Universidad Carlos II de Madrid

Escuela Politécnica Superior

Ingeniería Técnica de Telecomunicaciones: Sonido e Imagen

Proyecto de Fin de Carrera

Desarrollo de una aplicación de transferencia de ficheros basada en NFC y Bluetooth

Autora: Laura Tolsada Bris

Tutor: Mario Muñoz Organero

Febrero 2012

El futuro pertenece a quienes creen en la belleza de sus sueños

Eleanor Roosevelt

Agradecimientos

En primer lugar quisiera dar las gracias a mis padres, Antonio y Dolores, sin ellos no podría haber comenzado esta aventura y mucho menos terminarla. Gracias por apoyarme y comprenderme durante todo este tiempo y sobre todo por darme la oportunidad de llegar hasta aquí.

A mi hermano Javi, por estar ahí siempre que lo he necesitado, y por tener esa chispa que nos alegra a todos. Y por supuesto a mis primos y tíos, que aunque todavía les cuesta saber qué he estudiado, me apoyan incondicionalmente.

A mis amigas, por todo lo que hemos compartido juntas. Gracias por escucharme, animarme y ayudarme a evadirme de todo.

A María, ¿Quién nos ha comprendido mejor que la una a la otra? La luz ya no está al final, está aquí.

A mis compañeros y sobre todo amigos de la universidad, por todos los buenos momentos que hemos pasado durante todos estos años, aunque ya queden lejos, pero sobre todo porque los seguimos pasando.

A mi novio Javi, por su paciencia, y por saber sacarme una sonrisa en cualquier momento. Gracias por ser tú.

A mi tutor y profesor Mario, por aceptarme para realizar este proyecto bajo su dirección y por su capacidad de guiar mis ideas.

Resumen

En la sociedad actual, existe una creciente necesidad de comunicarse y tener acceso a la información independientemente de la ubicación y del momento en el que nos encontremos. Según ha ido evolucionando la sociedad de la información, el papel de las comunicaciones inalámbricas ha ido creciendo en importancia y hoy en día desempeña un papel de liderazgo. Existen múltiples ejemplos que hacen uso de ellas, dos de los cuales son la televisión y las redes de datos. Sin embargo, uno de los más importantes y que más extendido se encuentra es el teléfono móvil.

El teléfono móvil es actualmente uno de los productos más populares e indispensables en la sociedad, es común que en cada hogar al menos haya uno por persona. Hace unos años era inimaginable pensar como se iba a crear tanta dependencia de los teléfonos móviles, pero echando un vistazo a las ventajas que nos proporcionan es fácil de comprender: nos liberas de los cables que nos impiden la movilidad, nos ayudan a comunicarnos en cualquier momento o lugar, ya sea mediante voz, texto o vídeo; son una herramienta de entretenimiento ya que podemos escuchar música, ver vídeos o conectarnos a internet; podemos usarlos como sustitutos de otras herramientas como tarjetas de crédito o cámaras, etc.

La movilidad y flexibilidad que proporcionan los teléfonos móviles se debe al uso de las tecnologías inalámbricas. Es común encontrar los dispositivos más básicos equipados con infrarrojos o Bluetooth, lo cual nos da la posibilidad de utilizar el dispositivo para un gran número de aplicaciones. Sin embargo, la tendencia y la actualidad es que estén equipados con muchas más tecnologías como WIFI, NFC, 3G o HSDPA. Basándonos en las grandes posibilidades que ofrecen los teléfonos, se decide la realización del presente proyecto, basado en las tecnologías NFC y Bluetooth.

NFC y Bluetooth son tecnologías de corto alcance y cada una se utiliza para un fin determinado, por lo que se pueden complementar a la perfección. La combinación de ambas tecnologías es el pilar básico de este proyecto. En él, se crea un caso práctico en el cual un dispositivo móvil se comunica con otro dispositivo y obtiene información mediante estas tecnologías de una manera intuitiva, rápida y sencilla.

Como complemento, y continuando el uso de las tecnologías de la información, se crea una aplicación Web totalmente independiente que crea los datos que posteriormente obtendrá el usuario del dispositivo móvil. Gracias a tratarse de un entorno Web es accesible desde cualquier lugar con acceso a internet dotando de comodidad y flexibilidad.

Abstract

In today's society, there is a growing need to communicate and have access to the information regardless of our location and time where we are. As the information society has been evolving, the role of wireless communications has been growing in importance and today plays a leadership role. There are many examples that make use of them, two of which are television and data networks. However, one of the most important and most widespread is the mobile phone.

The mobile phone is now one of the most popular and essential products in society, it is common that in every home has at least one per person. A few years ago was unimaginable to think about the current dependency on mobile phones, but looking at the benefits they provide, is easy to understand: we are free from the wires which stop us from mobility, they help us to communicate in any time or place, whether by voice, text or video; they are a entertainment tool as we can listen to music, watch videos or connect to the internet; we can use them as substitutes for other tools such as credit cards or cameras, etc.

Mobility and flexibility provided by mobile phones are due to the use of wireless technologies. It is common to find the most basic devices equipped with infrared or Bluetooth, which gives us the possibility of using the device for a large number of applications. However, the trend and present is that the mobiles are equipped with thousands of other technologies such as WiFi, NFC, 3G or HSDPA. Based on the great possibilities offered by the phones, we decided to carry out this project, based on NFC and Bluetooth technologies.

NFC and Bluetooth are short-range technologies and each one is used for a particular purpose, so you can complement them perfectly. The combination of both technologies is the mainstay of this project. It creates a practical case in which a mobile device communicates with another device and obtains information through these technologies in an intuitive, fast and easy way.

As a complement, and continuing the use of information technology, it is created a completely separate Web application that creates the data which the mobile device user will get later. Thanks to being a web environment, it is accessible from anywhere with internet access providing comfort and flexibility.

Índice General

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1 Motivación del proyecto

Hoy en día las tecnologías inalámbricas han cobrado gran importancia. Las personas están en constante movimiento y quieren tener la posibilidad de acceder a las redes en todo momento, ya sea para consultar información en internet, para el uso de las redes sociales o para transferencias de archivos e información entre distintos dispositivos sin necesidad de cables que nos limitan la movilidad o que tendríamos que llevar siempre con nosotros.

