

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 Especialización a mis padres Trigidia y Luis, por brindarme y aconsejarme siempre lo mejor para mi bienestar, mi carrera como profesional, como persona e hijo.

Resumen

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

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, ingreso de mercadería, etcétera, así como también posibilita la visualización y manejo de información estadístico para el control y análisis de los movimientos que surgen en un comercio.

La elaboración de este TFE consiste de tres aplicaciones. Una aplicación representa al TPV implementado en un Dispositivo de Pantalla Táctil (DPT) de 7 pulgadas cuyo sistema operativo es AndroidTM, la otra aplicación funciona bajo un Dispositivo de Telefonía Celular (DTC) con AndroidTM y cuya finalidad es la de poder brindar un dispositivo que funcione como Terminal de Ingreso de Datos (TID) al sistema, y por último se desarrolla un Sistema Integral de Gestión (SIG) cuya función es la de recibir, procesar y enviar información desde y hacia el TPV.

Este sistema permite utilizar un dispositivo de Lector de Tarjetas Magnéticas (LTM) que funciona conectado a la TID como una interfaz de ingreso de datos para el TPV.

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 elaboran distintas pruebas con etiquetas NFC simulando transacciones de compras cuyos resultados fueron expuestos en este TFE.

Se introduce el uso de algunas Application Programming Interfaces (Interfases de Programación de Aplicaciones) (APIs) existentes como las de la red social TwitterTM y el sistema de cobro MercadoPagoTM para extender las características que el sistema puede ofrecer a los usuarios.

Índice general

Introducción	1
Objetivos	3
Metodología	4
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 Android	12
1.4.1. Ejemplos de algunas herramientas para el desarrollo de aplicaciones	13
2. El desarrollo en Android	17
2.1. La Dalvik Virtual Machine (Máquina Virtual Dalvik) (DVM)	17
2.2. La arquitectura Android	17
2.3. Partes que componen un proyecto de desarrollo Android	18
2.3.1. Las activities	18
2.3.2. El ciclo de vida de una activity	19
2.3.3. El back stack de activities	20
2.3.4. Los Intents	21
2.3.5. El Manifest	21
2.3.6. Los layouts	22
2.3.7. Los eventos de User Interface (Interfaz de Usuario) (UI) . .	22
2.3.8. El Style	22
2.3.9. El Theme	23
2.3.10. El AdapterView	23

2.3.11. Los menús	23
2.3.12. Los Dialogs	23
2.3.13. Los Resources	23
2.3.14. El AsyncTask	23
2.3.15. El Content Provider	24
2.4. Los Servicios	24
2.5. Los secretos de performance	25
2.5.1. Mitos de performance	26
2.5.2. Siempre medir	26
2.6. La interacción con servicios remotos	26
2.7. El Dalvik Debug Monitor Server (DDMS)	26
2.8. El LogCat	26
3. El Terminal de Punto de Venta (TPV)	28
3.1. El código fuente y paquetes que componen el proyecto	28
3.2. Las librerías utilizadas	28
3.3. El Quick Response (Respuesta Rápida) (QR)	28
3.4. El uso de la cámara óptica	30
3.5. La Base de Datos (BD) SQLite	30
3.6. El LTM	31
3.7. La tecnología NFC	31
3.8. Los gráficos estadísticos	31
4. El Sistema Integral de Gestión (SIG)	33
4.1. Descripción de los archivos que componen el proyecto en Eclipse	33
4.2. Librerías utilizadas para el SIG	33
5. La User Interface (Interfaz de Usuario) (UI)	36
5.1. Acceso a la aplicación (login)	36
5.2. Menu principal	36
5.3. Abrir caja	41
5.4. Agregar venta	41
5.4.1. Ingreso de datos de los productos a vender	41
5.4.2. Ingreso de pagos, cliente y vendedor	43
5.4.3. Ingreso de datos de tarjeta de crédito	46
5.4.4. Firma digital	46
5.4.5. Realización de pago mediante la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) de MercadoPago	46
5.4.6. Lectura y escritura de datos mediante dispositivo NFC	49
5.4.7. Generación de ticket	49
5.4.8. Marketing a través de la red social Twitter	53
5.5. Listado de clientes	53
5.6. Opciones	53

ÍNDICE GENERAL	VI
5.7. Administración de stocks	60
5.8. Estadísticas	60
5.9. Cierre de caja	60
6. Testing Unitario y Funcional de la aplicación	64
6.1. Testing Unitario en Android	64
6.2. Testing Funcional	64
Conclusión	65
Apéndice	66
A. Siglas y acrónimos	67
B. Glosario	69

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.	Las versiones de Android en el tiempo	11
1.3.	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
1.4.	Crecimiento de dispositivos Android	12
1.5.	La tienda de aplicaciones Google Play	16
2.1.	La arquitectura Android	18
2.2.	Ciclo de vida de una activity	20
2.3.	Diagrama del back stack de activities	21
2.4.	Jerarquía de vistas y layouts	22
3.1.	Paquetes y archivos que componen el proyecto MiPos en el entorno Eclipse	29
3.2.	Paquete de librerías utilizadas para el proyecto MiPos	30
3.3.	El LTM	32
3.4.	La etiqueta NFC	32
4.1.	Paquetes y archivos que componen el SIG en el entorno Eclipse	34
4.2.	Librerías utilizadas para el SIG	35
5.1.	Mapa de la UI con sus activities y relaciones entre las mismas	37
5.2.	Mapa de la UI (continuación)	38
5.3.	Activity de acceso a la aplicación	39
5.4.	El menú principal	40
5.5.	Activity de apertura de caja	42
5.6.	Mapa de la UI con las distintas variantes de pagos	43
5.7.	Activity de ingreso de productos para la venta	44
5.8.	Activity de ingreso de métodos de pagos, cliente y vendedor.	45
5.9.	Activity para ingresar los datos de la tarjeta de crédito.	47

5.10. Activity para registrar la firma del cliente	48
5.11. Ingreso al sistema de MercadoPago para poder realizar el cobro de la venta	50
5.12. Pantalla que indica la activación del dispositivo NFC para realizar la lectura y escritura de datos	51
5.13. Pantalla que surge de una transacción no aprobada	52
5.14.	54
5.15.	55
5.16.	56
5.17. La aplicación brinda la posibilidad de enviar mensajes a través de la API de Twitter	57
5.18. Listado de compras de un cliente registrado en el sistema	58
5.19. Las opciones de configuración	59
5.20. El activity para la administración del stock del comercio	61
5.21. Gráfico estadístico generado a partir de los datos almacenados en el sistema	62
5.22. Activity para mostrar los distintos totales de la caja abierta previo al cierre	63

Índice de tablas

1.1. Versiones de Android	9
-------------------------------------	---

Introducción

La necesidad de cambiar la forma en la que los sistemas de gestión y stock funcionan actualmente genera una idea de una nueva metodología en la que el cliente tenga una experiencia de compra diferente, sumamente atractiva y que pueda lograr marcar una diferencia por sobre la competencia.

Esta nueva metodología pude vivirla y comprobarla a partir 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 realidad 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).

Objetivos

- Desarrollar software para el sistema operativo Android.
- Conocer las posibilidades y limitaciones que un dispositivo móvil pueda ofrecer a un usuario.
- Conocer nuevas tecnologías para realizar transacciones de compras.
- Ofrecer una solución de fácil manejo y mantenimiento para los usuarios de un sistema de gestión.
- Explicar distintas librerías y herramientas que permitan extender las posibilidades que Android ofrece.

Metodología

Para este TFE¹ se utilizaron dispositivos con sistemas operativos Android™. Se desarrollaron aplicaciones en la plataforma Eclipse™ con el Software Development Kit (Kit de Desarrollo de Software) (SDK) de Android™ y se llevó a cabo la investigación y utilización de distintas librerías para cumplir los objetivos planteados y requeridos para un Sistema Integral de Gestión (SIG).

Denominamos SIG a toda la plataforma de venta, en donde el funcionamiento de cada una de sus partes y sus interacciones entre estas permite cumplir la finalidad propuesta para este TFE la cual es generar un sistema que pueda cumplir con lo que un sistema de gestión actual pueda brindar.

El SIG se encuentra desarrollado en tres partes:

1. El Terminal de Punto de Venta (TPV) que se encuentra representado por 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. El objetivo de esta parte del SIG es brindar al vendedor una plataforma de cobro y para el administrador, una herramienta de gestión.

Se analizaron 4 formas posibles de poder realizar un pago utilizando este TPV:

- a) 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.
- b) Pago en tarjeta de crédito: el cliente pasa su tarjeta por el Lector de Tarjetas Magnéticas (LTM) y los datos se envían a un gateway de pagos para autenticar los datos de la tarjeta y autorizar la transacción del mismo.

¹A 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 67).

- c) 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.
 - d) Pago utilizando el sistema de cobro MercadoPago™: a través de la implementación del SDK que se ofrece en forma gratuita, se permite la posibilidad de realizar pagos utilizando una cuenta de MercadoPago™, permitiendo así ofrecer otra opción más a los clientes.
2. La Terminal de Ingreso de Datos (TID) constituido por un Dispositivo de Telefonía Celular (DTC) con Android™ conectado a un LTM que consiste en un dispositivo que se conecta a la entrada de audio de 3.5 milímetros y actúa como un sensor de sonido que permite decodificar los datos almacenados en una tarjeta de crédito para utilizarlos en el TPV.
 3. Un *back-end* elaborado por medio de una aplicación *Java* que permite funcionar como un Servidor de Manejo de datos (SMD), recibiendo y enviando datos al TPV las cuales son persistidos mediante la implementación de una base de datos *MySQL* y cuyo servidor que contiene a la aplicación corresponde a un *Tomcat*.

Las distintas etapas de elaboración de este TFE consisten en las siguientes:

- **Desarrollo de una aplicación Android™ que representa a un TPV sobre un 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, simulando la interacción con el SMD. A esta aplicación se agregaron distintas librerías tanto para el uso de la cámara del dispositivo, la lectura y generación de códigos Quick Response (Respuesta Rápida) (QR), la generación de documentos PDF, entre otras. El uso de la cámara, además de utilizarse como lector de códigos también se implementó para la toma de fotos de los productos que ingresan en el stock del local.

- **Primeras pruebas de comunicación entre dispositivos**

Se realizó una aplicación de prueba en el 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 de un LTM, dando por nombre a esta parte del sistema el de una 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 SIG.

- **Inclusión del manejo de transacciones con tarjetas de créditos**

Se agregó la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) de MercadoPagoTM 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 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 NFC**

Para poder utilizar la tecnología NFC y poder sacar provecho del lector incorporado que trae el DPT modelo *Nexus* de marca *ASUS*, se agregó en la aplicación del TPV un módulo que permite hacer lectura y escritura de etiquetas NFC. Las pruebas se realizaron mediante una etiqueta que contiene información de una cuenta 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, 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 la API de TwitterTM para poder brindar al cliente la posibilidad de que pueda ingresar con su cuenta y envíe un mensaje relacionado a la venta a todos sus contactos para que se pueda generar una publicidad viral y de esta manera utilizar al SIG como una herramienta de marketing.

Resultados y Discusión

1. El Sistema Operativo (OS) Android

En primer lugar se realizará un breve descripción del sistema operativo para dispositivos móviles Android, con sus herramientas y características que brinda a sus usuarios.

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 Google™.

Fue desarrollado inicialmente por Android Inc., una firma comprada por la compañía Google™ en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio. Las unidades vendidas de teléfonos inteligentes con Android se ubican en el primer puesto en los Estados Unidos, en el segundo y tercer trimestres de 2010, con una cuota de mercado de 43,6 % en el tercer trimestre. A nivel mundial alcanzó una cuota de mercado del 50,9 % durante el cuarto trimestre de 2011, más del doble que el segundo sistema operativo (iOS de Apple, Inc.) con más cuota (ver figura 1.4).

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 (figura 1.5) 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 hard-

ware, 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™ (figura 2.1) 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 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++ (?).

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 americanos. En cada versión el postre elegido empieza por una letra distinta siguiendo el orden alfabético (?):

Tabla 1.1: Versiones de Android

Nombre	Versión	Traducción
Apple Pie	1.0	Tarta de manzana
Banana Bread	1.1	Pan de plátano
Cupcake	1.5	Magdalena glaseada
Donut	1.6	Rosquilla
Eclair	2.0/2.1	Tipo de pastel francés
Froyo	2.2	Yogur Helado
Gingerbread	2.3	Pan de jengibre
Honeycomb	3.0/3.1/3.2	Panal de miel
Ice Cream Sandwich	4.0	Sándwich de helado
Jelly Bean	4.1/4.1.2/4.2	Judía de gelatina
Key Lime Pie	5.0	Pastel de lima

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

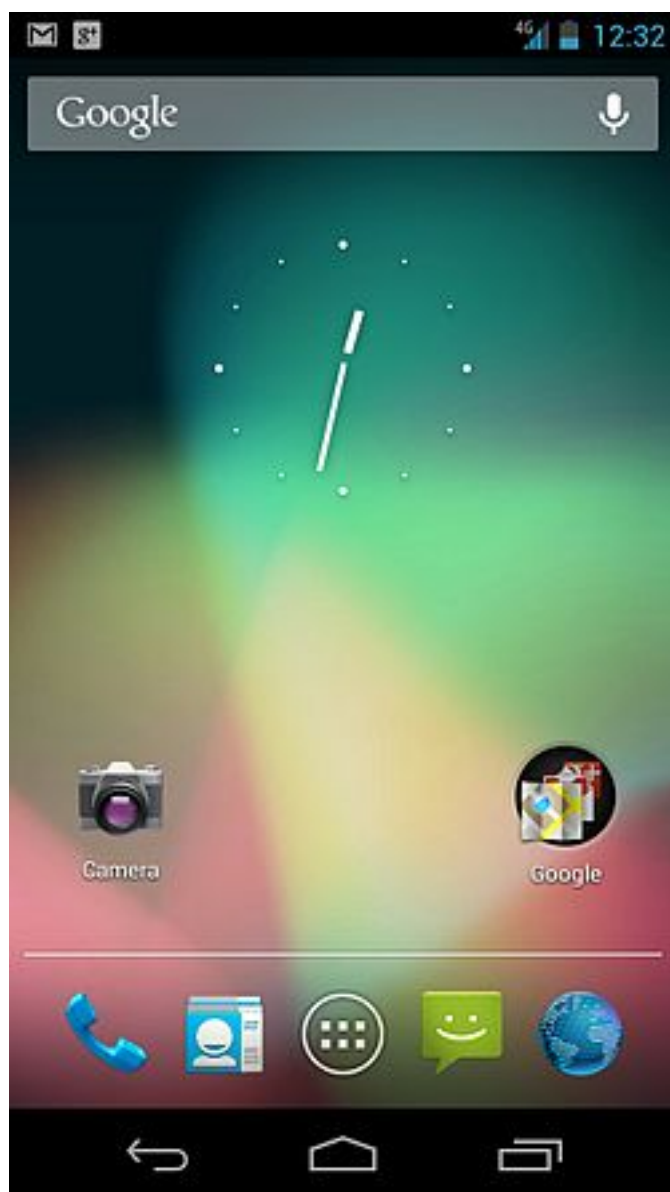


Figura 1.1: Pantalla inicial de Android 4.1 (Jelly Bean) ejecutándose en un Galaxy Nexus

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

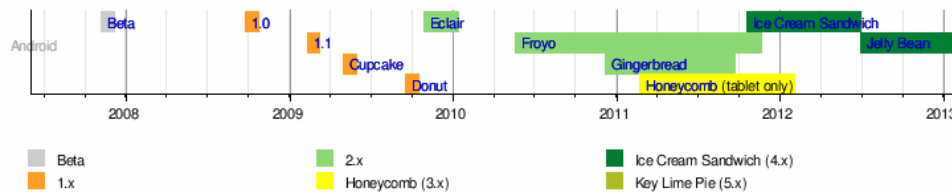


Figura 1.2: Las versiones de Android en el tiempo

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.

