Ingeniería Técnica en Telecomunicaciones especialidad Telemática

SCAN AND GO

Memoria

DANIEL BRAÑAS VILLARDÓN
PONIENTE: PERE BARBERAN

PRIMAVERA AÑO 2012

TecnoCampus
Mataró-Maresme

Dedicatoria

Dedicado a toda mi familia y a todos los amigos que han confiado siempre en mí

Resum

L’objectiu principal del projecte es crear una aplicació per a smartphones que permeti a usuaris d’un establiment, en el aquest cas un supermercat, poder fer la compra i la mateixa vegada la seva pròpia factura de tots els productes mitjançant l’escaneig de codis de barres.

Un altre objectiu seria després de tindre la llista de la compra feta sincronitzar el dispositiu móvil amb el caixer per tal de poder pagar directament l´import final de la compra i així estalviar la demora que provoca el fet de fer cua a l´hora de pagar.

Resumen

El objetivo principal del proyecto es crear una aplicación para smartphones que permita a usuarios de un establecimiento, en este caso un supermercado, poder hacer la compra y al mismo tiempo obtener su propia factura de todos los productos mediante el escaneo de código de barras.

Otro objetivo sería después de tener la lista de la compra hecha sincronizar el dispositivo móvil con el cajero para poder pagar directamente el precio final de la compra en cuestión y así evitar la espera que conlleva el hacer cola a la hora de pagar.

Abstract

The main goal of the project is to create an application for smartphones, which enables the users of a shop, in this case a supermarket, to do their shopping and at the same time receive their own invoice of all the products through the scanning of bar codes.

Another goal would be, once the user has elaborated his/her own shopping list, to synchronize the user's mobile phone with the cashier in order to be able to pay directly the shopping final price in particular and, in that way, avoid any long waitings in line when paying.

Índice

Índice de figuras

1. Introducción

Son muchas las horas las en que nos pasamos a lo largo de nuestras vidas esperando en sitios como bancos,tiendas,conciertos,supermercados,etc. Y este tiempo es un tiempo perdido, un tiempo sin ninguna producción y que malgastamos desafortunadamente porque nos vemos obligados a ello.

Hoy en día la tecnología ha ido teniendo un constante aumento en especial en los dispositivos móviles, los cuales cada día son más y cuentan con más funciones donde sin duda alguna ayudan a acomodarle cada vez más la vida a los usuarios mostrando así su satisfacción adquiriendo los diferentes dispositivos móviles.

La última actualización del Instituto Nacional de Estadística (*www.ine.es)* revelaba que solamente en España, en Enero de 2011 se superaba la cifra de 47 millones de habitantes, y en Diciembre de 2011 el Observatorio Nacional de Telecomunicaciones calculó que en España había más de 55 millones de dispositivos móviles, es decir, que se supera con creces el numero teléfonos móviles al número de personas, un dato que resume muy bien la expansión de las nuevas tecnologías en estos últimos años.

Este gran crecimiento de los teléfonos móviles en la actualidad ha conllevado también a que los teléfonos móviles sean mas sofisticados tanto exteriormente, con pantallas táctiles, diseños ultra planos, etc. como interiormente, con sistemas operativos como Android, iOS,etc. Y es que prácticamente la mitad de los teléfonos vendidos en Europa durante el primer trimestre de 2011 fueron inteligentes, a los que se les debe añadir las tabletas. El número de tabletas comercializadas durante el primer tramo de 2011 ascendió a 1,5 millones.

En el mundo en que vivimos la necesidad para hacer nuestra vida cotidiana lo más ágil y eficiente posible es básico para contentar a los ciudadanos, por ello, se ha optado este proyecto.

Con esta idea se pretende mejorar el funcionamiento de muchos ciudadanos en sus vidas cotidianas a la hora de evitar la demora en los supermercados.

1.1 Problemática de los supermercados

Con este proyecto se intenta buscar una manera más eficiente de hacer la compra y evitar estas incómodas situaciones que perjudican tanto a los usuarios, como a los trabajadores del establecimiento.

Algunas de estas incómodas situaciones se resumen a continuación :

- Las cajeras en más de una ocasión entablan conversación con los clientes, y el cliente por no rechazar la conversación al final acaba entrando al trapo y a lo mejor sin darse cuenta acaba perdiendo un buen tiempo tanto para él como para los demás clientes que están esperando en esa caja.

- Este es uno de los problemas más comunes, escoger una caja que parece que haya menos clientes con lo cual a priori se acaba uno por acceder a esta cola , y al final resulta que la cola que parecía más corta y rápida se acaba convirtiendo en lenta, muy lenta, esto puede deberse a la falta de algún código de barras de un producto, o que la máquina registradora haya sufrido una avería, o porque la cajera es la más lenta.

- Hay gente que una vez está en la caja , o bien ya pagando o bien a punto de pagar, se acuerda de comprar algún producto de última hora porque se les olvidó en ese momento, y la gente que está en la cola esperando tienen que aguantarse de todo este tiempo demorado ya que la cajera se queda impasible y se lo permite.

- A cualquiera le puede pasar que a la hora de pagar no te alcance el dinero, pero si algo similar nos ocurre, cuando el cliente está pagando debería solucionarlo de la mejor manera posible, y no optar por quedarse pensando durante 5 minutos de qué producto se va uno a deshacer y no entrar a debatir con tu acompañante sobre la conveniencia de llevar ese producto o aquel. En esas circunstancias la solución es simple y no requiere de mayor esfuerzo, dejar lo menos importante y se acabó el problema.

- En las cajas rápidas suelen poner debajo el número máximo de artículos, normalmente suelen ser 5 o 10 artículos, dependiendo del establecimiento, pues bien el problema reside en que hay consumidores que no tienen en cuenta esto, o bien por equivocación , o bien por voluntariedad que es el problema más frecuente, para evitar esperarse en

otras cajas normales que están llenas en ese momento, y aquí el cajero/a de turno le vuelve a permitir en su mayoría de veces este hecho.

- Hay veces en las que el cajero o cajera, a la hora de hacer la devolución del dinero del cliente a la hora de cobrarle, intenta deshacerse en ese momento de todo el dinero en metálico que considera como "chatarra" , y a veces una compra de poco dinero te puede salir muy pesada ya que te dan una buena cantidad de dinero en metálico en monedas de 5 céntimos , 2 céntimos y hasta incluso de 1 céntimo por poner un ejemplo, esto evidentemente indigna bastante al cliente y también provoca la demora en el pago final.

1.2 Destino del proyecto

Va dirigido a los consumidores principalmente de supermercados e hipermercados, cualquier persona que quiera que su vida cotidiana sea eficiente en todos los aspectos, y en especial, en establecimientos grandes donde se acumula mucha gente a la hora de comprar , como pueden los supermercados.

La gente está muy ligada a las nuevas tecnologías , prueba de ello , es que difícilmente se encuentra en el mundo en que vivimos a alguien sin que tenga un móvil o un ordenador, por eso este pretende abarcar a todas las personas, sin importar la edad en cuestión, ya que se ha tenido en cuenta , la sencillez y la manejabilidad como factores principales.

Este proyecto pretende ayudar tanto a los usuarios como a los trabajadores del establecimiento, ya que , se evitaría la saturación en las cajas y esto conllevaría a un cansancio mínimo de los trabajadores que están en esos puestos de trabajo.

La comodidad es un aspecto muy importante cuando hablamos de cualquier proyecto de tipo tecnológico, una de las principales ventajas sin duda alguna.

1.3 Decisión del proyecto

La primera motivación para la realización de este proyecto fue adentrarse en el mundo de la telefonía móvil, en el mundo Android concretamente, como Ingeniero de Telecomunicaciones ,las nuevas tecnologías son clave en esta carrera y el hecho de poder programar y crear una aplicación por uno mismo, para un buen uso en el presente y un mejor uso en el futuro, supuso una buena motivación el conseguir este reto.

Este proyecto está diseñado para poder agilizar y facilitar más la vida de los consumidores, el sistema de caja rápida que tiene lugar en los grandes supermercados podrá ser sustituido por este sistema en cada una de las cajas, es decir, las cajas convencionales de los supermercados con este proyecto pasarán a ser cajas rápidas de por sí.

1.4. Objetivo del proyecto

El objetivo es desarrollar una aplicación basada en el sistema operativo Android, que permita a cualquier consumidor de un supermercado realizar la lista de la compra mediante su dispositivo móvil, primero registrando los productos que se añaden a la cesta de la compra, mediante el código de barras de estos, para posteriormente terminada la cesta, poder proceder al pago de los productos, mediante el uso de una factura, llevada a cabo gracias a volcado de información del dispositivo móvil al cajero que actuará como servidor.

1.5 Estructura de la memoria

Se puede dividir el proyecto en dos fases, la primera el dispositivo móvil interactúa con el producto mediante la lectura de código de barras, éste código se capta con el celular y esa referencia la enviamos al servidor con una petición que lo que pretende es conocer si dicho producto está en nuestra base de datos para posteriormente agregarlo a nuestra lista de la compra mediante el dispositivo móvil.

La segunda parte está basada en la interconexión de nuestro dispositivo móvil con el cajero del supermercado o en nuestro caso, nuestro ordenador, donde una vez se obtiene la lista de la compra mediante la aplicación móvil, se puede proceder al volcado de toda la información que hemos capturado para posteriormente realizar el pago final.

2. Trabajos relacionados

2.1 Dispositivos semejantes

Antes de empezar a desarrollar el proyecto se ha llevado a término un estudio de los sistemas que utilizan éste sistema o similares.

Tecnología RFID

La identificación por radiofrecuencia es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio.

Ejemplos prácticos de esta tecnología los tendríamos en los chips que están integrados en el Teletac, los chips que están integrados en los libros de una biblioteca también, y los que están integrados en la ropa.

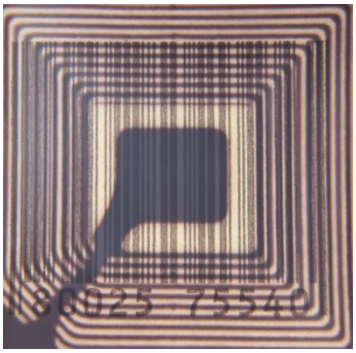


Figura 2.1 Chip RFID

Carrefour

Al entrar en el supermercado, utilizando la tarjeta de socio del Club Carrefour, el cliente recibe un lector de códigos de barras, con una pequeña pantalla, que utilizará durante el proceso de compra para escanear los artículos que compra. El aparato muestra el importe de cada artículo y el total de tu compra, pudiendo controlar lo que compras y lo que gastas.

Al llegar a las cajas especiales de Scan & Go, el sistema informará de si su compra ha sido seleccionada de manera aleatoria para una revisión de un determinado número de artículos, o de si se posee luz verde para realizar el pago. Posteriormente sólo queda realizar el pago con la tarjeta, sin sacar un sólo artículo de su carrito de la compra.

En Francia, el grupo Carrefour cuenta con el sistema de pago “Scan & Go” en el 25% de las cajas de salida de sus centros, en España tenemos este sistema ubicaciones como por ejemplo en Madrid, El Pinar de las Rozas y Alcobendas y en Barcelona, Cabrera de Mar.

Figura 2.2 Scan&Go Carrefour

2.2 Sistemas operativos

2.2.1 Tipos de sistemas operativos

En los siguientes puntos se analizan los diversos sistemas operativos que incorporan los Smartphone actuales y un lenguaje de programación más adecuado para el desarrollo de una aplicación que se ubicara en el Smartphone.

Android

Este sistema operativo es el empleado en el proyecto por varias razones, la primera y la más importante es porque hay más aplicaciones gratuitas en el market(llamado ahora Google Play) que en el sistema IOS empleado por Iphone, esto hace más accesible a los usuarios para descargar aplicaciones de todo tipo, un claro los tenemos en dos aplicaciones muy famosas que están disponibles para los dos sistemas, como son Angry Birds y Whatsapp, de pago las dos en IOS mientras que Android totalmente gratuitas y en segundo lugar, el proceso de multitarea de IOS no está al mismo nivel que Android.

Android es un sistema operativo desarrollado por la Open Handset Alliance, liderada por Google, es un consorcio de 78 compañías de hardware, software y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Este sistema operativo está basado en una versión modificada del Kernel de Linux.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. A la fecha, se han sobrepasado las 400.000 aplicaciones .

Figura 2.3 Logotipo Android

iOS

Es un sistema operativo móvil de Apple. Originalmente desarrollado para el iPhone, siendo después usado en dispositivos como el iPod Touch, iPad y el Apple TV**.**

iOS se deriva de Mac OS X, que a su vez está basado en Darwin BSD, y por lo tanto es un sistema operativo Unix.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz. iOS tiene 4 capas de abstracción: la capa del núcleo del sistema operativo, la capa de Servicios Principales, la capa de Medios de comunicación y la capa de Cocoa Touch. El sistema operativo ocupa bastante menos de medio gigabyte del total del dispositivo, de 8 GB o de 16 GB. Esto se realizó para poder soportar futuras aplicaciones de Apple.

Figura 2.4 Logotipo iOS

Whindows Phone

Fue desarrollado por Microsoft, y diseñado para su uso en teléfonos inteligentes (Smartphones) y otros dispositivos móviles. Windows Phone hace parte de los sistemas operativos con interfaz natural de usuario.

Se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows. Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente.

Anteriormente llamado Windows Mobile es un sistema operativo móvil compacto desarrollado por Microsoft, y diseñado para su uso en teléfonos inteligentes (Smartphone) y otros dispositivos móviles. Se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows. Está diseñado para ser similar a las versiones de escritorio de Windows estéticamente.

Windows Mobile ha evolucionado y cambiado de nombre varias veces durante su desarrollo, siendo la última versión la llamada Windows Phone 7, anunciada el 15 de febrero del 2010 y sujeta a disponibilidad a finales de 2010.

Figura 2.5 Logotipo Windows Phone

Blackberry

Fue desarrollado por Research In Motion para sus dispositivos BlackBerry. El sistema permite multitarea y tiene soporte para diferentes métodos de entrada adoptados por RIM para su uso en computadoras de mano, particularmente la trackwheel, trackball, touchpad y pantallas táctiles.

El sistema operativo proporciona soporte para Java MIDP 1.0 y WAP 1.2. Las versiones anteriores permitían la sincronización inalámbrica con Microsoft Exchange Server para el correo electrónico y calendario, al igual que Lotus Domino e-mail. El actual OS 5.0 proporciona un subconjunto de MIDP 2.0, y permite la activación inalámbrica completa y la sincronización con Exchange de correo electrónico, calendario, tareas, notas y contactos, y añade un soporte para Novell GroupWise y Lotus Notes.

Figura 2.6 Logotipo Blackberry

Symbian

Fue desarrollado gracias a la unión de varias empresas de telefonía móvil, entre las que se encuentran Nokia, Sony Ericsson, Psion, Samsung, Siemens, Arima, Benq, Fujitsu, Lenovo, LG, Motorola, Mitsubishi Electric, Panasonic, Sharp, etc.

Sus orígenes provienen de su antepasado EPOC32, utilizado en PDA's y Handhelds de PSION. El objetivo de Symbian fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o el Windows Mobile de Microsoft y ahora Android de Google Inc., iOS de Apple Inc. y Blackberry 6 RIM.

Figura 2.7 Logotipo Symbian

2.3 Códigos de barras

El código de barras es un código basado en la representación mediante un conjunto de líneas paralelas verticales de distinto grosor y espaciado que en su conjunto contienen una determinada información. De este modo, el código de barras permite reconocer rápidamente un artículo en un punto de la cadena logística y así poder realizar inventario o consultar sus características asociadas. Actualmente, el código de barras está implantado masivamente de forma global.

2.3.1 Definición de los códigos de barras

Es un sistema que permite la identificación de las unidades comerciales y logísticas de forma única, global y no ambigua. Este conjunto de barras y espacios codifican pequeñas cadenas de caracteres en los símbolos impresos.

La correspondencia o mapeo entre la información y el código que la representa se denomina simbología. Estas simbologías pueden ser clasificadas en dos grupos atendiendo a dos criterios diferentes:

Continua o discreta: los caracteres en las simbologías continuas comienzan con un espacio y en el siguiente comienzan con una barra (o viceversa). Sin embargo, en los caracteres en las simbologías discretas, éstos comienzan y terminan con barras y el espacio entre caracteres es ignorado, ya que no es lo suficientemente ancho.

Bidimensional o multidimensional: las barras en las simbologías bidimensionales pueden ser anchas o estrechas. Sin embargo, las barras en las simbologías multidimensionales son múltiplos de una anchura determinada (X). De esta forma, se emplean barras con anchura X, 2X, 3X, y 4X.

Nomenclatura básica

Módulo: Es la unidad mínima o básica de un código. Las barras y espacios están formados por un conjunto de módulos.

Barra: El elemento oscuro dentro del código. Se hace corresponder con el valor binario 1.

Espacio: El elemento claro dentro del código. Se hace corresponder con el valor binario 0.

Carácter: Formado por barras y espacios. Normalmente se corresponde con un carácter alfanumérico.

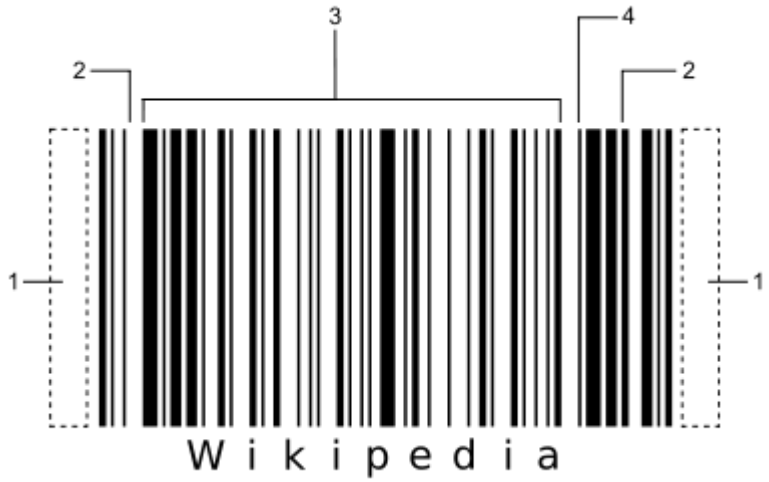


Figura 2.8 Composición código de barras

Funciones técnicas de los caracteres contenidos en un código de barras:

1-Quiet zone

2-Carácter inicio (derecha), carácter terminación (izquierda).

3-Carácter de datos

4-Checksum

2.3.2 Características de los códigos de barras

Un símbolo de código de barras puede tener, a su vez, varias características, entre las cuales podemos nombrar:

Densidad:

Es la anchura del elemento (barra o espacio) más angosto dentro del símbolo de código de barras. Está dado en mils (milésimas de pulgada). Un código de barras no se mide por su longitud física sino por su densidad.