Dentro de este mundo inalámbrico, los dispositivos más utilizados son los teléfonos móviles. Prueba de ello es que en España, a fecha de agosto de 2011, existe mayor número de líneas que de habitantes. La funcionalidad que ofrece un teléfono móvil o *Smartphone* ha cambiado considerablemente llegando a parecerse cada vez más a un ordenador con un tamaño mucho más reducido: tienen conexión a internet que permite tareas como consultar el correo o comunicarse con las redes sociales, se pueden hacer fotos y vídeos de cierta calidad, hacer videollamadas, usarlos como GPS, trabajar con archivos de formatos office o pdf entre otros, instalar muchísimas aplicaciones y juegos, comunicarse con otros dispositivos para compartir información o archivos mediante varias tecnologías como NFC, Bluetooth, WIFI, etc.

La tecnología Bluetooth es conocida por la mayoría de las personas ya que viene incluida en prácticamente todos los dispositivos móviles, consiste en una Red Inalámbrica de Área Personal para la comunicación de corto alcance (10 metros sin obstáculos). Su uso está muy extendido en varias disciplinas como la transmisión de datos y archivos de diversas clases, transmisión de audio de forma inalámbrica, creación de redes inalámbricas personales, control remoto, etc.

La tecnología NFC es bastante más reciente que Bluetooth. Se trata de una extensión del sistema de etiquetas RFID, que permite el intercambio de datos entre dispositivos a corto alcance (funciona a unos 10 cm de distancia). Aunque lleva varios años de existencia, es relativamente nueva porque no ha sido hasta el 2011 cuando parece que está despertando mayor interés. Varios fabricantes de telefonía móvil la están incluyendo en sus modelos con mayores prestaciones y otras empresas encargadas de desarrollo de software están realizando proyectos relacionados con la comunicación mediante NFC, principalmente a su uso relacionado con pagos seguros mediante el móvil.

La comunicación mediante NFC se produce entre dos dispositivos que estén preparados para ello, en este proyecto se ha usado el *Nokia 6131 NFC* que trae el chip NFC integrado en el dispositivo y para llevar a cabo la comunicación se ha hecho uso de una tarjeta o tag externo. Sin embargo, la integración del chip en la circuitería del móvil no es la única manera de tener conexión NFC, hay otras formas de hacer compatibles nuestros terminales como el uso de tarjetas SIM que integran el chip o mediante pegatinas con el chip que se podrían adherir al móvil.

Las tecnologías mencionadas anteriormente pueden dar lugar a diversas aplicaciones haciendo uso de ellas por separado pero **el objetivo de este proyecto es la combinación de ambas, NFC y Bluetooth**. Una de las características del Bluetooth es la transferencia de archivos entre dispositivos, para ello se necesitan una serie de pasos previos de descubrimiento y emparejamiento que permitan esta comunicación. Todos estos pasos se pueden automatizar la primera vez que se de la comunicación entre los dos dispositivos si estos se vinculan, por lo que se reduciría el tiempo para la transferencia entre transmisor y receptor, pero siempre habría una primera toma de contacto manual para la búsqueda. Por este motivo, se pensó en la combinación de las dos tecnologías, ya que una cualidad muy importante de NFC es que puede ejecutarse automáticamente sin necesidad de interacción con el usuario ni emparejamiento. Cuando se detecta otro dispositivo o tarjeta NFC en sus proximidades comienza la comunicación. Teniendo en cuenta estas premisas, se pensó en la aplicación de un caso práctico de transferencia de ficheros para trabajadores de una empresa de transportes indicando su itinerario del día haciendo uso de Bluetooth pero abriendo la comunicación automáticamente mediante NFC, actuaría como una pasarela de conexión.

El proyecto tiene dos partes diferenciadas, una referente a los usuarios de los dispositivos móviles que reciben el fichero desde un equipo y para los cuales es todo prácticamente transparente, y otra que se encarga de gestionar los itinerarios que se van a enviar al usuario. Se combinan la utilización de una aplicación J2ME, una aplicación Java de escritorio y la aplicación de gestión se ha creado mediante tecnologías Web.

Esta aplicación permitirá a los transportistas obtener su itinerario diario de una forma rápida y sencilla haciendo uso de su teléfono móvil de empresa, que es algo que van a llevar siempre con ellos. Aparte, tiene la ventaja de poder consultarlo en cualquier momento dentro de su ruta diaria. Por otro lado, es muy útil para los trabajadores encargados de gestionar esas rutas teniendo toda la información centralizada y con fácil accesibilidad para poder hacer modificaciones o consultas en cualquier momento.

1.2 Objetivos

El objetivo de este proyecto es la creación de varias aplicaciones para móvil y para PC. Estas aplicaciones combinan la utilización de las tecnologías inalámbricas NFC y Bluetooth para crear la comunicación entre dos dispositivos de distinta naturaleza como son el ordenador y un móvil. Como extra, el proyecto se complementa con el desarrollo de una aplicación web para administrar usuarios.

Para la consecución del objetivo general, se lleva a cabo el desglose en los siguientes objetivos menores:

1. Analizar las tecnologías inalámbricas que se van a usar, así como otras presentes en el proyecto. Se estudiarán sus características técnicas, estado actual de uso y su posible aplicación al proyecto.

2. Estudiar las posibilidades, ventajas y beneficios de la combinación de las tecnologías NFC y Bluetooth para sacar el máximo partido de ellas.

4. Desarrollo de las diferentes aplicaciones para el móvil y PC: aplicación para grabar los datos en la tarjeta NFC, aplicación para leer los datos e iniciar la comunicación automática con otro dispositivo Bluetooth y la aplicación que implementa un servidor Bluetooth en el PC a la espera de conexiones entrantes.

6. Desarrollo de la aplicación Web como soporte a la administración de la creación de rutas y gestión de usuarios.

7. Pruebas y validación del funcionamiento en entornos reales y simulados con el material necesario:

- Comunicación NFC de lectura y escritura.
- Uso de Bluetooth por separado y junto a NFC. Conexión con el PC.
- Comunicación real entre el PC y un servidor donde se almacenan los ficheros que serán transferidos al usuario. Descarga de ficheros mediante FTP y su posterior envío al móvil.
- Funcionamiento de una aplicación Web con diferentes funcionalidades integradas con el uso de formularios.

8. Posibles líneas de desarrollo futuro.

1.3 Fases del proyecto

El proyecto se estructura en un conjunto de fases que describen los aspectos del desarrollo del mismo. Estas tareas se irán desarrollando a lo largo del tiempo, la distribución de las mismas se verá en el punto 7.2. A continuación se muestra una descripción de cada fase:

Fase 1: *Búsqueda y lectura de documentación*. Se trata de un análisis previo y documentación respecto a las tecnologías (NFC, Bluetooth, Java, etc.) y los recursos que podemos obtener para trabajar con ellas.

