UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INTELIGENCIA AMBIENTAL POR PROXIMIDAD PARA UN MUSEO, UTILIZANDO TECNOLOGÍA INALÁMBRICA”

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS CON MENCIÓN EN TELEMÁTICA

ELABORADO POR

GUMERCINDO BARTRA GARDINI

ASESOR

M. Sc. ARTURO VILCA ROMÁN

LIMA – PERÚ

2012

Con todo mi corazón dedico esta tesis a mi esposa Gaby e hijo Gabriel, que son la fuente de mi inspiración.

El autor.

Expreso mi agradecimiento a los docentes de la sección de posgrado de la Facultad de Ingeniería Eléctrica y Electrónica de la Universidad Nacional de Ingeniería, por el apoyo que me brindaron en el desarrollo de ésta tesis, y particularmente a mi asesor el M.Sc. Ing. Arturo Vilca Román, por su constante guía y aliento, y a los excelentes revisores: M.Sc. Ing. Fernando Saldaña y M.Sc. Ing. José Díaz.

También expreso mi agradecimiento a todas las personas que directa e indirectamente me dieron el soporte para la finalización de éste trabajo.

El autor

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INTELIGENCIA AMBIENTAL POR PROXIMIDAD PARA UN MUSEO, UTILIZANDO TECNOLOGÍA INALÁMBRICA

TESIS

PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS
MENCIÓN: TELEMÁTICA

PRESENTADA POR:

ING. GUMERCINDO BARTRA GARDINI

LIMA-PERU

RESUMEN

Los entornos inteligentes (inteligencia ambiental) son una línea de investigación que está recibiendo bastante atención en el entorno de las tecnologías de Información y Comunicaciones (TIC). Estos entornos agrupan a un conjunto de sensores (nano y micro), protocolos de comunicación inalámbrica como Bluetooth (IEEE 802.15) y WIFI (IEEE 802.11 b/g/n), que permiten a los usuarios acceder a sistemas de información a través de dispositivos móviles como teléfonos celulares y tabletas, que incluyen en un solo y reducido equipo, funciones de comunicación y procesamiento de datos multimedia cada vez más complejos.

Así mismo, es importante recalcar el uso de Android, patrocinado por Google, como sistema operativo de dispositivos móviles, basado en Linux por lo tanto de código abierto, lo que permite a los desarrolladores el uso del código fuente para crear potentes aplicaciones, es una excelente opción frente a otros sistemas como para terminales móviles como iOS (Apple), Windows Mobile (Microsoft), Symbian OS y Palm OS.

La presente tesis está enfocada al diseño e implementación de un sistema inalámbrico inteligente de corto alcance (por proximidad), que permita a los usuarios acceder a un sistema de información multimedia de un museo, mediante el uso de dispositivos portátiles con tecnología inalámbrica WIFI y una cámara que permite la lectura de código QR Code. Para ello se hace un análisis del estado del arte en tecnologías inalámbricas de corto alcance, códigos de barra por proximidad QR, tecnologías de información para el almacenamiento gestión y distribución de información multimedia, programación en Android, escenarios de prueba y análisis de costos de implementación.

UNIVERSIDAD NACIONAL DE INGENIERIA
FACULTAD DE INGENIERIA ELECTRICA Y ELECTRONICA

DESIGN AND IMPLEMENTATION OF ENVIRONMENTAL INTELLIGENCE SYSTEM FOR CLOSE TO A MUSEUM, USING WIRELESS TECHNOLOGY

THESIS

REQUIREMENT FOR THE DEGREE OF MASTER IN SCIENCE
MAJOR: TELEMÁTICA

BY:

ING. GUMERCINDO BARTRA GARDINI

LIMA-PERU

ABSTRACT

Intelligent environments (ambient intelligence) are a line of research that is receiving considerable attention in the environment of the Information and Communications Technologies (ICT). These environments bring together an array of sensors (nano and micro), wireless communication protocols such as Bluetooth (IEEE 802.15) and WIFI (IEEE 802.11 b / g / n), which allow users to access information systems across devices as mobile phones and tablets, which include in a single, small team, functions of communication and multimedia data processing increasingly complex.

It is also important to emphasize the use of Android, sponsored by Google, as a mobile OS based on Linux, which is Open Source, allowing developers to use source code to

create powerful applications; this is an excellent alternative to other systems for mobile devices such as iOS (Apple), Windows Mobile (Microsoft), Symbian OS and Palm OS.

This thesis is focused on the design and implementation of a smart wireless short-range (proximity), which allows users to access multimedia information system of a museum, using portable devices and wireless technologies, and a camera that allows reading QR Code. This is an analysis of the state of the art in short-range wireless technologies, bar codes, QR proximity, information technology for storing information management and distribution of multimedia, programming in Android, test scenarios and analysis of implementation costs.

CAPÍTULO I

INTRODUCCIÓN

1.1 Introducción

Cuando Internet aparece como un servicio abierto y la telefonía móvil como fenómeno sociológico, se produce en el mundo una revolución tecnológica donde la comunicación y el acceso a la información dan inicio a la ubiquidad, es decir acceso desde cualquier lugar y momento.

Del mismo modo, la miniaturización de los circuitos y componentes electrónicos, producen la reducción del tamaño de los terminales móviles y el incremento de sus prestaciones, en consecuencia, el teléfono móvil se convierte en un centro de comunicaciones para el envío y recepción de información como voz, mensajes de texto, imágenes, sonidos y vídeo; es desde ese momento que se inventa un nuevo término: dispositivo móvil.

Un teléfono móvil, un smartphone, una tableta, una cámara digital, un reproductor de audio o video y una consola de videojuego, se conocen como dispositivos móviles.

Estos dispositivos móviles, generalmente cuentan con un sistema operativo que se adapta a las especificaciones técnicas y restricciones de memoria, CPU y consumo de energía de los dispositivos móviles. Los sistemas operativos más usados son iOS (Apple), Android, Symbian y Windows Mobile [1].

Android, es más que un sistema operativo, es una pila de software para dispositivos móviles que incluye gran cantidad de drivers, gestor de bases de datos, aplicaciones esenciales y midleware. El SDK (Software Development Kit) de Android proporciona las herramientas y las API (Application Programming Interface) necesarias para el desarrollo de aplicaciones para dispositivos móviles con plataforma Android [1].

Se ha elegido la plataforma Android, porque aprovecha cada vez más, la capacidad de los dispositivos móviles, que incluyen componentes como pantallas táctiles, GPS, bluetooth, WIFI, USB, clientes de correo, aplicaciones ofimáticas o videojuegos. La

ventaja de Android con respecto a otros sistemas operativos, es que cada aplicación se ejecuta en su propia máquina virtual, donde el sistema administra eficientemente y en forma transparente para el usuario, los escasos recursos del dispositivo móvil, permitiendo la navegación entre varias aplicaciones abiertas.

El uso del sistema operativo Android y los dispositivos móviles con interfaz WIFI y audífonos, servirán para el desarrollo del proyecto para el museo, y está orientado a proporcionar un ambiente de interactividad donde la información solicitada por el visitante, estará disponible desde un teléfono celular o una tableta provista con cámara y lector de códigos QR, lo que le permitirá recibir información sobre la obra de su interés en formato a elegir, como texto, imágenes, sonidos y videos.

Además, Android está patrocinado por Google y está basado en la versión 2,6 del kernel de Linux y todas sus aplicaciones se escriben en lenguaje de programación JAVA, y utiliza además su propia máquina virtual conocida como DALVIK.

La conexión entre el dispositivo móvil y la base de datos de información se realizará mediante tecnología inalámbrica de banda ancha como IEEE 802.11 b/g/n (WIFI).

Si utilizamos una plataforma de red inalámbrica y el uso de dispositivos portátiles que soportan WIFI y escáner de QR barcode, podemos recibir información de interés desde nuestros celulares o dispositivos electrónicos portátiles. No se requiere conexión a Internet por lo que sus aplicaciones son ilimitadas en cualquier lugar del país.

1.2 Formulación del problema

El visitante a un museo tiene la necesidad de obtener información adicional sobre la obra de su interés, y no cuenta con un guía personal ó grupal, para ello, por lo que se hace necesario el uso de la tecnología para el acceso inmediato a la gran cantidad de información en el formato que el visitante al museo elija. Para ello el visitante al museo estará provisto de un dispositivo móvil como un teléfono celular, una tableta, con soporte de multimedia y audífonos para recibir la información solicitada en el formato que considere conveniente. El dispositivo móvil y los audífonos, serán proporcionados al visitante en calidad de préstamo por la administración del museo.

Es importante mencionar que el museo elegido, es una construcción antigua y no se permiten realizar canalizaciones para la red, así mismo en un solo ambiente existen varias obras por lo que debemos resolver el problema del acceso a la información

mediante conexiones inalámbricas que soporten una gran concurrencia o simultaneidad, por lo que planteamos el uso del estándar de redes inalámbricas WIFI, IEEE 802.11n.

Otra restricción planteada es que el visitante al museo no debe digitar en el dispositivo portátil código alguno para el acceso a la información adicional solicitada, por lo que se propone la lectura o escaneo rápido del código bidimensional “QR barcode”.

1.3 Trabajos previos

Los trabajos desarrollados sobre inteligencia ambiental y el uso de dispositivos portátiles mayormente se han realizado utilizando la plataforma Java Micro Edition (J2ME), sin embargo con el lanzamiento y uso de los Smartphone con plataforma Android, cuyo nucleo del sistema operativo es Linux, se pueden integrar aplicaciones de software libre a la plataforma de android de los “smartphones” y tabletas, de una manera efectiva.

1.4 Ojetivos de la tesis

1.4.1 Objetivo general

Diseñar e implementar un sistema inalámbrico inteligente de proximidad, que permita a los usuarios acceder a un sistema de información multimedia de un museo, mediante el uso de dispositivos portátiles como teléfonos celulares inteligentes y tabletas.

1.4.2 Objetivos específicos

- Diseñar e implementar el sistema de información centralizado, la misma que será distribuida entre los diferentes dispositivos móviles de acceso.
- Diseñar e implementar la interfaz de acceso al sistema de información, mediante el uso de un lector de QR Code instalado en el dispositivo móvil provisto de una cámara de video.
- Diseñar e implementar la red de transporte de alta velocidad para conectar los puntos de acceso distribuidos en el museo.
- Diseñar e implementar la red de acceso inalámbrico, con tecnología WIFI, que permitirá a los usuarios el acceso a los sistemas de información multimedia de su interés (texto, imagen, audio, video).

1.5 Organización de la tesis

En el capítulo 2 se realiza un análisis detallado de la tecnología inalámbrica de corto alcance como bluetooth y WIFI, indicando su estado actual, las ventajas, desventajas y las distintas aplicaciones. Así mismo se analiza el código de barras QR y sus diversas aplicaciones como pueden ser, museos, sala de exposiciones, información publicitaria, espacios libres en estacionamientos, supermercados, etc., y se describen las plataformas operativas de diversos dispositivos móviles indicando sus fortalezas, debilidades y penetración en el mercado.

En el capítulo 3 se analiza la arquitectura y los componentes de los lenguajes de programación más utilizados en dispositivos móviles basados en Java como J2ME y Android.

En el capítulo 4 se muestran los procedimientos del modelamiento, diseño y desarrollo del sistema.

En el capítulo 5 se realizan las pruebas de validación del sistema y el análisis de resultados obtenidos en las pruebas de desempeño. Así mismo se realiza un análisis de costos de implementación.

En el capítulo 6 se muestra un resumen de los resultados más relevantes, conclusiones y recomendaciones para futuros trabajos.

También, se presenta el listado de la bibliografía utilizada en el desarrollo de la presente tesis. Finalmente, en los anexos se describen el uso de los entornos de desarrollo integrado como Eclipse y el simulador de Android.; los programas desarrollados para la aplicación del sistema de inteligencia ambiental, los mismos que se instalan en el dispositivo móvil.

CAPÍTULO II

TECNOLOGÍA INALÁMBRICA POR PROXIMIDAD

2.1 Tecnologías inalámbricas de corto alcance

El museo requiere brindar información multimedia, a solicitud del visitante dotado de un dispositivo móvil con interface de red inalámbrica y audífonos.

Los equipos móviles como teléfonos celulares y tabletas, soportan tecnologías inalámbricas como Bluetooth y WIFI. Por lo que en la presente tesis, elegiremos la tecnología IEEE 802.11 (WIFI) por un tema de mayor velocidad de transferencia de datos y alta concurrencia de usuarios por Punto de Acceso. El estándar IEEE 802.11, define que se pueden realizar entre 12 a 15 conexiones por cada Punto de Acceso (Access Point. Adicionalmente se pueden acomodar 3 canales simultáneamente en cada ambiente. La cobertura es mayor y el ancho de banda puede llegar hasta 108 Mbps. Mientras que el estándar IEEE 802.15 (Bluetooth), define que se pueden realizar como máximo 7 conexiones concurrentes; por lo que en un museo, por la alta concurrencia de visitantes, el uso de Bluetooth haría inviable el proyecto por su alto costo.

2.1.1 Bluetooth [2]

La tecnología Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs: IEEE 802.15) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Se diseñó pensando básicamente en tres objetivos: pequeño tamaño, mínimo consumo y bajo precio.

Los dispositivos que implementan Bluetooth pueden comunicarse entre ellos cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en

habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

Tabla 2.1. Clasificación de dispositivos Bluetooth por potencia [2]

clase	potencia máxima permitida (mW)	potencia máxima permitida (dBm)	rango aproximado
clase 1	100 mW	20 dBm	~ 100 metros
clase 2	2.5 mW	4 dBm	~ 10 metros
clase 3	1 mW	0 dBm	~ 1 metro

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte la mayor sensibilidad del dispositivo de clase 1 permite recibir la señal del otro pese a ser más débil.

Los dispositivos con Bluetooth también pueden clasificarse según su ancho de banda:

Tabla 2.2. Clasificación de dispositivos Bluetooth por ancho de banda [2]

versión	velocidad de transmisión
versión 1.2	1 Mbps
versión 2.0 + EDR	3 Mbps
versión 3.0 + HS	24 Mbps
versión 4.0	24 Mbps

2.1.1.1 Características de transmisión de Bluetooth

La especificación principal de Bluetooth (denominada core) define el nivel físico (PHY) y el control de acceso al medio (MAC) de una red inalámbrica de área personal.

2.1.1.2 Salto de frecuencia

Debido a que la banda ISM está abierta a cualquiera, el sistema de radio Bluetooth deberá estar preparado para evitar las múltiples interferencias que se pudieran producir. Éstas pueden ser evitadas utilizando un sistema que busque una parte no utilizada del espectro o un sistema de salto de frecuencia.

En este caso la técnica de salto de frecuencia es aplicada a una alta velocidad y una corta longitud de los paquetes (1600 saltos/segundo). Con este sistema se divide la banda de frecuencia en varios canales de salto, donde, los transceptores, durante la conexión van cambiando de uno a otro canal de salto de manera pseudo-aleatoria.

Los paquetes de datos están protegidos por un esquema ARQ (repetición automática de consulta), en el cual los paquetes perdidos son automáticamente retransmitidos.

2.1.1.3 El canal

Bluetooth utiliza un sistema FH/TDD (salto de frecuencia/división de tiempo duplex), que divide el canal en intervalos de 625 μs, llamados slots; éste mecanismo evita la interferencia, realizando el cambio inmediato de slot, de allí el nombre de salto de frecuencia.

Dos o más unidades Bluetooth pueden compartir el mismo canal dentro de una piconet (pequeña red que establecen automáticamente los terminales Bluetooth para comunicarse entre si), donde una unidad actúa como maestra, controlando el tráfico de datos en la piconet que se genera entre las demás unidades, donde éstas actúan como esclavas, enviando y recibiendo señales hacia el maestro.

El salto de frecuencia del canal está determinado por la secuencia de la señal, es decir, el orden en que llegan los saltos y por la fase de esta secuencia. En Bluetooth, la secuencia queda fijada por la identidad de la unidad maestra de la piconet (un código único para cada equipo), y por su frecuencia de reloj.

2.1.1.4 Estructura de la trama IEEE 802.15: Bluetooth

La información que se intercambia entre dos unidades Bluetooth se realiza mediante un conjunto de slots que forman un paquete de datos. Cada paquete comienza con un código de acceso de 72 bits, que se deriva de la identidad maestra, seguido de un paquete de datos de cabecera de 54 bits. Éste contiene importante información de

control, como tres bits de acceso de dirección, tipo de paquete, bits de control de flujo, bits para la retransmisión automática de la pregunta, y chequeo de errores de campos de cabecera. La dirección del dispositivo es en forma hexadecimal. Finalmente, el paquete que contiene la información, que puede seguir al de la cabecera, tiene una longitud de 0 a 2745 bits.

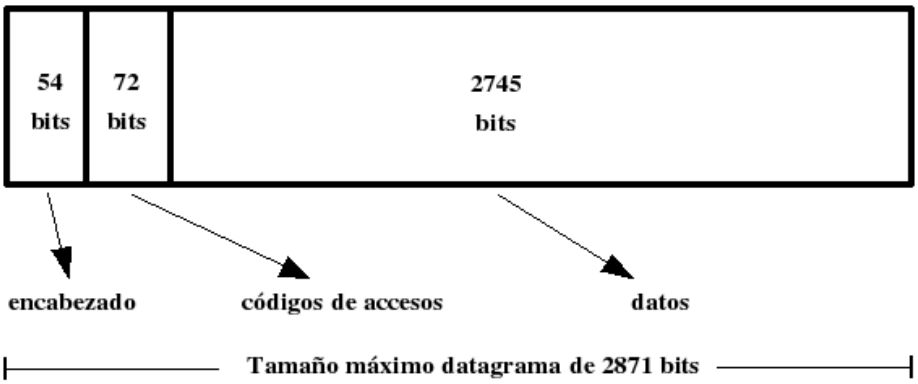


Fig. 2.1. Datagrama Bluetooth [3]

En cualquier caso, cada paquete que se intercambia en el canal está precedido por el código de acceso. Los receptores de la piconet comparan las señales que reciben con el código de acceso, si éstas no coinciden, el paquete recibido no es considerado como válido en el canal y el resto de su contenido es ignorado.

2.1.1.5 Piconets

Como hemos citado anteriormente si un equipo se encuentra dentro del radio de cobertura de otro, éstos pueden establecer conexión entre ellos. Cada dispositivo tiene una dirección única de 48 bits, basada en el estándar IEEE 802.11 para WLAN. En principio sólo son necesarias un par de unidades con las mismas características de hardware para establecer un enlace. Dos o más unidades Bluetooth que comparten un mismo canal forman una piconet

Para regular el tráfico en el canal, una de las unidades participantes se convertirá en maestra, pero por definición, la unidad que establece la piconet asume éste papel y todos los demás serán esclavos. Los participantes podrían intercambiar los papeles si una unidad esclava quisiera asumir el papel de maestra. Sin embargo sólo puede haber un maestro en la piconet al mismo tiempo. Hasta ocho usuarios o dispositivos pueden formar una piconet y hasta diez piconets pueden coexistir en una misma área de cobertura.

2.1.1.6 Scatternet

Los equipos que comparten un mismo canal sólo pueden utilizar una parte de su capacidad. Aunque los canales tienen un ancho de banda de un 1 Mbps, cuantos más usuarios se incorporan a la piconet, disminuye la capacidad hasta unos 10 Kbps más o menos. Para poder solucionar este problema se adoptó una solución de la que nace el concepto de scatternet.

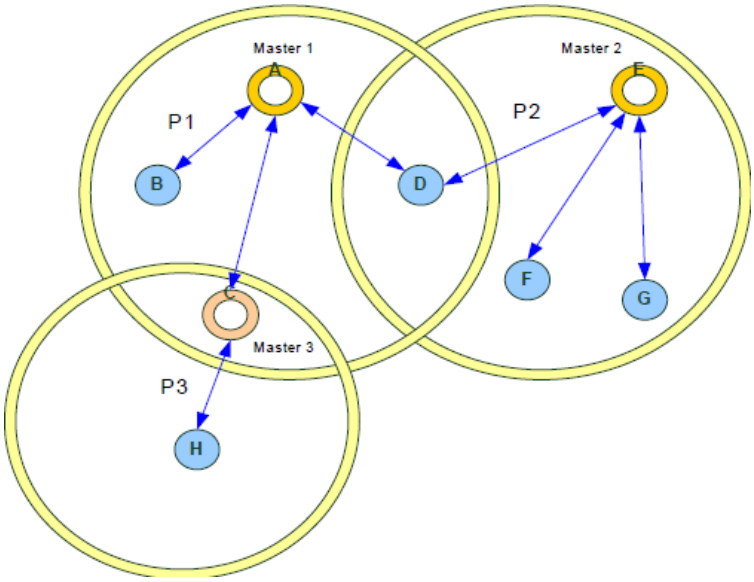


Fig. 2.2. Piconet y Scatternet [4]

Las unidades que se encuentran en el mismo radio de cobertura pueden establecer potencialmente comunicaciones entre ellas. Sin embargo, sólo aquellas unidades que realmente quieran intercambiar información comparten un mismo canal creando la piconet. Este hecho permite que se creen varias piconets en áreas de cobertura superpuestas.

A un grupo de piconets se le llama scatternet. El rendimiento, en conjunto e individualmente de los usuarios de una scatternet es mayor que el que tiene cada usuario cuando participa en un mismo canal de 1 Mbps. Además, estadísticamente se obtienen ganancias por multiplexación y rechazo de canales salto. Debido a que individualmente cada piconet tiene un salto de frecuencia diferente, diferentes piconets pueden usar simultáneamente diferentes canales de salto.

2.1.1.7 Conexiones concurrentes en Bluetooth

El estándar IEEE 802.15, define que se pueden realizar 7 conexiones concurrentes en bluetooth. Sin embargo, el módulo de bluetooth en la práctica, balancea el ancho de banda entre las conexiones abiertas, lo que hace técnicamente difícil llegar a éste número máximo, primero por que los móviles con bluetooth, no buscan mientras envían datos y segundo si se envía por algunos canales, el módulo no puede abrir más canales mientras ve que se está consumiendo el ancho de banda disponible.
El ancho de banda disponible, el número de dispositivos concurrentes y el costo hace inviable el uso de la tecnología Bluetooth en el proyecto del museo.

2.1.2 Redes LAN Inalámbricas: WIFI

El estándar IEEE 802.11, define el uso de los dos niveles inferiores de la arquitectura OSI (capas física y de enlace de datos), especificando sus normas de funcionamiento en una WLAN. Los protocolos de la rama 802.X definen la tecnología de redes de área local y redes de área metropolitana.

2.1.2.1 Conceptos Generales

- **Estaciones:** computadores ó dispositivos móviles con interfaz inalámbrica.
- **Medio**: se pueden definir dos: la radiofrecuencia y los infrarrojos.
- **Punto de Acceso (AP):** tiene las funciones de un puente (bridge), es decir conecta dos redes con niveles de enlace parecido o distinto; y realiza por tanto las conversiones de trama pertinentes.
- **Sistema de distribución**: proporcionan movilidad entre AP, para tramas entre distintos puntos de acceso o con los terminales móviles.
- **Conjunto de servicio básico (BSS)**: grupo de estaciones que se comunican entre ellas. Se definen dos tipos:
    - **Independientes**: cuando las estaciones se comunican directamente entre sí.
    - **Infraestructura**: cuando se comunican a través de un Punto de Acceso (AP).
- **Conjunto de servicio extendido (ESS): es la unión de varioas BSS**

- **Área de servicio básico (BSA):** es la zona donde se comunican las estaciones de una misma BSS.
- **Movilidad**: este es un concepto importante en las redes 802.11, ya que lo que indica es la capacidad de cambiar la ubicación de los terminales, variando la BSS.

La transición será correcta si se realiza dentro del mismo ESS en otro caso no se podrá realizar.

- **Límites de la red**: los límites de las redes 802.11 son difusos ya que pueden solaparse diferentes BSS.
- **IFS:** Interframe space o espacio entre tramas

2.1.2.2 Asignación de bandas WIFI [5]

Los canales permitidos no son los mismos en cada país. Esto dificulta la creación de equipos de tecnología globalizada. En USA la FCC(Comisión Federal de Comunicaciones) y sus contra partes fuera de los Estados Unidos tienen separadas un conjunto de anchos de bandas para uso no regulado, en la llamada Banda ISM (Industrial, Scientific and Medical). Se determinó la creación de 14 canales con separaciones de 5 MHz.

Tabla 2.3. Asignación de canales utilizados para la banda de 2.4 GHz [5]

relación entre canal y frecuencia	
canal	**frecuencia**
1	2,412 GHz
2	2,417 GHz
3	2,422 GHz
4	2,427 GHz
5	2,432 GHz
6	2,437 GHz
7	2,442 GHz
8	2,447 GHz
9	2,452 GHz
10	2,457 GHz
11	2,462 GHz
12	2,467 GHz
13	2,472 GHz
14	2,477 GHz

Tabla. 2.4. Asignación de canales por paises para la banda de 2.4 GHz [5]

países y canales	
países	canales
Europa (ETSI)	1 - 13
USA (FCC)	1 - 11
Japón	1 - 14

2.1.2.3 Control de Acceso al medio [5]

La subcapa MAC 802.11 es la responsable de la administración y control del medio inalámbrico, sus funciones se especifican en términos de coordinación. La Función de Coordinación Distribuida (DCF) permite que los equipos transmiten en la modalidad de mejor esfuerzo, es decir, todas luchan por transmitir lo cual origina periodos de contención, similar a Ethernet. DCF está basada en el protocolo CSMA-CA o MACA.

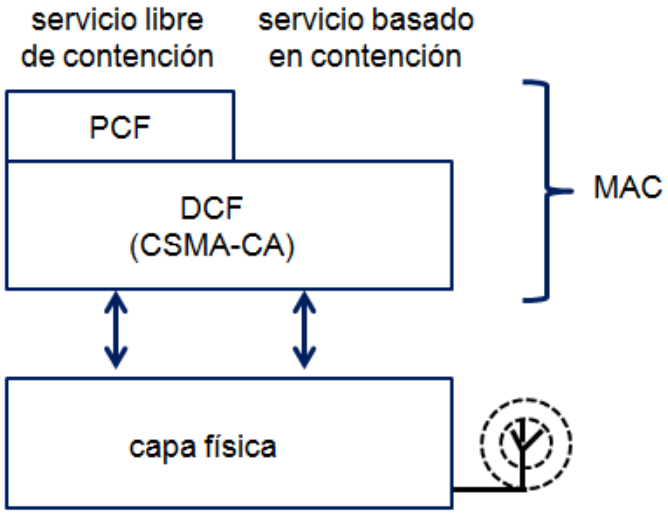


Fig. 2.3. Control de acceso al medio IEEE 802.11 [5]

Para servicios con límite de tiempo, por ejemplo vídeo y voz, existe una alternativa al CSMA/CA o DCF, es la funcionalidad opcional llamada Función de Punto de Coordinación (PCF), que utiliza un acceso por orden de prioridad, así el AP controla el acceso al medio y emite peticiones de sondeo a las estaciones para transmitir datos. El punto de acceso sondeará cada estación en busca de datos, y después de un tiempo cambia a la siguiente estación. Ninguna estación puede transmitir hasta que sea elegida,

y las estaciones reciben datos del punto de acceso solamente cuando sean elegidas, por lo que el PCF da a cada estación un turno para transmitir en un momento predeterminado, garantizando un retraso o latencia máxima. El hecho de un AP tenga el control de acceso al medio y sondee las estaciones hace que no sea eficaz para redes grandes.

2.1.2.4 Protocolo CSMA/CA [5]

802.11 usa un protocolo un poco modificado del CSMA/CD que es conocido como CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance), o como el DCF (Distributed Coordination Function). CSMA/CA intenta evitar las colisiones usando ACK explícitos, es decir que la estación que recibe los datos envía un paquete ACK si éstos han llegado correctamente. Si no se utilizaran ACKs explícitos cabría la posibilidad de que varias estaciones estuvieran sondeando el canal y al detectar que está libre simultáneamente intentarán transmitir al mismo tiempo, provocando colisión. Se establece también el uso de un campo "duración" en la trama enviada, todos los equipos al recibir la trama leen éste campo y determinan cuanto tiempo (en μs) deben esperar antes de intentar transmitir

El protocolo CSMA/CA funciona de la siguiente manera:
Una estación que desee transmitir mira si el medio esta ocupado, si el canal esta libre entonces está autorizada a transmitir, espera un tiempo aleatorio por seguridad y entonces transmite al medio si éste continúa libre. Si por el contrario el canal esta ocupado, la estación dejará la transmisión para más tarde.
Si el paquete transmitido se recibe correctamente (se comprueba el CRC), la estación receptora envía un ACK.

Si la estación emisora recibe el ACK se completa el proceso. Si el ACK no es detectado por la estación emisora, porque el paquete original no ha sido recibido correctamente ó porque el ACK se ha perdido, se asume que se ha producido una colisión y el paquete de datos se retransmite de nuevo después de esperar otro tiempo aleatorio.
Si el medio ha estado libre durante un intervalo de tiempo DIFS (DCF IFS) entonces se transmite el paquete de datos. Una vez recibido, el receptor enviará una confirmación de recepción (ACK) despues de transcurrido un tiempo SIFS (Short IFS). Si el transmisor ha encontrado el medio ocupado, espera a que se acabe la transmisión actual, y cuando vuelva a intentar transmitir tendrá que esperar el tiempo DIFS, más un tiempo de contención (back-off) pseudoaleatorio.

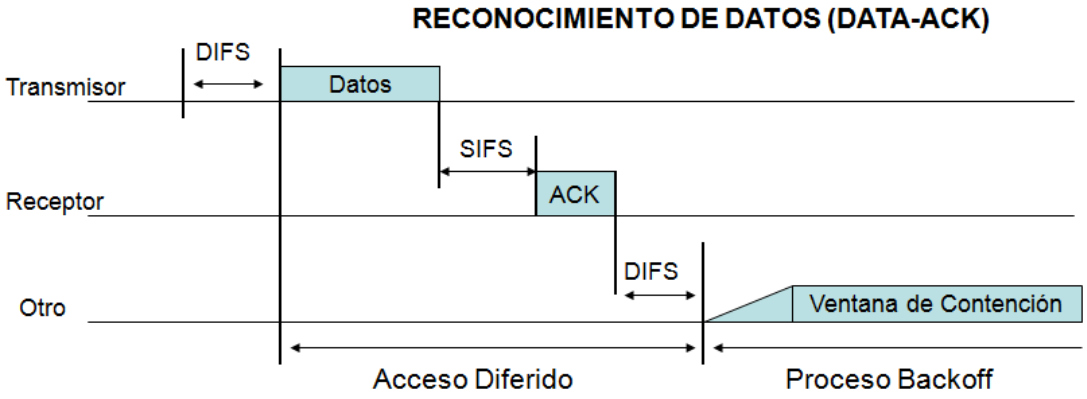


Fig. 2.4. Protocolo CSMA/CA IEEE 802.11 [5]

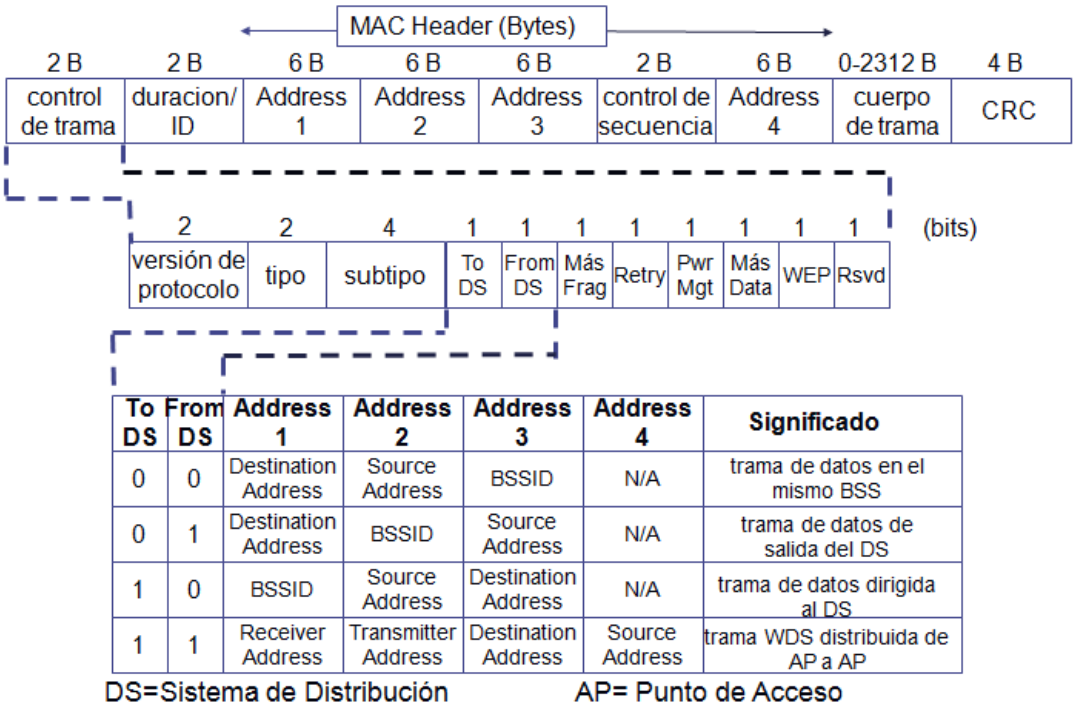
2.1.2.5 Estructura de la trama IEEE 802.11: WIFI [5]

El campo **control de trama** es de 16 bits. Tenemos el campo **versión de protocolo**, y luego el campo **tipo** que indica si la trama es de gestión, control o datos. Según el valor del campo **To** (Hacia) DS y el campo **From** (De) DS (Distribution System) recién toman significado los campos Dirección 1, 2, 3 y 4 de la trama 802.11, direcciones que son de tipo MAC (Dirección física ó Media Access Control). En general representan si el paquete va o viene en ese momento de un Access Point respectivamente. Si ambos bits están desactivados representa una comunicación entre equipos en el mismo BSS (Basic Service Set), si ambos bits están activados representa una comunicación en un WDS (Wireless Distribution System), a través de dos AP (Access Point - Receiver y Transmitter). El campo dirección 2 representa el equipo que debe recibir el mensaje ACK de confirmación de recepción de trama.

