

UNIVERSIDAD CATOLICA ARGENTINA

FACULTAD DE INGENIERÍA



TRABAJO FINAL DE LA CARRERA DE
ESPECIALIZACIÓN EN INGENIERÍA DE
SOFTWARE

PUNTO DE VENTA MÓVIL
DESARROLLO DE SOFTWARE EN DISPOSITIVO
MÓVIL SOBRE SISTEMA OPERATIVO ANDROID

AUTOR: MAURICIO CALGARO

DIRECTOR: EMILIO GUTTER

2012

Agradecimientos

A mis padres	por haberme brindado todo el apoyo necesario para poder perfeccionarme en mis estudios.
A Sergio Villagra	por los consejos que me dió sobre su experiencia con respecto a los gateways de pagos.
A Steve Shipman	por enviarme un lector de tarjetas de créditos para poder realizar las pruebas necesarias en el sistema.
A Emilio Gutter	por compartir parte de su tiempo en el asesoramiento de este trabajo final de especialización.
A todos mis colegas	del curso de especialización que lograron hacer que sea una cursada 2011 inolvidable y muy divertida.
A los profesores	con los que compartimos unas agradables tardes y noches conociendo sus experiencias e intercambiando opiniones y conocimientos referentes a la tecnología informática.

Dedicatoria

Quiero dedicar este Trabajo Final de Carrera principalmente a tres personas muy importantes en mi vida:

A mi mamá “la Trigi”, por todo el cariño que me brindó en la vida y por el gran apoyo recibido durante todos los años de mi carrera universitaria.

A mi papá “el Billo”, por ser un excelente padre que siempre hizo todo lo posible para que nunca me faltara nada en la vida.

A mi abuela María, por todo el cariño que me brinda y que me brindó cuando yo era chico y por estar siempre deseándome lo mejor para mi y para mi futuro.

También quiero dedicárselos a todos mis amigos de Posadas (Misiones) por haber disfrutado junto a ellos muchos momentos agradables y lindos de la vida que estarán siempre en mis recuerdos.

Por último, este Trabajo Final de Carrera (TFC) está dedicado para todas aquellas personas que sientan pasión y entusiasmo por la tecnología.

Resumen

La elaboración de este Trabajo Final de Especialización (TFE) consiste en el desarrollo de un sistema integral de Terminal de Punto de Venta (TPV).

Este sistema tiene como objetivo manejar las transacciones de ventas diarias de un local comercial, almacenando datos sobre los productos vendidos, clientes, formas de pagos, stock, etcétera.

La elaboración de este TFE está compuesto por el desarrollo de tres aplicaciones. La primer aplicación consiste en el TPV implementado en un Dispositivo de Pantalla Táctil (DPT) de 7 pulgadas cuyo sistema operativo es Android™. La segunda aplicación funciona bajo un Dispositivo de Telefonía Celular (DTC) también funcionando con el sistema Android™ y cuya finalidad es la de poder brindar a los clientes un dispositivo donde pueda funcionar como una Terminal de Ingreso de Datos (TID) al sistema. La última de las aplicaciones elaboradas consiste en un Servicio de Manejo de datos (SMD) que se encuentra ubicado en un servidor **Tomcat** conectado a una base de datos **MySQL** y cuya función es la de recibir, procesar y enviar información desde y hacia el TPV.

La conexión entre el TPV y el servidor se realiza mediante Servicios Webs (Web Services) (WS) y la transferencia de los datos a través de objetos **JSON**, en tanto la comunicación entre el TPV y la TID se realiza mediante la tecnología Bluetooth™.

El TPV contiene una base de datos local del tipo **SQLite** para poder continuar funcionando en el caso de una falla en la comunicación con el SMD. Al momento de reconectarse se activa un proceso de sincronización que transfiere todos los datos almacenados en forma local en el TPV al SMD.

Este sistema permite utilizar un dispositivo de lectura de tarjetas con bandas magnéticas denominado Lector de Tarjetas Magnéticas (LTM) que consiste en un dispositivo que se conecta a la entrada de sonido de 3.5 milímetros de la TID y actúa como un sensor de sonido que permite decodificar las muestras leídas y genera los bits necesarios para representar los datos de una tarjeta y poder utilizarlos en el TPV y poder procesarlos.

La tecnología Near Field Communication (Comunicación de Campo Cercano) (NFC) se encuentra presente en este TFE permitiendo su uso a través del TPV para poder realizar una transacción de compra segura. Se elaboraron distintas pruebas con distintas etiquetas NFC simulando transacciones de compras y cuyos resultados fueron expuestos en este TFE.

Algunas Application Programming Interfaces (Interfaces de Programación de Aplicaciones) (APIs) como las de Foursquare™, Facebook™ y Twitter™ se en-

cuentran implementadas para poder utilizarlas como herramientas de marketing para el local comercial que quiera implementar este sistema desarrollado, y como complemento, el TPV brinda al usuario un módulo de gráficos estadísticos para que sean de utilidad en el caso de hacer un estudio del mercado y la progresión en las actividades desarrolladas en el comercio en cuestión.

Índice general

Introducción	1
Metodología	5
Resultados y Discusión	7
1. El Sistema Operativo (OS) Android	8
1.1. Inicios y evolución	8
1.2. Versiones	9
1.3. ¿Qué se necesita tener en cuenta para poder comenzar a crear una aplicación en Android?	11
1.4. Las herramientas para trabajar con la plataforma	12
1.4.1. Ejemplos de algunas herramientas para el desarrollo de aplicaciones	13
2. El desarrollo en Android	15
2.1. La Dalvik Virtual Machine (Máquina Virtual Dalvik) (DVM) . . .	15
2.2. La arquitectura Android	15
2.3. Partes que componen un proyecto de desarrollo Android	16
2.3.1. Las Activities	16
2.3.2. El Manifest	16
2.3.3. El ciclo de vida de una Activity	16
2.3.4. El Stack de Activities	17
2.3.5. Los Intents	17
2.3.6. Los Layouts	17
2.3.7. Los eventos de User Interface (Interfaz de Usuario) (UI) .	17
2.3.8. El Style	17
2.3.9. El Theme	17
2.3.10. El AdapterView	17
2.3.11. Los menús	18
2.3.12. Los Dialogs	18
2.3.13. Los Resources	18

2.3.14. El AsyncTask	18
2.3.15. El Content Provider	18
2.4. Los Servicios	19
2.5. Los secretos de performance	19
2.5.1. Mitos de performance	20
2.5.2. Siempre medir	21
2.6. La interacción con servicios remotos	21
2.7. El Dalvik Debug Monitor Server (DDMS)	21
2.8. El LogCat	21
3. El Terminal de Punto de Venta (TPV)	22
3.1. Los paquetes que contienen la aplicación:	22
3.2. Las librerías utilizadas	22
3.3. El Quick Response (Respuesta Rápida) (QR)	22
3.4. El uso de la cámara óptica	23
3.5. La Base de Datos (BD) SQLite	23
3.6. El LTM	23
3.7. La tecnología NFC	24
3.8. Los gráficos estadísticos	24
4. La User Interface (Interfaz de Usuario) (UI)	25
4.1. Acceso a la aplicación (login)	25
4.2. Menu principal	25
4.3. Abrir caja	30
4.4. Agregar venta	30
4.4.1. Ingreso de datos de los productos a vender	30
4.4.2. Ingreso de pagos, cliente y vendedor	32
4.4.3. Ingreso de datos de tarjeta de crédito	35
4.5. Listado de clientes	35
4.6. Opciones	35
4.7. Administración de stocks	35
4.8. Estadísticas	35
4.9. Cierre de caja	35
5. La Terminal de Ingreso de Datos (TID)	36
5.1. Lectura de un código unidimensional utilizando el foco automático de la cámara	36
5.2. Firma Digital	36
5.3. Comunicación Bluetooth	36
6. Diseño de un bootloader para la IH!	37
6.1. El formato HEX de Intel	37

ÍNDICE GENERAL **VII**

7. Diseño del un <i>bootloader</i> para la IH!	38
7.1. El formato HEX de Intel	38
 Conclusión	 39
 Apéndice	 40
A. Siglas y acrónimos	41
B. Glosario	43
 Bibliografía	 45

Índice de figuras

1.	Apple Store en New York City, USA	1
2.	Dispositivo utilizado para cobrar a los clientes en las tiendas de Apple	2
1.1.	Pantalla inicial de Android 4.1 (Jelly Bean) ejecutándose en un Galaxy Nexus	10
1.2.	Distribución de las distintas versiones de Android de acuerdo a un análisis de datos recolectados en un período de 14 días finalizando el 1 de Noviembre de 2012	11
4.1.	Mapa de la UI con sus activities y relaciones entre las mismas	26
4.2.	Mapa de la UI (continuación)	27
4.3.	Activity de acceso a la aplicación	28
4.4.	El menú principal	29
4.5.	Activity de apertura de caja	31
4.6.	Mapa de la UI con las distintas variantes de pagos	32
4.7.	Activity de ingreso de productos para la venta	33
4.8.	Activity de ingreso de métodos de pagos, cliente y vendedor.	34

Índice de tablas

Introducción

La necesidad de cambiar la forma en la que los sistemas de gestión y stock funcionan actualmente lleva a disparar la idea de pensar en una nueva metodología en la que el cliente tenga una experiencia de compra diferente y sumamente atractiva, la cual pueda llevar a diferenciarse de la competencia y pueda, de una manera más eficiente, implementar las transacciones de compras en un local comercial.

El actual avance en el desarrollo de las aplicaciones móviles y el uso de los dispositivos de pantallas táctiles lleva al desarrollo de sistemas a una nueva forma de generar aplicaciones donde los usuarios se sienten más a gusto gracias a la facilidad en su uso.

La inspiración de este proyecto parte de mi experiencia de compra en un Apple Store de la ciudad de New York, USA. En ese momento pude darme cuenta que una nueva forma de hacer compras era posible, y ellos lo hicieron posible aplicando esta nueva metodología en todos sus locales alrededor del mundo.



Figura 1: Apple Store en New York City, USA

Mi experiencia comienza al ingresar al local, donde pude observar y probar distintos productos mientras veía como circulaban los distintos vendedores vestidos con una remera azul y el logo de la compañía a la vista.

Al momento de finalizar la elección de lo que deseaba comprar, mire hacia mi alrededor viendo donde se podría hacer el pago del producto, pero no encontraba un sector de “checkout” o caja donde pagar a simple vista, entonces, busco a uno de los vendedores que tenía más cerca mio y le consulto al respecto y el me comenta que no existe tal sector ya que ellos mismos son los que realizan los cobros.