WNR: (Wide to Narrow Ratio)

Es la razón del grosor del elemento más angosto contra el más ancho. Usualmente es 1:3 o 1:2.

Quiet Zone:

Es el área blanca al principio y al final de un símbolo de código de barras. Esta área es necesaria para una lectura conveniente del símbolo.

Figura 2.9 Áreas de un código de barras

2.3.3 Aplicaciones de los códigos de barras

El código de barras ha sido creado para identificar objetos y facilitar el ingreso de información, eliminando la posibilidad de error en la captura. En la actualidad la Tecnología de Código de Barras es utilizada en muchas áreas ya que ha probado ser adaptable y exitosa para los propósitos de una identificación automática de productos. El campo de acción que abarca va desde la recepción de los materiales, su procesamiento, hasta su despacho final.

El código de barras como sistema de codificación tiene aceptación mundial, y hoy en día es un requisito indispensable que sus productos puedan ser comercializados tanto en el mercado interno como en el mercosur. Pero su uso no sólo es aplicable al comercio de productos, sino también se puede emplear para uso interno de su empresa, para llevar un control exacto de su stock, toma de inventarios y operaciones de carga y descarga de mercadería, agilizar las ventas, y en todo aquello que involucre recolección y manipulación de datos.

La aceptación del código de barras es masiva, y hoy lo encontramos en todos lados, supermercados, farmacias, perfumerías, videos, depósitos, fábricas en general, etc. Y también los vemos en las boletas de servicios públicos (agua, luz, gas, teléfono).

Ésta aceptación por éste sistema de codificación se debe a la confiabilidad para la recolección automática de datos, eliminando errores humanos producidos por mal ingreso de datos, lo que redunda en un aumento de productividad, debido a la eficiencia, exactitud y rapidez del mismo, lo que trae como consecuencia inmediata una mejor atención al cliente y un aumento en sus ganancias por reducción de costos.

Las aplicaciones son amplias y variadas y crecen día a día.

EJEMPLOS DE APLICACIÓNES COMERCIALES

- Administración de Materias Primas

Para que la administración de las materias primas de una empres a posea información ágil y confiable el sistema EAN UCC es la herramienta confiable que se necesita, ya que posee las fortalezas ideales para un adecuado seguimiento y control de insumos. La identificación estándar que ofrece este sistema le permite a su empresa y a sus proveedores grandes beneficios, gracias a la información común que aporta valiosos datos para una labor conjunta.

Evitar los desperdicios, reciclar y lograr ahorros de materias primas son algunos de los pasos clave para optimizar el aprovechamiento de los insumos. Sin embargo, para lograrlo, se debe conocer a fondo el comportamiento de las materias primas, y realizar un seguimiento detallado de su utilización, lo cual exige un manejo de la información exacto y ágil, que se logra con los códigos de barra.

- Administración de Bodegas y Centros de Distribución

Administrar un centro de distribución es una tarea compleja que va más allá de la sola manipulación y control de la mercancía, el factor que realmente marca la diferencia es la administración de la información de la misma, pues es la que realmente permite desarrollar procesos confiables y eficientes dentro de éste eslabón de la cadena de abastecimiento.

La filosofía del sistema EAN UCC es mejorar la cadena de abastecimiento mediante la optimización de los procesos. En la bodega esta filosofía se aplica en cada proceso: almacenamiento, preparación, despacho.

Cada uno puede ser mejorado gracias a la utilización de las herramientas de éste sistema. El sistema está diseñado bajo una filosofía que permite mejorar ese control. La identificación estándar de la mercancía, la captura de información de la misma y la comunicación de su movimiento, hacen que los sistemas de información puedan responder mejor a las necesidades de los usuarios, ya que los datos que manejan son más confiables. Los tiempos muertos debido a las demoras en la captura de información, pueden evitarse usando la tecnología disponible.

- Producción

Controlar el producto que llegará a los consumidores es fundamental, sin embargo, éste control requiere de tres factores que deben encajar perfectamente para generar un información oportuna y real: personas, tiempos y recursos.

El código de Barras y EDI bajo los estándares EAN-UCC, facilitan las labores de captura y comunicación de información de manera eficiente, posibilitando a las personas hacer uso de ella sin el desgaste del proceso de ingreso de datos.

Implementar el código de barras y Edi en una compañía no solo redundará en un mejor aprovechamiento de los recursos en eficiencia, y agilidad en las diferentes etapas de los procesos de manufactura d los productos, sino que moderniza el servicio que se les presta a los clientes, proveedores y miembros de la compañía.

- Administración Eficiente de Puntos de Venta

Los beneficios más importantes para que una empresa administre eficientemente los puntos de venta y se desarrolle competitivamente son:

- Optimización en el control de inventarios y aumento de productividad en el punto de pago, eliminando colas y disminuyendo el tiempo de espera. Mejor servicio al cliente.

- Disminución de los procesos de marcación de precios, eliminación de errores de digitación y captura de datos de venta en forma rápida y segura.

- Identificación de las principales áreas de mermas.

- Obtención de información confiable para el manejo dl negocio.

- Establecimiento de un lenguaje común con sus proveedores a través del código de barras, incrementando la productividad de la relación comercial lo que facilita la implementación de otras tecnologías con el intercambio electrónico de documentos -EDI-

- Conocimiento del comportamiento de los productos en el mercado.

- Aumento de la eficiencia en el manejo de procesos como el recibo, despacho, y selección de mercancías.

- Aplicaciones de Comercio Exterior

En comercio exterior, el uso del sistema EAN-UCC puede ser muy variado, teniendo en cuenta que son muchas las entidades y los procesos que en esta actividad se encuentran involucrados.

Hoy las soluciones se están dando. La identificación única de la carga y la captura automática de la misma es totalmente factible. Igualmente la posibilidad de intercambiar cualquier tipo de información vía electrónica y que la misma sea procesada automáticamente, es hoy una realidad.

- Identificación de carga

La identificación única de contenedores por medio del código de barras, permite capturar la información con una simple lectura automática, facilitando las operaciones de recepción, almacenamiento y despacho de los mismos. Sin importar que se encuentren llenos o vacíos, que sean refrigerados o no lo sean.

Esta tecnología evita errores en la transcripción de información y asegura que cualquier movimiento quede registrado en el sistema de información. De esta forma se puede cubrir una de las grandes necesidades de los agentes de carga y las navieras: saber en todo momento dónde se encuentra un contenedor y cuál es su disponibilidad.

- Identificación de Personal

Es importante identificar de forma única las personas, bien sea de una empresa, un club, una biblioteca etc, para controlar las actividades y operaciones que realizan; por ejemplo con control de activos fijos, control de acceso a áreas restringidas y manejo automático de nómina.

Es de gran utilidad que dicha identificación sea estándar, a través del Código de Barras simbolizado e impreso en el carné de identificación, es posible leer automáticamente la información de cada persona al realizar operaciones internas que requieran algún tipo de control.

Tome nota de los beneficios que alcanzaría con el aprovechamiento del uso del código de barras en su empresa:

Eliminación procesos de digitación en cualquier proceso que implique registrar la identificación de una persona.

Agilización en la prestación de cualquier servicio.

Seguridad en el registro de información.

Posibilidad de mantener bases centrales de datos de las personas, que sirvan a diversas entidades y eviten el fraude.

Agilización de procesos administrativos.

Control de acceso a áreas restringidas.

2.3.4 Lectura de los códigos de barras

Es también conocido en Europa como sistema EAN -European Article Numbering o Numeración Europea de Artículos- es un método de codificación que permite identificar casi de inmediato todo tipo de productos mediante un lector especial conectado a una caja registradora informatizada.

Las ventajas de que ofrece este sistema son varias: Por un lado le permite a los fabricantes, distribuidores y detallistas mantener un control pormenorizado de los movimientos de sus mercancías, y por otro lado evitar errores de cobro e inútiles esperas del cliente ante la caja, proporcionándole además un detallado listado de sus compras.

El código EAN (existen muchos otros, pero ahora solo hablaremos del EAN 13) consta de trece números sobre los cuales figura su correspondiente transcripción en forma de barras. Los dos primeros dígitos representan la asociación que asigna los códigos a las empresas fabricantes y distribuidoras. La Asociación Española de Codificación Comercial (AECOC) tiene atribuido el número 84, por lo que los códigos de todos los artículos producidos por empresas españolas empiezan por esta cifra.
Las cinco posiciones que siguen a la clave del país corresponde al código asignado a la empresa, mientras las cinco siguientes están reservados para designar el producto concreto, numerado por el propio fabricante o distribuidor. El último dígito es una cifra de control, que resulta de aplicar un algoritmo matemático a los otro doce dígitos.

Si en el proceso de lectura del código de barras el número de control no coincide con el resultado de las operaciones indicadas por el algoritmo -que la caja registradora efectúa casi de forma instantánea-, esto significa que se ha producido un error y el sistema pide una nueva lectura.

Cada uno de los dígitos está representado como un grupo de siete módulos de tonalidades claras y oscuras repartidas de manera que cada dígito está formado siempre por dos zonas claras y dos oscuras de anchura variable, según el número de módulos contiguos de un mismo tipo. Esta anchura variable es la que permite que el dispositivo lector decodifique las barras del sistema EAN.

Figura 2.10 Lectura de código de barras

Figura 2.11 Lector de código de barras de sobremesa

Figura 2.12 Lector de código de barras de mano

Figura 2.13 Lector de código de barras de Smartphone (cámara integrada)

2.3.5 Otros códigos similares

NFC

NFC significa Near Field Communication. Se trata de una tecnología inalámbrica que funciona en la banda de los 13.56 MHz (en esa banda no hace falta licencia para usarla) y que deriva de las etiquetas RFID, las cuales están presentes en abonos de transporte o incluso sistemas de seguridad de tiendas físicas.

NFC es una plataforma abierta pensada desde el inicio para teléfonos y dispositivos móviles. El enfoque de esta tecnología es para comunicación instantánea, es decir, identificación y validación de equipos/personas.

Su punto fuerte está en la velocidad de comunicación, que es casi instantánea sin necesidad de emparejamiento previo. Como contrapartida, el alcance de la tecnología NFC es muy reducido, pues se mueve como máximo en un rango de los 20 cm. A su favor también juega que su uso es transparente a los usuarios y que los equipos con tecnología NFC son capaces de enviar y recibir información al mismo tiempo.

La tecnología NFC puede funcionar en dos modos:

Activo, en el que ambos equipos con chip NFC generan un campo electromagnético e intercambian datos.

Pasivo, en el que solo hay un dispositivo activo y el otro aprovecha ese campo para intercambiar la información.

La premisa básica a la que se acoge el uso de la tecnología NFC es aquella situación en la que es necesario un intercambio de datos de forma inalámbrica. Lo usos que más futuro tienen son la identificación, la recogida e intercambio de información y sobre todo, el pago.

Identificación: el acceso a lugares donde es precisa una identificación podría hacerse simplemente acercando nuestro teléfono móvil o tarjeta con chip NFC a un dispositivo de lectura. Los abonos de autobús son un ejemplo muy válido.

Recogida/intercambio de datos: Google es el principal protagonista de este uso, pues en combinación con las etiquetas RFID, utilidades como marcar dónde estamos, recibir información de un evento o establecimiento son inmediatas.

Pago con el teléfono móvil: sin duda alguna es el punto fuerte de los usos del NFC. La comodidad de uso y que el gasto pueda estar asociado a nuestra factura o una cuenta de banco son armas muy poderosas y esta tecnología está camino de ser el método de pago del futuro.

Precisamente en España ha finalizado una de las mayores pruebas con esta tecnología como método de pago. Ha sido en Sitges, con la colaboración de Visa, La Caixa y Telefónica.

Sobre la implantación de la tecnología en dispositivos móviles, Nokia tiene previsto integrarla en todos sus teléfonos nuevos de este año 2011, Apple tenía previsto utilizarla en el Iphone 4 S pero al final no lo hicieron mientras que Blackberry si que lo ha hecho en su terminal Bold 9900/9930. Y en cuanto a Google, ya lo ha colocado en el Nexus S y ha dado soporte para el mismo en Android 2.3.

Figura 2.14 Sistema NFC

CÓDIGOS QR

Los códigos QR, (en inglés QR Code) son un tipo de códigos de barras bidimensionales. A diferencia de un código de barras convencional (por ejemplo EAN-13, Código 3 de 9, UPC), la información está codificada dentro de un cuadrado, permitiendo almacenar gran cantidad de información alfanumérica.

Los códigos QR son fácilmente identificables por su forma cuadrada y por los tres cuadros ubicados en las esquinas superiores e inferior izquierda.

Figura 2.15 Identificación código QR

Aunque el desarrollo inicial de los Códigos QR tenía como objetivo principal su utilización en la industria de la automoción, hoy por hoy la posibilidad de leer códigos QR desde teléfonos y dispositivos móviles permite el uso de Qr Codes en un sinfín de aplicaciones completamente diferentes de las que originales como pueden ser:

-Publicidad

-Campañas de marketing

-Merchandising

-Diseño Gráfico

-Papelería corporativa (tarjetas de visita, catálogos)

-Internet, Webs, blogs

Para poder leer estos códigos existen múltiples lectores QR gratuitos para la mayoría de móviles y marcas, (Nokia, IPhone, BlackBerry, Samsung, Siemens, etc..)

Un detalle muy importante sobre el código QR es que su código es abierto y que sus derechos de patente (propiedad de Denso Wave) no son ejercidos.

Figura 2.16 Código QR

CÓDIGOS BOKODES

Los Bokodes son un nuevo tipo de tecnología de etiquetas digitales, elaboradas por la Camera Group, del MIT media Lab. Esta propuesta es capaz de almacenar cientos de veces más información que los métodos actuales (código de barras, QR code, entre otros). Y lo que lo hace aún más sorprendente es el tamaño, aproximadamente 3mm contra los métodos tradicionales de alrededor de 30mm.

Sin embargo, a pesar de ser pequeño y almacenar gran cantidad de información; la gran ventaja radica en que a diferencia a los códigos actuales , que requieren al menos un posicionamiento de 30 cm para su lecturas siendo capaces de ser leídos a distancias desde 4 metros a incluso 20 metros. Estamos hablando de un código de largo alcance, donde solo es necesaria una cámara fotográfica común, otorgando una numerosa cantidad de usos posibles.

Los desarrolladores de este nuevo tipo de etiquetas digitales, afirman que incluso es posible su uso en otras áreas donde antes no había sido utilizada. Un ejemplo, dice el Dr. Mohan, puede ser en Google Streetview, los locales podrían colocar en las puertas un Bokode con la información de su negocio y el menú de especialidades, cuando la cámara de Google tome la fotos del local, automáticamente adquirirá la data del local y demás información insertada en la etiqueta. Y dada a la característica del Bokode, de presentar diferente tipo de data según su observación angular, podría ser utilizada en la industria de video juegos ó para avances en realidad aumentada.

El diseño explota el efecto Bokeh originalmente encontrado en los lentes de cámaras, los cuales mapean los rayos salientes de un punto en una escena sin foco en forma de discos semi-borrosos en el sensor de la cámara. Utilizan código binario para estimar la distancia relativa y el ángulo de la cámara.

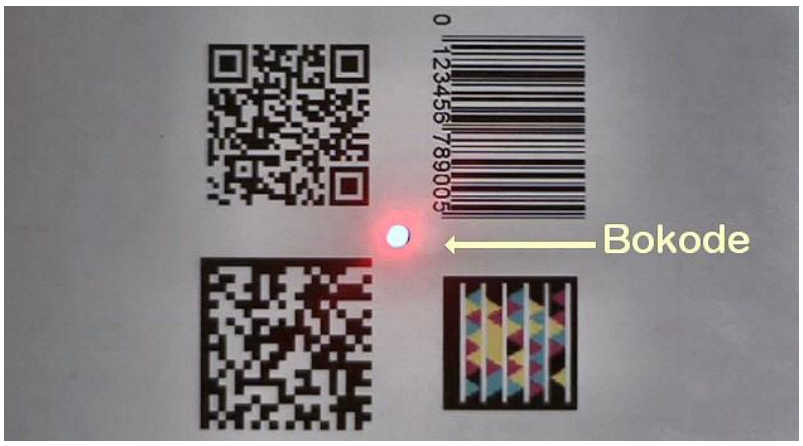


Figura 2.17 Código bokode

CÓDIGOS DATAMATRIX

Datamatrix, o codificación de datos 2D, es un nuevo sistema industrial de codificación bidimensional que permite la generación de un gran volumen de información en un formato muy reducido, con una alta fiabilidad de lectura gracias a sus sistemas de información redundante y corrección de errores (legible hasta con un 20%-30% dañado). Además no es necesario un alto contraste para reconocer el código. El código está formado por celdas de color blanco y negro (perforadas o no perforadas en el caso de la micropercusión) que forman una figura cuadrada o rectangular. Cada una de esas celdas representa un bit de información. La información puede estar codificada como texto o datos en bruto."

Es un código de barras de 2 dimensiones compuesto por módulos cuadrados blancos y negros, que puede almacenar números y texto hasta 2kB. Además permite incluir códigos de corrección de errores para mejorar su decodificación en códigos parcialmente dañados. Debido a su formato y la capacidad de corregir errores, se han convertido en una solución óptima para ser leídos de pantallas de equipos celulares o handhelds.

Aplicaciones:

- Tickets
- Cupones
- Trazabilidad
- Mobile Marketing
- Identificación

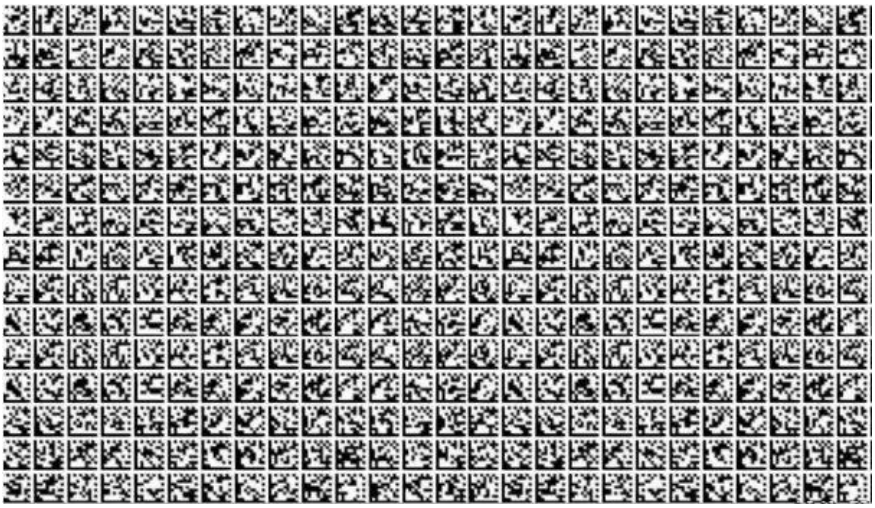


Figura 2.18 Código datamatrix

2.4 Historia del sistema operativo Android

A continuación se muestra la historia de Android, ya que es el principal programa que se eligió para desarrollar este proyecto y por ello que es interesante abordar un poco más a fondo los conocimientos sobre la creación de este sistema operativo a lo largo de la historia.