El bit **más fragmentos (More Fragments)**, indica fragmentación de la información original. El campo **retry** permite marcar e identificar tramas enviadas por segunda o más veces. El bit power managment activa el modo de ahorro de energía. El campo **más datos,** indica que existen almacenados en el AP más tramas por enviar. El campo WEP (Wired Equivalent Privacy), indica que la zona de datos está encriptada. El campo **control de secuencia** es de 16 bits, permite identificar la secuencia de entrega de fragmentos, sólo es válido si está activado el bit **más fragmentos**.

El campo **cuerpo de trama** traslada la información de gestión, control o datos según lo definido en el campo **tipo** especificado en el campo **control de trama**.

El campo CRC (comprobación de redundancia cíclica) es una secuencia de 32 bits calculada en base a la cabecera MAC y el **cuerpo de trama**.



To DS	From DS	Address 1	Address 2	Address 3	Address 4	Significado
0	0	Destination Address	Source Address	BSSID	N/A	trama de datos en el mismo BSS
0	1	Destination Address	BSSID	Source Address	N/A	trama de datos de salida del DS
1	0	BSSID	Source Address	Destination Address	N/A	trama de datos dirigida al DS
1	1	Receiver Address	Transmitter Address	Destination Address	Source Address	trama WDS distribuida de AP a AP

DS=Sistema de Distribución AP= Punto de Acceso

Fig. 2.5. Estructura de las tramas IEEE 802.11 [5]

2.1.2.6 Conexiones concurrentes en WIFI

El estándar IEEE 802.11, define que se pueden realizar entre 12 a 15 conexiones por cada Punto de Acceso (Access Point). Adicionalmente se pueden acomodar 3 canales simultáneamente en cada ambiente. La cobertura es mayor y el ancho de banda puede llegar hasta 108 Mbps.

El ancho de banda disponible, el número de dispositivos concurrentes y el costo, hace muy atractivo el uso de la tecnología WIFI en el proyecto del museo.

2.1.2.7 Protocolos IEEE 802.11 [5].

2.1.2.7.1 IEEE 802.11b.

La revisión 802.11b del estándar original fue ratificada en el año 1999, el IEEE. 802.11b tiene una velocidad máxima de transmisión de 11 Mbps y utiliza el mismo método de acceso definido en el estándar original CSMA/CA. El estándar 802.11b funciona en la

banda de 2,4 GHz, y debido al espacio ocupado por la codificación del protocolo CSMA/CA, en la práctica, la velocidad máxima de transmisión con este estándar es de aproximadamente 5,9 Mbps sobre TCP y 7,1 Mbps sobre UDP.

2.1.2.7.2 IEEE 802.11g.

En junio del 2003, se ratificó un tercer estándar de modulación: 802.11g, que es la evolución del estándar 802.11b, Este utiliza la banda de 2,4 GHz (al igual que el estándar 802.11b) pero opera a una velocidad teórica máxima de 54 Mbps, que en promedio es de 22,0 Mbps de velocidad real de transferencia, similar a la del estándar 802.11a. Es compatible con el estándar IEEE 802.11b y utiliza las mismas frecuencias.

Los equipos que trabajan bajo el estándar IEEE 802.11g llegaron al mercado muy rápidamente, incluso antes de su ratificación que fue dada aproximadamente, el 20 de junio del 2003. Esto se debió en parte a que para construir equipos bajo este nuevo estándar se podían adaptar los ya diseñados para el estándar IEEE 802.11b.

Actualmente se venden equipos con esta especificación, con potencias de hasta 0,5 Watts de potencia, que permite hacer comunicaciones de hasta 50 km con antenas direccionales tipo parabólicas y equipos de radio apropiados.

2.1.2.7.3 Interacción de IEEE 802.11g y IEEE 802.11b.

IEEE 802.11g tiene la ventaja de poder coexistir con los estándares IEEE 802.11a y IEEE 802.11b, esto debido a que puede operar con las Tecnologías RF DSSS (Direct Sequence Spread Spectrum) y OFDM (Orthogonal Frequency Division Multipexing). Sin embargo, si se utiliza para implementar usuarios que trabajen con el estándar IEEE 802.11b, el rendimiento de la celda inalámbrica se verá afectado por ellos, permitiendo solo una velocidad de transmisión de 22 Mbps. Esta degradación se debe a que los clientes IEEE 802.11b no comprenden OFDM.

2.1.2.7.4 IEEE 802.11n

En enero del 2004, el IEEE anunció la formación de un grupo de trabajo 802.11 (Tgn) para desarrollar una nueva revisión del estándar IEEE 802.11. La velocidad real de transmisión podría llegar a los 600 Mbps (lo que significa que las velocidades teóricas de transmisión serían aún mayores), y debería ser hasta 10 veces más rápida que una red bajo los estándares IEEE 802.11a y IEEE 802.11g, y unas 40 veces más rápida que una

red bajo el estándar IEEE 802.11b. También se espera que el alcance de operación de las redes sea mayor con este nuevo estándar gracias a la tecnología MIMO (Multiple Input – Multiple Output), que permite utilizar varios canales a la vez para enviar y recibir datos gracias a la incorporación de varias antenas [5]. Existen también otras propuestas alternativas que podrán ser consideradas. El estándar ya está redactado, y se viene implementando desde el año 2008.

A principios del año 2007, se aprobó el segundo boceto del estándar. Anteriormente ya había dispositivos adelantados al protocolo y que ofrecían de forma no oficial este estándar (con la promesa de actualizaciones para cumplir el estándar cuando el definitivo estuviera aprobado). Ha sufrido una serie de retrasos y el último lo lleva hasta noviembre de 2009. Habiéndose aprobado en enero de 2009 el proyecto 7.0 y que va por buen camino para cumplir las fechas señaladas. A diferencia de las otras versiones de WIFI, el IEEE 802.11n puede trabajar en dos bandas de frecuencias: 2,4 GHz (la que emplean 802.11b y 802.11g) y 5 GHz (la que usa 802.11a). Gracias a ello, el IEEE 802.11n es compatible con dispositivos basados en todas las ediciones anteriores de WIFI y además, es útil, que trabaje en la banda de 5 GHz, ya que está menos congestionada. En resumen, el IEEE 802.11n permite alcanzar un mayor rendimiento.

El estándar 802.11n fue ratificado por la organización IEEE el 11 de septiembre del año 2009 con una velocidad de 600 Mbps en capa física.

Actualmente ya existen varios productos que cumplen el estándar N con un máximo de 300 Mbps.

El estándar IEEE 802.11n hace uso simultáneo de ambas bandas, 2,4 GHz y 5,4 GHz. Las redes que trabajan bajo los estándares IEEE 802.11b y IEEE 802.11g, tras la reciente ratificación del estándar, se empiezan a fabricar de forma masiva y es objeto de promociones por parte de los distintos ISP, de forma que la masificación de la citada tecnología parece estar en camino. Todas las versiones de 802.11, aportan la ventaja de ser compatibles entre sí, de forma que el usuario no necesitará nada más que su adaptador WIFI integrado, para poder conectarse a la red.

Sin duda esta es la principal ventaja que diferencia WIFI de otras tecnologías propietarias, como LTE, UMTS y WIMAX, las tres tecnologías mencionadas, únicamente están accesibles a los usuarios mediante la suscripción a los servicios de un operador que está autorizado para uso de espectro radioeléctrico, mediante concesión de ámbito

nacional. La mayor parte de los fabricantes ya incorpora a sus líneas de producción equipos WIFI 802.11n, por este motivo la oferta ADSL, ya suele venir acompañada de WIFI IEEE 802.11n, como novedad en el mercado de usuarios domésticos.

Se conoce que el futuro estándar sustituto de IEEE 802.11n será IEEE 802.11ac con tasas de transferencia superiores a 1 Gbps.[6].

2.2 Código de Respuesta Rápida (Quick Response Barcode) [7]

El código de Respuesta Rápida (QR Code o Quick Response Code), es un estándar Iinternacional (ISO/IEC 18004) de "código de barras bidimensional" (matricial); creado en Japón en el año 1994 por la empresa Denso Wave Inc. Esta empresa japonesa distribuye las especificaciones del mismo, de manera libre y aunque posee una patente sobre el QR Code, no ejerce los derechos sobre la misma. Esto ha permitido la proliferación de lectores QR Code gratuitos y su incorporación en dispositivos móviles; puesto que sólo necesitan tener una cámara de fotos para la captura de los códigos QR y una aplicación para descifrar la información contenida en los mismos.

La característica predominante de la mayoría de aplicaciones, es la utilización de los QR Codes como "almacén" de información, que al ser decodificada mediante un dispositivo lector, redirige a una página web que el usuario puede consultar a través de su dispositivo móvil.

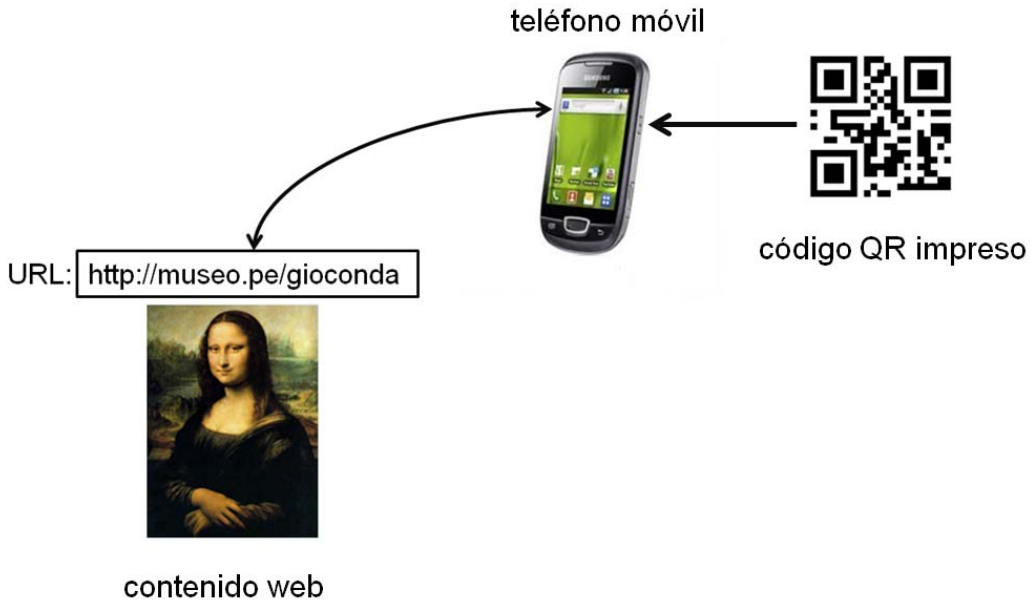


Fig. 2.6. Código QR y Teléfono Móvil [7]

El QR Code almacena información en ambas direcciones (verticalmente y horizontalmente) a diferencia de los tradicionales códigos de barra (de una dimensión), que tan sólo son capaces de almacenar información en una dirección. Precisamente por este motivo, la capacidad de almacenamiento es mayor en el caso del QR Code (es posible almacenar 7089 caracteres numéricos, 4296 códigos alfanuméricos o 2953 Bytes).

La unidad de información de un código unidimensional es la barra, en uno bidimensional es el módulo (cuadradito).

Los códigos QR se caracterizan por los tres cuadrados que encontramos en las esquinas (superior izquierda, superior derecha e inferior izquierda) y que permiten detectar al lector la posición de QR-Code para una lectura de alta velocidad desde todas las orientaciones (360°).

Fig. 2.7 Código QR y Código de Barras [7]

Adicionalmente, Los QR Codes tienen la capacidad de corregir errores. Se pueden restaurar los datos si parte del código está dañado o manchado. Existen varios niveles de corrección de errores, pudiendo llegar a restaurar hasta el 30% de la información perdida debido a la suciedad ó deterioro del código, etc. El sistema de corrección de errores se basa en Reed Solomon, cuyo codificador toma un bloque de información y le añade bits redundantes. Si consideramos “s” bits por símbolo, “n” Bytes, el tamaño del mensaje total (datos + paridad), “k” el tamaño del mensaje de datos; entonces la capacidad de corrección de errores “t” viene expresada como (n-k)/2. Por ejemplo si usamos Reed-Solomon RS (255,223), con símbolos s = 8 bits, n = 255 Bytes, k=223 Bytes, t=(255-223)/2 = 16, entonces la paridad “2t” = 32 bits. Un decodificador Reed-Solomon puede corregir “t” bits errados y “2t” bits borrados. Si 2e + r < 2t, (“e” errores, “r” borrados), entonces la palabra de código original transmitida puede ser siempre recuperada.

Existen dos estándares de los QR Code, el japonés JIS X 0510, creado por la JIS y distribuido en enero de 1999; y el correspondiente estándar de la ISO (International Standard Organization) junto con la IEC (International Electrotechnical Commision) , ISO/IEC 18004 aprobado en junio de 2000 y revisado en 2006 (ISO/IEC 18004:2006).

La terminología utilizada por el estándar QR Code para describir la estructura de un símbolo QR Code, aplicable a la versión 2, se describe a continuación:

Patrón de alineamiento*:*
Patrón de referencia fijada en algunas posiciones del símbolo matricial que permite al software decodificador, resincronizar las coordenadas de mapeo de la imagen QR Code ante posibles distorsiones moderadas de la imagen.
Indicador de contador de caracteres:

Secuencia de bits que define el número de caracteres que contiene una cadena de datos en un modo.

Interpretación del canal extendido (ECI):

Protocolo usado en algunos símbolos y que permiten interpretar una cadena de datos de forma diferente a una cadena por default.

Designador ECI:

Número de seis dígitos que identifica un ECI específico asignado al símbolo.

Región de codificación*:*

Región del símbolo no ocupada por patrones de función y sí por codewords de datos y de corrección de errores, y también por la información de formato y versión.

Patrón de Extensión*:*

Utilizado en símbolos del Modelo 1, es un patrón de función que no codifica datos.

Información de formato:

Patrón codificado que contiene información sobre el grado de corrección de errores con el que se han codificado los datos de la región de codificación y el tipo de máscara que se les ha aplicado.

Patrón de función*:*

Componente de la cabecera del símbolo utilizado para identificar y localizar el símbolo para su decodificación. Partes del símbolo que no contiene los datos codificados, sino información necesaria para la decodificación de éstos. Los patrones de función son: patrón de localización, separador, patrón de alineamiento y patrón temporizador.

Enmascaramiento de los datos*:*

Proceso en el cual, a los módulos de la región de codificación, se les realiza una operación XOR con un patrón máscara. Esto sirve para aumentar la diferenciación entre módulos blancos y negros y así mejorar la decodificación, reduciendo la ocurrencia de patrones que puedan interferir con el procesamiento rápido de la imagen.

Patrón de referencia de Máscara*:*

Identificador de tres bits del patrón de enmascaramiento aplicado al símbolo.

Modo:

Forma en que se representa un conjunto de datos como cadena de bits. La cadena de bits puede representar caracteres alfanuméricos, numéricos, Bytes, Kanji o cualquier otro que se defina en el futuro.

Indicador de modo:

Identificador de 4 bits que indica en qué modo está codificada la secuencia de bits que le sigue.

Bit de relleno:

Bits ceros, no representan datos y es usado para rellenar el codeword final después del terminador en una cadena de bits de datos.

Patrón de detección de posición (localizador*)*:

Patrón de función que existe por triplicado en el símbolo, situado en las esquinas superiores y la inferior izquierda. Sirven para calcular la orientación rotacional del símbolo.

Bits restantes:

Bits ceros, no representan datos y es usado para llenar posiciones de la región de codificación donde el espacio no se divide exactamente en 8 bits.

Codeword:

Conjunto de 8 módulos que puede tener diferentes formas dependiendo de su localización en el símbolo y que sirve para almacenar información codificada de los datos o de la corrección de errores.

Codeword restante:

Codeword de relleno para llenar posiciones sin codeword asignado para completar la capacidad total del símbolo. Va detrás de los codewords de corrección de error.

Segmento:

Secuencia de datos codificados de acuerdo a las reglas del modo de codificación ó por un modo de interpretación del canal extendido (ECI).

Separador:

Patrón de función formado por módulos blancos, cuyo ancho es de un módulo y que separa los patrones localizadores del resto del símbolo.

Terminador:

Secuencia de bits a 0 cuyo número varía según el símbolo y que sirve para señalar el fin de la cadena de bits que representan los datos.

Patrón temporizador*:*

Secuencia alternada de módulos blancos y negros que ayuda a calcular las coordenadas de los módulos del símbolo.

Versión:

Tamaño del símbolo que puede ir desde la versión 1 con 21x21 módulos hasta la versión 40 con 177x177 módulos. Dependiendo de la versión, el símbolo puede tener o no algunos de los elementos descritos y en diferente número; especialmente patrones de alineamiento y la información de versión. Puede indicar también el nivel de corrección aplicado al símbolo.

Información de versión*:*

Patrón codificado en los símbolos de versión 7 o superior que contiene información que indica la versión del símbolo junto con los bits de corrección de errores para los datos. El formato es Versión V-E donde V identifica el número de versión (1-40) y E indica el nivel de corrección de error (L, M, Q, H)

Módulo:

Cuadrito blanco o negro que en conjunto componen el símbolo QR Code. La posición de un módulo está determinada por las coordenadas fila y columna (i,j) de arriba hacia abajo y de izquierda a derecha y comienza en la esquina superior izquierda (0,0) del símbolo.

Símbolo:

Es toda la imagen QR Code, formada por módulos, que conforman los datos en la región de codificación, y los patrones de función.

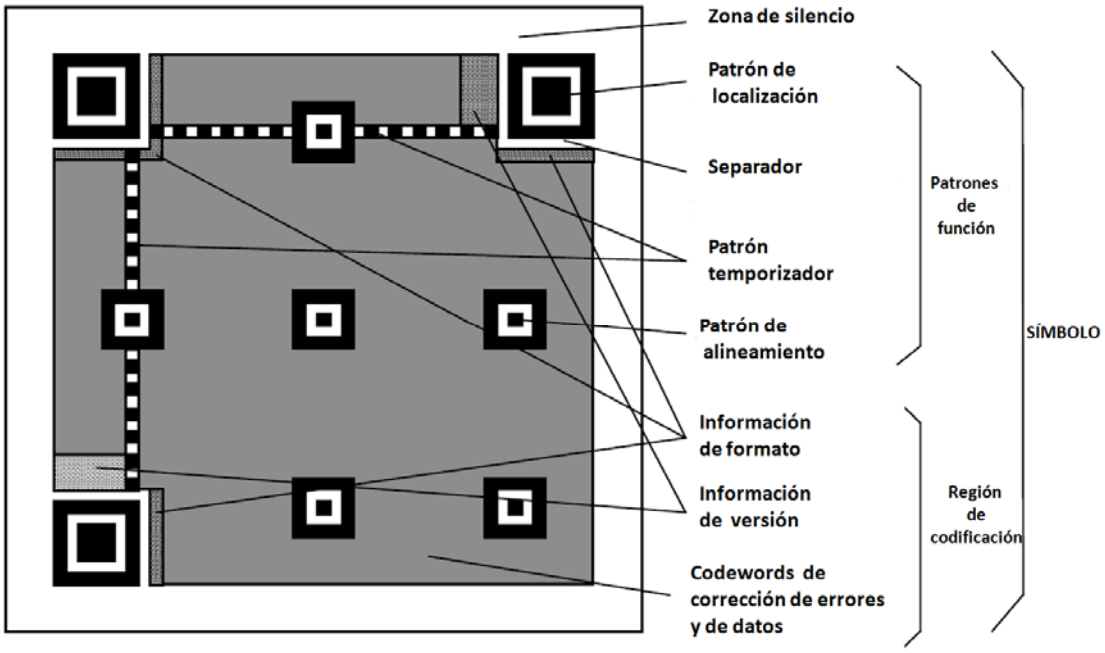


Fig. 2.8. Estructura de un símbolo QR versión 7 [7]

Zona de silen*cio:*

Zona que rodea al símbolo que debe estar en blanco (negro en caso de reflectancia inversa) para delimitar correctamente sus bordes, debe tener una anchura mínima de 4 módulos.

Las características principales del estándar QR Code son las siguientes:

Los símbolos QR Code tienen 40 versiones y 4 grados de corrección de error (L, M, Q, H). Un símbolo 40-H sería un símbolo de versión 40 y corrección de errores H. Cada versión tiene un tamaño, siendo la 1 de 21x21 módulos y la 40 de 177x177 módulos, creciendo en 4 módulos el tamaño de cada versión (la versión 2 sería de 25x25 módulos). Hay 4 modos de codificar los caracteres de datos:

1) Datos numéricos (0-9)

2) Datos alfanuméricos (0-9, A-Z y otros 9 caracteres: espacio, $, %, *, +, -, . , /, :)

3) Bytes (por defecto ISO/IEC 8859-1)

4) Caracteres Kanji, compactados en 13 bits (caracteres de la escritura japonesa)

Para un símbolo 40-L, tenemos lo siguiente:

- Datos numéricos: 7089 caracteres
- Datos alfanuméricos: 4296 caracteres
- Bytes: 2953 caracteres
- Alfabeto Kanji: 1817 caracteres

Fig. 2.9 Ejemplo de símbolo QR Code versión 1 [7]

El sistema corrección de errores se basa en Reed Solomon y tiene 4 niveles:

1) L (low) bajo, puede corregir hasta el 7% de los codewords del símbolo.

2) M (medium) medio, puede corregir hasta el 15% de los codewords del símbolo.

3) Q (quality) calidad, puede corregir hasta el 25% de los codewords del símbolo.

4) H (high) alto, puede corregir hasta el 30% de los codewords del símbolo.

Los módulos del símbolo QR Code pueden ser blancos o negros y representan respectivamente el 0 y el 1 binario. Sin embargo existe un modo de reflectancia inversa donde es al revés. QR Code puede soportar el que la imagen con el símbolo esté rotada o transpuesta lateralmente (mirror image), tiene independencia de orientación.

El patrón localizador se sitúa en las esquinas superior izquierda, superior derecha e inferior izquierda del símbolo QR Code. Está formado por un cuadrado relleno de 3x3 módulos negros, rodeado de un cuadrado de 5x5 módulos blancos que a su vez está rodeado por otro cuadrado de 7x7 módulos negros. Será muy difícil encontrar un patrón de módulos similar a este en otras partes del símbolo. Tener éxito en encontrar los 3 patrones localizadores de un símbolo supone poder calcular la orientación en el campo de visión de éste.

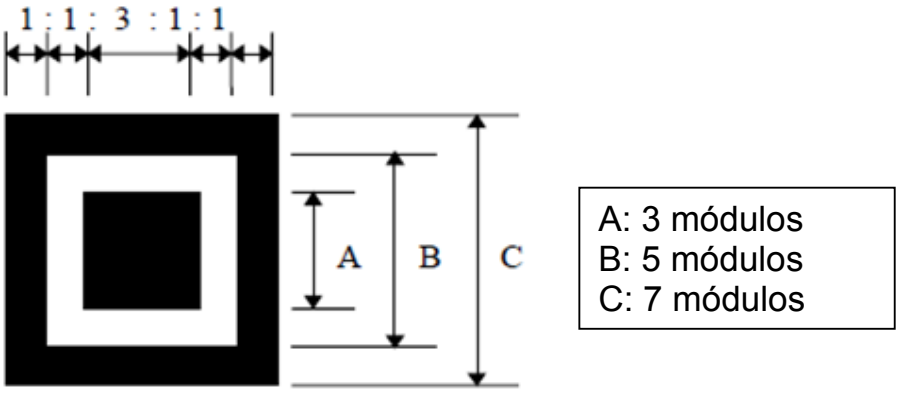


Fig. 2.10 Patrón Localizador [7]

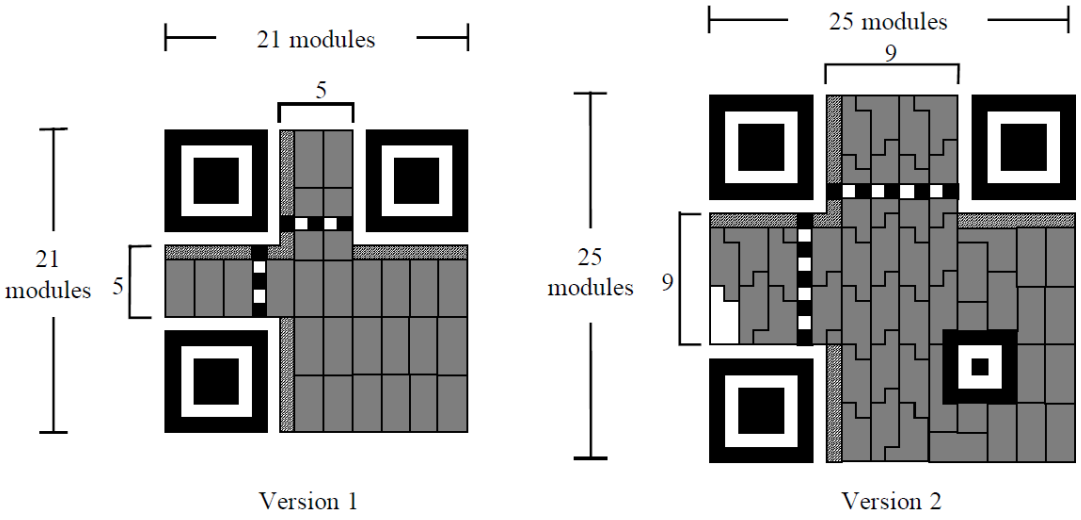


Fig. 2.11. Símbolos de la versión 1 y versión 2 [7]

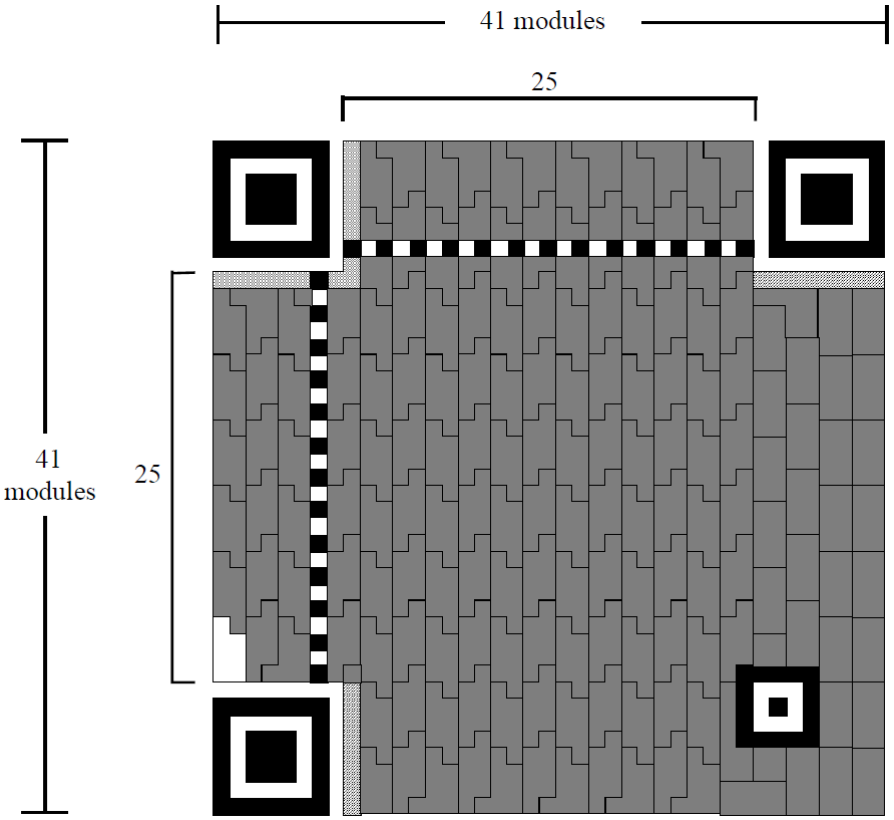


Fig. 2.12. Símbolo de la versión 6 [7]

Los patrones temporizador son dos, uno vertical y otro horizontal. Están formados por una línea o columna de módulos blancos y negros alternados, comenzando y terminando

en un módulo negro. Posibilitan que la versión del símbolo y las coordenadas de los módulos puedan ser determinadas. El temporizador horizontal cruza la fila número 6 entre los separadores superiores y el vertical igual pero cruzando la columna 6.

Los patrones de alineamiento están formados por un módulo negro, rodeado de un cuadrado de 3x3 módulos blancos que a su vez está rodeado por otro cuadrado de 5x5 módulos negros. Su número en el símbolo varía según la versión.

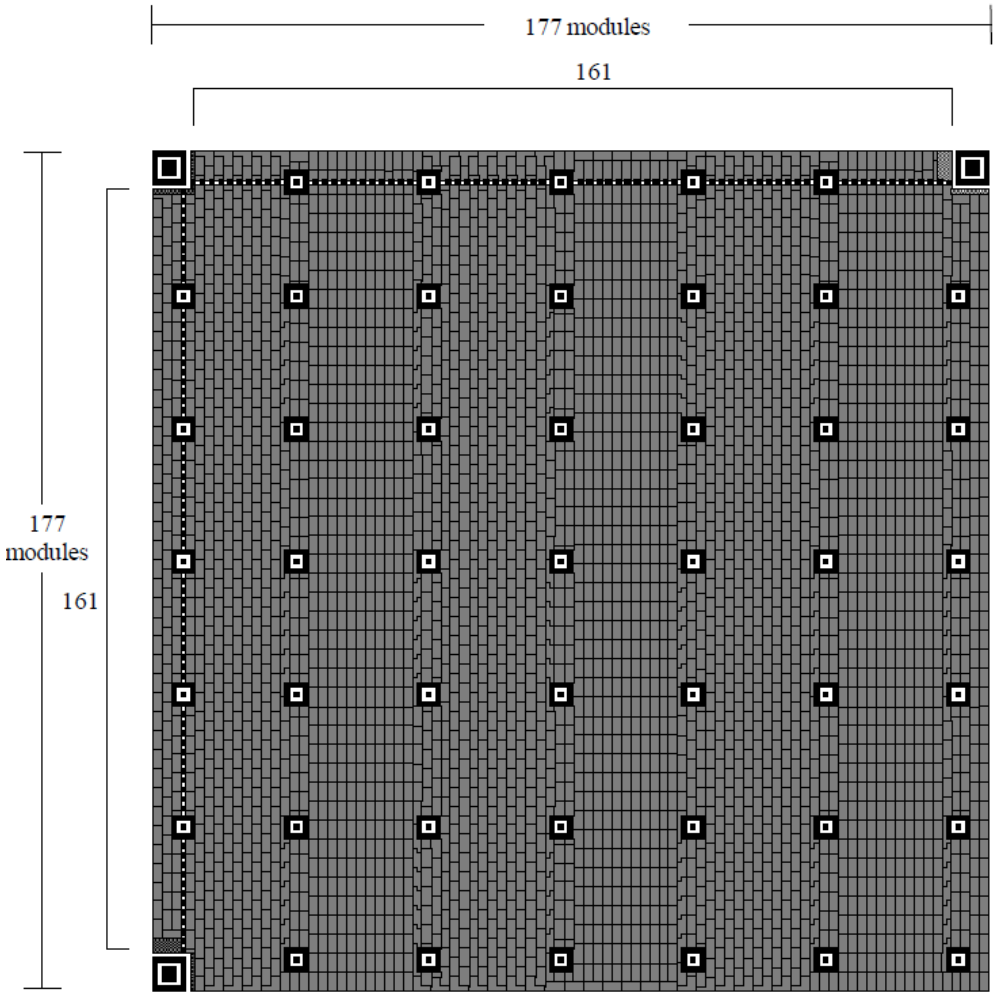


Fig. 2.13. Símbolo de la versión 40 [7]

La región de codificación contiene los codewords que representan los datos, también contiene codewords de corrección de errores, la información de formato y la información de versión en la mayoría de casos.

La zona de silencio debe tener un grosor de 4 módulos rodeando los cuatro bordes del símbolo.