Fase 2: *Configuración del entorno de trabajo*. En esta fase se instala y configura todo el software necesario para el desarrollo e implementación del proyecto.

Fase 3: *Implementación de la aplicación NFC*. Desarrollo de las aplicaciones para móvil de escritura y lectura NFC.

Fase 4: *Implementación de la aplicación Bluetooth*. Desarrollo de la aplicación móvil para la conexión Bluetooth y la aplicación para PC como servidor Bluetooth y obtención de ficheros.

Fase 5: *Integración de las aplicaciones*. Integración de las aplicaciones móviles con funcionalidad de lectura NFC y conexión Bluetooth que se complementan entre sí.

Fase 6: *Implementación de la aplicación WEB*. Desarrollo de la aplicación WEB con funcionalidad extra para la gestión de los itinerarios y de los trabajadores.

Fase 7: *Pruebas y depuración*. Realización de una batería de pruebas para comprobar el correcto funcionamiento de las diferentes aplicaciones y del sistema global. Soluciones en los casos necesarios.

Fase 8: *Documentación*. Redacción de la memoria donde se describen las tecnologías utilizadas, la arquitectura de la aplicación que se ha implementado y los detalles de la misma.

1.4 Estructura de la memoria

A continuación se muestra un breve resumen del contenido de la memoria:

El comienzo con el *Capítulo 2 (Estado del Arte)* realiza un análisis exhaustivo de las tecnologías usadas como son NFC y Bluetooth. Además se detallan el resto de plataformas y tecnologías del desarrollo como J2ME, J2SE, FTP, PHP y el lenguaje de etiquetado HTML junto con CSS.

En el núcleo central de la memoria entramos completamente en la descripción del proyecto realizado. El *Capítulo 3 (Descripción de la aplicación)* presenta el desarrollo de la aplicación a distintos niveles, se describe una visión general de la arquitectura con sus diferentes módulos para a continuación profundizar con detalles de más bajo nivel que muestran el diseño software de la misma. El *Capítulo 4 (Descripción Funcional)* muestra la parte visual de la aplicación, un recorrido por los diferentes módulos y las funciones que se podrían dar en los distintos escenarios para finalmente terminar con la descripción de las pruebas realizadas en el *Capítulo 5 (Pruebas)*.

Para finalizar, se encuentran los dos últimos temas tratados, *Capítulo 6 (Conclusiones y líneas futuras) y Capítulo 7 (Presupuesto y planificación).* En el primero se exponen las conclusiones obtenidas tras la realización del proyecto y las posibles líneas de desarrollo que se podrían dar para ampliarlo y mejorarlo. Finalmente se detallan los costes y tiempos que supondrían la realización del proyecto.

Como material extra se encuentran la *bibliografía*, un *glosario de términos* y un *Anexo* referente a la instalación del entorno para llevar a cabo el proyecto.

Capítulo 2

Estado del Arte

En este capítulo sobre el estado del arte se van analizar y estudiar las diferentes tecnologías que se utilizan para la realización del proyecto. El proyecto consta de una parte principal basada en tecnologías inalámbricas de corto alcance que son **NFC** y **Bluetooth** y una aplicación web secundaria de apoyo.

La tecnología NFC se utiliza en la aplicación móvil y es el punto de partida de toda la comunicación. Por otro lado, la tecnología Bluetooth se utiliza tanto en la aplicación destinada al dispositivo móvil como la aplicación de escritorio para hacer posible la comunicación entre ambos y el envío de la información.

Respecto a los datos que se van a transmitir al usuario, están previamente almacenados en un servidor remoto por lo que se utiliza el protocolo **FTP** para su descarga.

Toda la funcionalidad principal, que comprende las tecnologías mencionadas hasta ahora, está desarrollada en el lenguaje de programación Java, tanto en **Java ME** para los MIDlets como **Java SE** para la aplicación de escritorio.

Finalmente, en la aplicación secundaria se han utilizado tecnologías Web por lo que es accesible desde el navegador Web. El desarrollo se ha llevado a cabo con **PHP** y **HTML**.

En los siguientes puntos del capítulo se explicará en detalle cada una de las tecnologías para facilitar su comprensión.

2.1 Near Field Comunication

2.1.1 Breve introducción

Near Field Communication, más conocido como *NFC,* es una tecnología de comunicación inalámbrica de corto alcance y alta frecuencia, trabaja a menos de 20cm de distancia y a 13,56MHz. Esta tecnología surgió en el año 2002 debido a la necesidad de hacer compatibles otras tecnologías *contacless* existentes hasta ese momento como FeliCa de Sony y MiFareTM de Philips.

La tecnología NFC está basada en el estándar *ISO 14443*, comúnmente conocido como *RFID (Radio Frecuency Identification)*. Es un sistema de almacenamiento y recuperación de datos remotos que hace uso de tarjetas, etiquetas, transpondedores o tags RFID. Ejemplos de uso de esta tecnología son muy comunes en la vida diaria: el abono transporte, llaves de automóvil con antirrobo o llaves de acceso a habitaciones en un hotel.

NFC tiene menor alcance que RFID, aunque ambas utilizan el mismo tipo de chips y son muy similares. La diferencia fundamental es que NFC tiene capacidad de cómputo y puede realizar operaciones, por lo que se hace ideal su integración en un smartphone el cual nos proporciona el hardware y software necesarios. De esta manera, por ejemplo, no habría necesidad de utilizar una tarjeta para realizar pagos o para acceder a la habitación del hotel, podríamos hacerlo con nuestro smartphone y su chip NFC, ya que emularían la tarjeta convirtiéndose el móvil en ella. La emulación de tarjetas es una de las posibilidades que nos brinda NFC, otro modo de uso sería como lector de etiquetas RFID que podrían contener cualquier tipo de información para visualizar en el móvil y en modo *peer to peer (P2P)* entre dos dispositivos con conexión NFC.