El vendedor tenía en la mano un celular iPhone con un hardware especial que funcionaba como un lector de tarjetas de crédito y también como un lector de códigos de barras.



Figura 2: Dispositivo utilizado para cobrar a los clientes en las tiendas de Apple

A partir de ahí, le entrego el producto, el vendedor lo pasa por el lector para realizar la lectura del código y me pide la tarjeta de crédito para deslizarlo por el dispositivo que tiene el iPhone incorporado. Los datos son procesados a través de una aplicación en el iPhone y luego de que la transacción fue aprobada, ingresa mi correo electrónico para poder enviarme la factura.

Fue una experiencia de compra distinta, rápida e innovadora, algo que realmente dió lugar a que yo pueda plantear un desarrollo similar para este Trabajo Final de Especialización (TFE).

Para este proyecto se utilizaron dispositivos táctiles con sistemas operativos

Android™, por eso, para este TFE¹, se desarrollaron aplicaciones en la plataforma Eclipse™ con el Software Development Kit (Kit de Desarrollo de Software) (SDK) de Android™. Se llevó a cabo la investigación y utilización de distintas librerías para llevar a extender todas las funciones que estas aplicaciones puedan brindar.

Denominamos Sistema Integral de Gestión (SIG) a toda la plataforma de venta desarrollado para este TFE y en donde el funcionamiento de cada una de sus partes y sus interacciones entre estas permite abarcar todos los requerimientos que un local comercial pueda necesitar.

El SIG que representa este TFE se encuentra desarrollado en tres partes, las cuales tienen una dependencia entre cada uno de ellos para que el sistema pueda funcionar correctamente.

La primera parte consiste en el Terminal de Punto de Venta (TPV) que consistió en el desarrollo de una aplicación que funciona en un Dispositivo de Pantalla Táctil (DPT) de la marca *ASUS*. Esta aplicación permite realizar las actividades de aperturas y cierres de caja, almacena datos de las ventas realizadas, permite generar las transacciones correspondientes de acuerdo a la forma de pago del cliente, genera los tickets requeridos, lista el historial de un cliente en particular, entre otras cosas más. El objetivo de esta parte del TFE es brindar al vendedor una plataforma de cobro y gestión que pueda brindarle todo lo necesario para llevar adelante las actividades comerciales diariamente.

La segunda parte es la Terminal de Ingreso de Datos (TID) que se pensó para que un dispositivo con Android™ pueda interactuar con el cliente que realiza una compra en el local comercial. Este dispositivo tiene instalado una aplicación que permite hacer de interfaz con el comprador, dándole la posibilidad de ingresar los datos de su tarjeta de crédito mediante un Lector de Tarjetas Magnéticas (LTM), poder ingresar su firma para autenticar la transacción, poder darle la posibilidad de interactuar con las distintas redes sociales para que pueda esta persona generar publicidad en su perfil de usuario sobre el local y en cambio recibir un descuento en el total de su compra y, por último, permitir recibir información detallada respecto a la compra que realiza.

Se analizaron 4 formas posibles de poder realizar un pago utilizando este SIG. Las opciones consisten en:

1. Pago en efectivo: se recibe dinero por el monto a pagar de la compra y se ingresa la información en el TPV a través del vendedor.
2. Pago en tarjeta de crédito: el cliente pasa su tarjeta por el LTM y los datos se envían a un gateway de pagos para autenticar los datos de la tarjeta y autorizar la transacción del mismo.

¹ A lo largo de todo el presente desarrollo escrito se utilizaron distintas abreviaciones y acrónimos donde sus significados pueden encontrarse en la parte final de este trabajo, en la sección “Siglas y acrónimos” (ver página 41).

3. Pago con un dispositivo con tecnología Near Field Communication (Comunicación de Campo Cercano) (NFC): se procede a acercar el dispositivo del cliente a un lector NFC para que este pueda tomar lectura y una vez autorizada la transacción, almacenar la información en el dispositivo nuevamente.
4. Pago utilizando las plataformas PayPalTM, MercadoPagoTM y DineroMailTM: a través de una cuenta en cualquiera de estas tres entidades, y mediante la implementación de los correspondientes SDK para cada una de ellas, se ofrece la posibilidad de realizar pagos mediante estos medios, permitiendo así dar más opciones a los clientes, en especial para aquellos de nacionalidad extranjera.

La tercera parte de este TFE es la elaboración de un *back-end* que consiste en una aplicación *Java* que permite funcionar como un Servicio de Manejo de datos (SMD), recibiendo y enviando datos al TPV las cuales pueden ser almacenados y requeridos mediante la implementación de una base de datos *MySQL* instalada en el mismo servidor donde el *Tomcat* se encuentra corriendo la aplicación que actúa como SMD.

El objetivo principal de este TFE consistió en el diseño, el desarrollo, las pruebas y la implementación de este SIG utilizando las más actuales tecnologías, tanto en la parte de dispositivos móviles, como en la parte de aplicaciones webs que existen en el mercado.

Este escrito fue desarrollado sobre la tecnología *LaTeX* que permite crear documentos con un aspecto profesional, siguiendo una metodología similar a la de la creación de código compilado, teniendo archivos fuentes que contienen los textos y pudiendo aplicar distintas librerías que contienen plantillas que definen el aspecto final del escrito elaborado y utilizándose comúnmente este lenguaje en los escritos matemáticos debido a la facilidad para la generación de fórmulas.

La presentación elaborada para la demostración de este TFE se realizó en *HTML5* utilizando una librería conocida como *RevealJS* que permite, en forma similar al *LaTeX*, generar una presentación web a partir de archivos fuentes que contienen los textos y otros datos necesarios para que esta pueda ser creada y a su vez reproducida en un navegador o browser al momento de realizar la presentación del TFE ante las autoridades correspondientes.

Metodología

- **Desarrollo de una aplicación Android™ que representa a un Terminal de Punto de Venta (TPV) sobre un Dispositivo de Pantalla Táctil (DPT)**
La primera actividad consistió en el desarrollo de la User Interface (Interfaz de Usuario) (UI) del TPV. Luego se codificó la lógica de la aplicación utilizando datos de prueba, o sea, simulando la interacción con el Servicio de Manejo de datos (SMD). A esta aplicación se agregó una librería para poder generar y leer códigos Quick Response (Respuesta Rápida) (QR) a través de la cámara óptica del dispositivo. El uso de la cámara también se implementó para la toma de fotos de los productos que ingresan en el stock del local y para la toma de imágenes de los clientes al ser dados de alta.
- **Primeras pruebas de comunicación entre dispositivos**
Se realizó una aplicación de prueba en el Dispositivo de Telefonía Celular (DTC) que solamente contenía desarrollado la comunicación vía Bluetooth™. Una vez que se pudo comunicar satisfactoriamente esta aplicación con el TPV, se procedió a desarrollar las funciones que permiten el ingreso de datos por parte del cliente que realiza la compra en el local, dando por nombre a esta parte del sistema el de una Terminal de Ingreso de Datos (TID).
- **Implementación de un servidor web para manejar peticiones JSON**
Para el siguiente paso, se codificó una aplicación *Java* con las librerías adecuadas para poder manejar objetos *JSON* y que esta pueda correr en un servidor de aplicaciones *Tomcat*. Esta aplicación, conectada a una base de datos *MySQL*, permite almacenar y obtener información que luego se envía a través del servicio web al TPV. La base de datos se configuró y luego se crearon las tablas y se almacenaron datos para poder utilizarlos en las distintas pruebas de funcionamiento del Sistema Integral de Gestión (SIG).
- **Inclusión del manejo de transacciones con tarjetas de créditos**
Se agregó la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) de MercadoPago™ para poder realizar transacciones con tarjetas de créditos y poder tener disponible este método de pago en el sistema sin depender de otros gateways de pagos de terceros.

- **Utilización de un dispositivo de lectura especial denominado Lector de Tarjetas Magnéticas (LTM)**

Se anexó el LTM en el DTC y se implementó una librería para poder hacer uso del mismo y que pueda realizar la lectura de los datos de las tarjetas de crédito y débito. Estos datos se envían a el TPV para que este pueda ejecutar el pedido de autorización correspondiente para la transacción de compra que se quiera llevar a cabo.

- **Incorporación del uso de la tecnología Near Field Communication (Comunicación de Campo Cercano) (NFC)**

Para poder utilizar la tecnología NFC y poder sacar provecho del lector incorporado que trae el DPT modelo *Nexus* de marca *ASUS*, se agrego en la aplicación del TPV un módulo que permite hacer lectura y escritura de otros dispositivos que contengan NFC. Las pruebas se realizaron mediante unas etiquetas NFC que contienen una memoria interna en donde se pudieron almacenar y obtener datos para simular transacciones de compras.

- **Proveer al usuario final el análisis de los datos que se manejan diariamente en el SIG**

Los gráficos estadísticos se anexaron al TPV mediante el uso de una librería de GoogleTM que permite, a partir de una serie de datos y unos datos extras, generar imágenes que permiten mostrar la evolución de distintas variables de importancia para el vendedor en un cierto tiempo transcurrido.

- **Aprovechar la utilización de redes sociales como un canal de marketing**

Por último, se incorporó la utilización de las Application Programming Interfaces (Interfaces de Programación de Aplicaciones) (APIs) de FacebookTM, TwitterTM y FoursquareTM en la TID para poder brindar al cliente la posibilidad de que pueda ingresar con su cuenta a cualquiera de estas redes sociales y envíe un mensaje relacionado con el local a todos sus contactos para que se pueda generar una publicidad viral y que de esta forma pueda servir este SIG como una herramienta de marketing.

Resultados y Discusión

1. El Sistema Operativo (OS) Android

Primeramente, para poder llegar al desarrollo propuesto en este Trabajo Final de Especialización (TFE), se realizará un breve descripción del sistema operativo para dispositivos móviles que es parte fundamental para la implementación de las distintas aplicaciones desarrolladas.

1.1. Inicios y evolución

Android es un sistema operativo móvil basado en Linux, que junto con aplicaciones middleware está enfocado para ser utilizado en dispositivos móviles como teléfonos inteligentes, tabletas, Google TV y otros dispositivos. Es desarrollado por la Open Handset Alliance, la cual es liderada por GoogleTM. Este sistema por lo general maneja aplicaciones como Google Play.