Tabla 2.5. Capacidad de las diferentes versiones de QR Code [7]

Version	No. of Modules/ side (A)	Function pattern modules (B)	Format and Version Information modules (C)	Data modules except (C) ($D=A^2-B-C$)	Data capacity [codewords][a] (E)	Remainder Bits
1	21	202	31	208	26	0
2	25	235	31	359	44	7
3	29	243	31	567	70	7
4	33	251	31	807	100	7
5	37	259	31	1 079	134	7
6	41	267	31	1 383	172	7
7	45	390	67	1 568	196	0
8	49	398	67	1 936	242	0
9	53	406	67	2 336	292	0
10	57	414	67	2 768	346	0
11	61	422	67	3 232	404	0
12	65	430	67	3 728	466	0
13	69	438	67	4 256	532	0
14	73	611	67	4 651	581	3
15	77	619	67	5 243	655	3
16	81	627	67	5 867	733	3
17	85	635	67	6 523	815	3
18	89	643	67	7 211	901	3
19	93	651	67	7 931	991	3
20	97	659	67	8 683	1 085	3
21	101	882	67	9 252	1 156	4
22	105	890	67	10 068	1 258	4
23	109	898	67	10 916	1 364	4
24	113	906	67	11 796	1 474	4
25	117	914	67	12 708	1 588	4
26	121	922	67	13 652	1 706	4
27	125	930	67	14 628	1 828	4
28	129	1 203	67	15 371	1 921	3
29	133	1 211	67	16 411	2 051	3
30	137	1 219	67	17 483	2 185	3
31	141	1 227	67	18 587	2 323	3
32	145	1 235	67	19 723	2 465	3
33	149	1 243	67	20 891	2 611	3
34	153	1 251	67	22 091	2 761	3
35	157	1 574	67	23 008	2 876	0
36	161	1 582	67	24 272	3 034	0
37	165	1 590	67	25 568	3 196	0
38	169	1 598	67	26 896	3 362	0
39	173	1 606	67	28 256	3 532	0
40	177	1 614	67	29 648	3 706	0

[a] All codewords shall be 8 bits in length.

2.2.1 Procedimiento de Codificación de un QR Code

El procedimiento de codificación se divide en 7 pasos:

1. Análisis de los datos:

Se analizan los datos a codificar identificando de qué tipo son sus caracteres, para calcular en que modos codificarlos para ahorrar el máximo espacio. El estándar soporta varios modos de codificación, pudiendo usarse a la vez diferentes modos para cada subconjunto de caracteres. Si no se especificó que versión usar, se debería usar la menor necesaria.

2. Codificación de datos:

Se convierten los datos en un flujo de bits acorde al modo usado. Se inserta un indicador de modo delante de cada subconjunto de datos para saber en qué modo están codificados. También se inserta un terminador y se dividen los datos en codewords de 8bit.

3. Codificación de corrección de errores

Se ejecuta el algoritmo de corrección de error, para generar los codewords de corrección de errores. Éstos se añaden al final de los codewords de datos.

4. Estructurar mensaje

Entrelazar los codewords de datos y de error, y añadir bits restantes si es necesario.

5. Colocación de módulos

Se colocan lo módulos de los codewords en la matriz QR Code junto a los patrones de función para formar el símbolo QR Code.

6. Enmascarar datos

Aplicar los patrones de enmascarado a la región de codificación. Evaluar los resultados y seleccionar el patrón que optimice el equilibrio de módulos blancos y negros y minimice la aparición de patrones indeseables.

7. Información de versión y formato

Generar la información de formato y la de versión si es necesario, y colocarla en el símbolo para completarlo.

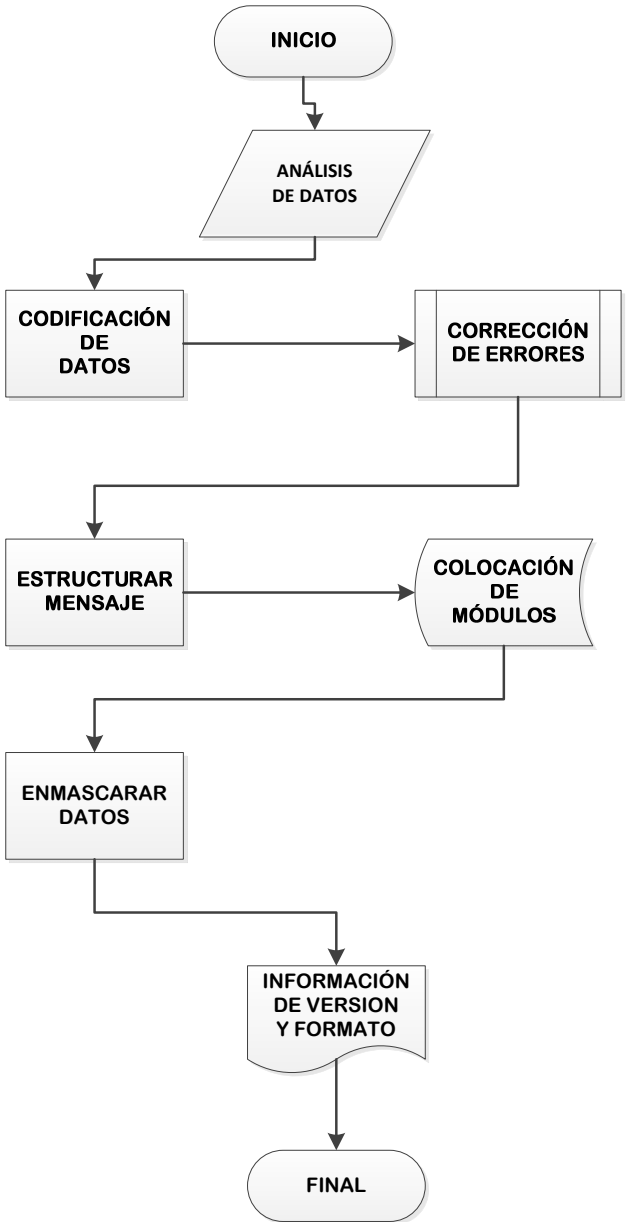


Fig. 2.14. Codificación de un QR Code [Diseño propio]

CAPÍTULO III

PROGRAMACIÓN EN DISPOSITIVOS MÓVILES

3.1 Dispositivos móviles [8]

Un dispositivo móvil es todo aparato electrónico que cumple unas características muy básicas: es de reducido tamaño, haciéndolo fácil de transportar, cuenta con una cierta capacidad de computación y almacenamiento de datos, incorpora elementos de E/S básicos (por lo general, pantalla y/o algún tipo de teclado).

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

Igualmente son habituales la cámara fotográfica y de vídeo, pantalla táctil, teclado QWERTY, receptor de radio, Bluetooth, conexión mediante infrarrojos, dispositivos de memoria extraíbles, localizador GPS, acelerómetro, etc. Desde el punto de vista del software, pueden incorporar también un amplio abanico de aplicaciones tales como programas ofimáticos, reproductores de audio y vídeo, organizadores, videojuegos, navegadores web o clientes de correo, entre otros.

El aumento de las prestaciones y funcionalidades que en la actualidad puede ofrecer cualquier dispositivo móvil dificulta el poder agruparlo dentro de un conjunto determinado. Por ejemplo, un smartphone representa una evolución de un teléfono móvil tradicional, esto es, su cometido es ofrecer comunicación telefónica; sin embargo, cuenta con otros servicios adicionales como la conexión a Internet y aplicaciones, servicios propios de un computador, cámara de fotos y de vídeo o la posibilidad de reproducir películas ó videojuegos.

Los dispositivos móviles pueden ser clasificados en los siguientes grupos:

3.1.1 Dispositivo de comunicación.

Un dispositivo de comunicación es aquel dispositivo móvil cuyo cometido principal es ofrecer una infraestructura de comunicación, principalmente telefónica. Estos dispositivos ofrecen además servicios como el envío de mensajes SMS y MMS, o acceso WAP. En esta categoría se incluiría el tradicional teléfono móvil, precursor indiscutible dentro de los dispositivos móviles, la BlackBerry y el *smartphone*, que amplía considerablemente las prestaciones del primero mediante pantalla táctil, conexión a Internet o la ejecución de aplicaciones.

3.1.2 Dispositivo de computación

Los dispositivos de computación son aquellos dispositivos móviles que ofrecen mayores capacidades de procesamiento de datos y cuentan con una pantalla y teclado más cercanos a un computador de sobremesa. Dentro de este grupo encontramos a las PDA y tablets, muy populares a finales de los años 90 y que permitían al usuario disponer de un organizador mucho más completo que los ofrecidos por los teléfonos móviles del momento, e incluso en ocasiones la visualización de documentos o acceso a Internet. Por otro lado, dispositivo de computación también es un computador portátil o laptop, que dentro de los dispositivos móviles son sin duda los que mayores prestaciones hardware ofrecen (igualando o superando a los de sobremesa) pero también los que tienen, con diferencia, un mayor tamaño, peso y precio. Las calculadoras gráficas pueden ser igualmente incluidas en este grupo de dispositivos de computación.

3.1.3 Reproductor multimedia

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

3.1.4 Grabador multimedia

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

3.1.5 Consola portátil

Una consola portátil es un dispositivo móvil cuya única función es la de proporcionar al usuario una plataforma de juego. Las consolas portátiles fueron, junto a los teléfonos, los primeros dispositivos móviles en convertirse en un producto de masas. Hoy en día representan un importantísimo volumen de ventas dada su gran aceptación en la sociedad y son objeto de auténticas guerras comerciales entre las principales compañías del sector. Algunos ejemplos de esta categoría son la Nintendo DS de Nintendo, o la PSP de Sony.

Dentro de los dispositivos móviles, un smartphone (cuya traducción en español sería “teléfono inteligente”) es una evolución del teléfono móvil tradicional que cuenta con ciertas características y prestaciones que lo acercan más a un ordenador personal que a un teléfono tradicional.

Entre dichas características, se puede encontrar una mejora en la capacidad de proceso y almacenamiento de datos, conexión a Internet mediante Wi-Fi, pantalla táctil, acelerómetro, posicionador geográfico, teclado QWERTY y diversas aplicaciones de usuario como navegador web, cliente de correo, aplicaciones ofimáticas, reproductores de vídeo y audio, etc. incluyendo la posibilidad de descargar e instalar otras nuevas.

A pesar de estas importantes mejoras con respecto a sus predecesores móviles, el reducido tamaño de los smartphones conlleva inexorablemente limitaciones de hardware que los mantienen claramente diferenciados de los computadores convencionales. Estas limitaciones se reflejan principalmente en pantallas más pequeñas, menor capacidad del procesador, restricciones de memoria RAM y memoria persistente, y necesidad de adaptar el consumo de energía a la capacidad de una pequeña batería.

Estas limitaciones obligan a tener muy presente la capacidad real del dispositivo a la hora de desarrollar su software, ya sean aplicaciones de usuario o el propio sistema operativo.

3.2 Sistemas operativos para dispositivos móviles [8]

El sistema operativo destinado a correr en un dispositivo móvil necesita ser fiable y tener una gran estabilidad, ya que incidencias habituales y toleradas en computadores personales como reinicios o caídas no tienen cabida en un dispositivo de estas características. Además, ha de adaptarse adecuadamente a las consabidas limitaciones de memoria y procesamiento de datos, proporcionando una ejecución exacta y excepcionalmente rápida al usuario.

Estos sistemas han de estar perfectamente probados y libres de errores antes de incorporarse definitivamente a la línea de producción. Las posibilidades que existen en un computador estándar de realizar actualizaciones e incluso reinstalar mejores versiones del sistema para cubrir fallos o deficiencias son más limitadas en un dispositivo móvil.

El consumo de energía es otro tema muy delicado: es importante que el sistema operativo haga un uso lo más racional y provechoso posible de la batería, ya que esta es limitada y el usuario siempre exige una mayor autonomía.

Todos estos aspectos de los dispositivos móviles, entre otros muchos, se deben tomar en cuenta a la hora de desarrollar un sistema operativo competente en el mercado, atractivo para los fabricantes y que permita al usuario sacar máximo provecho de su terminal. En la actualidad, existen varios sistemas operativos para toda la gama de dispositivos móviles:

3.2.1 Symbian [9]

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

El acuerdo bajo el cual se desarrolló Symbian es bastante simple: Symbian Ltd. desarrolla el sistema operativo Symbian, que incluye el microkernel, los controladores, el middleware y una considerable pila de protocolos de comunicación e interfaces de

usuario muy básicas. Los desarrolladores que obtienen la licencia correspondiente para trabajar con Symbian implementan sus propias interfaces de usuario y conjuntos de aplicaciones según las necesidades de sus propios dispositivos. Esto permitió a Symbian posicionarse como un sistema operativo muy flexible, que tenía en cuenta los requisitos de la mayoría de los dispositivos fabricados y, al mismo tiempo, permitía un alto grado de diferenciación [9].

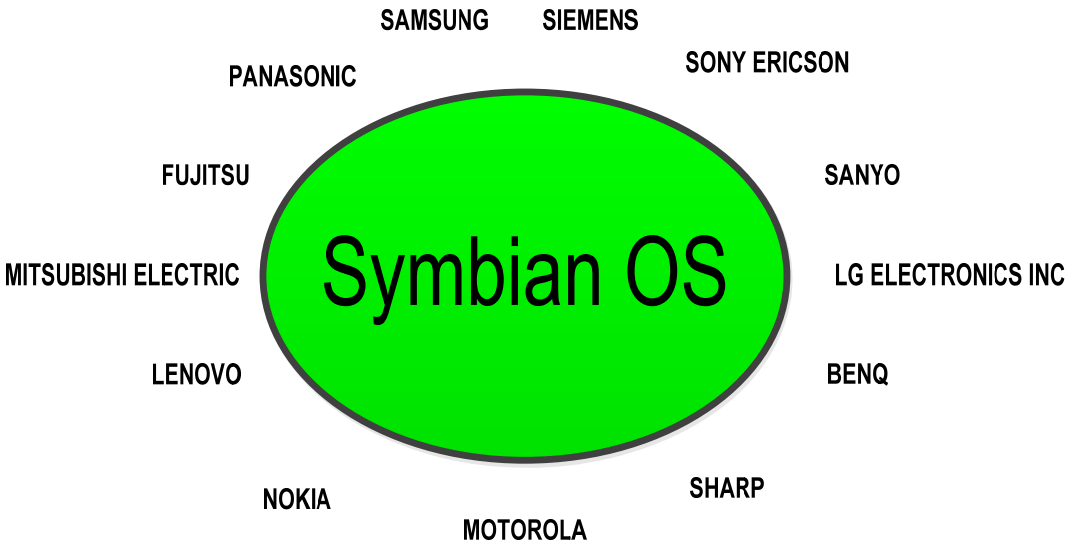


Fig. 3.1 Empresas ligadas a Symbian [9]

En Symbian, una mínima porción del sistema tiene privilegios de kernel; el resto se ejecuta con privilegios de usuario en modo de servidores, de forma que los procesos en ejecución y sus prioridades son manejados por este microkernel. Cada una de las aplicaciones corre en su propio proceso y tiene acceso únicamente a una exclusiva zona de memoria.

Symbian contempla cinco tipos de ediciones o series del sistema operativo según las características del dispositivo móvil [10]. La principal diferencia entre ediciones no radica tanto en el núcleo del sistema operativo como en la interfaz gráfica utilizada:

SERIE 60. El más popular de todos, debido fundamentalmente a que el gigante Nokia, uno de los fabricantes más importantes del mundo, ha hecho de Symbian y de su versión Serie60 el núcleo de casi todos sus modelos de *smartphones*. Los dispositivos con

Serie60 tiene una pantalla pequeña y un teclado del tipo 0-9#. También lo utilizan fabricantes como Siemens, Samsung y Panasonic.

SERIE 80. Esta edición, también usada por Nokia, está más orientada a dispositivos que tienen pantalla táctil y permiten multitarea, pudiendo tener varias aplicaciones abiertas simultáneamente.

SERIE 90. Muy similar a la edición Serie80, sólo que éstos dispositivos tienen una pantalla más grande y llevan incorporados sensores táctiles más desarrollados. Utilizan teclados virtuales, reconocimiento de trazos o teclados acoplables mediante bluetooth

UIQ. La interfaz de esta edición de Symbian se encuentra muy influenciada por Palm OS. Implementan una especie de multitarea virtual, dando al usuario la falsa sensación de poder realizar varias acciones simultáneas; suelen tener un alto costo computacional e influyen negativamente en el tiempo de respuesta apreciado por el usuario. Es utilizado en algunos modelos de Sony Ericcson y Motorola.

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

Desarrollar aplicaciones para Symbian es relativamente sencillo. No es necesario aprender ningún lenguaje de programación nuevo porque permite utilizar lenguajes habituales como Java, C++, Visual Basic o Perl, entre otros, para desarrollar aplicaciones. Este hecho ha permitido que actualmente sean cientos de miles las aplicaciones y utilidades disponibles para Symbian.

3.2.2 Windows mobile [10]

Windows Mobile es un sistema operativo diseñado por Microsoft y orientado a una gran variedad de dispositivos móviles. En realidad, Windows Mobile representa una particularización de otro gran sistema de Microsoft llamado Windows CE.

A principios de la década de los 90, cuando comenzaron a aparecer los primeros dispositivos móviles, Microsoft tomó la decisión de crear un sistema operativo capaz de hacer frente al entonces recientemente lanzado por Apple, el sistema Newton MessagePad. Fruto de esta iniciativa surgió Pegasus, cuyo nombre comercial definitivo fue Windows Compact Edition, o Windows CE [11]. El objetivo principal que buscaba Microsoft era que el nuevo sistema fuera lo suficientemente flexible y adaptable para

poder ser utilizados en un amplio abanico de dispositivos, cuyo única característica común es la de ser de reducido tamaño y tener, por tanto, una limitación obvia en sus recursos.

Las características principales con las que cuenta Windows CE son las siguientes [10]:

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

Un aspecto distintivo de Windows CE con respecto a otros productos desarrollados por Microsoft es que un elevado número de sus componentes se ofrece a los fabricantes y desarrolladores a través del propio código fuente. Esto les permite poder adaptar el sistema a sus dispositivos específicos. Aquellos componentes básicos de Windows CE que no necesitan ningún tipo de adaptación siguen siendo ofrecidos únicamente como código binario.

La arquitectura básica de Windows CE es la explicada a continuación [11]:

OEM Layer: es la capa situada entre el hardware del dispositivo y el kernel.

Permite a los fabricantes desarrollar sus propios drivers y funciones de control de los elementos de hardware.

Operating System Layer: incluye el kernel como elemento principal y el conjunto de API Win32 necesarias. En esta capa se sitúan los bibliotecas de comunicaciones, el gestor gráfico, gestor de ficheros y registros, así como otros componentes opcionales.

Application Layer: donde residen las aplicaciones por defecto de Windows CE y las aplicaciones del usuario.

3.2.3 Android [12]

Android es un sistema operativo para dispositivos móviles como teléfonos inteligentes y tabletas. Es desarrollado por la Open Handset Alliance, la cual es liderada por Google.

Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la Open Handset Alliance, un conglomerado de fabricantes y desarrolladores de hardware, software y operadores de servicio.

Tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos y están disponibles para la tienda de aplicaciones oficial de Android: Android Market, sin tener en cuenta aplicaciones de otras tiendas no oficiales para Android, como pueden ser la App Store de Amazon o la tienda de aplicaciones Samsung Apps de Samsung.

Android Market es la tienda de aplicaciones en línea administrada por Google, aunque existe la posibilidad de obtener software externamente. Los programas están escritos en el lenguaje de programación Java.

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

3.2.3.1 Etimología [12]

Tanto el nombre Android (androide en inglés) como Nexus One hacen alusión a la novela de Philip K. Dick ¿Sueñan los androides con ovejas eléctricas?, que posteriormente fue adaptada al cine como Blade Runner. Tanto el libro como la película se centran en un grupo de androides llamados replicantes del modelo Nexus-6. El logotipo es el robot "Andy".

3.2.3.2 Adquisición por parte de Google [12]

En julio de 2005, Google adquirió Android Inc., una pequeña compañía de Palo Alto, California fundada en 2003.21 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 aquel entonces, poco se sabía de las funciones de Android Inc. fuera de que desarrollaban software para 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.

3.2.3.3 Open Handset Alliance [12]

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. Junto con la formación de la Open Handset Alliance, la OHA estrenó su primer producto, Android, una 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.

3.2.3.4 Historial de actualizaciones [12]

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 en inglés. En cada versión el postre elegido empieza por una letra distinta siguiendo un orden alfabético:

Oficiales:

C: Cupcake (v1.5), magdalena glaseada.

D: Donut (v1.6), rosquilla.

E: Éclair (v2.0/v2.1), pastel francés conocido en España como pepito canuto.

F: Froyo (v2.2), (abreviatura de «frozen yogurt») yogur helado.

G: Gingerbread (v2.3), pan y jenjibre.

H: Honeycomb (v3.0/v3.1/v3.2), panal.

I: Ice Cream Sandwich (v4.0), sandwich de helado.

J: Jelly Bean (¿¿??) Gominola.

3.3 Lenguaje de programación Java Micro Edition [13]

La compañía Sun Microsystems lanzó al mercado el lenguaje de programación Java, a mediados de los años 90, con la finalidad de controlar equipos electrodomésticos. Con el paso de los años, Java se ha ido adaptando a las necesidades de los usuarios y las empresas, por lo que la compañía Sun Microsystems ha desarrollado soluciones personalizadas de Java, agrupándolas en ediciones: Java 2 Standard Edition (J2SE) que agrupa a un conjunto básico de herramientas usadas para desarrollar Java Applets y aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, Java 2 Enterprise Edition (J2EE) que agrupa herramientas enfocadas al entorno empresarial sobre una red de computadoras y totalmente distribuido, Java 2 Micro Edition (J2ME) que utiliza herramientas para dispositivos pequeños con capacidades restringidas de procesador, memoria y pantalla gráfica que una computadora de escritorio. J2EE es un superconjunto de J2SE y J2ME representa una versión simplificada de J2SE. El conjunto de J2ME, J2SE y J2EE se conoce como tecnología Java 2.

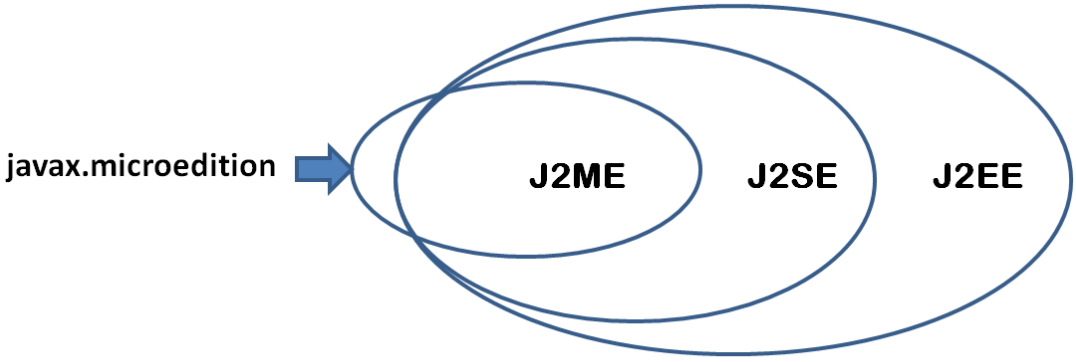


Fig. 3.2 Ediciones de Java [13]

La edición Java 2 Micro Edition, lanzada en 1999, tiene una parte de su API fija (API: Application Programming Interface: Interfaz de Programación de Aplicaciones o interfaz de comunicación entre componentes de software), que es aplicable a todos los dispositivos inalámbricos y una parte que es específica para ciertos dispositivos; ejemplo, la API especifica de Palm, la de teléfonos móviles, la de Handhelds o PDAs.

3.3.1 Tecnologías para móviles: J2ME y WAP

WAP es Wireless Application Protocol o protocolo de aplicación inalámbrico. WAP permite a dispositivos inalámbricos soportar un navegador web simplificado. Para comunicaciones WAP debe estar adaptado a esta tecnología el cliente, el servidor y un gateway intermedio debe existir. El gateway WAP es el responsable de convertir las petición WAP y peticiones web habituales y viceversa. Las páginas que se transmiten a través de WAP no son archivos HTML sino que son WML. Si la web habitual soporta javascript, WML cuenta con un lenguaje de script simplificado a partir de javascript que se llama WMLscript.

Wap es una tecnología que está funcionando para moviles adaptados. Mucha gente habla de la competencia que supone J2ME para WAP, cuando esta aseveración no tiene

ningún sentido. WAP es competencia a J2ME como lo es HTML a Java en el entorno web con cable. Es decir: son cosas distintas y no pueden competir entre sí. J2ME es una tecnología que permite la creación de aplicaciones que reciban y envíen datos a través de redes inalámbricas. WAP es sencillamente un protocolo para navegar la web en dispositivos móviles. Por tanto ambas tecnologías coexistirán sin problemas.

3.3.2 J2ME y SMS

SMS es la tecnología que permite hacer algo que vemos todos los días: mandar mensajes cortos entre dispositivos móviles, así cómo recibir otro tipo de mensajes. Por tanto, J2ME y SMS son cosas lo suficientemente diferentes como para no tener que competir. Salvo casos de aplicaciones de chat o de mensajería con J2ME, es muy lateral la competencia de J2ME sobre SMS.

3.3.3 J2ME y Bluetooth

La filosofía de Bluetooth es habilitar la comunicación en rangos relativamente cortos entre dispositivos. En la práctica sirve para quitarnos de encima los cables que conectan los ordenadores sustituyendo estos por una conexión de radio. Esto da más comodidad y libertad en el uso del ordenador. Bluetooth no representa por tanto ninguna relación directa con J2ME.

3.3.4 Configuración y CLDC

A la vez que J2ME supone un novedoso nuevo campo por desarrollar, de la misma manera introduce términos que debemos entender para asumir la arquitectura del sistema.

El primer término que debemos asumir es el de Configuración. Una configuración es un conjunto mínimo de APIs que son útiles para desarrollar aplicaciones para un conjunto definido de dispositivos. Son muy importantes porque describen las funcionalidades más importantes requeridas para unos dispositivos determinados.

Hay una configuración estándar para dispositivos inalámbricos que se conoce como Connected Limited Device Configuration o CLDC. Este estándar describe el conjunto de funcionalidades mínimas de los dispositivos inalámbricos, acorde a su potencia y a sus características. CLDC es resumen el conjunto de APIs básicas para construir aplicaciones para dispositivos móviles.

CLDC es un estándar que SUN ha especificado. Según vaya J2ME aplicándose a nuevas familias de dispositivos, SUN especificará nuevos estándares adecuados a cada familia.

Extendiendo un poco el concepto de CLDC, este estándar especifica los siguientes aspectos de la programación wireless:

- El subconjunto del lenguaje java que puede ser usado.
- El subconjunto de funciones de la Máquina Virtual Java.
- Las APIs fundamentales para este tipo de desarrollo.
- Los requerimientos de hardware de los dispositivos móviles enfocados a CLDC.

Algunas características de java han sido deshabilitadas y la razónes sencilla: los dispositivos móviles tienen capacidad limitada en hardware lo que restringe el uso de algunos aspectos de java y de la máquina virtual
La función más importante que cumple CLDC es indicar un conjunto de APIs que debe incorporar cualquier dispositivo.

Finalmente, CLDC establece los mínimos de hardware requeridos para J2ME. Estos son los siguientes:

- 160 KiB de memoria disponible para Java
- Procesador de 16-bits
- Bajo consumo energético
- Conexión a una red (9600 bps, puede ser menor la velocidad de transmisión)

Los dispositivos aptos para CLDC son teléfonos móviles, PDAs, ciertos electrodomésticos, etc. Muchos otros se irán incorporando al estándar CLDC siempre y cuando cumplan los mínimos marcados por el estándar e incorporen la máquina virtual.
Examinando estos requerimientos más a fondo, es conveniente saber qué significa el requerimiento de 160 KiB de memoria disponible para java. CLDC explicita que esta cantidad está compuesta de 128 KiB de memoria no volatil para la Máquina Virtual Java y las APIs de CLDC y otros 32 KiB de memoria volatil para el sistema Java runtime.

3.3.5 CLDC y Java

Véamos que restricciones le impone el CLDC al lenguaje de programación java. La primera limitación es la falta de soporte para los números de punto flotante, de manera que no ofrece soporte para este tipo de operaciones matemáticas. La limitación se impone porque los dispósitivos carecen de hardware para estas operaciones y hacerlo vía software sería cargarlos por encima de sus posibilidades.

La siguiente restricción es la eliminación del método Object.finalize (). Este método es llamado para que un óbjeto sea borrado de memoria y CLDC no lo soporta ni lo requiere.

Una restricción a tener en cuenta es la limitada capacidad de CLDC para el manejo de excepciones. Esto es debido a que gran parte del manejo de excepciones depende exclusivamente de las APIs de cada dispositivo en concreto, pues las excepciones dependen de las características de cada aparato. Por ello CLDC maneja un número limitado de excepciones y delega el resto del manejo de excepciones en las APIs específicas de cada familia de dispositivos.

3.3.6 CLDC y seguridad

Al igual que la version estándar de Java, J2ME tiene su propio modelo de seguridad. Si estás habituado a desarrollar applets estarás familiarizado con el modelo sandbox de seguridad. Este modelo establece que un applet sólo puedo ejecutar ciertas operaciones que se consideran seguras. Hay otra serie de operaciones que estan fuera del sandbox y que no pueden ser ejecutadas por el applet.

¿Dónde está el límite? La filosofía de este modelo de seguridad es dotar a los desarrolladores de grandes posibilidades para hacer aplicaciones potentes, pero por otra parte minimizar los riesgos que supone un dispositivo que ejecuta aplicaciones descargadas de la red. Esto define las líneas maestras del modelo sandbox de seguridad:

- Los archivos de clases Java deben ser verificados como aplicaciones Java válidas.
- Sólo se permite el uso de APIs autorizadas por CLDC.
- No está permitido cargar clases definidas por el usuario.
- Sólo características nativas que entren dentro del CLDC pueden ser accedidas.
- Por lo general, las restricciones impuestas por CLDC son restricciones de bajo nivel y esto es así porque CLDC trabaja a este nivel. Se supone que una capa adicional, que está definida en una configuración, impondrá nuevas restricciones.

Veámos que es exactamente una configuración o profile.

3.3.7 MID Profile (MIDP)

Ahora ya estamos preparados para entender el papel de las configuraciones dentro de la arquitectura J2ME. Sabemos que son un conjunto de APIs pensadas para un tipo concreto de dispositivos. De hecho, son una descripción de una familia de dispositivos que añade un conjunto de APIs adicionales al CLDC que se corresponden con las funcionalidades específicas de estos dispositivos.

CLDC es la configuración básica de J2ME. MIDP lleva CLDC más allá y añade nuevos requerimientos y APIs obligatorios para dispositivos MIDP. Los requerimientos de memoria de MIDP son:

- 128 KiB de memoria no volatil para las librerias MIDP API.
- 32 KiB de memoria volatil para el sistema Java runtime.
- 8 KiB de memoria no volatil para datos de aplicación persistente.

Respecto a CLDC, MIDP sólo incremente los 8kb destinados a datos persistentes.

Los requerimientos de entrada para MIDP exigen la existencia de un teclado o una pantalla tactil o ambos. No se exige ratón (raro en un teléfono móvil).

Los requerimientos de salida para MIDP son algo más importantes, porque la pantalla es una de las restricciones mayores de los dispositivos móviles. MIDP exige al menos una pantalla de 97 x 54 pixels (anchura, altura) con 1-bit de profundidad de color (blanco y negro). El ratio de salida debe ser 1:1.

MIDP tiene también requerimientos de red. El mínimo soporte de red exigido es disponer de una conexión inalámbrica de 2 sentidos. Se supone que estos dispositivos pueden tener un ancho de bando limitado (9600 bps).

MIDP no establece ninguna obligación respecto a qué sistema operativo debe tener el dispositivo. Esto es posible gracias a que Java es multiplataforma. Sin embargo sí que se esperan ciertos mínimos:

- Un kernel mínimo que maneje el hardware a bajo nivel.
- Un mecanismo que lea y escriba en memoria persistente o no volatil.
- Un mecanismo de temporización para establecer mediciones temporales y dotar de información de tiempo a datos persistentes.

- Acceso de lectura y escritura hacia la conexión inalambrica.
- Acceso a la entrada por teclado o pantalla.
- Soporte mínimo para mapas de bits.
- Un mecanismo que controle el ciclo de vida de una aplicación.

3.3.8 Arquitectura de Java ME

Una aplicación Java ME típica está formada por un archivo JAR, que es el que contiene a la aplicación en sí, y un archivo opcional JAD (Java Archive Descriptor) que contiene diversa información sobre la aplicación.

Una aplicación en Java ME se desarrolla a partir de una combinación de:

Máquina virtual: existen disponibles dos máquinas virtuales de Java ME con diferentes requisitos, cada una pensada para tipos distintos de pequeños dispositivos:

KVM, o KiloByte Virtual Machine, se corresponde con la máquina virtual más pequeña desarrollada por Sun. Se trata de una implementación de máquina virtual reducida y orientada a dispositivos de 16 ó 32 bits con al menos 25 MHz de velocidad y hasta 512 KiB de memoria total disponible.

CVM, o Compact Virtual Machine, soporta las mismas características que la Máquina Virtual de Java SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2 MiB o más de memoria RAM.

Configuración: una configuración consiste en un conjunto de clases básicas destinadas a conformar el corazón de la aplicación. En concreto, dentro de Java ME existe dos configuraciones:

Connected Limited Device Configuration (CLDC) enfocada a dispositivos con restricciones de procesamiento y memoria.

Connected Device Configuration (CDC) enfocada a dispositivos con más recursos.

Perfil: bibliotecas de clases específicas, orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.