Figura 2.1. El Smartphone es la tarjeta

La comunicación mediante NFC no está pensada para el intercambio de grandes cantidades de datos dado que su velocidad de transferencia puede llegar hasta los 424 kbps, para esto ya existen otras tecnologías como Bluetooth o WIFI que funcionan muy bien. Con un solo toque se puede iniciar la comunicación entre ambos dispositivos NFC por lo que podría ser muy útil como combinación de otras tecnologías como Bluetooth que necesita unos pasos previos de emparejamiento para el comienzo de la comunicación y con NFC eso se podría dar automáticamente. [1]

Figura 2.2. Diferentes usos de NFC

2.1.1 RFID

RFID y NFC

NFC es un nuevo estándar que nace a raíz de RFID, conserva muchas de sus características y funciones pero no está creado para sustituir a RFID, es un complemento. NFC es una forma de aumentar las posibilidades que puede brindar la tecnología RFID, su principal punto a favor es la integración de NFC con los teléfonos móviles y la mayor seguridad y privacidad al reducir considerablemente el rango de trabajo a escasos centímetros. Por todo esto, es necesario hacer un análisis de RFID y comprender así los principios en los que se basa la tecnología NFC. [2]

El significado de las siglas *RFID* es *Radio Frecuency Identification (Identificación por Radio Frecuencia)* y nos da una ligera idea de cuál es el cometido de esta tecnología. RFID es un término genérico usado para describir un sistema que transmite una entidad (un número único de serie) de una persona u objeto de manera remota, por esto está incluida dentro de la categoría de tecnologías de identificación automática.

RFID tiene muchas aplicaciones hoy en día y aunque podría pasar desapercibido, está ahí para hacer ciertos trabajos y aspectos de la vida diaria más fáciles y rápidos. Podemos encontrar chips RFID en prevención de robos de automóviles, en acceso a edificios, seguimiento de libros en bibliotecas, chip de identificación para animales domésticos, etc.

Las principales diferencias entre NFC y RFID son:

- Capacidad de computo de NFC
- Posibilidad de comunicación entre dos dispositivos NFC sin necesidad de utilizar un lector
- Integración en dispositivos como móviles, permitiendo la emulación
- Rango de trabajo que implica seguridad inherente
- Comunicación en ambos sentidos

Historia

Aunque esta tecnología esté muy presente hoy en día, dio sus primeros pasos en la Segunda Guerra Mundial por los británicos. Debido a la necesidad de identificar a los aviones como amigos o enemigos, colocaban el tag RFID en la aeronave para ver si daba la respuesta adecuada, a esta tecnología se la conocía como *IFF*. En ese tiempo la tecnología resultaba demasiado cara para fines comerciales, por ello no se ha utilizado de forma generalizada hasta tiempos relativamente recientes, cuando su costo se ha reducido.

RFID tuvo un mayor crecimiento y desarrollo durante los años 70, en 1973 Charles Watson patentó la tecnología que actualmente conocemos como RFID pasiva.

Durante esta década nació también la que quizá es la primera aplicación comercial con RFID, de mano de las empresas *Sensormatic* y *Checkpoint.* Desarrollaron el control electrónico de artículos (*EAS*) para evitar robos, se pudo llegar a este desarrollo comercial debido a que utilizaron tags con un único bit haciendo así el uso de RFID más económico. Más tarde, se desarrollaron sistemas de identificación de transportes como flotas de camiones y ferrocarriles entre otros, aparte de la identificación de vehículos como precursor de los sistemas de *Telepeaje* actuales.

Lo que actualmente conocemos como RFID entró en vigor en 1999 cuando se creó el Laboratorio de Identificación Automática (Auto-ID Lab) en el Instituto de *Tecnología de Masachusets (MIT)* financiado por Coca-Cola, Wal-Mart, Gillette y Sun Microsystems entre otros. Crearon un sistema de sustitución a los códigos de barras, combinando la tecnología RFID y EPC (Códigos de Producto Electrónico). Los nuevos tags RFID que el centro Auto-ID Lab creó, almacenan el código electrónico de producto unívoco en un microchip, este código se transmite a través de una antena a los lectores RFID. El centro desapareció en 2004 después de haber conseguido uno de los principales objetivos para el que fue creado, la coordinación y definición de los estándares técnicos que rigen la tecnología RFID. [3]

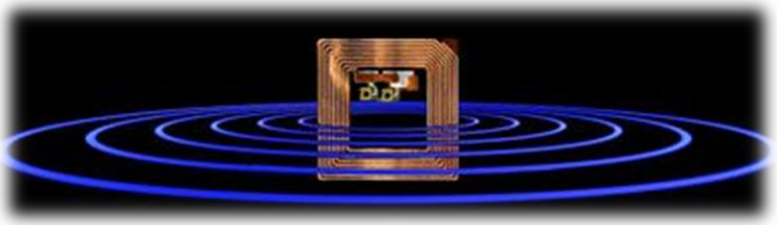


Figura 2.3. Simulación ondas en tag RFID

Respecto a la estandarización de RFID, actualmente existen cuatro aspectos importantes que determinan la compatibilidad de equipos: protocolo de comunicación (determina el modo en que las etiquetas y los lectores se comunican entre sí), contenido (especifica el formato y la semántica de los datos que se comunican), certificación (los productos RFID tienen que superar unas pruebas de calidad que garanticen el cumplimiento de los estándares y la interoperabilidad con otros fabricantes) y aplicaciones (la precisión de la frecuencia es clave y puede ser susceptible a diferentes materiales).

La *ISO* y *EPC* son los grupos de estandarización RFID competidores e incompatibles entre sí aunque ambos persiguen conseguir etiquetas de bajo coste que operen en UHF. Por otra parte EPC Global está desarrollando una nueva generación de estándares llamados Gen2 para lograr la interoperabilidad entre ambos estándares.

La regulación de RFID está fijada por cada país ya que no existe ninguna corporación púbica encargada de fijar una normativa internacional. En Europa las encargadas de llevarlo a cabo son *ERO, CEPT, ETSI* y las administraciones públicas. Las administraciones nacionales tienen que ratificar el uso de una frecuencia específica antes de que pueda ser utilizada en dicho país. Las etiquetas de alta y baja

frecuencia se pueden usar globalmente sin necesidad de licencia pero en el caso de UHF viene determinado por la legislación de cada país.

Actualmente los códigos de barras son la tecnología más usada y extendida para la identificación de productos. En comparación con la tecnología RFID, presenta algunas desventajas como el tener una información fija no reprogramable y la pequeña cantidad de datos que pueden almacenar. El uso de RFID a la hora de realizar la compra para el usuario final es una gran ventaja y proporciona mucha comodidad y rapidez de uso, no es necesario el movimiento de los productos del carro para confeccionar el ticket y hay mayor seguridad respecto a los errores que pudieran surgir. Por otro lado, esto provoca la reducción de puestos de trabajo en caja para los establecimientos lo que supondría un ahorro en los costes para el establecimiento pero pérdida de puestos de trabajo.

Componentes y arquitectura

Como definición básica podemos decir que RFID es un método simple y automático de recoger datos sobre un determinado activo o producto de una forma rápida y fácil, sin requerir la intervención de las personas.