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

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 700.000 aplicaciones (de las cuales, dos tercios son gratuitas) disponibles para la tienda de aplicaciones oficial de Android: Google Play, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como pueden ser la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung. Google Play es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java. No obstante, no es un sistema operativo libre de malware, aunque la mayoría de ello es descargado de sitios de terceros.

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

La estructura del sistema operativo Android™ se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2,8 millones de líneas de lenguaje C, 2,1 millones de líneas de Java y 1,75 millones de líneas de C++ ([Wikipedia, 2012a](#)).

Android se encuentra instalado en cientos de millones de dispositivos móviles en mas de 190 países alrededor del mundo. Cada día, más de 1 millón de dispositivos con Android son activados mundialmente.

1.2. Versiones

Las versiones de Android reciben el nombre de postres en inglés. En cada versión el postre elegido empieza por una letra distinta siguiendo un orden alfabético ([Google, 2012a](#)):

1. A: Apple Pie (v1.0), Tarta de manzana
2. B: Banana Bread (v1.1), Pan de plátano
3. C: Cupcake (v1.5), Magdalena glaseada.
4. D: Donut (v1.6), Rosquilla.
5. E: Eclair (v2.0/v2.1), pastel francés conocido en España como pepito, petisú, suso o canuto.
6. F: Froyo (v2.2), (Abreviatura de Frozen Yogurt) Yogur Helado.
7. G: Gingerbread (v2.3), Pan de jengibre.
8. H: Honeycomb (v3.0/v3.1/v3.2), Panal de miel.
9. I: Ice Cream Sandwich (v4.0), Sándwich de helado.
10. J: Jelly Bean (v4.1/v4.1.2/v4.2), Judía de gelatina.

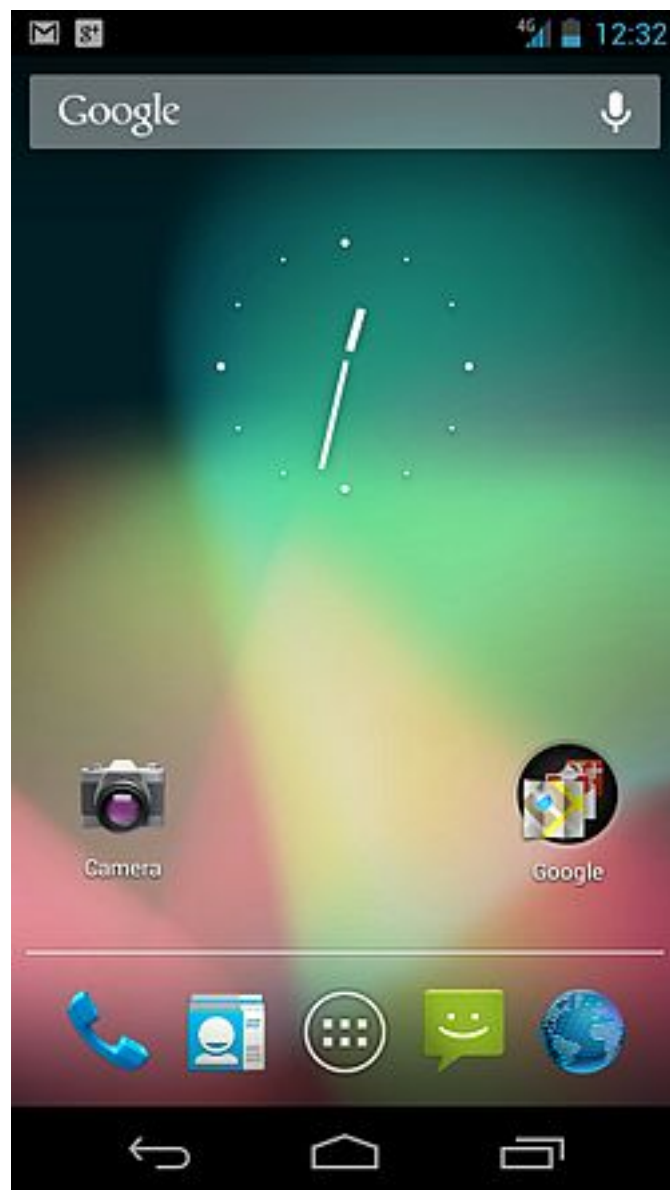


Figura 1.1: Pantalla inicial de Android 4.1 (Jelly Bean) ejecutándose en un Galaxy Nexus

11. K: Key Lime Pie (v4.2), Proximamente.

Cada versión de Android tiene un número de API relacionado. Para el caso del Dispositivo de Pantalla Táctil (DPT) utilizado para el Terminal de Punto de Venta (TPV), el dispositivo contiene la versión 4.2 Jelly Bean (JB) de Android que funciona con desarrollos realizados con la API 17 y el Dispositivo de Telefonía

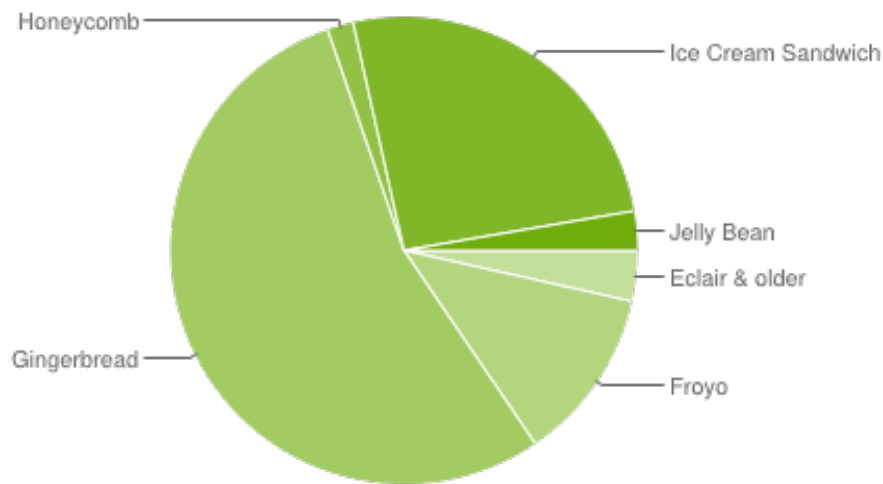


Figura 1.2: Distribución de las distintas versiones de Android de acuerdo a un análisis de datos recolectados en un período de 14 días finalizando el 1 de Noviembre de 2012

Celular (DTC) contiene la versión 4.0.4 Ice Cream Sandwich (ICS) cuya API es la número 15.

1.3. ¿Qué se necesita tener en cuenta para poder comenzar a crear una aplicación en Android?

La información que un desarrollador necesita para poder construir una aplicación en Android se encuentra organizada en tres secciones que representan el orden general para el desarrollo de las aplicaciones:

- **Diseño:** Antes de comenzar a escribir una línea de código, se necesita diseñar la User Interface (Interfaz de Usuario) (UI) y hacer que esta encaje en la experiencia del usuario Android. Aunque uno sepa que es lo que el usuario pueda llegar a hacer con la aplicación, uno debe enfocarse en como el usuario debe interactuar con la misma. El diseño debe ser simple, llamativo, y debe utilizar los lineamientos que el mundo Android brinda. Este es el primer paso.
- **Desarrollo:** Una vez que el diseño finaliza, todo lo que se necesita son las herramientas para hacer realidad las ideas de la aplicación. El framework de Android provee las APIs para construir aplicaciones que puedan utilizar todo el potencial del hardware del dispositivo, accesorios conectados al mismo, la internet, y demás.

- **Distribución:** Una vez que la aplicación se encuentra terminada y que se probó para distintos tamaños de pantallas y densidades, y cuyas pruebas fueron realizadas a través de un emulador Android y en dispositivos reales, es el momento indicado para comenzar a publicar la aplicación. Existen varios factores que hacen depender la estrategia de venta de una aplicación, como por ejemplo, los dispositivos que soporta, la moneda de cobro, etcetera.

([Google, 2012e](#))

1.4. Las herramientas para trabajar con la plataforma

El Software Development Kit (Kit de Desarrollo de Software) (SDK) de Android incluye una variedad de herramientas que ayudan a crear aplicaciones móviles para la plataforma Android. Las herramientas se encuentran clasificadas en dos grupos: SDK Tools (SDKT) y Platform Tools (PT). Las SDKTs son independientes de la plataforma y son requeridas sin tener relación alguna con la plataforma de Android con la que se está desarrollando. Las PTs son customizadas para soportar las características de la más reciente plataforma de Android.

Las SDKTs son instaladas con el SDK Starter Package (SDKSP) y son periódicamente actualizadas. Estas actualizaciones se encuentran disponibles cada vez que se desea instalar una nueva SDK y están coordinadas con el desarrollo general de Android. Las más importantes SDKTs incluyen el Android SDK Manager (ASM), el Android Virtual Device Manager (AVDM), el emulador y el Dalvik Debug Monitor Server (DDMS).

El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad.

El SDK de Android, incluye un conjunto de herramientas para el desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen Linux (cualquier distribución moderna), Mac OS XTM 10.4.9 o posterior, y Windows XPTM o posterior. El Entorno Integrado de Desarrollo (IDE) soportada oficialmente es EclipseTM junto con el complemento Android Development Tools (ADT) que se encuentra como un plugin al entorno EclipseTM, aunque también puede utilizarse un editor de texto para escribir ficheros Java y Xml y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados (por ejemplo, reiniciarlos, instalar aplicaciones en remoto, etcetera). ([Wikipedia, 2012b](#))

Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (este directorio necesita permisos de superusuario, root, por razones de

seguridad). Un paquete APK incluye archivos .dex (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

1.4.1. Ejemplos de algunas herramientas para el desarrollo de aplicaciones

- Android SDK Manager (ASM): Permite manejar Android Virtual Device (AVD)s, proyectos, y los componentes instalados del SDK.
- Dalvik Debug Monitor Server (DDMS): Permite realizar debug a las aplicaciones Android.
- Android Emulator: Un QEMU¹-based device-emulation tool que permite ser utilizado para diseñar, debugear, y testear las aplicaciones en un ambiente de runtime Android real.
- mksdcard: Ayuda a la creación de una imagen de disco que puede ser utilizado con el emulador, para simular la presencia de una tarjeta externa de almacenamiento, como una tarjeta de memoria Secure Digital (SD), por ejemplo.
- sqlite3: Permite acceder a los archivos SQLite creados y utilizados por las aplicaciones Android.
- Android Debug Bridge (ADB): Herramienta versátil que permite manejar el estado de una instancia de un emulador o un dispositivo Android. Uno puede también utilizarlo para instalar un archivo que represente una aplicación Android (.apk) en un dispositivo.