En julio de 2005, Google adquirió Android Inc., una pequeña compañía de Palo Alto, California fundada en 2003. Entre los cofundadores de Android que se fueron a trabajar a Google están Andy Rubin (co-fundador de Danger), Rich Miner (co-fundador de Wildfire Communications, Inc.), Nick Sears (alguna vez VP en T-Mobile), y Chris White (quien encabezó el diseño y el desarrollo de la interfaz en WebTV). En ese entonces, poco se sabía de las funciones de Android Inc. fuera de que desarrollaban software para teléfonos móviles. Esto dio pie a rumores de que Google estaba planeando entrar en el mercado de los teléfonos móviles.

En Google, el equipo liderado por Rubin desarrolló una plataforma para dispositivos móviles basada en el kernel de Linux que fue promocionado a fabricantes de dispositivos y operadores con la promesa de proveer un sistema flexible y actualizable. Se informó que Google había alineado ya una serie de fabricantes de hardware y software y señaló a los operadores que estaba abierto a diversos grados de cooperación por su parte.

La especulación sobre que el sistema Android de Google entraría en el mercado de la telefonía móvil se incrementó en diciembre de 2006. Reportes de BBC y The Wall Street Journal señalaron que Google quería sus servicios de búsqueda y aplicaciones en teléfonos móviles y estaba muy empeñado en ello. Medios impresos y en línea pronto reportaron que Google estaba desarrollando un teléfono con su marca.

En septiembre de 2007, «InformationWeek» difundió un estudio de Evalueserve que reportaba que Google había solicitado diversas patentes en el área de la telefonía móvil.

El 5 de noviembre de 2007 la Open Handset Alliance, un consorcio de varias compañías entre las que están Texas Instruments, Broadcom Corporation, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, Intel, LG, Marvell Technology Group, Motorola, y T-Mobile; se estrenó con el fin de desarrollar estándares abiertos para dispositivos móviles.9 Junto con la formación de la Open Handset Alliance, la OHA estrenó su primer producto, Android, un plataforma para dispositivos móviles construida sobre la versión 2.6 del kernel de Linux.

El 9 de diciembre de 2008, se anunció que 14 nuevos miembros se unirían al proyecto Android, incluyendo PacketVideo, ARM Holdings, Atheros Communications, Asustek, Garmin, Softbank, Sony Ericsson, Toshiba, Vodafone y ZTE.

2.4.1 Versiones de Android

Historial de actualizaciones

Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones. Generalmente cada actualización del sistema operativo Android es desarrollada bajo un nombre en código de un elemento relacionado con postres.

Android ha sido criticado muchas veces por la fragmentación que sufren sus terminales al no ser soportado con actualizaciones constantes por los distintos fabricantes. Sin embargo, esa situación cambiará con un anuncio que hizo oficial Google en el que comunicó que los fabricantes se comprometerán a aplicar actualizaciones al menos 18 meses desde su salida al mercado.

Los nombres en código están en orden alfabético.

1.0

Liberado el 23 de septiembre de 2008

1.1

Liberado el 9 de febrero de 2009

1.5 (Cupcake)

Basado en el kernel de Linux 2.6.27 El 30 de abril de 2009, la actualización 1.5 (Cupcake) para Android fue liberada. Hubo varias características nuevas y actualizaciones en la interfaz de usuario en la actualización 1.5:38 Habilidad de grabar y reproducir videos a través del modo camcorder Capacidad de subir videos a YouTube e imágenes a Picasa directamente desde el teléfono

-Un nuevo teclado con predicción de texto

-Soporte para Bluetooth A2DP y AVRCP

-Capacidad de conexión automática para conectar a auricular Bluetooth a cierta distancia

-Nuevos widgets y carpetas que se pueden colocar en las pantallas de inicio

-Transiciones de pantalla animadas

1.6 (Donut)

Basado en el kernel de Linux 2.6.29

El 15 de septiembre de 2009, el SDK 1.6 (Donut) fue liberada.40 41 Se incluyó en esta actualización:

-Una experiencia mejorada en el Android Market

-Una interfaz integrada de cámara, filmadora y galería

-La galería ahora permite al usuario seleccionar varias fotos para eliminarlas

-Búsqueda por voz actualizada, con respuesta más rápida y mayor integración con

aplicaciones nativas, incluyendo la posibilidad de marcar a contactos

-Experiencia de búsqueda mejorada que permite buscar marcadores, historiales,

contactos y páginas web desde la pantalla de inicio.

-Actualización de soporte para CDMA/EVDO, 802.1x, VPN y text-to-speech

-Soporte para resoluciones de pantalla WVGA

-Mejoras de velocidad en las aplicaciones de búsqueda y cámara

-Framework de gestos y herramienta de desarrollo GestureBuilder

-Navegación gratuita turn-by-turn de Google

2.0 / 2.1 (Eclair)

Basado en el kernel de Linux 2.6.29

El 26 de octubre de 2009, el SDK 2.0 (Eclair) fue liberado. Los cambios incluyeron:

-Velocidad de hardware optimizada

-Soporte para más tamaños de pantalla y resoluciones

-Interfaz de usuario renovada

-Nuevo interfaz de usuario en el navegador y soporte para HTML5

-Nuevas listas de contactos

-Una mejor relación de contraste para los fondos

-Mejoras en Google Maps 3.1.2

-Soporte para Microsoft Exchange

-Soporte integrado de flash para la cámara

-Zoom digital

-MotionEvent mejorado para captura de eventos multi-touch

-Teclado virtual mejorado

-Bluetooth 2.1

-Fondos de pantalla animados

-El SDK 2.0.1 fue liberado el 3 de diciembre de 2009.

-El SDK 2.1' fue liberado el 12 de enero de 2010.

2.2 (Froyo)

Basado en el kernel de Linux 2.6.32

El 20 de mayo de 2010, el SDK 2.2 (Froyo) fue liberada. Los cambios incluyeron:

-Optimización general del sistema Android, la memoria y el rendimiento

-Mejoras en la velocidad de las aplicaciones, cortesía de la implementación de JIT51

-Integración del motor JavaScript V8 del Google Chrome en la aplicación Browser.

-Soporte mejorado de Microsoft Exchange (reglas de seguridad, reconocimiento automático, GAL look-up, sincronización de calendario, limpieza remota)

-Lanzador de aplicaciones mejorado con accesos directos a las aplicaciones de teléfono y Browser.

-Funcionalidad de Wi-Fi hotspot y thetering por USB

-Permite desactivar el tráfico de datos a través de la red del operador

-Actualización del Market con actualizaciones automáticas

-Cambio rápido entre múltiples idiomas de teclado y sus diccionarios

-Marco por voz y compartir contactos por Bluetooth

-Soporte para contraseñas numéricas y alfanuméricas

-Soporte para campos de carga de archivos en la aplicación Browser

-Soporte para la instalación de aplicación en la memoria expandible

-Soporte para Adobe Flash 10.1

-Soporte para pantallas de alto número de Puntos por pulgada, tales como 4" 720p

2.3 (Gingerbread)55

Basado el kernel de Linux 2.6.35.7 El 6 de diciembre de 2010, el SDK 2.3 (Gingerbread) fue liberada. Los cambios incluyeron:

-Actualización del diseño de la interfaz de usuario

-Soporte para pantallas extra grandes y resoluciones WXGA y mayores

-Soporte nativo para telefonía VoIP SIP

-Soporte para reproducción de videos WebM/VP8, y decodificación de audio AAC

-Nuevos efectos de audios como reverberación, ecualización, la virtualización de los auriculares, y refuerzo de graves

-Soporte para Near Field Communication

-Funcionalidades de cortar, copiar y pegar disponibles en a lo largo del sistema

-Teclado multi-táctil rediseñado

-Soporte mejorado para desarrollo de código nativo

-Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos

-Recolección de elementos concurrentes para un mayor rendimiento

-Soporte nativo para más sensores (como giroscopios y barómetros)

-Un administrador de descargas para descargar archivos grandes

-Administración de la energía mejorada y control de aplicaciones mediante la

administrador de tareas

-Soporte nativo para múltiples cámaras

-Cambio de sistema de archivos de YAFFS a ext4

3.0 / 3.1 / 3.2 (Honeycomb)

-Mejor soporte para tablets

-Escritorio 3D con widgets rediseñados

-Sistema multitarea mejorado

-Mejoras en el navegador web predeterminado, entre lo que destaca la navegación por pestañas, autorelleno de formularios, sincronización de favoritos con Google Chrome y navegación privada

-Soporte para videochat mediante Google Talk

-Mejor soporte para redes Wi-Fi

-Añade soporte para una gran variedad de periféricos y accesorios con conexión USB: teclados, ratones, hubs, dispositivos de juego y cámaras digitales. Cuando un accesorio está conectado, el sistema busca la aplicación necesaria y ofrece su ejecución.

-Los widgets pueden redimensionarse de forma manual sin la limitación del número de cuadros que tenga cada escritorio.

-Se añade soporte opcional para redimensionar correctamente las aplicaciones inicialmente creadas para móvil para que se vean bien en Tablets

4.0 (Ice Cream Sandwich)

-Interfaz estilo Honeycomb, en cualquier dispositivo, homogeneidad entre teléfonos, televisiones, tablets, netbooks

-Barra de estado redimensionable

-Reconocimiento de voz del usuario

-Reconocimiento facial, lo que haría que puedas cambiar la vista

-Un único y nuevo framework para las aplicaciones

En la gráfica siguiente se muestra claramente que actualmente la versión más utilizada de Android es la correspondiente a la 2.2.

La versión 4.0 no aparece en el gráfico ya que actualmente funciona solamente en un dispositivo móvil el Nexus One y con muchos problemas.

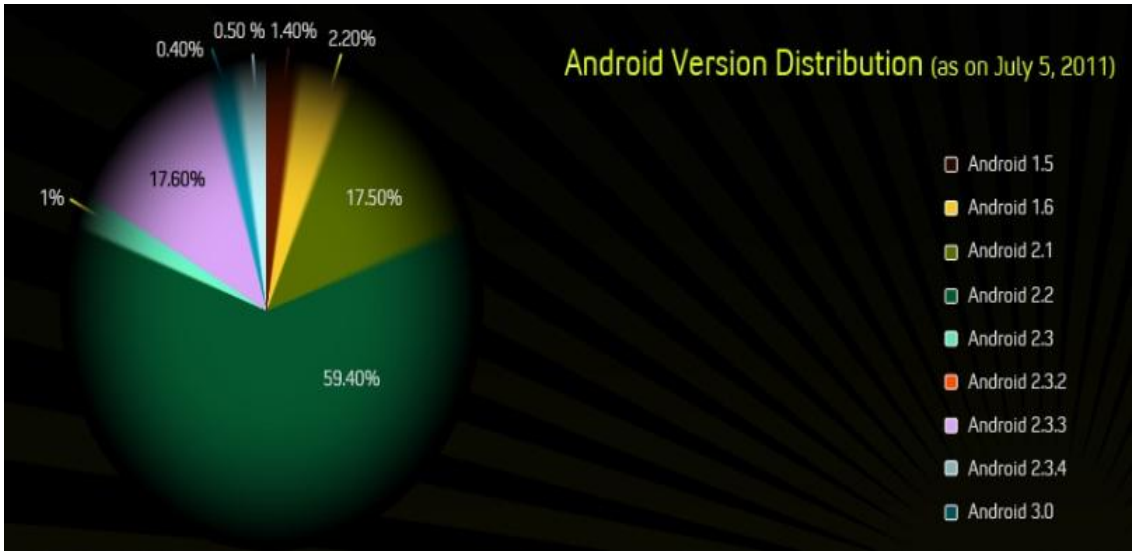


Figura 2.19 Gráfica circular de la distribución de las versiones de Android

Extraída de http://www.ticbeat.com/tecnologias/historia-android-infografi/

2.4.2 Programación en Android

En lo que se refiere al desarrollo de aplicaciones, se pretendía que el entorno fuese cómodo y fácil para los programadores, y de coger aplicaciones ya existentes fuese lo más rápido y fácil posible. De ahí la elección de Java, por ser un lenguaje popular del cual había muchas aplicaciones hechas, a parte de diversos entornos gráficos de desarrollo integrados.

A pesar de esto, utilizar Java ME implicaba la compra de licencias a Sun Microsystems y según Google, esta empresa no estaba preparada para la siguiente generación de teléfonos inteligentes. Por otro lado, utilizar la maquina virtual y las API de Java SE directamente (las cuales no habían estado optimizadas para dispositivos pequeños) no se adaptaban a las necesidades de Android.

Por esto Google desarrolló la maquina virtual Dalvik la cual permitió ejecutar los programas de Android en ella.

Esta maquina virtual, tiene las siguientes mejoras respecto a la maquina virtual de JAVA SE:

Está optimizada para utilizar poca memoria gracias a que es una maquina basada en registros y no en pila

Está diseñada para ejecutar cada aplicación aislada, en una maquina virtual diferente

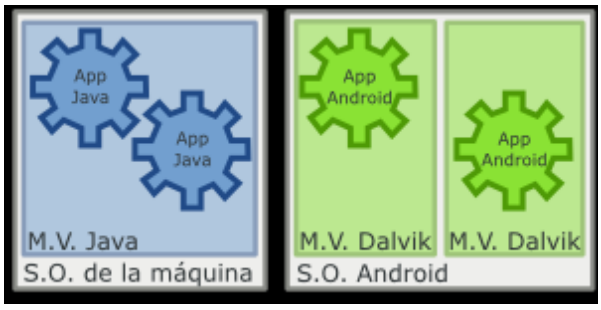


Figura 2.20 Gráfica máquina virtual Android

Gracias al proyecto Apache Harmony, además, Android proporciona un subconjunto relativamente completo de la API de Java SE. De esta forma, crear o llevar aplicaciones Java, ya sean en código fuente o en formato binario resulta en algunos casos trivial.

A pesar de esto, la API de Google no es 100% compatible con la de Java SE.

Bibliotecas como las de entorno grafico no están disponibles en Android. Además conceptos como la gestión de la seguridad han sido completamente reinventados.

El resultado es una plataforma que no es estrictamente Tecnología Java, simplemente usa el lenguaje Java como base para crear aplicaciones.

Estructura de una aplicación

En este apartado se definen y explican los 4 componentes o bloques básicos que puede contener una aplicación.

No todas las aplicaciones requieren de los 4 tipos, pero si una combinación entre algunos de ellos.

Cada vez que queramos usar uno de estos componentes se debe de especificar en un archivo llamado AndroidManifest.xml.

Activity

Un Activity es el componente inicial de una aplicación, y por ello el más importante. Se podría definir como una simple pantalla de la aplicación. Cada actividad se define en una clase aparte y se identifica por extender de la clase Activity. Sus funciones son mostrar una interfaz con vistas (Views) y responde a eventos concretos. Para pasar de una Activity a otra, se utiliza la clase Intent, en la que se debe indicar la activity origen y la activity destino.

Intent Receiver

Los Intent Receiver se utilizan para códigos que se quieren ejecutar cuando sucede un evento externo, por ejemplo cuando llaman al teléfono o se ha localizado la posición GPS. A diferencia de los activities, estos no muestran una interfaz, en vez de eso muestran una notificación avisando al usuario que algo ha ocurrido.

No es necesario definir los intents receivers en el AndroidManifest.xml, ya que pueden llamarse usando *Context.registerReceiver().*

Service

Service o servicio es un código que define una acción que se realizará en segundo plano, como puede ser localizar la posición GPS del usuario, mientras el usuario está consultando el tiempo que hará hoy. Por ello, se trata de un código no visible, suele utilizarse mientras se está en un activity. Se ejecuta usando *Context.startService()* y este terminará cuando finalice su función.

Content Provider

Cuando queremos almacenar datos en bases de datos, como por ejemplo con SQLite, no podemos compartir esos datos con otras aplicaciones. Mediante un Content Provider se consigue compartir esa información. Se trata de una clase que implementa métodos para dejar que otras aplicaciones puedan guardar y obtener esos datos mediante el gestor contentprovider.

AndroidManifest.xml

El AndroidManifest.xml es un fichero necesario para cualquier aplicación Android. Se encuentra en la carpeta raíz por defecto y describe valores globales de la aplicación. Incluye la descripción de las actividades y servicios, el tipo de información de cada actividad y que pueden soportar y como serán ejecutadas. También se definen los permisos, librerías y actividades que se usarán en la aplicación.

A continuación se mencionan las opciones más importantes que se pueden definir en el manifest.

Package: situación de los ficheros que se Uses-premission: permisos que se le otorgan a la aplicación que por defecto no tiene.

Ej: Acceso a Internet.

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

Uses-library: librerías de Google en el programa. Ej: Mapas de Google.

<uses-library android:name="com.google.android.maps" />

Activity: permite a una actividad iniciarse. Todas las activities deben estar especificadas en el AndroidManifest.xml. El siguiente código debe incluirse en la actividad inicial para que sea la primera en ejecutarse.

<category android:name="android.intent.category.LAUNCHER" />

Ciclo de vida de las activities y tratamiento.

Muy parecido al ciclo de vida de las aplicaciones, aunque solo tratando de los componentes activities. A diferencia de las aplicaciones, para hablar del ciclo de vida de una activity tenemos que hacer referencia al "stack de activities" del sistema que apila exclusivamente activities que están ejecutándose. Cuando se cambia de una activity a otra, la nueva activity que es creada se colocada en la parte superior de la pila y pasa a ser la activity que se ejecuta en ese momento, mientras que la activity anterior queda por debajo de ésta en la pila, quedando así en background hasta que la principal no sea eliminada de la fila. Las activities tienen cuatro estados según su interacción con el usuario:

Activa o en ejecución. Este estado se da cuando la activity está en la parte superior de la pila, es la última activity creada por la aplicación y será la última que el sistema intente eliminar.

En pausa. Este estado se da cuando una activity deja de ser la principal pero todavía queda visible en pantalla, esto es posible en activities transparentes que dejan ver la inferior. Ésta continúa viva manteniendo todo la información del usuario.

Parada. A diferencia del el estado "pausa" la activity en este estado no es visible en pantalla. Sigue manteniendo la información del usuario pero frecuentemente será eliminada por el sistema para liberar memoria.

Eliminada, puede ser porque el sistema necesite memoria o simplemente porque la aplicación lo ha ordenado.