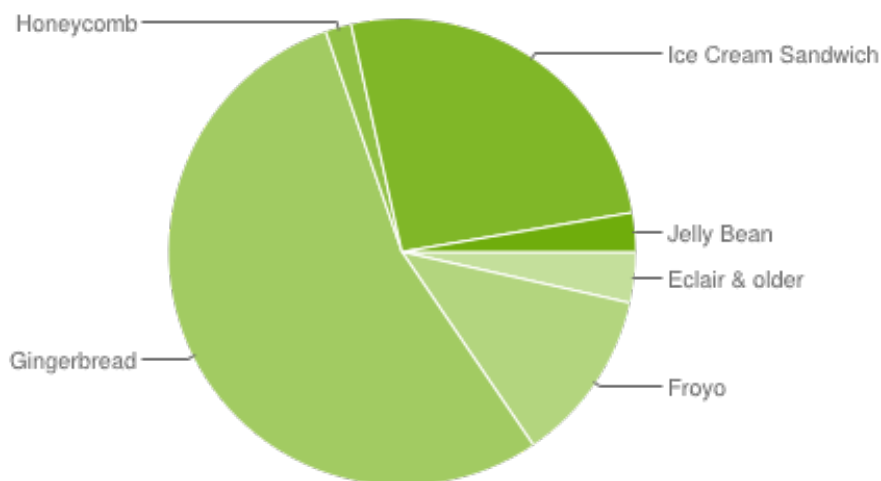


Figura 1.3: Distribución de las distintas versiones de Android de acuerdo a un análisis de datos recolectados en un período de 14 días finalizando el 1 de Noviembre de 2012



Figura 1.4: Crecimiento de dispositivos Android

- **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, 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, etcétera.

(?)

1.4. Las herramientas para trabajar con la plataforma Android

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 consiste en 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 QEMU¹, 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 Java Development Kit (JDK) y Apache Ant) para crear y depurar aplicaciones. Además, pueden controlarse dispositivos Android que estén conectados (por ejemplo, reiniciarlos, instalar aplicaciones en remoto, etcétera). (?)

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 Android Package (APK) incluye ficheros .dex (ejecutables Dalvik, un código intermedio compilado), recursos, entre otros.

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 depuración a las aplicaciones Android.
- Android Emulator: Un QEMU-based device-emulation tool que permite ser utilizado para diseñar, depurar, y probar las aplicaciones en un ambiente de runtime Android real.

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).

- **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, depurar 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 Eclipse™ que está diseñado para brindar un entorno integrado en el cual se puedan crear aplicaciones para Android.

ADT extiende las capacidades de Eclipse™ para poder rápidamente setear un proyecto nuevo de Android, crear la UI de la misma, agregar paquetes basados en la API del framework Android, depurar las aplicaciones y poder exportar los archivos .apk para la distribución en el Google Play™. (?)

El emulador de Android

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. (?)

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, etcétera.
- **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, etcétera) y la información de la tarjeta SD emulada se encuentra almacenada en esta área.

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. (?)



Figura 1.5: La tienda de aplicaciones Google Play

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 dar soporte y manejo a las mismas.

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

Dalvik es la máquina virtual que utiliza la plataforma para dispositivos móviles Android. El diseño fue realizado por Dan Bornstein con contribuciones de otros ingenieros de Google.

Dalvik está optimizada para requerir poca memoria y está diseñada para permitir la ejecución de 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. (?)

2.2. La arquitectura Android

La arquitectura del sistema operativo esta compuesto por las siguientes características (ver figura 2.1):

- 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 ejecutadas en la máquina virtual.

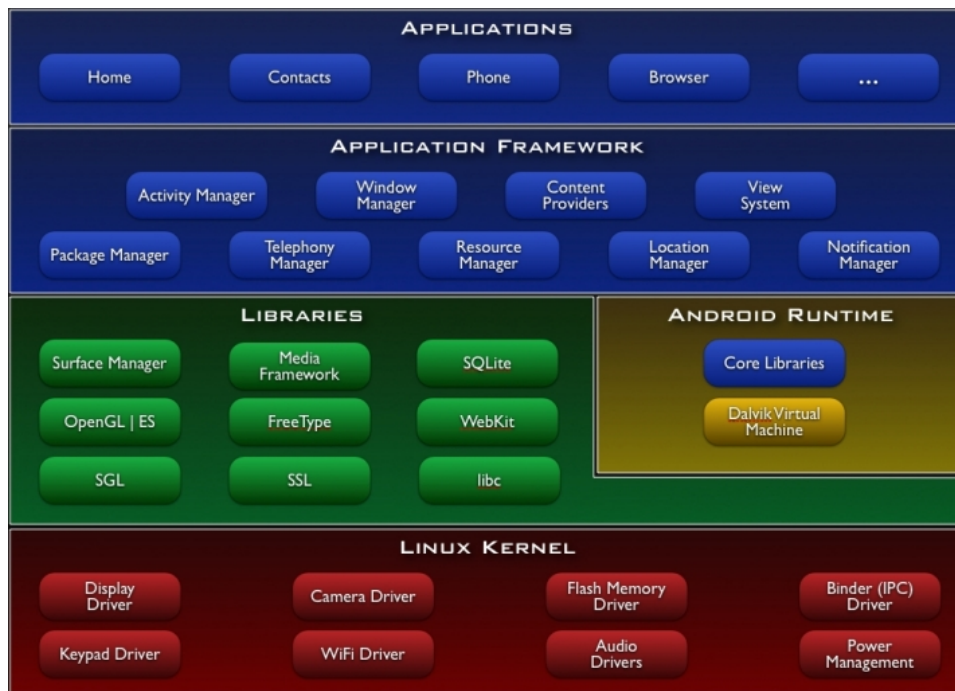


Figura 2.1: La arquitectura Android

- 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
 - 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

Una activity conforma las siguientes características:

- Consiste en 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 `setContentView`.

2.3.2. El ciclo de vida de una activity

Una activity tiene tres estados principales:

1. Active, actualmente en pantalla con foco.
2. Paused, perdió foco, continúa parcialmente visible.
3. 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 cualquiera de estos métodos.

Los siete métodos del ciclo de vida (figura 2.2) 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, para mantener recursos que afecten la UI.
- Primer plano, para mantener la interactividad con el usuario.

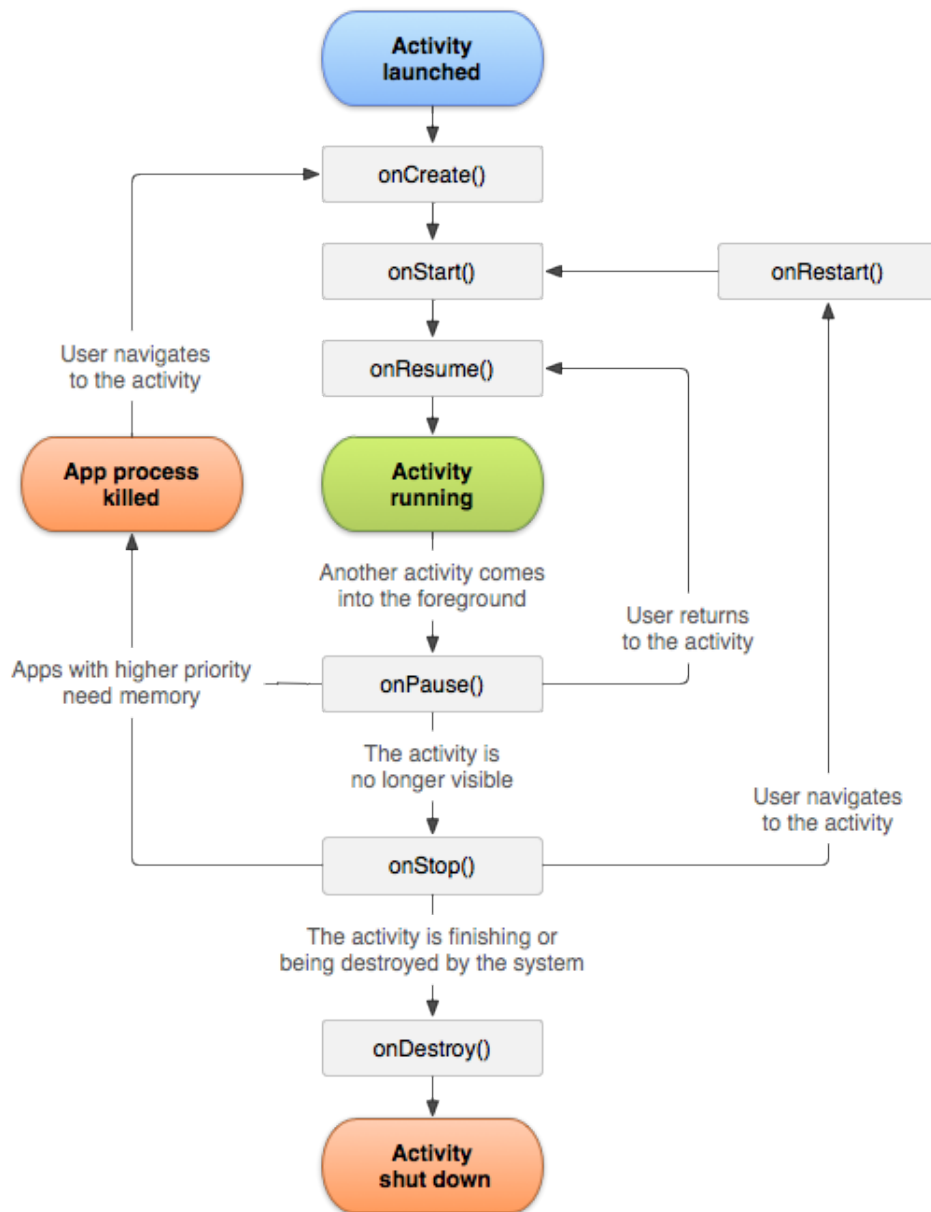


Figura 2.2: Ciclo de vida de una activity

2.3.3. El back stack de activities

Las activities son administradas mediante una pila denominada back stack en el orden en el cual cada una de estas activities se va creando.

Cuando una activity en pantalla inicia otra, la nueva activity se coloca al tope del back stack y toma el foco de la pantalla. La actividad anterior se mantiene en el

back stack, pero se detiene. Cuando se ingresa al estado de stop, el sistema retiene el estado actual de su User Interface (Interfaz de Usuario) (UI).

En el momento en el que el usuario presiona el botón de retroceso del dispositivo, la actividad actual se destruye y la anterior a esta ejecuta el método `resume` para volver recuperar su estado y pueda ser visualizado en pantalla.

De ninguna forma las activities pueden cambiar su posición dentro del back stack, ya que justamente se opera como una pila de último en entrar, primero en salir. La figura 2.2 visualiza el comportamiento en el tiempo mostrando el progreso entre las distintas activities y como el back stack se encuentra formado en las distintas etapas.

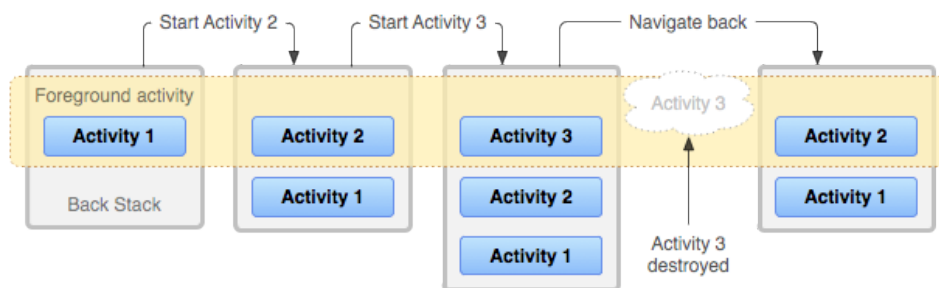


Figura 2.3: Diagrama del back stack de activities

2.3.4. Los Intents

Un intent es una descripción abstracta de una operación a ser ejecutada. Puede ser utilizado con el método `startActivity` para transferir información entre distintas activities, como una de las funciones más importantes, entre otras más.

Por medio de un intent, se hace disponible realizar late binding en tiempo de ejecución entre el código en diferentes aplicaciones. Básicamente consiste en una estructura de datos pasiva conteniendo una descripción abstracta de una acción a ser realizada.

Un intent puede referenciar un componente, y si la referencia no es explícita Android buscará el componente que mejor responda.

2.3.5. El Manifest

El manifest consiste en un archivo en formato `.xml` donde cada aplicación describe y declara todos sus componentes, sus permisos que requiera para acceder a partes protegidas de la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) y su interacción con otras aplicaciones, su nivel

mínimo de API Android requerido y todas las librerías a las que la aplicación hace uso.

Cada aplicación tiene su manifest que se incluirá en el archivo `.apk`.

2.3.6. Los layouts

El layout define la distribución de las vistas y contenedores, como la UI de una actividad o una aplicación tipo widget.

Los layouts pueden definirse:

- en el formato `.xml`: brindando un vocabulario que corresponde a las clases y subclases de `View`, aquellos los cuales están definidos para layouts y widgets.
- instanciando los elementos en tiempo de ejecución: la aplicación puede crear objetos del tipo `View` y `ViewGroup` (y manipular sus propiedades) programáticamente.

Cada tipo de layout se encuentra ubicado en una jerarquía de vistas (views) (figura 2.4) con parámetros de layout asociados a cada una de estas vistas.

