

**CAPÍTULO II**

**CONCEPTOS GENERALES.**

***Cuando uno es joven los pensamientos se vuelven amor, con la edad el amor se vuelve pensamientos****. (Einstein)*

# CAPÍTULO II: CONCEPTOS GENERALES

## 2.1 Dispositivos Móviles

Se podría denominar dispositivo móvil a todo aparato electrónico que cumple unas características muy básicas.

* Es de reducido tamaño, haciéndolo fácil de transportar.
* Cuenta con una cierta capacidad de computación y almacenamiento de datos.
* Incorpora elementos de E/S básicos (por lo general, pantalla y/o algún tipo de teclado).

Más allá de estas características comunes, los dispositivos móviles forman en la actualidad un grupo sumamente heterogéneo y pueden incorporar casi cualquier componente de hardware y software que amplía y diversifica su función inicial. El más frecuente sin duda es la conexión telefónica (incluyendo servicios como el envío de SMS, MMS, y acceso WAP) o la conexión a internet.

## 2.2 Clasificación de los Dispositivos móviles

Al igual que ocurre a la hora de definir qué es un dispositivo móvil, la clasificación que se pueda hacer de estos aparatos está sujeta a diferentes valoraciones y a veces no existe un acuerdo amplío para ubicar un dispositivo móvil en una determinada familia.

Los dispositivos móviles pueden ser clasificados en los siguientes grupos:

**Dispositivo de comunicación**

Un dispositivo de comunicación es aquel dispositivo móvil cuyo cometido principal es ofrecer una infraestructura de comunicación, principalmente telefónica .Estos dispositivos ofrecen además servicios como el envío de mensajes SMS, y MMS, o acceso WAP. En esta categoría se incluiría el tradicional teléfono móvil, la BlackBerry y el Smartphone.

**Dispositivo de computación**

Los dispositivos de computación son aquellos dispositivos móviles que ofrecen mayores capacidades de procesamiento de datos y cuentan con una pantalla y teclado más cercanos a un ordenador de sobremesa. Dentro de este grupo encontramos los PDA. Por otro lado, dispositivo de computación también es un ordenador portátil o laptop. Las calculadoras gráficas pueden ser igualmente incluidas en este grupo.

**Reproductor multimedia**

Un reproductor multimedia es aquel dispositivo móvil que ha sido específicamente diseñado para proporcionar al usuario la reproducción de uno o varios formatos de datos digitales, ya sea audio, vídeo o imágenes. Dentro de estos dispositivos encontramos reproductores de MP3, los DVD portátiles, los eBooks, y en los últimos años los reproductores multimedia de la popular familia iPod de Apple, que ofrecen tanto audio y vídeo. Estos dispositivos son con frecuencias los de más reducido tamaño y, junto a los teléfonos móviles y smartphones, los más extendidos.

**Grabador multimedia**

Dentro de los dispositivos móviles, un grabador multimedia es aquel dispositivo que posibilita la grabación de datos en un determinado formato digital, principalmente de audio y vídeo. En esta categoría se hallan las cámaras fotográficas o las cámaras de vídeo digital

**Consola portátil**

Una consola portátil es un dispositivo móvil cuya única función es la de proporcionar al usuario una plataforma de juego. Algunos ejemplos de esta categoría son la Nintendo DS de Nintendo, o la PSP de Sony.

## 2.3 Sistemas Operativos para dispositivos móviles

El sistema operativo destinado a correr en un dispositivo móvil necesita ser fiable y tener una gran estabilidad, ya que incidencias habituales y toleradas en ordenadores personales como reinicios o caídas no tienen cabida en un dispositivo de estas características.

Estos sistemas han de estar perfectamente testeados y libres de errores antes de incorporarse definitivamente a la línea de producción.

En la actualidad, existen varios sistemas operativos para toda la gama de dispositivos móviles.

### 2.3.1 Symbian

Symbian es un sistema operativo para dispositivos móviles desarrollados por Psion, Nokia, Motorola y Ericsson. El principal objetivo de estas compañías era el de crear un nuevo y compartido sistema operativo que estuviera perfectamente adaptado a los teléfonos móviles del momento, y fuese además capaz de competir con Palm OS y Windows Mobile. La primera versión de Symbian, basada en el sistema EPOC de Windows, se lanzó en 1998.

Symbian contempla cinco tipos de ediciones o series del sistema operativo según las características del dispositivo móvil. La principal diferencia entre ediciones no radica tanto en el núcleo del sistema operativo como en la interfaz gráfica utilizada:

**Serie60**. El más popular de todos debido fundamentalmente a que el gigante Nokia, ha hecho de Symbian y de su versión Serie60 el núcleo de casi todos los modelos de smartphones. Los dispositivos con Serie60 tiene una pantalla pequeña y un teclado del tipo 0-9#.

**Serie80**. Esta edición también usada por Nokia, está más orientada a dispositivos que tienen pantalla táctil y permiten multitarea, pudiendo tener varias aplicaciones abiertas simultáneamente.

**Serie90**. Muy similar a la Serie80, sólo qué éstos dispositivos tienen una pantalla más grande y llevan incorporados sensores táctiles más desarrollados. Utilizan teclados virtuales, reconocimiento de trazos o teclados acoplables mediante, por ejemplo, Bluetooth.

**UIQ**. La interfaz de esta edición de Symbian se encuentra muy influenciada por Palm OS. Implementan una especie de multitarea virtual, dando al usuario la falsa sensación de poder realizar varias acciones simultáneas. Es utilizado en algunos modelos de Sony Ericsson y Motorola.

**MOAP.** Esta edición se da únicamente en Japón, principalmente el fabricante FOMA.



**Figura 1.** **Celular Nokia con S.O. Symbian**

### 2.3.2 Windows Mobile

Windows Mobile es un sistema operativo diseñado por Microsoft y orientado a una gran variedad de dispositivos móviles. En realidad, Windows Mobile representa una particularización de otro gran sistema de Microsoft llamado Windows CE[[6]](#footnote-6).

Las características principales con la que cuenta Windows CE son las siguientes:

* Es un sistema modular, lo que permite que cada fabricante pueda seleccionar pares que le benefician más para su dispositivo.
* Contempla una considerable gama de recursos hardware: teclado, cámara, pantalla táctil, etc.
* Tiene un tamaño en memoria relativamente pequeño y bajo coste computacional.
* Es capaz de trabajar con distintas familias de procesadores de 32 bits.
* Permite interactuar con otros dispositivos móviles.



**Figura 2. Celulares con S.O. Windows Mobile**

La arquitectura básica de Windows CE es explicada a continuación.

* **OEM Layer**: es la capa situada entre el hardware del dispositivo y el kernel. Permite a los fabricantes desarrollar sus propios drivers y funciones de control de los elementos de hardware.
* **Operating System Layer**: incluye el kernel[[7]](#footnote-7) como elemento principal y el conjunto de API Win32 necesarias. En esta capa se sitúan las bibliotecas de comunicaciones, el gestor gráfico, gestor de ficheros y registros, así como otros componentes opcionales.
* **Aplication Layer**: donde residen las aplicaciones por defecto de Windows CE y las aplicaciones del usuario.

Windows Mobile está a su vez dividido en tres ediciones**:**

* **Windows Mobile Classic**: antes llamado Pocket PC, orientado a dispositivos del tipo PDA sin funciones de comunicación telefónica.
* **Windows Mobile Standard**: hasta ahora conocido como Smartphone, esta edición está destinado a dispositivos del tipo Smartphone sin pantalla táctil.
* **Windows Mobile Professional**: anteriormente Pocket PC Phone Edition, para cubrir aquellas PDA u ordenadores de bolsillo con capacidad de comunicación telefónica.



**Figura 3. Windows Mobile 6**

### 2.3.3 Android

**Android** es un sistema operativo para teléfonos celulares y tabletas, propiedad de la imponente empresa tecnológica Google. Android es un sistema operativo móvil basado en Linux y Java que ha sido liberado bajo la licencia Apache versión 2.

Dicho sistema ha ganado en los últimos tiempos un gran espacio en la telecomunicación, tanto así, que empresas como Nokia, Motorola, HTC, Samsung, han apostado por integrar el sistema en sus equipos, garantizando una comunicación versátil, eficiente y eficaz.

El sistema busca, nuevamente, un modelo estandarizado de programación que simplifique las labores de creación de aplicaciones móviles y normalice las herramientas en el campo de la telefonía móvil. Al igual que ocurriera con Symbian, lo que se busca es que los programadores sólo tengan que desarrollar sus creaciones una única vez y así, ésta sea compatible con diferentes terminales. Google promete una plataforma de desarrollo gratuita, flexible, económica en el desarrollo de aplicaciones y simple, diferenciada de los estándares que ofrecen Microsoft o Symbian.



**Figura 4. Sistema Operativo Android**

### 2.3.4 Blackberry OS

Un sistema operativo móvil o SO móvil es un [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo) que controla un [dispositivo móvil](http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil) al igual que los [PCs](http://es.wikipedia.org/wiki/Computadora_personal) utilizan [Windows](http://es.wikipedia.org/wiki/Windows) o [Linux](http://es.wikipedia.org/wiki/Linux) entre otros. Sin embargo, los sistemas operativos móviles son mucho más simples y están más orientados a la [conectividad inalámbrica](http://es.wikipedia.org/wiki/Conectividad_inal%C3%A1mbrica), los formatos multimedia para móviles y las diferentes maneras de introducir información en ellos.

BlackBerry tiene fuerte presencia en el ámbito empresarial, en especial por sus servicios para correo y teclado QWERTY.



**Figura 5. Celular con S.O. Blackberry OS**

BlackBerry aparece en el mercado justo en el momento en que comenzaba a demandarse un sistema operativo que permitiera utilizar de una forma fácil, cómoda y rápida los servicios de correo electrónico. Hoy en día es también proveedor de servicios de correo electrónico a dispositivos que no son BlackBerry, gracias al programa BlackBerry Connect. Así, en líneas generales, en un dispositivo BlackBerry es posible redactar, enviar y recibir todo tipo de mensajes de correo electrónico, al igual que en el programa que se utiliza en un ordenador. Además, es posible realizar y contestar a las llamadas que se emitan a través de la red de telefonía móvil, lo que permite sustituir el teléfono móvil. También, como evolución lógica, los dispositivos de este fabricante permiten la navegación por internet en páginas HTML o WAP y tienen la capacidad de enviar o recibir mensajes SMS.

### 2.3.5 iOS

**iOS** es un [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo) móvil de la empresa [Apple Inc.](http://es.wikipedia.org/wiki/Apple_Inc.) Originalmente desarrollado para el [iPhone](http://es.wikipedia.org/wiki/IPhone) (iPhone OS), siendo después usado en dispositivos como el [iPod Touch](http://es.wikipedia.org/wiki/IPod_Touch), [iPad](http://es.wikipedia.org/wiki/IPad) y el [Apple TV](http://es.wikipedia.org/wiki/Apple_TV). Apple, Inc. no permite la instalación de iOS en hardware de terceros. Tenía el 26% de cuota de mercado de sistemas operativos móviles vendidos en el último cuatrimestre de [2010](http://es.wikipedia.org/wiki/2010), detrás de [Google](http://es.wikipedia.org/wiki/Google) [Android](http://es.wikipedia.org/wiki/Android) y [Nokia](http://es.wikipedia.org/wiki/Nokia) [Symbian](http://es.wikipedia.org/wiki/Symbian_OS). En mayo de 2010 en los [Estados Unidos](http://es.wikipedia.org/wiki/Estados_Unidos), tenía el 59% de consumo de datos móviles (incluyendo el iPod Touch y el iPad).

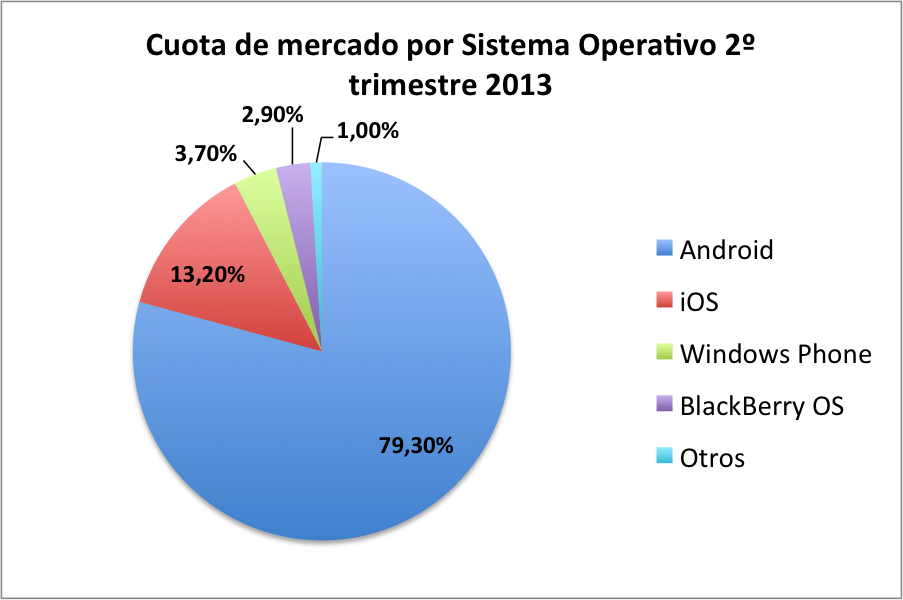
La [interfaz](http://es.wikipedia.org/wiki/Interfaz) de usuario de iOS está basada en el concepto de manipulación directa, usando gestos [multitáctiles](http://es.wikipedia.org/wiki/Multit%C3%A1ctil). Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida.

La interacción con el [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo) incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. Se utilizan acelerómetros internos para hacer que algunas aplicaciones respondan a sacudir el dispositivo (por ejemplo, para el comando deshacer) o rotarlo en tres dimensiones (un resultado común es cambiar de modo vertical al apaisado u horizontal).

iOS se deriva de [Mac OS X](http://es.wikipedia.org/wiki/Mac_OS_X), que a su vez está basado en [Darwin BSD](http://es.wikipedia.org/wiki/Darwin_BSD), y por lo tanto es un sistema operativo Unix.



**Figura 6. Celular con S.O. iOS**



**Figura 7.** **Cuota de mercado de distintos S.O. para dispositivos móviles**

## 2.4 Geolocalización

Se entiende por Geolocalización la identificación de la posición geográfica real de un objeto o persona, ya sea un dispositivo conectado a internet, un teléfono móvil o cualquier otro aparato que sea posible rastrear. Dicha localización puede ser en un plano de dos dimensiones (por ejemplo, Google Maps), como en un plano de tres dimensiones (GPS).