Podemos ver que el ciclo de vida de una aplicación está relacionando con el ciclo de vida de una activity, dando prioridad a los procesos que contienen una. Como conclusión Android considera como imprescindibles aquellos procesos con los que el usuario está interaccionando y hace todo lo posible para que no le afecten estados críticos de memoria que se puedan dar. Capítulo 1: Presentación de ANDROID 17

Para manejar el estado de las activities tenemos encontramos los siguientes métodos incluidos en la clase activity: OnCreate(), OnDestroy(), OnPause(), OnStop(), OnFrezze(), OnResume() , OnRestart() y Finish(). Cada vez que se lanza una aplicación o pasamos de una activity a otra, ésta es creada por el sistema, en cambio si se navega hacia atrás ésta puede ser restablecida (recuperando toda la información del usuario) o volviendo a ser creada de nuevo, también se pueden poner en pausa, en ambos casos ésta pasa a estar en segundo plano. Una activity puede estar finalizada por la aplicación, o por el sistema en caso de que éste necesite memoria.

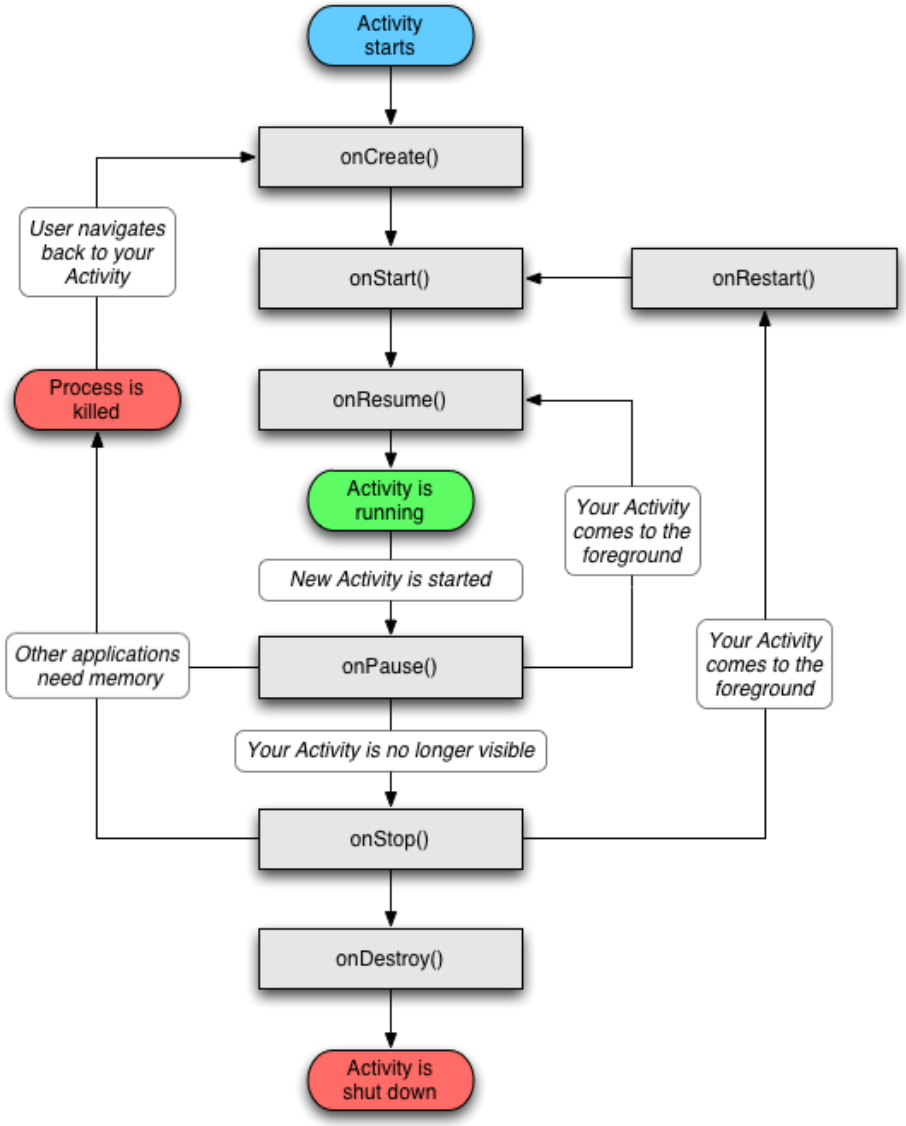


Figura 2.21 Esquema ciclo de vida de una Activity

Cambiar de Activity

Seguramente la aplicación que tendrá crear tendrá varias pantallas y intuitivamente se piensa que desde una clase se podrá pasar de una pantalla a otra solo poniendo las instrucciones adecuadas, lo más lógico sería pensar en SetContentView, una instrucción que indica que XML o View mostrar. Pues bien, de esta manera el programa en Android

no funcionaría. En Android cada pantalla que tenga la aplicación es una activity diferente y para poder pasar de una a otra se utilizan los Intents. Como ya se ha dicho anteriormente un Intent utilizando para abrir una activity es como decir "quiero inicializar esta pantalla" y Android ya tiene este método predefinido para hacerlo de la mejor manera. Igualmente todas las activities tendrán que estar declaradas en el AndroidManifiest. Cada vez que pasemos a una activity nueva la anterior será retenida y ésta quedará en Background pero a diferencia de un proceso ésta no seguirá activa. Hay que comprender muy bien el ciclo de vida de estas para que la aplicación no abuse de memoria.

Paso de datos entre activities.

De una manera muy parecida al cambio de una activity con Intents también se realiza el paso de información entre estas. Antes de llamar al Intent se crean bundles donde se introducen los datos a traspasar y, en la activity donde se reciben los datos, se crea otro Bundle que los recoja. Resulta muy sencillo ya que son dos líneas de código. También puede darse el caso que se quiera devolver un valor al finalizar una activity. Esto también es posible usando los métodos startSubActivity(Intent, int) y onActivityResult(int, int, String), El primero creara la activity y los valores que ésta devuelva serán recogidos en el segundo método.

Versión m5 de el emulador.

La última versión lanzada para Android es la m5-RC14, de un total de cuatro versiones disponibles. Esta última todavía continua teniendo limitaciones importantes a tener en cuenta para el desarrollo de aplicaciones, quizás la más importante es que no soporta

Bluetooth. Además de emular los programas creados por los usuarios, también se puede acceder a otras aplicaciones, navegar por Internet, escuchar música, ver videos, almacenar datos en la agenda de contactos etc. También contiene un numeroso paquete de elementos de desarrollo para que el programador pueda observar su funcionamiento y coger ideas y un programa que transforma los archivos .class en .dex que es el usado por la máquina Dalvik.

El emulador viene incorporado con el entorno de desarrollo cuando nos lo descargamos de Internet. Puede ser ejecutarlo directamente o, lanzado cuando compilamos nuestras aplicaciones para Android con el Eclipse. Puede que incluso sea necesario meterse dentro del sistema para emular señales de GPS o restablecer los parámetros del usuario. Entre las limitaciones más importantes como la del bluetooth ya nombrada, el sistema permite simular llamadas entrantes, la utilización de una tarjeta de memoria externa o incluso la interrupción de un programa si se ha recibido un SMS.

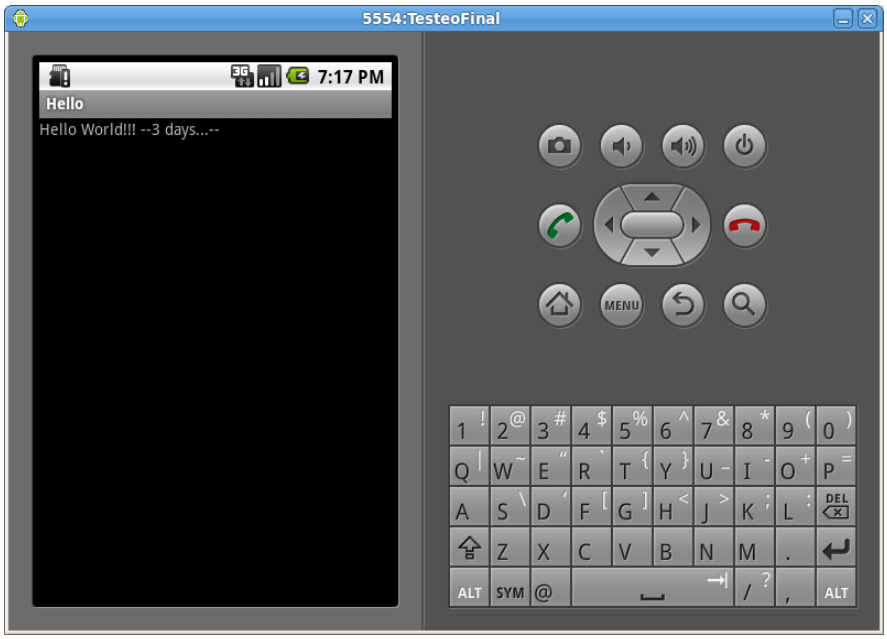


Figura 2.22 Emulador android de eclipse

Plug-in de Android para Eclipse

Para crear aplicaciones para Android, Google ha creado un Plug-in perfectamente adaptado para Eclipse que se puede descargar gratuitamente desde la página oficial de Android. Al instalarse el plug-in y crear un proyecto nuevo vemos que éste está estructurado de la siguiente manera:

Un directorio (src/) donde estará el package con la clase principal que hayamos creado (tendrá el código genérico que crea Android) y el archivo R.java,

R.java es una clase que se va modificando sola. Cada vez que creamos una variable ésta aparece reflejada en el R.java. Se podría decir que es un índice a todos los recursos declarados en el proyecto y el motivo de éste es que sea más rápido localizar las referencias de una aplicación que se estén buscando.

AndroidManifiest.XML. Donde se declaran las activities y se indica que Intents realizarán. Éste es un archivo con extensión XML que dice al sistema que es lo que hay que hacer, con qué capacidad y qué componentes utiliza. Cuando se genera un nuevo proyecto, este archivo aparecerá automáticamente y medida que se vayan creando activities, estas se tendrán que ir declarando en el Manifiest.

Si se observa la primera activity que se crea junto con un proyecto nuevo aparece en el Manifiest como la pantalla principal a la aplicación.

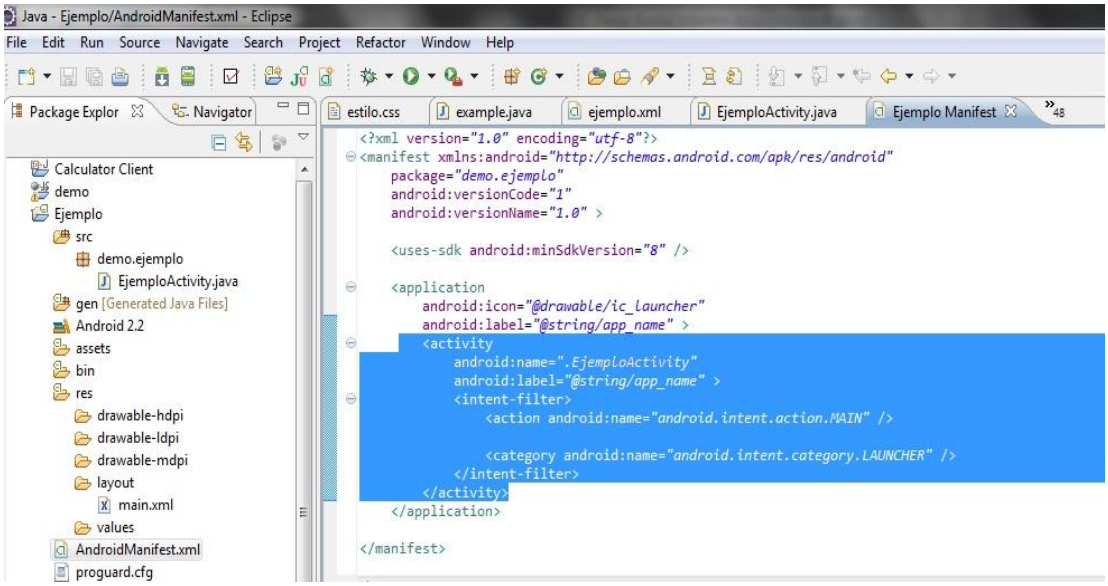


Figura 2.23 Ejemplo de Manifest.xml

Un directorio de carpetas (res/) que contiene las carpetas drawable, layout y values. Donde drawable contendrá las imágenes (en formato .PNG) que utilice nuestra aplicación, values contendrá un archivo llamado strings.XML que definirá cadenas de caracteres constantes y el subdirectorio layout, también en formato .XML donde definiremos los elementos que forman la interfaz gráfica de nuestra aplicación. Es posible crear nuevas carpetas dentro del directorio para poner otro tipo de contenidos que necesitamos.

En la siguiente figura vemos un ejemplo de cómo poner una imagen para el fondo de una activity, en este caso la imagen es la utilizada en el proyecto como ventana principal.

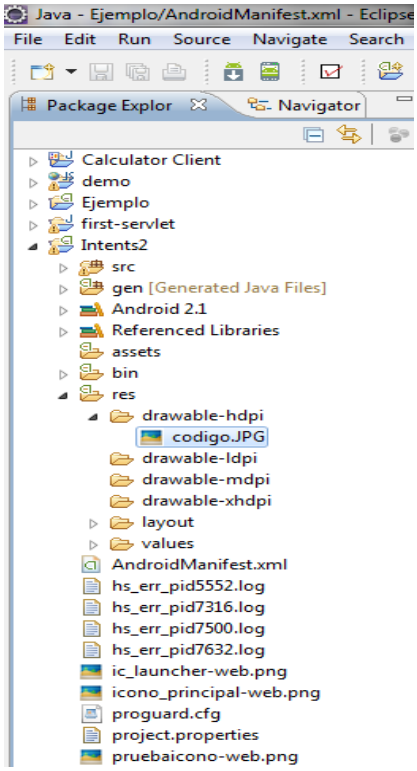


Figura 2.24 Ejemplo drawable-hdpi

XML-based layout

Android plantea un modelo alternativo para la construcción de interfaces gráficas, llamado "XML-based Layout". En Android las UIs pueden ser construidas de dos maneras, directamente invocando operaciones a las API desde las clases del programa o a través de relacionar estas con archivos XML. De la primera manera, puede resultar complicado realizar una modificación de una UI, además la conexión de cada elemento con las Views de manera incorrecta puede ocasionar errores en la creación de las activities y demasiada pérdida de tiempo debugando problemas de creación de interfaz. Por ello Android separa la creación de UI en archivos XML donde cada etiqueta del árbol es un elemento del tipo View representado dentro de la pantalla del programa, con sus atributos correspondientes que lo personalizan. De esta manera resulta mucho más

rápido y sencillo crear una UI. Este modelo está basado en el diseño de páginas Web, donde se separan el diseño y las manipulación de los datos.

En la siguiente figura se muestra el estado de un xml dentro de la pestaña “graphical layout”, se observa que a la izquierda hay una paleta con una gran variedad de carpetas y dentro de estas se encuentran todo tipo de herramientas que podemos agregar a nuestro xml, todas ellas aparecen de forma muy visual y son reconocidas muy fácilmente, como parecido razonable nos vendría a la mente Visual Basic.

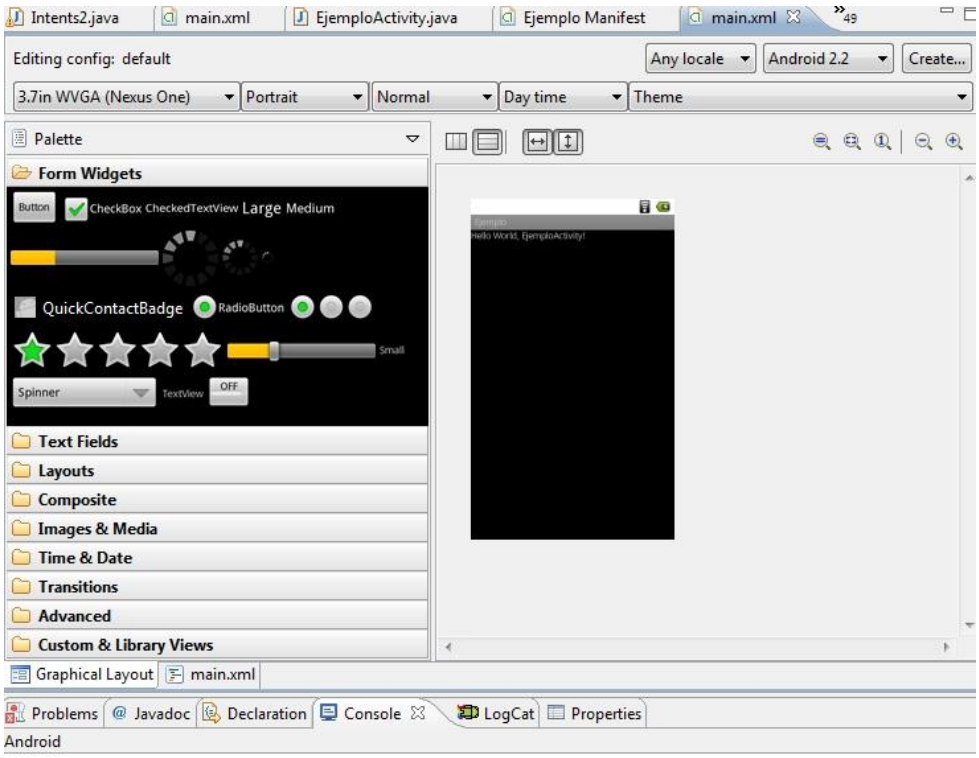


Figura 2.25 Archivo xml

Instalar un .apk

El archivo apk es el archivo ejecutable que se generará automáticamente en el proyecto que se ha creado, este archivo servirá para posteriormente instalar el proyecto android en el dispositivo móvil, para saber donde se guarda este archivo, primero procederemos a irnos a la ruta donde se guarda el Workspace en eclipse

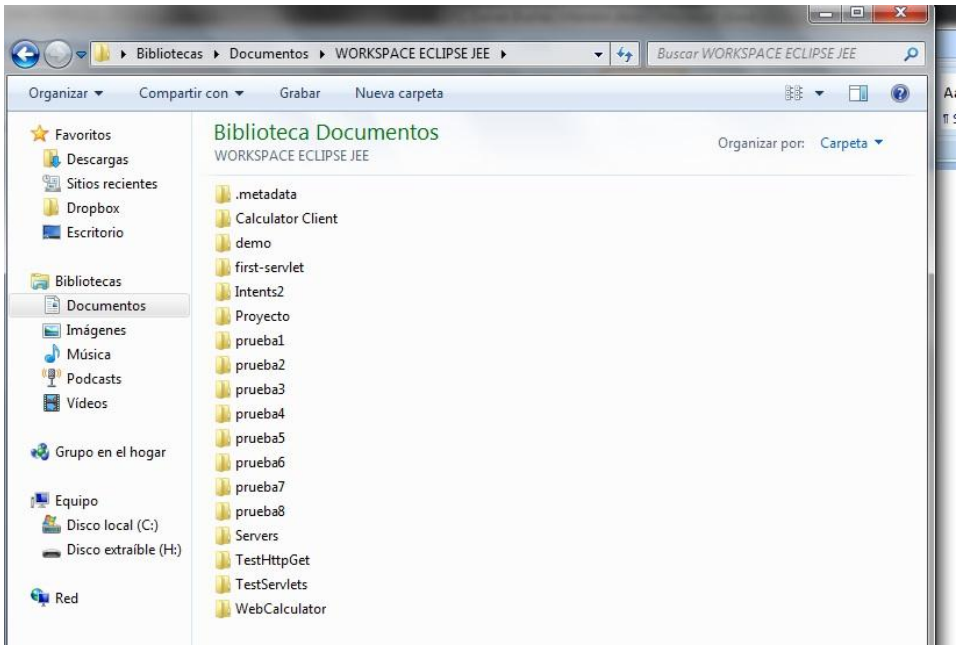


Figura 2.26 Workspace eclipse

Después se procederá a la ubicación en la carpeta de nuestro proyecto, en este caso será el proyecto llamado Intents2