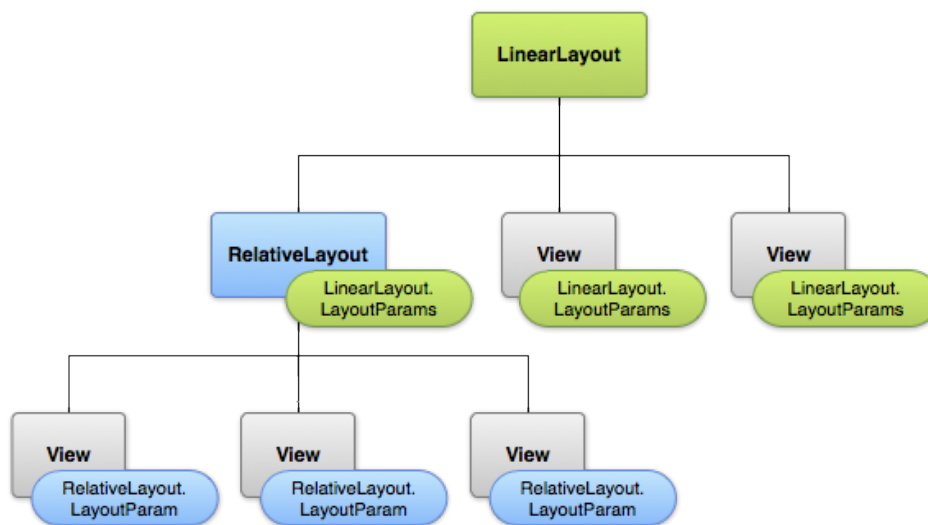


Figura 2.4: Jerarquía de vistas y layouts

2.3.7. Los eventos de UI

Son interfaces en la clase `View` para 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 `onLong 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 data sources, 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 database. 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 podrian 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. El codigo fuente y paquetes que componene el proyecto

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 librerias, 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

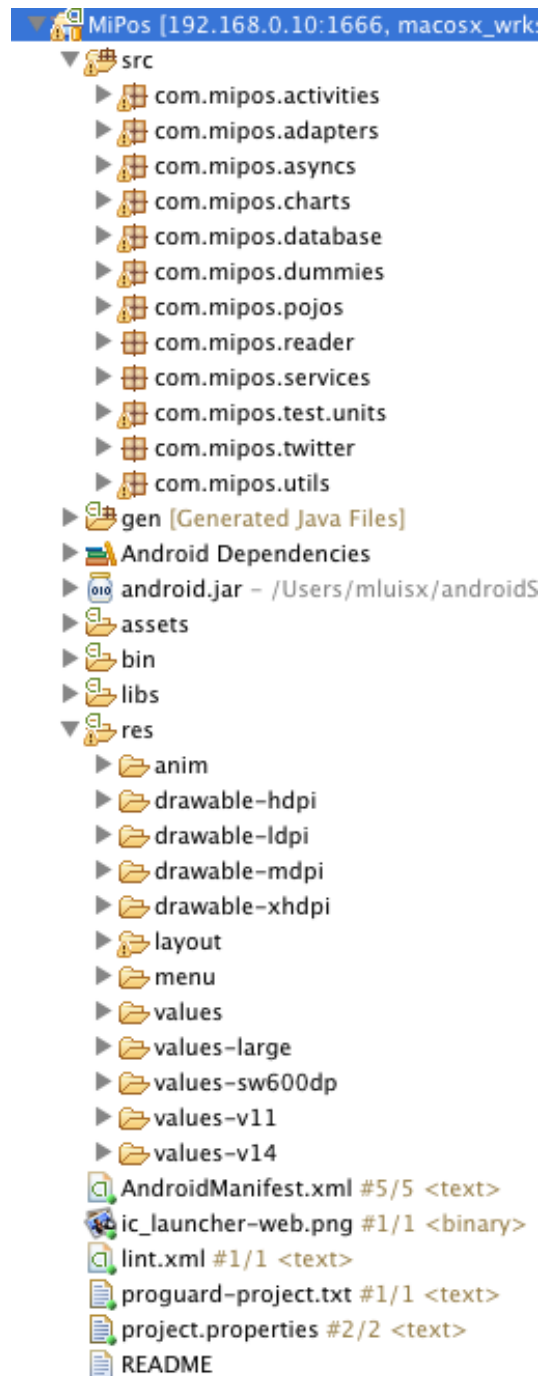


Figura 3.1: Paquetes y archivos que componen el proyecto MiPos en el entorno Eclipse

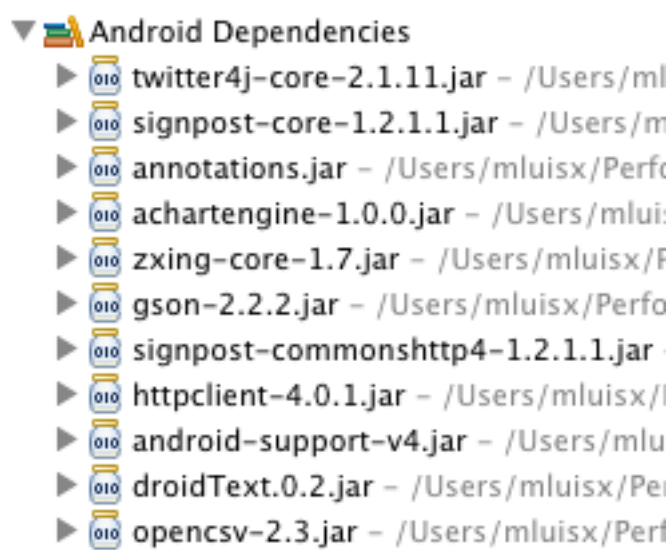


Figura 3.2: Paquete de librerías utilizadas para el proyecto MiPos

de presentación también se está haciendo común, simplificando 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



Figura 3.3: El LTM

activity. The model and the graphing code is well optimized such as it can handle and display huge number of values. AChartEngine 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 ultimos 6 meses como fue en el tiempo.
- 3) los rubros más vendidos en los ultimos 6 meses.
- 4) la venta con valor más bajo y más alto mes x mes (temperature range).



Figura 3.4: La etiqueta NFC

4. El Sistema Integral de Gestión (SIG)

Introducción:

4.1. Descripción de los archivos que componen el proyecto en Eclipse

A escribir...

4.2. Librerías utilizadas para el SIG

A escribir...

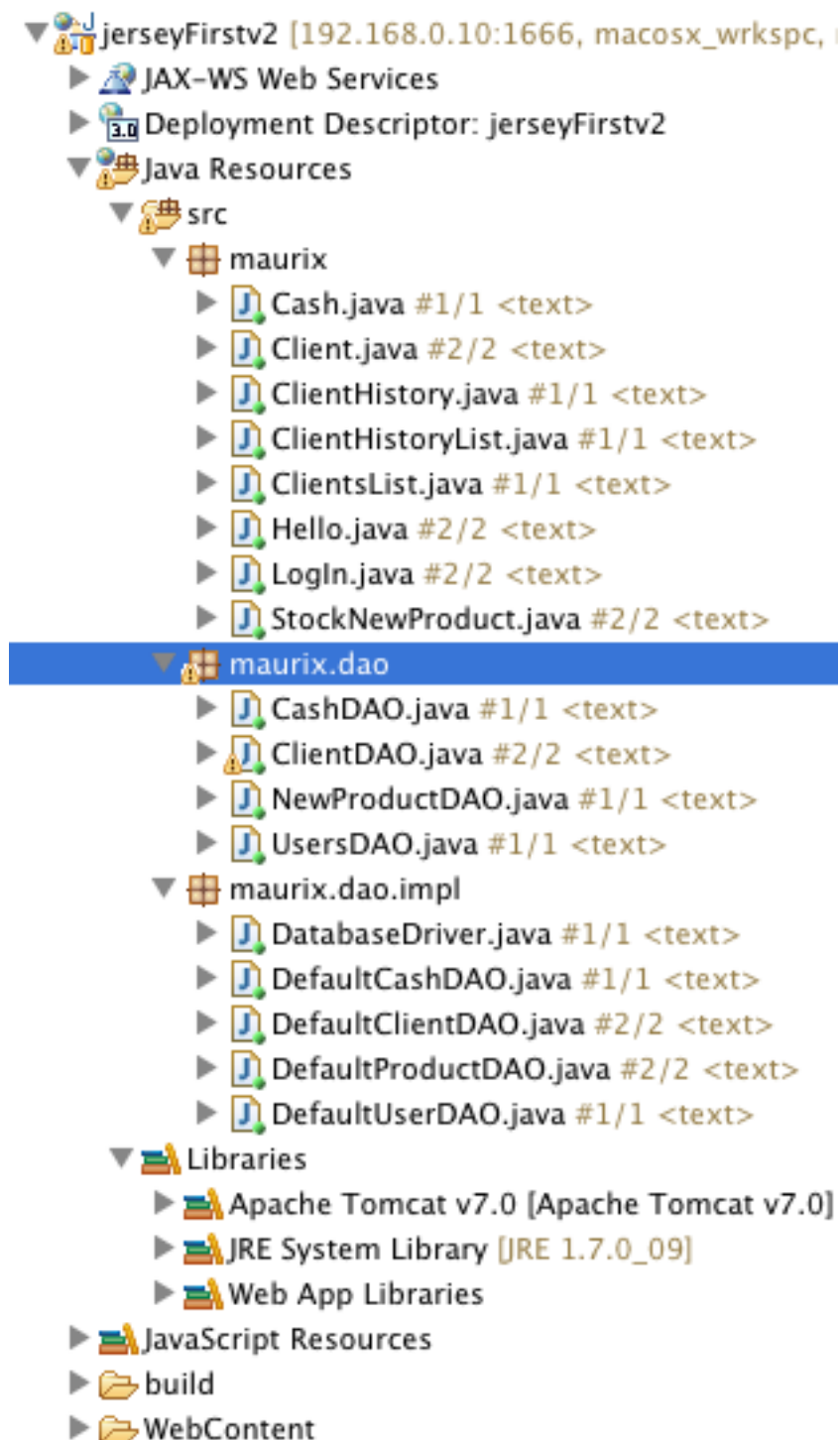


Figura 4.1: Paquetes y archivos que componen el SIG en el entorno Eclipse

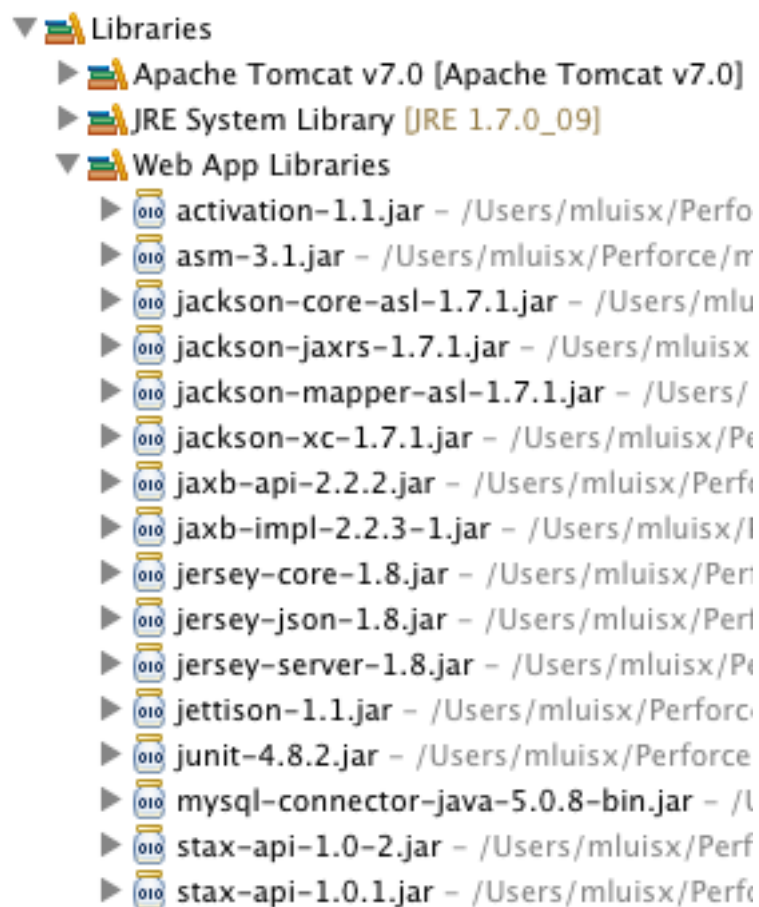


Figura 4.2: Librerías utilizadas para el SIG

5. 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.

5.1. Acceso a la aplicación (login)

Al iniciar la aplicación, la primer activity que se puede ver es la del acceso a la misma 5.3. Con un usuario y password se procede al ingreso del menú principal.

5.2. Menu principal

En el menú principal 5.4 podemos ver una serie de botones con las distintas opciones que la aplicación nos brinda. Todas las opciones son descritas a continuación.