Cada una de las dos configuraciones mencionadas requiere en realidad de una determinada máquina virtual. De esta forma, si es escoge la configuración

CLDC, será necesario utiliza la máquina virtual denominada CVM; si, por el contrario, se decide utilizar la configuración CDC, la máquina virtual necesaria es la conocida como KVM.

Con la elección de perfiles, se da una situación similar. Existen unos perfiles que se utilizan sobre la configuración CDC (Foundation Profile, Personal Profile y RMI Profile) y otros que lo hacen sobre CLDC (PDA Profile y Mobile Information Device Profile, conocido como MIDP).

Las aplicaciones en Java ME que se realizan utilizando el perfil MIDP reciben el nombre de MIDlets. Se dice así que un MIDlet es una aplicación Java ME realizada con el perfil MIDP, sobre la configuración CLDC, y usando la máquina virtual KVM.

Desde un punto de vista práctico, MIDP es el único perfil actualmente disponible.

3.4 La plataforma Android

Android utiliza la licencia Apache 2,0, lo que significa que, como software libre, cualquier desarrollador tiene acceso completo al SDK (Kit de Desarrollo de Software) del sistema, incluidas todas sus API, documentación y emulador para pruebas, pudiendo distribuirlo y modificarlo. A los desarrolladores se les proporciona de forma gratuita un SDK y la opción de un plug-in para el entorno de desarrollo Eclipse, que incluyen todas las APIs necesarias para la creación de aplicaciones, así como un emulador integrado para su ejecución.

Android constituye una pila de software pensada especialmente para dispositivos móviles y que incluye tanto un sistema operativo, como middleware y diversas aplicaciones de usuario. Representa la primera incursión seria de Google en el mercado móvil y nace con la pretensión de extender su filosofía a dicho sector.

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

3.4.1 Arquitectura Android

Cada una de las capas que forman la arquitectura de android, utiliza servicios ofrecidos por las anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores

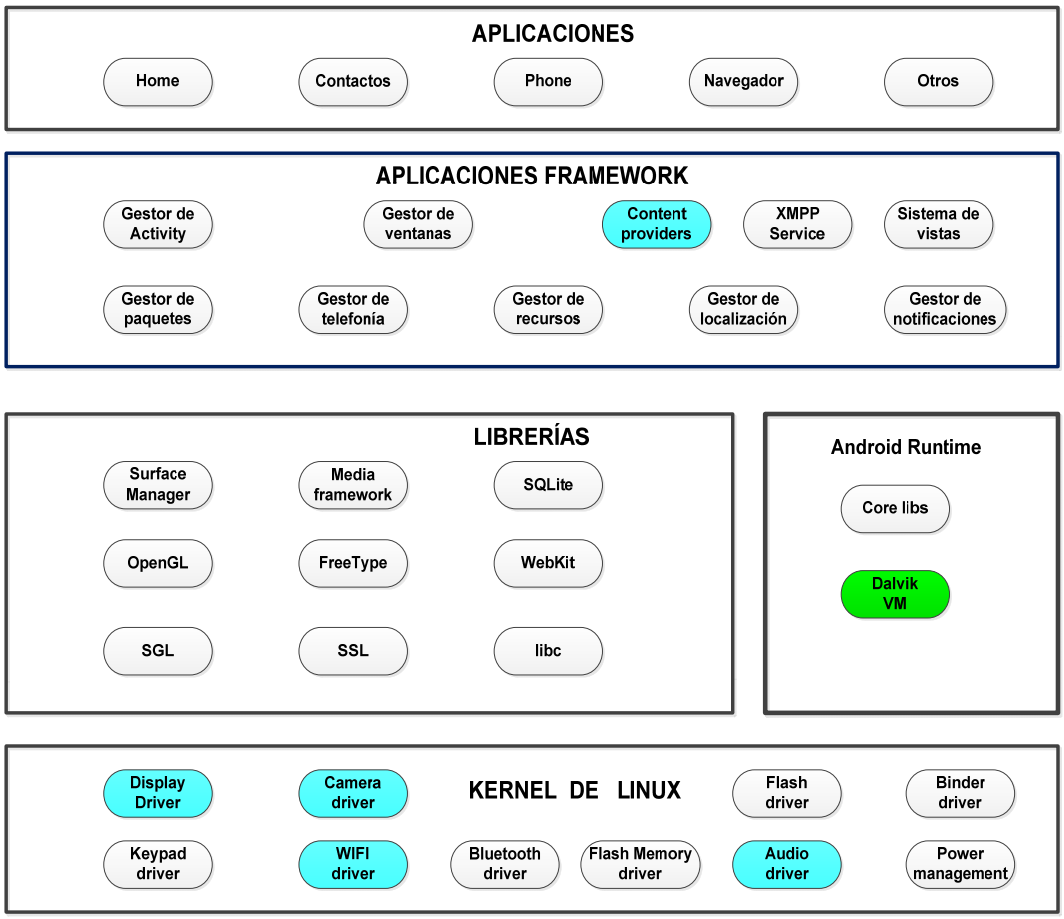


Fig. 3.3 Arquitectura de Android [14]

La capa más baja es la corresponde al **núcleo de Android**. Android utiliza el núcleo de Linux 2.6 como una capa de abstracción para el hardware disponible en los dispositivos móviles. Esta capa contiene los drivers necesarios para que cualquier componente hardware pueda ser utilizado mediante las llamadas correspondientes.

Siempre que un fabricante incluya un nuevo elemento de hardware, lo primero que se debe realizar para que pueda ser utilizado desde Android es crear las librerías de control o drivers necesarios dentro de este kernel de Linux embebido en el propio Android.

La siguiente capa se corresponde con las librerías utilizadas por Android. Éstas han sido escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características. Junto al núcleo basado en Linux, estas librerías constituyen el corazón de Android.

Entre las librerías más importantes de este nivel, se pueden mencionar las siguientes:

La librería **libc** incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.

La librería **Surface Manager** es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.

OpenGL/SL y SGL representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de Android. OpenGL/SL maneja gráficos en 3D y permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el hardware encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de Android es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

La librería **Media Libraries** proporciona todos los códecs necesarios para el contenido multimedia soportado en Android (vídeo, audio, imágenes estáticas y animadas, etc.)

FreeType, permite trabajar de forma rápida y sencilla con distintos tipos de fuente

La librería **SSL** posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.

A través de la librería **SQLite**, Android ofrece la creación y gestión de bases de datos relacionales, pudiendo transformar estructuras de datos en objetos fáciles de manejar por las aplicaciones.

La librería **WebKit** proporciona un motor para las aplicaciones de tipo navegador, y forma el núcleo del actual navegador incluido por defecto en la plataforma Android.

Al mismo nivel que las librerías de Android se sitúa el entorno de ejecución. Éste lo constituyen las Core Libraries, que son librerías con multitud de clases de Java, y la **máquina virtual Dalvik**.

Los dos últimos niveles de la arquitectura de Android están escritos enteramente en Java. El **framework de aplicaciones** representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación. Toda aplicación que se desarrolle para Android, ya sean las propias del dispositivo, las desarrolladas por Google o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API y el mismo framework, representado por este nivel.

Entre las API más importantes ubicadas aquí, se pueden encontrar las siguientes:

Activity Manager, importante conjunto de API que gestiona el ciclo de vida de las aplicaciones en Android (del que se hablará más adelante).

Window Manager, gestiona las ventanas de las aplicaciones y utiliza la librería ya vista Surface Manager.

Telephone Manager, incluye todas las API vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.)

Content Providers, permite a cualquier aplicación compartir sus datos con las demás aplicaciones de Android. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.

View System, proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, check-boxes, tamaño de ventanas, control de las interfaces mediante tacto o teclado, etc.

Incluye también algunas vistas estándar para las funcionalidades más frecuentes.

Location Manager, posibilita a las aplicaciones la obtención de información de localización y posicionamiento, y funcionar según ésta.

Notification Manager, mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc.

Si llevan asociada alguna acción, en Android denominada Intent, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

XMPP Service, colección de API para utilizar este protocolo de intercambio de mensajes basado en XML.

El último nivel del diseño arquitectónico de Android son las **aplicaciones**. Éste nivel incluye tanto las incluidas por defecto de Android como aquellas que el usuario vaya añadiendo posteriormente, ya sean de terceras empresas o de su propio desarrollo.
Todas estas aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

3.4.2 La máquina virtual Dalvik

En Android, todas las aplicaciones se programan en el lenguaje Java y se ejecuta mediante una máquina virtual de nombre Dalvik [15], específicamente diseñada para Android. Esta máquina virtual ha sido optimizada y adaptada a las peculiaridades propias de los dispositivos móviles (menor capacidad de proceso, baja memoria, alimentación por batería, etc.) y trabaja con ficheros de extensión .dex (Dalvik Executables). Dalvik no trabaja directamente con el bytecode de Java, sino que lo transforma en un código más eficiente que el original, pensado para procesadores pequeños.

Gracias a la herramienta “dx”, esta transformación es posible: los ficheros .class de Java se compilan en ficheros .dex, de forma que cada fichero .dex puede contener varias clases. Después, este resultado se comprime en un único archivo de extensión .apk (Android Package), que es el que se distribuirá en el dispositivo móvil. Dalvik permite varias instancias simultáneas de la máquina virtual, y a diferencia de otras máquinas virtuales, está basada en registros y no en pila, lo que implica que las instrucciones son más reducidas y el número de accesos a memoria es menor

Así mismo, Dalvik no permite la compilación Just-in-Time.

Según los responsables del proyecto, la utilización de esta máquina virtual responde a un deseo de mejorar y optimizar la ejecución de aplicaciones en dispositivos móviles, así como evitar la fragmentación de otras plataformas como Java ME. A pesar de esta aclaración oficial, no ha pasado inadvertido que con esta maniobra Android ha esquivado tener que utilizar directamente Java ME, evitando así que los futuros desarrolladores de aplicaciones tengan que comprar ninguna licencia, ni tampoco que publicar el código

fuente de sus productos [16]. El lenguaje Java utilizado en Android no sigue ningún JSR determinado, y Google se afana en recordar que Android no es tecnología Java, sino que simplemente utiliza este lenguaje en sus aplicaciones [17].

3.4.3 Componentes de una aplicación

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

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

3.4.4 Activity

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

La mayoría de las aplicaciones permiten la ejecución de varias acciones a través de la existencia de una o más pantallas. Por ejemplo, piénsese en una aplicación de mensajes de texto. En ella, la lista de contactos se muestra en una ventana. Mediante el despliegue de una segunda ventana, el usuario puede escribir el mensaje al contacto elegido, y en otra tercera puede repasar su historial de mensajes enviados o recibidos. Cada una de estas ventanas deberías estar representada a través de un componente Activity, de forma que navegar de una ventana a otra implica lanzar una actividad o dormir otra. Android permite controlar por completo el ciclo de vida de los componentes **Activity**.

Muy vinculado a este componente se encuentran los Intents, una interesante novedad introducida por Android. Un **Intent** consiste básicamente en la voluntad de realizar alguna

acción, generalmente asociada a unos datos. Lanzando un **Intent**, una aplicación puede delegar el trabajo en otra, de forma que el sistema se encarga de buscar qué aplicación entre las instaladas es la que puede llevar a cabo la acción solicitada. Por ejemplo, abrir una URL en algún navegador web, o escribir un correo electrónico desde algún cliente de correo.

3.4.5 Broadcast Intent Receiver

Un componente ***Broadcast Intent Receiver*** se utiliza para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente, abrir un componente ***Activity***), por ejemplo, una llamada telefónica entrante ó un SMS recibido. Este componente, no tiene interfaz de usuario asociada, pero puede utilizar el API ***Notification Manager***, mencionada anteriormente, para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil. Este componente se implementa a través de una clase de nombre ***BroadcastReceiver***.

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

3.4.6 Service

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

Un ejemplo típico de este componente es un reproductor de música. La interfaz del reproductor muestra al usuario las distintas canciones disponibles, así como los típicos botones de reproducción, pausa, volumen, etc. En el momento en el que el usuario reproduce una canción, ésta se escucha mientras se siguen visionando todas las acciones anteriores, e incluso puede ejecutar una aplicación distinta sin que la música deje de sonar. La interfaz de usuario del reproductor sería un componente ***Activity***, pero la música en reproducción sería un componente Service, porque se ejecuta en background.

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

3.4.7 Content Provider

Con el componente ***Content Provider***, cualquier aplicación en Android puede almacenar datos en un archivo, en una base de datos **SQLite** o en cualquier otro formato que considere. Además, estos datos pueden ser compartidos entre distintas aplicaciones. Una clase que implemente el componente ***Content Provider*** contendrá una serie de métodos que permite almacenar, recuperar, actualizar y compartir los datos de una aplicación.

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

3.4.8 Ciclo de vida de las aplicaciones Android [14]

En Android, cada aplicación se ejecuta en su propio proceso. Esto aporta beneficios en cuestiones básicas como seguridad, gestión de memoria, o la ocupación de la CPU del dispositivo móvil. Android se ocupa de lanzar y parar todos estos procesos, gestionar su ejecución y decidir qué hacer en función de los recursos disponibles y de las órdenes dadas por el usuario. El usuario desconoce este comportamiento de Android; simplemente es consciente de que mediante un simple clic pasa de una aplicación a otra y puede volver a cualquiera de ellas en el momento que lo desee. No debe preocuparse sobre cuál es la aplicación que realmente está activa, cuánta memoria está consumiendo, ni si existen o no recursos suficientes para abrir una aplicación adicional. Todo eso son tareas propias del sistema operativo.

Android lanza tantos procesos como permitan los recursos del dispositivo. Cada proceso, correspondiente a una aplicación, estará formado por una o varias actividades independientes (componentes ***Activity***) de esa aplicación. Cuando el usuario navega de una actividad a otra, o abre una nueva aplicación, el sistema duerme dicho proceso y realiza una copia de su estado para poder recuperarlo más tarde. El proceso y la actividad siguen existiendo en el sistema, pero están dormidos y su estado ha sido guardado.

Es entonces cuando crea, o despierta si ya existe, el proceso para la aplicación que debe ser lanzada, asumiendo que existan recursos para ello.

Cada uno de los componentes básicos de Android tiene un ciclo de vida bien definido; esto implica que el desarrollador puede controlar en cada momento en qué estado se encuentra dicho componente, pudiendo así programar las acciones que mejor convengan. El componente ***Activity***, probablemente el más importante, tiene un ciclo de vida como el mostrado en la Figura 3.4.

Cuando se crea una actividad, se invoca el evento ***onCreate().*** Este evento sólo se invoca la primera vez que se llama a una actividad, o bien cuando se llama después de que el sistema haya tenido que eliminarla por falta de recursos . ***onStart()*** es el evento invocado cada vez que la actividad se muestra al usuario. Es decir, la primera vez que se muestra, y las veces que en las que vuelve a aparecer tras haber estado oculta. En este último caso, se invoca ***onStop()*** al desaparecer y ***onRestart()*** inmediatamente antes de reaparecer. ***onFreeze()*** y ***onPause()*** son llamadas secuencialmente cuando otra actividad va a encargarse de la interacción con el usuario. Tras ***onPause()*** la actividad permanece en un estado de espera en el que puede ocurrir que la aplicación sea destruida, por lo que estos eventos se usan para consolidar la información que no queremos que se pierda. Si la actividad no se destruye volverá al primer plano con el evento ***onResume().*** La idea importante es que una actividad que esté pausada o detenida (tras ***onPause()*** u ***onStop()***) puede ser destruida por el sistema si previo aviso, por lo que debemos guardar antes la información necesaria (durante ***onFreeze()*** y ***onPause()).***

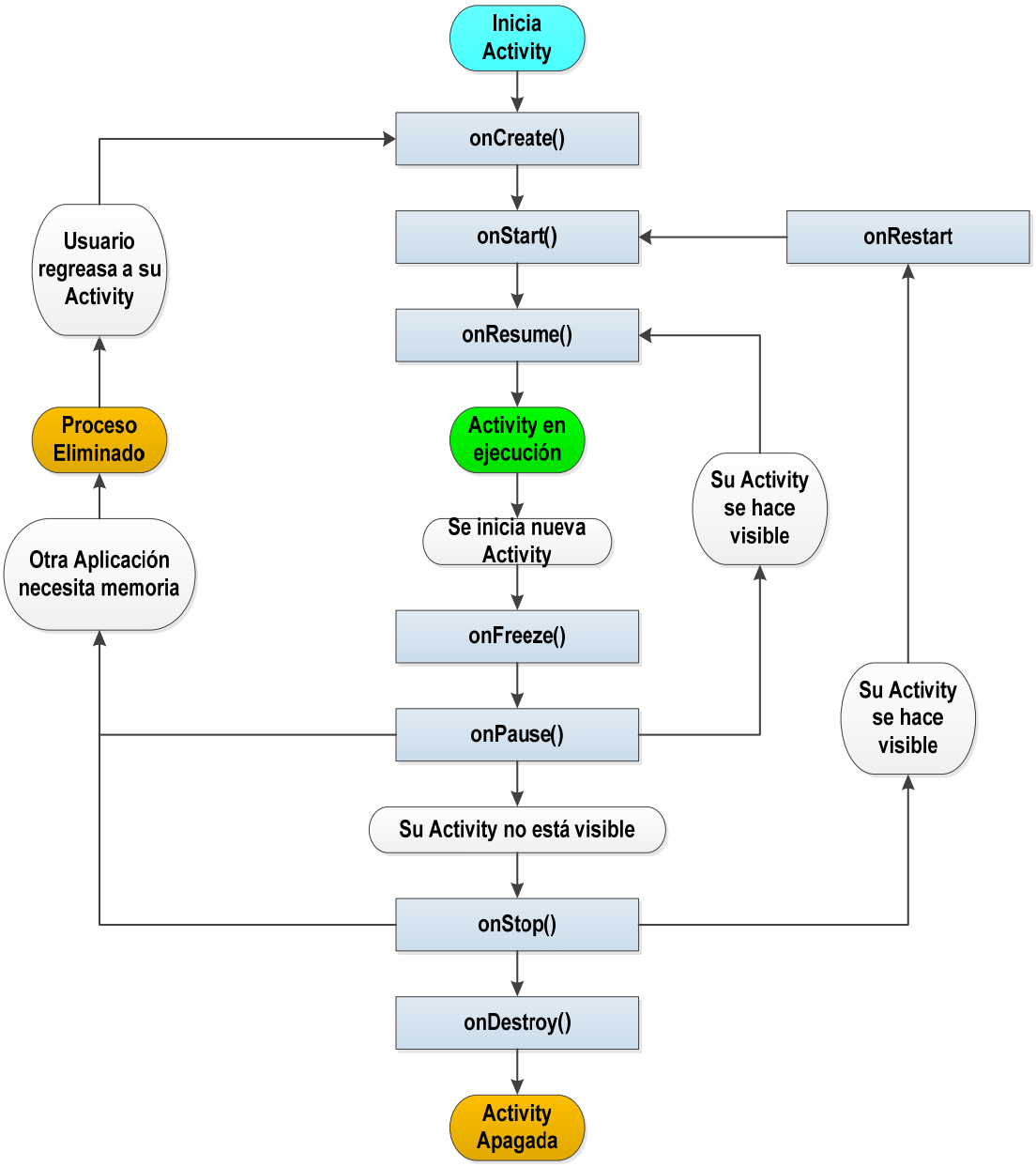


Fig. 3.4 Ciclos de vida de las aplicaciones Android [14]

3.4.9 Política de eliminación de procesos

Cada aplicación Android se ejecuta en su propio proceso. Este proceso se crea cada vez que una aplicación necesita ejecutar parte de su código, y seguirá existiendo hasta que la aplicación finalice o hasta que el sistema necesite utilizar parte de sus recursos para otra aplicación considerada prioritaria.

Por ello, es importante saber cómo los componentes de una aplicación en Android (***Activity, Broadcast Intent Receiver, Service y Content Provider***) determinan e influyen en el ciclo de vida de la aplicación. No usar los componentes correctamente a la hora de construir una aplicación puede significar que el sistema operativo la termine cuando en realidad está haciendo una tarea importante para el usuario.

Android construye una jerarquía donde evalúa la clase de componentes que están ejecutándose y el estado de los mismos. En orden de importancia, serían los siguientes:

A. **Procesos en primer plano**: aquellos necesarios para lo que el usuario está haciendo en ese momento. Un proceso se encuadra en esa categoría si cumple alguna de las siguientes condiciones:

a. tiene un componente Activity ejecutándose, con la que el usuario está interactuando.

b. tiene un componente Broadcast Intent Receiver ejecutándose.

c. ha lanzado algún otro proceso que tiene un componente Service ejecutándose en el momento Idealmente, sólo debería haber algunos de estos procesos en el sistema, y su eliminación debería darse únicamente en casos extremos en los que la falta de recursos impide siguen ejecutándolos todos.

B. **Procesos visibles**: aquellos procesos que contienen un componente Activity visible en la pantalla, pero no con el foco de actividad en ese momento.

C. **Procesos de servicio**: aquellos procesos que tienen un componente Service y que están ejecutándose en background. Aunque no sean visibles directamente al usuario, desempeñan tareas sí percibidas por este.

D. **Procesos en segundo plano**: procesos con un componente Activity, que no son visibles al usuario. Estos procesos no tienen una importancia directa para el usuario en ese momento.

E. **Procesos vacíos**: procesos que ya no ejecutan ninguna actividad, pero que se mantienen en memoria para agilizar una posible nueva llamada por parte del usuario.

Según esta jerarquía, Android prioriza los procesos existentes en el sistema y decide cuáles han de ser eliminados, con el fin de liberar recursos y poder lanzar la aplicación requerida.

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

4.1 Ingeniería de requerimientos y restricciones del sistema

En términos generales, el proyecto permitirá que los visitantes al museo, obtengan mayor información sobre la obra que están observando en un determinado momento. La información adicional se presentará al usuario visitante en modo texto, imagen, audio y video. Para ello el usuario debe contar con un dispositivo portátil como teléfono celular o tablet dotado de cámara para la lectura del QR code, pantalla con soporte de video y audífonos.

4.1.1 Alcances

Los alcances del sistema propuesto se describen proporcionando una noción de la funcionalidad del sistema y los beneficios esperados.

Aportes

- Diseñar e implementar el sistema de información multimedia (BBDD)
- Diseñar e implementar la interfaz de usuario con Android
- Diseñar e implementar la red de transporte
- Diseñar e implementar la red de acceso inalámbrico con WIFI

4.1.2 Identificación del sistema y funcionalidad general

El sistema debe ser una herramienta que aporte y ayude a mostrar información en línea en el formato requerido a los visitantes al museo, ya que en la actualidad ningún museo peruano cuenta con el apoyo de la tecnología para brindar respuestas individualizadas a las consultas de los usuarios visitantes sobre el tema de su interés sin la necesidad de contar con un guía grupal.

El sistema permitirá mostrar información en formato de texto, imagen, audio y video en un dispositivo móvil conectado inalámbricamente con tecnología WIFI, y debe estar disponible para todos los usuarios que estén autorizados a consultar esta información.

4.1.3 Beneficios esperados

Los beneficios más importantes de este desarrollo son:

- Brindar información fidedigna, suministrando datos veraces en cada uno de los formatos que el usuario seleccione: texto, imagen, audio ó video.
- Ofrecer un acceso sencillo, rápido y seguro donde puedan consultar la información de una forma homogenizada y facilitando la realización de consultas en diferentes formatos.
- Contar con un sistema con información que permita a sus usuarios mantener la información actualizada.

4.2 Casos de uso

Para mostrar las funcionalidades que la aplicación Museo presentará al usuario, se utilizan los casos de uso. Se muestran dos usuarios, pero en realidad se trata del mismo ususrio accediendo a las diferentes funcionalidades de la aplicación.

En ingeniería del software, un caso de uso representa un uso típico que se le da al sistema. La técnica de los casos de uso permite capturar y definir los requisitos que debe cumplir una aplicación, y describe las típicas interacciones que hay entre el usuario y la aplicación. Dicha técnica es utilizada con frecuencia por los ingenieros del software para, mostrar al cliente de forma clara y sencilla qué tipo de acciones podrá realizar su futuro sistema.

A continuación se muestra un diagrama con los casos de uso asociados a Museo.

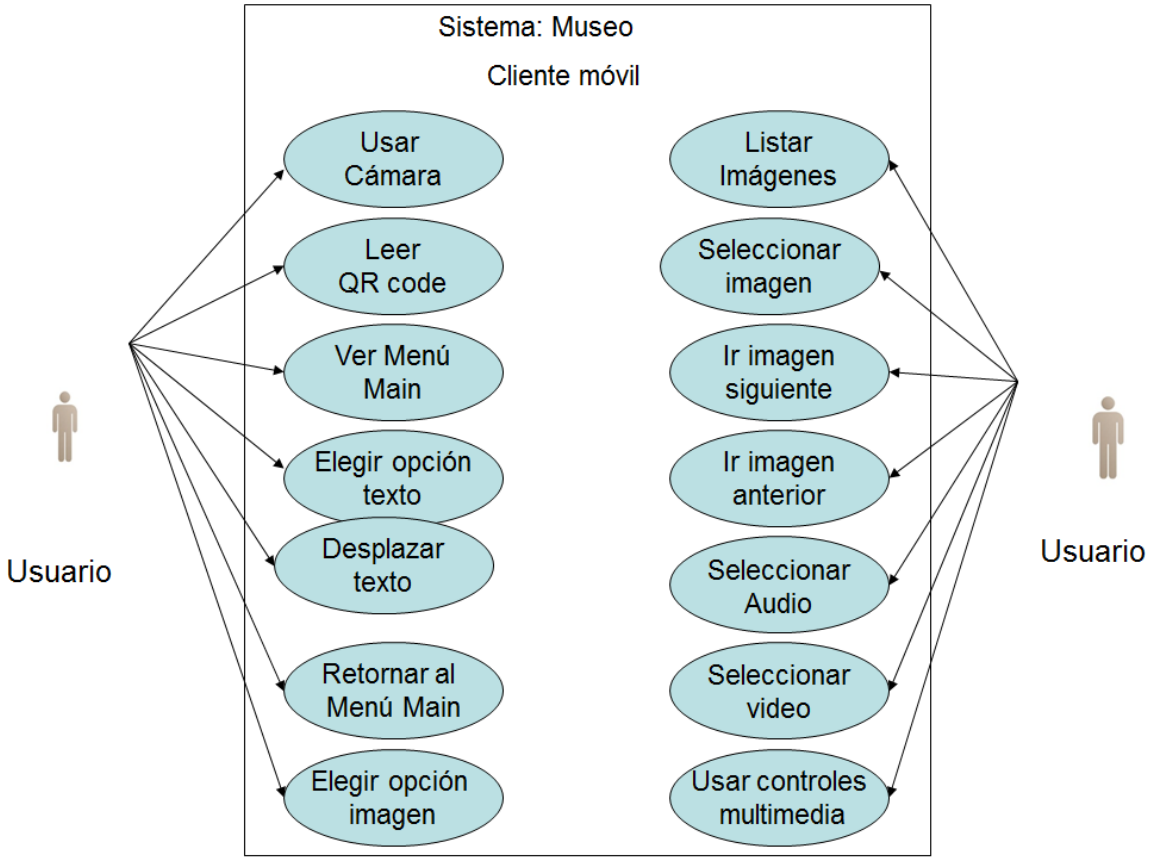


Fig. 4.1. Casos de uso de la aplicación Museo [diseño propio]

En el diagrama mostrado en la Fig. 4.1, el sistema, está representado por una caja que contiene los casos de uso. Cada caso de uso consiste en un óvalo con un nombre descriptivo en su interior. Fuera del sistema se encuentra los actores que pueden interactuar con él. En este caso, existe un único actor de nombre “usuario” que es el que realiza todos los casos de uso desde su dispositivo móvil. Los detalles de cada uno de los casos son autodescriptivos.

También se muestra a continuación los casos de uso asociados al servidor al que se conecta la aplicación para solicitar información el formato de texto, imagen, audio y video.

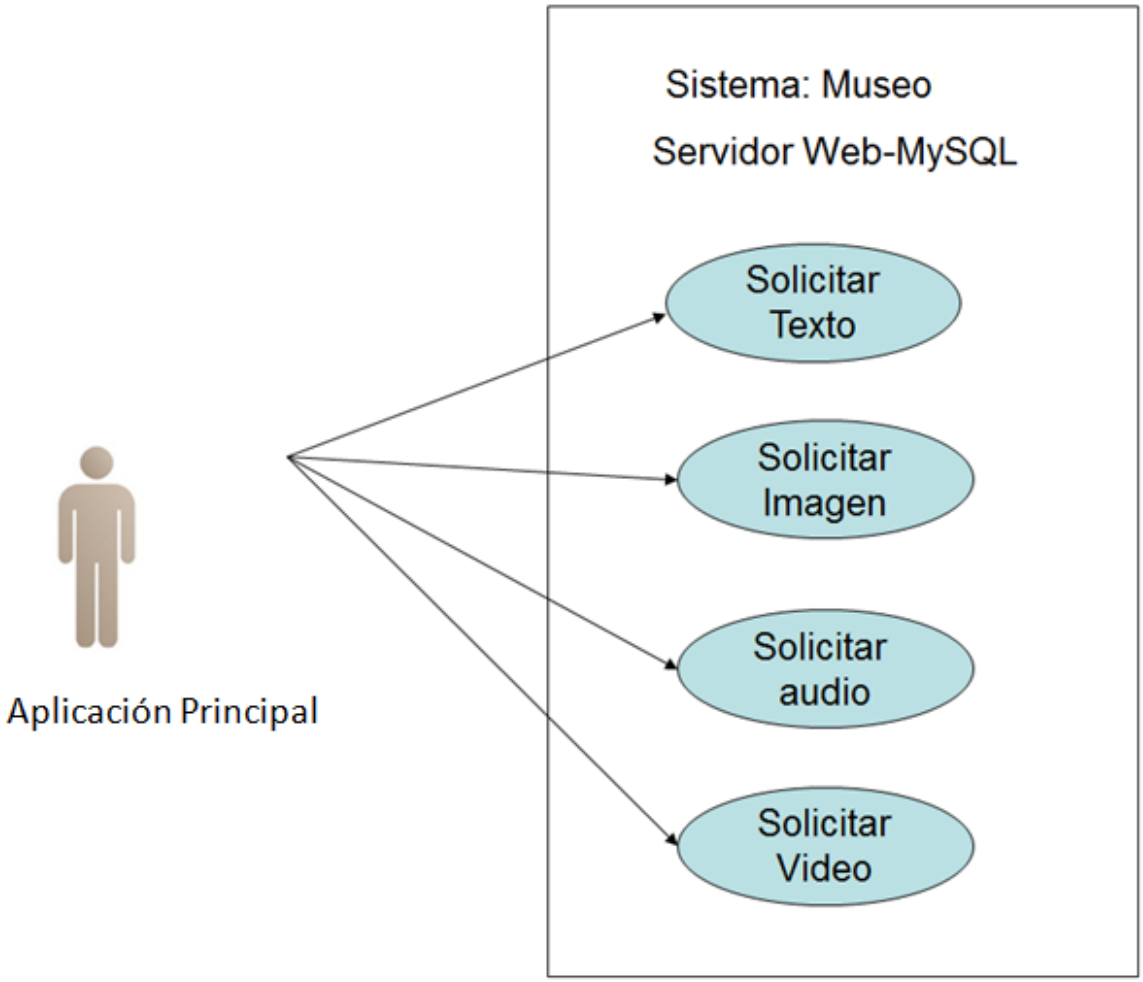


Fig. 4.2. Casos de uso asociados al servidor [diseño propio]

En el diagrama de la Fig. 4.2, el único actor que realiza los casos de uso es la propia aplicación Principal, la que establece la conexión con el servidor. Los casos de uso considerados también son autodescriptivos.

Los casos de uso expresan el punto de vista del usuario sobre cómo debe funcionar la aplicación y qué puede realizar a través de ella, y son una forma de facilitar su comprensión. No tiene por qué existir ninguna correspondencia entre los casos de uso y las clases que se implementan; sin embargo, todas las clases que forman el sistema completo deben realizar, lo que los casos de uso muestran.

4.3 Modelamiento del sistema

A continuación, se muestra el diagrama de flujo del sistema completo. La aplicación permite mostrar información a los usuarios visitantes en formato de texto, imagen, audio y video a partir de la lectura de un código bidimensional QR Code.