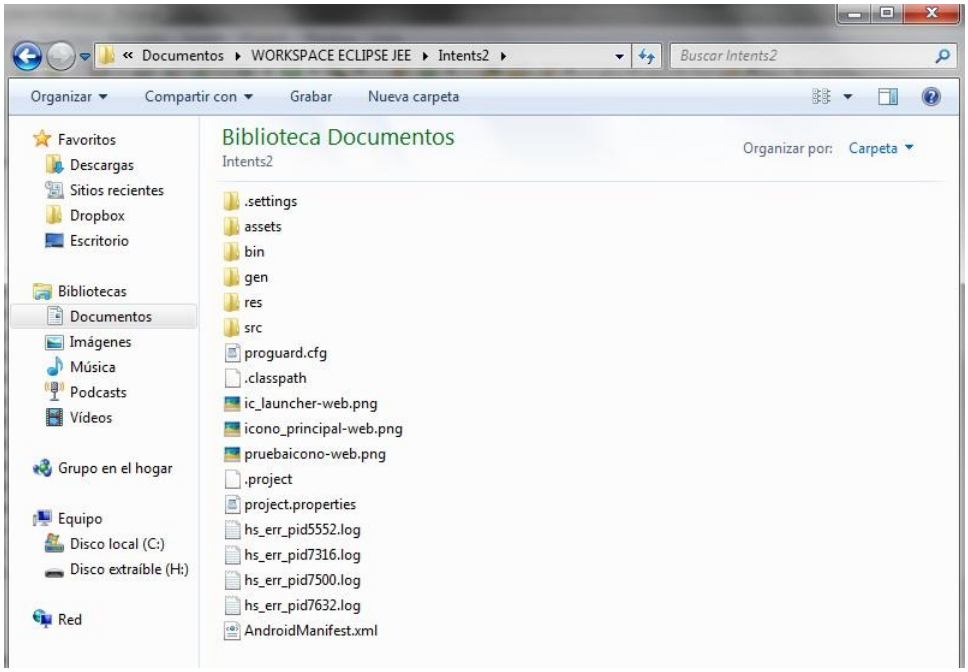


Figura 2.27 Carpeta Proyecto en Workspace

Una vez dentro se observa que tenemos las mismas carpetas que veíamos en la perspectiva de explorador de paquetes de Eclipse.

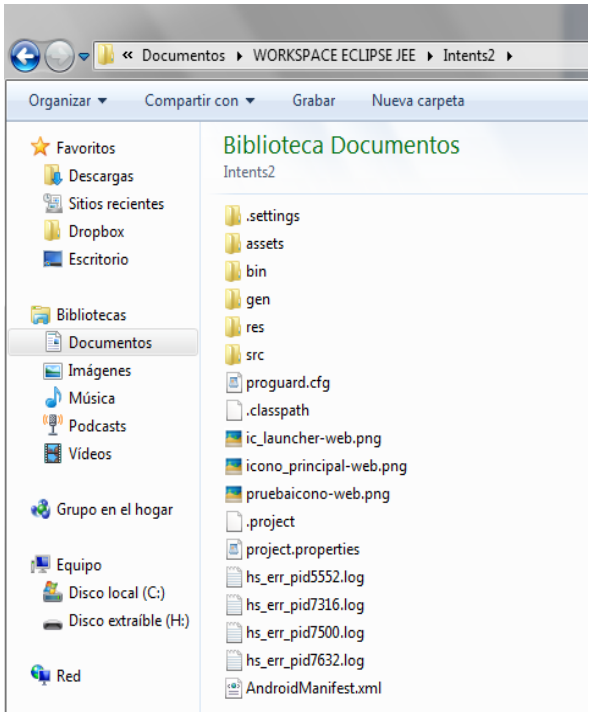


Figura 2.28 Contenido carpeta proyecto de nuestro Workspace

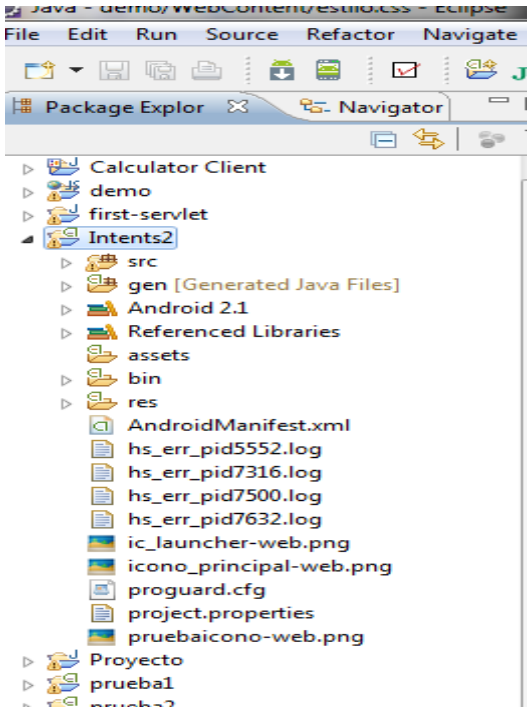


Figura 2.29 Contenido perspectiva del explorador de paquetes de Eclipse

Finalmente nos adentraremos en la carpeta bin del proyecto y dentro de esta tendremos el archivo apk que es el que nos servirá para instalar la aplicación en el dispositivo móvil.

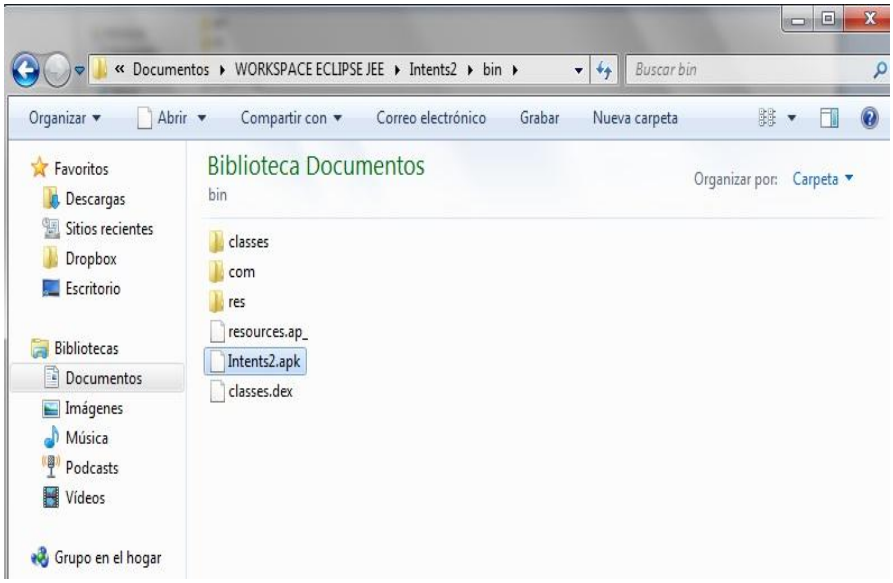


Figura 2.30 Archivo apk

Mediante el cable usb se conectará el dispositivo móvil y el ordenador y se procederá a copiar el archivo apk del proyecto en el directorio raíz del dispositivo móvil.

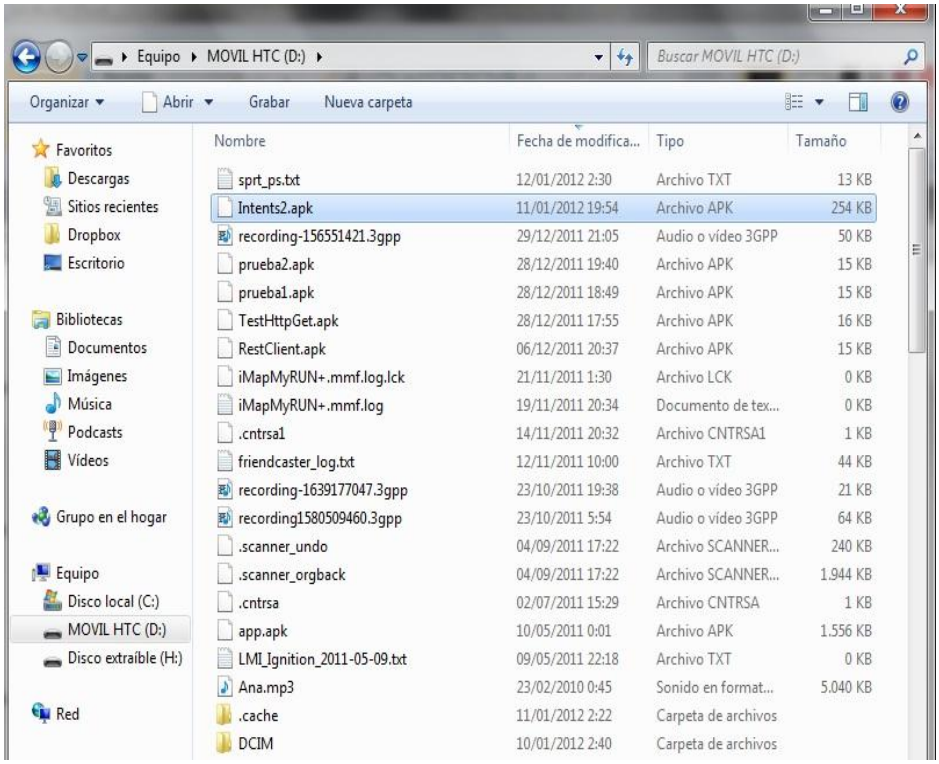


Figura 2.31 Directorio raíz dispositivo móvil

3. Desarrollo técnico

En este apartado se pretende dar a conocer el funcionamiento de la aplicación de forma explicativa, de forma un poco más detallada también, y saber cuáles han sido las herramientas que se han llevado a cabo y el por qué de las mismas, para la obtención del producto final.

También se pretende dar a conocer el uso de las clases de este proyecto y su relación entre ellas.

3.1 Aplicación Scan & Go

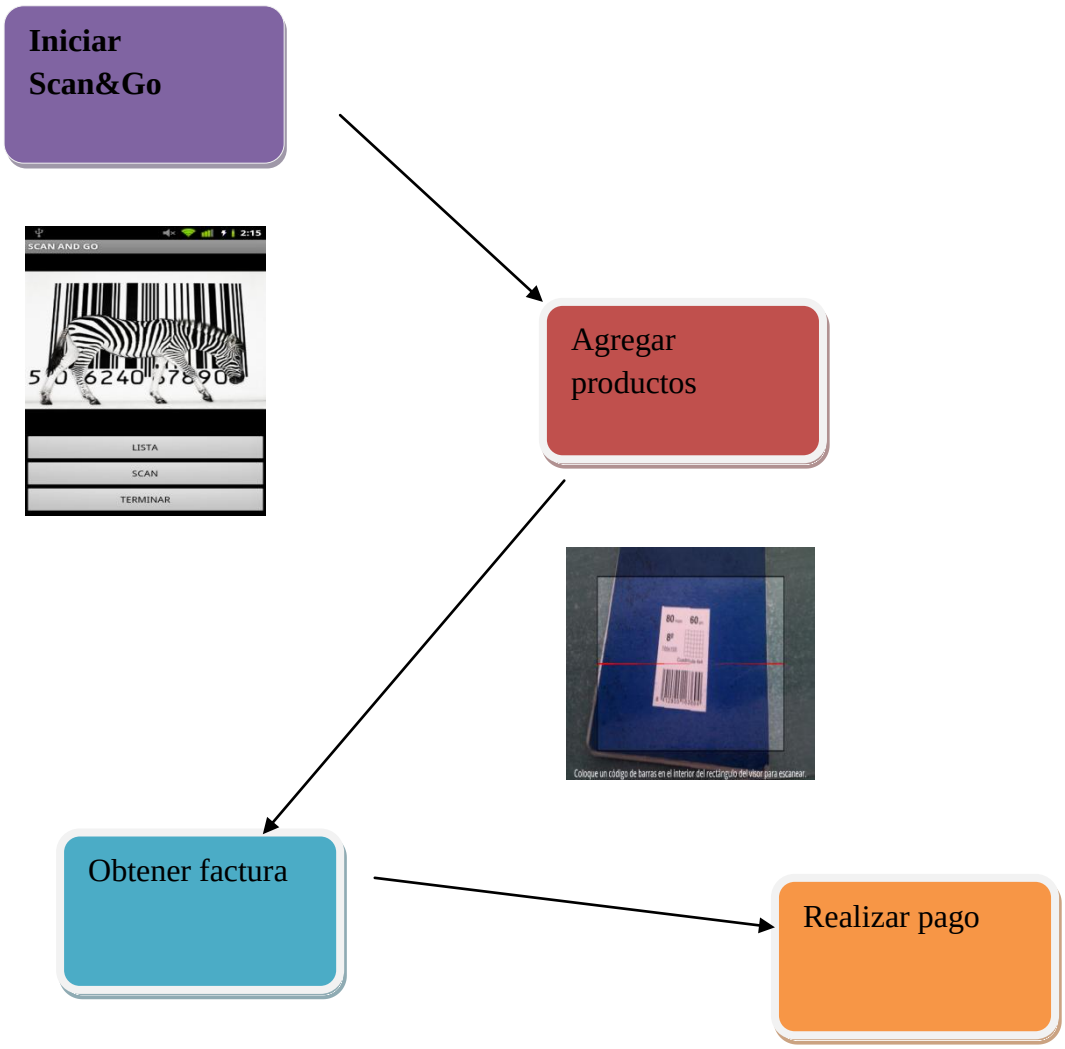


Figura 3.1 Gráfico explicativo del funcionamiento de Scan&Go

En el anterior gráfico explicativo se intenta dar a conocer las diferentes fases a nivel práctico de cómo funciona el proyecto de Scan and Go.

La primera fase el usuario o consumidor, interactúa con el dispositivo móvil, abriendo en primer lugar la aplicación para poder utilizarla posteriormente.

En la segunda fase el usuario interactúa la aplicación móvil con el producto que se desea añadir al carrito de la compra, se procede al escaneo del código de barras del producto para añadirlo a la lista virtual.

En la tercera y última fase, después de agregar todos los productos a la cesta virtual de la aplicación, procederemos a la sincronización con la caja o ordenador , para posteriormente realizar el pago de nuestra compra.

Barcode Scanner, aplicación Android que es completamente gratuita y que ocupa 589 KB , esta aplicación sirve para escanear códigos de barras o códigos mediante el uso de la cámara integrada del Smartphone, es de las aplicaciones más descargadas del market de Android superando los 10 millones de descargas.

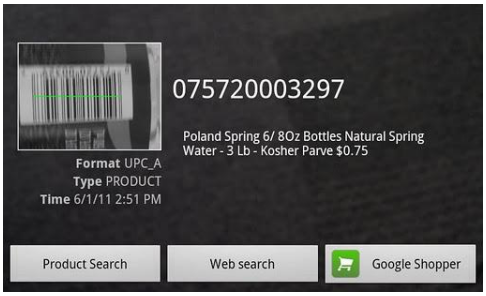


Figura 3.2 Barcode Scanner

Eclipse, un software sobre el que se pueden montar herramientas de desarrollo para programar bajo cualquier lenguaje mediante la instalación de los plugins adecuados. Se creó con la idea de aligerar la programación en Java (Eclipse está desarrollado íntegramente en Java) ya que las herramientas Java son realmente pesadas y requieren de muchos de los recursos de nuestro sistema. La solución a estos problemas llegó con Eclipse y sus librerías nativas para cada S.O. llamadas SWT.

Eclipse tiene la ventaja de que no necesita instalación. Se puede descargar el programa de la web de eclipse (www.eclipse.org/downloads).

Figura 3.3 Eclipse

SDK Android, Otro elemento muy importante para desarrollar aplicaciones en Android, es su SDK (Sofware Development Kit). En el SDK se encuentran todas las librerías y utilidades necesarias para poder trabajar con Android.

Plugin Eclipse, Para integrar Android con Eclipse, se necesita un Plugin, que dependiendo de la versión del IDE que se posea, se instalará de una forma concreta.

Apache Tomcat, se ha optado por montar un servidor Tomcat, ya que es un servidor web con soporte de servlets y JSPs, ideal para este proyecto, ya que la clase que hace de servidor, BarCodeServer, tiene conexión a Base de Datos, y está basado en Servlets java.

Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Las versiones más recientes son las 7.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

3.2 Descripción de la aplicación

La aplicación que se ha desarrollado tiene como objetivo la lectura de códigos de barras mediante el uso del programa Barcode Scanner, que proporcionará la información correspondiente al producto en cuestión, creando así una lista de la compra con los productos escaneados, que podremos ir visualizando en nuestro smartphone.

Se puede explicar mejor la aplicación, teniendo en cuenta varios aspectos :

3.2.1 Conexión de la aplicación

Desde el punto de vista de las conexiones, la aplicación consta de tres proyectos compilados en Eclipse:

-Barcode : Cliente de BarCodeServerAndroid

-BarcodeServer : Servidor con conexión a Base de Datos, basado en Servlets java.

-BarcodeDB : Cliente de base de datos (crea la base de datos, la tabla y permite insertar nuevos elementos).