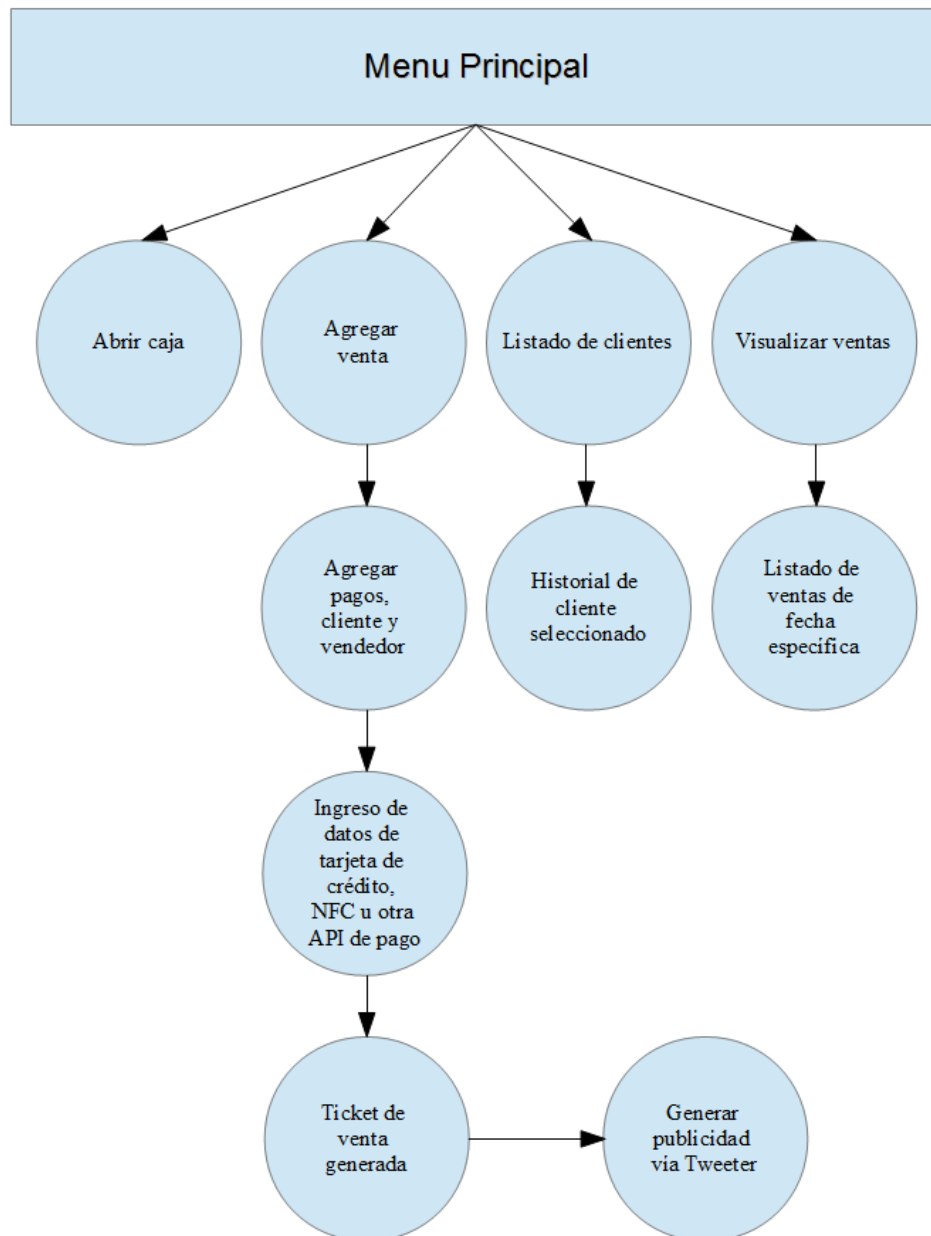


Figura 5.1: Mapa de la UI con sus actividades y relaciones entre las mismas

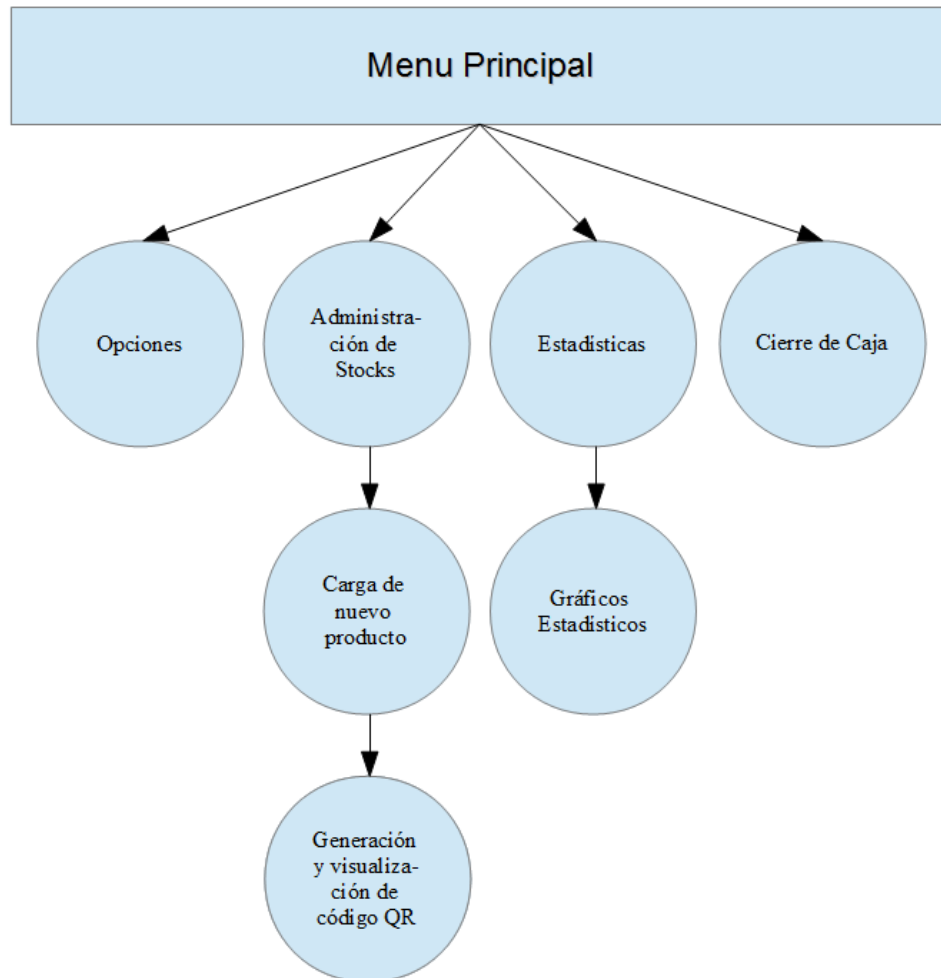


Figura 5.2: Mapa de la UI (continuación)

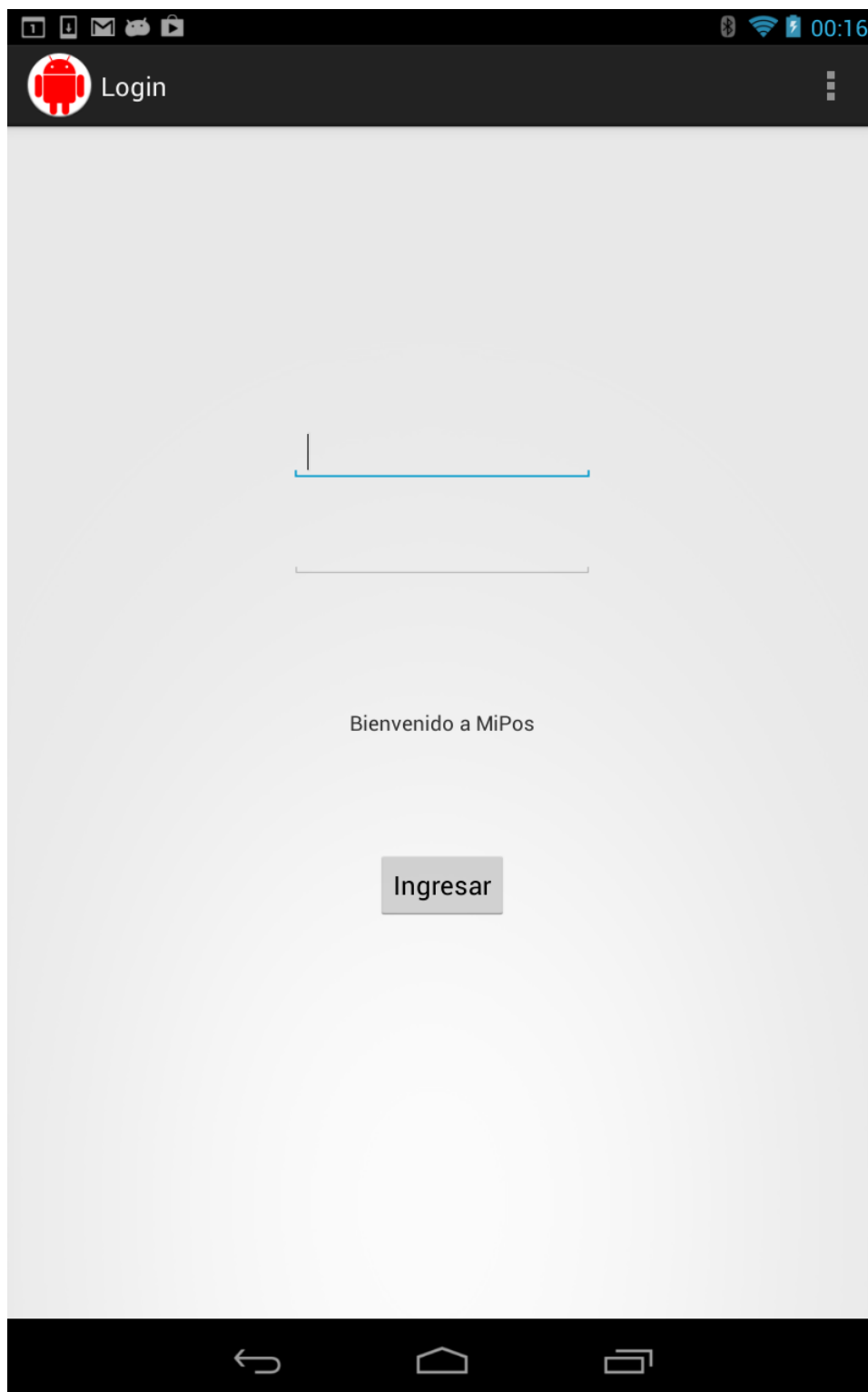


Figura 5.3: Activity de acceso a la aplicación

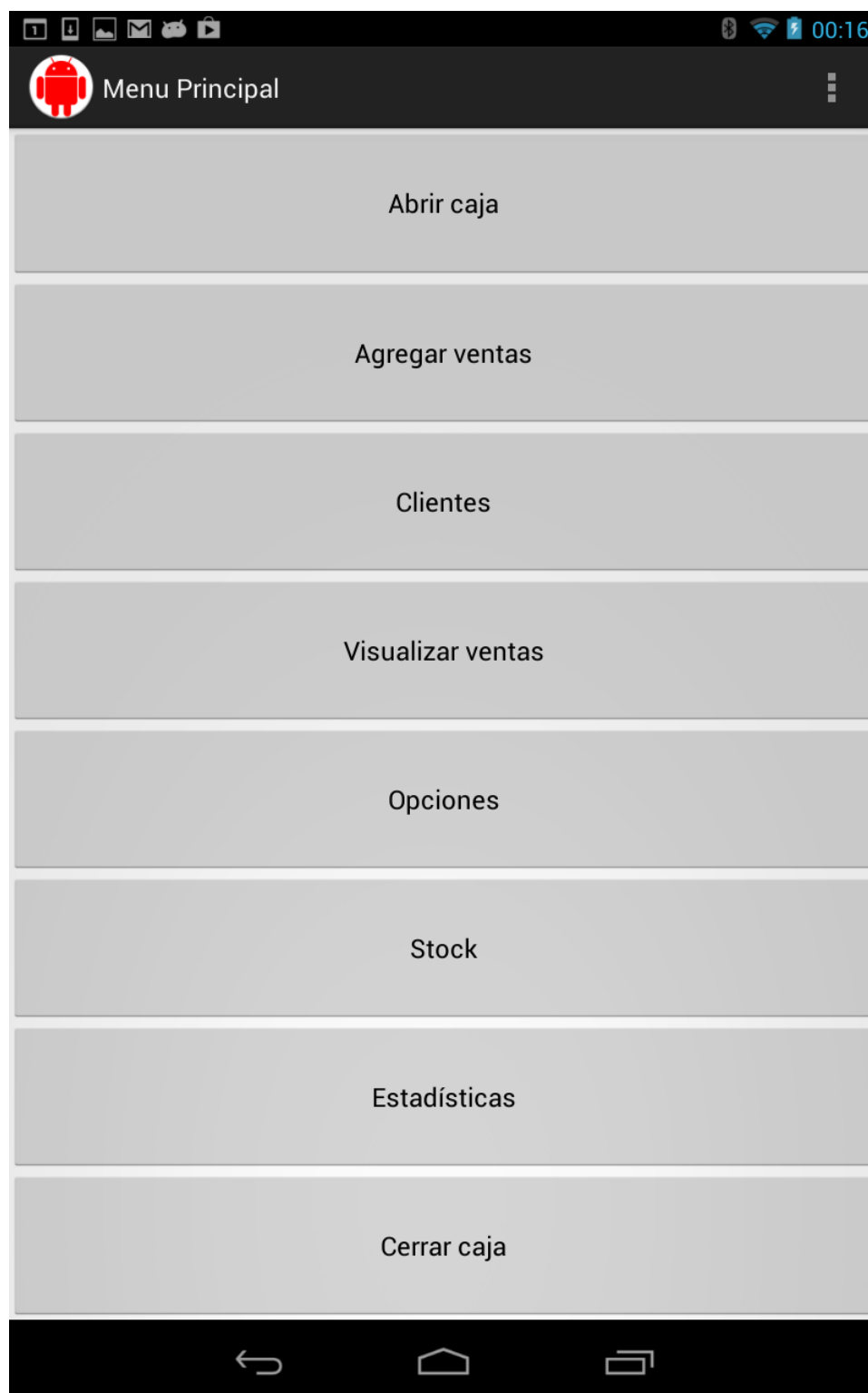


Figura 5.4: El menú principal

5.3. 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 5.5.

5.4. 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. Firma digital.
5. Realización de pago mediante la Application Programming Interface (Interfaz de Programación de Aplicaciones) (API) de MercadoPago.
6. Lectura y escritura de datos mediante dispositivo Near Field Communication (Comunicación de Campo Cercano) (NFC).
7. Generación de ticket.
8. Marketing a través de la red social Twitter.

5.4.1. Ingreso de datos de los productos a vender

En esta actividad 5.7 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.