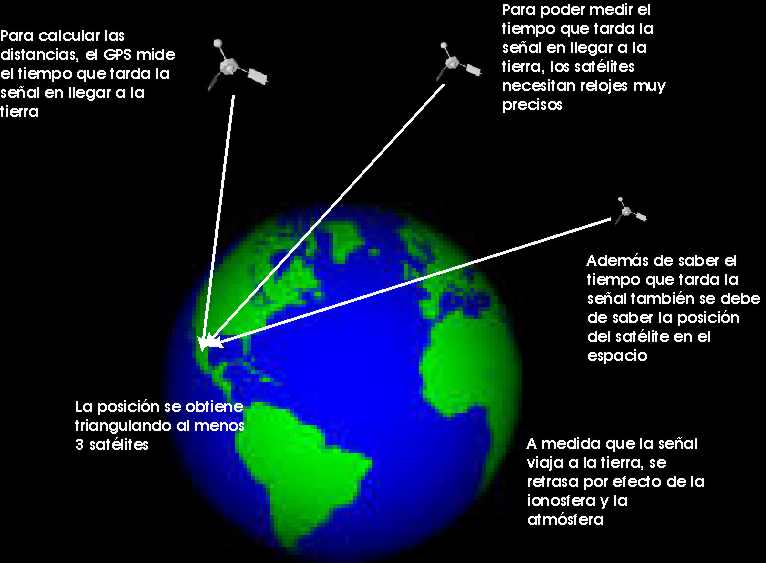
Actualmente es posible el uso de la Geolocalización en la mayoría de las plataformas:

* En el caso de la Geolocalización de un ordenador, esta se hace a través de una serie de bases de datos que aproximan la zona en la que el usuario se encuentra.
* En el caso de un dispositivo Móvil, existen diferentes tecnologías actualmente como GPS o la localización por celdas.

### 2.4.1 GPS

Global Positioning System, o conocido comúnmente por sus siglas GPS, es un sistema basado en la terminal que permite conocer la situación de un objeto o persona en cualquier lugar del mundo. Se trata de una red de 27 satélites que emiten una señal con el tiempo de emisión y su posición. Está señal llega al GPS con un cierto retraso, lo cual nos permite calcular de una manera aproximada la distancia del satélite, ya que sabemos que esa señal viaja a la velocidad de la luz.

En cuanto conocemos la distancia de un satélite y su posición, podemos definir una esfera con el satélite como centro. Conociendo también la posición y la distancia de un segundo satélite, podemos conocer también su esfera, que cortara con la del primer satélite en un círculo, definiendo nuestra posición dentro de ese círculo.



**Figura 8.** **Funcionalidades de un GPS**

### 2.4.2 A-GPS

El AGPS, es un sistema hibrido basado en GPS, creado para solucionar los problemas de localización con GPS en ciudades grandes o interiores con poca cobertura de señal de los satélites, y aumentar su precisión. Se trata de recibirla señal con algún otro tipo de señal, ya sea wireless, telefonía, etc.

Tiene dos sistemas de funcionamiento, el modo online, donde el A-GPS necesita una conexión activa y constante a una red de teléfono (por ejemplo GSM), de la cual puede recibir su propia posición gracias a la celda en que se encuentra, recibir información sobre el entorno como posición de los satélites o condiciones ionosféricas para aumentar la precisión del GPS, o incluso enviar el computo con la información del GPS al servidor de asistencia. En el modo offline, la conexión (ya sea a través de GPRS, Ethernet, Wireles), no es constante, por lo que el A-GPS descarga un fichero con información de su celda o posición de satélites, entorno, etc., mientras disponga de esta conexión, para más tarde usar dicha información incluso durante varios días.



**Figura 9. A-GPS**

## 2.5 Aplicaciones tecnológicas

El conocimiento de la posición de un aparato electrónico puede definir la posición de una persona física en el mundo lo cual tiene una infinidad de posibilidades.

Sobre la Geolocalización dispositivos móviles el conocer la posición actual en un momento determinado nos ofrece el poder disponer de una numerosa información de nuestro alrededor.

### 2.5.1 Sistemas de localización de dispositivos móviles

Existen diferentes maneras de localizar un dispositivo móvil, pero la efectividad del método dependerá de algunas variables como el medio o la disponibilidad de esta medición en el terminal.

Es posible clasificar los diferentes sistemas en tres grandes grupos.

* **Basados en la red**: Estos sistemas utilizan el sistema del proveedor de servicios para determinar la posición del terminal, por lo que no necesitamos ninguna aplicación específica funcionando en el móvil. El problema principal de este sistema es que es preciso estar cerca del proveedor para que funcione.
* **Basados en el terminal**: Los dispositivos que utilizan estos sistemas dependen de un receptor de señales y un software cliente para determinar la posición del terminal a través de las señales externas. Cabe destacar que es preciso instalar una aplicación en el móvil, haciendo que el funcionamiento de esta dependa de la adaptación de los diferentes sistemas operativos.
* **Híbridos**: Los sistemas híbridos son una combinación de sistemas basados en el terminal y la red. Aunque contenga los métodos más fiables, también adquiere los problemas de los dos grupos anteriores.

## 2**.6 Historia de Google Inc.**

En 1995 dos estudiantes de la Universidad de Standford, Larry Page y Sergey Brin, descubren su común interés en averiguar alguna forma de poder conocer el nivel de importancia de una determinada página web[[8]](#footnote-8).En concreto, Larry Page trabajaba en una idea según la cual una página tenía mayor o menor importancia en la red en función del número de páginas que la enlazaran a través de links. La idea deriva del mundo universitario en donde Page se movía, enel que la relevancia de un artículo académico venía dada por las veces que era citado a su vez en el trabajo de otros investigadores.

De este modo Page creó un primer prototipo llamado BackRub, que rastreaba los sitios web y almacenaba los enlaces hallados en ellas en una base de datos. Cuando comenzó a realizar pruebas con BackRub, Page consiguió determinar cuántos enlaces había entre las páginas, pero no hallaba la forma de clasificarlas en importancia, ya que la compleja telaraña de enlaces que se formaba implicaba un considerable cálculo recursivo. Es en este punto donde entra Sergey Brin, licenciado en Informática y Ciencias Matemáticas, que andaba buscando un tema interesante para su tesis doctoral. Brin siempre había destacado por su habilidad matemática y se puso a trabajar junto a Page en elaborar un algoritmo que pudiera clasificar la importancia de los enlaces.

Fruto de esta colaboración nació PageRank, la evolución del proyecto que además de buscar y almacenar enlaces era capaz de clasificar la importancia de una determinada página en función de éstos.

Viendo los buenos resultados, Page y Brin bautizaron de nuevo su creación con el nombre de Google. En 1997, publicaron su buscador en un servidor web de la Universidad de Standford, convirtiéndose en muy poco tiempo en una de las herramientas web favoritas de los alumnos de la Universidad. Este éxito animó a sus creadores a continuar desarrollándola y ampliándola, comprando discos duro y CPU de segunda mano a sus propios compañeros e instalando sus servidores en la misma habitación de Sergey Brin. En pocos meses, Google comenzó a saturar la red de la Universidad de Standford. Es entonces cuando deciden adquirir el dominio *google.com.*

Las necesidades de hardware y espacio Google no dejaban de crecer, hasta el punto de que Larry Page y Sergey Brin se ven incapaces de financiarlo, a pesar de las labores de intermediación de la propia universidad para vender el proyecto. Finalmente,en 1998 y tras una charla informal de 10 minutos con Andy Bechtolsheim(cofundador de Sun Microsystems y vicepresidente de Cisco Systems), éste les extiende un cheque por valor de 100.000 dólares para que paguen “sus discos duros y a los dichosos abogados, para los temas legales”(sic). Para poder hacer efectivo el cobro se ven obligados a fundar sobre la marcha Google Inc., ya que fue el nombre que escribió Bechtolsheim en el cheque.

Desde entonces, Google no ha parado de indexar páginas y se convirtió de forma gradual en el buscador consultado por la abrumadora mayoría de los usuarios de Internet. Larry Page y Sergey Brin no invirtieron ni un solo dólar en publicidad: el boca a boca y sus resultados fueron suficientes para darlos a conocer al mundo. Actualmente una media de 200 millones de consultas al día.

En 2008, Google Inc. recibió el Premio Príncipe de Asturias de Comunicación y Humanidades, por “favorecer el acceso generalizado del conocimiento” y su “contribución decisiva al progreso de los pueblos, por encima de fronteras ideológicas, económicas, lingüísticas o raciales”[[9]](#footnote-9).

### 2.6.1 Productos Google

Sin lugar a dudas, el producto estrella de Google Inc.es su potente buscador web. Hoy en día el uso de internet sería muy diferente sin la posibilidad de realizar búsquedas con la precisión y rapidez que Google puede ofrecer.

En pocos años ha ido creando nuevos servicios que, en la mayoría de los casos, han adquirido tal popularidad entre la gente, que para mucho usuarios sería difícil imaginar el uso cotidiano de Internet si no existieran. En todo ellos, Google Inc. parece querer imprimir la misma filosofía de empresa: un servicio web, de acceso libre, gratuito en su mayor parte, con una interfaz minimalista pero funcional, y orientado siempre a mejorar la experiencia del usuario y a la interconexión con otros servicios en red.

Entre los productos más populares a través de la Web que actualmente ofrece Google, podemos encontrar los siguientes[[10]](#footnote-10).

**Gmail**

Gmail es un completo servicio de correo electrónico lanzado en 2004, que pronto destacó por la gran capacidad de almacenamiento, tamaño de archivos adjuntos, facilidad para realizarbúsquedas entre el historial, su formato de etiquetado y clasificación de mensajes, configuración de idiomas y la sencilla interfaz que ofrecía.

**Google Calendar**

Google Calendar es un servicio web que ofrece una completa agenda y planificación de horarios. Cada usuario dispone de calendarios mensuales o semanales, diferenciados por colores según la actividad, con la posibilidad de detallar cada día de la semana y de programar alertas dirigidas a un teléfono móvil o un correo electrónico. Los usuarios pueden compartir entre sí calendarios y actividades.

**Google Documents**

Representa uno de los proyectos más ambiciosos que actualmente desarrolla Google Inc. El objetivo es crear una completa plataforma de herramientas ofimáticas utilizando únicamente la web, sin la necesidad de instalar ningún software. Pensado para poder competir algún día con el extendido Microsoft Office.

**Google Chrome**

Google Chrome es el navegador web desarrollado por Google Inc. Es un proyecto de software libre y está basado en componentes de otros navegadores de código abierto, como Webkit y Mozilla. Su principal objetivo es ofrecer una navegación más rápida, estable y segura .La versión beta de Google fue lanzada en septiembre de 2008 y aunque recibió una buena acogida, todavía presentaba algunas limitaciones que no le permiten competir con Explorer de Microsoft o con Mozilla Firefox.

**Youtube**

Aunque no es un producto desarrollado por Google Inc., fue adquirido por éste en 2006.Youtube es un sitio web que permite a los usuarios subir y visualizar toda clase de vídeos. Además, los videos pueden recibir comentarios, votaciones, y los usuarios pueden crear sus propios canales de distribución.

**Picasa**

Picasa es otra plataforma software no creada inicialmente por Google Inc., pero sí adquirida en 2004. Este software permite organizar, editar y buscar las fotografías de un usuario. A diferencia de otros servicios, Picasa sí necesita ser instalado en el equipo del usuario, aunque tiene algunas funcionalidades web.

**Blogger**

Adquirido por Google Inc. En 2003, Blogger es uno de los portales de blogs más populares en todo el mundo. Su facilidad de uso permite al usuario registrarse, crear un blog e introducir nuevas entradas en tan sólo unos pequeños pasos.

**Google Maps**

Google Maps es un servicio de localización a través de la Web, que ofrece una vista aérea de casi cualquier lugar del mundo. Mediante este servicio, el usuario puede cambiar entre el punto de vista de una fotografía real de satélite o el de mapa. Además, permite navegar libremente por el plano, alejando y acercando la imagen hasta un nivel de precisión considerable .Ofrece también un servicio de mejor ruta entre dos puntos y de localización exacta mediante la introducción de una dirección cualquiera del globo. Ha sido mejorado con Google Street View, que añade una nueva perspectiva panorámica de 360º desde pie de calle.

Google Maps es el nombre de un servicio gratuito de Google. Es un servidor de aplicaciones de mapas.

El API Google Maps permite insertar Google Maps en aplicaciones de terceros. La API está especialmente diseñada para proporcionar una mayor velocidad y que se pueda aplicar más fácilmente tanto a dispositivos móviles como a las aplicaciones de navegadores web de escritorio tradicionales.

El API proporciona diversas utilidades para manipular mapas (como la de la página [http://maps.google.com](http://maps.google.com/)) y para añadir contenido al mapa mediante diversos servicios, permitiéndote crear sólidas aplicaciones de mapas.

La versión 3 del API de JavaScript de Google Maps es un servicio gratuito disponible para cualquier sitio web que sea gratuito para el consumidor.

Google Maps es un servicio gratuito, sus características principales incluyen:

* Callejeros: se muestra el detalle de la ubicación elegida, con la etiqueta calles y en algunos casos, puntos de referencia.
* Planificador de rutas: ayuda a las personas que viajan a pie, en coche o cualquier otro servicio de transporte público para organizar sus viajes.
* Localizador Urbano de negocios: ayuda a localizar las empresas en muchos países alrededor del mundo.
* Vista satelital: usa fotografías aéreas tomadas desde aviones en algunas zonas. El inconveniente de esta opción es que parte de la información puede cambiar de vez en cuando si las actualizaciones pertinentes no se han hecho en Google Maps.

### 2.6.2 Google en Google Maps

**Google Local**

Google integra las vistas de Google Maps (en una imagen de menor tamaño) en las búsquedas locales. Se pueden buscar por cierto tipo de negocio en cualquier área geográfica, donde Google Local esté disponible.

**Google Ride Finder**

Google lanzó en forma experimental una herramienta basada en Google Maps llamada **Ride Finder**, el cual se basa en la localización permanente de [taxis](http://es.wikipedia.org/wiki/Taxi) y [limusinas](http://es.wikipedia.org/wiki/Limusina) mediante el uso de [GPS](http://es.wikipedia.org/wiki/GPS). La herramienta muestra la ubicación actual de todos los vehículos permitidos de los servicios que participan en las principales ciudades de EE.UU., incluyendo Chicago y San Francisco, en un mapa de Google Maps. A partir de 2009 la herramienta parece ser interrumpida.

**Google Moon**

El [20 de julio](http://es.wikipedia.org/wiki/20_de_julio) de [2005](http://es.wikipedia.org/wiki/2005) y en honor al 36° aniversario del alunizaje y, posterior, caminata lunar, del [Apolo 11](http://es.wikipedia.org/wiki/Apolo_11), Google hizo público el uso de las imágenes de la [NASA](http://es.wikipedia.org/wiki/NASA) acerca de la geografía lunar, siendo integrado a la interfaz de Google Maps.

**Google Mars**

Google Mars proporciona imágenes de satélite de [Marte](http://es.wikipedia.org/wiki/Marte_(planeta)), al igual que Google Moon, pero además muestra imágenes [infrarrojas](http://es.wikipedia.org/wiki/Infrarrojo) e imágenes del relieve de Marte (elevación). Los usuarios pueden accionar los botones elevación, visible, y los datos infrarrojos, de manera semejante a como se cambia entre el mapa, el satélite, y los modos híbridos de Google Maps. Este proyecto es una colaboración entre científicos de la [NASA](http://es.wikipedia.org/wiki/NASA) y de la [Universidad de Arizona](http://es.wikipedia.org/wiki/Universidad_de_Arizona). Para ello Google ha utilizado los datos públicos recogidos a partir de dos misiones de la NASA en Marte, la [Mars Global Surveyor](http://es.wikipedia.org/wiki/Mars_Global_Surveyor) y [Mars Odyssey](http://es.wikipedia.org/wiki/Mars_Odyssey).