3.2.2 Relación de la aplicación

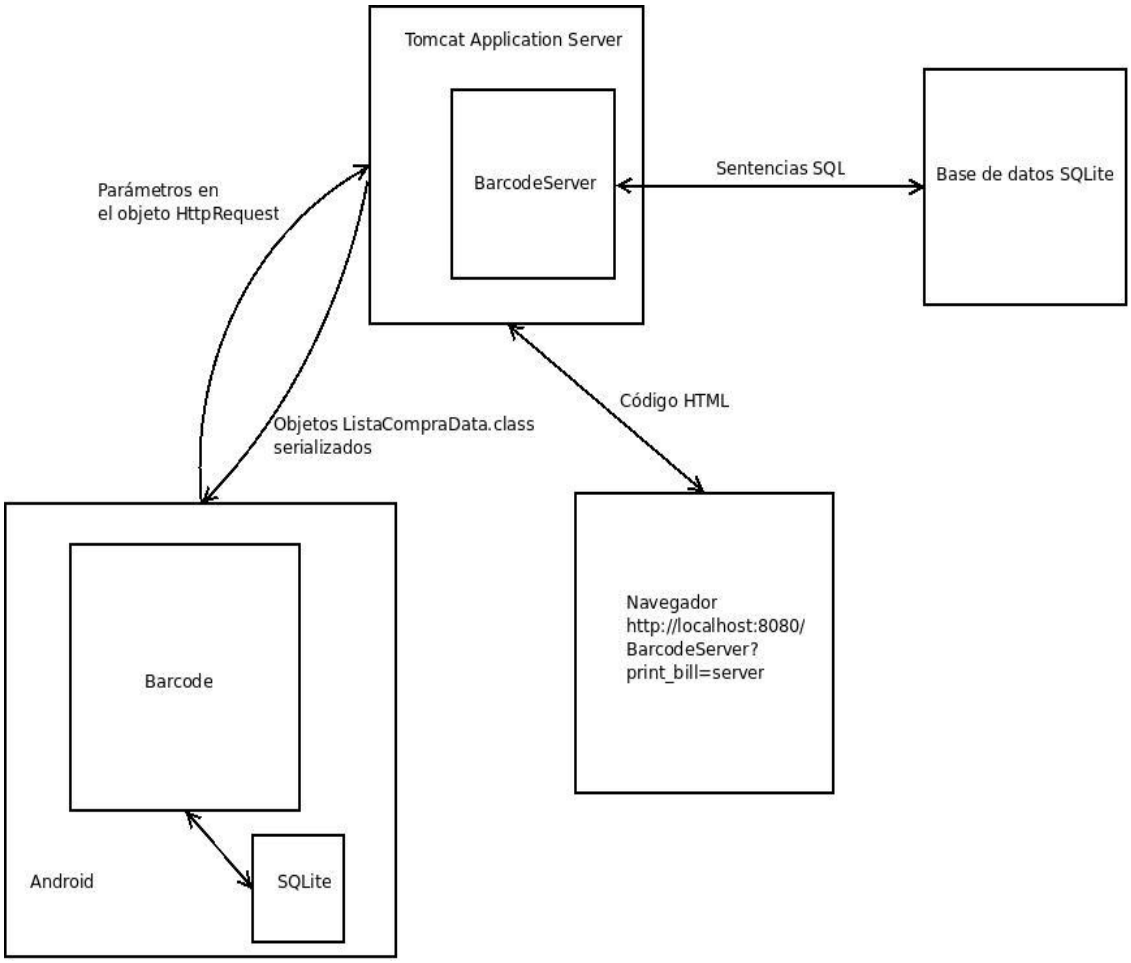


Figura 3.4 Diagrama relacional aplicación Scan&Go

Vemos que la clase Barcode mediante su base de datos SQLite peticiona al servidor los parámetros que quiere, y éste le devuelve los objetos de la clase ListaCompraData serializados.

La clase BarCodeServer como se aprecia en el diagrama, está encapsulado en el Apache Tomcat, ya que está en pleno contacto con el servidor a la hora de bajar las imágenes de los productos y sus posteriores datos, también está en contacto con la base de datos SQLite, mediante las sentencias SQL podremos accederá a la base de datos para recoger información de los productos. Por otro lado se observa también, que mediante el código html especificado, podremos acceder a la factura del cliente pasándole los datos al servidor Tomcat.

3.2.3 Clases utilizadas en la aplicación

Proyecto Barcode

Esta parte, cuyo proyecto es BarCode referencia al cliente , el usuario que manejará los datos mediante la aplicación.

Consta de ocho clases:

CustListItemAdapter, DB, DownloadFile, Intents2, ListaCompra , ListaCompraData, IntentIntegrator y IntentResult.

Clase Intents2

Activity principal que permite:

a) Ver la lista;

b) Enviar los datos de la factura al servidor para procesarlos;

c) Escanear códigos de barras

Mediante el layout main obtenemos la vista inicial de la aplicación , es decir , la imagen y los dos botones correspondientes.

```
setContentView(R.layout.main);
```

Para obtener una instancia de la base de datos.

```
db = new DB(this);
```

Para conseguir una instancia de IntentIntegrator para llamar a las clases que escanean los códigos de barras y poder recuperar el resultado.

```
integrator = new IntentIntegrator(this);
```

Al tener estas dos instancias ya podemos llamar mediante intents al método con el cual se iniciará el escaneo de códigos de barras al pulsar el botón SCAN.

```
Intents2.this.integrator.initiateScan();
```

Para enviar la lista de productos al servidor y hacer la factura se tiene que implementar el botón Terminar de la pantalla de inicio, primero se envía el código del producto y el número de productos para posteriormente llamar a la clase DownloadedFile que se conectará con el servidor

```
DownloadFile dlf = new DownloadFile(Intents2.this.getApplicationContext());

dlf.execute(params);
```

Después de hacer el escaneo de código de barras se llama al método onActivityResult

```
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
```

que lo que hace es comprobar que se ha escaneado un código de barras, obtiene la clase que contiene el resultado del escaneo, se guarda el código obtenido y se inserta en la base de datos , una vez se guarda ya está guardado se puede volver a realizar el proceso de escaneado nuevamente.

Clase CustListItemAdapter

Esta clase se utiliza para personalizar la lista de la compra mediante un adaptador personalizado de manera que se pueda poner en cada fila de la lista varios objetos.

En el constructor de estas clase se inicializa el contexto con los datos de la aplicación e inicializa el array de objetos ListaCompraData ,que usa el adaptador. Además, se personaliza, se infla, el layout para obtener la vista de cada fila de la lista.

En cada fila de la lista se obtiene una imagen del producto escaneado, el nombre del producto , el precio y el número de veces que tenemos ese mismo producto en la lista.

Si por casualidad el consumidor se equivoca a la hora de escanear un producto y quiere quitarlo de la lista , se puede borrar el producto manteniendo pulsado el dedo en la fila en cuestión del producto.

Al final de la lista aparece el total del precio acumulado del listado de productos que vamos teniendo.

Clase DB

Clase que gestiona las relaciones con la base de datos SQLite, reside en Android y mantiene los productos escaneados por el cliente, así como su información, para mostrarlos, calcular el total y enviar los datos de la factura al servidor.

Para insertar los datos de un producto escaneado se utilizará el método :

```
public void insertData(ListaCompraData lcd) {
```

que lo que hará es insertar en una tabla llamada Códigos las columnas de código, nombre, precio, número e imagen.

Para obtener un array de objetos del tipo ListaCompraData con la información de cada uno de los productos que existan en la base de datos se tendrá que llamar al método

```
public ListaCompraData[] selectCodes() {
```

Clase ListaCompra

Activity que muestra información sobre los productos escaneados, permite borrarlos de la lista y aumentar el número de productos de la lista mediante un control numérico.

En el constructor de esta clase se crea un adaptador para la correspondencia de datos de la BD con la lista .

```
clia = new CustListItemAdapter(this, getValues());
```

Este adaptador se añadirá a la lista.

```
lista.setAdapter(clia);
```

Con el anterior método :

```
protected ListaCompraData[] getValues() {
```

se obtienen los datos de la base de datos y los devuelve como un array de objetos ListaCompraData para que los use el adaptador.

Es importante destacar que cuando se borra un producto es necesario borrar la fila entera de la base de datos , inutilizar el adaptador de esa fila y crear uno de nuevo, para poder seguir agregando productos a la lista, mediante la utilización del método :

```
protected void deleteCode(String codigo) {
```

Clase DownloadFile

Clase que gestiona las conexiones al servidor, envía los parámetros de los productos y recibe objetos ListaCompraData serializados, los deserializa y almacena la información en la BD.

```
private static final String SERVER =
"http://192.168.1.129:8080/BarCodeServer?code=";
```

Con esta línea de código se especifica la dirección IP del servidor y el puerto activo del Tomcat, que servirán para proceder a la descarga de la imagen, donde se utilizará para identificar el producto en la lista de la compra.

En el constructor de esta clase se inicializa el control ImageView donde se cargará la imagen y una referencia a la lista para guardarla posteriormente en la base de datos.

Con el siguiente método pasamos como parámetros el código del producto.

```
protected ListaCompraData doInBackground(String... arg0) {
```

Si la url comienza con el parámetro "print_bill=client" se pretende mandar la factura al servidor, esta url pasaría los parámetros de la factura, en caso contrario se pasaría la url donde se descargaría la imagen construida con el codigo del producto.

Lo pasa por un lector de streams con buffer (BufferedInputStream) y lo convierte en un lector de objetos serializados (ObjectInputStream), después lee el stream y deserializa el objeto.

Una vez acabado el método anterior se ejecuta el siguiente método:

```
protected void onPostExecute(ListaCompraData result) {
```

El parámetro es el que se devuelve desde doInBackGround, en este caso el objeto ListaCompraData recuperado del servidor.

Solo se insertarán los datos en el caso de que se haya recuperado algo del servidor, estos datos serán la asignación de la imagen, del nombre, del precio y del número de elementos del producto, guardaremos todos estos datos en la base de datos y en la fila de la lista.

Si por el contrario no se ha recurado ningún dato del servidor, quiere decir que la lista está vacía y por tanto se pasará como nombre del producto sin identificación y se actualizará el número de productos a uno, posteriormente se procederá a guardar los datos de la fila.

Clase IntentIntegrator

Clase de utilidad que ayuda a facilitar la integración con el explorador de código de barras a través de Intents.

Esta es una sencilla forma de invocar el escaneo de código de barras y recibir el resultado, sin ninguna necesidad de integrar, modificar, o aprender el código fuente de un proyecto.

Con este método se cogen los resultados obtenidos en esta clase para posteriormente pasarlos a la otra clase llamada IntentResult .

```
public static IntentResult parseActivityResult(int requestCode, int resultCode, Intent intent) {
```

Requiere que BarcodeScanner esté instalado en el dispositivo móvil , si no está instalado esta clase nos informará que tenemos que instalarlo.

En resumen, esta clase nos facilita la interactuación con las librerías Zxing de BarcodeScanner para la lectura de códigos de barra.

Clase IntentResult

Esta clase obtiene el resultado del escaneo de códigos de barras invocado a través del intent de la clase anterior IntentIntegrator.

Aquí se obtienen por tanto los datos obtenidos por la anterior clase IntentIntegrator, datos que en este caso será, la referencia del código de barras del producto que se ha escaneado, una vez con esta referencia obtenida, ya tenemos capturado el producto mediante el uso del BarcodeScanner.

Proyecto BarcodeServer

Este proyecto consta de dos clases : BarcodeServer y DB.

Clase BarcodeServer

Clase que implementa el servlet que recibirá las conexiones vía HTTP, tanto desde Android como desde el navegador.

Si recibe conexiones desde Android puede pasar lo siguiente:

a) conectar a la BD, recuperar los datos de los productos y reenviarlos a la aplicación Android como objetos ListaCompraData serializados;

b) recibir los datos necesarios para generar la factura, consultar la información a la base de datos, y enviar la factura al navegador web;

c) recibir una petición HTTP de un navegador para mostrar la factura del cliente.

En este servlet aparecerán los dos métodos por defecto como son doGet y doPost.

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
```

El método anterior doGet inicializa la base de datos, recoge el parámetro que indica el envío de la factura para posteriormente preparar la factura, donde se creará una la tabla de productos y precios, para después poder hacer la consulta a la base de datos del nombre del producto y el precio.

Con la siguiente dirección, se podrá acceder desde el navegador web al cajero, mientras la aplicación y el servidor están funcionando, para poder conocer si se ha enviado factura de los productos del cliente o por el contrario si aún no se ha enviado nada.

```
localhost:8080/BarCodeServer/?print_bill=server
```

Por último, en el segundo método, el doPost, se pasarán los parámetros a la función doGet.

Clase DB

Clase que gestiona las relaciones con la base de datos SQLite, que mantiene todos los productos de la aplicación (no sólo los que el cliente escanea, como pasa con la base de datos de Barcode en Android).

Recupera los datos de los productos y construye objetos ListaCompraData o líneas de código HTML que devuelve a la clase BarcodeServer para que los procese.

Primero de todo esta clase se conecta a la base de datos y crea las tablas en caso necesario.

En caso necesario, crea la tabla de productos e inserta los productos.

Posteriormente ejecutada esta sentencia, se procedería a recoger el fichero del sistema de archivos en la carpeta contenedora de nuestras imágenes, se leería el fichero y se almacenaría como imagen, después se guardaría el objeto en la base de datos , y finalmente, se asignaría los demás valores, precio , nombre y código.

Primero se comprueba si existe un determinado código en la base de datos con el método :

```
public boolean isCode(String code) throws SQLException {
```

Para devolver la información del producto se requiere el método :

```
public ListaCompraData getProduct(String code) throws SQLException {
```

Si la query devuelve un valor, se crea un objeto ListaCompraData y se le asigna la información del producto y sino se creará un objeto vacío.

```
conn=DriverManager.getConnection("jdbc:sqlite:C:/Users/DANI/Documents/WORKSPACE/BarCodeServer/barcode.sqlite3");
```

Con el siguiente método:

```
public String getHtml(String code, int num) throws SQLException {
```

se obtiene la información de un producto y lo devuelve codificado como una fila de tabla html, también se obtiene el precio total multiplicando el numero de productos por el precio del producto.

Proyecto BarcodeDB

Contiene solamente la clase BarcodeDB.

Clase BarcodeDB

Clase que crea la base de datos SQLite, la tabla de productos, y que inserta nuevos productos en la base de datos.

Aquí se comprobará si existe la tabla Productos en la base de datos donde se almacenan todos los productos de la lista de la compra, si ya existe un determinado producto asignará los valores pertinentes (nombre, código, precio, etc.) y los recuperará.

Si por el contrario lo que se pretende es agregar nuevos productos a la base de datos, lo que se debe hacer es ejecutar la sentencia siguiente en nuestra terminal del ordenador:

```
java -jar BarcodeDb.jar fichero_imagen codigo_producto nombre_producto precio_euros
```

3.2.4 Secuencia de acciones

a) Utilizar BarcodeDB para crear la base de datos, la tabla de productos e insertar productos. Una vez creada la base de datos, se pueden insertar nuevos productos en cualquier momento, sin afectar al rendimiento de la aplicación.

b) El cliente escanea productos desde su teléfono móvil, donde tiene instalada la aplicación "Scan and Go", que contiene las clases del producto Barcode. Si es la primera vez que escanea, se crea la base de datos y la tabla Productos en Android. Cuando se escanea un producto, se inserta en la BD, y se conecta con el servidor para que devuelva la información del producto (imagen, nombre y precio), y se graba esa información en la BD.

c) El cliente consulta la lista de productos escaneados. La lista muestra los elementos que el cliente ha escaneado, con la imagen, el nombre y el número de elementos de cada producto que quiere comprar. El número de elementos se puede incrementar y decrementar. La lista muestra también el total de la factura actualizada.

d) El cliente termina la compra y envía la factura al servidor. El servidor procesa los elementos de la lista del cliente, busca el precio en la base de datos y calcula el precio total de la factura (aquí podría complicarse la cosa con tipos de cliente, descuentos, etc.).

e) El navegador (que simula el terminal del cajero), muestra la factura que se utilizará para que el cliente pague el precio.

3.2.5 Esquema de la aplicación

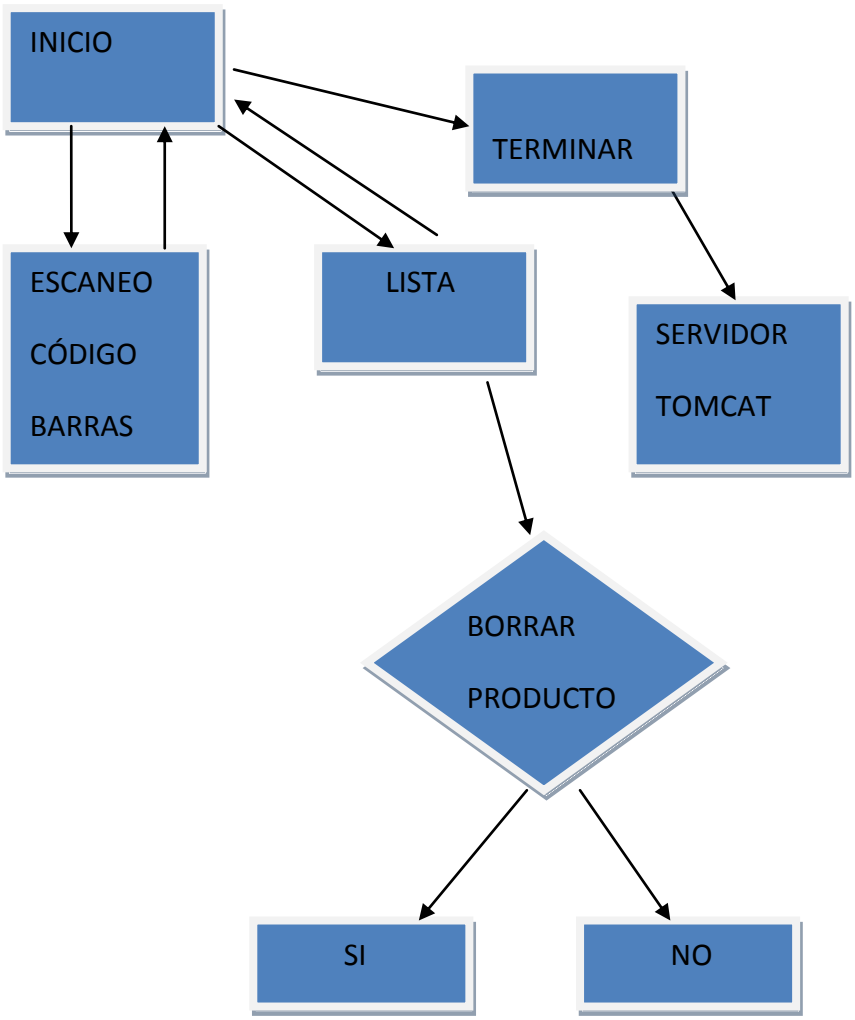


Figura 3.5 Ciclo de vida de la aplicación

El anterior esquema viene a representar el ciclo de vida de la aplicación, los avances o retrocesos que podemos dar con la aplicación en marcha.

1-Inicio

La primera fase de todas cuando se abre la aplicación y se accede al menú principal del programa, aquí se visualizar la imagen en portada que representa a la aplicación, en este caso una cebra con un código de barras al fondo haciendo un pequeña simbiosis entre el color de la cebra y el código de barras en cuestión.

En esta plana se visualiza a parte de la imagen, el nombre de la aplicación en la parte superior, y en la parte inferior los tres botones principales de la aplicación, que son el de Lista para poder visualizar la lista, Scan que sirve para escanear el producto y el de Terminar que sirve para pasar los datos de la lista hacia el cajero.

Figura 3.6 Pantalla principal Scan&Go

2-Escaneo

Después de pulsar el botón de SCAN se llamará a las librerías del BarcodeScanner para poder realizar la lectura del código de barras del producto en cuestión.

Una vez se haga la lectura del código se preparará para hacer la lectura del siguiente código de barras, si se quiere ver la lista de la compra se deberá pulsar el botón de retroceso del celular, para poder volver a la pantalla inicial.

Figura 3.7 Pantalla Scan de Scan&Go

3-Lista

Al seleccionar en el botón del menú principal la lista, se accede a la lista donde se visualizarán todos los productos que se hayan escaneado previamente.

Aquí se visualizará una imagen del producto para identificarlo más fácilmente , también se verá el nombre del producto, su precio y la cantidad de ese mismo producto, se podrá variar de forma forzada, es decir, empleando los botones más y menos situados al lado derecho del precio del producto, o bien se podrá variar de forma automática cuando ese mismo producto se vuelva a escanear nuevamente.

En la lista de la compra también dispondremos de un total del precio de los productos que se vayan agregando, para recordar en todo momento, la cantidad de dinero que se va a gastar el cliente.

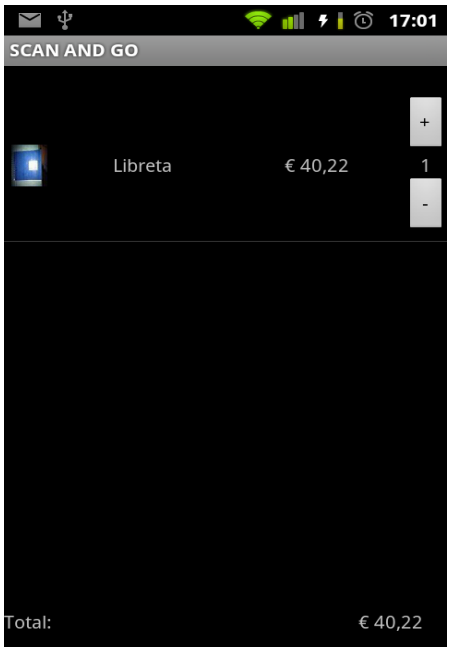


Figura 3.8 Pantalla lista llena Scan&Go

El producto también se podrá borrar de manera manual manteniendo pulsado con el dedo el producto en cuestión, se abrirá un texto informando que se va a borrar ese producto, y solamente se deberá aceptar si así se desea.

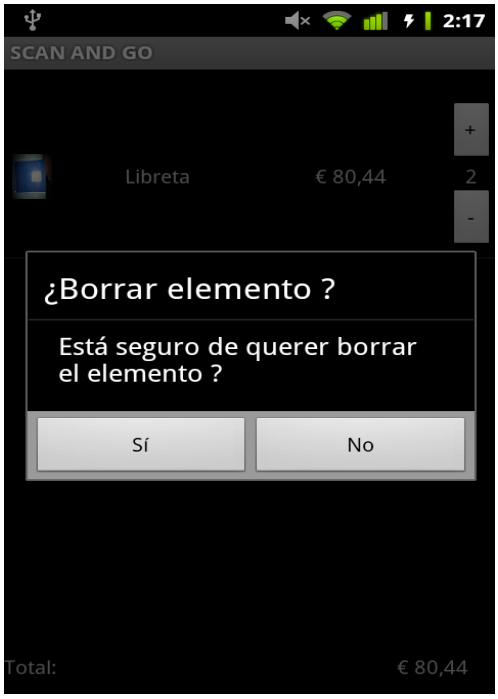


Figura 3.9 Pantalla borrar lista Scan&Go

Si no hay ningún producto agregado la lista permanecerá por defecto siempre vacía.

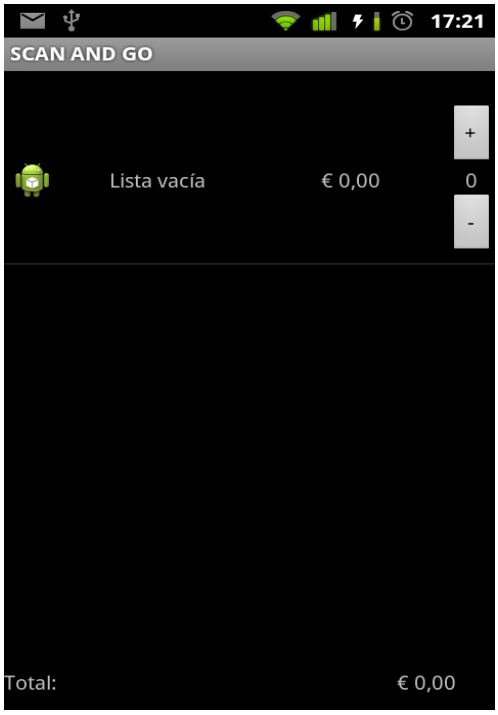


Figura 3.10 Pantalla borrar lista vacía Scan&Go

5- Terminar

Este botón situado también en el menú principal de la aplicación sirve para mandar la lista que tenemos echa de productos al ordenador para poder llevar a cabo la factura final del cliente.

Como se aprecia en la siguiente captura cuando se acciona el botón, hay un mensaje informativo que explica que la factura ha sido enviada al servidor, este mensaje se mantiene durante unos segundos.

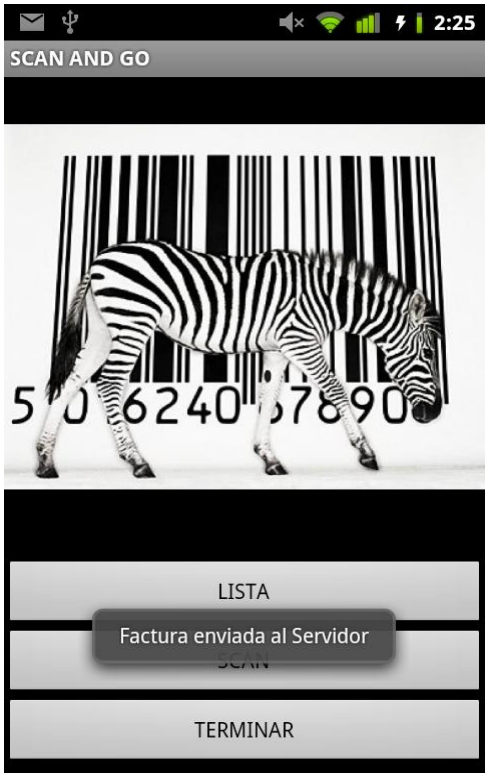


Figura 3.11 Pantalla factura Scan&Go

6- Servidor Tomcat

En este paso se reflejará el estado de la factura en el servidor en todo momento, si el servidor no le llega la petición para pasar la factura del cliente, es decir, si aún no se ha

pulsado el botón Terminar, el servidor mostrará en todo momento un mensaje advirtiendo que no hay ninguna factura.

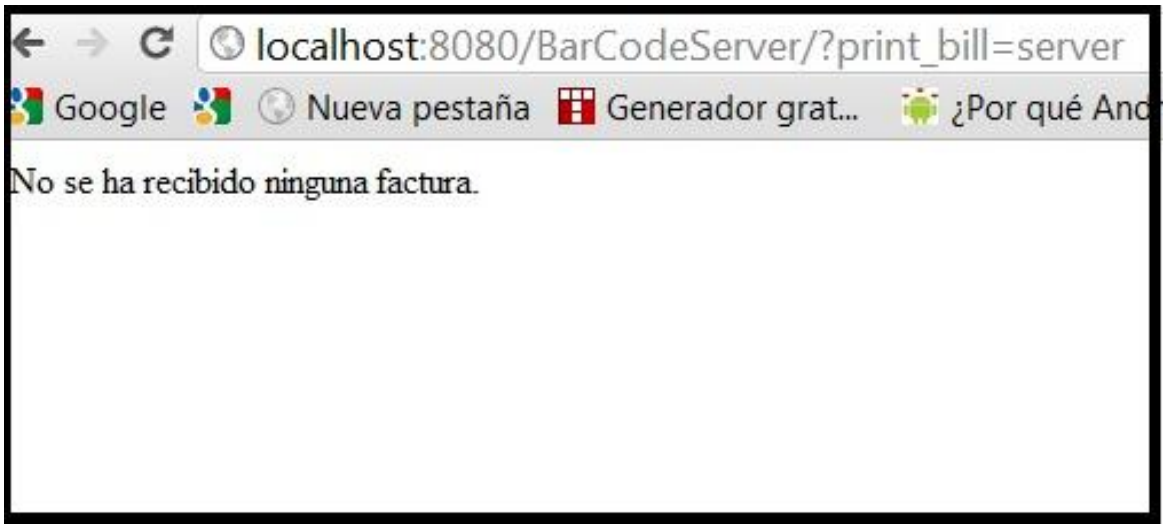


Figura 3.12 Factura vacía servidor

Mientras que por el contrario, si el servidor recibe la petición del cliente para que éste muestre la factura, se obtendrá una tabla correspondiente a la lista de la compra que el cliente haya realizado mediante su terminal.

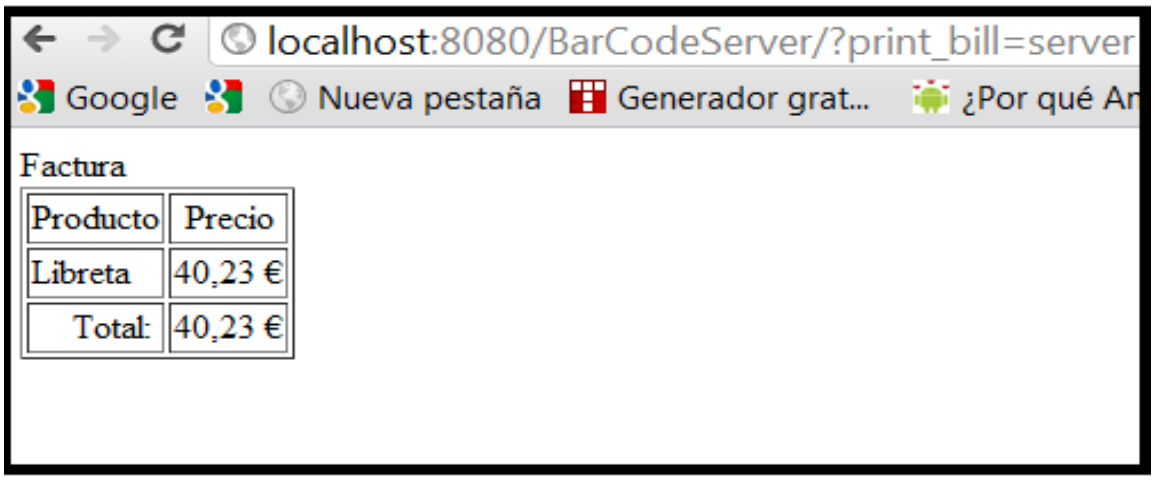


Figura 3.13 Factura llena servidor

Finalizada este paso , el cliente ya podría realizar el pago de su compra

4. Conclusiones y trabajo futuro

4.1 Conclusiones finales

El objetivo principal del proyecto era desarrollar una aplicación Android, que permitiera a cualquier consumidor de un supermercado, poder realizar la lista de la compra mediante su dispositivo móvil, primero registrando los productos que se iban añadiendo a nuestra cesta, mediante el código de barras de estos, para posteriormente terminada la lista de dichos productos, poder proceder al pago de los mismos, mediante el uso de una factura, llevada a cabo gracias a la interconexión de datos del dispositivo móvil y el cajero, que en el caso, sería el servidor creado.

Al final, se ha conseguido todo este objetivo final, gracias a la programación en Android mediante el programa eclipse , y también al servidor montado Apache Tomcat, y a la base de datos SQLite creada en él.

En términos generales, se puede decir que el objetivo principal ha sido conseguido, aunque queda abierta la posibilidad de muchas mejoras del proyecto.

4.2 Trabajo futuro

Como se mencionaba anteriormente, el proyecto queda como en la mayoría de los casos, con un amplio abanico de mejoras futuras.

Una de éstas mejoras, podría ser la de crear una web del supermercado en cuestión, dando cabida a los usuarios del mismo, a poder registrarse vía online, para posteriormente administrarles una especie de login, un usuario y contraseña, sólo válidos para las personas que en su terminal dispongan de la aplicación Scan&Go, que servirán para poder acceder al pago de la compra mediante la validación del servidor.

Con esta idea, se pretende abarcar más otro tipo de clientes, con herramientas tecnológicamente avanzadas, para después beneficiarse de ventajas como el pago más rápido y con menos problemas que el convencional, y por qué no , también de ventajas, descuentos, promociones y bonos que se podría llevar a cabo mediante el uso de determinada información legal del consumidor.

5. Bibliografía

[1] Android. Guía para desarrolladores, Segunda edición

[3] Artículo: *foro android*

http://www.android.es/foro/viewtopic.php?f=5&t=58

[3] Artículo:*instituto nacional de estadística*

http://www.ine.es

[3] Artículo: *ministerio de indústria*

http://www.ontsi.red.es/ciudadanos/indicator/141

[3] Artículo: *el economista smartphones*

http://www.eleconomista.es/interstitial/volver/empresas/empresas-finanzas/noticias/3620452/12/11/Un-49-de-los-telefonos-en-Espana-en-el-tercer-trimestre-son-smartphones.html

[3] Artículo: *adsl zone android*

http://www.adslzone.net/article4637-android-es-considerado-el-sistema-operativo-del-futuro.html

[3] Artículo: *aprendiendo android*

http://www.elandroidelibre.com/2010/06/aprendiendo-android-ii.html

[3] Artículo: *curso android*

http://blog.vidasconcurrentes.com/android/creando-una-aplicacion-de-android-empaquetado-y-publicacion/

[3] Artículo: adslzone *android futuro*

http://www.adslzone.net/article4637-android-es-considerado-el-sistema-operativo-del-futuro.html

[3] Artículo: *wikipedia información general*

http://www.wikipedia.es

[3] Artículo: *logit android futuro*

http://www.logit42.com/archives/5788 [11] Artículo: *foro android*

[3] Artículo: redusers ice cream versión 4 android

http://www.redusers.com/noticias/ice-cream-sandwich-funciona-en-menos-del-1-de-los-dispositivos-android/

[3] Artículo: historia android

http://www.ticbeat.com/tecnologias/historia-android-infografi/ [13] Artículo: foro android

[3] Artículo: Blog scan and go Madrid

http://blog.vivevaldemoro.com/2010/10/27/carrefour-scan-go-la-vida-mas-facil/

[3] Artículo: youtube instalación de sofware eclipse y tomcat

http://www.youtube.es

[3] Artículo: problemàtica supermercados

http://www.educarm.es/templates/portal/ficheros/websDinamicas/30/cajadelhiper.pdf

[3] Artículo: android developers intents

http://developer.android.com/reference/android/content/Intent.html

[3] Artículo: autoservicio supermercados

http://blog.sage.es/innovacion-tecnologia/analisis-de-un-sistema-de-autoservicio-en-un-supermercado

Ingeniería Técnica en Telecomunicaciones especialidad Telemática

SCAN AND GO

Anexos

DANIEL BRAÑAS VILLARDÓN
PONIENTE: PERE BARBERAN

PRIMAVERA AÑO 2012

TecnoCampus
Mataró-Maresme

Índice

Anexo I. Software

ECLIPSE

ANDROID SDK EN WINDOWS

Este sofware es necesario para configurar Eclipse correctamente de forma que pueda trabajar con Android para poder crear aplicaciones.

El ordenador que se ha utilizado para este proyecto ha sido un AMD a 1,9 GHZ con 6 GB de Memoria RAM, el sistema operativo usado ha sido Windows 7.

Es necesario tener instalado el Java Development Kit , para ello solo hace falta bajarse de manera gratuita en la web oficial de Oracle, mediante el siguiente enlace linkaremos en la opción de Java Platform (JDK):

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Posteriormente bajaremos Eclipse y el plugin ADT. Para descargar Eclipse se puede entrar en su web oficial **http://www.eclipse.org/downloads/** y seleccionar la opción Eclipse IDE for Java Developers.

Descarga de Android SDK

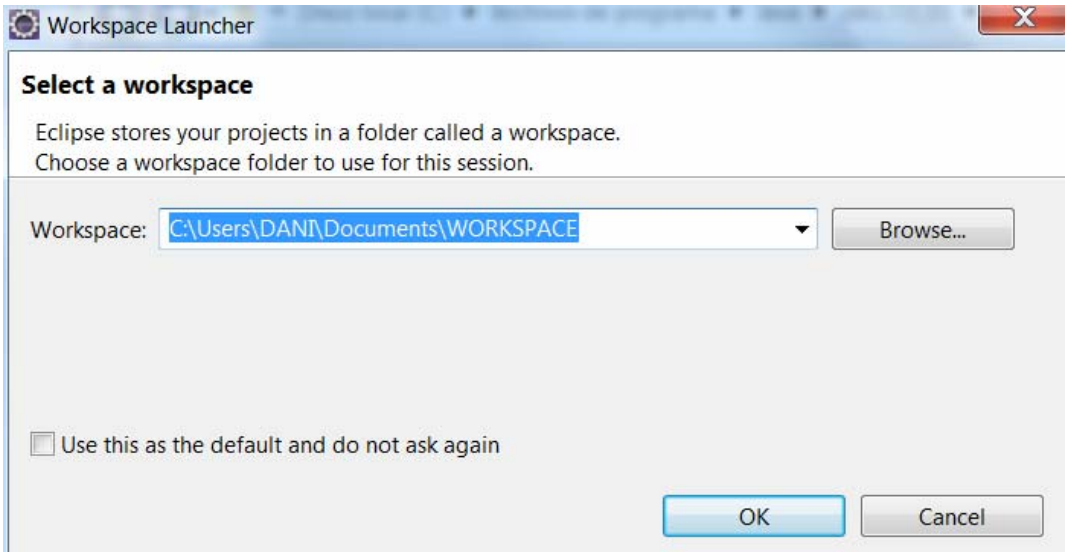
Para descargar el Android SDK basta con acceder al siguiente enlace y descargarnos la versión de Windows:

http://developer.android.com/sdk/index.html

Instalación del plugin ADT en Eclipse

ADT (Android Development Tools) son una serie de herramientas para poder programar en Android desde eclipse.

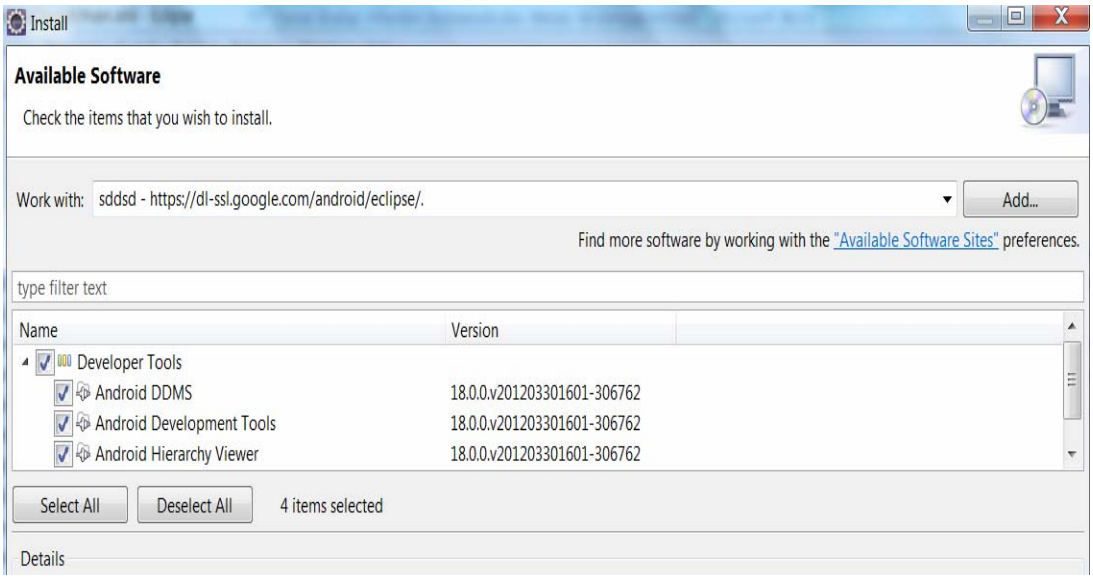
Ahora se iniciará Eclipse y una vez puesto en nos pedirá que se elija una ruta para almacenar los datos en nuestro lugar de trabajo



Ahora se accederá a la pestaña Help > Install New Software.

En la ventana que aparece, se seleccionará el botón *Add*, que se encuentra en la parte superior derecha.

En el recuadro que sale, en el apartado “Name” se escribirá el nombre que identificará al plugin ADT “*ADT Plugin*“, mientras que en “Location” se deberá fijar la siguiente dirección URL: https://dl-ssl.google.com/android/eclipse/.



A continuación tendremos que aceptar y seleccionar todas las herramientas, aceptar nuevamente y aceptar las condiciones de licencia.

Ahora se instalarán los paquetes. Si sale alguna advertencia de seguridad solo hace falta presionar Ok. Por último hara falta reiniciar Eclipse para implementar los cambios.

Configurar el plugin ADT para Eclipse

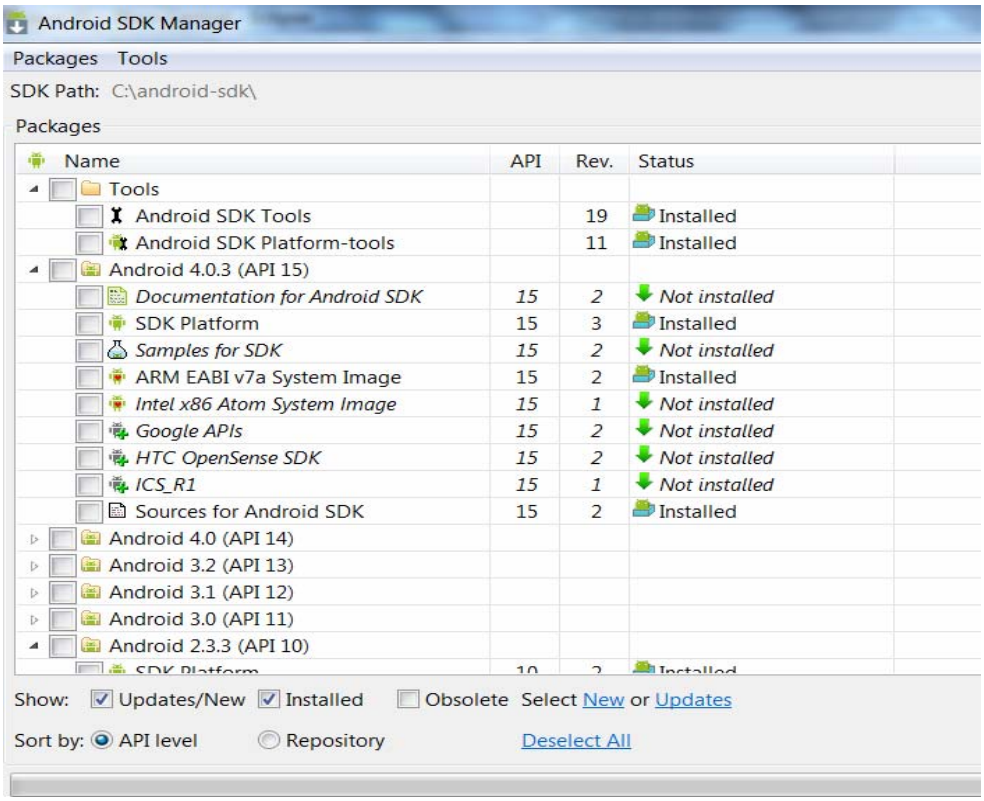
Para configurar el plugin ADT una vez se ha reiniciado Eclipse se debe de acceder a la pestaña Window > Preferences y seleccionar en el panel izquierdo Android.

Allí nos pedirá seleccionar los paths del Android SDK, por lo que seleccionaremos en Browse la ruta en donde se guarda el SDK , una vez hecho esto se acepta y se aplican los cambios.

Añadir plataformas Android

Ya solo queda añadir plataformas para poder ejecutar nuestras aplicaciones, es decir, las distintas versiones de la plataforma **Android** ejecutar las aplicaciones dependiendo de la versión .

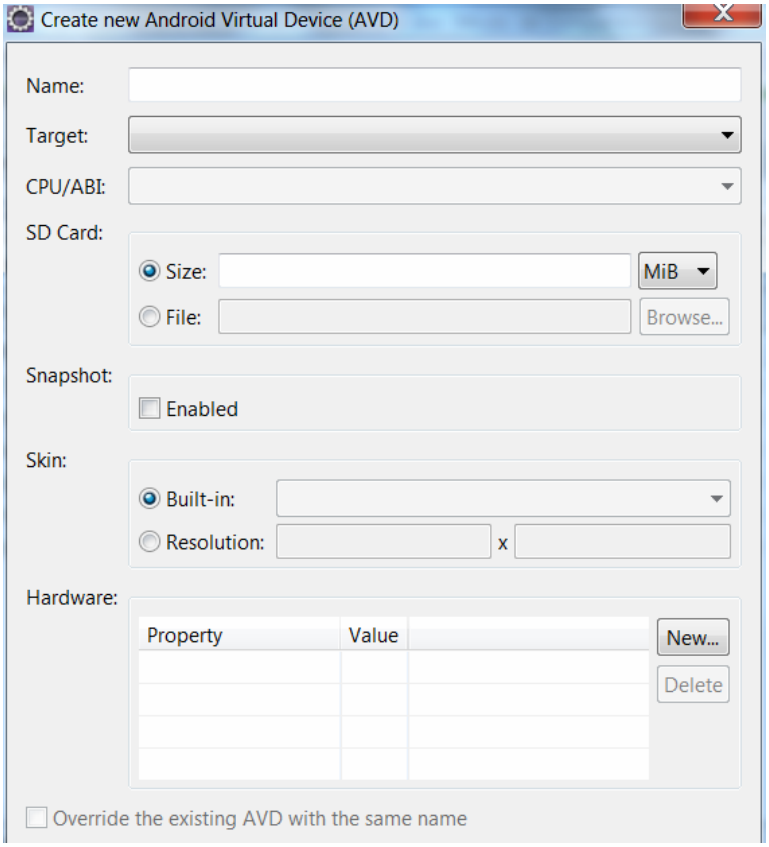
Se debe acceder a la pestaña de Windows > Android SDK Manager, aquí se escogerá los paquetes de las diferentes versiones de Android que se deseen instalar, en este proyecto se ha optado por la versión 2.2, una de las más estables de Android.



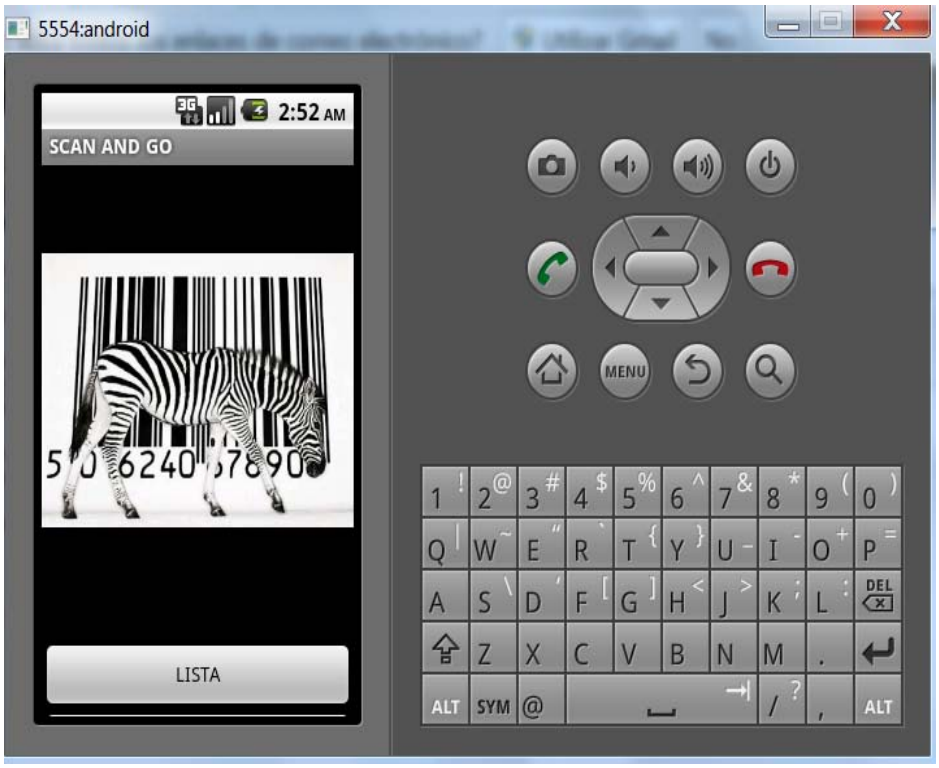
Una vez instalado las plataformas Andrid se debe configurar la máquina virtual para poder simular la aplicación en cuestión.

Para ello hay que acceder al Android Virtual Device Manager , botón que está situado al lado del SDK Manager que anteriormente se utilizó.

Habrá que seleccionar el botón de New para poder configurar una nueva máquina virtual.



Seleccionaremos un Nombre y lo más importante en Target, especificaremos la plataforma que queramos utilizar para nuestra máquina virtual, aceptaremos y procederemos a darle Start , después Launch y ya tendriamos lanzada la aplicación desde la máquina virtual de Android.

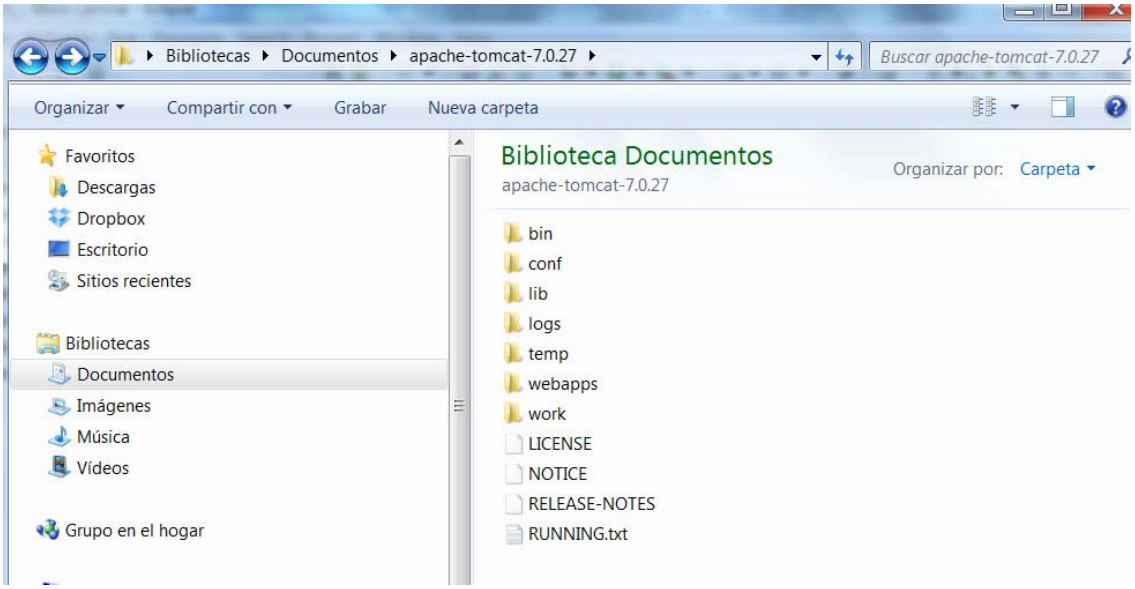

5554:android
2:52 AM
SCAN AND GO
LISTA
MENU

APACHE TOMCAT

La instalación de Apache Tomcat se podrá hacer accediendo al siguiente enlace y eligiendo la versión adecuada del sistema operativo, en este caso Windows 64 bits.

http://tomcat.apache.org/download-70.cgi

Una vez descomprimido el archivo , se obtendrá una carpeta resultante , pues bien, con esta carpeta ya tendriamos el Tomcat instalado, solamente haría falta ubicarla, en este caso , se ha optado por ubicarla en C:\Users\DANI\Documents.



Una vez instalado y ubicado en algún lugar, se procederá a la configuración del mismo de acuerdo con el proyecto.

En primer lugar, se modificará el fichero tomcat-users.xml ubicado en la carpeta conf, simplemente para darle al servidor permisos de acceso , es decir, un simple login(usuario y contraseña) añadiendo las siguientes líneas de código.