El Android Development Tools (ADT) de Eclipse

Para ayudar a desarrollar eficientemente, el ADT ofrece un IDE Java con opciones avanzadas para desarrollar, debugear y empaquetar aplicaciones Android. Usando la IDE, uno puede desarrollar para cualquier dispositivo Android o puede crear dispositivos virtuales para emular distintas configuraciones de hardware.

El ADT consiste en un plugin para la plataforma EclipseTM que está diseñado para brindar un entorno integrado en el cual se puedan crear aplicaciones para Android.

ADT extiende las capacidades de EclipseTM para poder rápidamente setear un proyecto nuevo de Android, crear la UI de la misma, agregar paquetes basados en la API del framework Android, debugear las aplicaciones y poder exportar los archivos .apk para la distribución en el Google MarketTM. (Google, 2012b)

¹QEMU es un emulador de procesadores basado en la traducción dinámica de binarios (conversión del código binario de la arquitectura fuente en código entendible por la arquitectura huésped).

El Android Emulator

El SDK de Android incluye un emulador de dispositivo móvil que funciona en una computadora convencional. Esto permite agregar la posibilidad de probar el desarrollo de una aplicación sin la necesidad de hacerlo a través de un dispositivo físico. (Google, 2012c)

Android Virtual Device (AVD)

Un AVD consiste en:

- Un perfil de hardware: Se define las características del dispositivo virtual. Por ejemplo, uno puede definir si el dispositivo contiene una cámara, si utiliza un teclado físico QWERTY o no, cuanta memoria tiene, etcetera.
- Un mapeo a una imagen del sistema: Uno puede definir que versión de la plataforma Android el dispositivo utilice.
- Otras opciones: Uno puede especificar el skin que se desea utilizar con el AVD, lo que permite a uno controlar las dimensiones de la pantalla, la apariencia, y demás. Uno también puede especificar el tipo de tarjeta SD a utilizar con el AVD.
- Un espacio de almacenamiento dedicado en el propio equipo de desarrollo: Toda la información relacionada al usuario del dispositivo (aplicaciones instaladas, configuraciones, etcetera) y la información de la tarjeta SD emulada se encuentra almacenada en esta area.

Uno puede crear los AVDs que uno necesite, basado en los tipos de dispositivos que uno quiera que la aplicación pueda soportar. Al realizar las pruebas, es necesario hacerlas en cada uno de los AVDs para verificar su correcto funcionamiento y visualización. (Google, 2012d)

2. El desarrollo en Android

En este capítulo se describen las distintas partes que componen una aplicación Android y como la plataforma se encuentra estructurada para poder soportar y manejar estas aplicaciones.

2.1. La Dalvik Virtual Machine (Máquina Virtual Dalvik) (DVM)

Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. Dalvik ha sido diseñada por Dan Bornstein con contribuciones de otros ingenieros de Google.

Dalvik está optimizada para requerir poca memoria y está diseñada para permitir ejecutar varias instancias de la máquina virtual simultáneamente, delegando en el sistema operativo subyacente el soporte de aislamiento de procesos, gestión de memoria e hilos.

A menudo Dalvik es nombrada como una máquina virtual Java, pero esto no es estrictamente correcto, ya que el bytecode con el que opera no es Java bytecode. Sin embargo, la herramienta dx incluida en el Software Development Kit (Kit de Desarrollo de Software) (SDK) de Android permite transformar los archivos Class de Java compilados por un compilador Java al formato de archivos Dex.

El nombre de Dalvik fue elegido por Bornstein en honor a Dalvík, un pueblo de Islandia donde vivieron antepasados suyos. ([Wikipedia, 2012c](#))

2.2. La arquitectura Android

Basicamente, Android tiene las siguientes capas:

- Aplicaciones (escritas en Java, ejecutadas en Dalvik)
- Librerías y servicios del framework (escritas en su mayoría en Java)

Las aplicaciones y la mayor parte del código del framework se ejecutan en una máquina virtual.

- Librerías nativas, daemons y servicios (escritos en C o C++)

- El kernel de linux, que incluye
drivers para hardware, redes, sistema de acceso a archivos e Interprocess Communication (Comunicación Interprocesos) (IPC).

2.3. Partes que componen un proyecto de desarrollo Android

En esta sección se presentan las distintas clases, vistas y recursos. Como interactúan cada uno de ellos y cual es el propósito de cada uno.

2.3.1. Las Activities

Es una pantalla dentro de una aplicación.
Una aplicación puede contener una o más Activities.
Pueden ser invocadas desde otras aplicaciones.
Cada activity es independiente de las demás.
Todas las Activities extienden de la clase Activity.
Cada Activity tiene una ventana en la cual mostrarse.
Normalmente ocupa toda la pantalla.
Puede utilizar ventanas adicionales como Dialogs.
Se muestra utilizando un árbol de vistas que extienden de la clase View.
El raíz del árbol se define llamando al método Activity setContentView().

2.3.2. El Manifest

Es el xml donde cada aplicación describe y declara sus componentes.
Cada app tiene su manifest que se incluye en el apk.

2.3.3. El ciclo de vida de una Activity

Tiene tres estados principales:
Active, actualmente en pantalla con foco.
Paused, perdió foco, continúa parcialmente visible.
Stopped, fue tapado por otra Activity.
Las transiciones de estado son notificadas al Activity por una serie de métodos protegidos.
Se deberá mantener una llamada al método original en la clase padre al sobrescribir uno de estos métodos.
Los siete métodos del ciclo de vida pueden ser utilizados para monitorear los siguientes subciclos.
Tiempo de vida, usado para liberar recursos en el onDestroy() que fueron creados en el onCreate().
Visibilidad, mantener recursos que afecten la UI.

Primer plano, mantener la interactividad con el usuario.
figuraoo().

2.3.4. El Stack de Activities

<http://developer.android.com/guide/components/tasks-and-back-stack.html>
figuraoo().

2.3.5. Los Intents

Un intent puede referenciar un componente.
Si la referencia no es explícita android buscará el componente que mejor responda.

2.3.6. Los Layouts

El layout define la distribución de las vistas y contenedores.
Los layouts pueden definirse en XML o instanciarse programáticamente.
Tipos: `FrameLayout`, `LinearLayout`, `RelativeLayout`, `TableLayout`.
figuraoo()

2.3.7. Los eventos de User Interface (Interfaz de Usuario) (UI)

View permite registrar distintos listeners para los siguientes eventos.
`onClick()` `onLongClick()` `onFocusChange()` `onKey()` `onTouch()` `onCreateContextMenu()`

2.3.8. El Style

Es una colección de propiedades que especifican el formato de una vista. Los estilos se definen en XML.

2.3.9. El Theme

Cuando un Style es aplicado a un Activity o Application se lo denomina Theme.
Todas las vistas del Activity o Application tendrán el Style aplicado.

2.3.10. El AdapterView

Es una subclase de `ViewGroup`.
Sus vistas hijas son definidas por un `Adapter`.
Se utiliza para mostrar datos almacenados.
Algunos `AdapterView` son: `Gallery` `ListView` `Spinner`
El `adapter` es una clase utilizada por un `AdapterView` para obtener datos y construir las vistas hijas.

2.3.11. Los menús

Option menu, es el menú principal de un activity, se muestra al presionar el boton menú.

Context Menu, es un menú flotante que se muestra al hacer `long press` sobre una vista.

Submenu, es un menú flotante de items que el usuario abre al seleccionar un item de menú.

2.3.12. Los Dialogs

El dialog toma el primer plano quitandole el foco al Activity que le dio origen.

Para crear un dialog se extiende de la clase Dialog.

El API de Android provee los siguientes tipos de dialogs:

AlertDialog ProgressDialog DatePickerDialog TimePickerDialog

2.3.13. Los Resources

A cada resource se le asigna un ID.

Los IDs están definidos en la clase R del proyecto.

La clase R es generada durante la compilación.

R tiene una clase estática para cada tipo de resource.

Por cada resource hay un integer estático.

Los resources pueden ser accedidos desde el código o desde XML.

2.3.14. El AsyncTask

El método `doInBackground()` se ejecuta en un thread aparte.

Los métodos `onPreExecute()` y `onPostExecute()` se ejecutan en el thread de UI.

2.3.15. El Content Provider

An SQLite database is private to the application which creates it. If you want to share data with other applications you can use a ContentProvider.

A ContentProvider allows applications to access data. In most cases this data is stored in an SQLite database. A ContentProvider can be used internally in an application to access data. If the data should be shared with another application a ContentProvider allows this. The access to a ContentProvider is done via an URI. The basis for the URI is defined in the declaration of the ContentProvider in the AndroidManifest.xml file via the `android:authorities` attribute. Many Android datasources, e.g. the contacts, are accessible via ContentProviders. Typically the implementing classes for a ContentProviders provide public constants for the URIs

SQLite stores the whole database in a file. If you have access to this file, you can work directly with the data base. Accessing the SQLite database file only works in the emulator or on a rooted device.

A standard Android device will not grant read-access to the database file.

Content providers can be used from other processes and are required by some mechanisms on Android like the global search. There are also some classes available that help you deal with content providers that save you some of the hassle of managing memory.

- * Other apps will be able to access your data.

- * You can wrap and abstract a lot of the query logic in your content provider, and limit access.

- * You will be able to lean on the system to allow for things like managed queries.

2.4. Los Servicios

Un servicio es un componente de la aplicación que puede realizar operaciones long-running o de larga corrida en el background o en segundo plano y no provee de una UI.

Otro componente de la aplicación puede iniciar un servicio y este continuará corriendo en segundo plano aún si el usuario cambia a otra aplicación. Adicionalmente, un componente puede unirse a un servicio para interactuar con este y también para realizar IPC. Por ejemplo, un servicio podría manejar transacciones de red, reproducir música, ejecutar acciones de entrada y salida sobre archivos (file I/O), o interactuar con un content provider, todo desde un segundo plano que resulta invisible al usuario.

Hay que considerar que un servicio corre en el thread principal del proceso que lo contiene. El servicio no crea su propio thread y no corre en procesos separados, a menos que se lo especifique.

2.5. Los secretos de performance

<http://developer.android.com/training/articles/perf-tips.html>

1. Evitar crear objetos innecesarios: La creación de objetos siempre tiene un costo. Un garbage collector generacional con pools de asignación por thread para objetos temporarios puede hacer la asignación de objetos un poco más eficiente, pero la asignación de memoria es siempre más costosa que no asignar memoria para los objetos.
2. Es preferible un estático sobre un virtual: Si no es necesario acceder a las variables de un objeto, es recomendable hacer los métodos estáticos. Las llamadas tendrán un aumento de velocidad entre 15 %-20 %. Es una buena práctica

también, porque se podría dar la información desde la firma del método que al llamar al método, este no pueda alterar el estado del objeto.