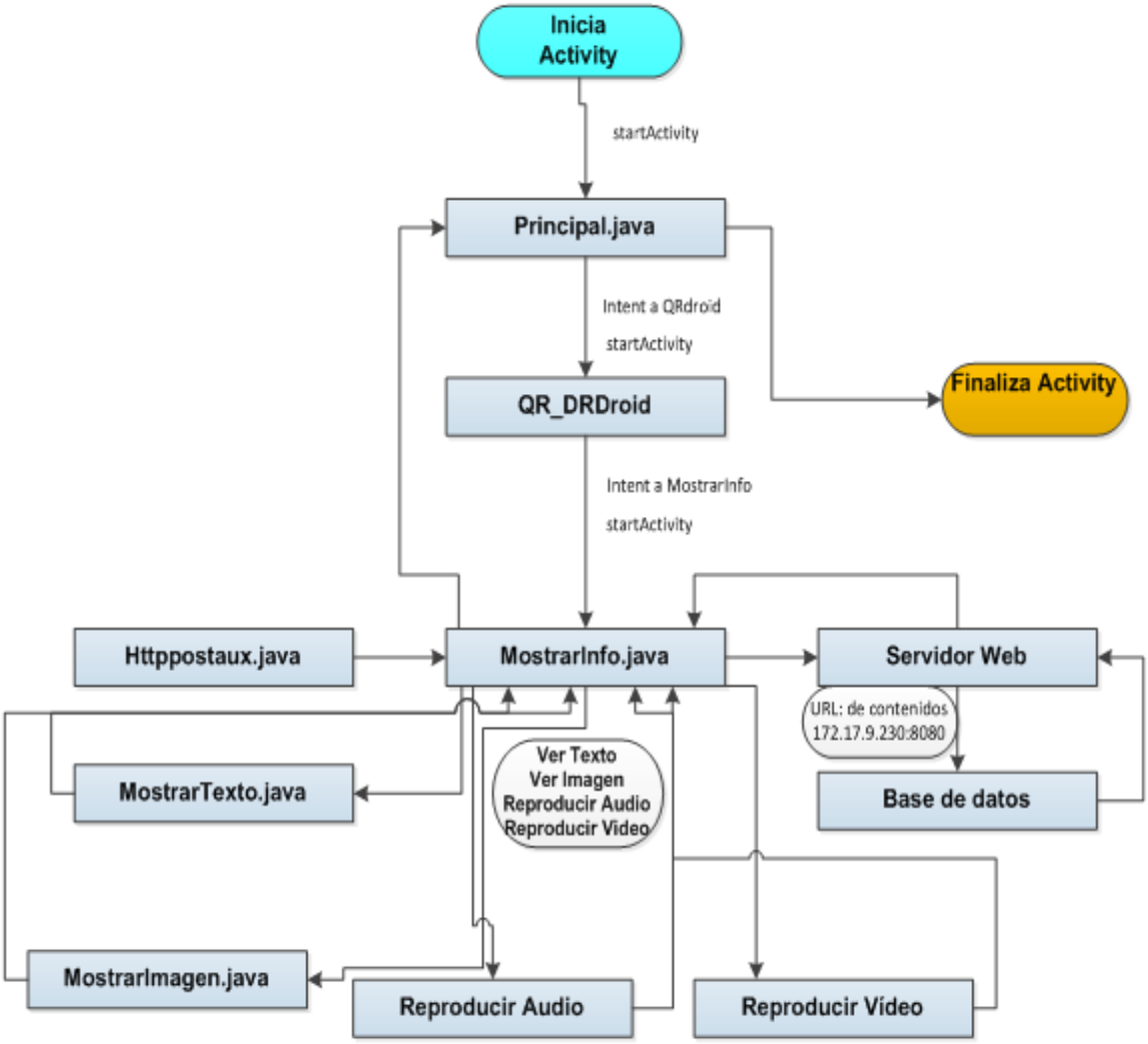


Fig. 4.3. Diagrama de Flujo del sistema [diseño propio]

El funcionamiento del diagrama de flujo, se describe a continuación: La aplicación se inicia cuando el usuario selecciona la actividad **Principal.java** desde el escritorio del dispositivo móvil, ésta actividad se inicia con el método ***startActivity***, e inmediatamente envía un mensaje (***Intent)*** a la aplicación **QR_DRDroid**, instalado también en el

dispositivo móvil. La aplicación **QR-DRDroid** se activa y luego espera hasta que el usuario seleccione la ***opción QR Scanner*** que aparece en el escritorio del dispositivo móvil. El usuario selecciona la opción ***QR Scanner,*** la cámara de video se activa y el usuario tiene la posibilidad de elegir el QR Code adosado a la obra de su interés, dicha lectura es automática y el resultado se direcciona a un servidor web por lo que hace uso de una actividad auxiliar llamada **Httpostaux.java** para soporte del protocolo ***HTTP,*** que permite el intercambio de información ***XML*** entre un servidor web y el dispositivo móvil. La actividad **Httpostaux.java** brinda apoyo a la actividad **MostrarInfo.java**, que es la encargada de controlar el menú principal de opciones para que el usuario seleccione el formato de la información que desea visualizar o reproducir. Así mismo la actividad **MostrarInfo.java** se comunica con el servidor web Apache y éste a su vez realiza transacciones con el servidor de base de datos MySQL.

El menú de opciones permite al usuario elegir las siguientes opciones o formatos de información: Ver texto, Ver Imagen, Reproducir Audio y reproducir Vídeo, todas las opciones permiten retornar **a MostrarInfo.java**, para una nueva selección o para salir del menú o el programa principal.

Para visualizar texto en el dispositivo móvil, desde la actividad **MostrarInfo.java**, enviamos un mensaje *(**Intent***) a la actividad **MostrarTexto.java**, que permite leer la información en formato de texto y desplazarnos con el uso de la pantalla táctil y luego retroceder a la actividad **MostrarInfo.java**.

Para visualizar imágenes en el dispositivo móvil, desde la actividad **MostrarInfo.java**, enviamos un mensaje (***Intent***) a la actividad **MostrarImagen.java**, que permite ver la información en formato de imagen, los formatos soportados son: JPEG, GIF, PNG, BMP; podemos desplazarnos con el uso de la pantalla táctil y luego retroceder a la actividad **MostrarInfo.java**.

Para reproducir audio en el dispositivo móvil, desde la actividad **MostrarInfo.java**, enviamos un mensaje (***Intent***) a la aplicación **Reproducir Audio**, que permite escuchar audio en los audífonos del dispositivo móvil. Los formatos que soporta son: MP3, MIDI, Vorbis, PCM/WAVE; el audio se puede detener, avanzar, retroceder y finalizar con los

controles físicos respectivos ó la pantalla táctil del dispositivo móvil y luego retornar a la actividad **MostrarInfo.java**.

Para reproducir vídeo, desde la actividad **MostrarInfo.java**, enviamos un mensaje (***Intent***) a la aplicación **Reproducir Vídeo**, que permite visualizar en la pantalla el video relacionado con la obra elegida. Los formatos que soporta son: H.263, H.264, MPEG-4, VP8; el vídeo se puede detener, avanzar, retroceder y finalizar con los controles de la pantalla táctil del dispositivo móvil y luego retornar a la actividad **MostrarInfo.java**.

4.3.1 Análisis y diseño de la aplicación

Para desarrollar la aplicación hemos elegido la plataforma Android, porque combina la universalidad de los teléfonos móviles, el tremendo entusiasmo del software de código abierto y el gran respaldo corporativo que ofrecen Google y otros miembros de la Open Handset Alliance como Motorola, Verizon, Samsung, ATt&T, etc.

Los principales componentes de una aplicación en plataforma Android son las “Actividades”, que viene a ser como un componente gráfico, los “Servicios”, que se ejecutan en segundo plano y los “Intentos”, que permite el envío de mensajes y la comunicación entre las “Actividades”.

El archivo principal de una aplicación escrita en Android es el “**AndroidManifest.xml**”, que tiene una etiqueta “aplication”, donde se colocan las actividades y dentro de la actividad están los “intentos”, que permiten interactuar con las otras actividades. Un estracto del archivo **Androidmanifest.xml** se muestra a continuación. El archivo completo está en el apéndice B (Listado de programas).

```
<activity
   android:name=".QRScanner"
   android:label="@string/title_activity_qrscanner"
   android:screenOrientation="portrait">
   <intent-filter>
```

```
        </intent-filter>
    </activity>
    <activity
        android:name=".Principal"
        android:label="@string/title_activity_principal"
        android:screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
```

La clase que se encarga de la actividad principal se llama “**Principal.java**”, que tiene un método “onCreate” que llama al proceso “super” y luego utiliza el componente “Intent” para llamar a la actividad **MostrarInfo.java**. Un estracto del programa principal.java se muestra a continuación. El código completo se encuentra en el apéndice B (listado de programas).

```
public class Principal extends Activity {

        private static final int ACTIVITY_RESULT_QR_DRDROID = 0;
        public static final String SCAN = "la.droid.qr.scan";
         Button button ;
         Button btnSalir ;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
     /*
      Intent i=new Intent(Principal.this, MostrarInfo.class);
      i.putExtra("cadena","gioconda");
      startActivity(i);


****************************************

//Set action to button
                button.setOnClickListener( new OnClickListener() {
                        public void onClick(View v) {
                        //Create a new Intent to send to QR Droid
Intent qrDroid = new Intent( "la.droid.qr.scan" ); //Set action "la.droid.qr.scan"
                                qrDroid.putExtra( "la.droid.qr.complete" , true);
                        startActivityForResult(qrDroid, 0);
**********************************

Intent i=new Intent(Principal.this, MostrarInfo.class);
```

```
i.putExtra("cadena",result);
startActivity(i);
```

El archivo R.java, ubicado en la carpeta “gen”, maneja los recursos como valores (values), cadenas (strings) y pantallas (layouts), que son los componentes gráficos en formato xml. Estos recursos son indexados en el archivo R.java, para referencias rápidas posteriores. Todos los archivos que se encuentran en la carpeta **/res**, se compilan para generar el archivo **R.java**, que guarda una constante por cada tipo de recurso. Un estracto del archivo **R.java** se muestra a continuación. El código completo se encuentra en el apéndice B (listado de programas).

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.qrinfo;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;
        public static final int ic_launcher=0x7f020001;
    }
    public static final class id {
        public static final int Button06=0x7f070010;
        public static final int TextView01=0x7f070012;
        public static final int TextView02=0x7f070011;
        public static final int TextView03=0x7f07000f;
        public static final int btnDesAudio=0x7f070008;
        public static final int btnDesImagen=0x7f070007;
        public static final int btnDesTexto=0x7f070005;
        public static final int btnDesVideo=0x7f070009;
        public static final int btnImagenRegresar=0x7f070002;
        public static final int btnInfoPrincipal=0x7f07000a;
        public static final int btnRepAudio=0x7f07000d;
        public static final int btnRepVideo=0x7f07000e;
        public static final int btnSalir=0x7f070018;
        public static final int btnTextoRegresar=0x7f070016;
        public static final int btnVerImagen=0x7f07000c;
        public static final int btnVerTexto=0x7f07000b;
```

Los archivos para diseñar la interfaz de usuario se encuentran en la carpeta **/res/layout** y **/res/menu**; parte del código **activity_principal.xml**, se muestra a continuación. El código completo se encuentra en el apéndice B (listado de programas).

```
<TextView
   android:id="@+id/textView1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_alignParentLeft="true"
   android:layout_alignParentRight="true"
   android:layout_alignParentTop="true"
   android:layout_marginTop="92dp"
   android:text="            QR INFO"
   android:textSize="30dp"
   tools:context=".Principal" />

<Button
   android:id="@+id/button1"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_alignRight="@+id/textView1"
   android:layout_below="@+id/textView1"
   android:layout_marginTop="54dp"
   android:text="QR Scanner" />
```

Todos los archivos que forman parte de la aplicación se compilan en un archivo classes.dex, que se encuentra en la carpeta /bin. Luego Android hace una segunda compilación del código java con Dalvik Virtual Machine para generar el archivo empaquetado con extensión **.apk** (QRInfo.apk), que se copia y se ejecuta en el dispositivo móvil.

Nuestra aplicación tiene los siguientes objetos:

- **Principal.java**: programa principal que activa el aplicativo QRCode Scanner.
- **MostrarInfo. Java**: programa que interactúa con los demás programas para reproducir audio, vídeo y para mostrar texto e imagen.
- **Httppostaux.java**: programa auxiliar utilizado por MostrarInfo.java
- **MostrarTexto.java**: permite visualizar texto en el dispositivo móvil

- **MostrarImagen.java**: Permite visualizar imágenes en el dispositivo móvil

Por otro lado, mediante el establecimiento de una conexión a la red a través de WIFI, la aplicación desarrollada, se conectará al servidor WEB y a partir de este servidor, se conectará a la base de datos MySQL utilizando código php. El intercambio de información entre la aplicación Principal del móvil y el servidor WEB, se realiza utilizando documentos XML (eXtended Markup Language), que permite expresar de forma sencilla y abierta diferentes estructuras de datos, manteniendo intacta su semántica y contenidos. XML, cuenta además con diferentes APIs en Java.

Para trasladar el formato XML a Java, utilizamos SAX ("Simple API for XML") que procesa ("parse") el documento o información en formarto XML por eventos, es decir conforme se vaya presentando (evento por evento).

Cuando se desea utilizar documentos XML que cuenten siempre con una misma estructura, se definen unas plantillas que expresan la forma en la que debe ser construido el documento XML para ser considerado válido. Una de estas plantillas es la que se denomina DTD (Definición de Tipo de Documento), y expresa de forma muy clara qué tipo de elementos e información puede contener un determinado documento XML.

4.3.2 El servidor web y el servidor de base de datos

Toda la información solicitada por el usuario, se encuentra almacenada en una base de datos libre (MySQL).

La aplicación **Principal** se conecta al servidor Web enviando el formato XML sobre el protocolo de capa de aplicación HTTP. El servidor WEB traslada la consulta al servidor de base de datos utilizando un programa (script) escrito en PHP.

El requerimiento de información que hace la aplicación cliente**"Principal**", instalada el el dispositivo móvil, es recibida por el servidor web (**Apache Tomcat**), que luego la traslada al servidor de base de datos. El servidor de base de datos lee la consulta y devuelve la información solicitada

La parte del servidor, está formada por tres componentes:

- **Una base de datos**, que almacena toda la información reqerida por el usuario.
- **Un servlet**, que atiende la petición recibida, la procesa y envía la respuesta correspondiente.
- **Un servidor web**, donde reside y se ejecuta el servlet, y que permanece a la espera de conexiones HTTP entrantes.

Para la **base de datos** se utiliza el gestor MySQL (de libre distribución), usándolo bajo la licencia gratuita que permite su explotación para desarrollos no lucrativos, como es la aplicación que ocupa este proyecto.

El **servlet** es el componente que procesa las peticiones que envían los usuarios, accede a la base de datos en busca de la información solicitad y devuelve la respuesta. Un servlet es un componente Java, generalmente pequeño e independiente de la plataforma, que se ejecuta en un servidor WEB al que extiende su funcionalidad.

Cuando recibe una petición, el servlet utiliza la librería SAX de Java para leer el documento XML completo, consulta en la base de datos la petición del usuario.

Esta consulta la realiza gracias a las librerías de JDBC y MySQL de Java. Finalmente, el servlet compone y envía a su vez una respuesta en un nuevo documento XML, donde incluirá los datos solicitados por el usuario y que están almacenados en el servidor de base de datos.

El **servidor WEB** utilizado es Apache Tomcat (de libre distribución) y es el entorno donde se ejecuta el servlet, quedando siempre a la espera de recibir conexiones HTTP. Todas las comunicaciones entre la aplicación Principal y el servidor se envían a través de dicho protocolo. En las peticiones, el método HTTP utilizado es POST, correspondiente al envío de datos, y se encapsula el documento XML en una variable de nombre “xml”. En las respuestas, se envía el documento XML correspondiente en el campo de datos.

4.3.3 Arquitectura del sistema

Luego de conocer el funcionamiento del sistema MuseoFiee, las bondades ofrecidas al usuario, el comportamiento del lado servidor y el intercambio de información mediante XML, se muestra la arquitectura del sistema.

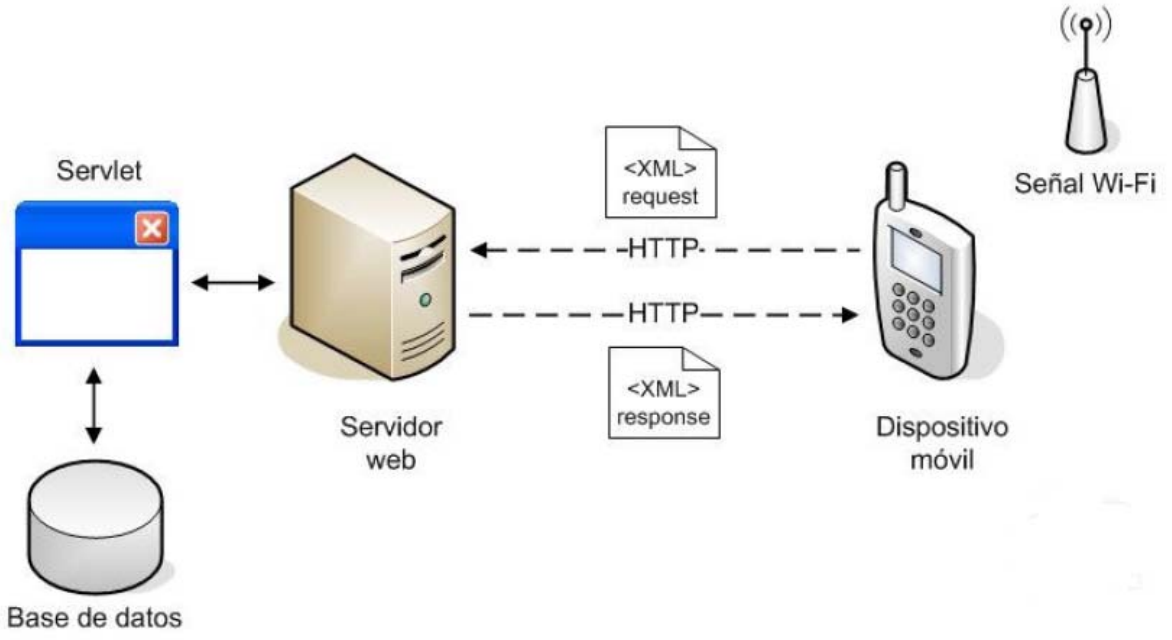


Fig. 4.4 Arquitectura del sistema [diseño propio]

En la Fig. 4.4 se observan los siguientes componentes:

- **Dispositivo móvil**: dispositivo con el sistema Android corriendo y donde está instalada la aplicación principal. A través de la conexión WIFI, realiza intercambios de información con el servidor web.
- **Documentos XML**: el dispositivo móvil realiza intercambios de documentos XML con el servidor utilizando el protocolo HTTP.
- **Servidor web**: permanece siempre a la espera de conexiones por parte de los diferentes usuarios de MuseoFiee. Aloja el servlet que procesa las peticiones y respuestas sobre XML.
- **Servlet**: pequeño programa en Java que recibe peticiones, consulta la base de datos, y construye y envía las respuestas XML de vuelta al dispositivo móvil.
- **Base de datos**: la base de datos almacena toda la información en formato de texto, imagen, audio y video y es consultada por el servlet.

4.3.4 Desarrollo e implementación

Luego de explicar el principio de funcionamiento, características de la aplicación y los pormenores del diseño, a continuación abordaremos los temas relacionados con la implementación.

4.3.4.1 Estructura y acceso a la base de datos

El método utilizado para conectar un dispositivo móvil con Android a un servidor de base de datos remote como MySQL, es colocar en el medio algún tipo de servicio. Dado que MySQL es utilizado con PHP, entonces la forma más sencilla y obvia es escribir un script en PHP para manejar la base de datos y ejecutar este script utilizando el protocol HTTP desde el dispositivo con Android.

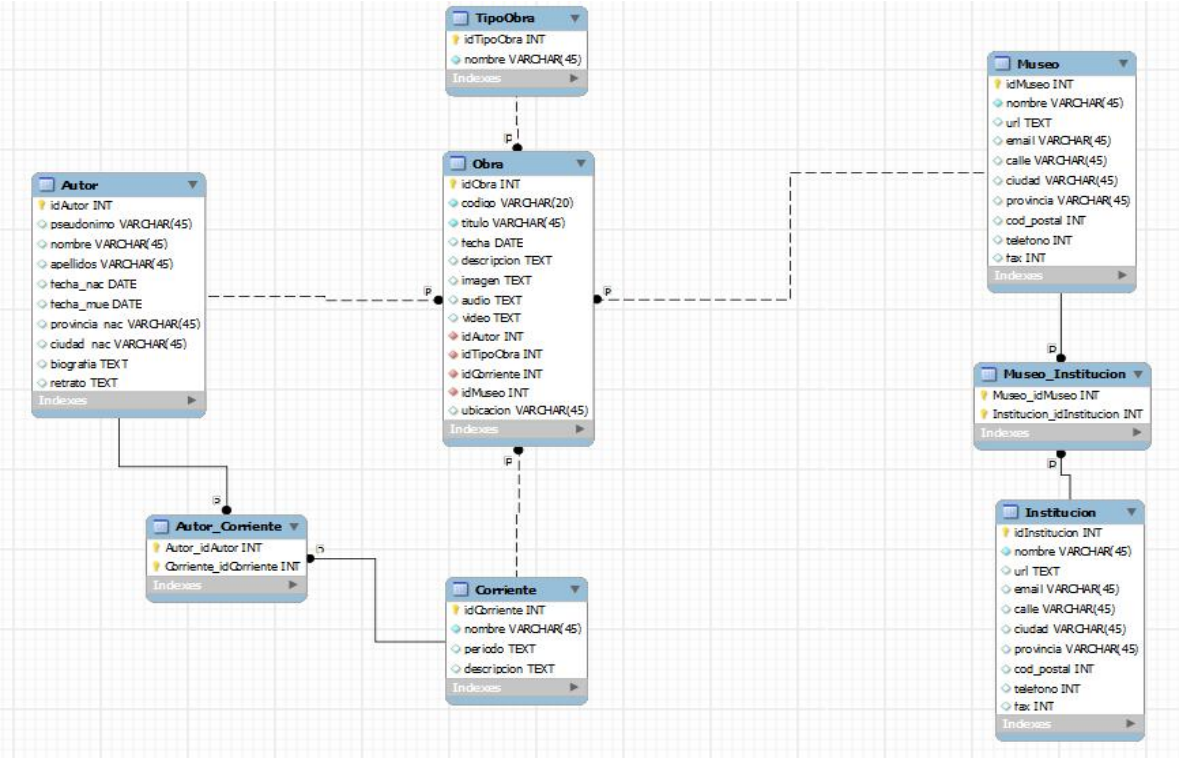


Fig. 4.5 Estructura de la base de datos [diseño propio]

Código PHP para acceder a la base de datos:

```
<?php
mysql_connect("host","username","password");
```

```
mysql_select_db("MuseoBD");
 $q=mysql_query("SELECT * FROM texto);
while($e=mysql_fetch_assoc($q))
        $output[]=$e;
print(json_encode($output));
mysql_close();
?>
```

Extracto incompleto del código para conexión http, utilizando el archivo **MostrarInfo.java**:

```
public class MostrarInfo extends Activity {

        Button btnVerTexto;
        Button btnVerImagen;
        Button btnRepAudio;
        Button btnRepVideo;
        **********
        TextView txtInfoTitulo;
        Button btnSalir;
        Button btnPrincipal;
TextView mProgressText;
        private ProgressDialog pDialog;
        String cadena;
        String urls[] = null;
        Httppostaux post;

String IP_Server = "http://172.17.9.230:8080/";//IP del Server
        String URL_connect =  IP_Server + "acces.php";

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                urls = new String[5];
                setContentView(R.layout.activity_mostrar_info);
                post = new Httppostaux();

**************************

Intent i = new Intent(MostrarInfo.this, MostrarTexto.class);
                                        i.putExtra("url", urls[1]);
                                        i.putExtra("cadena",cadena);
                                        startActivity(i);

**********TEXTO

btnVerTexto.setOnClickListener(new OnClickListener() {

                                @Override
```

```
                        public void onClick(View v) {
        Intent i = new Intent(MostrarInfo.this, MostrarTexto.class);
                                i.putExtra("url", urls[1]);
                                i.putExtra("cadena",cadena);
                                startActivity(i);
```
**********IMAGEN

```
btnVerImagen.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                Intent i = new Intent(MostrarInfo.this, MostrarImagen.class);
                                //i.putExtra("url", "http://jonsegador.com/wp-
content/apezz.png");
                                i.putExtra("url", IP_Server + urls[2]);
                                i.putExtra("cadena",cadena);
                                startActivity(i);
```

**********AUDIO

```
btnRepAudio.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {

                                Intent i = new Intent(android.content.Intent.ACTION_VIEW);

        //i.setDataAndType(Uri.parse("http://www.infocodqu.vacau.com/Ma.mp3"),
"audio/*");
                                i.setDataAndType(Uri.parse(IP_Server + urls[3]), "audio/*");
                                MostrarInfo.this.startActivity(i);
```

**********VIDEO

```
btnRepVideo.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                        /*
                                Intent videoClient = new Intent(Intent.ACTION_VIEW);
                //videoClient.setData(Uri.parse("http://www.youtube.com/watch?v=LKL-
efbiIAM"));
                videoClient.setData(Uri.parse(urls[4]));
**********

                        videoClient.setClassName("com.google.android.youtube",
"com.google.android.youtube.WatchActivity");
                         startActivity(videoClient);
```

```
**********MOSTRAR TEXTO
//          Intent i = new Intent(MostrarInfo.this, MostrarTexto.class);
```

4.3.4.2 Construcción del menú principal y la interfaz de usuario

Para poder acceder a algunas de las opciones ofrecidas por la aplicación, el usuario debe utilizar el menú principal de la aplicación. Este menú aparece como respuesta al escaner de código QR. Este es un estracto del archivo principal.java. El código completo se encuentra en el apéndice B (listado de programas).

Principal.java

```
//Set action to button
                    button.setOnClickListener( new OnClickListener() {
                              public void onClick(View v) {
                              //Create a new Intent to send to QR Droid
Intent qrDroid = new Intent( "la.droid.qr.scan" ); //Set action "la.droid.qr.scan"
                                        qrDroid.putExtra( "la.droid.qr.complete" , true);
                                  startActivityForResult(qrDroid, 0);
```

El diseño de la interfaz de usuario se realiza en la carpeta res/menú y res/values. Se muestra un estracto del código utilizado. El archivo strings.xml muestra los títulos del menú principal. Los códigos completos están en el apéndice B (listado de programas)

activity_principal.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

activity_qrscanner.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

activity_mostrar_info.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

activity_mostrar_text.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

activity_mostrar_imagen.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

strings.xml

```
<resources>

   <string name="app_name">QRInfo</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_mostrar_info">MostrarInfo</string>
   <string name="title_activity_mostrar_texto">MostrarTexto</string>
   <string name="title_activity_mostrar_imagen">MostrarImagen</string>
   <string name="title_activity_qrscanner">QRScanner</string>
   <string name="title_activity_principal">Principal</string>