Para el funcionamiento de RFID se necesitan una serie de componentes que permitan la comunicación: tag o etiqueta (también conocido como transpondedor), antena y lector RFID. Las etiquetas son las que almacenan los datos de identificación del objeto que las contiene y generan una señal de radiofrecuencia para ser captada por un lector RFID, ambos hacen uso de antenas para llevar a cabo el proceso. La antena permite que se transfiera la información del chip a un lector y este convierte las ondas de radio de las etiquetas RFID en datos legibles para un ordenador.

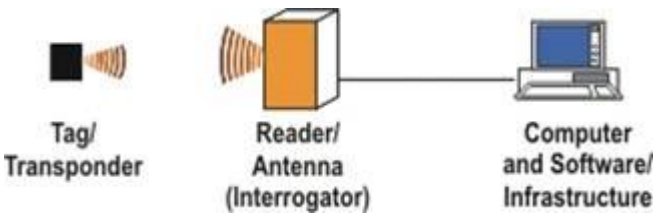


Figura 2.4. Componentes de un sistema RFID

Los componentes básicos de las etiquetas son un chip o circuito integrado donde se almacena la información y una pequeña antena. Sin embargo, existe otra modalidad de etiquetas que carecen del chip, son las llamadas *Chipless.* En este caso las etiquetas se crean con pequeñas fibras de aleaciones que son colocadas durante la fabricación de ciertos productos tales como papel, plástico, vidrio, cartón, etc. Este tipo de etiquetas son más baratas que las que contienen chip y mucho más versátiles ya que pueden ser colocadas en materiales muy diversos e incluso ser invisibles al ojo humano.

Figura 2.5. Etiqueta RFID

En la actualidad hay una gran variedad de etiquetas para poder escoger la más apropiada en cada caso y están clasificadas en función de diversos factores como el origen de su fuente de alimentación, tipo de memoria de almacenamiento (lectura o lectura/escritura), frecuencias de trabajo (más propio del sistema RFID general que de la etiqueta en sí), características físicas de las etiquetas, etc. Una de las más representativas es la clasificación por fuente de alimentación:

- **Etiquetas pasivas**: este tipo de etiquetas son fáciles de producir y son las más económicas. No poseen fuente de alimentación interna, de manera que utilizan la energía que reciben del lector la cual induce una pequeña corriente eléctrica que es suficiente para operar el circuito integrado y para que se genere y transmita la respuesta mediante la antena del tag.

  La mayoría de los tags pasivos utiliza backscatter (reflexión de la señal en la dirección que había sido transmitida) para comunicarse con el lector, por lo que la antena tiene que estar diseñada para obtener la energía que necesita para funcionar y a la vez poder transmitir la respuesta por backscatter. El rango de lectura de este tipo de etiquetas es de unos 5 metros y es menor que en los tags activos que pueden llegar a 500 metros, esto es debido a la dependencia de la energía electromagnética necesaria tanto para el funcionamiento como para la comunicación.

- **Etiquetas semi-pasivas**: estas etiquetas son similares a las pasivas aunque también toman parte del funcionamiento de las activas, son etiquetas asistidas por batería. Hacen uso de una pequeña batería para alimentar el chip del circuito pero para la transmisión utilizan la energía del lector. Esta variante de etiqueta mejora el tiempo de respuesta respecto a las pasivas y aumenta el rango de lectura. Gracias a la batería obtienen características de los tags activos como mayor memoria, capacidad de procesamiento adicional e incluso tienen mayor duración. Estos tags aún resultan poco económicos para ser usados en productos de bajo coste.

- **Etiquetas activas**: este tipo de tags tienen autonomía respecto a la energía que necesitan ya que tienen una fuente de alimentación propia (que suelen ser baterías) para dar corriente a sus circuitos y transmitir la señal.

Las ventajas que presentan respecto a las otras clases de tags son numerosas, aunque siempre teniendo presente que dependen de la vida útil de su batería para funcionar, que es menor que en los tags pasivos, y sobre todo la mayor desventaja es el precio el cual es muy superior. Sin embargo tienen muchos puntos a favor como la posibilidad de poder leerse a grandes distancias pudiendo generar respuestas claras aunque la señal inicial sea débil, pueden enviar señales mucho más potentes por lo que son más eficientes en entornos de difícil propagación de la señal como el agua o el metal, son mucho más fiables ya que poseen la capacidad de establecer sesiones, algunos tags son capaces de comunicarse entre sí, pueden contener sensores para controlar aspectos como la temperatura y tienen mayor capacidad de almacenamiento y funciones de procesamiento.

Dependiendo del diseño, un tag activo puede transmitir su identificación y datos adicionales sin ser interrogado por un lector, a estos tags se les llama beacon tags. Transmiten su señal periódicamente a intervalos preestablecidos mientras que los tags activos normales esperan a que el lector los consulte para activarse y transmitir su señal. [4]

Usos y seguridad de RFID

La tecnología RFID está muy extendida hoy en día y tiene múltiples usos diferenciados por la frecuencia en la que se trabaja ya que esta afecta directamente al coste y alcance del sistema. En el caso de las bajas frecuencias los sistemas RFID son mucho más económicos pero a la vez tienen un rango de trabajo más reducido y menor velocidad, lo contrario ocurre en los sistemas que trabajan en frecuencias más altas.

Algunos ejemplos de uso son:

- Identificación de animales como mascotas, ganado o especies en peligro: se usan chips pasivos con una vida indefinida, muchas mascotas llevan un microchip subcutáneo implantado que permite su identificación en caso de extravío y en el caso del ganado se utilizan crotales.

Figura 2.6. Crotales para ganado

- Sistemas de control de acceso: sustituyen a las tarjetas de control de acceso con banda magnética. Controla el acceso a edificios de los empleados además de la gestión y el control de horarios, etc.

Figura 2.7. Control de acceso a edificios RFID

- Identificación de pacientes: una pulsera identificativa de radiofrecuencia permite tener localizado al paciente en cualquier área del hospital y además permite acceder a su historial.

Figura 2.8. Pulsera identificativa hospitalaria

- Logística y transporte: la logística es una de las aplicaciones más importantes ya que mediante RFID se puede tener localizado cualquier producto dentro de la cadena de suministro, aparte se puede ir almacenando información de los estados del producto ya que se puede grabar en las etiquetas.