**Google Send-to-Car**

Esta aplicación permite enviar una dirección de Google Maps directamente al [sistema de navegación del vehículo](http://es.wikipedia.org/wiki/Sistema_de_navegaci%C3%B3n_para_autom%C3%B3viles) o [GPS](http://es.wikipedia.org/wiki/GPS), sin necesidad de teclearla en el mismo.

Recientemente, Google ha incorporado un servicio experimental en [Google Maps que permite conocer el estado del tráfico](http://lbspro.com/?q=Google-maps-estado-trafico) en tiempo real. Este piloto está en fase de pruebas en [Estados Unidos](http://es.wikipedia.org/wiki/Estados_Unidos).

### 2.6.3 Impactó en la sociedad

Como apoyo al sitio web, Google lanzó [Google Earth](http://es.wikipedia.org/wiki/Google_Earth) para permitir un uso más personalizado de los mapas, haciendo posible colocar nombres a las calles sobre los mapas, sin perder la información. Como Google Maps está desarrollado casi por entero con JavaScript y [XML](http://es.wikipedia.org/wiki/XML), algunos usuarios han hecho la [ingeniería inversa](http://es.wikipedia.org/wiki/Ingenier%C3%ADa_inversa), y han desarrollado códigos para aumentar las capacidades de la interfaz de Google Maps.

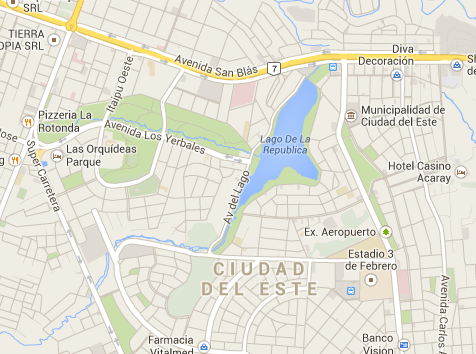
Usando el núcleo de las imágenes almacenadas por Google, muchas herramientas pueden personalizar los iconos de localización, conocer la posición exacta en la tierra, e incluso, personalizar las imágenes dentro de la interfaz de Google Maps. Algunos de los "Hacks" de Google Maps son como la herramienta para ubicar las propiedades en renta de la empresa Craigslist, las llamadas de Seattle al 911 o los datos de crímenes en Chicago.

### 2.6.4 Google Maps API Android

Permite a los usuarios explorar el mundo con mapas ricos proporcionados por Google. Identificar lugares con marcadores personalizados, aumentar los datos de los mapas con superposiciones de imágenes, insertar uno o más mapas como fragmentos, y mucho más.

Para ayudar a añadir potentes capacidades de creación de mapas a las aplicación móvil, el complemento para las API de Google incluye una biblioteca externa de Google Maps, [*com.google.android.maps*](https://developers.google.com/maps/documentation/android/reference/index?hl=es). Las clases de la biblioteca de Google Maps ofrecen descargas, procesamiento y almacenamiento en caché integrados de mosaicos de Google Maps, así como numerosos controles y opciones de visualización.

La clase clave de la biblioteca de Google Maps es [MapView](https://developers.google.com/maps/documentation/android/reference/com/google/android/maps/MapView.%20html?hl=es), una subclase de [ViewGroup](http://developer.android.com/reference/android/view/ViewGroup.html) de la biblioteca estándar de Android. Una clase MapView muestra un mapa con datos obtenidos del servicio de Google Maps. Cuando la clase MapView está enfocada, puede capturar pulsaciones de teclas y formas de tocar la pantalla para desplazar y acercar o alejar el mapa de forma automática, incluso administrar solicitudes de redes para obtener mosaicos de mapas adicionales. También proporciona todos los elementos de interfaz de usuario que necesitan los usuarios para controlar el mapa. Tu aplicación solo puede utilizar los métodos de clase MapView para controlar MapView mediante programación y dibujar algunos tipos de superposiciones (Overlay) en la parte superior del mapa.



**Figura 10. Ciudad del Este en Google Maps**

**CAPÍTULO III**

**PLATAFORMA ANDROID.**

***Educar no es dar carrera para vivir, sino templar el alma para las dificultades de la vida.*** *(Pitágoras)*

# Capitulo III: Plataforma Android

Android es una plataforma de software para dispositivos móviles que incluye un [sistema operativo](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dandroid%2Bdeveloper%2Bhistory%26biw%3D1024%26bih%3D537&rurl=translate.google.com.py&sl=en&u=http://en.wikipedia.org/wiki/Operating_system&usg=ALkJrhhPJt5kJHgtLz6fJMGxpgjig4mUyA) ,[middleware](http://translate.googleusercontent.com/translate_c?depth=1&hl=es&prev=/search%3Fq%3Dandroid%2Bdeveloper%2Bhistory%26biw%3D1024%26bih%3D537&rurl=translate.google.com.py&sl=en&u=http://en.wikipedia.org/wiki/Middleware&usg=ALkJrhgFfaNR8-lcEMjBqWr_8_ZCr1Bt5w) y diversas aplicaciones de usuario. Representa la primera incursión seria de Google en el mercado móvil y nace con la pretensión de extender su filosofía a dicho sector.

Todas las aplicaciones para Android se programan en el lenguaje Java y son ejecutadas en una máquina virtual especialmente diseñada para esta plataforma, que ha sido bautizada con el nombre de Dalvik. El núcleo de Android está basado en Linux 2.6.

La licencia de distribución elegida para Android ha sido Apache 2.0[[11]](#footnote-11), lo que lo convierte en software de libre distribución. A los desarrolladores se les proporciona de forma gratuita un SDK y la opción de un *plug-in* para el entorno de desarrollo Eclipse, que incluyen todas las API necesarias para la creación de aplicaciones, así como un emulador integrado para su ejecución. Existe además disponible una amplia documentación de respaldo para este SDK[[12]](#footnote-12).

El proyecto Android está capitaneada por Google y un conglomerado de otras empresas tecnológicas agrupadas bajo el nombre de Open Handset Alliance (OHA)[[13]](#footnote-13).

Con Android se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, videojuegos, etc.), así como poder crear aplicaciones que sean verdaderamente portable, reutilizable y de rápido desarrollo.

Además de todo ello, otro aspecto básico para entender la aparición de Android es que pretende facilitar la integración de estos dispositivos con las posibilidades ofrecidas por la web.

## 3.1 Arquitectura

Como ya se ha mencionado, Android es una plataforma para dispositivos móviles que incluye un sistema operativo, middleware y aplicaciones básicas para el usuario.

En las siguientes líneas se dará una visión global por capas de cuál es la arquitectura empleada en Android[[14]](#footnote-14). Cada una de estas capas utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores.



**Figura 11. Arquitectura de Android**

La capa más inmediata es la correspondiente al núcleo de Android. El núcleo del sistema operativo Android está basado en el kernel de Linux versión 2.6, similar al que puede incluir cualquier distribución de Linux, como Ubuntu, solo que adaptado a las características del hardware en el que se ejecutará Android, es decir, para dispositivos móviles.

El núcleo actúa como una capa de abstracción entre el hardware y el resto de las capas de la arquitectura. El desarrollador no accede directamente a esta capa, sino que debe utilizar las librerías disponibles en capas superiores. Para cada elemento de hardware del teléfono existe un controlador (o driver) dentro del kernel que permite utilizarlo desde el software.

El kernel también se encarga de gestionar los diferentes recursos del teléfono (energía, memoria, etc.) y del sistema operativo en sí: procesos, elementos de comunicación (networking), etc.

La siguiente capa corresponde con las librerías utilizadas por Android. Están escritas en C o C++ y compiladas para la arquitectura hardware específica del teléfono. Estas normalmente están hechas por el fabricante, quien también se encarga de instalarlas en el dispositivo antes *de* ponerlo a la venta. El objetivo de las librerías es proporcionar funcionalidad a las aplicaciones para tareas que se repiten con frecuencia, evitando tener que codificarlas cada vez y garantizando que se llevan a cabo de la forma “más eficiente”.

Entre Librerías más importante se puede mencionar las siguientes:

* La librería *libc* incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
* La librería *Surface Manager* es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
* *Open*GL*/SL y SGL* representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android.
* La librería *Media Libraries* proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animaciones, etc.).
* *Free Type*, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
* La librería *SSL* posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
* La librería SQLite, a través de esta librería, Android ofrece la creación y gestión de base de datos relacionales, pudiendo transformar estructuras de datos en objetos fáciles de manejar por las aplicaciones.
* La librería *Webkit* proporciona un motor para las aplicaciones de tipo navegador, y forma el núcleo del actual navegador incluido por defecto de la plataforma Android.

Los dos últimos niveles de la arquitectura de Android están escritos enteramente en Java. El framework de aplicaciones representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación.

Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo framework, representado por este nivel.

Entre las API Más importante ubicadas aquí, se pueden encontrar las siguientes:

* Activity Manager. Se encarga de administrar la pila de actividades de nuestra aplicación así como su ciclo de vida.
* Windows Manager. Se encarga de organizar lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.
* Content Provider. Esta librería es muy interesante porque crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información.
* Views. En Android, las vistas los elementos que nos ayudarán a construir las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados como un navegador web o un visor de Google Maps.
* Notification Manager. Engloba los servicios para notificar al usuario cuando algo requiera su atención mostrando alertas en la barra de estado. Un dato importante es que esta biblioteca también permite utilizar sonidos, activar el vibrador o los LEDs del teléfono en caso de tenerlos.
* Package Manager. Esta biblioteca permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android, estos contienen el archivo .apk, que a su vez incluyen los archivos. dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.
* Telephony Manager. Con esta librería podremos realizar llamadas o enviar y recibir SMS/MMS, aunque no permite reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.
* Resource Manager. Con esta librería podremos gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, imágenes, sonidos o layouts. En un post relacionado a la estructura de un proyecto Android veremos esto más a fondo.
* Location Manager. Permite determinar la posición geográfica del dispositivo Android mediante GPS o redes disponibles y trabajar con mapas.
* Sensor Manager. Nos permite manipular los elementos de hardware del teléfono como el acelerómetro, giroscopio, sensor de luminosidad, sensor de campo magnético, brújula, sensor de presión, sensor de proximidad, sensor de temperatura, etc.
* Cámara: Con esta librería podemos hacer uso de la(s)  cámara(s) del dispositivo para tomar fotografías o para grabar vídeo.
* Multimedia. Permiten reproducir y visualizar audio, vídeo e imágenes en el dispositivo.

## 3.2 Componentes de una aplicación Android

Todas las aplicaciones de Android pueden descomponerse en cuatro tipos de bloques o componentes principales. Cada aplicación será una combinación de uno o más de estos componentes, que deberán ser declarados de forma explícita en un fichero con formato XML, denominado *AndroidManifest.xml*, junto a otros datos asociados como valores globales, clases que implementa, datos que puede manejar, permisos, etc. Este fichero es básico en cualquier aplicación en Android y permite al sistema desplegar y ejecutar correctamente la aplicación.

A continuación se exponen los cuatro tipos de componentes a los que puede dividirse una aplicación para Android.

### 3.2.1 Activity

Sin duda es el componente más habitual de las aplicaciones para Android. Un componente Activity refleja una determinada actividad llevada a cabo por una aplicación, y que lleva asociada típicamente una ventana o interfaz de usuario; es importante señalar que no contempla únicamente el aspecto gráfico, sino que éste forma parte del componente *Activity,* a través de vistas representadas por clases como View y sus derivadas. Este componente se implementa mediante la clase del mismo *nombre Activity.*

### 3.2.2 Broadcast Intent Receiver

Un componente *Broadcast Intent Receiver* se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente *Activity*). Este componente implementa a través de una clase de nombre *BroadcastReceiver.*

Para que *Broadcast Intent* *Receiver* funcione, no es necesario que la aplicación active en el momento de producirse el evento. El sistema lanzará la aplicación si es necesario cuando el evento motorizado tenga lugar.

### 3.2.3 Service

Un componente *Service* representa una aplicación ejecutada sin interfaz de usuario, y que generalmente tiene lugar en segundo plano mientras otras aplicaciones (éstas con interfaz) son las que están activas en la pantalla del dispositivo.

Un ejemplo típico de este componente es un reproductor de música. La interfaz de usuario del reproductor sería un componente *Activity,* pero la música en reproducción sería un componente *Service*, por que se ejecuta el background.

Este elemento está implementado por la clase del mismo nombre *Service*.

### 3.2.4 Content Provider

Con el componente Content Provider, cualquier aplicación en Android puede almacenar datos en un fichero, en una base de datos SQLite o en cualquier otro formato que considere. Una clase que implemente el componente *Content Provider* contendrá una serie de métodos que permite almacenar, recuperar, actualizar y compartir los datos de una aplicación*.*

Existe una colección de clases para distintos tipos de gestión de datos en el paquete *android.provider*. Además, cualquier formato adicional que se quiera implementar deberá pertenecer a este paquete y seguir sus estándares de funcionamiento**.**

## 3.3 Ciclo de vida de una aplicación Android

En Android cada aplicación se ejecuta en su propio proceso. Esto aporta beneficios en cuestiones básicas como seguridad, gestión de memoria, o la ocupación de la CPU del dispositivo móvil. Android se ocupa de lanzar y parar todos estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario.

Android lanza tantos procesos como permitan los recursos del dispositivo. Cada proceso, correspondiente a una aplicación, estará formado por una o varias actividades independientes (componentes Activity) de esa aplicación. Cuando el usuario navega de una a otra actividad siguen existiendo en el sistema, pero están dormidos y su estado ha sido guardado. Es entonces cuando crea, o despierta si ya existe, el proceso para la aplicación que debe ser lanzada, asumiendo que existan recursos para ello.

Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido; esto implica que el desarrollador puede controlar a cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan.

Android clasifica los procesos de manera jerárquica según la importancia que tengan en un momento determinado para el sistema:

* Foreground process: éste es un proceso en primer plano con el que el usuario estaría interactuando, éste hospeda una activity y el sistema lo eliminaría como última opción.
* Visible process: también hospeda un activity pero que no está en primer plano, aunque si visible. Este tipo de proceso solo será eliminado en estado de memoria muy críticos igual que el Foreground process.
* Background process: proceso que hospeda una activity pero no está visible en pantalla, es decir, que si lo eliminamos no tiene una recuperación directa con el usuario.
* Empty process: un proceso que no hospeda ningún componente visualmente activo. El sistema lo eliminará con frecuencia y puede mantenerlos vivos si hay memoria suficiente para mejorar el tiempo de activación de otro componente de esta aplicación.