3. Usar static final para constantes: Es una buena práctica declarar las constantes como static final cuando sea posible, ya que de esta manera se evita el uso de un método de inicialización de clase.
4. Evitar getters/setters internos: Llamadas a métodos virtuales son costosas, mucho más que la búsqueda de variables instanciadas. Es razonable seguir prácticas de programación orientada a objetos y tener getters y setters en una interfaz pública, pero dentro de una clase uno debería siempre acceder a las variables directamente.
5. Usar la nueva sintaxis del ciclo For: La nueva sintaxis del ciclo For, conocida también como el ciclo "for-each", puede ser utilizada para collections que implementen la interfaz Iterable y para arrays. Con las collections, un iterator es asignado para hacer llamadas a la interfaz de hasNext() y next(). Con un ArrayList, un ciclo con contador es cerca de 3 veces más rápido (con o sin Just in Time Compiler (Compilador En Tiempo Real) (JIT)), pero para otras collections el nuevo ciclo For será exactamente equivalente al de un iterator explícito.
6. Evitar usar puntos flotantes (floating-points): Los puntos flotantes son 2 veces más lentos que los integers o enteros en los dispositivos Android. En términos de velocidad, no hay diferencia entre el float y el double en los hardware modernos. En tamaño, double es 2 veces mas grande. Con Máquinas de escritorio, asumiendo que el espacio no es un problema, sería preferible double antes que float.
7. Saber usar las librerías: Además de conocer las razones de utilizar una librería que ya se encuentre disponible antes que armar una propia, hay que tener en cuenta que el sistema ofrece la libertad de poder reemplazar las llamadas a métodos de librerías con otras que contengan código assembler, las cuales podrían ser mejores en comparación con el mejor código que el JIT pueda producir para su equivalente en Java.

2.5.1. Mitos de performance

En dispositivos sin JIT, es cierto que invocar métodos a través de una variable con un tipo exacto que a través de una interfaz es sutilmente más eficiente (por ejemplo, es mejor invocar métodos en un HashMap Map que en un Map map, aún cuando en ambos casos el map fuera un HashMap).

En dispositivos sin JIT, realizar caching para los accesos a las variables de objetos logra una mejora del 20 % que repetidamente acceder a la misma variable. Con un JIT, el acceso a variables cuesta lo mismo que a un acceso a variables de un objeto en forma local, por lo tanto, no es una optimización requerida a menos que

uno sienta que esto facilita la lectura del código (ocurre lo mismo para los tipos final, static y static final).

2.5.2. Siempre medir

Antes de comenzar a optimizar, hay que asegurarse que uno tenga un problema que necesite resolver. Para esto, es importante medir con detalle la performance existente, porque de otra manera, no se podrá medir el beneficio de la implementación de alguna alternativa de mejora.

Se recomienda utilizar Caliper para correr los microbenchmarks creados por uno mismo. El Caliper es un framework de microbenchmarking para Java.

2.6. La interacción con servicios remotos

El parser de JSON que provee Android es poco performante.

JSON con GSON (mejor performance que el parser provisto por Android).

2.7. El Dalvik Debug Monitor Server (DDMS)

Perspectiva de Eclipse.

Debugging.

Permite ver memoria y file system.

Profiling.

2.8. El LogCat

El método Log.e() se utiliza para registrar errores. El método Log.w() se utiliza para registrar precauciones o warnings. El método Log.i() se utiliza para registrar mensajes de información. El método Log.d() se utiliza para registrar mensajes del tipo debug. El método Log.v() se utiliza para registrar mensajes verbales. El método Log.wtf() se utiliza para registrar errores muy graves que nunca deberían ocurrir (What a Terrible Failure).

La salida básica del LogCat incluye información de diferentes fuentes. Para su uso en actividades de debug, puede ser muy útil filtrar la salida del LogCat de acuerdo a diferentes etiquetas o tags para la aplicación específica que se quiera analizar. De esta manera, uno puede enfocar en los logs de la propia aplicación y evitar recibir información de otras fuentes o partes que para el caso a analizar sean de poco interés.

3. El Terminal de Punto de Venta (TPV)

Introduccion.

3.1. Los paquetes que contienen la aplicación:

Los paquetes creados.

3.2. Las librerías utilizadas

Las librerías utilizadas:

3.3. El Quick Response (Respuesta Rápida) (QR)

1) Se pueden leer utilizando el ZXING como activity y listo. 2) Para generar baje tres librerías, qrgen, zxing-j2se-1.7 y zxing-core-1.7

El QR o código de respuesta rápida es un módulo para almacenar información en una matriz de puntos o un código de barras bidimensional creado por la compañía japonesa Denso Wave, subsidiaria de Toyota, en 1994. Se caracteriza por los tres cuadrados que se encuentran en las esquinas y que permiten detectar la posición del código al lector. La sigla QR se deriva de la frase inglesa Quick Response (Respuesta Rápida en español), pues los creadores (Joaco Retes y Euge Damm1) aspiran a que el código permita que su contenido se lea a alta velocidad.

Aunque inicialmente se usó para registrar repuestos en el área de la fabricación de vehículos, hoy los códigos QR se usan para administración de inventarios en una gran variedad de industrias. La inclusión de software que lee códigos QR en teléfonos móviles, ha permitido nuevos usos orientados al consumidor, que se manifiestan en comodidades como el dejar de tener que introducir datos de forma manual en los teléfonos. Las direcciones y los URLs se están volviendo cada vez más comunes en revistas y anuncios. El agregado de códigos QR en tarjetas de presentación también se está haciendo común, sim-

plificando en gran medida la tarea de introducir detalles individuales de un nuevo cliente en la agenda de un teléfono móvil.

<http://www.qrcode.com/en/>

Los códigos QR contienen información tanto en su posición horizontal como vertical, adoptando así el término de bidimensional. Al manejar información en ambas direcciones, un código QR puede representar hasta varios cientos de códigos de barras tradicionales.

Los códigos XQR...

3.4. El uso de la cámara óptica

Using the camera on the Android device can be done via integration of the existing Camera application. In this case you would start the existing Camera application via an Intent and to get the data after the user returns to our application. You can also directly integrate the camera into your application via the Camera API.

3.5. La Base de Datos (BD) SQLite

http://www.vogella.com/articles/AndroidSQLite/article.html#todo_database

SQLite is an Open Source Database which is embedded into Android. SQLite supports standard relational database features like SQL syntax, transactions and prepared statements. In addition it requires only little memory at runtime (approx. 250 KByte).

SQLite supports the data types TEXT (similar to String in Java), INTEGER (similar to long in Java) and REAL (similar to double in Java). All other types must be converted into one of these fields before saving them in the database. SQLite itself does not validate if the types written to the columns are actually of the defined type, e.g. you can write an integer into a string column and vice versa.

SQLite is available on every Android device. Using an SQLite database in Android does not require any database setup or administration.

If your application creates a database, this database is by default saved in the directory DATA/data/APP_NAME/databases/FILENAME.

Access to an SQLite database involves accessing the filesystem. This can be slow. Therefore it is recommended to perform database operations asynchronously, for example inside the AsyncTask class.

3.6. El Lector de Tarjetas Magnéticas (LTM)

The magnetic stripe consists of 3 physically separated "tracks". Track 1 is closest to the bottom of the card, and track 3 is the highest. Square's reader is positioned to read track 2. Track 2 is the most commonly used track, but most credit

cards also use track 1. Track 2 includes card numbers and expiration dates. Track 1 includes that plus names. There may be other data too, depending on the particular card. These tracks are specced to be .11 inches wide, so to read track 1 with Square's reader, we just need to reposition the stripe so that track 1 is lined up with the read head. To do that, we just need to raise it by .11 inches. And we can do that by cutting a .11 (or in my case 1/8) inch slice from a card we do not care about, and putting that at the bottom of the reader. This is called a "shim"

<http://cosmodro.me/blog/2011/mar/25/rhombus-square-iskewedi/>

3.7. La tecnología Near Field Communication (Comunicación de Campo Cercano) (NFC)

3.8. Los gráficos estadísticos

ACChartEngine is a charting library for Android applications. It currently supports the following chart types: * line chart * area chart * scatter chart * time chart * bar chart * pie chart * bubble chart * doughnut chart * range (high-low) bar chart * dial chart / gauge * combined (any combination of line, cubic line, scatter, bar, range bar, bubble) chart * cubic line chart All the above supported chart types can contain multiple series, can be displayed with the X axis horizontally (default) or vertically and support many other custom features. The charts can be built as a view that can be added to a view group or as an intent, such as it can be used to start an activity. The model and the graphing code is well optimized such as it can handle and display huge number of values. ACChartEngine is currently at the 1.0.0 release. New chart types will be added in the following releases. Please keep sending your feedback such as we can continually improve this library.

La idea en este proyecto es incluir las siguientes estadísticas:

1) Comparación de dos años de ventas mes x mes. 2) Las ventas los últimos 6 meses como fue en el tiempo. 3) los rubros más vendidos en los últimos 6 meses. 4) la venta con valor más bajo y más alto mes x mes (temperature range).

4. La User Interface (Interfaz de Usuario) (UI)

Las principales actividades que componen la UI son las siguientes:

1. Acceso a la aplicación (login).
2. Menu principal.
3. Abrir caja.
4. Agregar venta.
5. Listado de clientes.
6. Opciones.
7. Administración de stocks.
8. Estadísticas.
9. Cierre de caja.

Se denomina “activity” a la visualización de la UI de la aplicación en la pantalla del dispositivo móvil.

4.1. Acceso a la aplicación (login)

Al iniciar la aplicación, la primer activity que se puede ver es la del acceso a la misma 4.3. Con un usuario y password se procede al ingreso del menú principal.

4.2. Menu principal

En el menú principal 4.4 podemos ver una serie de botones con las distintas opciones que la aplicación nos brinda. Todas las opciones son descritas a continuación.