Figura 2.9. Uso de RFID en logística

Aparte de los descritos anteriormente, existen otros usos como la gestión y seguimiento de documentos, sistemas de pago electrónico, sistemas de control y cobro de peajes, seguimiento de contenedores (puertos, zonas francas, etc),

trazabilidad de medicamentos, control de equipajes en aeropuertos, llaves antirrobo para coches, sistemas antirrobo e inventario en el sector textil u otros sectores, etc.

Figura 2.10. Telepeaje y control de maletas RFID

El uso de esta tecnología plantea mejoras en la eficiencia y comodidad en los sistemas de uso diario pero también hay que tener en cuenta los posibles riesgos para la seguridad y privacidad personal.

Uno de los riesgos presentes en todas las tecnologías inalámbricas deriva de la exposición a las radiaciones, en este caso, según estudios de la OMS, los campos que se crean entre lector y etiqueta tienen un riesgo mínimo para la salud. Como dato, un sistema RFID emite entre 200000 y 10000 veces menos potencia que un teléfono móvil. [5]

Los riesgos de seguridad se centran en los posibles ataques que podría sufrir un sistema RFID como pueden ser el aislamiento de etiquetas evitando la comunicación, clonación de etiquetas, denegación del servicio con envío masivo de datos, etc. Algunas medidas para evitar los problemas de seguridad podrían ser: mejorar la memoria y capacidad de procesamiento de las etiquetas para poder implementar mecanismos avanzados de seguridad y cifrado, evitar la escritura directa de datos en los tags haciéndolo en las bases de datos o usar tags de solo lectura, utilizar técnicas de autentificación, etc.

Si se llevan a cabo los ataques antes mencionados u otros similares, también se podría obtener la información que contienen las etiquetas fuera de su uso principal o averiguar hábitos personales de los compradores de productos poniendo así en riesgo su privacidad personal. Muchos consumidores no son conscientes de que los productos que adquieren llevan incorporados tags RFID que incluso después de su compra pueden continuar activos y producirse así la intromisión en sus datos. Por esto, las personas pueden ser reticentes a utilizar este tipo de tecnologías por miedo a poner en riesgo su información personal.

La información personal es totalmente confidencial por lo que las medidas para evitar los riesgos sobre la privacidad son una prioridad para las organizaciones. Como medida extra se puede evitar incluir información personal en las etiquetas y ofrecer

mayor información a los usuarios respecto al uso de RFID (como los lugares y productos en los que está presente) para que haya un mayor conocimiento de la tecnología y su funcionamiento. Respecto a los usuarios, si tienen mayor información, serán capaces de protegerla con métodos de aislamiento como fundas de materiales metálicos o plásticos que hagan la función de "jaula de Faraday" para que solo sea utilizado en los casos necesarios.

2.1.2 Características NFC

NFC se caracteriza por una serie de **aspectos generales** que definen esta tecnología y que proporcionan beneficios a los usuarios y a las empresas [6]:

- Se trata de una **tecnología intuitiva** ya que únicamente requiere un toque para comenzar su comunicación. De esta manera no es necesario que los usuarios tengan un conocimiento profundo del dispositivo y hace que la utilización de NFC se extienda entre un mayor número de personas.

- Es **versátil** ya que puede tener cabida en un amplio rango de industrias y entornos. Como ejemplo puede usarse para el control de acceso a edificios, transportes, pagos, intercambio de información, etc.

Figura 2.11. Versatilidad de NFC

- Se trata de una **tecnología abierta** y basada en estándares. Las bases sobre las que se asienta NFC siguen las normas ISO, ECMA y ETSI.

- Tiene un gran **alcance y disponibilidad** ya que puede ser integrado en cualquier teléfono móvil, no necesita que sea uno de última generación. Esto es así debido a la integración de NFC en los móviles ya que si no lo traen de serie en la circuitería del dispositivo se puede añadir en la

tarjeta SIM o mediante pegatinas adheridas al móvil. Japón es uno de los países donde el uso de NFC está más extendido y se usan este tipo de pegatinas NFC adheridas al móvil para que cualquier usuario pueda usar los servicios NFC disponibles.

Figura 2.12. Stickers NFC

- **Facilita el uso de otras tecnologías** como Bluetooth, WIFI, etc. Las complementa haciendo un uso más sencillo, por ejemplo se podría abrir el navegador del móvil con una web determinada o en el caso de bluetooth se podrían emparejar dos dispositivos y comenzar la comunicación entre ellos sin interacción con el usuario.

- NFC está dotada de una **seguridad inherente** debido a la corta distancia a la que trabaja. Los rangos de trabajo van desde el simple contacto a pocos centímetros y necesita que el propio usuario lleve a cabo la acción de acercar el teléfono a otro dispositivo o estación NFC.

- Es capaz de **intercambiar información con otras infraestructuras** sin contactos. El despliegue de NFC es una extensión de los servicios que ya existen pero proporciona más posibilidades gracias al teléfono móvil y la conexión a Internet.

- Está **preparada para la seguridad** que necesitan otras aplicaciones externas ya que ha incorporado capacidades para dar soporte a aplicaciones de seguridad.

Aparte de los aspectos generales, NFC tiene una serie de **especificaciones técnicas básicas** que hay que conocer:

Especificaciones básicas NFC	
Alcance	Hasta 20 cm aunque es más efectivo a menos de 10cm
Tiempo de conexión	< 0,2s
Frecuencia	13,56 MHz
Velocidad de transmisión	106, 212 o 424 kbps
Estándares	ISO/IEC 18092, ECMA-340 y ETSI

Tabla 2.1. Especificaciones técnicas básicas NFC

Como se ha señalado anteriormente, NFC es una tecnología inalámbrica que usa radio frecuencia. Opera dentro de la banda *ISM (Industrial, Scientific and Medical)* en la frecuencia de 15.56 MHz con un ancho de banda de casi 2 MHz que está globalmente disponible y no regulada, por lo tanto no se requieren licencias para operar en ella.

La comunicación NFC es *half-duplex,* se utiliza el mismo canal para el envío y la recepción por lo que la comunicación solo ocurre en un sentido. Para prevenir transmisiones simultáneas y colisiones, el protocolo define métodos específicos de escucha antes de comenzar la transmisión, pero teniendo en cuenta el corto rango de trabajo de NFC no serán protocolos tan complicados como en otras tecnologías inalámbricas. Solo se transmitirá si previamente se comprueba que no hay otros dispositivos transmitiendo.

Aparte de definir los protocolos encargados del control de la colisión los estándares que componen NFC especifican otras características importantes como los esquemas de inicialización y modulación, codificación, velocidades de transferencia, formato de la trama y el protocolo de transporte.