**Figura 12. Ciclo de vida de un objeto Activity**

## 3.4 Gestión de la información en una aplicación Android

A diferencia de lo que ocurre en otros sistemas, en Android cualquier repositorio de datos, como por ejemplo archivos, son privados y únicamente accesibles por la aplicación a la que pertenecen. Sin embargo, esto no implica necesariamente que no puedan ser compartidos o accedidos por otras aplicaciones: Android facilita una serie de mecanismos que posibilitan en ciertas circunstancias el acceso a dicha información.

### 3.4.1 Preferencias de usuarios

Las preferencias son todos aquellos valores asociados a una determinada aplicación y que permiten adaptar ésta a los gustos o necesidades del usuario. Solamente se pueden acceder a las preferencias dentro de un mismo paquete: Android no permite compartir estos valores con otras aplicaciones.

Para poder compartir las preferencias entre todos los componentes que forman la aplicación, dentro del mismo paquete, debe asignarse un nombre al conjunto de valores que forman las preferencias de usuario. Después, éstas se pueden recuperar a lo largo de todo el paquete haciendo una llamada a *Contex.getSharedPreferences()*. Si las preferencias solamente serán accedidas desde un componente Activity y no es necesario compartirla con los demás componentes, es posible omitir el nombre asociado a las preferencias y recuperarlas simplemente llamando a *Activity.getPreferences ().*

### 3.4.2 Ficheros

Para leer o escribir fichero, Android facilita los métodos *Context.openFileInput()* y *Context.openFileOutput()*, respectivamente. Estos ficheros deben ser locales a la aplicación en curso, es decir, al igual que con las preferencias, un Fichero estándar de datos no puede ser compartido con otras aplicaciones.

Si el fichero es estático y puede ser adjuntado a la aplicación en el momento que se compila, éste puede ser añadido como un recurso más .Su ubicación correcta será en la carpeta de recursos.

### 3.4.3 Bases de datos

Android incluye una librería de SQLite que permite crear bases de datos relacionales, navegar entre las tablas, ejecutar sentencias en SQL y otras funcionalidades propias del sistema SQLite.

### 3.4.4 Acceso por red

Como es lógico Android permite la comunicación entre dispositivos a través de una infraestructura de red, siempre y cuando ésta esté disponible. Los paquetes *java.net* y *android.net* ofrecen multitud de clases y posibilidades en ese sentido.

### 3.4.5 Content Provider

Android incluye una serie de componentes Content Provider que permiten publicar todo tipo de datos básicos que pueden resultar útiles entre aplicaciones: información de los contactos, fotografías, imágenes, etc. Todos estos Content Provider ya definidos y listos para utilizar se pueden encontrar en el paquete *android.provider*. Además Android ofrece la posibilidad de que el desarrollador pueda crear sus propios Content Provider.

Un Content Provider es un objeto de la clase ContentProvider, ubicada en el paquete *android.content*.y que almacena datos de un determinado tipo que pueden ser accedido desde cualquier aplicación. Cada elemento Content Provider tiene asociado una URL única que lo representa y a través de la cual los otros componentes de una aplicación pueden acceder a él.

## 3.5 Instalación de Eclipse con el SDK de Android

En este apartado se exponen los pasos necesarios para empezar a desarrollar y a entender aplicaciones para Android. Las instrucciones de instalación aquí descritas se basan en el sistema operativo Windows XP o Windows Vista.

### 3.5.1 Descargar el SDK de Android

Android es una plataforma de software libre, por lo que cuenta con un SDK disponible para todo desarrollador que lo desee e incluye, entre otros elementos, el conjunto completo de API que este sistema soporta. Para descargarlo, basta con visitar la web de Android[[15]](#footnote-15) y asegurarse de acceder a la última versión publicada.

Una vez descargado el SDK, es necesario descomprimirlo. La ubicación de los ficheros resultantes no es relevante, pero conviene recordar la ruta para posteriores.

### 3.5.2 Descargar Eclipse IDE

La descarga de Eclipse no es muy diferente al SDK de Android. La web de Eclipse[[16]](#footnote-16) ofrece multitud de versiones de este entorno de desarrollo según las necesidades del desarrollador.



**Figura 13. Entorno de desarrollo integrado Eclipse**

Finalizada la descarga, no se realiza ningún proceso de instalación; simplemente se debe descomprimir los ficheros y pulsar el ejecutable para abrir la aplicación. La primera vez que se inicie Eclipse, pide al usuario una localización para el workspace, donde se ubicarán por defecto todos los proyectos desarrollados.

### 3.5.3 Instalar el plug-in de Android

El siguiente paso consiste en instalar un plug-in específico de Android para la plataforma Eclipse. Esta herramienta llamada ADT (Android Development Tools), facilita enormemente la creación de proyectos, su implementación, depuración y ejecución, por lo que es altamente recomendable si se quiere trabajar con Android.

Para instalar el *plug-in* ADT en Eclipse, es necesario seguir las siguientes indicaciones:

* Iniciar Eclipse
* Seleccionar la pestaña *Help> Software Updates*. Esta acción abrirá una nueva ventana llamada *Software Updates and Add-ons.*
* Pinchar en la pestaña *Available Software* y pulsar el botón *Add Site.*
* Introducir la siguiente URL y pulsar OK:

[*https: / /dl-ssl .google.com/*](https://dl-ssl.google.com/) *android /eclipse/*

* Volviendo a la ventana *software Updates and Add-ons*, marcar la casilla correspondiente a Developer Tools y pulsar el botón Install. Se abrirá una nueva ventana.
* Cerciorarse de que las opciones Android Developer Tools y Android Editors están marcadas y pulsar el botón Finish.
* El proceso de instalación dará comienzo y puede llevar algunos minutos .Con el fin de que los cambios tengan efecto, es necesario reiniciar Eclipse.

### 3.5.4 Referenciar el SDK

Tras abrir de nuevo Eclipse, debe indicarse en las preferencias de Eclipse la localización del SDK a utilizar para los proyectos de Android:



**Figura 14. SDK (Kit de Desarrollo de Software)**

* Seleccionar la pestaña *Windows >Preferences*, lo que abrirá una nueva ventana.
* Elegir Android en el panel izquierdo.
* Pulsar el botón *Browse* e indicar la ruta del *SDK de Android*.
* Pulsar el botón *Apply* y después ok.

### 3.5.5 Actualizaciones del plug-in

Para comprobar las posibles actualizaciones a través de Eclipse, se debe realizar lo siguiente:

* Iniciar Eclipse
* Seleccionar la pestaña *Help> Software Updates*.
* Elegir la pestaña *Installe software.*
* Pulsar el botón *Update.*
* Si existe alguna actualización para el ADT, seleccionarla y pulsar el botón Finish.

### 3.5.6 Creación de una Aplicación de Ejemplo

La creación de un nuevo proyecto de Android no representa mayor dificultad que crear cualquier otro tipo de proyecto en Eclipse. Una vez abierto este entorno de desarrollo, deben realizarse los siguientes pasos:

**Crear un nuevo proyecto**

1. **Crear el proyecto**

Seleccionar la pestaña File > New > Proyect, y en el menú resultante desplegar la opción Android y seleccionar Android Proyect. Pulsar el botón Next.

1. **Especificar las propiedades**

A continuación se muestra una ventana donde es necesario detallar algunas propiedades del proyecto.

Project name: es el nombre del proyecto. En la práctica, será el nombre que reciba la carpeta donde se guardará todo lo relativo al presente proyecto, dentro del workspace. Por ejemplo. “*HolaMundo”*.

* **Package name**: el nombre del paquete bajo el cuál será desarrollado todo el código. Por ejemplo, se le puede asignar el valor “*com.android.hola*”.
* **Activity name**: es el nombre de la clase Activity que será creada de forma automática por el plug-in. Esta clase Activity simplemente es una clase ejecutable, capaz de realizar alguna tarea, y es imprescindible en la mayoría de las aplicaciones para Android. Por ejemplo, póngase el nombre “*HolaMundo*”.
* **Application name**: el nombre de la aplicación que se va a desarrollar .Constituye el nombre visible para el usuario del dispositivo móvil. Por ejemplo, “*Saludo al Mundo”.*

Tras pulsar el botón Finish, Eclipse mostrará en el explorador de paquetes los elementos que conforman el actual proyecto. Además, si se visita el *workspace* asignado `para los proyectos de eclipse, podrá observarse que existe una nueva carpeta denominada “*HolaMundo*”.El contenido de esta carpeta se explica más adelante**.**

### 3.5.7 Añadir interfaz de usuario

Para continuar con el ejemplo, debe desplegarse en el explorador de paquetes la carpeta denominada “*src*”. En esta carpeta existirán dos ficheros: “*HolaMundo.java*” y “*R.java”*. Ambos constituyen hasta el momento los dos únicos ficheros fuente del proyecto.

Si se abre el fichero “*HolaMundo.java*”, puede encontrarse el siguiente código:

|  |  |  |  |
| --- | --- | --- | --- |
| **package** com.android.hola; | | |  |
|  |  |  |  |
| **import** android.app.Activity; | | |  |
| **import** android.os.Bundle; | | |  |
|  |  |  |  |
| **public class** HolaMundo **extends** Activity { | | | |
| /\*\* Called when the activity is first created. \*/ | | | |
| @Override | | | |
| **public void** onCreate (Bundle savedInstanceState) { | | | |
| super.onCreate(savedInstanceState) ; | | | |
| setContentView (R.layout.*main*) ; | | | |
|  | | | |
| } | | | |
| } | | | |
|  | | | |

**Figura 15. Código inicial de “HolaMundo.java”**

Este código es perfectamente ejecutable, pero para este ejemplo se van a añadir algunos cambios. El objetivo es crear una simple interfaz de usuario que permita mostrar el mensaje “HolaMundo”. El nuevo código resultante ha de ser el siguiente**:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **package** com.android.hola; | | |  |
|  |  |  |  |  |
|  | **import** android.app.Activity; | | |  |
|  | **import** android.os.Bundle; | | |  |
|  | **import** android.widget.TextView; | | |  |
|  |  |  |  |  |
|  | **public class** HolaMundo **extends** Activity { | | | |
|  |  |  |  |  |
|  | / \*\* Called when the activity is first created. \*/ | | | |
|  | @Override  **public void** onCreate (Bundle savedInstanceState) { |  |  |  |
|  | **super**.onCreate ( savedInstanceState) ; | | | |
|  | TextView tv = **new** TextView(this); | | |  |
|  |  |  |  |  |
|  | tv.setText ("Hola Mundo”); |  | |  |
|  | setContentView(tv); | |  |  |
|  | } |  |  |  |
| } |  |  |  |  |

**Figura 16. Código de HolaMundo.java.modificado**

En Android las interfaces de usuario están formadas por una jerarquía de clases llamadas *Views* (vistas).

Una vista o *View* presenta cualquier elemento que se pueda dibujar en pantalla, como un botón, una animación, una etiqueta de texto, etc. En este ejemplo se crea una etiqueta de texto mediante una clase denominada *TextView.*

Al crear una instancia de la clase *TextView,* es necesario pasarle al constructor un determinado contexto de Android. Un contexto representa una especie del manejador del sistema, que facilita la resolución de recursos, el acceso a base de datos o la configuración de ciertas características de la aplicación. La clase *Activity* hereda de la clase Context, por lo que la clase HolaMundo también es un contexto válido en Android.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | TexView tv = **new** TextView(**this**) ; | | |  |
|  |  |  |  |  |

**Figura 17. Creación de un objeto TextView**

Una vez creada la etiqueta de texto, para asignarle un valor simplemente se llama al método setText ( ) con la cadena “Hola Mundo” como argumento.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | tv.setText ("hola Mundo") ; | | |  |
|  |  |  |  |  |

**Figura 18. Especificación del mensaje para el objeto TextView**

Como paso final, deben vincularse los elementos visuales que se hayan creado con la pantalla que la aplicación mostrará al usuario. Este paso se realiza mediante el método *setContentView*( ) de la clase *Activity*, que asigna una determinada vista indicando al sistema que es la que debe ser mostrada por la pantalla.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  | setContentView (tv); | |  |  |
|  |  |  |  |  |

**Figura19. Establecer la vista para el usuario**

### 3.5.8 Ejecutar la aplicación

Antes de lanzar y probar la aplicación, debe crearse una configuración específica para la ejecución .En esta configuración se debe indicar el proyecto a ejecutar, la Activity que iniciará la ejecución, las opciones del emulador y otros valores opcionales.

* Seleccione la pestaña *Run* > *Run Configurations* .Se abrirá una nueva ventana.
* En el panel de la izquierda, hay que localizar Android Application y pulsar el icono con la leyenda *New launch configuration*.
* Asignar un nombre para esta configuración. Por ejemplo: “*Hola Mundo Conf*”.
* En la etiqueta Android, pulsar el botón Browse para localizar el proyecto “*HolaMundo”* y seleccionar la opción Launch para localizar la actividad “*com.android.hola.HolaMundo”*.
* Pulsar el botón Run.

Esto lanzará el emulador de Android. La primera vez que se lanza, puede tardar unos segundos en arrancar .Es recomendable no cerrar el emulador mientras se éste editando y probando un proyecto, para evitar así tener que repetir siempre el proceso de carga.

**CAPÍTULO IV**

**DESARROLLO DE PROYECTO.**

***No hay ningún viento favorable para el que no sabe a qué puerto se dirige.*** *(Schopenhauer)*

# Capitulo IV: Desarrollo del Proyecto

## TuristaApp: Aplicativo de Geolocalización de Hoteles y Restaurantes

En este capítulo se explica paso a paso el desarrollo del aplicativo de geolocalización de hoteles y restaurantes, cuyo objetivo es mostrar en un mapa la ubicación de los mismos a través de los datos obtenidos de un webservice[[17]](#footnote-17), añadiendo una serie de funcionalidades.

Mediante este proceso se busca ilustrar de una forma práctica, las características de desarrollo de una aplicación móvil Android y la integración con servicios de datos de internet.

Algunos aspectos que se tendrán en cuenta para el desarrollo de la aplicación pueden ser clasificados desde el lado del servicio y el lado de la aplicación móvil.

Servicio

* Aplicación web para el registro lugares de interés turístico.
* Utilización de Api Google Maps for Web para la ubicación de lugares.
* Construcción de un servicio rest[[18]](#footnote-18) para el envío de datos a la aplicación móvil.

Aplicación Android

* Uso de componentes *Activity.*
* Comunicación de componentes a través de *Intents.*
* Utilización de la Api Google Maps for Android.
* Comunicación HTTP con el servicio de datos.
* Composición de interfaces de usuario siguiendo las líneas de diseño y usabilidad de una Aplicación Android.

## 4.1 Análisis y Diseño de la Aplicación

El objetivo básico es el desarrollo de una aplicación para mostrar la ubicación geográfica de Hoteles y Restaurantes de Ciudad del Este ofreciendo información sobre los mismos. Para ello se necesario disponer de un servicio de visualización de mapas, Google Maps, un servicio de publicación de datos sobre lugares de interés turístico y una conexión a internet con el fin de enviar información desde el servicio de datos al dispositivo móvil.

El usuario visualiza la ubicación geográfica de hoteles y restaurantes de Ciudad del Este a través de la aplicación utilizando el servicio de Google Map, obteniendo los datos a través de un servidor donde están previamente registrados, pudiendo interactuar con funcionalidades que utilizan los datos provistos, como ser visitar el sitio web, realizar una llamada al local, enviar un mensaje o un email entre otras.