```
<role rolename="manager-gui"/>

<user username="dani" password="1234" roles="manager-gui" />
```

Ahora se agregará a la carpeta webapps el proyecto BarcodeServer que se realizó anteriormente en Eclipse.

Una cosa muy a tener en cuenta, es que cuando se modifica el código en eclipse del proyecto BarcodeServer hay que modificar en Tomcat las clases correspondientes a dicho proyecto que teniamos en Tomcat, para ello nos iriamos a C:\Users\DANI\Documents\WORKSPACE\BarCodeServer\build\classes\com\android\example\server y copiariamos las clases BarCodeServer y DB que son de extensión .class al directorio del proyecto del Tomcat donde tenemos estas clases, C:\Users\DANI\Documents\apache-tomcat-7.0.27\webapps\BarCodeServer\WEB-INF\classes\com\android\example\server, esto se hace para pasar la compilación de las clases llevadas a cabo en eclipse al servidor Tomcat.

Una vez se ha configurado el Tomcat, se procederá a la iniciación del mismo. Para ello, habrá que acceder a bin\startup.bat y ejecutar dicho archivo.

Para apagarlo habrá que acceder al mismo sitio pero ejecutar el archivo shutdown.bat en lugar del startup.bat.

BASE DE DATOS

Para insertar productos en la base de datos nuevos, lo que se deberá hacer es lo siguiente:

Previamente, lo que se ha hecho es crear una nueva configuración de Java para poder exportar el archivo jar de la base de datos, esto se hace mediante el eclipse en el menú Run>Run Configurations y aquí se crea una nueva configuración del tipo JavaApplication en lugar de AndroidApplication para el caso de los proyectos anteriores.

Una vez creada la nueva configuración, se hace clik con el botón derecho en el proyecto Android llamado BarcodeDB y seleccionamos la opción de Export, una vez dentro accederemos a la columna de la izquierda, abrimos la carpeta de java y luego seleccionamos la opción de Runnable Jar File, continuaremos seleccionando la configuración java creada anteriormente y seleccionaremos la ubicación donde se exportará ese archivo jar.

Se accede al WorksSpace del eclipse , se ubica uno en el proyecto BarCodeServer, una vez dentro, está creada una carpeta llamada imágenes que contendrá las imágenes de los productos que queramos agregar de forma que su nombre sea el código a agregar.

Aquí entonces, se ubicará una nueva foto del producto que se quiera añadir, después procederemos a abrir el terminal del pc (símbolo del sistema), nos ubicaremos en el sitio donde habiamos exportado el arhivo .jar referente a la base de datos de java, y se ejecutaría la siguiente sentencia de la siguiente forma :

```
java -jar BarcodeDb.jar (fichero_imagen) (codigo_producto) (nombre_producto) (precio_euros)
```

Entre paréntesis se ha puesto los campos que hay que rellenar para cada producto, el de fichero imagen por ejemplo ,hace falta seleccionar la ruta donde está ubicada la imagen que hemos puesto, como se dijo anteriormente, en la carpeta del proyecto BarcodeServer, dentro de la carpeta imágenes.

En los demás campos hay que meter el código del producto tal cual, el nombre que queramos darle al producto, y finalmente su precio.

Con todo esto, se añadiría finalmente el producto nuevo en la base de datos.

BARCODE SCANNER

Es necesario también instalar la aplicación Android de BarcodeScanner, que lo que nos permitirá es utilizar sus librerías zxing para leer los códigos de barras con la aplicación Scan&Go.

Para proceder a la instalación, solamente habrá que acceder al Play Store de Android , buscarla y descargarla totalmente gratis.

Anexo II. Contenido del CD-ROM

- Proyectos Android compilados en Eclipse

- Archivos extra para su correcto funcionamiento (librerías, etc.)

- Código de los proyectos unificado en un archivo de texto.

- Carpeta de Apache Tomcat con las modificaciones oportunas

Ingeniería Técnica en Telecomunicaciones especialidad Telemática

SCAN AND GO

Estudio económico

DANIEL BRAÑAS VILLARDÓN
PONIENTE: PERE BARBERAN

PRIMAVERA AÑO 2012

TecnoCampus
Mataró-Maresme

Índice

Presupuesto

Aquí se realiza el coste total del proyecto, teniendo en cuenta que lo que cobra un Ingeniero de Telecomunicaciones es aproximadamente de 40 euros por hora, se ha realizado los siguientes cálculos .

Apróximadamente la dedicación diaria de este proyecto ha sido de unas 3 horas y media, lo que conlleva a un total de unas 315 horas.

Desglose de actividades

Estudio previo	15 horas
Módulo Android	100 horas
Módulo Java	100 horas
Documentación	100 horas

Coste personal

	Personas	Precio por hora	Horas totales	Coste total
Estudio previo	1	40	15 horas	600 euros
Módulo Android	1	40	100 horas	400 euros
Módulo Java	1	40	100 horas	400 euros
Documentación	1	40	100 horas	400 euros
			315 horas	**1800 euros**

Coste material

	Coste
HTC desire	400 euros
Portátil Hacer X53series	600 euros
Eclipse	0 euros
Tomcat	0 euros
Barcode Scanner	0 euros
Herramientas ofimáticas	0 euros
	1000 euros

Coste total proyecto

Coste personal	1800 euros
Coste material	1000 euros
Coste total	2800 euros