Debido a la aceptación de NFC y sus múltiples aplicaciones el sistema ha sido estandarizado por una serie de organismos de estándares globalmente aceptados como ISO (18092), ECMA (340) y ETSI. Además NFC es compatible con los protocolos de tarjetas inteligentes de Philips (MIFARE -ISO 14443 A-) y Sony (FeliCa)

2.1.3 Arquitectura de NFC

Un móvil NFC contiene los principales componentes comunes en cualquier dispositivo móvil y aparte los correspondientes al módulo NFC. Estos son un chip NFC junto con una antena para hacer posible la comunicación con el exterior y el chip denominado *elemento seguro* con características de seguridad similares a los encontrados en las tarjetas inteligentes:

- **Chip NFC y antena**: se trata de un chip de silicio que junto con la antena llevan a cabo el contacto con otros sistemas NFC. Se crea un campo magnético sin necesidad de contacto o a muy poca distancia y comienzan a compartir datos. El chip está conectado a la *banda base (baseband)* del teléfono que es la parte que se encarga de las comunicaciones de móvil propiamente dichas (el envío de voz y datos), por lo tanto, es una parte muy importante del dispositivo, el cerebro del móvil.

- **Elemento seguro** (SE, Secure Element): es un chip aparte que contiene el procesador de seguridad y se encarga de llevar a cabo de forma segura las transacciones. A diferencia de un procesador de PC, su único propósito es permitir transacciones seguras. Debido a esto, el elemento seguro contiene las aplicaciones que se basan en claves de seguridad.

  Existen varias implementaciones para el elemento seguro dentro de un móvil. Se puede encontrar basado en la tarjeta SIM, integrado en la circuitería del teléfono móvil o como tarjeta de memoria externa.

La figura mostrada a continuación muestra esquemáticamente los componentes de un móvil con NFC y la comunicación con el exterior:

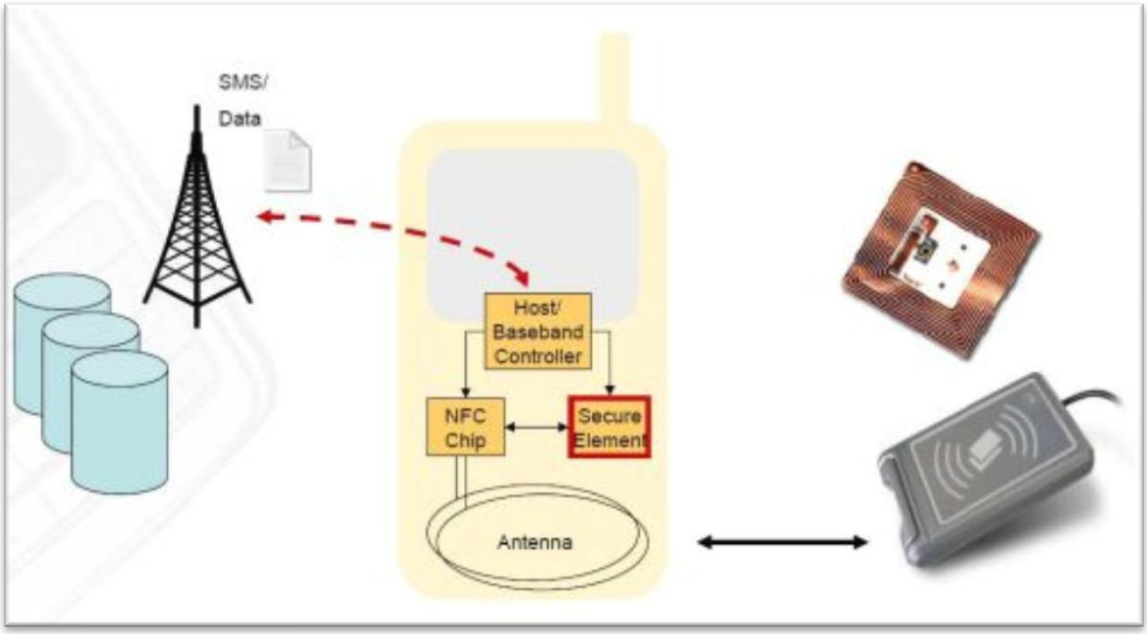


Figura 2.13. Comunicación sistema NFC en dispositivos

La arquitectura de la tecnología NFC es sólida, y a pesar de que tiene características similares a RFID, se puede decir que NFC es una tecnología única pudiendo trabajar en tres configuraciones diferentes, lo que hace que sea más versátil y eficiente que otras.

En la siguiente figura se observan los tres modos de operación que son: comunicación **peer-to-peer**, **lectura/escritura** y **emulación de tarjeta inteligente**. Antes de pasar a analizarla, conviene hacer referencia a la institución NFC Forum ya que es donde nació la definición de la arquitectura NFC que se conoce. Se trata de una asociación sin ánimo de lucro que promueve el uso de la tecnología inalámbrica de corto alcance NFC en electrónica de consumo, dispositivos móviles y PC (en el capítulo 2.2.6 se verá con mayor detalle). [7]