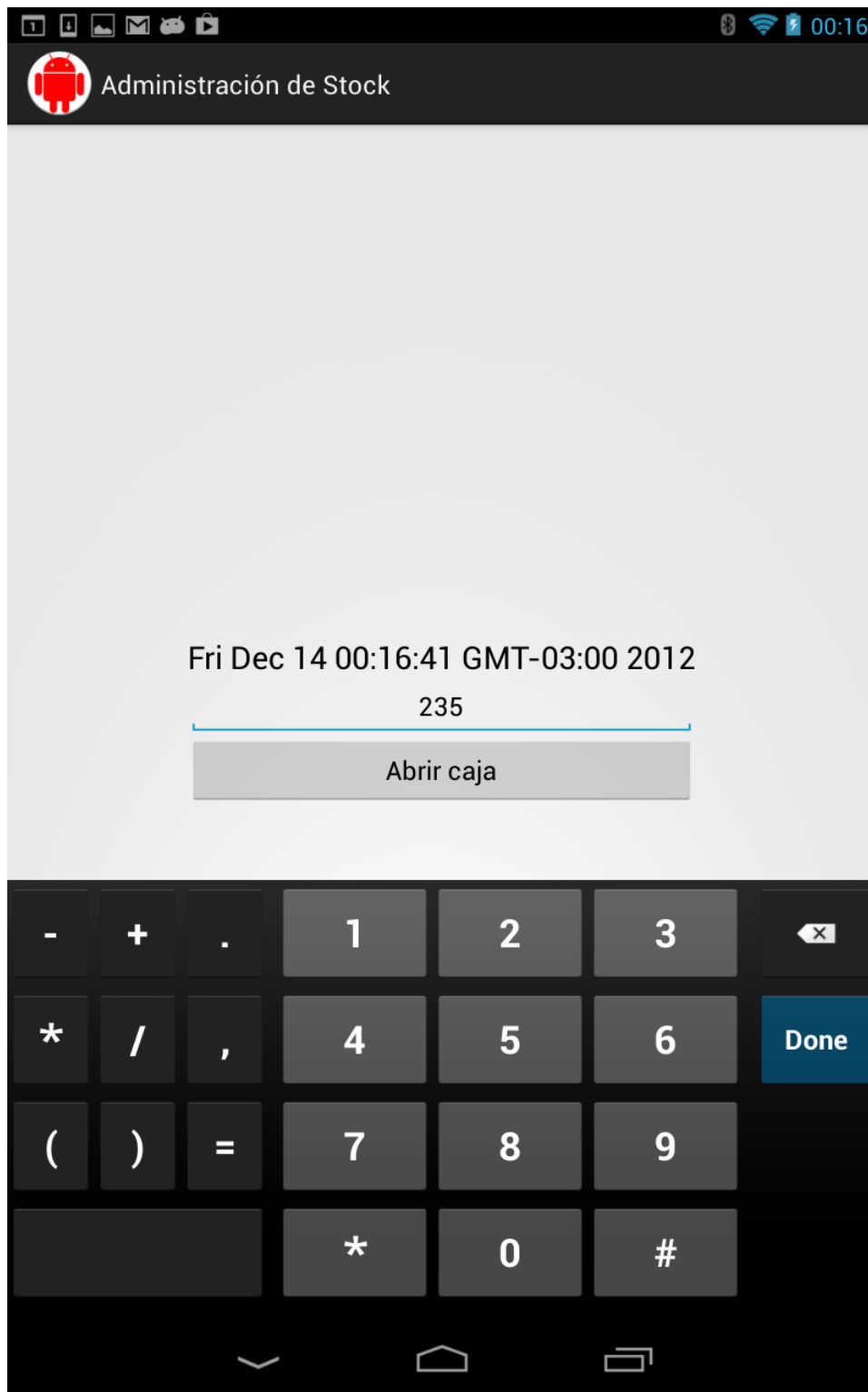


Figura 5.5: Activity de apertura de caja

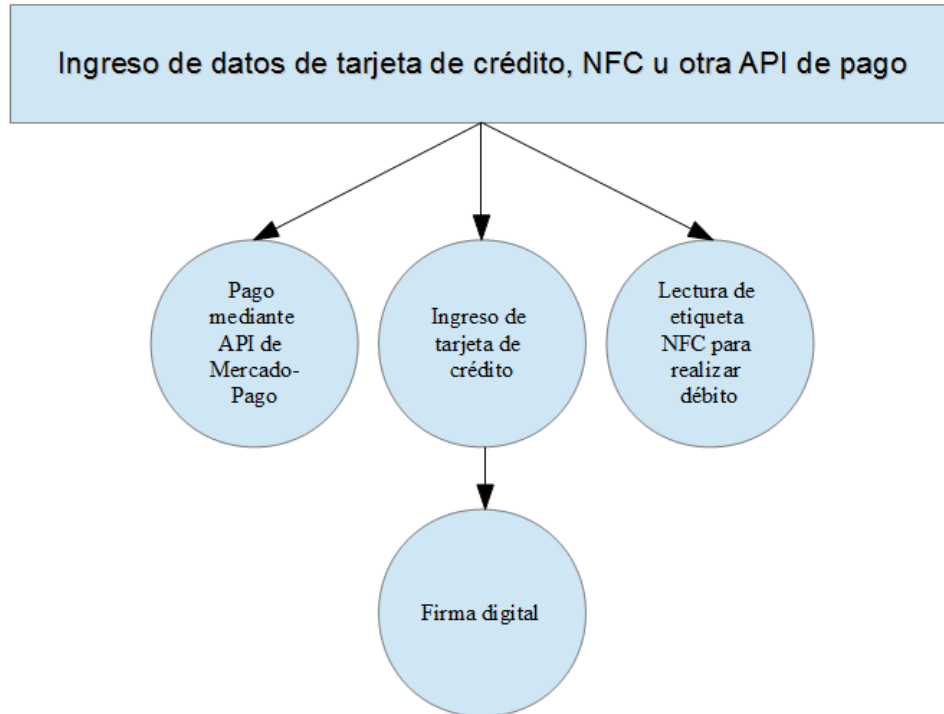


Figura 5.6: Mapa de la UI con las distintas variantes de pagos

5.4.2. 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 5.7: Activity de ingreso de productos para la venta

The screenshot displays the 'Pago' (Payment) activity. At the top, there's a header bar with an Android icon and the title 'Pago'. Below the header, the status bar shows various icons and the time 00:18. The main content area is divided into several sections:

- Total de Venta \$345**
Cliente: Final
- Agregar Cliente** (button)
- Formas de Pagos** (section header)
 - Efectivo ☐
 - Tarjeta de Crédito ☐
 - Dispositivo NFC ☐
 - Paypal ☐
 - MercadoPago ☐
- Pagos a Recibir** (section header)
 - Efectivo - \$200 ☐
 - Tarjeta de Crédito - \$145 ☐
 - 0** (text)
- Agregar Pago** (button)
- Eliminar Pagos Seleccionados** (button)
- Vendedor** (section header)
 - Gabriel ☐
 - Thiago ☒
 - Lorena ☐
- Pasar al Próximo Paso** (button)

The bottom of the screen features a standard Android navigation bar with back, home, and recent apps icons.

Figura 5.8: Activity de ingreso de métodos de pagos, cliente y vendedor.

5.4.3. Ingreso de datos de tarjeta de crédito

En esta parte de la aplicación se se ingresa el número, fecha de vencimiento y código de seguridad de la tarjeta de crédito del comprador. Es aquí donde la aplicación puede utilizar como dispositivo de entrada de datos al Lector de Tarjetas Magnéticas (LTM) mediante la entrada de audio del Dispositivo de Pantalla Táctil (DPT).

De acuerdo a las pruebas realizadas se pudo obtener como resultado que por cuestiones de hardware el LTM conectado al DPT tiene problemas en la parte de la captación del audio, tanto el generado por la tarjeta como el del ambiente donde se encuentra el dispositivo. En cambio, en el Dispositivo de Telefonía Celular (DTC) el lector, junto a la librería necesaria para hacerlo funcionar, realiza las lecturas correctas de las tarjetas de crédito.

Se definió realizar una aplicación instalada en el DTC y que por comunicación bluetooth pueda establecer un vínculo con el Terminal de Punto de Venta (TPV) para poder hacer la lectura en el dispositivo donde funciona el lector y poder enviarlo al TPV para su almacenamiento.

El usuario de la aplicación puede pasar la tarjeta por el LTM y todo el audio captado por el micrófono interno que incorpora el lector es procesado por el DTC para que luego el número de la tarjeta de crédito aparezca en pantalla.

En el caso de no poder realizar la lectura, la entrada de datos manual a través del teclado virtual en pantalla es posible.

Una vez que los datos fueron ingresados se procede a realizar la transacción mediante algún gateway de pago como puede ser el caso de authorize.net o braintreepayments.com (estos gateways no fueron probados en el desarrollo de este Trabajo Final de Especialización (TFE)).

5.4.4. Firma digital

Una vez realizado la carga de datos de la tarjeta de crédito, se presenta la pantalla de la figura 5.10 que habilita al cliente poder generar una firma que quedará registrado en el sistema como comprobante de que el cliente efectivamente hizo la compra y que también será impreso en el ticket luego de la venta generada.

5.4.5. Realización de pago mediante la API de MercadoPago

Una de las alternativas de pago que esta aplicación presenta es la de poder utilizar el servicio que ofrece MercadoPago. La figura 5.11 muestra la pantalla de acceso al sistema mercadoPago, implementado con un componente de UI de Android llamado WebView, el cual permite interpretar código HTML para visualizarlo en pantalla teniendo como motor de renderizado al del navegador Google Chrome que el dispositivo contiene.

Para poder realizar pruebas con este medio de pago se generaron usuarios de pruebas, uno para comprador y otro para vendedor. Mediante la Software Development Kit (Kit de Desarrollo de Software) (SDK) de MercadoPago se logra generar

The screenshot displays an Android application interface. At the top, a black status bar shows various icons and the time 00:19. Below it, a dark header bar contains a red Android icon and the title 'Datos Tarjeta de Crédito'. The main content area has a light gray background. It features a title 'Datos Tarjeta de Crédito' followed by four input fields: 'Número de Tarjeta', 'Nombre y Apellido', 'Fecha de Vencimiento', and 'Código de Seguridad'. Below these fields, the text 'Lector: Desconectado' is centered above a gray circle representing a card reader. At the bottom of the form is a large gray button labeled 'Iniciar Transacción'. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Figura 5.9: Activity para ingresar los datos de la tarjeta de crédito.

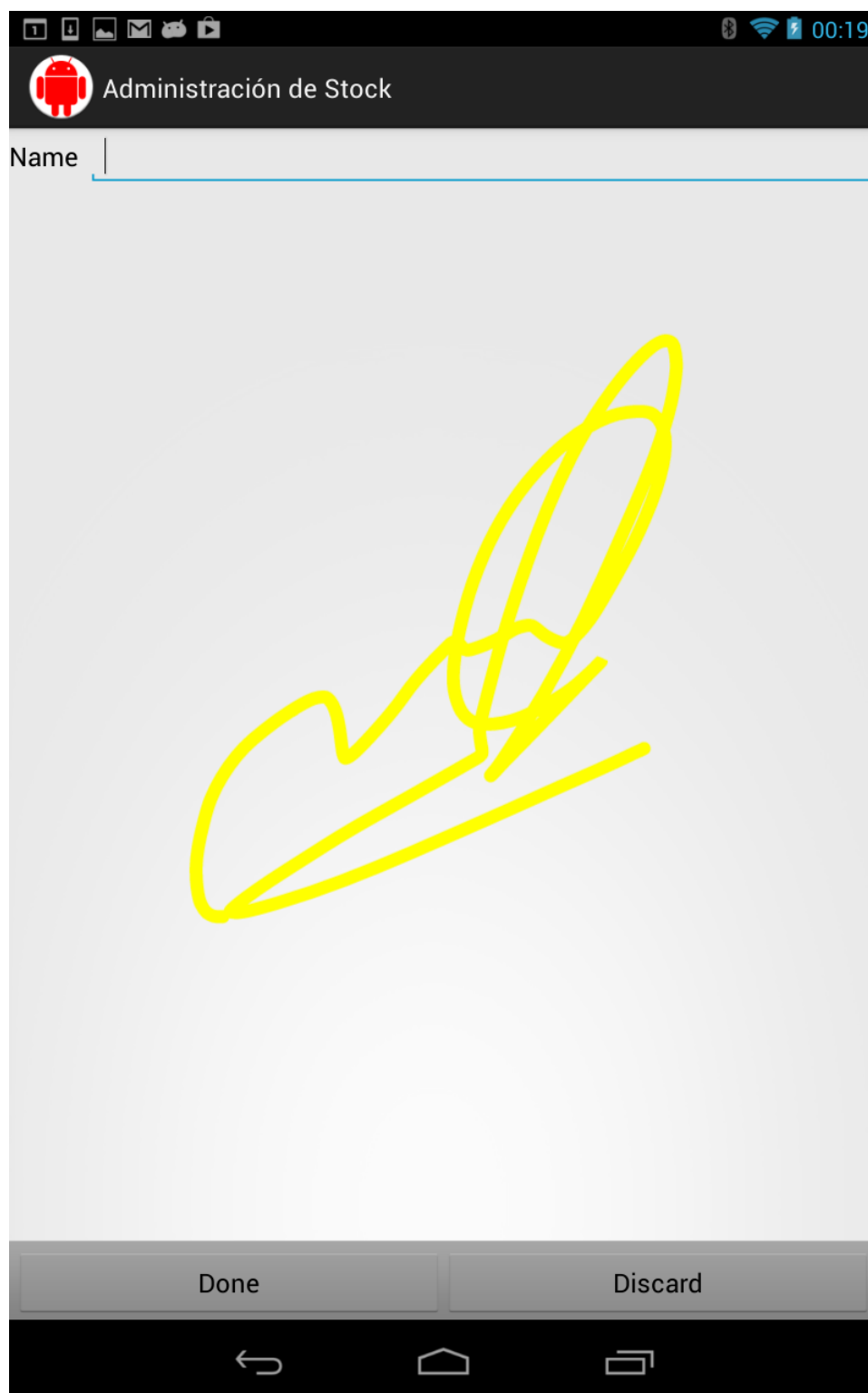


Figura 5.10: Activity para registrar la firma del cliente

un cobro a partir de los datos relacionados a la compra a realizar y es el cual se visualiza en pantalla luego de que el usuario ingreso al sistema, permitiendo así visualizar el monto a pagar y las tarjetas que el sistema tenga registrado de acuerdo a los datos del cliente (datos propiamente internos de MercadoPago).

Al obtener una transacción satisfactoria, el botón “salir” se habilita para poder continuar con la impresión del ticket de compra.

5.4.6. Lectura y escritura de datos mediante dispositivo NFC

Cuando el usuario selecciona la opción de pagar mediante NFC se activa el dispositivo NFC que se encuentra detrás del DPT para tomar la información de una etiqueta o tag (figura 5.12).

Esta etiqueta contiene una memoria que se utiliza para almacenar el saldo del cliente. En primer lugar se realiza la lectura del mismo y se obtiene el saldo correspondiente para luego verificar con el monto a pagar. Si el monto supera al valor del saldo se produce el resultado que muestra la figura 5.13, donde indica que la transacción no puede realizarse y que la compra con esta etiqueta no puede efectuarse.

Si el caso de que el saldo supere al valor de la compra, se informa que la transacción fue aprobada y se procede a almacenar los datos de la venta en la base de datos y a generar el ticket para el cliente.

5.4.7. Generación de ticket

Al finalizar el pago de la compra, la aplicación genera, mediante una librería de creación de archivos con formato PDF, el ticket que será entregado al cliente.