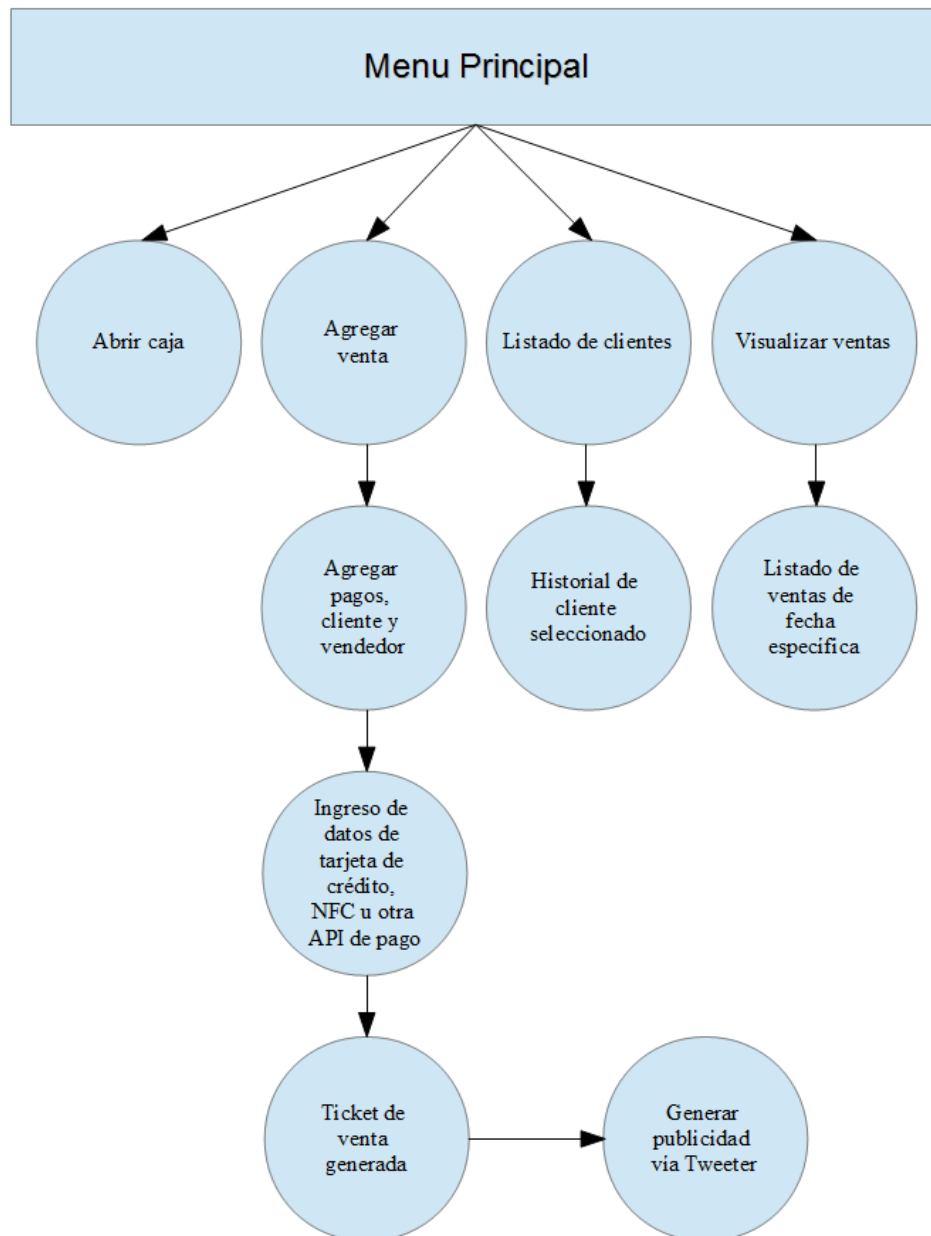


Figura 4.1: Mapa de la UI con sus actividades y relaciones entre las mismas

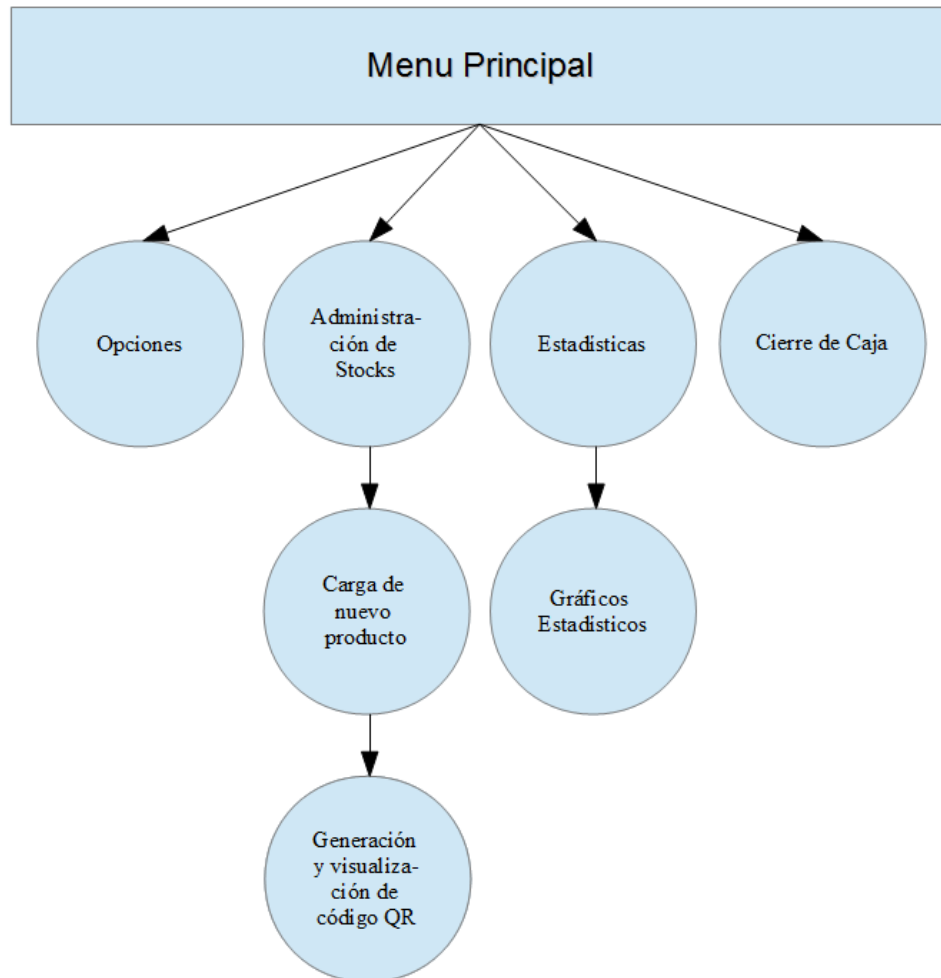


Figura 4.2: Mapa de la UI (continuación)