</resources>
```

Cada elemento declarado tiene como nombre el propio de la clase a la que pretende instanciar, en este caso **<TextView>.** Dependiendo de dicha clase, los elementos podrán declarar unos atributos u otros, aunque aquí los atributos son muy similares entre ambos. Se pueden observar los siguientes atributos:

- **android:id**: todo elemento declarado debe tener un identificador único, escrito en el atributo android:id. Este identificador es a través del cual se referencia el elemento en el código fuente, y es el nombre con el que aparece declarado el recurso dentro del archivo **"R.java**" (se recordará que dicho archivo mantiene una sincronización con cualquier elemento declarado como recurso en la carpeta "\res", de forma que permite a las demás clases de la aplicación Android poder utilizar el recursos deseado simplemente nombrándolo).
- **android:layout_width**: indica la anchura que debe tener este elemento. Si se especifica wrap_content, entonces ocupará tanto espacio como necesite; por otra parte, una configuración como fill_parent indica al widget que ocupe tanto espacio como tenga disponible. Otra alternativa, es especificar exactamente el tamaño deseado, ya sea en píxeles (px), en píxeles escalados (ps), en pulgadas (in), milímetros (mm).
- android:layout_height: indica la altura del elemento. Se comporta igual que el atributo layout_width.
- **android:layout_marginTop**: especifica el margen superior que se desea establecer para el elemento. Se comporta igual que el atributo layout_width.
- **android:layout_marginLeft**: especifica el margen izquierdo que se desea establecer para el elemento. Se comporta igual que el atributo layout_width.
- **android:Text**: el texto que, por defecto, debe mostrar el elemento.
- **android:textSize**: el tamaño del texto.
- **android:maxLength**: el número máximo de caracteres.

Cada elemento de interfaz cuenta con su propia familia de atributos, dando así un alto grado de libertad al desarrollador para configurar sus widgets de la forma que considere más oportuna.

Además de elementos de interfaz, Android también contempla algunas vistas, también llamados diseños, que agrupan los distintos widgets y los ordenan siguiendo unos patrones frecuentemente utilizados en las aplicaciones.
La clase LinearLayout permite colocar los elementos de una interfaz de forma consecutiva, ya sea imaginando para ellos unas columnas en la pantalla (orden horizontal) o unas filas (orden vertical).

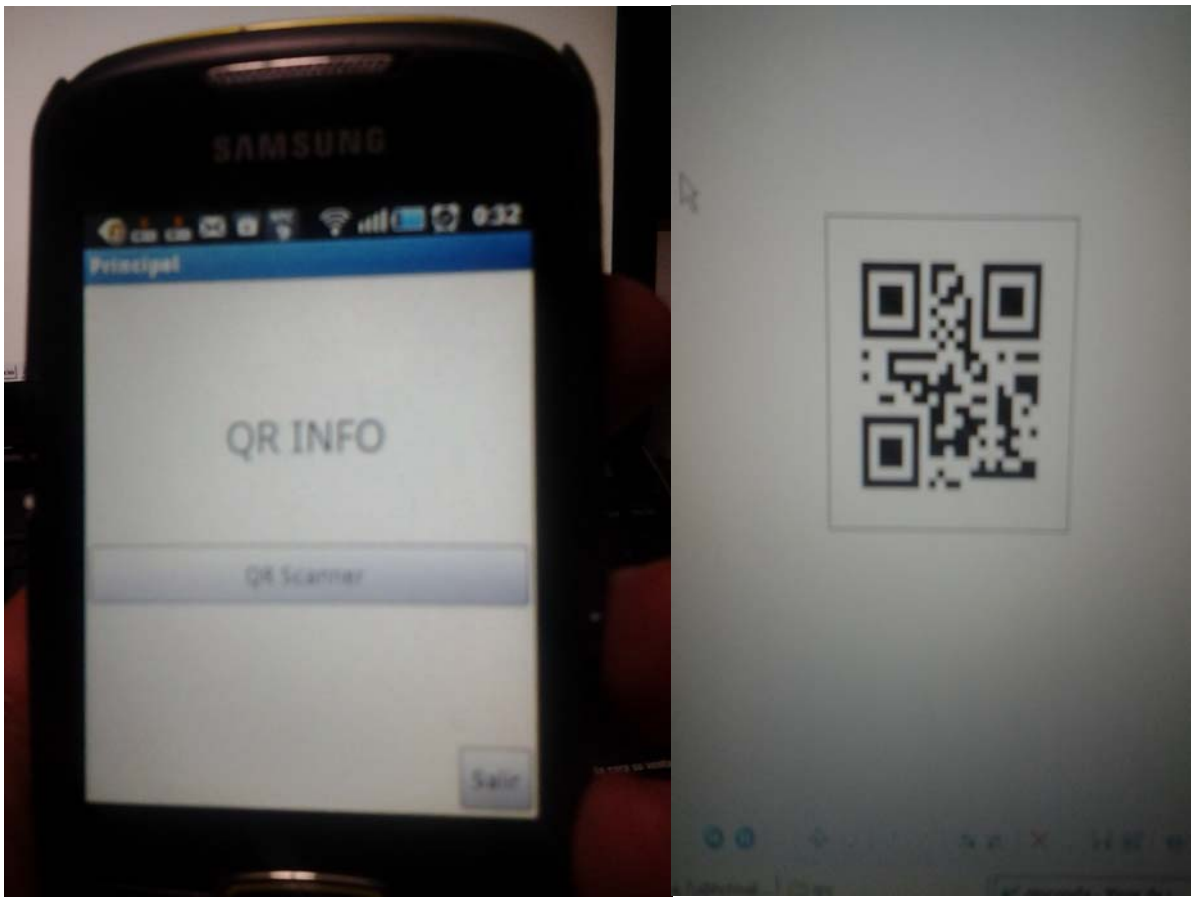


Fig. 4.6 Escáner del QR code desde la aplicación [diseño propio]

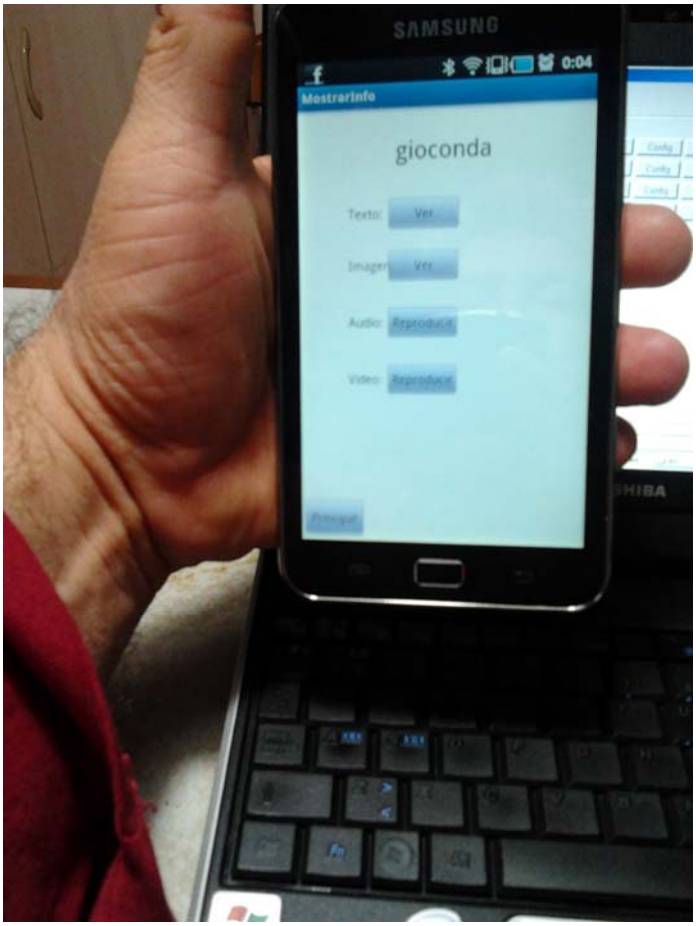


Fig. 4.7 Menú principal de la aplicación [diseño propio]

4.3.4.3 Lanzar una nueva “Activity”

Toda nueva Activity debe ser lanzada a través de un Intent, que especifica qué es lo que se quiere hacer y con qué datos debe hacer. Habitualmente, un Intent es una forma de delegar determinadas acciones en otras aplicaciones instaladas en el sistema y que, obviamente, tengan la capacidad para llevarlas a cabo. Sin embargo, también es posible especificar que un Intent debe ser realizado por una determinada instancia de una clase.

Existen dos formas para lanzar una nueva actividad en Android. Si se quiere lanzar la Activity sin más, sin esperar ningún resultado final, se utiliza el método **startActivity().** Si se espera un valor devuelto por la **Activity**, entonces se lanza con el método **startActivityForResult(),** de forma que al terminar dicha actividad se invocará el método **onActivityResult(),** que permite manejar el resultado obtenido.

Debemoos recordar que una **Activity** debe ser declarada explícitamente en el archivo “**AndroidManifest.xml**”. Esto es importante para poder utilizarlo.

4.4 Requerimientos no funcionales

- Aspecto de la Interfaz de Usuario. La interfaz debe ser atractiva, y fácil de usar.
- Tolerancia a fallos. El sistema debe poder recuperarse ante fallos.
- Seguridad. El sistema debe tener al menos consideraciones mínimas de seguridad.
- Debe permitir mantenibilidad para subsecuentes desarrollos.
- El sistema debe ser concurrente, es decir, el sistema debe soportar que un mismo programa sea usado por dos o más usuarios distintos.
- El sistema debe ser portable, capaz de ser instalado en diferentes plataformas.

4.5 Diseño de la red inalámbrica del museo

El museo cuenta con varios ambientes repartidos en 2 pisos. De acuerdo a los planos y la cantidad de personas por cada ambiente se considera en el diseño el uso de 30 Access Points IEEE 802.11n. Estos APs estarán controlados mediante un Controlador Central Wireless con soporte de roaming. Adicionalmente, todos los AP estarán conectados a un switch con tecnología PoE (Power over Ethernet), para recibir la energía eléctrica.

4.5.1 Análisis de cobertura de señal WIFI en el museo

El objetivo del análisis de cobertura de la señal WIFI, es determinar los puntos muertos, es decir, zonas donde la señal inalámbrica está fuera del rango mínimo de potencia de recepción. Mediante el uso de un Access Point ubicados en los puntos AP1, AP2 y AP3, y luego con la ayuda de un dispositivo móvil Samsung Galaxy mini GT-S5570L y con un programa de aplicación como WIFI Analyzer para plataforma Android se pudo determinar la potencia de la señal recibida en dBm para cada punto indicado en el gráfico.

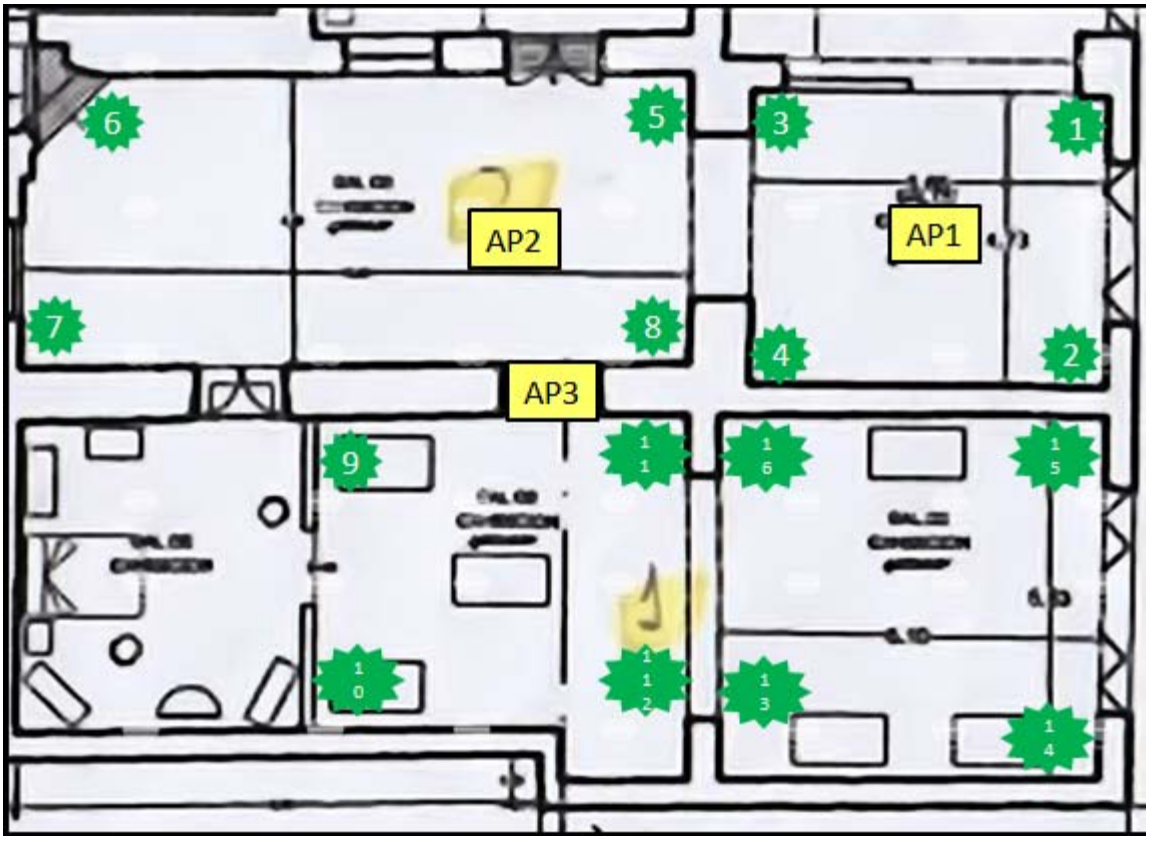


Fig 4.8 Análisis de cobertura WIFI primer piso Museo [diseño propio]

Tabla 4.1 Resultados de la prueba de cobertura AP-1 primer piso [diseño propio]

AP1 (ACCESS POINT 1)	
PUNTO DE PRUEBA	**POTENCIA RX (dBm)**
1	**-37**
2	**-36**
3	**-42**
4	**-40**
5	**-56**
6	**-51**
7	**-43**

8	**-70**
9	**-78**
10	**-72**
11	**-81**
12	**-74**
13	**-70**
14	**-73**
15	**-80**
16	**-79**

Tabla 4.2 Resultados prueba de cobertura AP-2 primer piso [diseño propio]

AP 2 (ACCESS POINT 2)	
PUNTO DE PRUEBA	**POTENCIA RX (dBm)**
1	**-48**
2	**-64**
3	**-65**
4	**-64**
5	**-42**
6	**-52**
7	**-41**
8	**-43**

9	**-78**
10	**-76**
11	**-69**
12	**-65**
13	**-68**
14	**-68**
15	**-80**
16	**-80**

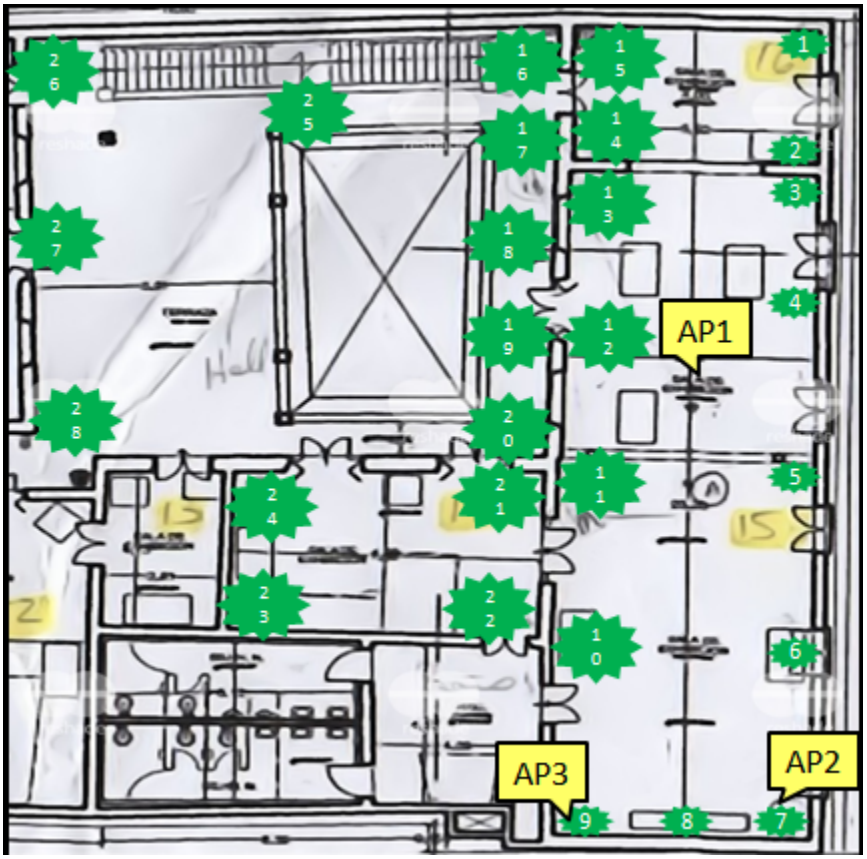


Fig 4.9 Análisis de cobertura WIFI segundo piso Museo [diseño propio]

Tabla 4.3 Resultados prueba de cobertura AP-1 segundo piso [Diseño propio]

AP1 (ACCESS POINT 1)	
PUNTO DE PRUEBA	**POTENCIA RX (dBm)**
1	**-66**
2	**-61**
3	**-53**
4	**-53**
5	**-41**
6	**-44**
7	**-55**
8	**-41**
9	**-44**
10	**-42**
11	**-41**
12	**-44**
13	**-45**
14	**-54**
15	**-61**
16	**-79**
17	**-58**
18	**-67.5**
19	**-62.5**
22	**-46**
23	**-53**
24	**-60**
25	**-60**
26	**-62**
27	**-62**
28	**-65**

Tabla 4.4 Resultados prueba de cobertura AP-2 segundo piso [diseño propio]

AP 2 (ACCESS POINT 2)	
PUNTO DE PRUEBA	**POTENCIA RX (dBm)**
11	**-42**
12	**-54**
13	**-55**
15	**-65**
16	**-76**
25	**-71**
26	**-67**
27	**-76**

Tabla 4.5 Resultados prueba de cobertura AP-3 segundo piso [diseño propio]

AP 3 (ACCESS POINT 3	
PUNTO DE PRUEBA	**POTENCIA RX (dBm)**
15	**-56**
16	**-79**
23	**-72**
25	**-64**
26	**-69**
27	**-77**

Las pruebas de cobertura, se repitieron para el primer y segundo piso del museo. La idea es determinar la cantidad de Access Points para cubrir todas las areas del museo con la suficiente potencia de señal y velocidad de transmisión. Sin embargo debido a que los Access Points soportan hasta 15 usuarios concurrentes, se ha implementado la red inalámbrica con 15 Access Points por cada piso lo que nos da un total de 30 Access Point y un total de 450 usuarios concurrentes.

4.5.2 Arquitectura de la plataforma inalámbrica.

La plataforma está basada en un conjunto de Access Points distribuidos en los dos pisos del museo. Los Access points (AP) están conectados a la red cableada por intermedio de Switches PoE, los que a su vez los energizan. Esta red inalámbrica es supervisada por un controlador central de conexión y acceso al sistema. Del análisis realizado se determina un parque inicial de 30 Access points, 2 Switches PoE de 24 puertos, antenas omnidireccionales de 9 dBi, antenas omnidireccionales de 12 dBi, antenas sectoriales de 12 dBi. Se utilizan los Access Point que cumplan con el estándar b/g/n y deben proporcionar conectividad a velocidades de hasta 300 Mbps, lo que constituye una característica importante de los mismos para estos escenarios.

4.5.2.1 Access Point (Punto de Acceso).

Access Points, ó puntos de acceso, son los equipos encargados de proveer de acceso a los recursos informáticos disponibles en los servidores del museo.

- Las especificaciones principales son:
- Los equipos deben estar basados en el estándar 802.11b/g/n.
- Debe soportar conexión a la red LAN vía interfaz Ethernet.
- Debe soportar alimentación eléctrica a través del cable Ethernet o por energía local en el equipo.
- Debe soportar tecnologías de seguridad inalámbrica como WEP, WPA-PSK, WPA-TKIP, WPA2 AES, 802.11i.
- Debe soportar alimentación por la red de datos: Power over Ethernet (802.3af)
- La temperatura de operación de 0 a 50 ºC con PoE, y de 0º a 40ºC con AC.
- Debe soportar hasta 20 estaciones concurrentes.
- Debe soportar 802.11e y 802.1Q

4.5.2.2 Switches PoE de 24 puertos y switch de 8 puertos.

- Switch PoE de 24 puertos Administrable
- Switch de 8 puertos administrable.
- Debe proporcionar energía a dispositivos de red como puntos de acceso inalámbricos, Puertos Ethernet auto-MDIX 10/100Mbps PoE con auto-negociación.
- Debe soportar módulo opcional de 2 puertos SFP para extender la capacidad del switch para conectarse a tecnologías Gigabit a fibra óptica.
- Debe incorporar características QoS con cuatro diferentes clasificaciones de tráfico.
- Debe soportar puerto de captura en espejo para monitoreo de tráfico.
- Debe soportar la característica VLAN basada en puerto y VLAN IEEE802.1Q, para aislar el tráfico de diferentes usuarios, proporcionando mejor seguridad y mejor rendimiento
- Debe soportar la autenticación de usuario para cada intento de acceso a la red
- Debe soportar Agrupación de puertos (Port Trunking) con LACP 802.3ad y QoS 802.1p
- Debe soportar stacking virtual.
- Debe limitar el número de estaciones por cada puerto mediante direcciones MAC.
- Debe soportar el monitoreo visual mediante LEDs.
- Debe soportar administración por interfaz de usuario basada en web.

4.5.2.3 Antenas Omnidirecionales de 12 dBi

- Para aplicaciones 802.11b/g/n en un rango de frecuencia de 2.4 GHz
- Operación Omni-direccional para cobertura extendida punto a multi-punto
- Ganancia mínima 12 dBi.
- De proporcionar cable de extensión para fácil instalación con N-Plug a RPSMA de 3 metros.
- Impedancia de 50 ohm
- Potencia de 1 W
- Temperatura operacional:-30º C + 60ºC. Uso en interiores

4.5.2.4 Controlador Central Wireless.

- Gestión centralizada de la red interna de Access Points
- Administración y control centralizado de 200 ó más Access Points
- Soporte de 3,000 cuentas locales y 3,000 cuentas por demanda.
- Despliegue simplificado y confiable de conexiones a Internet.
- Capacidades de seguridad de redes.
- Control de seguridad por zonas.
- Soporte de control de acceso seguro y gestión de usuarios.
- Soporte de cuotas de acceso a Internet para usuarios.
- Control de Hotspot basado en ubicación del Access Point.
- Soporte de sistemas de gestión de propiedad y cargos por conexión a Internet.

4.5.2.5 Tarjeta de red de los dispositivos móviles.

- Soporte del estándar IEEE 802.11n, IEEE 802.11g
- Velocidad de transferencia en IEEE 802.11n 300 Mbps
- Velocidad de transferencia en IEEE 802.11g 54 Mbps
- Frecuencia de operación 2.4 a 2.485 GHz.
- Potencia de transmisión inalámbrica de 20 dBm
- Sensibilidad de recepción a 108 metros -68 dBm al 10% de tasa de pérdidas de paquetes (@10% PER)
- Sensibilidad de recepción a 6 metros -88dBm@105 PER
- Seguridad WPA/WPA2, WPA-PSK/WPA2-PSK (TKIP/AES)

4.5.2.6 Topología de la red de transporte.

Los Access points estarán conectados a los switches PoE a través de cable STP/FTP categoría 6 con soportee de hasta 1 Gbps.

Los switches PoE estarán conectados al controlador central wireless mediante cable STP/FTP de 1 Gbps como mínimo.

Los servidores estarán conectados mediante un switch PoE de 8 puertos de las mismas características técnicas que el switch de 24 puertos.

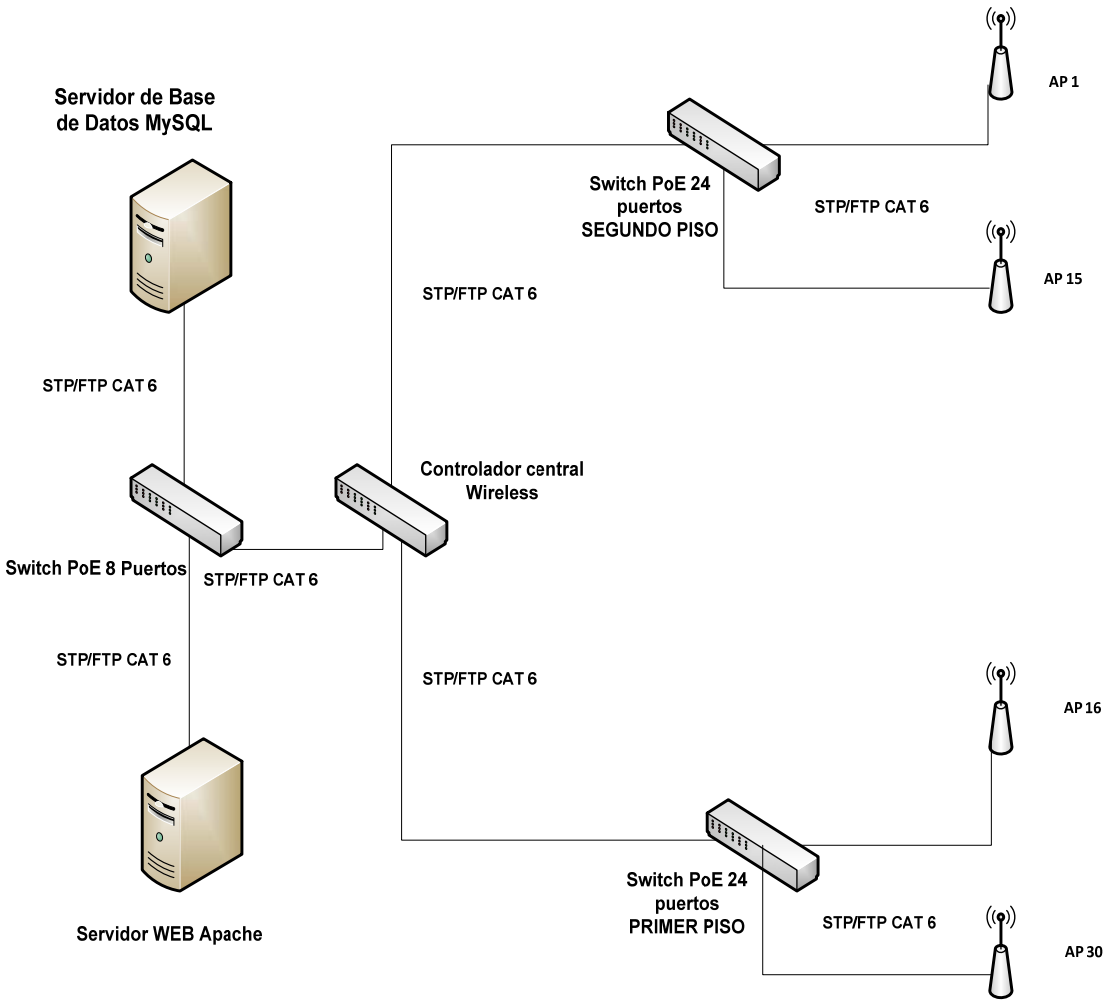


Fig 4.10 Topología de la red de transporte para el Museo [diseño propio]

4.5.2.7 Planos de ubicación de los puntos de acceso (AP) de la red inalámbrica del museo [Ver Anexo A.3]

CAPÍTULO V

ANÁLISIS DE RESULTADOS

5.1 Pruebas y validación del sistema piloto

Para realizar las pruebas del sistema piloto, se utilizó un servidor con plataforma Linux con la distribución Ubuntu Server 11.10. En éste servidor implementamos la base de datos MySQL 5.0 y el Servidor Web Apache 2.0, que soporta **Servlets**, servicios web (JAX-WS), arquitectura java para enlaces XML.

El servidor está conectado a la red de datos del museo mediante una interface de red de 1Gbps. El switch 100/1000 Mbps que conecta el servidor, también permite la conexión y la administración de los puntos de acceso de la red inalámbrica (AP), ubicados en los 2 pisos del museo y que permitirá que los visitantes provistos de dispositivos móviles y audífonos, puedan tener acceso a la información de texto, imagen, audio y video a solicitud como información adicional a las obras que está observando en el museo.

Se realizará el envío de contenidos desde el servidor local, instalado en el museo, hasta el dispositivo móvil a través de la red inalámbrica WIFI implementada en el museo.

5.2 Pruebas de envío de contenidos

La distribución de contenidos por la red inalámbrica WIFI, requiere la lectura de un código QR, adosado a la obra, que el usuario está observando y no es invasiva porque el usuario decide obtener más información relacionada con la obra visualizada.

El museo entregará a cada visitante un dispositivo móvil con plataforma operativa Android y audífonos. Cada dispositivo móvil contará con una aplicación llamada "**Principal.apk**", la misma que fue copiada de la computadora al dispositivo móvil utilizando el cable USB del dispositivo móvil. Una aplicación empaquetada y comprimida en Android, tiene la extensión .**APK** (**A**pplication **P**ac**K**age File), que es una variante del formato **JAR** (**J**ava

ARchives) de java. Una aplicación **.apk**, contiene los siguientes archivos: AndroidManifest.xml, clases.dex, resources.arsc, carpeta res y carpeta META-INF.

Para realizar las pruebas de funcionamiento de la aplicación, se utilizaron dos dispositivos móviles Marca Samsung. Primer modelo Galaxy mini GT-S5570L. Segundo modelo Galaxy S.

5.3 Escritorio principal de Android

Se realizó la instalación del aplicativo **Principal.apk** en la pantalla del dispositivo móvil marca **Samsung** modelo **Galaxy mini GT-S5570L**. Tal como se muestra en la Fig. 5.1

Fig. 5.1 Escritorio de equipo Android con aplicativo principal [diseño propio]

Cuando el usuario selecciona la aplicación “Principal” mostrada en el escritorio del dispositivo móvil, se muestra en la pantalla la opción QR Scanner, la misma que luego de ser seleccionada, permitirá realizar el escaneo del QR Code, adosado al costado de la obra visualizada, utilizando para ello, la cámara del dispositivo móvil.

Fig. 5.2 Opción QR Scanner [diseño propio]

El código QR escaneado por la cámara del dispositivo móvil, se conecta con el servidor Web vía URL, que le permitirá al usuario, obtener mayor información sobre la obra que está visualizando. La información adicional se mostrará en formato de texto, imagen, audio o video. Cada dispositivo móvil cuenta con audífonos para personalizar el contenido y evitar interferencias de audio que molesten a los demás visitantes.

Los formatos de imagen soportados por la aplicación en Android son JPEG, GIF, PNG, BMP. Los formatos de audio soportados son: MP3, MIDI, Vorbis, PCM/WAVE. Los formatos de video que soporta son: H.263, H.264, MPEG-4.

El nivel de sofisticación del dispositivo móvil influirá en la calidad de la imagen, sonido y video.

5.4 Menú principal del sistema

Se realizó la prueba de visualización del menú principal luego de la lectura del código QR, comprobándose que dicha tarea cumple con las funciones previstas en el diseño.

El resultado se muestra en la Fig. 5.3

Fig. 5.3 Menú principal del QR code escaneado [diseño propio]

5.5 Escenario de prueba de texto

Se realizó la prueba de lectura de texto en la pantalla del dispositivo móvil, comprobándose que dicha tarea cumple con las funciones previstas en el diseño; así mismo, se pueden utilizar los controles de “avance”, “retroceso” táctil.

El resultado se muestra en la Fig. 5.4.

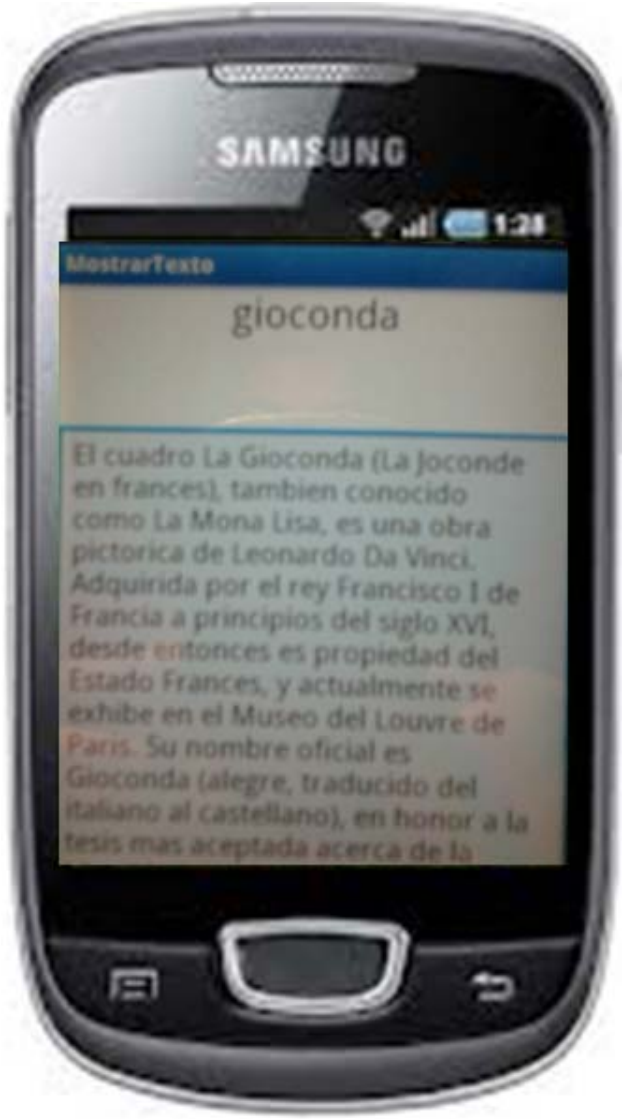


Fig. 5.4 Escenario de prueba de texto [diseño propio]

5.6 Escenario de prueba de imagen

Se realizó la prueba de visualización de imágenes, comprobándose que dicha tarea cumple con las funciones previstas en el diseño.

El resultado se muestra en la Fig. 5.5

Fig. 5.5 Escenario de prueba de imagen [diseño propio]

5.7 Escenario de prueba de audio

Se realizó la prueba de reproducción del sonido, utilizando el programa reproductor de audio del dispositivo Android, comprobándose que dicha reproducción cumple con las funciones previstas en el diseño; así mismo, se muestran y utilizan los controles de “play”, “stop”, “pause” y el estado de reproducción.

El resultado se muestra en la Fig. 5.6.

Fig. 5.6 Escenario de prueba de audio [diseño propio]

5.8 Escenario de prueba de vídeo

Se realizó la prueba de reproducción del video, utilizando el reproductor estándar del dispositivo Android, comprobándose que dicha reproducción cumple con las funciones previstas en el diseño.

El resultado se muestra en la Fig. 5.7

Fig. 5.7 Escenario de prueba de video [diseño propio]

5.9 Proceso de Captura y resultados fotográficos obtenidos

En la fotografía se muestran los equipos utilizados para las pruebas: Una computadora como servidor Web Apache, Servidor de base de datos MySQL y repositorio de contenidos. También se utilizó una netbook marca TOSHIBA modelo NB100 con CPU N270 de 1.6 GHz y 2 GiB de memoria RAM, como equipo alternativo al servidor Web Apache y Base de datos MySQL. Los dispositivos móviles son de marca Samsung modelo Galaxy mini GT-S5570L y Galaxy S; se comunican con el servidor mediante un Access Point que forma parte de la red inalámbrica de Infraestructura del museo. El Acces Point es de marca D-Link modelo DIR-400. La computadora es de marca Olidata compatible con CPU Intel Core 2 Duo de 2.4 GHz y 4 GiB de memoria RAM. La foto fue tomada por el móvil Galaxy S.

Fig. 5.8 Fotografía de equipos de prueba: AP, Móvil, PC [diseño propio]

5.9.1 Programa principal en el dispositivo móvil

En la fotografía se muestra el programa principal de la aplicación desarrollada, la misma que fue transferida vía cable USB desde la computadora. Dicho programa fue diseñado utilizando una herramienta de desarrollo como Eclipse, y comprobado en un Emulador Android que forma parte de Eclipse.

Fig. 5.9 Fotografía de programa principal en Galaxy S [diseño propio]

5.9.2 Generación de códigos QR

Mediante el uso del programa generador de códigos QR de Kaywa (http://qrcode.kaywa.com), se generaron tres códigos bidimensionales de prueba: ecce homo, Gioconda y la encajera. Cada Código generado representa una dirección URL, que apunta a un servidor de Aplicación Web Apache, y desde aquí se comunica con la Base de datos donde reside la información que será visualizada o mostrada o escuchada por el usuario del dispositivo móvil.

ECCE HOMO: http://172.17.9.230:8080/ecce homo

Fig. 5.10 QR code del cuadro ecce homo [20]

GIOCONDA: http://172.17.9.230:8080/gioconda

Fig. 5.11 QR code del cuadro Gioconda [20]

LA ENCAJERA: http://172.17.9.230:8080/laencajera

Fig. 5.12 QR code del cuadro la encajera [20]

5.9.3 Activación del programa principal

Se selecciona la aplicación Principal del escritorio del dispositivo móvil y luego se activa QR Scanner para apuntar el QR code con la cámara del dispositivo móvil.

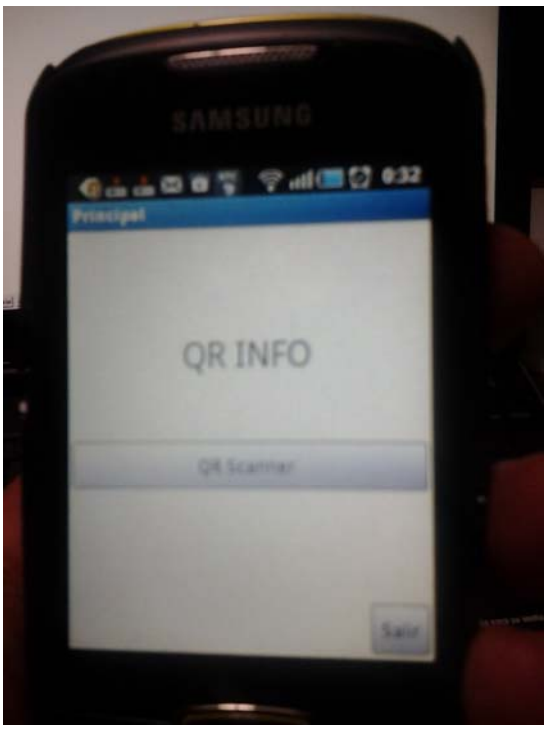


Fig. 5.13 Opción QR Scanner del programa principal [diseño propio]

5.9.4 Lectura del QR code

Se escanea el código QR generado y que está adosado al costado de cada obra del museo, con la cámara del dispositivo móvil. El código bidimensional de ejemplo se muestra en la fotografía de la Fig. 5.14

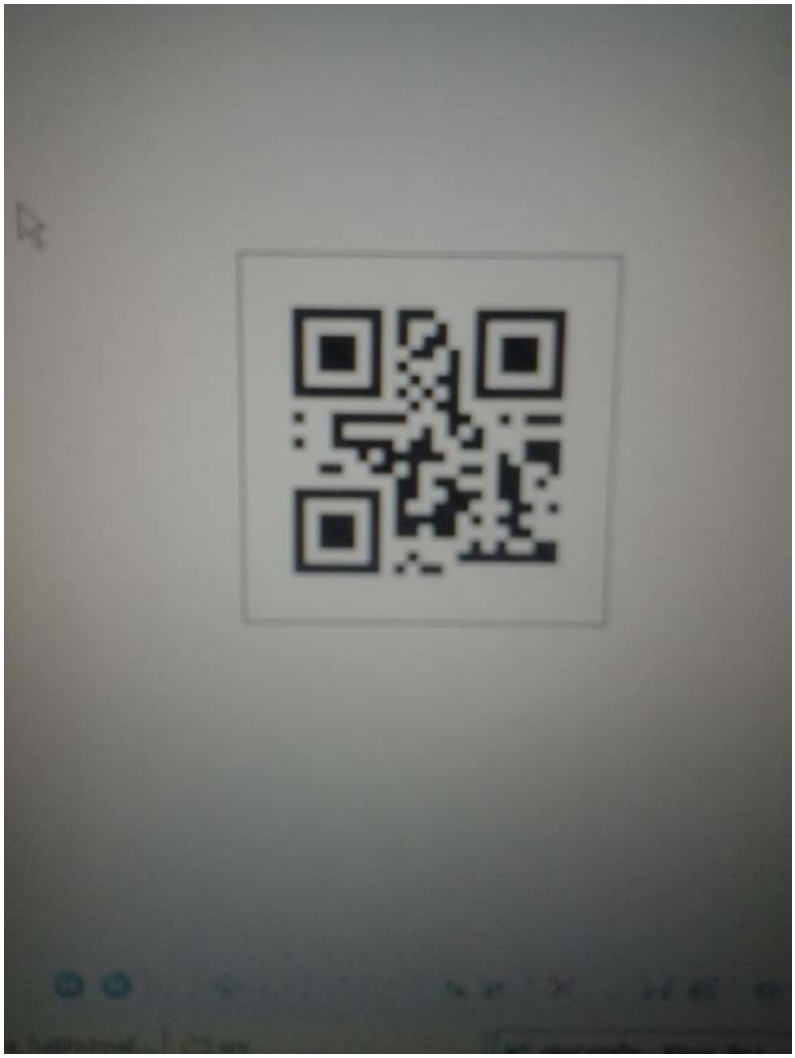


Fig. 5.14 QR code de la gioconda [20]

5.9.5 Menú principal luego de la captura

Luego de realizar el escaneo del código QR, se puede apreciar el menú principal de la obra, para nuestro ejemplo tenemos la Gioconda. Aquí se observa el menú completo de la aplicación. El usuario puede seleccionar el formato de la información que desea visualizar o escuchar: texto, imagen, audio y video.

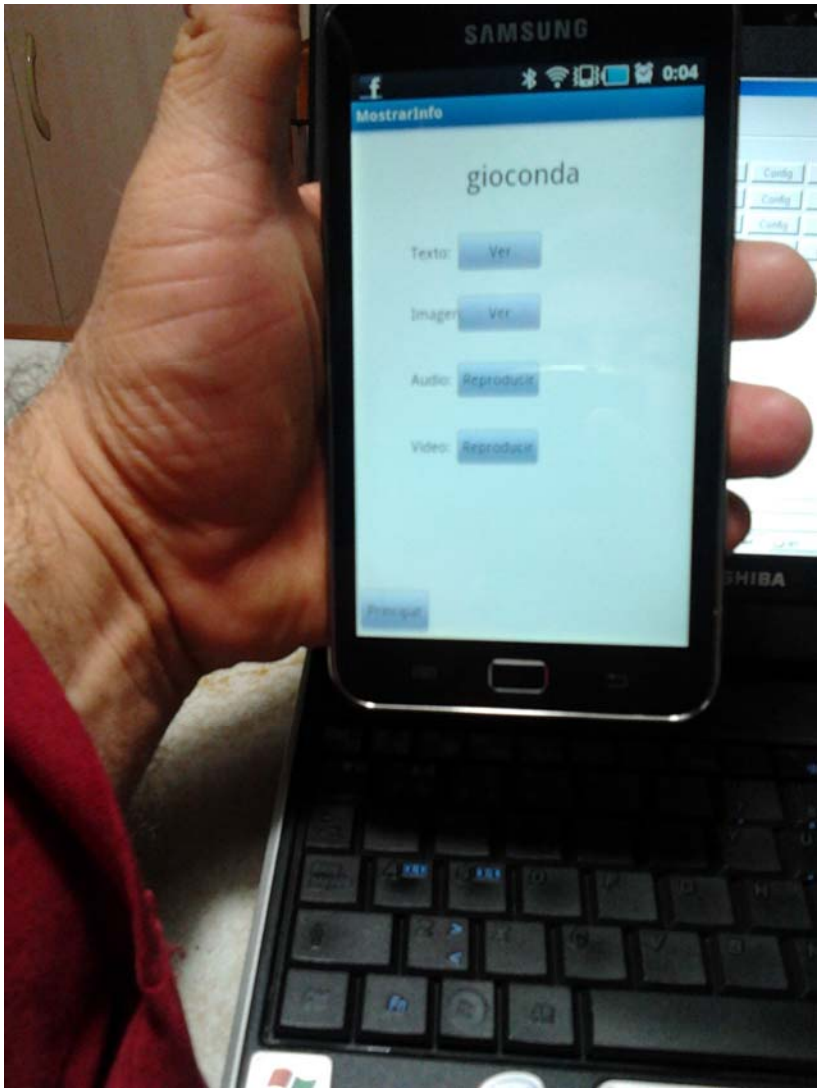


Fig. 5.15 Menú principal de la gioconda [diseño propio]

5.9.6 Prueba de Texto

Si seleccionamos Ver texto, nos mostrará lo siguiente:

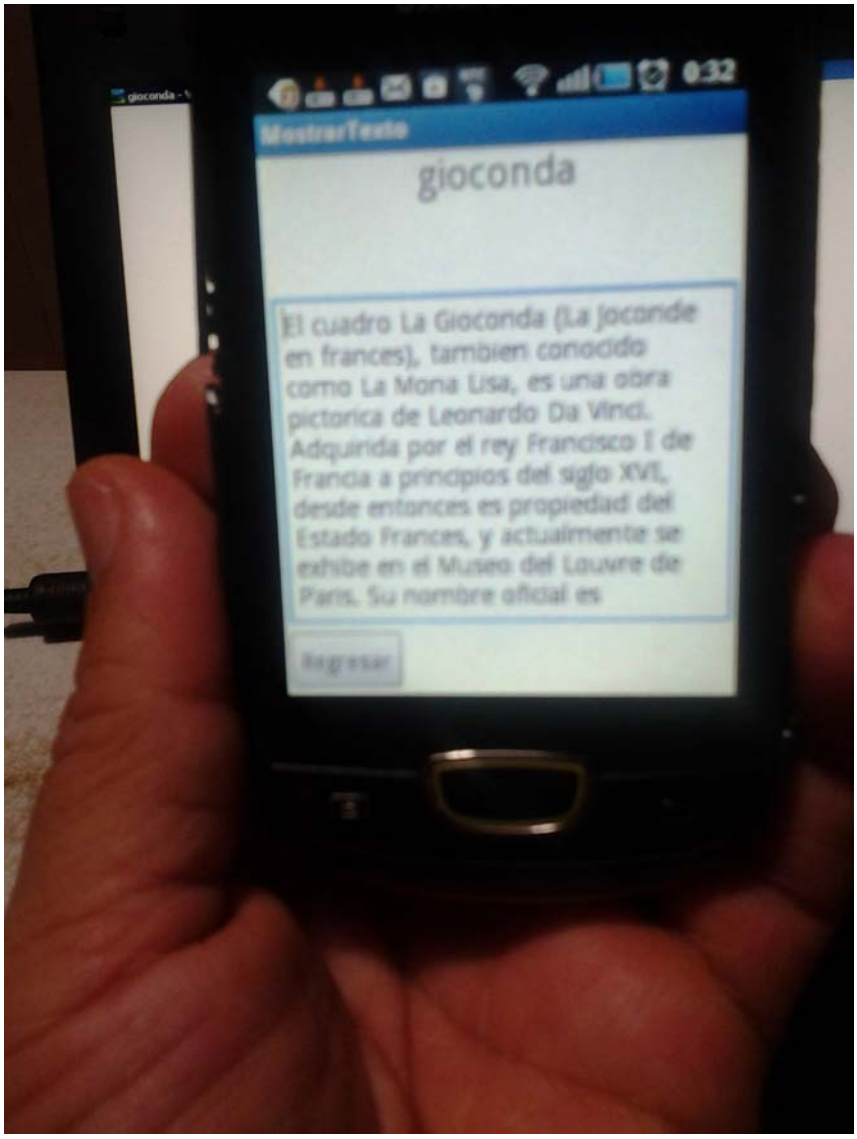


Fig. 5.16 Submenú Ver Texto de la Gioconda [diseño propio]

5.9.7 Prueba de Imagen

Si seleccionamos Ver Imagen, nos mostrará lo siguiente

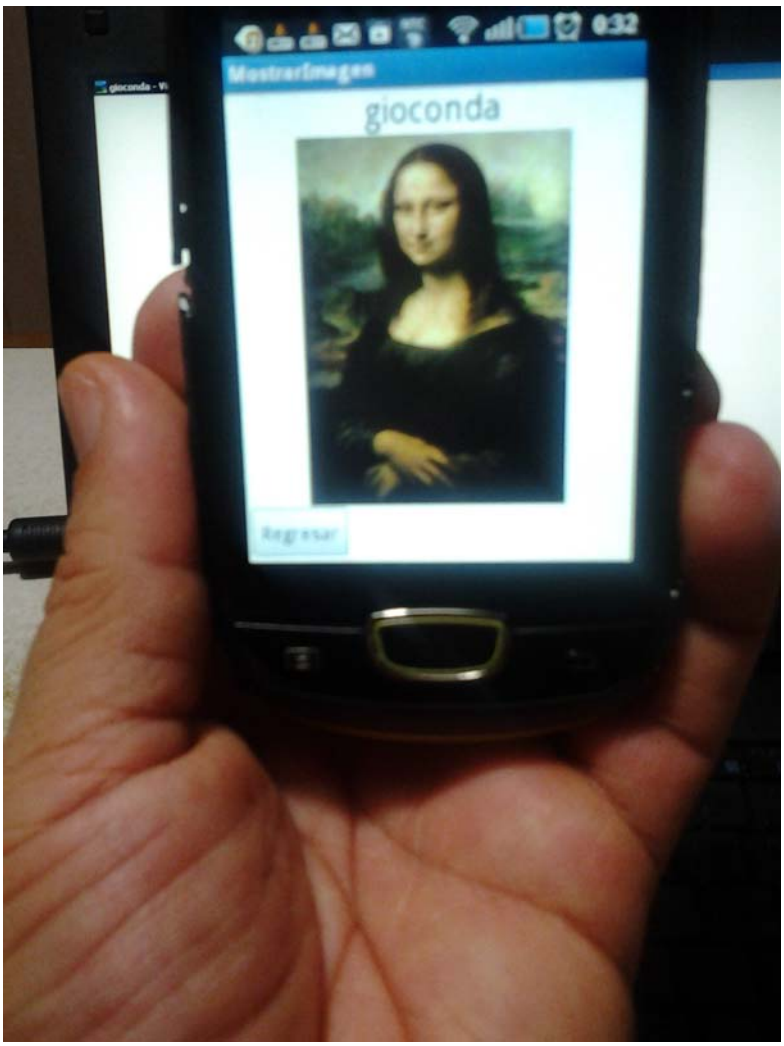


Fig. 5.17 Submenú Ver Imagen de la Gioconda [diseño propio]

5.9.8 Prueba de Audio

Si seleccionamos Reproducir Audio, nos mostrará la siguiente pantalla y escucharemos el sonido por los audífonos del dispositivo móvil.

Fig. 5.18 Submenú Reproducir Audio de la Gioconda [diseño propio]

5.9.9 Prueba de Video

Si seleccionamos Reproducir Video, nos mostrará el vídeo y escucharemos el sonido por los audífonos del dispositivo móvil.

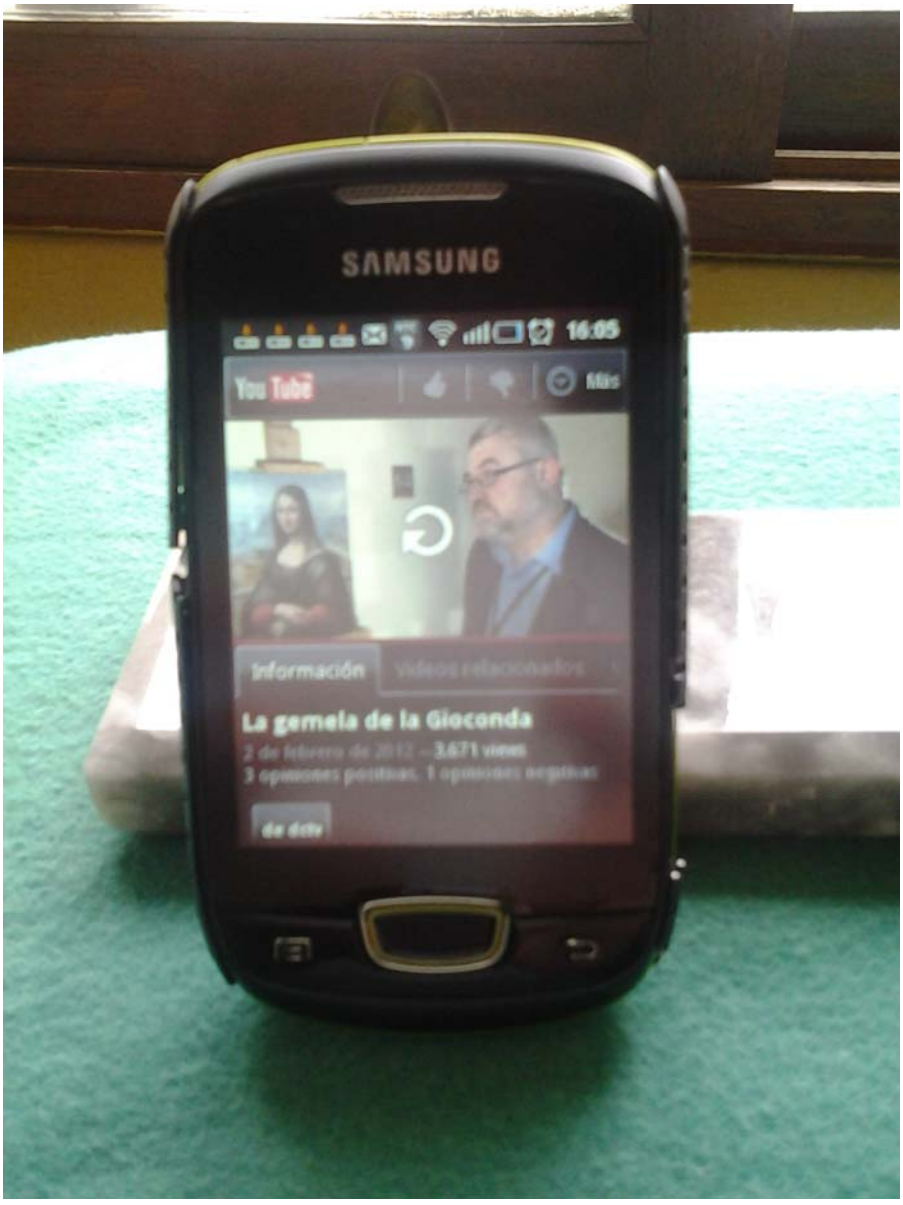


Fig. 5.19 Submenú Reproducir Video de la Gioconda [diseño propio]

5.10 Análisis de costos del proyecto para el museo

Instalación y configuración de servidores: Incluye la instalación y configuración del servidor de base de datos MySQL 5.0, el servidor web con Apache 2.0.

Diseño y programación del proyecto piloto: Tiene un costo de S/ 30,000 (TREINTA MIL NUEVOS SOLES).

En cuanto al presupuesto necesario para la implementación de este proyecto, el monto total es de **S/. 239,434.74 (DOSCIENTOS TREINTA Y NUEVE MIL CUTROCIENTOS TREINTA Y CUATRO Y 74/100 NUEVOS SOLES**). Incluye el costo del diseño y programación del aplicativo utilizado por los dispositivos móviles.

Tabla 5.1 Inversión para la implementación del proyecto del museo [diseño propio]

COSTOS DETALLADOS					
N°	Descripción	Producto	Cantidad	Precio Unitario	Precio Total
1	Expediente Técnico	ESTUDIO	1	S/. 10,500.00	S/. 10,500.00
2	Supervisión	GLOBAL	1	S/. 10,500.00	S/. 10,500.00
3	Dispositivos móviles para préstamo (Inc. audífonos)	EQUIPO	100	S/. 300.00	S/. 30,000.00
4	Diseño, programación e implementación del programa en Android	GLOBAL	1	S/. 30, 000.00	S/. 30,000.00
5	Servidor WEB	EQUIPO	1	S/. 15,000.00	S/. 15,000.00
6	Servidor de base de datos	EQUIPO	1	S/. 15,000.00	S/. 15,000.00
7	Controlador Central de Wireless A.P.	EQUIPO	1	S/. 21,313.80	S/. 21,313.80
8	Access Point	EQUIPO	30	S/. 700.00	S/. 21,000.50
9	Switch 8 puertos	EQUIPO	1	S/. 680.50	S/. 680.50
10	Switch PoE 24 puertos	EQUIPO	2	S/. 2,233.00	S/. 4,466.00
11	Antena Omni Indoor 9dBi	DISPOSITIVO	30	S/. 215.00	S/. 6,450.00
12	Rack de comunicaciones	DISPOSITIVO	1	S/. 500.00	S/. 500.00
13	Sistema de aire acondicionado	EQUIPO	1	S/. 6,000.00	S/. 6,000.00
14	Instalación de cableado de red UTP Cat. 6 x 30 puntos	GLOBAL	1	S/. 10,000.00	S/. 10,000.00
15	Sistema de pozo a tierra	DISPOSITIVO	1	S/. 2,000.00	S/. 2,000.00
16	Configuración de servidores	Licencias Software	1	S/. 6,000.00	S/. 6,000.00
17	Instalación y configuración de Aps	GLOBAL	1	S/. 13,500.00	S/. 13,500.00
			Sub total		S/. 202,910.80
			IGV		S/. 36,523.94
			Total		S/. 239,434.74
	SON DOSCIENTOS TREINTA Y NUEVE MIL CUATROCIENTOS TREINTA Y CUATRO Y 74/100 NUEVOS SOLES				

CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se presentan las conclusiones finales, en contraste con los objetivos determinados inicialmente y los posibles trabajos futuros.

Conclusiones finales

1. Se implementó el sistema inalámbrico inteligente de proximidad, lo que permite a los usuarios acceder a un sistema de información multimedia de un museo, mediante el uso de dispositivos móviles como teléfonos celulares inteligentes y tabletas.