Este archivo PDF se almacena en la memoria interna del DPT y se procede a mostrar la pantalla de la figura 5.14 donde entre las opciones que muestra se puede enviar el ticket por mail utilizando el cliente de mail instalado en el DPT, visualizar el ticket mediante una aplicación de visualización de archivos PDF, o únicamente finalizar la venta y volver al menú principal o a un Activity que permite enviar un mensaje a través de la red Twitter para promocionar el local comercial y de esa forma obtener un descuento en futuras compras, recibir un regalo extra o cualquier otro beneficio para el cliente, generando así una publicidad de tipo viral y en forma electrónica por medio de la red Twitter.

Cuando la opción seleccionada es la de enviar el ticket por mail, se abre el cliente de correo y se visualiza un mail con el cuerpo del mensaje ya redactado y con el ticket en formato PDF como un archivo adjunto al mismo (figura 5.15). Esto permite que solamente se tenga que ingresar la dirección de mail del cliente y presionar el botón para enviar para que de esta forma fácil se pueda lograr el envío del ticket.

Si el caso es la de querer visualizar el ticket para su posterior impresión, el visualizador muestra el ticket generado (figura 5.16) donde se pueden observar



Administración de Stock

Volver Ingresar con cuenta

E-mail o apodo en MercadoLibre

Clave

☒ Recordarme

Ingresar

[¿Olvidaste tu clave?](#)
[Recibir un e-mail para ingresar](#)

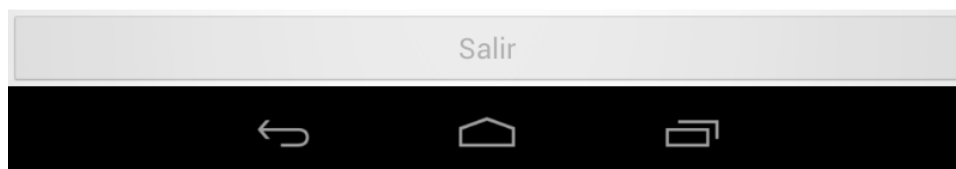


Figura 5.11: Ingreso al sistema de MercadoPago para poder realizar el cobro de la venta

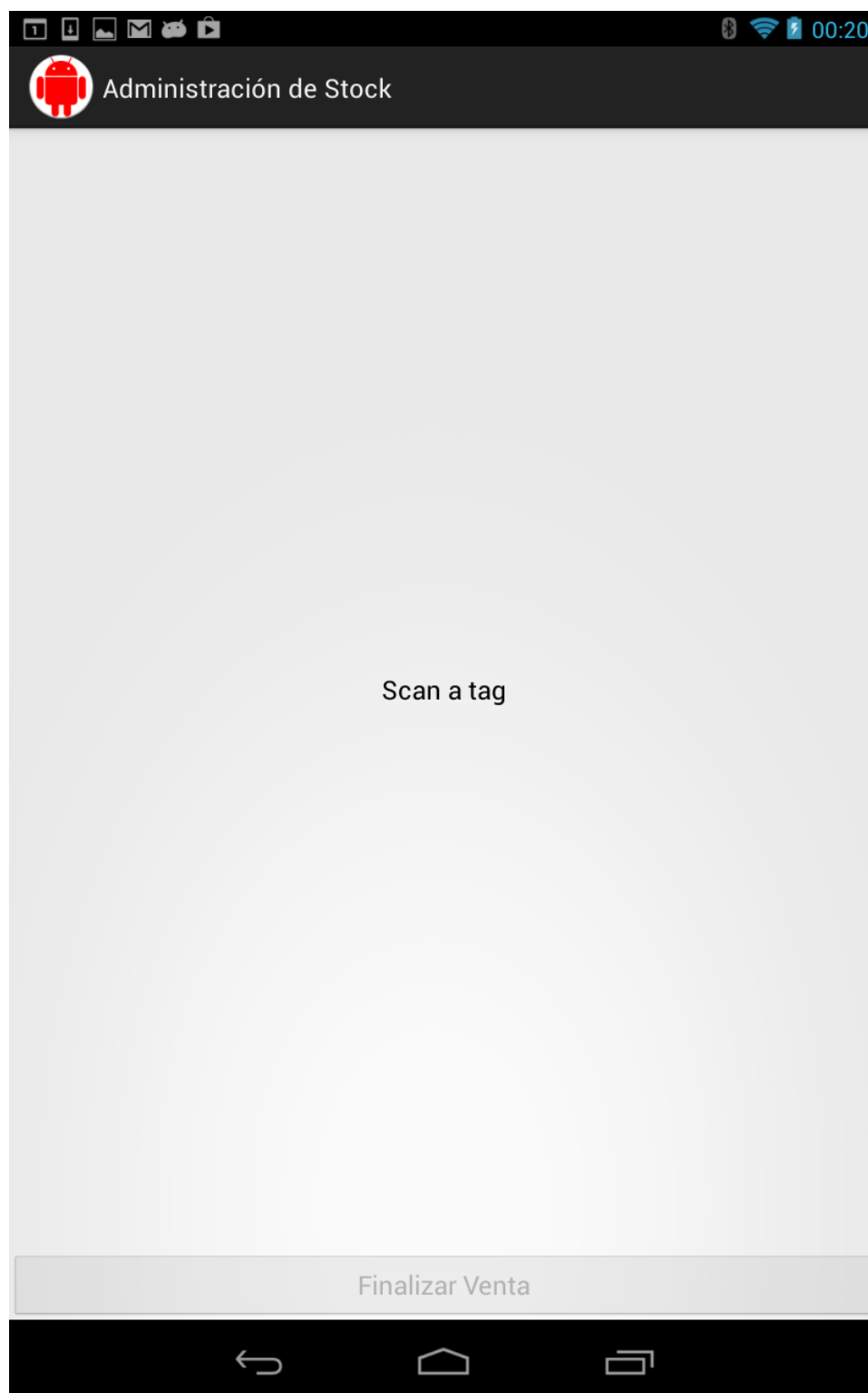


Figura 5.12: Pantalla que indica la activación del dispositivo NFC para realizar la lectura y escritura de datos

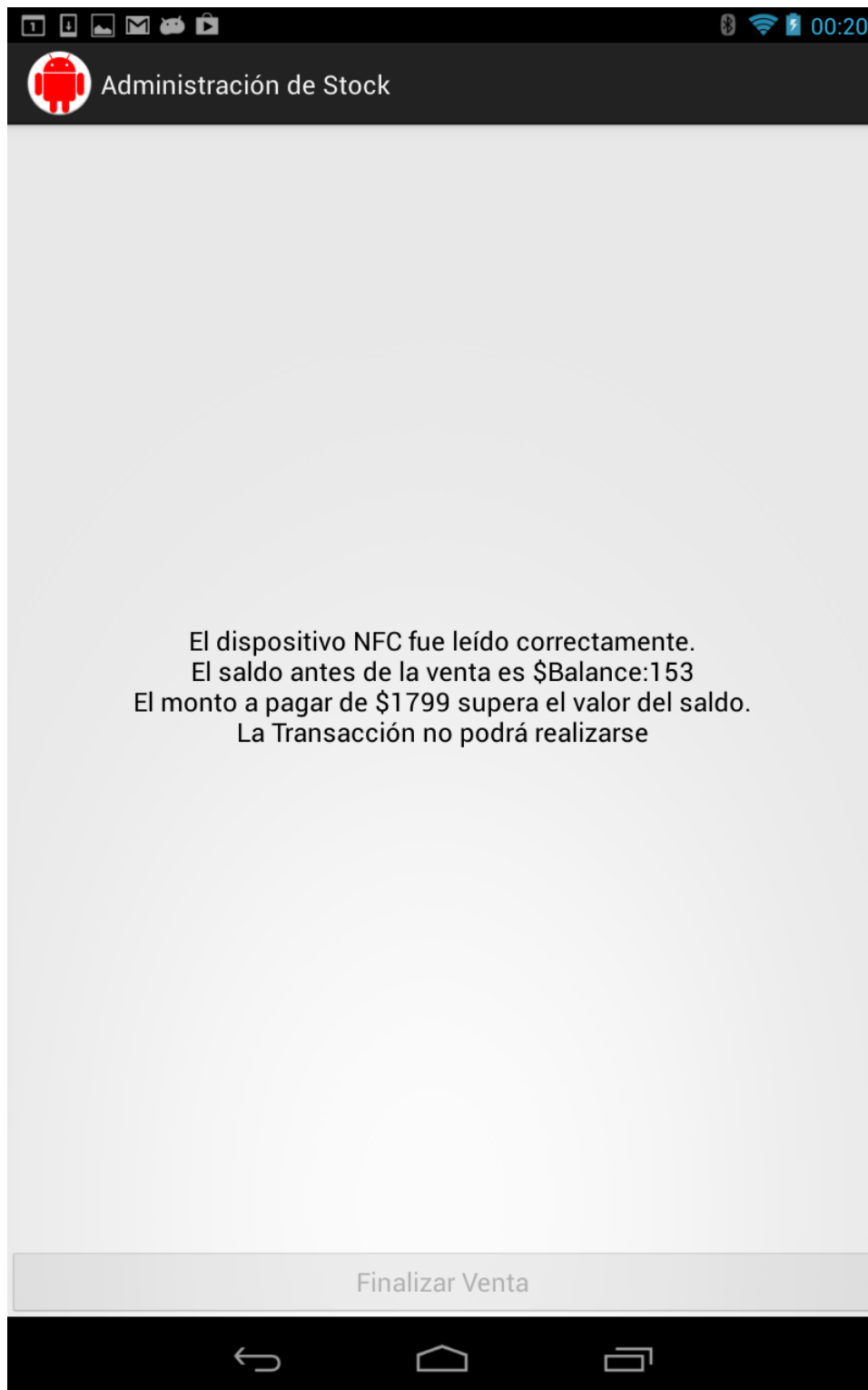


Figura 5.13: Pantalla que surge de una transacción no aprobada

todos los datos de la venta realizada (nombre del cliente, vendedor, detalle de los productos, totales, formas de pago, dirección y nombre del local, etc).

5.4.8. Marketing a través de la red social Twitter

En la figura 5.17 se puede apreciar una conexión establecida a la red social Twitter para el envío de un mensaje publicitario sobre la compra realizada. El comprador debe ingresar a su cuenta de Twitter y permitir el uso de la aplicación del TPV para que se pueda enviar el mensaje a través de la aplicación y lograr que esto, al ser publicado, genere una atracción a futuros clientes.

5.5. Listado de clientes

El TPV permite visualizar un listado de todas las compras que fueron realizadas por un cliente en particular. Al abrir esta opción desde el menú principal, se despliega una lista de todos los clientes registrados donde uno puede verificar su historial al presionar en el nombre del que se desea conocer.

5.6. Opciones

En la figura 5.19 se puede observar las opciones de configuración del sistema. Entre estas podemos encontrar:

- Protocolo: el utilizado para conectar al SIG.
- Localizador uniforme de recursos (URL): la dirección IP del SIG.
- Puerto: al funcionar con un servidor tomcat, se utiliza el puerto por defecto 8080.
- Protocolo: el utilizado para conectar al SIG.
- Dispositivo conectado por Bluetooth: aquí es donde al presionar el botón “Conectar con dispositivo” permite, una vez que la conexión fue exitosa, visualizar el nombre del dispositivo con el cual se conectó.
- Texto a enviar: se permite enviar un texto de prueba hacia el dispositivo con el cual se realizó la conexión. Mediante una aplicación en el otro dispositivo, se puede visualizar el texto y responder al TPV.
- Texto de respuesta: se puede recibir un texto desde el otro dispositivo conectado y visualizarlo aquí para comprobar que la comunicación funcione correctamente.

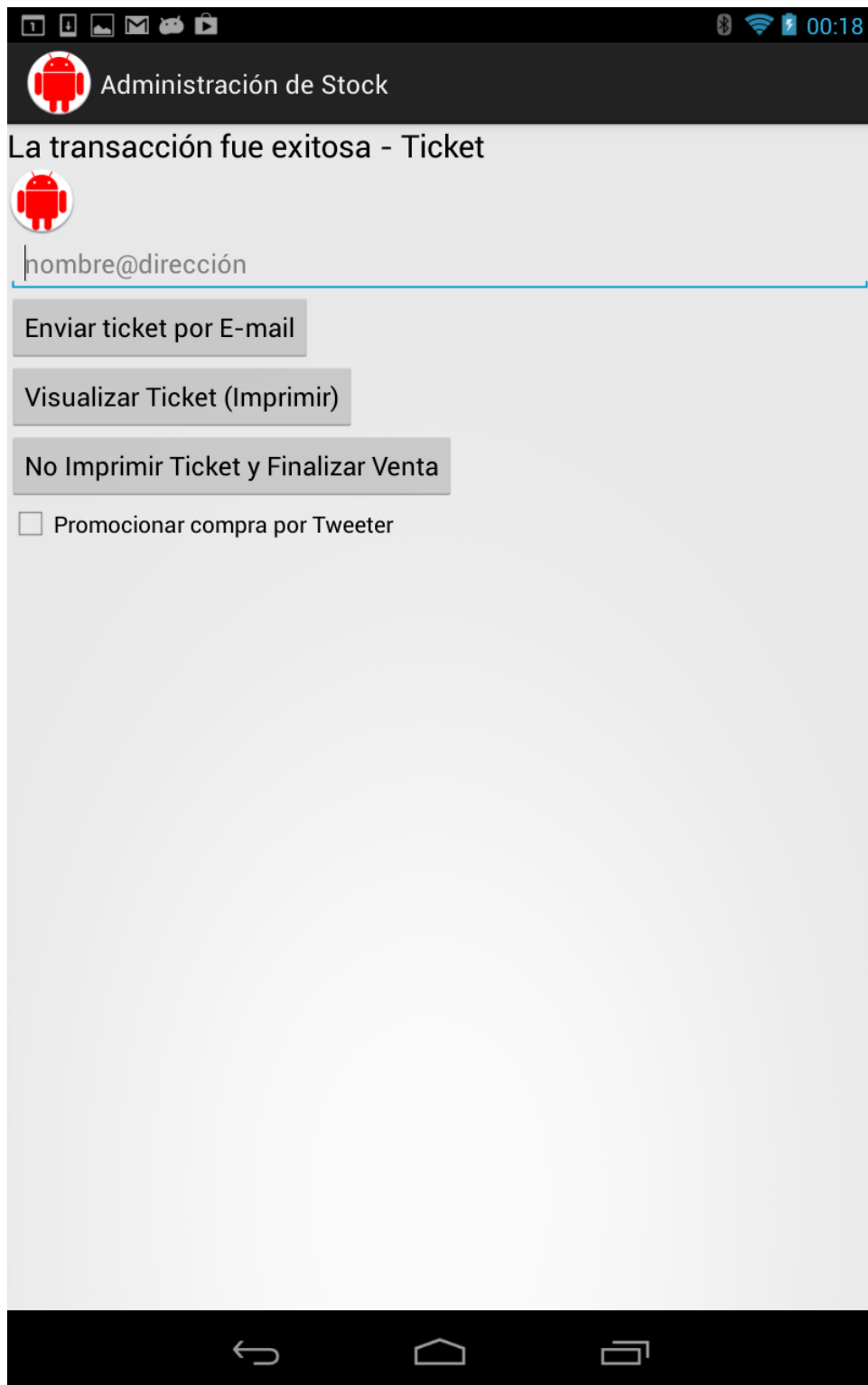


Figura 5.14:

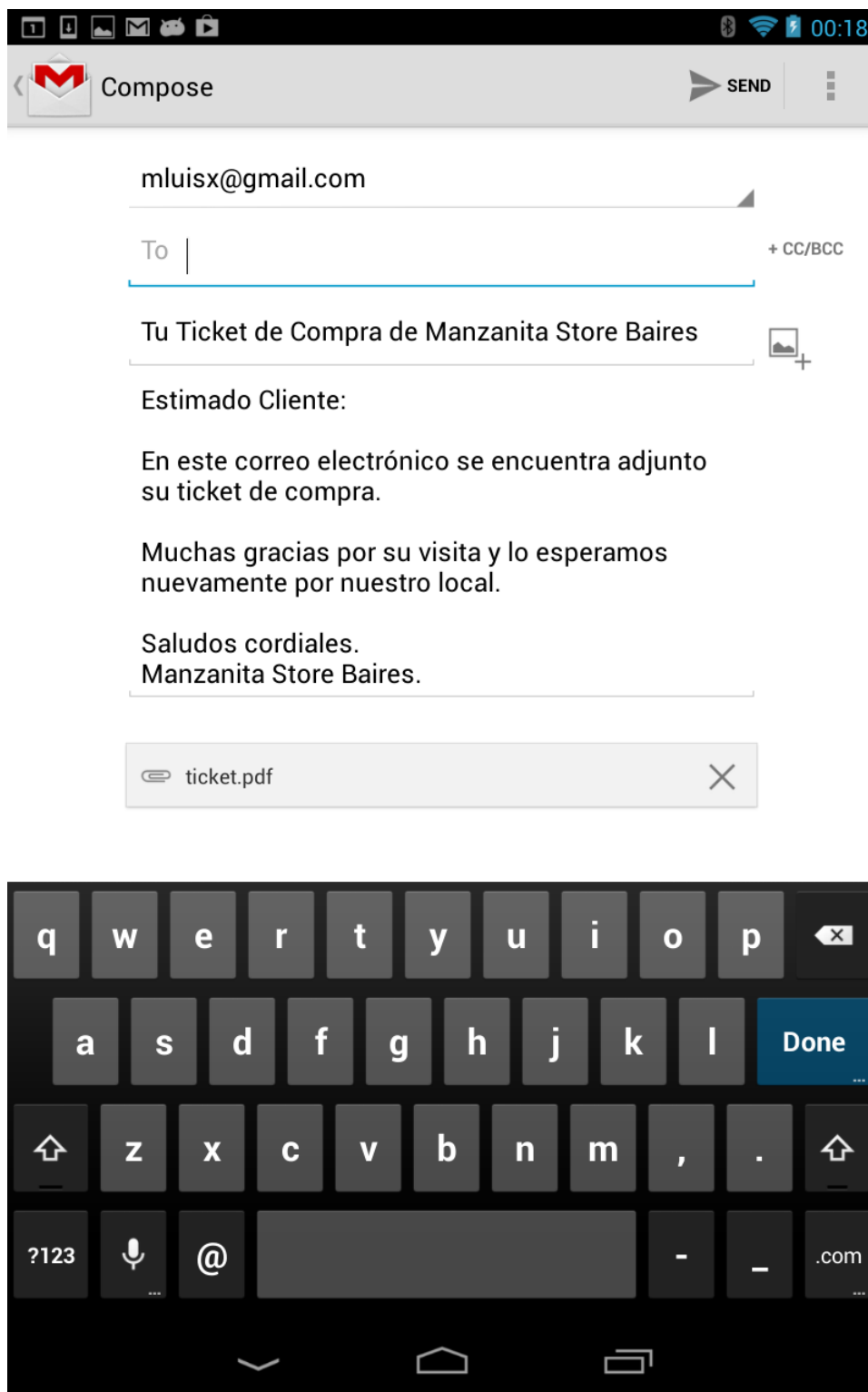


Figura 5.15:

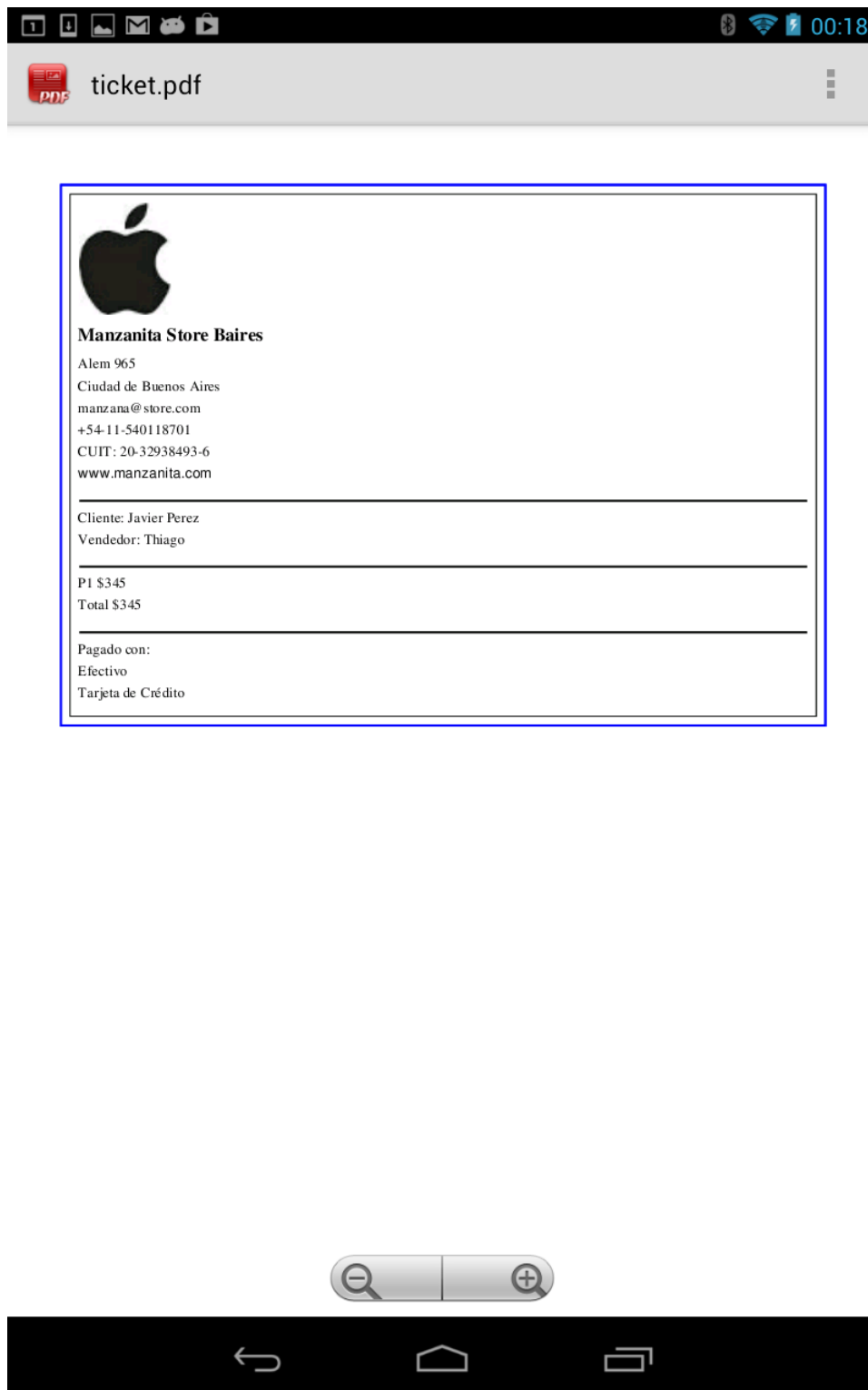


Figura 5.16:

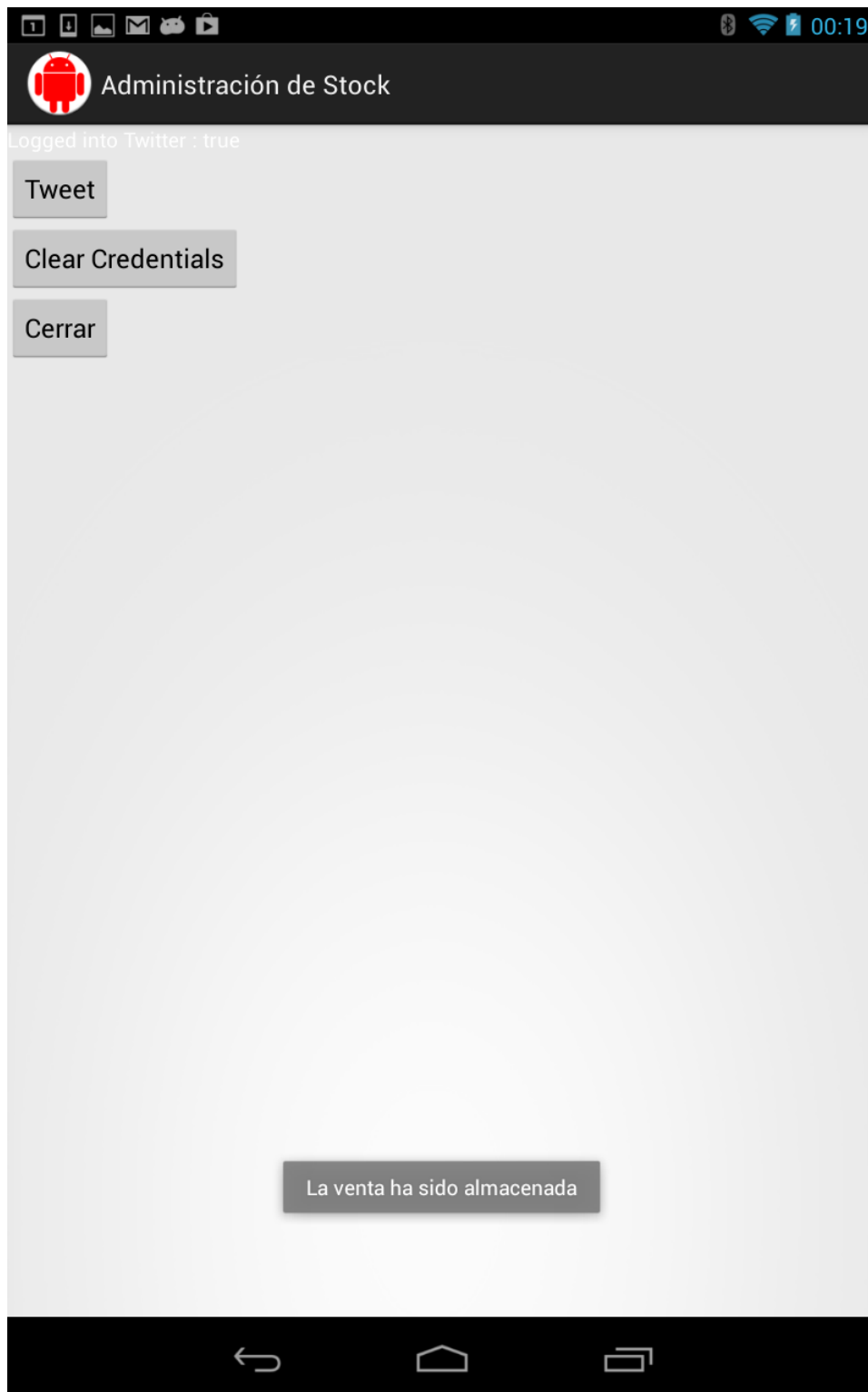


Figura 5.17: La aplicación brinda la posibilidad de enviar mensajes a través de la API de Twitter

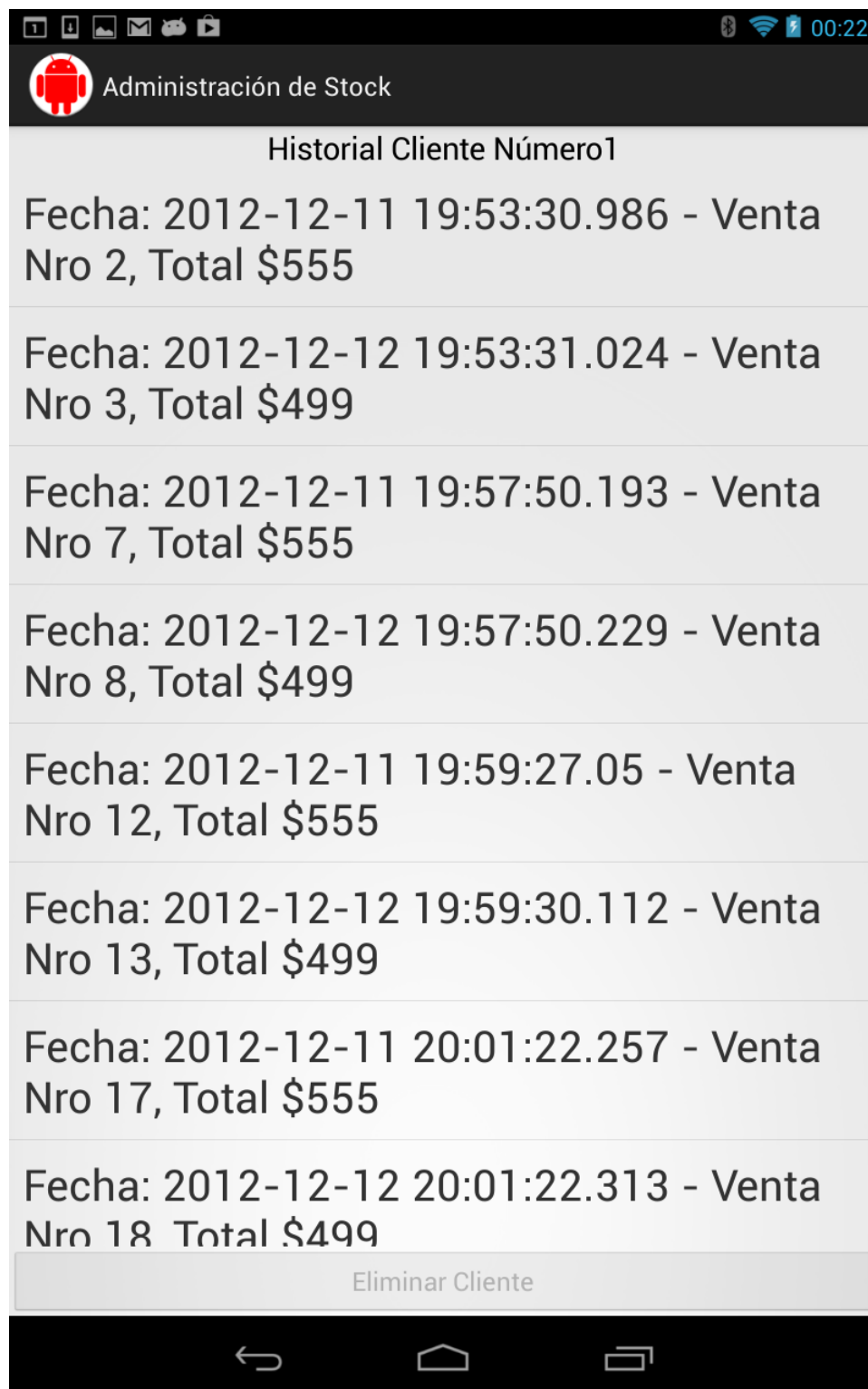


Figura 5.18: Listado de compras de un cliente registrado en el sistema

The screenshot shows an Android application interface with a dark header bar containing the title 'Administración de Stock' and an Android logo. The main content area is titled 'Opciones de Configuración' and contains several form fields and buttons. The fields are labeled 'Protocolo', 'URL del Servidor', 'Puerto', 'Dispositivo Conectado por Bluetooth', 'Texto a Enviar', and 'Texto de Respuesta'. Each field has a placeholder text 'Texto' or a specific value like 'http', '192.168.0.102', and '8080'. Below each field is a button: 'Grabar Cambios' after the port field, 'Conectar con Dispositivo' after the Bluetooth field, 'Enviar Texto' after the 'Texto a Enviar' field, and a large empty text area below the 'Texto de Respuesta' field. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