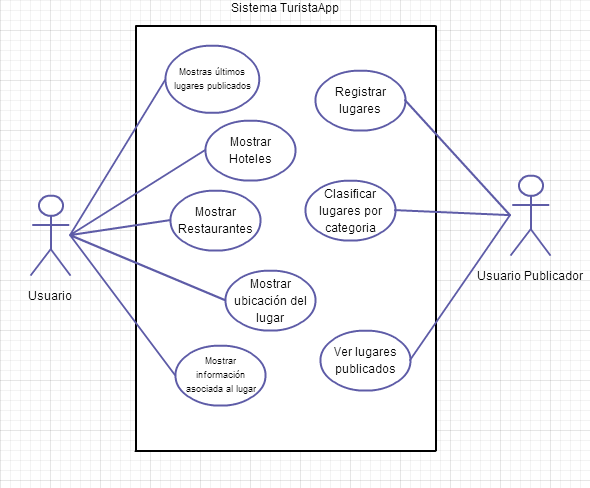
Por otro lado, a través de la conexión a Internet, la aplicación móvil realiza la recepción de los datos sobre los lugares. Esta comunicación se realiza a partir de una base de datos central donde previamente se deben registrar los lugares a publicarse. El intercambio de datos se realiza utilizando el estándar JSON sobre el protocolo HTTP.

## 4.2 Caso de Uso

En este apartado se utiliza los *casos de uso[[19]](#footnote-19)* como forma de detallar mejor las funcionalidades que la aplicación *TuristaApp* va a desempeñar frente al usuario de la aplicación móvil, así como al registro de los hoteles y restaurantes, resaltando los aspectos más ilustrativos que ayuden a comprender mejor la aplicación desarrollada.

Siempre es bueno recordar que la ingeniería de software[[20]](#footnote-20) nos dice que un *caso de uso*representa un uso típico que se le da al sistema. La técnica del caso de uso permite capturar y definir requisitos que deben cumplir una aplicación, describiendo las típicas interacciones que hay entre el usuario y esta.

A continuación se muestra un diagrama de caso de uso asociado a la aplicación TuristaApp utilizando el estándar UML 2.0[[21]](#footnote-21)



**Figura 20. Diagrama de Caso de Uso**

En el diagrama mostrado en la Figura 20 el sistema es representado por la caja que contiene los casos de uso, cada caso de uso consiste en un óvalo con un nombre descriptivo en su interior. Fuera del sistema se encuentran los actores que interactúan con él, quienes realizan los casos de uso.

A continuación se ofrece una breve descripción del contenido de cada caso de uso mostrado en el diagrama:

* Mostrar últimos lugares publicados: muestra en una lista los lugares publicados más recientemente.
* Mostrar Hoteles: muestra los lugares registrados correspondientes a la categoría Hoteles.
* Mostrar Restaurantes: muestra los lugares registrados correspondientes a la categoría Restaurantes.
* Mostrar ubicación del lugar: muestra en un mapa la posición del lugar seleccionado.
* Mostrar información asociada al lugar: muestra información relacionada al lugar, como ser datos de contacto y descripción del mismo.
* Registrar Lugares: permite registrar un lugar, especificando la posición geo referenciada y asociando información de interés al mismo.
* Clasificar lugares por categoría: permite ver los lugares registrados clasificándolos de acuerdo a su categoría.
* Ver lugares publicados: permite seleccionar un lugar registrado y ver la información asociada al mismo.

Es importante recalcar que los casos de uso simplemente expresan el punto de vista del usuario sobre cómo debe funcionar la aplicación y que pueden realizar a través de ella.

## 4.3 Intercambio de información

La aplicación TuristaApp obtiene la información de la localización de los lugares a través de un servidor, donde se cuenta con una base de datos con los lugares registrados. Este intercambio de información entre la aplicación móvil y el servidor se realiza utilizando JSON[[22]](#footnote-22), el cual es un formato ligero de intercambio de datos, que permite expresar de forma sencilla estructura de datos, manteniendo intacta la semántica de los contenidos.

El uso del formato JSON, ante otras alternativas, responde principalmente a dos criterios. Por un lado, se trata de un estándar regulado y ampliamente utilizado, más aun en intercambio de datos en aplicaciones móviles, lo cual facilita su comprensión y la posible ampliación o interconexión con otros servicios en el futuro. Por otro lado, es una tecnología fácil de utilizar y cuenta con librerías que permiten la creación y manipulación de documentos en formato JSON de manera automática. En concreto, en el lado del servidor se opto por utilizar la implementación de referencia para Java, la cual es la librería JERSEY[[23]](#footnote-23).

Vale la pena señalar que Jersey es la implementación de referencia para la especificación JAX-RS[[24]](#footnote-24) que permite la implementación de servicios en el estilo arquitectónico REST en Java de manera sencilla y fácil. Entre tanto, en la aplicación móvil Android, si bien cuenta con soporte nativo para JSON, se optó utilizar la librería GSON[[25]](#footnote-25), desarrollada con el patrocinio de Google y utilizada por los mismos en sus aplicaciones, por las facilidades que presentan para la manipulación de datos en formato JSON en comparación con el soporte nativo de Android.

Cuando se utiliza el formato JSON para el intercambio de información entre el aplicativo cliente Android y el Servidor de Datos (Aplicación Web), se debe tener en cuenta que ambas aplicaciones deben manejar un mismo esquema de dominio. Utilizando el leguaje java en ambos contextos, este dominio viene determinado por las clases de dominio que representan el modelo de datos relacional presente en la base de datos y los objetos creados a partir de los registros. Estos objetos son convertidos al formato JSON por el servidor a partir de una requisición del aplicativo cliente y enviados al mismo, el cual a partir de los datos recibidos, manipula el formato JSON convirtiéndolo nuevamente a un objeto java para ser utilizado.

Para comprender mejor ese proceso de comunicación es necesario conocer las reglas del formato JSON que permite representar objetos java en estructura de texto.

JSON está constituido por dos estructuras:

1. Una colección de pares de nombre/valor. En varios lenguajes esto es conocidos como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
2. Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Un objeto es un conjunto desordenado de pares nombre/valor. Un objeto comienza con { (llave de apertura) y termina con } (llave de cierre). Cada nombre es seguido por: (dos puntos) y los pares nombre/valor están separados por, (coma).



**Figura 21. Objeto en formato JSON**

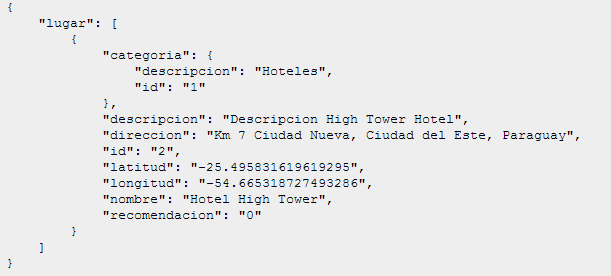
Un arreglo es una colección de valores. Un arreglo comienza con [ (corchete izquierdo) y termina con ] (corchete derecho). Los valores se separan por, (coma).



**Figura 22. Arreglo en formato JSON**

Un valor puede ser una cadena de caracteres con comillas dobles, o un número, o true o false o null, o un objeto o un arreglo. Estas estructuras pueden anidarse.

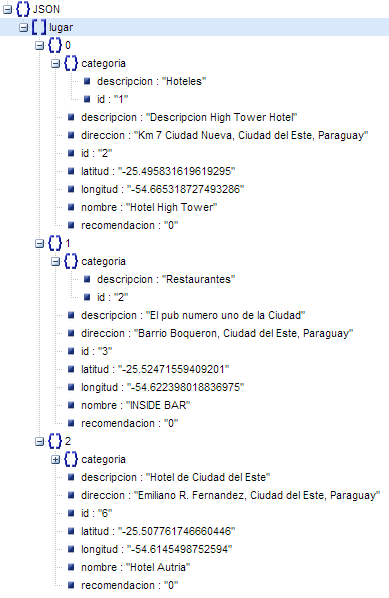
A partir de esto se puede determinar la composición y estructura de los datos intercambiados por el servidor y la aplicación móvil Android.



**Figura 23. Respuesta en formato JSON**

La figura 23 muestra la respuesta del servidor de datos en formato JSON válida para el aplicativo móvil. Se puede observar que la respuesta está estructurada de la siguiente manera: un objeto JSON determinado por los símbolos {} iniciales, luego un array determinado por los símbolos [], finalmente dentro del array, encontramos nuevamente un objeto pero este corresponde a nuestro modelo de datos, cabe señalar también que este objeto puede estar compuesto a su vez por otros objetos y/o colecciones.

Observando una estructura de respuesta con más datos, gracias a herramientas de visualización JSON, podemos ver que el formato mantiene su consistencia aún con composiciones más compleja.



**Figura 24. Estructura JSON con múltiples datos**

La requisición de los datos proviene del aplicativo cliente a través de una llamada http, con lo cual, el servidor envía una respuesta con los datos de localización de los lugares y la información asociada a los mismos para ser utilizada en el aplicativo Android.

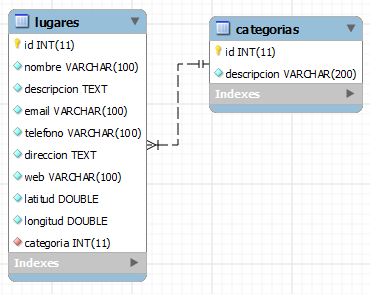
## 4.4 Servidor

El servidor para el aplicativo TuristaApp posee dos funciones principales: realizar el registro de Lugares de interés con la información asociada al mismo y enviar la información de esos lugares al aplicativo móvil Android.

En la composición del servidor funciona de modo conjunto los siguientes elementos:

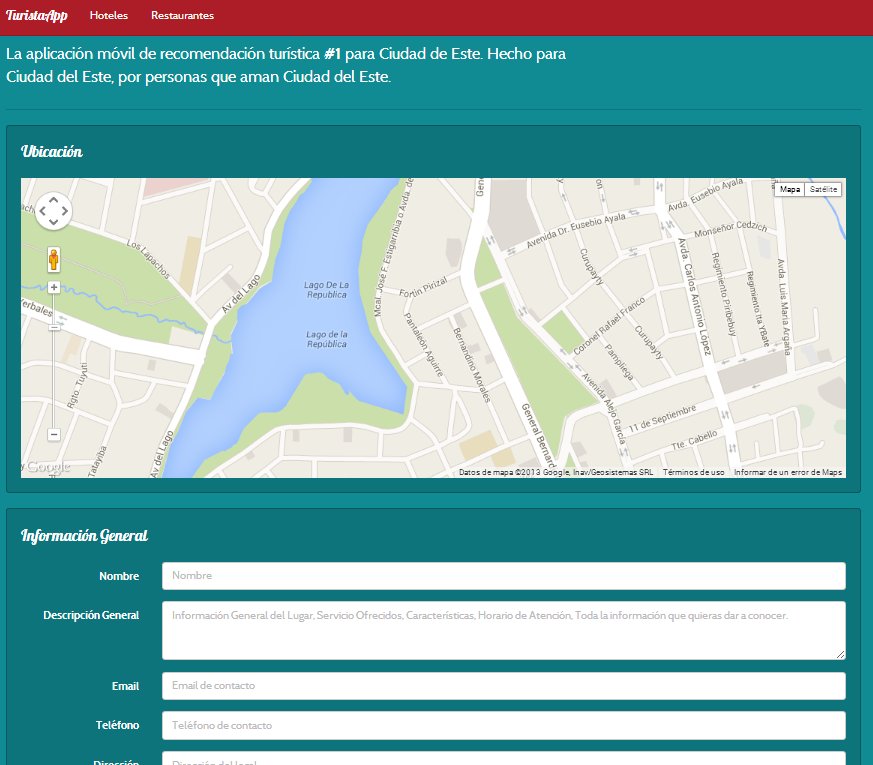
* Una base de datos que almacena toda la información de localización de los lugares.
* Una aplicación web para el registro de lugares.
* Un servicio rest para el envío de los datos en formato json.

Para la base de datos se utiliza MySQL. En la misma se ha implementado una base de datos llamada *turista* que cuenta con las tablas *lugares* y *categorías* que permiten almacenar los datos de lugares con la información asociada, así como la latitud y la longitud.



**Figura 25. Modelo relacional de la aplicación**

La aplicación web es el componente utilizado para registrar los lugares, guardando la información en la base de datos. La aplicación está construida utilizando la tecnología java, específicamente JSP y HTML para la creación de componentes visuales, SERVLET para el procesamiento de las requisiciones y JDBC para el acceso a los datos. Así también se utiliza Javascript para el acceso a Google Maps y la manipulación de los datos obtenidos a través de la API.



**Figura 26. Aplicación para registro de lugares.**

La utilización de Google Maps facilita enormemente el proceso de ubicación del lugar a registrar, lo cual se realiza con un clip en el mapa, indicando la posición cuyas coordenadas geográficas son automáticamente almacenadas.

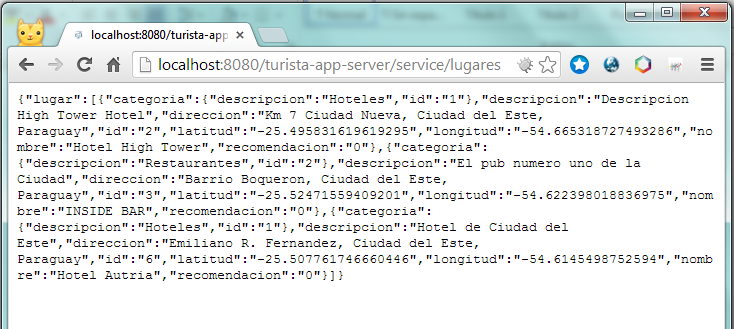


**Figura 27. Obtención de posición a través del evento click en el mapa**

Por su parte el Servicio REST permite el envío de los datos de lugares al aplicativo cliente utilizando el formato JSON sobre el protocolo HTTP. Como fue señalado anteriormente, el mismo está construido utilizando REST, el cual es un estilo arquitectónico para la construcción de aplicaciones interoperable entre diversos sistemas.

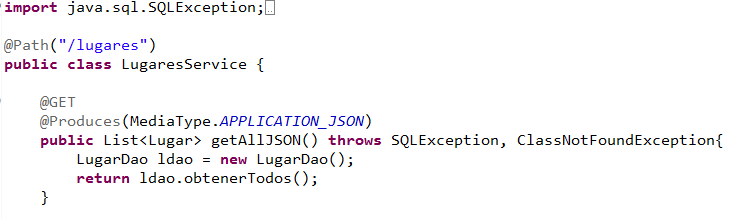
Una de las principales características de la arquitectura *rest* es la diferenciación que la misma establece sobre los recursos y las acciones que se pueden realizar sobre estos recursos. En el caso particular de nuestra aplicación, el recurso viene a ser los lugares que están registrados en nuestra base de datos. Sobre las acciones que se pueden realizar, lo que en *rest* se denominar verbos, toman como base los métodos HTTP y su correspondencia con las acciones en la manipulación de los datos. Sobre la conjugación de estos verbos se logra abstraer el acceso a esos datos y su manipulación, independientemente de la base datos, de la arquitectura e incluso del lenguaje de programación utilizado.