2. Se diseñó la base de datos centralizada con MySQL, la misma que distribuye la información en diferentes formatos a través de la red inalámbrica WIFI.

3. Se instaló y configuró un servidor WEB Apache de libre distribución, el mismo que permite interactuar con el servidor de base de datos, y con el programa de aplicación instalado en el dispositivo móvil. Las pruebas de desempeño fueron exitosas.

4. Se implementó la interfaz de usuario para el acceso a la información, sobre dispositivos móviles con plataforma Android, en el cual, se utilizó un programa lector de códigos bidimensional QR, instalado en el móvil, que lee un código adosado a cada obra del museo, lo que facilita el acceso a información adicional en formato elegible como texto, imagen, audio y video. Las pruebas de desempeño dieron excelentes resultados.

5. Se diseñó la red de transporte de alta velocidad para conectar los puntos de acceso distribuidos en el museo, utilizando cableado estructurado de cobre. Se realizaron las pruebas de cobertura en cada piso del museo lo que permitió determinar el número óptimo de Access Points. Las pruebas de desempeño realizadas fueron exitosas.

6. Se diseñó la red de acceso inalámbrico, con tecnología IEEE 802.11n, por sus excelentes características de desempeño, alta velocidad de transmisión y elevado soporte de usuarios concurrentes. Las pruebas de cobertura de potencia de señal realizadas con el WIFI Analyzer instalados en los dispositivos móviles, fueron exitosas.

7. Finalmente, basado en las conclusiones anteriores, podemos afirmar que los objetivos de la tesis se han cumplido satisfactoriamente, el balance es muy positivo, y se espera que sirva de aliciente para los desarrolladores y programadores de dispositivos móviles actuales y futuros.

Trabajos futuros

1. Desarrollar aplicaciones de Realidad Aumentada, donde el mundo real se combina con el mundo virtual mediante el uso de cámaras y proyectores.

2. Desarrollar aplicaciones para móviles sobre plataforma Android, que incluyan sensores inerciales como magnetómetros, acelerómetros, geoposicionamiento con GPS, localización de personas, alertas tempranas de sismos y el reporte inmediato de incidentes que ponen en riesgo la salud y la vida del ser humano.

3. Desarrollar aplicaciones sobre redes WIFI, independiente de la conexión a Internet, que sería de gran gran utilidad en museos y parques temáticos de zonas rurales del Perú, donde no se cuenta con servicio de Internet. Este aspecto es importante para la inclusión socio económica y para el desarrollo cultural de dichas zonas del país.

GLOSARIO

Add-on: ver plug-in.

AIDL: siglas de Android Interface Definition Language, en español Lenguaje para la Definición de Interfaces en Android. Constituye un lenguaje de sintaxis muy básica que permite describir interfaces que pueden ser utilizadas de forma remota.

API: siglas de Application Programming Interface, en español Interfaz de Programación de Aplicaciones. Consiste en un conjunto de llamadas que ofrecen acceso a funciones y procedimientos, representando una capa de abstracción para el desarrollador.

Background: representa un proceso que se ejecuta con pocos recursos, que no requiere interacción directa con el usuario y que existe sin el conocimiento de este.

Biblioteca: agrupación de código y datos que proporcionan servicios a programas independientes, pasando a formar parte de éstos. Permiten la distribución de funcionalidades y la construcción modular. También conocido como librería, por su similitud con el inglés library.

Bluetooth: protocolo que permite la transmisión de datos entre dispositivos, más o menos próximos y alineados, mediante un enlace de radiofrecuencia. Está especialmente diseñado para dispositivos de bajo consumo y costo.

Bytecode: código intermedio, más abstracto que el código máquina, y que necesita de un mediador o máquina virtual para poder ser transformado y ejecutado en un hardware local.

Callback: se denomina así a la relación que existe entre dos procesos cuando el origen de la comunicación es a su vez llamado o invocado por el proceso destino.

Checkbox: elemento de interfaz de usuario que permite hacer una selección múltiple en un conjunto de opciones.

Códec: abreviatura de Compresor-Decompresor, en español Compresor-Descompresor. Representa un algoritmo software o componente hardware que permite transformar un flujo de datos en una señal interpretable.

Controlador: programa que permite al sistema operativo interactuar con un periféricos, abstrayéndolo y propocionando una interfaz para usarlo. También conocido como driver.

CORBA: siglas de Common Object Request Broker Architecture, en español Arquitectura Común de Intermediarios en Peticiones a Objetos. Define un estándar mediante el cual elementos distribuidos programados en distintos lenguajes pueden intercambiar datos y objetos.

CPU: siglas de Central Processing Unit, en español Unidad Central de Procesamiento. Elemento de hardware que controla el funcionamiento de un computador y lleva a cabo sus funciones de procesamiento de instrucciones y datos.

CVM: siglas de Compact Virtual Machine. Representa, junto a KVM, una de las máquinas virtuales de Java disponibles en Java ME.

Dalvik: nombre de la máquina virtual utilizada por el sistema operativo Android. Dalvik esta específicamente adaptada a las características de rendimiento de un dispositivo móvil y trabaja con ficheros de extensión ".dex", obtenidos desde el bytecode de Java.

Dispositivo móvil: aparato electrónico que es de reducido tamaño, cuenta con cierta capacidad tanto para la computación como para el almacenamiento de datos y cuenta con elementos de E/S básicos, por ejemplo pantalla y/o teclado.

Driver: ver controlador.

DTD: siglas de Document Type Definition, en español Definición de Tipo de Documento. Permite definir el formato que ha de tener un determinado documento XML.

Ebook: dispositivo electrónico que permite la visualización de documentos, como por ejemplo libros, en formato digital a través de una pantalla.

EDGE: siglas en inglés de Enhanced Data rates for GSM of Evolution, en español Tasas de Datos Mejoradas para la evolución de GSM. Es una tecnología para teléfonía móvil que representa un puente entre la segunda y tercera generación de estos dispositivos.

Fragmentación: fenómeno que describe una situación en la que una misma tecnología ha evolucionado de forma que se hace incompatible entre sí.

Framework: término con el que se define un amplio conjunto de elementos que permite desarrollar y organizar software utilizando un determinado lenguaje, sistema o tecnología. Habitualmente incluye bibliotecas, programas de desarrollo o manuales.

Googol (Google): término con el que se designa al número 10 elevado a 100.

GPS: siglas de Global Positioning System, en español Sistema de Posicionamiento Global. Es un sistema global de navegación que, mediante satélites, permite ubicar un objeto en la superficie terrestre con una precisión que va desde varios metros a centímetros.

GSM: siglas de Groupe Spécial Mobile, más conocido como Sistema Global para las Comunicaciones Móviles, es el estándar más extendido para las comunicaciones con telefonía móvil. Permite llamadas, navegación por Internet o envío de SMS.

GPRS: siglas de General Packet Radio Service, en español Servicio General de Paquetes vía Radio. es una extensión del estándar GSM que permite mejorar sus prestaciones originales, como el envío de datos, uso de correo electrónico, navegación web o el aumento de las tasas de transferencia de datos.

GUI: siglas de Graphical User Interface, en español Interfaz Gráfica de Usuario. Representa la parte del software que, mediante un contexto o lenguaje principalmente visual y simbólico, permite al usuario utilizar una aplicación.

Hilo: en sistemas operativos, un hilo constituye cada uno de los flujos de ejecución en el que puede ser divido un proceso. Todos los hilos de un proceso comparten espacio en memoria, archivos abiertos, variables globales, semáforos, etc. Permiten la ejecución concurrente de varias tareas. También llamado thread.

HTTP: siglas de HyperText Transfer Protocol, en español Protocolo de Transferencia de Hipertexto. Constituye el protocolo utilizado para la transmisión de documentos a través de la Web entre un cliente y servidor.

Interfaz: en computación, una interfaz se refiere a una abstracción que una determinada elemento ofrece de sí mismo al exterior, facilitando de esta forma su acceso y uso por otros elementos de hardware o software.

JAD: siglas de Java Aplication Descriptor, en español Descriptor de Aplicación Java. Archivo que acompaña a una aplicación Java ME y que ofrece información general y de despliegue sobra la misma.

JAR: acrónimo de Java ARchive, en español Archivo Java. Representa una agrupación de varios ficheros Java y se usa generalmente para la distribución conjunta de clases y metadatos.

Java ME: Java Micro Edition, edición de Java especialmente dirigida a los dispositivos móviles.

Java Micro Edition: ver Java ME.

JSR: siglas de Java Specification Request, en español Solicitud de Especificación Java. Representa un documento formal que describe las especificaciones y tecnologías propuestas para ser añadidas oficialmente a la plataforma Java.

JVM: siglas de Java Virtual Machine, en español Máquina Virtual de Java. Constituye un elemento software de la tecnología Java, encargado de transformar el código intermedio universal o bytecode en código máquina específico del hardware donde está instalado.

Kernel: parte fundamental de un sistema operativo, responsable de facilitar acceso seguro al hardware, gestionar recursos y hacer llamadas al sistema. También conocido como núcleo.

KVM: siglas de Kilobyte Virtual Machine. Representa, junto a CVM, una de las máquinas virtuales de Java disponibles en Java ME.

Listener: objeto que está a la espera de determinado evento.

Máquina virtual: representa un software que emula el comportamiento de una determinada arquitectura o que permite adaptar un código fuente a las características de la máquina nativa.

Microkernel: tipo de kernel de un sistema operativo que provee un conjunto de primitivas o llamadas al sistema mínimas. También llamado micronúcleo.

Middleware: capa de abstracción software que posibilita el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas.

MMS: siglas de Multimedia Messaging System o en español Sistema de Mensajería Multimedia. Es un estándar de mensajería que le permite a los teléfonos móviles enviar y recibir contenidos multimedia, incorporando sonido, video, fotos o cualquier otro contenido disponible en el futuro.

MOAP: siglas de Mobile Oriented Applications Platform, una plataforma software basada en Symbian para los teléfonos del fabricante japonés FOMA.

MP3: formato de compresión de audio digital cuyo nombre completo es MPEG-1 Audio Layer 3. También utilizado, por extensión, para nombrar al dispositivo móvil reproductor de audio digital en el que se almacena, organiza o reproduce archivos de audio digital

OHA: siglas de Open Handset Alliance, un conglomerado de empresas de sectores tecnológicos lideradas por Google que promueve la innovación y desarrollo de dispositivos móviles y sus aplicaciones. Su primera contribución es el sistema operativo Android.

PDA: siglas de Personal Digital Assistant, en español Asistente Digital Personal. Dispositivo móvil utilizado como organizador personal, que cuenta generalmente con pantalla táctil, agenda, calendario, conectividad Wi-Fi, y aplicaciones ofimáticas, entre otros.

Plug-in: componente de software que se relaciona y ejecuta con otro para aportarle una función nueva y generalmente muy especifica.

Proceso: un proceso es un programa en ejecución, y representa la unidad de procesamiento básica gestionada por el sistema operativo.

Radiobutton: elemento de interfaz de usuario que permite seleccionar un único elemento dentro un conjunto definido de valores.

RAM: siglas de Random Access Memory, o en español Memoria de Acceso Aleatorio. Componente de memoria volátil, con palabras de acceso individual mediante una dirección de memoria específica y con rápida ejecución de operaciones de lectura y escritura.

RMI: siglas de Remote Method Invocation, en español Invocación de Métodos Remotos, es una tecnología de Java que permite comunicar objetos distribuidos escritos en este lenguaje.

RMS: siglas de Record Management System, en español Sistema de Gestión de Registros. Representa un mecanismo de almacenamiento permanente en dispositivos con Java ME.

SAX: siglas de Simple API for XML, en español API Simple para XML, representa una conocida API para Java que facilita el procesamiento de documentos XML.

SDK: siglas de Software Development Kit, en español Kit de Desarrollo de Software. Constituye un conjunto de herramientas que permiten a un desarrollador crear aplicaciones para una determinada plataforma o lenguaje.

Servlet: elemento de la tecnología Java, que extiende la funcionalidad de un servidor Web, aceptando y procesando peticiones.

Sistema operativo: programa cuya finalidad principal es simplificar el manejo y explotación de un elemento con capacidad computacional, gestionando sus recursos, ofreciendo servicios a las demás aplicaciones y ejecutando mandatos del usuario.

Smartphone: dispositivo móvil que representa una evolución de los teléfonos móviles, con la inclusión de pantalla táctil, teclado, conexión Wi-Fi, aplicaciones de usuario como navegador web o cliente de correo, entre otros.

SMS: siglas de Short Message Service, en español Servicio de Mensajes Cortos, es un estándar de la telefonía móvil que permite enviar mensaje de texto con un número de caracteres limitado.

Socket: abstracción software, identificada por una dirección IP, un protocolo y un puerto, que permite la comunicación de dos programas, generalmente situados en computadores distintos**.**

Teclado QWERTY: teclado cuya distribución de letras es la más común hoy día en computadores y otros elementos de computación. Toma su nombre de sus 5 primeras letras alfabéticas: Q, W, E, R, T, e Y.

Thread: ver hilo.

UIQ: siglas de User Interface Quartz, una plataforma software basada en Symbian usada en algunos teléfonos de los fabricantes Sony Ericcson y Motorola.

UMTS: siglas de Universal Mobile Telecommunications System, en español Sistema Universal de Telecomunicaciones Móviles. Constituye en estándar de comunicación para dispositivos de tercera generación o 3G, que ofrece capacidades multimedia y conexiones de alta velocidad en Internet.

WAP: siglas de Wireless Application Protocol, en español o Protocolo de Aplicaciones Inalámbricas. Es un estándar para aplicaciones que utilizan las comunicaciones inalámbricas, como el acceso a servicios de Internet desde un teléfono móvil.

Widget: componente gráfico utilizado en interfaces de usuario, con el cual el usuario puede interactuar, como por ejemplo cajas de texto, botones, ventanas, etc.

WIFI: acrónimo de Wíreless Fidelity, estándar de envío de datos que utiliza ondas de radio en lugar de cables.

XML: siglas de Extensible Markup Language, en español Lenguaje de Marcado Extensible. Representa un lenguaje estándar que, mediante el uso de etiquetas y atributos, permite expresar e intercambiar fácilmente estructuras de datos.

XMPP: siglas de Extensible Messaging and Presence Protocol, en español Protocolo Extensible de Mensajería y Presencia. Es un protocolo basado en XML que se utiliza en servicios de mensajería instantánea.

BIBLIOGRAFÍA

[1] Android, el sistema operativo de Google. Último acceso, enero de 2012. http://code.google.com/intl/es-ES/android/index.html

[2] Bluetooth. Ultimo acceso , enero 2012 http://es.wikipedia.org/wiki/Bluetooth

[3] Tesis: Software de control remoto para aplicaciones del escritorio GNOME: Último acceso, enero de 2012 http://gbtcr.chileforge.cl/info_web/node77.html

[4] Tesis: Diseño de la interfaz de usuario para la PDA y programa de adquisición de datos: Ultimo acceso Enero 2012 http://dspace.epn.edu.ec/bitstream/15000/8553/13/T10508CAP1.pdf

[5] Wi-Fi Handbook, Building 802.11b Wireless Networks" Orthman, Roeder

[6] http://standards.ieee.org/about/get/802/802.11.html.Ultima consulta Enero 2012

[7] INTERNATIONAL STANDARD ISO/IEC 18004 First edition 2000-06-15 Information technology — Automatic identification and data capture techniques — Bar code symbology — QR Code

[8] "Programming for the Series 60 platform and Symbian", Digia Inc. Editorial Wiley, 2003.

[9] "Symbian OS Communications Programming", Michael J. Jipping. Editorial Wiley, 2002.

[10] "Diseño y desarrollo de una herramienta de control y configuración de opciones de seguridad en Symbian", Daniel Marcos Martín. Proyecto Fin de Carrera, 2007.

[11] "How to Do Everything with Windows Mobile", Frank McPherson. Editorial McGraw-Hill, 2006.

[12] http://es.wikipedia.org/wiki/Android#cite_note-32

[13] Java a Tope", Sergio Gálvez Rojas y Lucas Ortega Días. Editado por la Universidad de Málaga, 2003

[14] http://es.wikipedia.org/wiki/Android

[15] Dalvik VM Internals, especificaciones de la máquina virtual. Último acceso en Enero 2012 http://sites.google.com/site/io/dalvik-vm-internals

[16] Barrapunto, artículo publicado el 5 de diciembre de 2007: "¿Por qué Google desarrolla Dalvik en vez de usar Java Micro Edition?". Último acceso en Enero 2012 http://softlibre.barrapunto.com/article.pl?sid=07/12/05/1357216

[17] Android Comunity, entrevista a Jason Chen, desarrollador de Android. Último acceso http://androidcommunity.com/jason-chen-answers-questions-about-android- 20080603/

[18] Descripción del ciclo de vida de una aplicación Android, por Google. Último acceso en Enero 2012 http://es.youtube.com/watch?v=fL6gSd4ugSI

[19] Proyecto Zxing para integrar QR Code scanner a nuestra aplicación MuseoFiee http://code.google.com/p/zxing/source/browse/trunk/android-integration/src/com/google/zxing/integration/android/IntentIntegrator.java

[20] Generador de QR code. http://qrcode.kaywa.com/

ANEXOS

ANEXO A

IDE Eclipse

A.1 Entorno de desarrollo integrado: Eclipse

La descarga de Eclipse lo realizamos de la web de eclipse:

http://www.eclipse.org/downloads/

Elegir la versión Eclipse Indigo. No se realiza ningún proceso de instalación; simplemente se debe descomprimir los archivos en una carpeta y pulsar el ejecutable. La primera vez que se inicie Eclipse, pide al usuario una localización para el *workspace*, donde se ubicarán por defecto todos los proyectos desarrollados.

A.1.1 Descargar el SDK de Android

Android es una plataforma de software libre. Cuenta con un SDK que incluye todas las APIs. Para descargarlo, basta con visitar la web de Android.

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

A.1.2 Instalar el plug-in de Android

El siguiente paso consisten en instalar un *plug-in* específico de Android para la plataforma Eclipse. Esta herramienta, llamada ADT (Android Development Tools), facilita enormemente la creación de proyectos, su implementación, depuración y ejecución, por lo que es altamente recomendable si se quiere trabajar con Android.
Para instalar el *plug-in* ADT en Eclipse Indigo, es necesario seguir las siguientes indicaciones:
1. Iniciar Eclipse
2. Seleccionar la pestaña *Help* > *Software Updates*. Esta acción abrirá una nueva ventana llamada *Software Updates and Add-ons*.

3. Elegir la pestaña *Available Software* y pulsar el botón *Add Site*.
4. Introducir la siguiente URL y pulsar *OK*:

http://dl-ssl.google.com/android/eclipse/

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

A.1.3 Referenciar el SDK de Android

Tras abrir de nuevo Eclipse, debe indicarse en las preferencias de Eclipse la localización del SDK a utilizar para los proyectos de Android:
1. Seleccionar la pestaña *Window* > *Preferences*, lo que abrirá una nueva ventana.
2. Elegir *Android* en el panel izquierdo.
3. Pulsar el botón *Browse* e indicar la ruta del SDK de Android.
4. Pulsar el botón *Apply* y después *OK*.

A.1.4 Actualizaciones del plug-in

Para comprobar las posibles actualizaciones a través de Eclipse, se debe realizar lo siguiente:
1. Iniciar Eclipse.
2. Seleccionar la pestaña *Help* > *Software Updates*. Esta acción abrirá una nuevaventana llamada *Software Updates and Add-ons*.
3. Elegir la pestaña *Installed Software*.
4. Pulsar el botón *Update*.
5. Si existe alguna actualización para el ADT, seleccionarla y pulsar el botón *Finish*.

A.1.5 Crear un nuevo proyecto

1. Crear el proyecto

Seleccionar la pestaña *File* > *New* > *Project*, y en el menú resultante desplegar la opción *Android* y elegir *Android Project*. Pulsar el botón *Next*.

2. Especificar las propiedades

A continuación se muestra una ventana donde es necesario detallar algunas propiedades del proyecto. En concreto, se precisa asignar un valor para:

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

Package name: el nombre del paquete bajo el cual será desarrollado todo el código. Por ejemplo, se le puede asignar el valor "edu.gbartra.museo.fiee".

Activity name: es el nombre de la clase Activity que será creada de forma automática por el *plug-in*. Esta clase Activity simplemente es una clase ejecutable, capaz de realizar alguna tarea, y es imprescindible en la mayoría de las aplicaciones para Android.

Application name: el nombre de la aplicación que se va a desarrollar.

Tras pulsar el botón *Finish*, Eclipse mostrará en el explorador de paquetes los elementos que conforman el actual proyecto. Además, si se visita el *workspace* asignado para los proyectos de Eclipse, podrá observarse que existe una nueva carpeta con el nombre del proyecto.

A.2 Emulador Android

En Android, las interfaces de usuario están formadas por una jerarquía de clases llamadas Views (vistas). Una vista o View representa cualquier elemento que se pueda dibujar en pantalla, como un botón, una animación, una etiqueta de texto, etc.

A.2.1 Ejecutar la aplicación

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

1. Seleccione la pestaña *Run* > *Run Configurations*. Se abrirá una nueva ventana.

2. En el panel de la izquierda, hay que localizar *Android Application* y pulsar el icono con la leyenda *New launch configuration*.

3. Asignar un nombre para esta configuración.

4. En la etiqueta *Android*, pulsar el botón *Browse* para localizar el proyecto seleccionar la opción *Launch* para localizar la actividad

5. Pulsar el botón *Run*.

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

A.2.2 Contenido de un proyecto Android

Al crear un proyecto Android, en el *workspace* de Eclipse se genera una nueva carpeta con el nombre de dicho proyecto. Esta carpeta contiene una serie de subcarpetas y de ficheros que constituyen la anatomía completa de un proyecto Android.
Las carpetas y archivos han sido creadas por el *plug-in* ADT, y sus características dependerán siempre del tipo de proyecto que se esté construyendo. Los principales elementos que podemos encontrar se describen a continuación.

A.2.3 Carpeta \src

Carpeta que contiene los archivos fuente .java del proyecto. Utiliza la misma jerarquía de carpetas que la indicada por el nombre del paquete que se haya asignado.
Archivo, “R.java”, es un archivo que siempre se adjunta por defecto a cualquier proyecto Android y que declara una serie de índices o referencias a los recursos externos que se utilizan en el proyecto actual.

A.2.4 Carpeta \res

La carpeta “\res” alberga los recursos utilizados en el proyecto. Por recurso se entiende cualquier archivo externo que contenga datos o descripciones referentes a la aplicación y que debe ser compilado junto a los ficheros fuente. Esta compilación permite al recurso ser accedido de forma más rápida y eficiente.
Android contempla muchos tipos de recursos, como XML, JPEG o PNG, entre otros.

A.2.5 Carpeta \bin

Esta carpeta contiene los archivos binarios del proyecto generados a partir de los archivos fuente. Al igual que la carpeta “\src", se mantiene la misma jerarquía de subcarpetas que la representada en el nombre del paquete.

A.3 Ubicación de los puntos de acceso de la red inalámbrica del museo

ANEXO B

Listado de programas

B.1 Principal.java