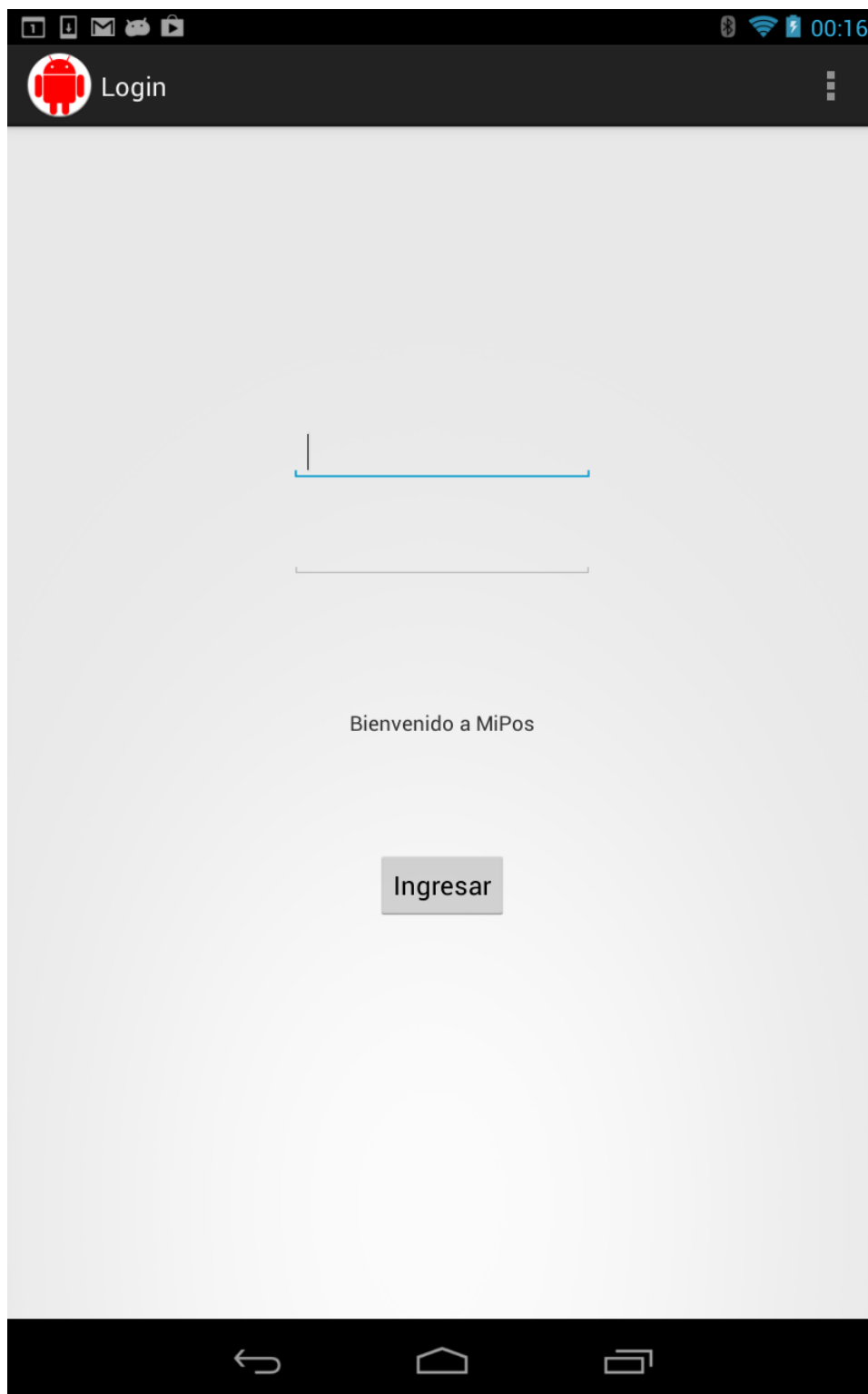


Figura 4.3: Activity de acceso a la aplicación

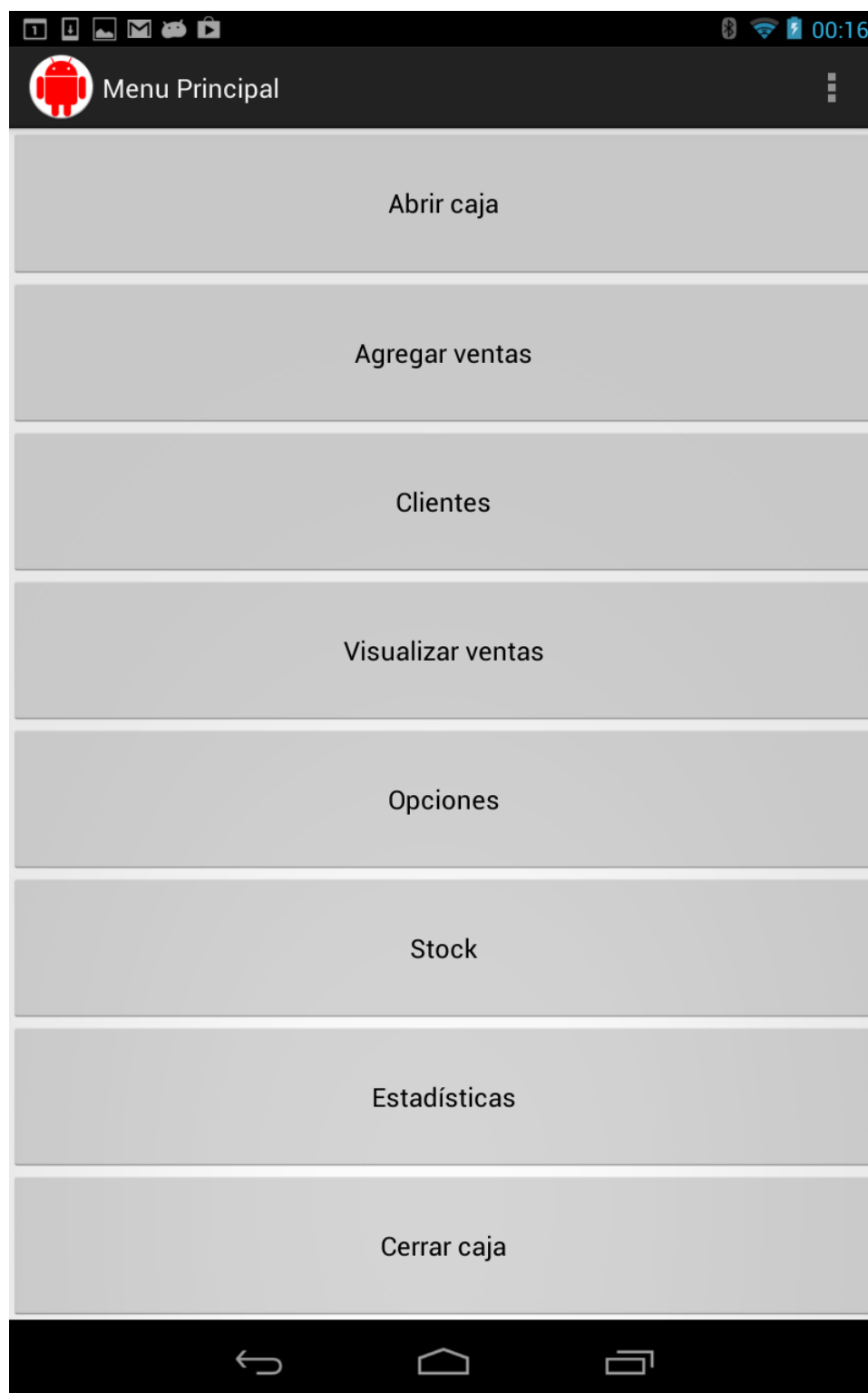


Figura 4.4: El menú principal

4.3. Abrir caja

Como primera actividad diaria en un local comercial, es necesario realizar la apertura de caja del día. Para poder hacerlo, el sistema muestra la fecha correspondiente al momento de realizar una nueva apertura y la posibilidad de ingresar un monto que corresponde al efectivo con la que la caja va a disponer para luego realizar los intercambios de billetes necesarios al generar ventas con pagos en efectivo 4.5.

4.4. Agregar venta

Esta es una de las más importantes funciones que la aplicación ofrece, ya que aquí es donde se realizan los ingresos de ventas que corresponden al local durante todo un ciclo diario de caja. El ciclo que desarrolla un ingreso de venta en la aplicación consiste de las siguientes actividades:

1. Ingreso de datos de los productos a vender.
2. Ingreso de pagos, cliente y vendedor.
3. Ingreso de datos de tarjeta de crédito.
4. Realización de pago mediante la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) de MercadoPago.
5. Lectura y escritura de datos mediante dispositivo Near Field Communication (Comunicación de Campo Cercano) (NFC).
6. Generación de ticket.
7. Marketing via Tweeter.

4.4.1. Ingreso de datos de los productos a vender

En esta actividad 4.7 se ingresan los códigos de los productos que el cliente quiere comprar. Al ingresar el código, el sistema muestra el precio del mismo y la disponibilidad de stock actual que el producto tenga mediante una búsqueda de la información en la base SQLite interna de la aplicación.

Al ingresar los productos la ventana muestra una lista con todos los productos y sus precios, así como también el total de lo que sería la venta a generar.

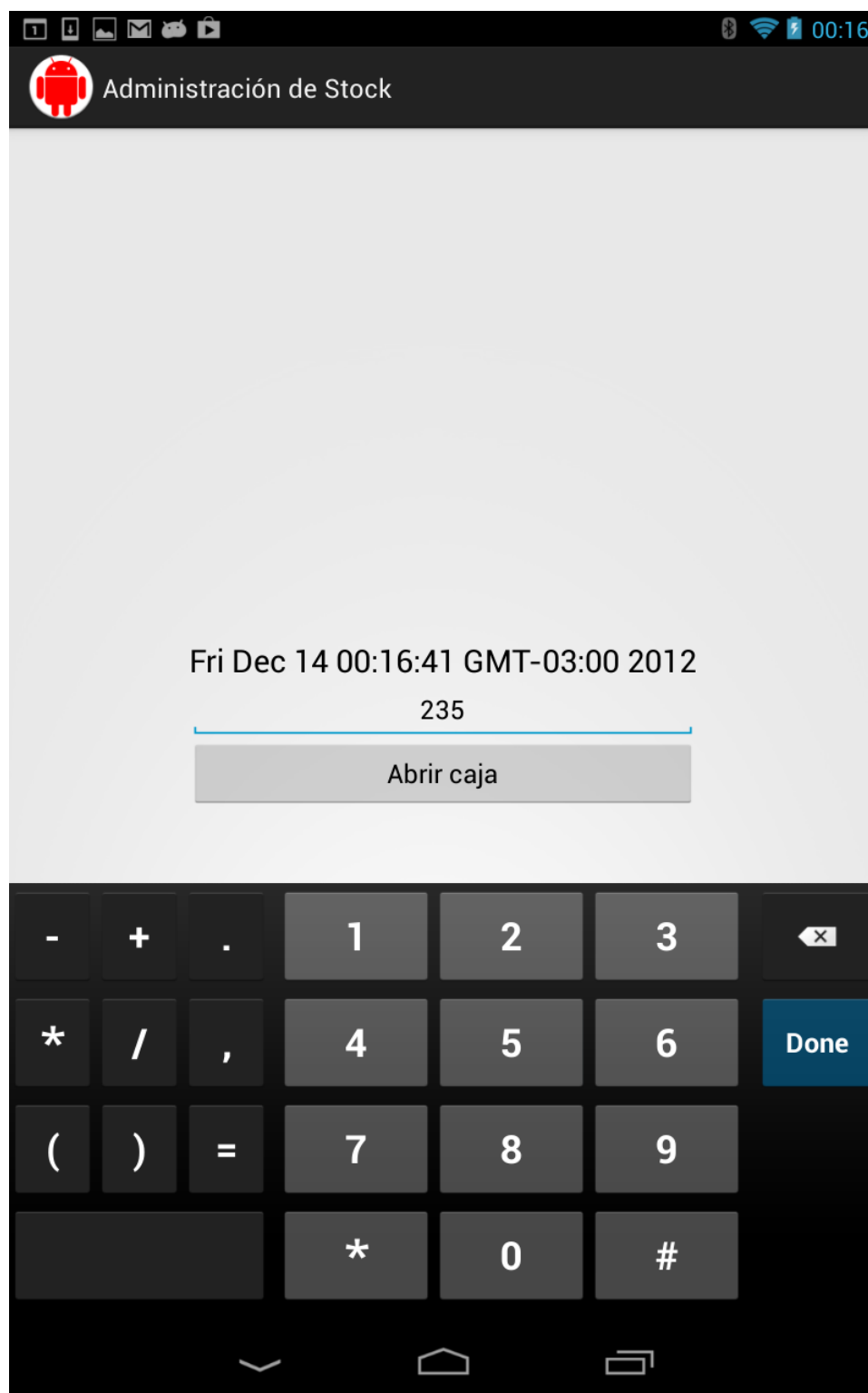


Figura 4.5: Activity de apertura de caja

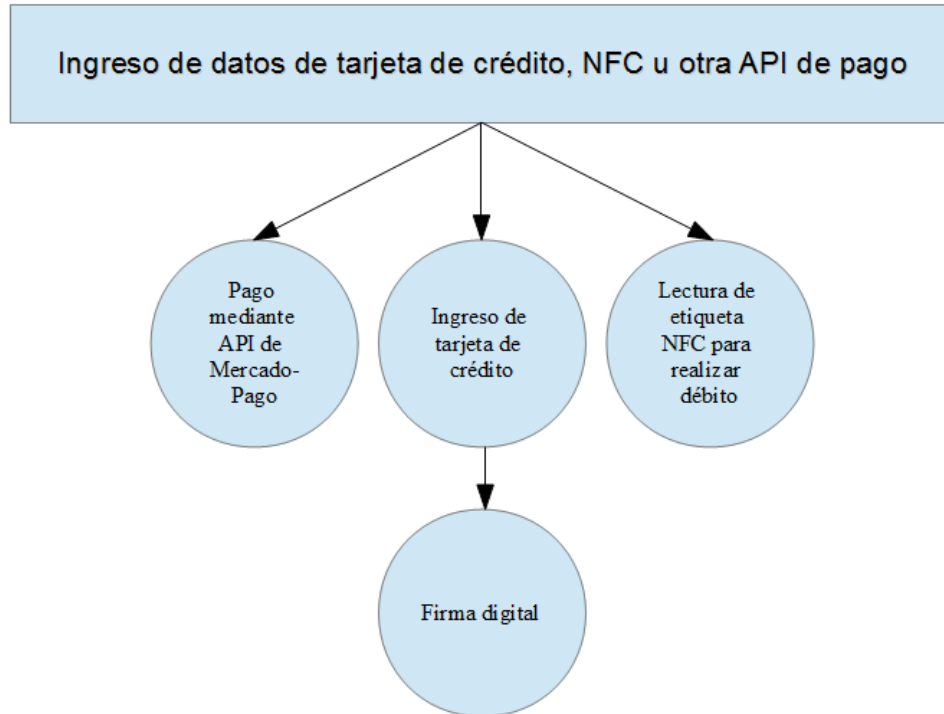


Figura 4.6: Mapa de la UI con las distintas variantes de pagos

4.4.2. Ingreso de pagos, cliente y vendedor

El objetivo de esta ventana es la de poder ingresar la forma de pago sobre la compra del cliente, brindando la flexibilidad al comprador de optar por distintas formas de pagos para una misma compra.