Como se dijo anteriormente el recurso publicado por nuestro servicio vienen a ser los *lugares* por lo cual se establece la url: *service/lugares* a partir de la cual se puede acceder a ese recurso. A partir de esa url se debe establecer el verbo correcto para realizar la acción de recuperar la colección de datos de lugares. Este verbo, establecido por *rest* es el método HTTP GET, por lo cual la realización de una petición GET sobre la url *service/lugares* devolverá una colección de objetos *lugar.*



**Figura 28. Petición HTTP GET sobre la url del servicio REST**

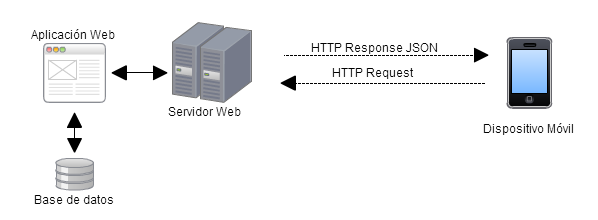
Para la implementación del servicio rest se utilizó la implementación de referencia de la especificación JAX-RS, la cual viene a ser la librería Jersey. Además la implementación del servicio dentro de la aplicación web permite la reutilización de componentes de la misma, como ser los componentes de acceso a datos.



**Figura 29. Implementación del Servicio Rest con Jersey**

## 4.5 Arquitectura

Una vez descrito, el funcionamiento general y las funcionalidades ofrecidas al usuario, tenemos el esquema del comportamiento del lado del servidor.



**Figura 30. Arquitectura del sistema TuristaApp**

En la figura puede apreciarse los elementos que interactúan para el funcionamiento del sistema:

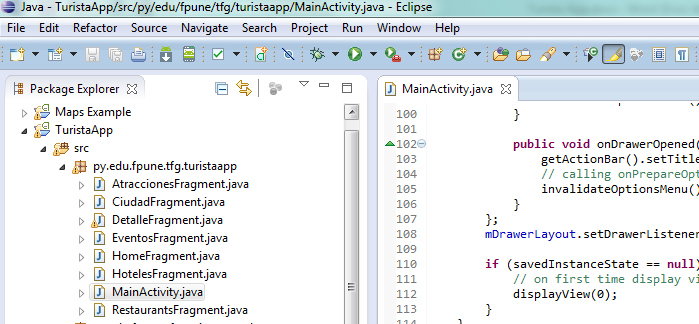
* Dispositivo móvil: dispositivo con el sistema operativo Android, donde está instalada la aplicación. Mediante una conexión de datos realiza el intercambio de información con el servidor utilizando el protocolo HTTP.
* Response JSON: a cada petición del aplicativo móvil, el servidor envía una respuesta en formato JSON con los datos de los hoteles y restaurantes registrados.
* Servidor Web: permanece siempre a la espera de requisiciones de parte de la aplicación móvil. Aloja la aplicación java que procesa las peticiones y devuelve la respuesta en formato JSON.
* Aplicación Web: una aplicación java que recibe las peticiones, consulta la base de datos y genera la respuesta en formato JSON. También se tiene implementada la funcionalidad para el registro de los hoteles y restaurantes.
* Base de datos: la base de datos almacena toda la información de localización de hoteles y restaurantes.

## 4.6 Desarrollo de la Aplicación Móvil

Para el desarrollo del aplicativo móvil Android se hizo uso de la herramienta Android SDK con el entorno de programación Eclipse y el plugín ADT, además de seguir las orientaciones de diseño recomendadas para una aplicación Android.

En ese sentido se dispuso especial atención en el aspecto visual de la aplicación móvil, dotándola de las características más eficientes en cuanto a facilitar el uso intuitivo de la misma.

También, en el sentido de disponer de las más adecuadas herramientas para la construcción de la aplicación y luego de un análisis de la presencia de versiones del sistema operativo Android en nuestro mercado, se decidió soportar desde la versión 4.0, ya que a partir de esa versión están presentes elementos que facilitan el diseño visual y la programación de funcionalidades.

\1

**Figura 31. Entorno de programación Eclipse**

## 4.7 Estructura del Proyecto

Como toda aplicación Android, la aplicación TuristaApp tienen, los componentes visuales definidos en archivos XML, los mismos son conocidos como *layout,* cada unocon su respectivo archivo java para la manipulación, este archivo extiende de la clase *Activity* y es a través del cual se maneja las funcionalidades de interacción con el usuario.

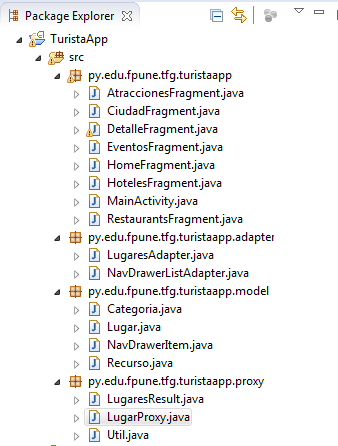
A partir de la versión de Android 3.0 se introduce el concepto de *Fragment*, lo que permite la construcción de componentes visuales reutilizables dentro de una *Activity*. Al igual que una Activity, un Fragment puede tener asociado un layout xml pero el Fragment debe ser siempre instanciado a través una Activity.

La aplicación TuristaApp hace uso intensivo de los componentes Fragment, para lo cual se dispone de un paquete exclusivo para los archivos java que extienden de Fragment *py.edu.fpune.tfg.turistaapp.fragment*, además dentro del mismo paquete, la aplicación cuenta con una sola clase Activity, cuya única responsabilidad es instanciar los Fragments adecuados según la solicitud del usuario en el menú de opciones. Esta Activity es el punto de entrada de la aplicación.

Además se cuenta con otro paquete para los componentes visuales, este es el paquete *py.edu.fpune.tfg.turistaapp.adapter* que agrupa las clases que extienden de *BaseAdapter* y son utilizadas para la construcción de elementos de la interfaz de usuario. Un *Adapter*, básicamente define el método de construcción de componentes visuales complejos como las listas seleccionables y los menús de navegación.

El siguiente paquete que tenemos es el paquete *py.edu.fpune.tfg.turistaapp.model* que contiene las clases que sirven de modelo de objeto de los datos, a ser manipulados por la aplicación.

Por último, pero no menos importante, tenemos el paquete *py.edu.fpune.tfg.turistaapp.proxy* que contiene las clases de comunicación con el servidor web, manipulación de los resultados y conversión a objetos java para uso de la aplicación.

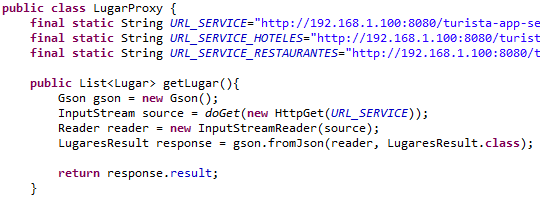


**Figura 32. Estructura de Paquete del Proyecto TuristaApp**

## 4.8 Conexión de Datos con el Servicio

Para la conexión con el servicio de datos y recuperar la información de los hoteles y restaurantes registrados en la base de datos, la aplicación móvil hace uso de los métodos definidos en la clase *LugarProxy* que encapsula la utilización de los métodos de la clase *org.apache.http.client.HttpClient* presente en Android y través del cual se realiza todas las conexiones http.

Además de la conexión con el servidor, la clase *LugarProxy* hace uso de la librería Gson para la conversión de los datos obtenidos en formato Json a objetos java de la clase modelo.



**Figura 33. Clase LugarProxy.java**

## 4.9 Acceso a Google Maps

La utilización del popular servicio Google Maps es una de las posibilidades más atractivas de Android. En efecto existe un gran número de las aplicaciones que utilizan estas bibliotecas con fines distintos.

El paquete que incluye todas las clases relacionadas con la carga y el manejo del servicio es el paquete *com.google.android.maps.*

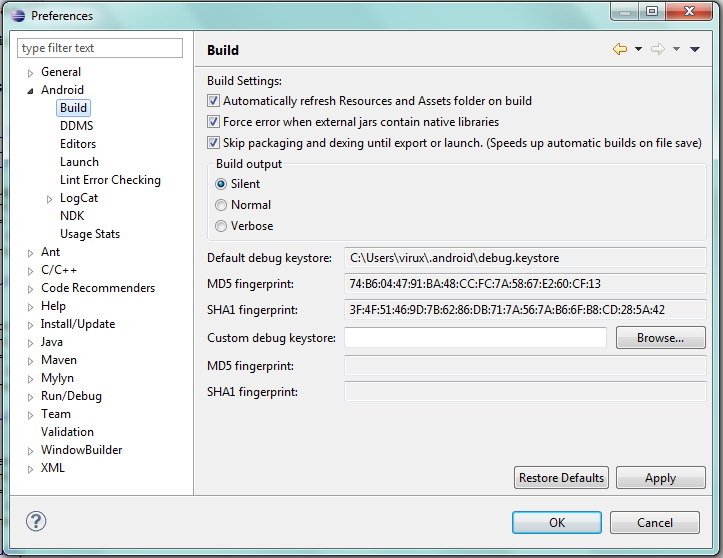
Un paso previo a la utilización de la API de Google Maps es el registro de la aplicación y la aceptación de los términos y condiciones de uso. Este registro se realiza a través de una clave denominada en Android como *API Key.*

Una vez procedido al registro, la utilización de mapas en nuestra aplicación no requiere más que unas cuantas llamadas de las clases pertinentes.

### 4.9.1 Obtención de la API Key

Para utilizar los servicios de Google Maps en una aplicación Android es imprescindible realizar unos pasos previos que implican el registro del desarrollador y la aceptación de las condiciones de uso. De esta forma, Google se asegura que en todo momento se hará un uso adecuado y apropiado de los servicios y datos que nos van a ser proporcionados a través de Google Maps.

Toda aplicación en Android está acompañada de un certificado que asegura su autoría y la vincula con su desarrollador. Si se utiliza el *plugin*de Android para Eclipse, todas las aplicaciones están firmadas por un certificado al que se puede calificar de prueba y que permite al desarrollador crear y probar sus aplicaciones sin más espera. Este certificado se obtiene a través del fichero de clave de pruebas *debug.keystore*, presente por defecto en el Android SDK.

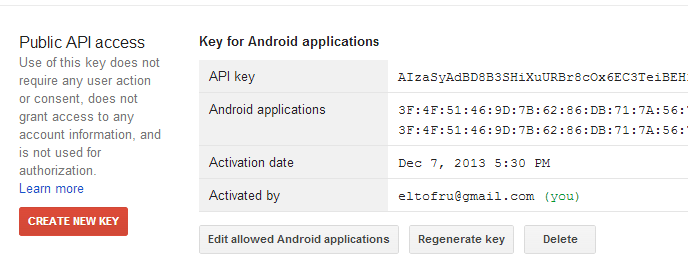
****

**Figura 34. Certificado debug.keystore por defecto**

El registro para tener acceso a Google Maps se hace a través de dicho certificado. En Primero lugar se ha de obtener la Huella Digital MD5 del certificado a utilizar para firmar la aplicación donde se hará uso del mapa. Utilizando Eclipse esta huella se puede obtener fácilmente como se muestra en la Figura 34.

A continuación se debe registrar la Huella MD5 obtenida en la página *https://code.google.com/apis/console.* Es necesario disponer de una cuenta Google para el efecto.

Se proporciona entonces una clave alfanumérica, llamada *API Key.* Esta es la que se debe utilizar para acceder a los servicios de Google Maps, a través de nuestra aplicación, siendo además única y exclusiva para la aplicación firmada con el certificado utilizado.



**Figura 35. API Key proporcionada por Google**

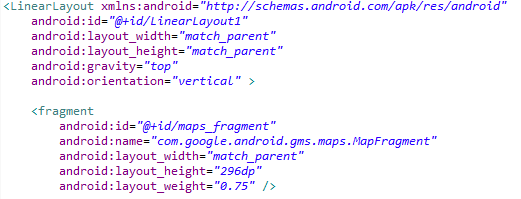
### 4.9.2 Mostrar un mapa

Una vez que se dispone de una *Api Key* con la que se puede acceder a los servicios de Google Maps, no resta más que hacer las llamadas pertinentes a las clases del paquete *com.google.android.maps.*

A partir de la versión 2.0, la API de Google Maps para Android presenta una facilidad substancial en los pasos necesarios para instanciar y manipular un mapa. Para poder visualizar un mapa de Google Maps, es necesario realizar dos pasos:

* Registrar la *API Key* en el archivo de configuración de la aplicación Android Manifest.
* Añadir el componente *MapFragment* dentro del *layout* como componente visual de la aplicación.

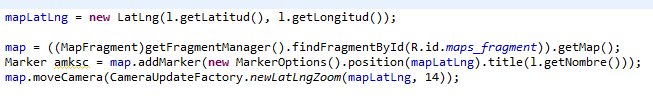
Así también es necesario habilitar el permiso de acceso a internet tanto para la comunicación con el servidor de datos de la aplicación TuristaApp, como para el acceso a los servicios de Google Maps.



**Figura 36. Componente MapFragment dento de un layout.**

Para manipular el mapa como un objeto a través del lenguaje de programación es necesario referenciarlo como un objeto, esto se logra a través de la clase *Fragment*a través del método *findFragmentById(R.id.maps\_fragment)* que recibe como parámetro una referencia del identificador del MapFragment en archivo *layout****.***

Una vez que se tiene la referencia del objeto del mapa se le puede pasar un objeto del tipo *com.google.android.gms.maps.model.Marker* con los datos de posicionamiento del punto que se quiere ubicar en el mapa, o sea su latitud y su longitud. También se pude especificar la posición donde se muestra el punto en pantalla y el nivel de acercamiento del mapa.