```
package com.example.qrinfo;

import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class Principal extends Activity {

        private static final int ACTIVITY_RESULT_QR_DRDROID = 0;
        public static final String SCAN = "la.droid.qr.scan";
         Button button ;
         Button btnSalir ;
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
     /*
      Intent i=new Intent(Principal.this, MostrarInfo.class);
      i.putExtra("cadena","gioconda");
      startActivity(i);

      */


      setContentView(R.layout.activity_principal);
         //final Spinner spinner = (Spinner) findViewById(R.id.spin_complete);
         //"Scan" button
                  Button button = (Button) findViewById(R.id.button1);
                 btnSalir = (Button) findViewById(R.id.btnSalir);

                 btnSalir.setOnClickListener(new OnClickListener() {
```

```
                @Override
                public void onClick(View v) {
                        // TODO Auto-generated method stub
                        Principal.this.finish();
                }
        });

    //Set action to button
    button.setOnClickListener( new OnClickListener() {
        public void onClick(View v) {
        //Create a new Intent to send to QR Droid
                Intent qrDroid = new Intent( "la.droid.qr.scan" ); //Set action
"la.droid.qr.scan"
                qrDroid.putExtra( "la.droid.qr.complete" , true);
            startActivityForResult(qrDroid, 0);
                //Check whether a complete or displayable result is needed
                //if( spinner.getSelectedItemId()==0 ) { //First item selected
("Complete content")
                        //Notify we want complete results (default is FALSE)
                //      qrDroid.putExtra( "la.droid.qr.complete" , true);
                //}

                //Send intent and wait result
                //try {
                //      startActivityForResult(qrDroid,
ACTIVITY_RESULT_QR_DRDROID);
                //} catch (ActivityNotFoundException activity) {
                //      Services.qrDroidRequired(Scan.this);
                //}
        }
    });
  //Intent qrDroid = new Intent( "la.droid.qr.scan" );
  //qrDroid.putExtra( "la.droid.qr.complete" , true);
  //startActivityForResult(qrDroid, 0);
 }
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

if(      ACTIVITY_RESULT_QR_DRDROID==requestCode      &&      null!=data      &&
data.getExtras()!=null ) {
//Read result from QR Droid (it's stored in la.droid.qr.result)
String result = data.getExtras().getString("la.droid.qr.result");
//Just set result to EditText to be able to view it
//TextView resultTxt = ( TextView ) findViewById(R.id.textView1);
//resultTxt.setText( result );
//resultTxt.setVisibility(View.VISIBLE);


Intent i=new Intent(Principal.this, MostrarInfo.class);
        i.putExtra("cadena",result);
```

```
                startActivity(i);
                  }
        }

}
```

B.2 MostrarInfo.java

```
package com.example.qrinfo;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;

import org.apache.http.NameValuePair;
import org.apache.http.message.BasicNameValuePair;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.net.Uri;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.ActivityNotFoundException;
import android.content.DialogInterface;
import android.content.Intent;

public class MostrarInfo extends Activity {

        Button btnVerTexto;
        Button btnVerImagen;
        Button btnRepAudio;
        Button btnRepVideo;
        Button btnDesTexto;
        Button btnDesImagen;
        Button btnDesAudio;
        Button btnDesVideo;
        TextView txtInfoTitulo;
```

```
        Button btnSalir;
        Button btnPrincipal;
TextView mProgressText;
        private ProgressDialog pDialog;
        String cadena;
        String urls[] = null;
        Httppostaux post;
// String URL_connect="http://www.infocodqu.vacau.com/droidlogin/acces.php";
String IP_Server = "http://172.17.9.230:8080/";//IP del Server
        String URL_connect =  IP_Server + "acces.php";

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                urls = new String[5];
                setContentView(R.layout.activity_mostrar_info);
                post = new Httppostaux();


                Bundle extras = getIntent().getExtras();
                cadena = extras.getString("cadena");


                new asynclogin().execute();

                mProgressText=(TextView)findViewById(R.id.mProgressText);

                btnPrincipal=(Button)findViewById(R.id.btnInfoPrincipal);

btnPrincipal.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                MostrarInfo.this.finish();

                        }
                });

                txtInfoTitulo=(TextView)findViewById(R.id.txtInfoTitulo);


                btnVerTexto = (Button) findViewById(R.id.btnVerTexto);
                btnVerImagen = (Button) findViewById(R.id.btnVerImagen);
                btnRepAudio = (Button) findViewById(R.id.btnRepAudio);
                btnRepVideo = (Button) findViewById(R.id.btnRepVideo);
                btnDesTexto = (Button) findViewById(R.id.btnDesTexto);
                btnDesImagen = (Button) findViewById(R.id.btnDesImagen);
                btnDesAudio = (Button) findViewById(R.id.btnDesAudio);
                btnDesVideo = (Button) findViewById(R.id.btnDesVideo);
```

```
		btnVerTexto.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {
	Intent i = new Intent(MostrarInfo.this, MostrarTexto.class);
				i.putExtra("url", urls[1]);
				i.putExtra("cadena",cadena);
				startActivity(i);
			}
		});

		btnVerImagen.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {
				Intent i = new Intent(MostrarInfo.this, MostrarImagen.class);
				//i.putExtra("url",                    "http://jonsegador.com/wp-
content/apezz.png");
				i.putExtra("url", IP_Server + urls[2]);
				i.putExtra("cadena",cadena);
				startActivity(i);
			}
		});

		btnRepAudio.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {

				Intent i = new Intent(android.content.Intent.ACTION_VIEW);

	//i.setDataAndType(Uri.parse("http://www.infocodqu.vacau.com/Ma.mp3"),
"audio/*");
				i.setDataAndType(Uri.parse(IP_Server + urls[3]), "audio/*");
				MostrarInfo.this.startActivity(i);

			}
		});

		btnRepVideo.setOnClickListener(new OnClickListener() {

			@Override
			public void onClick(View v) {

				/*
				Intent videoClient = new Intent(Intent.ACTION_VIEW);
		 //videoClient.setData(Uri.parse("http://www.youtube.com/watch?v=LKL-
efbiIAM"));
		 videoClient.setData(Uri.parse(urls[4]));
		//                videoClient.setClassName("com.google.android.youtube",
"com.google.android.youtube.PlayerActivity");
```

```
                videoClient.setClassName("com.google.android.youtube",
"com.google.android.youtube.WatchActivity");
                 startActivity(videoClient);

                                                                    */
                        Intent i = new Intent(android.content.Intent.ACTION_VIEW);

        //i.setDataAndType(Uri.parse("http://www.infocodqu.vacau.com/Ma.mp3"),
"audio/*");
                        i.setDataAndType(Uri.parse(IP_Server + urls[4]), "video/*");
                        MostrarInfo.this.startActivity(i);

                    }
            });

            /*
            btnDesTexto.setOnClickListener(new OnClickListener() {

                @Override
                public void onClick(View v) {
                        try {
                //set the download URL, a url that points to a file on the internet
                //this is the file to be downloaded

                                mProgressText.setText("begining");
                URL url = new URL("http://jonsegador.com/wp-content/apezz.png");

                //create the new connection
                HttpURLConnection urlConnection = (HttpURLConnection)
url.openConnection();

                //set up some things on the connection
                urlConnection.setRequestMethod("GET");
                urlConnection.setDoOutput(true);

                //and connect!
                urlConnection.connect();

                //set the path where we want to save the file
        //in this case, going to save it on the root directory of the
                //sd card.
                File SDCardRoot = Environment.getExternalStorageDirectory();
        //create a new file, specifying the path, and the filename
                //which we want to save the file as.
                File file = new File(SDCardRoot,"somefile.png");

                //this will be used to write the downloaded data into the file we created
                FileOutputStream fileOutput = new FileOutputStream(file);
```

```
                        //this will be used in reading the data from the internet
                        InputStream inputStream = urlConnection.getInputStream();

                        //this is the total size of the file
                        int totalSize = urlConnection.getContentLength();
                        //variable to store total downloaded bytes
                        int downloadedSize = 0;

                        //create a buffer...
                        byte[] buffer = new byte[1024];
                        int bufferLength = 0; //used to store a temporary size of the buffer

                        //now, read through the input buffer and write the contents to the file
                        while ( (bufferLength = inputStream.read(buffer)) > 0 ) {
 //add the data in the buffer to the file in the file output stream (the file on the sd card
                                fileOutput.write(buffer, 0, bufferLength);
                                //add up the size so we know how much is downloaded
                                downloadedSize += bufferLength;
                                //this is where you would do something to report the prgress, like
this maybe
                                updateProgress(downloadedSize, totalSize);

                        }
                        //close the output stream when done
                        fileOutput.close();

                //catch some possible errors...
                } catch (MalformedURLException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }

                          }
                });
*/

        }
        public void updateProgress(int currentSize, int totalSize){
                mProgressText.setText(Long.toString((currentSize/totalSize)*100)+"%");
                }
        public String[] getUrls(String cadena) {

                String urlsLeido[] = new String[5];
                ArrayList<NameValuePair>          postparameters2send          =          new
ArrayList<NameValuePair>();

                postparameters2send.add(new BasicNameValuePair("cadena", cadena));

        JSONArray jdata = post.getserverdata(postparameters2send, URL_connect);
```

```
		if (jdata == null) {
			urlsLeido[0] = "error de jdata nulo";
			return urlsLeido;
		}
		// si lo que obtuvimos no es null
		if (jdata != null && jdata.length() > 0) {

			JSONObject json_data; // creamos un objeto JSON
			try {
			json_data = jdata.getJSONObject(0); // leemos el primer segmento

// en nuestro caso el unico

				urlsLeido[1] = json_data.getString("texto");
				urlsLeido[2] = json_data.getString("imagen");
				urlsLeido[3] = json_data.getString("audio");
				urlsLeido[4] = json_data.getString("video");

			} catch (JSONException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
				urlsLeido[0] = "error al leer el texto,imagen,..";
				return urlsLeido;
			}
			if (urlsLeido[1].compareTo("error")==0 ){
				urlsLeido[0] = "error de cadena invalida";
			}else{
				urlsLeido[0] = "";
			}


			return urlsLeido;

		} else { // json obtenido invalido verificar parte WEB.
			urlsLeido[0] = "error de jdata es vacio";
			return urlsLeido;
		}

}

class asynclogin extends AsyncTask<Void, Void, Boolean> {

		String user, pass;

		protected void onPreExecute() {
			// para el progress dialog
			pDialog = new ProgressDialog(MostrarInfo.this);
			pDialog.setMessage("Autenticando....");
			pDialog.setIndeterminate(false);
			pDialog.setCancelable(false);
			pDialog.show();
```

```
		}

		protected Boolean doInBackground(Void... params) {

			urls = getUrls(cadena);
			if (urls[0].compareTo("") == 0)
				return true;
			return false;
		}

		protected void onPostExecute(Boolean result) {
			pDialog.dismiss();
			 if (result) {
				 txtInfoTitulo.setText(cadena);
				//pDialog.dismiss();// ocultamos progess dialog.

//		Intent i = new Intent(MostrarInfo.this, MostrarTexto.class);
				//i.putExtra("user", urls[0] + "\n" + urls[1] + "\n" + urls[2]
					//		+ "\n" + urls[3] + "\n" + urls[4] + "\n");
				//startActivity(i);
			} else {
				//pDialog.dismiss();
			//		txtInfoTitulo.setText("Error:" + cadena);
				 AlertDialog alerta= new AlertDialog.Builder(MostrarInfo.this).create(); //Aqui me marca el siguiente error The constructor AlertDialog.Builder(new View.OnClickListener(){}) is undefined

				alerta.setTitle("QR leido invalido");

				alerta.setMessage("Se leyo:\n" + cadena);

				alerta.setButton("OK", new DialogInterface.OnClickListener() {

				  public void onClick(DialogInterface dialog, int which) {

				    //return;
				     MostrarInfo.this.finish();
				} });

				alerta.show();

			}

			// }

		}

	}

}
```

CLASE AUXILIAR PARA EL ENVÍO DE PETICIONES A NUESTRO SISTEMA Y MANEJO DE RESPUESTA

B.3 Httppostaux.java

```
package com.example.qrinfo;

import java.io.BufferedReader;

import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.json.JSONArray;
import org.json.JSONException;
import android.util.Log;
/*CLASE AUXILIAR PARA EL ENVIO DE PETICIONES A NUESTRO SISTEMA
 * Y MANEJO DE RESPUESTA.*/
public class Httppostaux{

	InputStream is = null;
	String result = "";

	public JSONArray getserverdata(ArrayList<NameValuePair> parameters, String urlwebserver ){

		//conecta via http y envia un post.
	httppostconnect(parameters,urlwebserver);

	if (is!=null){//si obtuvo una respuesta

		 getpostresponse();

		return getjsonarray();

	}else{

	    return null;

	}

	}
```

```
          //peticion HTTP
   private void httppostconnect(ArrayList<NameValuePair> parametros, String
urlwebserver){

        //
        try{
              HttpClient httpclient = new DefaultHttpClient();
              HttpPost httppost = new HttpPost(urlwebserver);
              httppost.setEntity(new UrlEncodedFormEntity(parametros));
              //ejecuto peticion enviando datos por POST
              HttpResponse response = httpclient.execute(httppost);
              HttpEntity entity = response.getEntity();
               is = entity.getContent();

        }catch(Exception e){
              Log.e("log_tag", "Error in http connection "+e.toString());
        }

   }

  public void getpostresponse(){

        //Convierte respuesta a String
        try{
              BufferedReader reader = new BufferedReader(new
InputStreamReader(is,"iso-8859-1"),8);
              StringBuilder sb = new StringBuilder();
              String line = null;
              while ((line = reader.readLine()) != null) {
                    sb.append(line + "\n");
              }
              is.close();

              result=sb.toString();
              Log.e("getpostresponse"," result= "+sb.toString());
        }catch(Exception e){
              Log.e("log_tag", "Error converting result "+e.toString());
        }
 }

  public JSONArray getjsonarray(){
        //parse json data
        try{
       JSONArray jArray = new JSONArray(result);
       return jArray;
        }
        catch(JSONException e){
              Log.e("log_tag", "Error parsing data "+e.toString());
              return null;
        }
```

```
 }


 }
```

B.4 MostrarTexto.java

```
package com.example.qrinfo;

import android.os.Bundle;
import android.preference.EditTextPreference;
import android.app.Activity;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.support.v4.app.NavUtils;

public class MostrarTexto extends Activity {
EditText txt;
TextView txtTexto;
Button btnRegresar;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mostrar_texto);
        Bundle extras = getIntent().getExtras();
        String user  = extras.getString("url");//usuario
        String cadena  = extras.getString("cadena");//usuario
       txt=(EditText)findViewById(R.id.txtTextoInput);
       txt.setText(user);


                btnRegresar=(Button)findViewById(R.id.btnTextoRegresar);

btnRegresar.setOnClickListener(new OnClickListener() {

                        @Override
                        public void onClick(View v) {
                                MostrarTexto.this.finish();

                        }
                });

txtTexto=(TextView)findViewById(R.id.txtTexto);
txtTexto.setText(cadena);
```

```
    }


}
```

B.5 MostrarImagen.java

```
package com.example.qrinfo;

import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import org.w3c.dom.Text;

import android.os.Bundle;
import android.app.Activity;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;

public class MostrarImagen extends Activity {
ImageView im;

Button btnRegresar;
TextView txtImagen;
private Bitmap loadedImage;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mostrar_imagen);

        Bundle extras = getIntent().getExtras();
        String url  = extras.getString("url");//usuario
        String cadena  = extras.getString("cadena");//usuario
       im=(ImageView)findViewById(R.id.viewImagen);

       URL imageUrl;
           try {
                    imageUrl = new URL(url);

       HttpURLConnection conn = (HttpURLConnection) imageUrl.openConnection();
       conn.connect();
       loadedImage = BitmapFactory.decodeStream(conn.getInputStream());
```

```
        im.setImageBitmap(loadedImage);
            } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }


            btnRegresar=(Button)findViewById(R.id.btnImagenRegresar);

            btnRegresar.setOnClickListener(new OnClickListener() {

                                    @Override
                                    public void onClick(View v) {
                                            MostrarImagen.this.finish();

                                    }
                            });

            txtImagen=(TextView)findViewById(R.id.txtImagen);
            txtImagen.setText(cadena);



    }


}
```

ARCHIVOS PARA EL DISEÑO DE LA INTERFAZ DE USUARIO

Carpeta QRInfo/res/layout

B.6 activity_principal.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:layout_marginTop="92dp"
        android:text="            QR INFO"
```

```
        android:textSize="30dp"
        tools:context=".Principal" />

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignRight="@+id/textView1"
        android:layout_below="@+id/textView1"
        android:layout_marginTop="54dp"
        android:text="QR Scanner" />

    <Button
        android:id="@+id/btnSalir"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:text="Salir" />

</RelativeLayout>
```

B.7 activity_mostrar_info.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/txtInfoTitulo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginTop="28dp"
            android:layout_weight="1"
            android:textSize="25dp"
            tools:context=".MostrarInfo" />
    </LinearLayout>

    <LinearLayout
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:layout_alignParentRight="true"
    android:orientation="vertical" >
</LinearLayout>

<Button
    android:id="@+id/btnDesTexto"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_below="@+id/linearLayout1"
    android:layout_marginRight="50dp"
    android:layout_marginTop="34dp"
    android:text="Descargar"
    android:visibility="invisible" />

<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/btnDesTexto"
    android:layout_alignBottom="@+id/btnDesTexto"
    android:layout_alignParentLeft="true"
    android:layout_marginLeft="55dp"
    android:text="Texto:"
    android:textAppearance="?android:attr/textAppearanceSmall" />

<Button
    android:id="@+id/btnDesImagen"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btnDesTexto"
    android:layout_below="@+id/btnDesTexto"
    android:layout_marginTop="20dp"
    android:text="Descargar"
    android:visibility="invisible" />

<Button
    android:id="@+id/btnDesAudio"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btnDesImagen"
    android:layout_below="@+id/btnDesImagen"
    android:layout_marginTop="26dp"
    android:text="Descargar"
```

```
    android:visibility="invisible" />

<Button
    android:id="@+id/btnDesVideo"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/btnDesAudio"
    android:layout_below="@+id/btnDesAudio"
    android:layout_marginTop="28dp"
    android:text="Descargar"
    android:visibility="invisible" />

<Button
    android:id="@+id/btnInfoPrincipal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentLeft="true"
    android:text="Principal" />

<Button
    android:id="@+id/btnVerTexto"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/btnDesImagen"
    android:layout_alignLeft="@+id/btnVerImagen"
    android:layout_alignRight="@+id/btnVerImagen"
    android:text="Ver" />

<Button
    android:id="@+id/btnVerImagen"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/btnDesAudio"
    android:layout_alignLeft="@+id/btnRepAudio"
    android:layout_alignRight="@+id/btnRepAudio"
    android:text="Ver" />

<Button
    android:id="@+id/btnRepAudio"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBaseline="@+id/btnDesAudio"
    android:layout_alignBottom="@+id/btnDesAudio"
    android:layout_marginRight="16dp"
    android:layout_toLeftOf="@+id/btnDesAudio"
    android:text="Reproducir" />
```

```
    <Button
        android:id="@+id/btnRepVideo"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnDesVideo"
        android:layout_alignBottom="@+id/btnDesVideo"
        android:layout_alignLeft="@+id/btnRepAudio"
        android:text="Reproducir" />

    <TextView
        android:id="@+id/TextView03"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnRepVideo"
        android:layout_alignBottom="@+id/Button06"
        android:layout_alignLeft="@+id/textView1"
        android:text="Video:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/TextView02"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnRepAudio"
        android:layout_alignBottom="@+id/btnRepAudio"
        android:layout_alignLeft="@+id/TextView03"
        android:text="Audio:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/btnVerImagen"
        android:layout_alignBottom="@+id/btnVerImagen"
        android:layout_alignLeft="@+id/textView1"
        android:text="Imagen:"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <TextView
        android:id="@+id/mProgressText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnInfoPrincipal"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true" />

</RelativeLayout>
```

B.8 activity_mostrar_texto.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="top" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/txtTexto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:textSize="25dp"
            tools:context=".MostrarInfo" />

    </LinearLayout>

    <EditText
        android:id="@+id/txtTextoInput"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnTextoRegresar"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/linearLayout1"
        android:layout_marginTop="56dp"
        android:clickable="false"
        android:ems="10"
        android:inputType="textMultiLine"
        android:lines="30"
        android:maxLines="30" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/btnTextoRegresar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
```

```
        android:layout_alignParentLeft="true"
        android:text="Regresar" />

</RelativeLayout>
```

B.9 activity_mostrar_imagen.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="top" >

    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:orientation="vertical" >

        <TextView
            android:id="@+id/txtImagen"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_weight="1"
            android:textSize="25dp"
            tools:context=".MostrarInfo" />

    </LinearLayout>

    <Button
        android:id="@+id/btnImagenRegresar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:text="Regresar" />

    <ImageView
        android:id="@+id/viewImagen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btnImagenRegresar"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/linearLayout1"
```

```
        android:src="@drawable/ic_action_search" />

</RelativeLayout>
```

Carpeta QRInfo/res/menu

B.10 activity_principal.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_settings"
        android:title="@string/menu_settings"
        android:orderInCategory="100" />
</menu>
```

B.11 activity_qrscanner.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_settings"
        android:title="@string/menu_settings"
        android:orderInCategory="100" />
</menu>
```

B.12 activity_mostrar_info.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_settings"
        android:title="@string/menu_settings"
        android:orderInCategory="100" />
</menu>
```

B.13 activity_mostrar_texto.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/menu_settings"
        android:title="@string/menu_settings"
        android:orderInCategory="100" />
</menu>
```

B.14 activity_mostrar_imagen.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/menu_settings"
      android:title="@string/menu_settings"
      android:orderInCategory="100" />
</menu>
```

QRInfo/res/values

B.15 strings.xml

```
<resources>

   <string name="app_name">QRInfo</string>
   <string name="hello_world">Hello world!</string>
   <string name="menu_settings">Settings</string>
   <string name="title_activity_mostrar_info">MostrarInfo</string>
   <string name="title_activity_mostrar_texto">MostrarTexto</string>
   <string name="title_activity_mostrar_imagen">MostrarImagen</string>
   <string name="title_activity_qrscanner">QRScanner</string>
   <string name="title_activity_principal">Principal</string>

</resources>
```

/QRInfo/res/values

```
<resources>

   <style name="AppTheme" parent="android:Theme.Light" />

</resources>
```

QRInfo

B.16 AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.qrinfo"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
```

```
    android:targetSdkVersion="15" />

<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity
        android:name=".MostrarInfo"
        android:label="@string/title_activity_mostrar_info"
        android:screenOrientation="portrait"
        >
        <intent-filter>

        </intent-filter>
    </activity>
    <activity
        android:name=".MostrarTexto"
        android:label="@string/title_activity_mostrar_texto"
        android:screenOrientation="portrait">
        <meta-data
            android:name="android.support.PARENT_ACTIVITY"
            android:value="com.example.qrinfo.MostrarInfo" />
    </activity>
    <activity
        android:name=".MostrarImagen"
        android:label="@string/title_activity_mostrar_imagen"
        android:screenOrientation="portrait">
        <intent-filter>
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.qrinfo.MostrarInfo" />
        </intent-filter>
    </activity>
    <activity
        android:name=".QRScanner"
        android:label="@string/title_activity_qrscanner"
        android:screenOrientation="portrait">
        <intent-filter>

        </intent-filter>
    </activity>
    <activity
        android:name=".Principal"
        android:label="@string/title_activity_principal"
        android:screenOrientation="portrait">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
```

```
    </application>

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

</manifest>
```

B.17 R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.qrinfo;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int ic_action_search=0x7f020000;
        public static final int ic_launcher=0x7f020001;
    }
    public static final class id {
        public static final int Button06=0x7f070010;
        public static final int TextView01=0x7f070012;
        public static final int TextView02=0x7f070011;
        public static final int TextView03=0x7f07000f;
        public static final int btnDesAudio=0x7f070008;
        public static final int btnDesImagen=0x7f070007;
        public static final int btnDesTexto=0x7f070005;
        public static final int btnDesVideo=0x7f070009;
        public static final int btnImagenRegresar=0x7f070002;
        public static final int btnInfoPrincipal=0x7f07000a;
        public static final int btnRepAudio=0x7f07000d;
        public static final int btnRepVideo=0x7f07000e;
        public static final int btnSalir=0x7f070018;
        public static final int btnTextoRegresar=0x7f070016;
        public static final int btnVerImagen=0x7f07000c;
        public static final int btnVerTexto=0x7f07000b;
        public static final int button1=0x7f070017;
        public static final int linearLayout1=0x7f070000;
        public static final int mProgressText=0x7f070013;
        public static final int menu_settings=0x7f070019;
        public static final int textView1=0x7f070006;
```

```
        public static final int txtImagen=0x7f070001;
        public static final int txtInfoTitulo=0x7f070004;
        public static final int txtTexto=0x7f070014;
        public static final int txtTextoInput=0x7f070015;
        public static final int viewImagen=0x7f070003;
    }
    public static final class layout {
        public static final int activity_mostrar_imagen=0x7f030000;
        public static final int activity_mostrar_info=0x7f030001;
        public static final int activity_mostrar_texto=0x7f030002;
        public static final int activity_principal=0x7f030003;
    }
    public static final class menu {
        public static final int activity_mostrar_imagen=0x7f060000;
        public static final int activity_mostrar_info=0x7f060001;
        public static final int activity_mostrar_texto=0x7f060002;
        public static final int activity_principal=0x7f060003;
        public static final int activity_qrscanner=0x7f060004;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
        public static final int hello_world=0x7f040001;
        public static final int menu_settings=0x7f040002;
        public static final int title_activity_mostrar_imagen=0x7f040005;
        public static final int title_activity_mostrar_info=0x7f040003;
        public static final int title_activity_mostrar_texto=0x7f040004;
        public static final int title_activity_principal=0x7f040007;
        public static final int title_activity_qrscanner=0x7f040006;
    }
    public static final class style {
        public static final int AppTheme=0x7f050000;
    }
}
```