Para ello, el sistema permite elegir más de un método de pago y los montos los cuales el cliente quiera para cada uno de los métodos de pagos elegidos.

En esta parte del ingreso de datos de la venta se puede asociar la venta a un cliente cuyos datos pueden estar cargados en el sistema o en el caso de ser un cliente nuevo, este pueda ser incorporado al sistema mediante el ingreso de sus datos en una nueva ventana de clientes.

Un listado de los vendedores se puede observar en la parte inferior de la ventana para poder asociar el vendedor a la venta y poder utilizar esa información al momento de generar el ticket de la venta.

Las siguientes ventanas corresponden a las que se visualizan de acuerdo a las distintas formas de pago que el cliente haya elegido para pagar.



Figura 4.7: Activity de ingreso de productos para la venta

Pago

Total de Venta \$345
Cliente: Final

Agregar Cliente

Formas de Pagos

Efectivo ☐

Tarjeta de Crédito ☐

Dispositivo NFC ☐

Paypal ☐

MercadoPago ☐

Pagos a Recibir

Efectivo - \$200 ☐

Tarjeta de Crédito - \$145 ☐

0

Agregar Pago

Eliminar Pagos Seleccionados

Vendedor

Gabriel ☐

Thiago ☒

Lorena ☐

Pasar al Próximo Paso

Figura 4.8: Activity de ingreso de métodos de pagos, cliente y vendedor.

4.4.3. Ingreso de datos de tarjeta de crédito

Aquí es donde...

4.5. Listado de clientes

4.6. Opciones

4.7. Administración de stocks

4.8. Estadísticas

4.9. Cierre de caja

5. La Terminal de Ingreso de Datos (TID)

Introduccion.

5.1. Lectura de un codigo unidimensional utilizando el foco automatico de la camara

Escrito.

5.2. Firma Digital

Escrito.

5.3. Comunicacion Bluetooth

Escrito.

6. Diseñ;o del un *bootloader* para la IH!

6.1. El formato HEX de Intel

7. Dise o del un *bootloader* para la IH!

7.1. El formato HEX de Intel

aqui puedo escribir sobre robotium y las pruebas unitarias desarrolladas para este Trabajo Final de Especializaci n (TFE).

Conclusión

Se desarrolló una aplicación capaz de interpretar mensajes SMS y poder realizar distintas acciones de acuerdo a los datos interpretados. Se logró la conexión y transferencia de datos entre la **PC! (PC!)** y un Teléfono Celular (TC) por medio de la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) *JSMSEngine* el cual sus clases principales se incorporaron a la aplicación final. En éstas clases se definen los métodos que permiten leer y enviar mensajes SMS a la red GSM utilizando al TC en el modo de módem.

Este trabajo incorpora un framework para el tratamiento de los mensajes SMS que puede ser de mucha utilidad a futuros desarrolladores que quieran incorporarlo a aplicaciones que requieran el uso de los mensajes SMS.

La idea de este Trabajo Final de Carrera (TFC) fue no solamente el desarrollo de una aplicación, sino la de poder compararla con distintas aplicaciones que actualmente circulan en el mercado, para que cada uno pueda sacar conclusiones de las funcionalidades que cada uno presta y del precio que se pueda llegar a manejar en el caso de que este TFC sea utilizado de forma comercial.

Finalmente, uno de los propósitos es la continua investigación y desarrollo de esta aplicación para poder producir mejoras en los servicios prestados actualmente y generar nuevos servicios para poder incrementar el poder de esta aplicación en el futuro.

Apéndice

A. Siglas y acrónimos

TFE	Trabajo Final de Especialización
TPV	Terminal de Punto de Venta
DPT	Dispositivo de Pantalla Táctil
DTC	Dispositivo de Telefonía Celular
NFC	Near Field Communication (Comunicación de Campo Cercano)
TID	Terminal de Ingreso de Datos
SMD	Servicio de Manejo de datos
APIs	Application Programming Interfaces (Interfaces de Programación de Aplicaciones)
LTM	Lector de Tarjetas Magnéticas
WS	Servicios Webs (Web Services)
SIG	Sistema Integral de Gestión
SDK	Software Development Kit (Kit de Desarrollo de Software)
UI	User Interface (Interfaz de Usuario)
QR	Quick Response (Respuesta Rápida)
OS	Sistema Operativo
IPC	Interprocess Communication (Comunicación Interprocesos)
JIT	Just in Time Compiler (Compilador En Tiempo Real)
SDKT	Software Development Kit (Kit de Desarrollo de Software) (SDK) Tools
PT	Platform Tools
SDKSP	SDK Starter Package

ASM	Android SDK Manager
AVD	Android Virtual Device
AVDM	Android Virtual Device Manager
DDMS	Dalvik Debug Monitor Server
IDE	Integrated Development Environment (plataforma integral de desarrollo)
ADT	Android Development Tools
SD	Secure Digital
ADB	Android Debug Bridge
JB	Jelly Bean
ICS	Ice Cream Sandwich
DVM	Dalvik Virtual Machine (Máquina Virtual Dalvik)
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
BD	Base de Datos
IDE	Entorno Integrado de Desarrollo
TC	Teléfono Celular
TFC	Trabajo Final de Carrera

B. Glosario

bitrate Cantidad de informaci3n (bits) que se env3a o almacena en un tiempo determinado.

buffer Memoria o regi3n de una memoria para el almacenamiento temporal de datos.

codec Abreviatura del t3rmino Codificador-Decodificador. Describe una especificaci3n desarrollada en software, hardware o una combinaci3n de ambos, capaz de transformar un archivo con un flujo de datos o una se3al. [Consulta: 24 de Abril de 2007].
<<http://es.wikipedia.org/wiki/C%C3%B3dec>>

codificaci3n Forma que toma la informaci3n que se intercambia entre un emisor y un receptor de un lazo informaci3tico.

ratio Raz3n o porcentaje de ancho de banda utilizado.

hexadecimal El sistema hexadecimal, a veces abreviado como hex, es el sistema de numeraci3n posicional de base 16 –empleando por lo tanto 16 s3mbolos–. [Consulta: 20 de Febrero del 2008].
<<http://es.wikipedia.org/wiki/Hexadecimal>>

layout T3rmino en el idioma ingl3s que se utiliza para referirse a la disposici3n de los elementos en una composici3n.

linkedlist Consiste en una lista enlazada de objetos pertenecientes a una clase determinada.

ocurrencia Los datos que la Base de Datos (BD) contiene en un determinado momento.

octeto T3rmino utilizado para referirse a los ocho bits que conforman un byte.

protocolo Conjunto de est3ndares, normas y formatos para el intercambio de datos que asegura la uniformidad entre ordenadores y aplicaciones. [Consulta: 10 de Marzo de 2008].
<pulsar.ehu.es/pulsar/glosario/glosario_P>

reinicializar Volver a inicializar.

roaming Posibilidad que se le brinda a un abonado, a un servicio de telefonía celular, de moverse de un área de cobertura a otra de un diferente país sin la interrupción del servicio ni pérdidas de conectividad.

serialización

serializar Hacer serial algo. En este texto se refiere al fraccionamiento de tramas de datos en partes más pequeñas como ser un byte.

serialización Acción y efecto de serializar.

septeto Conjunto de 7 bits.

setear Viene de la palabra en inglés “set” que indica el establecer la configuración de un programa o componente físico para que funcione correctamente.

string Nombre que reciben las cadenas de texto en muchos lenguajes de programación.

swing Biblioteca gráfica para Java que forma parte de las **JFC! (JFC!)**. Incluye widgets para **IG! (IG!)** de usuario tales como cajas de texto, botones, despleables y tablas. [Consulta: 10 de Marzo de 2008].

transcodificación Es la conversión directa (De digital a digital) de un codec a otro, en general con pérdida de calidad.

tupla

query Término en el idioma inglés que en el ámbito informático se traduce como “consulta”

widgets Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual. [Consulta: 10 de Marzo de 2008].
<<http://es.wikipedia.org/wiki/Widgets>>

Bibliografía

- Google (2012a). Android developers. Web. [Consulta: 25 de Noviembre de 2012].
<<http://developer.android.com/about/dashboards/index.html>>.
- Google (2012b). Android developers. Web. [Consulta: 25 de Noviembre de 2012].
<<http://developer.android.com/tools/sdk/eclipse-adt.html>>.
- Google (2012c). Android developers. Web. [Consulta: 25 de Noviembre de 2012].
<<http://developer.android.com/tools/help/emulator.html>>.
- Google (2012d). Android developers. Web. [Consulta: 25 de Noviembre de 2012].
<<http://developer.android.com/tools/devices/index.html>>.
- Google (2012e). Android sdk. Web. [Consulta: 25 de Noviembre de 2012].
<<http://developer.android.com/about/start.html>>.
- Wikipedia (2012a). Android history. Web. [Consulta: 25 de Noviembre de 2012].
<http://es.wikipedia.org/wiki/Android#cite_note-9>.
- Wikipedia (2012b). Android sdk. Web. [Consulta: 25 de Noviembre de 2012].
<http://es.wikipedia.org/wiki/Desarrollo_de_Programas_para_Android>.
- Wikipedia (2012c). Dalvik virtual machine. Web. [Consulta: 25 de Noviembre de 2012].
<<http://es.wikipedia.org/wiki/Dalvik>>.