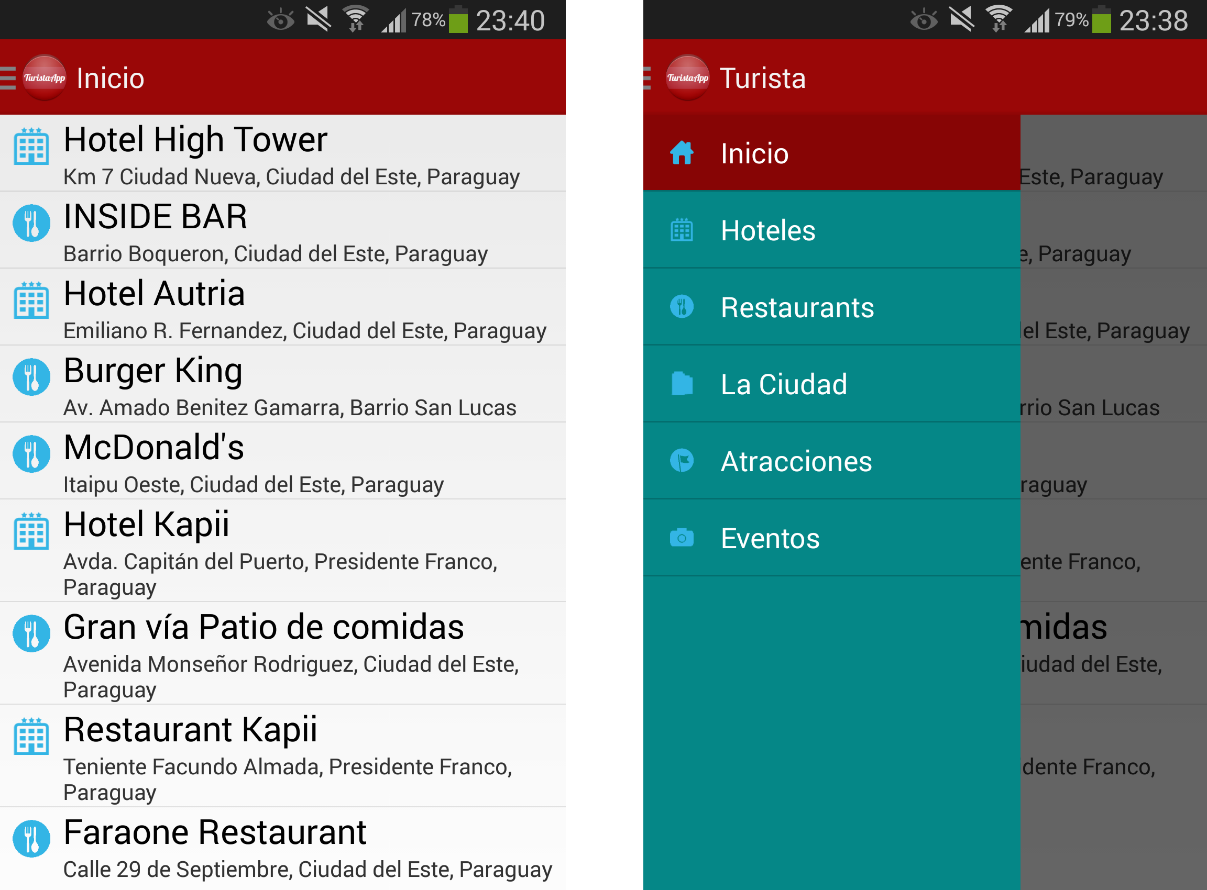
**Figura 37. Manipulación del Mapa para la ubicación de un punto geográfico.**

## 4.10 Menú de Opciones y Navegación.

Con relación al menú de opciones y la navegación entre las mismas, se puso especial cuidado en optimizar la experiencia de usuario de acuerdo a la guía de diseños sugerida para aplicaciones Android[[26]](#footnote-26).

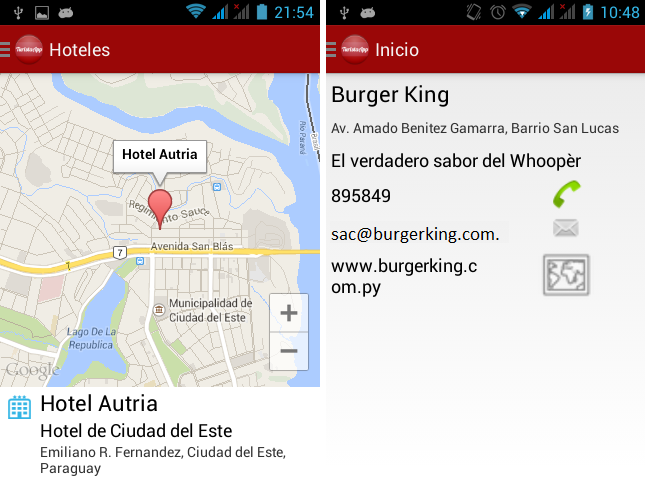
La aplicación inicia mostrando al usuario una lista de los últimos lugares registrados, siguiendo la recomendación de entregar al usuario la información lo más rápido posible. Esta lista puede ser estratégicamente utilizada para presentar recomendaciones y lugares destacados para el usuario.

Para la navegación de opciones se utiliza el componente *DrawerLayout,* el cual tiene la característica de poder ser utilizado como un menú deslizable, optimizando el uso de la pantalla del dispositivo móvil.



**Figura 38. Pantalla inicial y menú de navegación**

A partir del menú de opciones se puede elegir presentar los lugares por categoría. Al hacer seleccionar con un toque uno de los lugares se muestra el mapa con la ubicación del mismo, otro toque sobre el nombre del local en esa pantalla despliega la información asociada al mismo, todo esto teniendo siempre en cuenta el uso intuitivo de la aplicación.



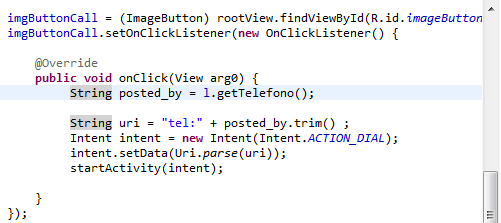
**Figura 39. Ubicación y detalle del lugar seleccionado**

## 4.11 Integración con funcionalidades del Teléfono

La aplicación *TuristaApp* tiene como principal características de integrar los datos de hoteles y restaurantes con las funcionalidades del teléfono.

Esta característica permite que a partir del dato, como ser el número de teléfono, se realice una actividad asociada al teléfono, en este caso llamar al número de teléfono.

Para esta finalidad, Android define un objeto *Intent* que permite invocar la funcionalidad apropiada para la acción que se desea realizar.



**Figura 40. Fragmento de código para lanzar un Intent**

La principal ventaja del uso de *Intent* es el hecho de permitir ampliar fácilmente las funcionalidades de nuestra aplicación integrándola de manera fácil a las funcionalidades del teléfono e incluso funcionalidades de otras aplicaciones.

En la aplicación se hace uso de los *Intent* para añadir las siguientes funcionaliades:

Llamar al número telefónico

Permite enviar el número telefónico asociado al lugar a la aplicación de llamadas para que el usuario pueda entrar en contacto a través de la llamada telefónica.

**Redactar un email**

Permite abrir la aplicación de email con la dirección de correo asociada al lugar para que el usuario pueda redactar un email.

**Visitar página web**

Permite abrir la dirección URL asociada al lugar en el navegador web para buscar mayor información.

**Compartir información**

Permite compartir la información asociada al lugar a través de cualquier aplicación preparada para soportar el Intent.*ACTION\_SEND*, como ser compartir la información en los aplicativos de redes sociales o enviarla por mensaje de texto.

**Trazar ruta**

Permite usar la funcionalidad de trazar la ruta desde la posición actual del usuario, obtenida a través del GPS hasta el lugar seleccionado.

## 4.12 Dificultades en el desarrollo del Aplicativo

Durante el desarrollo del Aplicativo se han encontrado dificultades. A continuación se detallan algunos de los problemas:

* El modelo del proceso, exige que los datos de hoteles y Restaurantes sean previamente registrados en la base de datos, de modo que no todos los hoteles y restaurantes podrán ser encontrados.
* Cambios en la utilización de la API de Google Maps, es decir, la migración de la versión 1 a la versión 2 de la API de Google Maps para Android.
* Fragmentación y versión soportada por los nuevos elementos utilizados para una aplicación Android: Fragment, ActionBar, Navigation, y tratamiento ANR para operaciones de conexión de datos.
* Versión mínima del Sistema soportado, considerando el mercado de teléfonos en Ciudad del Este, y las facilidades que posibilitan las versiones más actuales del sistema, se decide soportar las versiones del sistema operativo a partir de la 4.0 en adelante.

**CAPÍTULO V**

**CONCLUSIONES Y SUGERENCIAS**.

***Hay un método seguro para mejorar nuestra situación, y consiste en aumentar nuestra capacidad.*** *(Demócrito)*

# CAPÍTULO V: CONCLUSIONES Y SUGERENCIAS

En este capítulo se presentan las conclusiones finales respecto al trabajo.

## 5.1 Conclusiones del Sistema Android

### Ventaja del sistema Android

1. El código de Android es abierto: Google liberó Android bajo licencia Apache. Cualquier persona puede realizar una aplicación para Android.
2. Hoy día hay más de 650.000 aplicaciones disponibles para teléfonos Android, aproximadamente 2/3 son gratis. Además la libertad de código permite adaptar Android a bastantes otros dispositivos además de teléfonos celulares. Está implantado en Tablets, GPS, relojes, microondas… incluso hay por internet una versión de Android para PC.
3. El sistema Android es capaz de hacer funcionar a la vez varias aplicaciones y además se encarga de gestionarlas, dejarlas en modo suspensión si no se utilizan e incluso cerrarlas si llevan un periodo determinado de inactividad. De esta manera se evita un consumo excesivo de batería. Esta es una de sus mayores ventajas por la rapidez con la que carga una aplicación abierta previamente. Por ejemplo abro Google Maps y en un momento la aplicación localiza mi posición en el mapa.

Por último cabe destacar que la principal ventaja de la plataforma Android es su disponibilidad libre y gratuita para cualquier desarrollador que quiera iniciarse en el mundo de la programación móvil, ofreciendo un entorno de desarrollo abierto con potentes herramientas y sobre todo gratuitas, además considerando que la tendencia clara de la evolución de los dispositivos móviles donde su popularidad en uso y acceso a servicios de datos es cada vez mayor incluso por sobre los dispositivos como PCs y Nootebook resultará muy relevante el conocimiento sobre cómo desarrollar aplicaciones móviles en un ambiente con mucho futuro, como lo es Android.

### Desventajas del sistema Android

1. A pesar de ser una ventaja el ser un sistema multitarea: El hecho de tener varias aplicaciones abiertas hacen que el consumo de la batería aumente y como no todas las aplicaciones Android las cierra hay que instalar una aplicación para que las cierre. En el Google Play de Android hay una buena cantidad de aplicaciones para este fin, así que el problema es solucionable pero debería venir pre instalado de fábrica.
2. Duración de la batería: la batería en un celular Android se agota muy rápido. Utilizando las aplicaciones de manera moderada la batería puede llegar a durar más, pero para un usuario que usa mucho sus aplicaciones la batería no tiende a durar, lo que se puede solucionar con algunas aplicaciones, pero volvemos a lo mismo no viene pre instalada de fábrica, hace falta una aplicación externa al sistema Android para optimizar mejor la batería.

## 5.2 Comparaciones con servicios de Búsquedas de hoteles y restaurantes de Ciudad del Este.

### Buscador Google.

* + Al realizar una búsqueda con el término “hoteles de ciudad del este” provee de informaciones muchas veces innecesarias y/o no vinculadas a la información que se busca.
  + Trae informaciones de muchas ciudades y países del mundo, impidiendo el acceso rápido a la información.
  + Las direcciones de los hoteles o restaurantes, no son siempre de Ciudad del Este.
  + Las informaciones que se busca, no están organizadas en un mismo patrón, pudiendo variar de página en página, según el diseño de cada una.
  + Posee informaciones muy generalizadas y amplias.

### Página especializada de guía: [www.guía](http://www.guía)deleste.com

* Reducido número de resultados en la búsqueda de hoteles y restaurantes.
* Amplio campo de información, no específico para hoteles y restaurantes.
* No cuenta con servicios de mapas que faciliten la ubicación de los lugares.
* No esta optimizada para su uso en dispositivos móviles.

### Servicio Guía telefónica 112 Copaco.

* Se necesita de teléfono de línea baja para realizar las llamadas.
* No cuenta con abundantes informaciones.
* No es un servicio, en el que se pueda consultar las 24 horas.

## 5.3 Conclusiones del Aplicativo Implementado en el proyecto.

A partir de las pruebas del aplicativo implementado en este proyecto se han obtenido como resultado los siguientes puntos:

* Implementar una interfaz de usuario en el Dispositivo Móvil Android, utilizando XML.
* Comunicar datos desde un servidor externo en la Aplicación Móvil a través de un servicio con arquitectura REST, con formato JSON.
* Utilizar la APIS de Google Maps en una Aplicación Móvil para teléfonos Android.
* Crear una aplicación que proporcione información útil y de interés para visitantes y residentes de Ciudad del Este.

## 5.4 Beneficios obtenidos

* Disponibilizar rápidamente la ubicación de Hoteles y Restaurantes de la Ciudad a los usuarios
* Optimizar la realización de traslados a lugares de la Ciudad.
* Representa una oportunidad económica para los Hoteles y Restaurantes a través del cual dar a conocer estratégicamente su ubicación, productos y servicios ofrecidos.
* El beneficio del aprendizaje y la experiencia real de desarrollar una aplicación móvil en la plataforma Android considerando el futuro promisor de la misma.

## 5.5 Sugerencias para futuros trabajos.

* Integración de un modelo de Redes Sociales para la aplicación, donde el usuario puede indicar, los lugares que visita, comentar los mismos y compartir esa información con sus amigos.
* Integración de elementos multimedia para la información, vídeo, fotografías.
* Implementar una capa de seguridad para la API REST de la aplicación.
* Crear e implementar un modelo de negocios para las personas interesadas en publicar lugares de interés como estrategia publicitaria.
* Determinar un modelo de selección de informaciones relevante que sirvan para recomendaciones para los usuarios.

**REFERENCIAS BIBLIOGRÁFICAS.**

# REFERENCIAS

[1] Android <http://developer.android.com/guide/basics/what-is-android.html> [Fecha de Consulta; 5/11/2012]

[2] Georeferenciamiento http://es.wikipedia.org/wiki/Georreferenciaci%C3%B3n

[Fecha de Consulta; 16/12/2012]

[3] Interfaz de programación de aplicaciones (Application Programming Interface)

http://es.wikipedia.org/wiki/Interfaz\_de\_programaci%C3%B3n\_de\_aplicaciones [Fecha de Consulta; 29/11/2012]

[4] Google Maps http://code.google.com/intl/es-ES/apis/maps/index.html [Fecha de Consulta; 29/11/2012]

[5] Canalys, web de análisis de mercados tecnológicos. Informe sobre el ejercicio 2012. [http://www.canalys.com/pr/2013/r2013021.htm. [Fecha de Consulta; octubre de 2013]

[6] “How to Do Everything with Windows Mobile”, Frank McPherson. [Fecha de Consulta; agosto de 2013]

.

[7] MSDN, MS Windows CE Operating System Architecture.

<http://www.microsoft.com/technet/archive/wce/plan/modular.mspx?mfr=true>. [Fecha de Consulta; octubre de 2013]

[8] Noticias de Google en español. Artículo “Google cumple 10 años”. http://google.dirson.com/post/4090-google-cumple-10/

[Fecha de Consulta; noviembre de 2013]

[9]Público, edición digital. <http://www.publico.es/culturas/125240/google/obtiene/premio/principe/asturias/comunicacion/humanidades>. [Fecha de Consulta; noviembre de 2013]

[10] Servicios de Google Inc.. http://www.google.es/options/ [Fecha de Consulta; agosto de 2013]

[11] Condiciones de la Licencia Apache versión 2.0.

http://www.apache.org/licenses/LICENSE-2.0.html [Fecha de Consulta; noviembre de 2013]

[12] Página oficial del proyecto Android. http://code.google.com/android/index.html [Fecha de Consulta; septiembre de 2013]

[13] Página oficial de Open Handset Alliance.