Administración de Stock

Opciones de Configuración

Protocolo
http

URL del Servidor
192.168.0.102

Puerto
8080

Grabar Cambios

Dispositivo Conectado por Bluetooth
Texto

Conectar con Dispositivo

Texto a Enviar
Texto

Enviar Texto

Texto de Respuesta
Texto

Figura 5.19: Las opciones de configuración

5.7. Administración de stocks

La figura 5.20 describe las distintas características que ofrece la aplicación para poder ingresar un producto a la base de datos del sistema.

La lectura de un código Quick Response (Respuesta Rápida) (QR) puede realizarse a través del botón que permite habilitar la cámara del dispositivo para realizar la lectura del mismo.

Se debe ingresar el dato del precio, la descripción, la cantidad y la categoría del producto. Es posible también asociar a todos los datos del producto una foto del mismo.

Si el producto no contiene impreso un código el cual se tuvo que ingresar a mano en la aplicación, se encuentra disponible la opción de generar el código QR para que luego de que los datos se almacenen en la base, se pueda visualizar el código QR del producto y se pueda realizar una impresión del mismo para colocarlo junto al producto y así poder utilizarlo en futuras lecturas en el TPV.

5.8. Estadísticas

Debido a la necesidad de tener un análisis de como van surgiendo las ventas a medida que transcurre el tiempo, la aplicación brinda la característica de poder generar distintos gráficos (figura 5.21) tomando como fuentes diversos datos que el sistema va almacenando a medida las ventas van surgiendo.

Para poder utilizarse en la toma de decisiones respecto al negocio en el cual esta aplicación funciona, estos gráficos son importantes para poder realizar alguna acción determinada que permita mejoras aún más las ventas y las ganancias.

5.9. Cierre de caja

La última actividad de la figura 5.22 toma los datos de la caja y visualiza sus totales para ser verificados previo al cierre de la caja. Una vez que la caja fue cerrada no se pueden volver a ingresar ventas hasta que una nueva caja sea abierta.

The screenshot shows the 'Administración de Stock' application on an Android device. The interface includes a top status bar with system icons and a time of 00:22. Below the status bar is a dark header with the app's title and an Android logo. The main content area contains several buttons and input fields: 'Leer Código QR', 'Leer Código de Barras', a 'Código' input field, 'Descripción', 'Precio', 'Cantidad', a 'Categoría del Producto' section with a list of categories (Carteras, Cintos, Zapatos, Camperas, Billeteras, Aros, Perfumes) each with a checkbox, a 'Sacar Foto' button, an Android logo, a 'Generar Código QR' checkbox, and an 'Ingresar Datos' button. The bottom of the screen features a standard Android navigation bar with back, home, and recent apps icons.

Administración de Stock

Leer Código QR

Leer Código de Barras

Código

Descripción

Precio

Cantidad

Categoría del Producto

Carteras ☐

Cintos ☐

Zapatos ☐


Camperas ☐

Billeteras ☐

Aros ☐

Perfumes ☐

Sacar Foto



☐ Generar Código QR

Ingresar Datos

Figura 5.20: El activity para la administración del stock del comercio

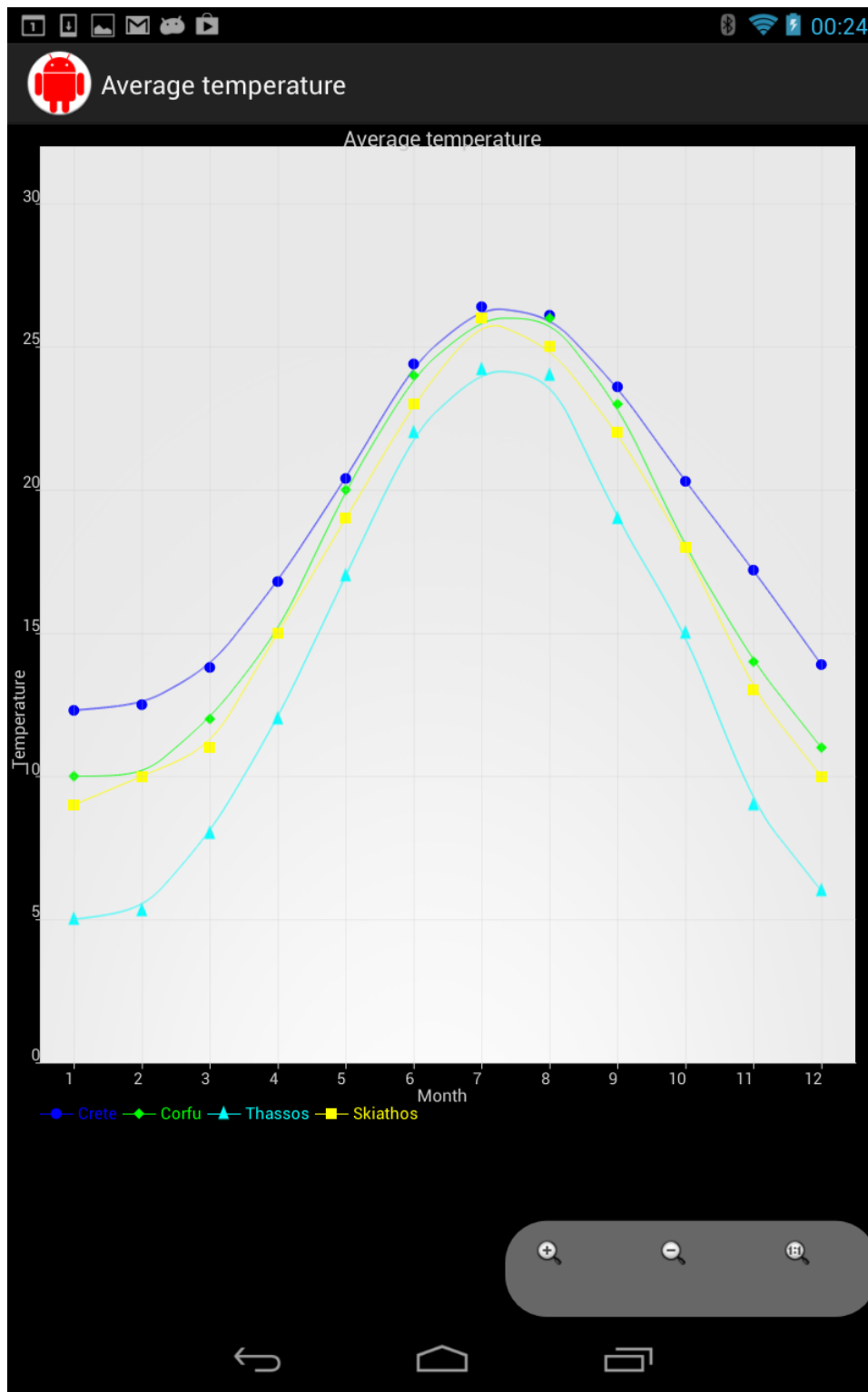


Figura 5.21: Gráfico estadístico generado a partir de los datos almacenados en el sistema

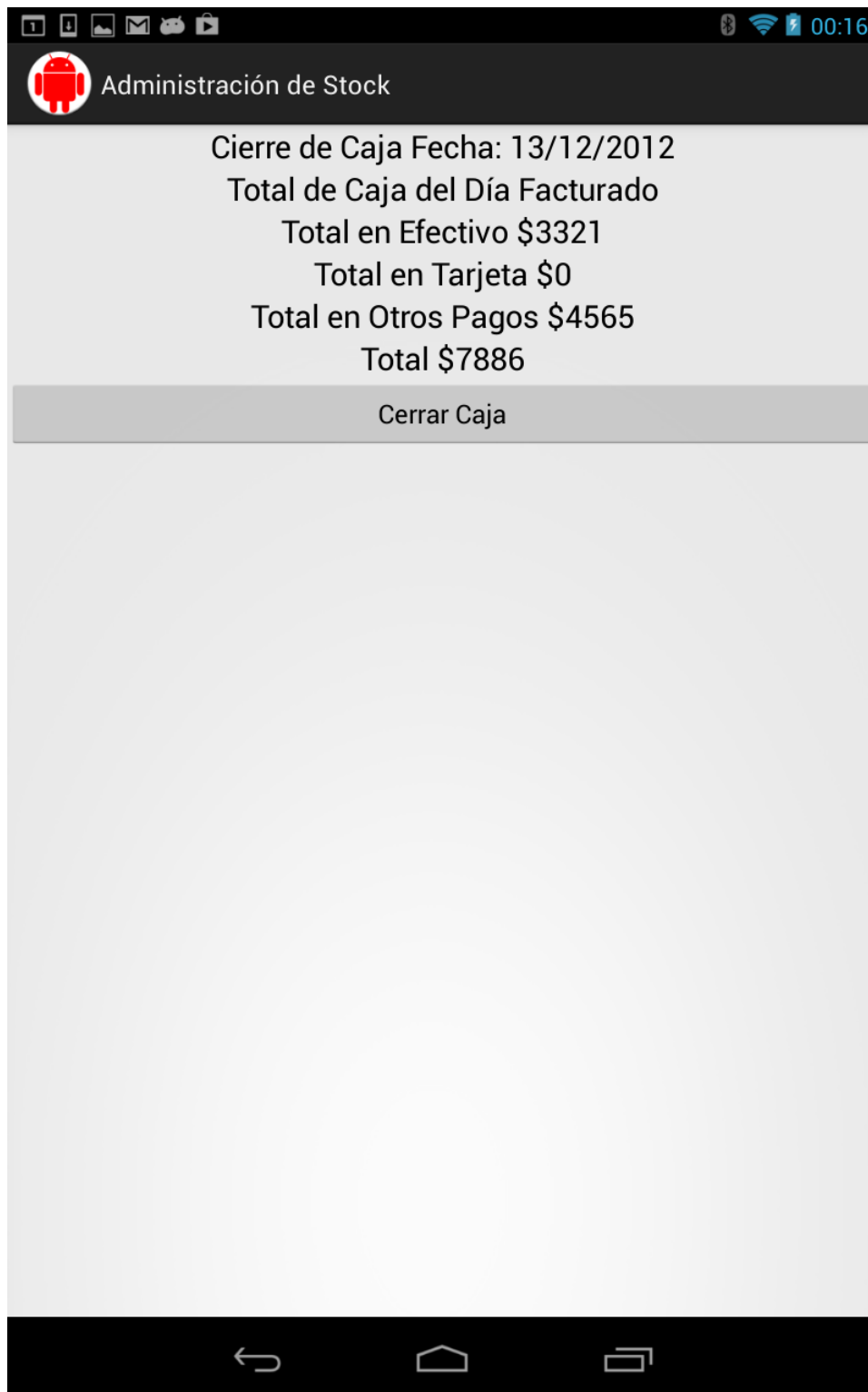


Figura 5.22: Activity para mostrar los distintos totales de la caja abierta previo al cierre

6. Testing Unitario y Funcional de la aplicación

Intro.

6.1. Testing Unitario en Android

Pruebas unitarias desarrolladas para este Trabajo Final de Especialización (TFE).

6.2. Testing Funcional

Aquí puedo escribir sobre robotium.

Conclusión

Se logró desarrollar una aplicación utilizando un sistema operativo móvil y cuya finalidad sea la de demostrar las distintas potencialidades que tienen los dispositivos móviles para reemplazar aplicaciones de computadores actuales.

Se incorporaron distintas librerías para diversos fines lo que enriquece aún más el potencial de esta aplicación y permite expandirlo a distintas necesidades requeridas por los usuarios.

Además del desarrollo, se elaboraron distintas investigaciones y pruebas sobre herramientas, dispositivos y servicios que se pueden utilizar para cumplir la finalidad de este Trabajo Final de Especialización (TFE).

Los objetivos planteados fueron cumplidos satisfactoriamente y la aplicación se encuentra disponible para su uso en cualquier dispositivo que cuente con un sistema operativo Android.

Finalmente, uno de los futuros objetivos posterior a la presentación de este TFE es la de realizar una continua investigación y desarrollo de este sistema en su conjunto para poder producir mejoras en las características que ofrece y posibilitar el uso del mismo a los comercios para que puedan optimizar sus ventas, mejorar el manejo de la información y lograr que los clientes puedan tener una experiencia de compra diferente, innovadora y atractiva.

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	Servidor de Manejo de datos
APIs	Application Programming Interfaces (Interfaces de Programación de Aplicaciones)
LTM	Lector de Tarjetas Magnéticas
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)
URL	Localizador uniforme de recursos
JDK	Java Development Kit
APK	Android Package
API	Application Programming Interface (Interfaz de Programación de Aplicaciones)
BD	Base de Datos
IDE	Entorno Integrado de Desarrollo

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>>