<http://www.openhandsetalliance.com/index.html> [Fecha de Consulta; septiembre de 2013]

[14] Descripción de la arquitectura de Android, por Google.

http://es.youtube.com/watch?v=QBGfUs9mQYY [Fecha de Consulta; noviembre de 2013]

[15] SDK de Android. Web del proyecto Android.

http://code.google.com/android/download.html [Fecha de Consulta; octubre de 2013]

[16] Descargas de distintos paquetes de Eclipse. Web oficial de la plataforma Eclipse.

http://www.eclipse.org/downloads/[Fecha de Consulta; octubre de 2013]

[17] UML http://www.uml.org/ [Fecha de Consulta; 12/01/2013]

[18] Java http://java.com/[Fecha de Consulta; 20/02/2013]

[19] Google Inc: http://es.wikipedia.org/wiki/Google

[Fecha de Consulta; 15/03/2013]

[20] Computer Word. BI salta a Dispositivos Móviles y acelera negocios http://computerworld.uol.com.br/tecnologia/2012/11/13/bi-salta-para-dispositivos-moveis-e-acelera-negocios/[Fecha de Consulta; 28/11/2013]

[21] Google Maps For Android: http://developer.android.com/resources/tutorials/views/hello-mapview.html [Fecha de Consulta: 02/05/2013]

[22] Android Reference http://developer.android.com/reference/packages.html

[Fecha de Consulta: 02/05/2013]

[23] Evolución de los Teléfonos http://www.webadictos.com.mx/2011/10/14/la-evolucion-de-los-telefonos-celulares-infografia/ [Fecha de Consulta: 08/05/2013]

[24] Descripción del ciclo de vida de una aplicación Android, por Google.

<http://es.youtube.com/watch?v=fL6gSd4ugSI> [Fecha de Consulta; octubre de 2013]

[25] Android SDK http://developer.android.com/sdk/index.html

[Fecha de Consulta; 29/10/2013]

[26] Google Maps <http://es.wikipedia.org/wiki/Google_Maps> [Fecha de Consulta; octubre de 2013]

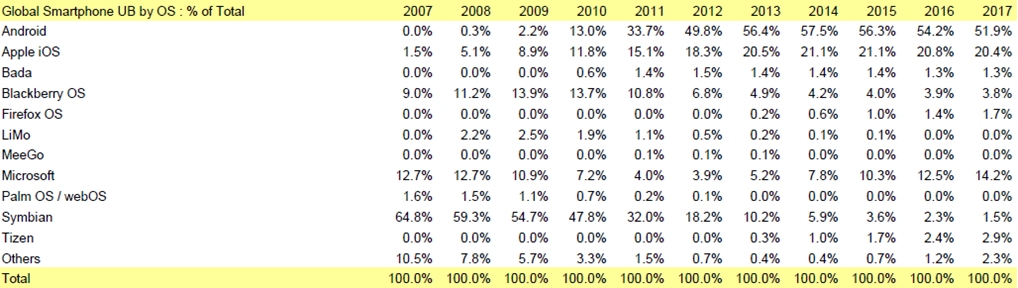
[27] Webservice http://www.w3schools.com/webservices/ws\_intro.asp[Fecha de Consulta; octubre de 2013]

[28] Rest: http://es.wikipedia.org/wiki/Representational\_State\_Transfer[Fecha de Consulta; noviembre de 2013]

**ANEXOS.**

# Anexos

## ANEXO 1: Tendencias en dispositivos móviles para el año 2014.



## Anexo 2: Términos

* **Android:** plataforma de software de código abierto que incluye un sistema operativo para dispositivos móviles basado en Linux. Desarrollado por Google y por la Open Handset Aliance.
* **Apache 2.0:** «Apache» es un acrónimo de «a patchy server» -un servidor de remiendos-, es decir un servidor construido con código preexistente y piezas y parches de código. Es la auténtica «kill app» del software libre en el ámbito de los servidores y el ejemplo de software libre de mayor éxito, por delante incluso del kernel Linux. Desde hace años, más del 60% de los servidores web de Internet empleanApache. desde master magazine [http://www.mastermagazine.info/ termino/3866.php#ixzz2nZZzXO96](http://www.mastermagazine.info/%20termino/3866.php#ixzz2nZZzXO96).
* **API:** siglas de Application Programming Interface, en español Interfaz de Programación de Aplicaciones. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos, representando una capa de abstracción para el desarrollador.
* **Background:** representa un proceso que se ejecuta con pocos recursos, que no requiere interacción directa con el usuario y que existe sin el conocimiento de este.
* **Biblioteca:** agrupación de código y datos que proporcionan servicios a programas independientes, pasando a formar parte de éstos. Permiten la distribución de funcionalidades y la construcción modular. También conocido como librería, por su similitud con el inglés library.
* **Dalvik:** nombre de la máquina virtual utilizada por el sistema operativo Android.

Dalvik está específicamente adaptada a las características de rendimiento de un dispositivo móvil y trabaja con ficheros de extensión “.dex”, obtenidos desde el bytecode de Java.

* **Dispositivo móvil:** aparato electrónico que es de reducido tamaño, cuenta con cierta capacidad tanto para la computación como para el almacenamiento de datos y cuenta con elementos de E/S básicos, por ejemplo pantalla y/o teclado.
* **Eclipse**: Consiste en una comunidad de código abierto cuyos proyectos son enfocados en la construcción de una plataforma de desenvolvimiento compuesta por ambientes de trabajo extensibles y herramientas para la construcción, desenvolvimiento y manipulación de programas durante todo su ciclo de vida.
* **Framework**: término con el que se define un amplio conjunto de elementos que permite desarrollar y organizar software utilizando un determinado lenguaje, sistema y tecnología. Habitualmente incluye bibliotecas y programas de desarrollo.
* **GPS:** siglas de Global Positioning System, en español Sistema de Posicionamiento

Global. Es un sistema global de navegación que, mediante satélites, permite ubicar un objeto en la superficie terrestre con una precisión que va desde varios metros a centímetros.

* **HTML** es un estándar que, en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web.
* **JSON** es un formato ligero de intercambio de datos  basado en un subconjunto del [Lenguaje de Programación JavaScript](http://javascript.crockford.com/), [Standard ECMA-262 3rd Edition - Diciembre 1999](http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf). JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. <http://www.json.org/json-es.html>
* **Kernel:** parte fundamental de un sistema operativo, responsable de facilitar acceso seguro al hardware, gestionar recursos y hacer llamadas al sistema. También conocido como núcleo.
* **Latitud**: distancia angular entre el ecuador y un punto de la superficie del planeta. Se mide en grados entre 0 y 90.
* **Longitud:** distancia angular entre el meridiano y un punto de la superficie del planeta. Se mide en grados entre 0 y 360.
* **Máquina virtual**: representa un software que emula el comportamiento de una determinada arquitectura o que permite adaptar un código fuente a las características de la máquina nativa.
* **MOAP:** siglas de Mobile Oriented Applications Platform, una plataforma software basada en Symbian para los teléfonos del fabricante japonés FOMA.
* **OHA**: siglas de Open Handset Alliance, es un grupo de empresas que se unieron para acelerar la innovación lideradas por Google, proporcionando al proyecto, conocimiento tecnológico para el desarrollo de dispositivos móviles y sus aplicaciones, además para dispositivos con soporte en la plataforma Android
* **Plug-in**: componente de software que se relaciona y ejecuta con otro para aportarle una función nueva y generalmente muy específica.
* **Página Web**: Una **página web** (o página electrónica, según el término recomendado por la [R.A.E.](http://es.wikipedia.org/wiki/R.A.E.)) es el nombre de un [documento](http://es.wikipedia.org/wiki/Documento) o información electrónica capaz de contener texto, sonido, vídeo, programas, enlaces, imágenes, y muchas otras cosas, adaptada para la llamada [*World Wide Web*](http://es.wikipedia.org/wiki/World_Wide_Web), y que puede ser accedida mediante un [navegador](http://es.wikipedia.org/wiki/Navegador_web). Esta información se encuentra generalmente en formato [HTML](http://es.wikipedia.org/wiki/HTML) o [XHTML](http://es.wikipedia.org/wiki/XHTML), y puede proporcionar navegación (acceso) a otras páginas web mediante [enlaces](http://es.wikipedia.org/wiki/Hiperv%C3%ADnculo) de [hipertexto](http://es.wikipedia.org/wiki/Hipertexto). Las páginas web frecuentemente también incluyen otros recursos como ser [hojas de estilo en cascada](http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada), [guiones](http://es.wikipedia.org/w/index.php?title=Script_del_lado_del_cliente&action=edit&redlink=1) (*scripts*), [imágenes digitales](http://es.wikipedia.org/wiki/Imagen_digital), entre otros.
* **Proceso**: un proceso es un programa en ejecución, y representa la unidad de procesamiento básica gestionada por el sistema operativo.
* **SDK**: siglas de Software Development Kit, en español Kit de Desarrollo de Software. Constituye un conjunto de herramientas que permiten a un desarrollador crear aplicaciones para una determinada plataforma o lenguaje.
* **Sistema operativo**: programa cuya finalidad principal es simplificar el manejo y explotación de un elemento con capacidad computacional, gestionando sus recursos, ofreciendo servicios a las demás aplicaciones y ejecutando mandatos del usuario.
* **XML:** siglas de Extensible Markup Language, en español Lenguaje de Marcado Extensible. Representa un lenguaje estándar que, mediante el uso de etiquetas y atributos, permite expresar e intercambiar fácilmente estructuras de datos.

1. Android: plataforma de software de código abierto que incluye un sistema operativo para dispositivos móviles basado en Linux. Desarrollado por Google y por la Open Handset Aliance. [↑](#footnote-ref-1)
2. Georeferenciamiento: identificación de la posición geográfica real de un objeto o persona. [↑](#footnote-ref-2)
3. Longitud: distancia angular entre el meridiano 0º y un punto de la superficie del planeta. [↑](#footnote-ref-3)
4. Latitud: distancia angular entre el ecuador y un punto de la superficie del planeta. Se mide en grados entre 0 y 90. [↑](#footnote-ref-4)
5. API: siglas de Application Programming Interface, en español Interfaz de Programación de Aplicaciones. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos, representando una capa de abstracción para el desarrollador. [↑](#footnote-ref-5)
6. Windows CE: (conocido oficialmente como Windows Embedded Compact y anteriormente como Windows Embedded CE, también abreviado como WinCE) es un [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo) desarrollado por [Microsoft](http://es.wikipedia.org/wiki/Microsoft) para [sistemas embebidos](http://es.wikipedia.org/wiki/Sistema_embebido). Windows CE no debe confundirse con [Windows Embedded Standard](http://es.wikipedia.org/w/index.php?title=Windows_Embedded_Standard&action=edit&redlink=1), que es un sistema basado en [Windows NT](http://es.wikipedia.org/wiki/Windows_NT); Windows CE está desarrollado independientemente. [↑](#footnote-ref-6)
7. Kernel: es un [software](http://es.wikipedia.org/wiki/Software) que constituye una parte fundamental del [sistema operativo](http://es.wikipedia.org/wiki/Sistema_operativo). Es el principal responsable de facilitar a los distintos programas [acceso seguro](http://es.wikipedia.org/wiki/Seguridad_inform%C3%A1tica) al [hardware](http://es.wikipedia.org/wiki/Hardware) de la [computadora](http://es.wikipedia.org/wiki/Computadora) o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema. [↑](#footnote-ref-7)
8. Página web: es el nombre de un [documento](http://es.wikipedia.org/wiki/Documento) o información electrónica capaz de contener texto, sonido, vídeo, programas, enlaces, imágenes, y muchas otras cosas, adaptada para la llamada [World Wide Web](http://es.wikipedia.org/wiki/World_Wide_Web). [↑](#footnote-ref-8)
9. <http://www.publico.es/culturas/125240/google/obtiene/premio/principe/asturias/comunicacion/humanidades>. [↑](#footnote-ref-9)
10. Servicios de Google Inc. http://www.google.es/options/ [↑](#footnote-ref-10)
11. Apache 2.0: es una licencia de [software libre](http://es.wikipedia.org/wiki/Software_libre) creada por la [Apache Software Foundation](http://es.wikipedia.org/wiki/Apache_Software_Foundation) (ASF). http://es.wikipedia.org/wiki/Apache\_License [↑](#footnote-ref-11)
12. SDK: siglas de Software Development Kit, en español Kit de Desarrollo de Software. Constituye un conjunto de herramientas que permiten a un desarrollador crear aplicaciones para una determinada plataforma o lenguaje. [↑](#footnote-ref-12)
13. OHA: siglas de Open Handset Alliance, es un grupo de empresas que se unieron para acelerar la innovación lideradas por Google. [↑](#footnote-ref-13)
14. http://es.youtube.com/watch?v=QBGfUs9mQYY. [↑](#footnote-ref-14)
15. http://code.google.com/android/download.html [↑](#footnote-ref-15)
16. Eclipse: Consiste en una comunidad de código abierto cuyos proyectos son enfocados en la construcción de una plataforma de desenvolvimiento compuesta por ambientes de trabajo extensibles y herramientas para la construcción, desenvolvimiento y manipulación de programas durante todo su ciclo de vida. [↑](#footnote-ref-16)
17. Webservice  tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores. [↑](#footnote-ref-17)
18. Representational State Transfer (REST) es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. [↑](#footnote-ref-18)
19. Caso de Uso: Los casos de uso son una técnica para especificar el comportamiento de un sistema. http://www-2.dc.uba.ar/materias/isoft1/2001\_2/apuntes/CasosDeUso.pdf [↑](#footnote-ref-19)
20. Ingeniería de Software: es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software. <http://www.computer.org/portal/web/swebok> [↑](#footnote-ref-20)
21. UML: Unified Modeling Language es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. [↑](#footnote-ref-21)
22. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. <http://www.json.org/json-es.html> [↑](#footnote-ref-22)
23. Jersey es la implementación de referencia de calidad de producción de Sun para JSR 311: JAX-RS: The Java API for RESTful Web Services. https://jersey.java.net/ [↑](#footnote-ref-23)
24. JAX-RS: Java API for RESTful Web Services es una [API](http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones) del [lenguaje de programación Java](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) que proporciona soporte en la creación de [servicios web](http://es.wikipedia.org/wiki/Servicio_web) de acuerdo con el ​​estilo arquitectónico REST. https://jax-rs-spec.java.net/ [↑](#footnote-ref-24)
25. GSON: (también conocido como Google Gson) es una [biblioteca](http://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)) de [código abierto](http://es.wikipedia.org/wiki/C%C3%B3digo_abierto) para el [lenguaje de programación Java](http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java) que permite la [serialización](http://es.wikipedia.org/wiki/Serializaci%C3%B3n) y deserialización entre [objetos](http://es.wikipedia.org/wiki/Objeto_(programaci%C3%B3n)) Java y su representación en notación [JSON](http://es.wikipedia.org/wiki/JSON). https://code.google.com/p/google-gson/ [↑](#footnote-ref-25)
26. Android Design: <http://developer.android.com/design/index.html> [↑](#footnote-ref-26)