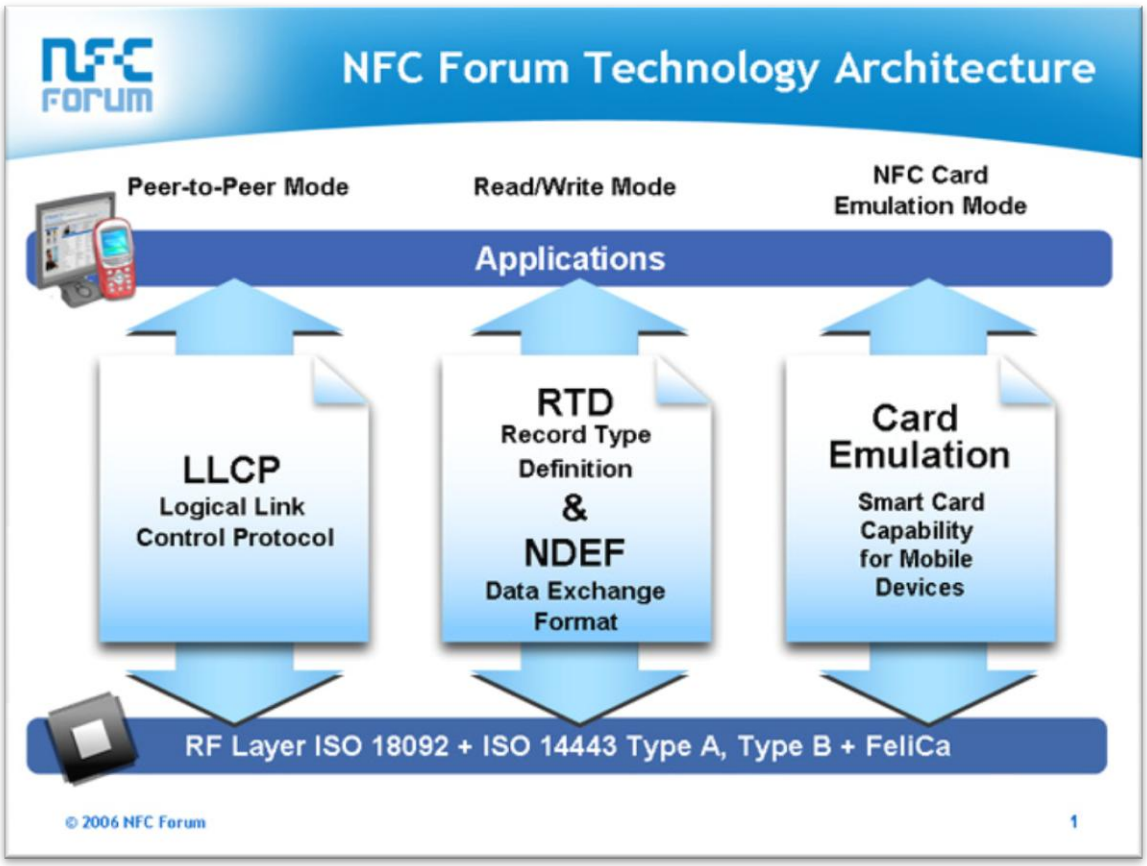


Figura 2.14. Configuraciones sistema NFC

- **Modo Peer to Peer**: este modelo de comunicación soporta la comunicación dispositivo a dispositivo a nivel de capa de enlace. Se produce el intercambio de pequeñas cantidades de datos. Si se quiere intercambiar un mayor volumen de información es preferible utilizar otras tecnologías como Bluetooth o WIFI y utilizar NFC para establecer los parámetros que posibiliten estas otras tecnologías.

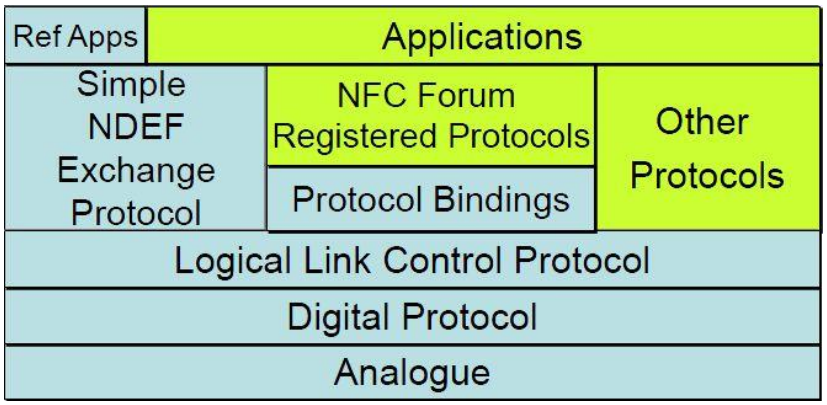


Figura 2.15. Torre modelo Peer to Peer

En esta configuración de NFC se utiliza **LLCP** *(Logical Link Control Protocol)* que trabaja en modo balanceado asíncrono. Cada nodo inmerso en la comunicación se encarga de la activación, supervisión y desactivación de la misma y puede comenzar la transmisión en cualquier momento, sin permiso del otro lado.

El protocolo de intercambio simple NDEF (**Simple NDEF Exchange Protocol**) se utiliza para enviar mensajes con el formato NDEF (se hablará de él en el punto *2.2.8*) en el modo Peer to Peer, al igual que en las especificaciones de operación de los tipos de etiquetas NFC.

Los Enlaces de Protocolo (**Protocol Bindings**) ofrecen enlaces estándar, como por ejemplo números de puertos, para permitir la interoperabilidad entre los protocolos registrados en el NFC Forum. A su vez, NFC Forum define un enlace a LLCP para los protocolos registrados (**NFC Forum Registered Protocols**), por ejemplo *IP* y *OBEX*. Finalmente, cualquier otro protocolo podría ejecutarse en la capa de enlace pero NFC Forum no proporcionaría el enlace a LLCP.

Respecto a las aplicaciones Peer to Peer, existen unas de referencia del Forum (**Ref Apps**) que corren sobre NDEF pero aparte, existen otras como por ejemplo imprimir desde una cámara, intercambiar una tarjeta de negocios, aplicaciones basadas en NDEF de terceros, etc.

- **Modo lectura/escritura**: en este modelo NFC permite transferir datos en un formato de mensaje definido por el Forum. Se define la forma de leer y escribir datos NDEF desde o a una etiqueta. Hay cuatro tipos de etiquetas que pueden ser leídas y que se verán con mayor profundidad en el apartado *2.2.5* (**Type 1-4 Tag Operation**).

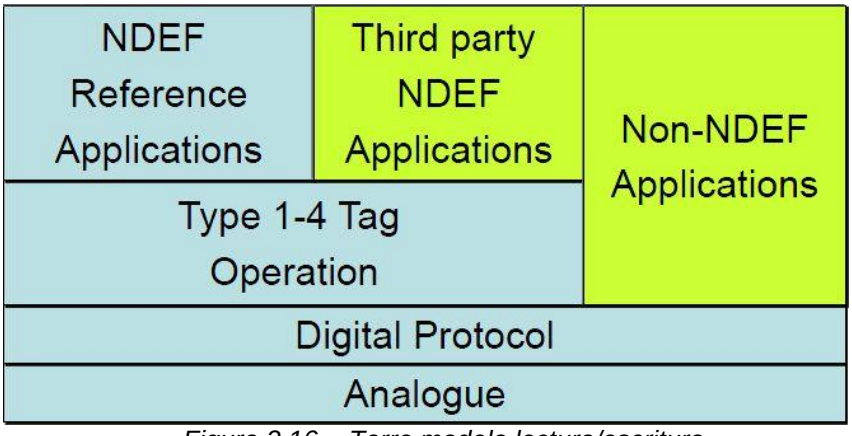


Figura 2.16. Torre modelo lectura/escritura

Existen varios tipos de aplicaciones de escritura/lectura que se diferencian entre las de referencia del NFC Forum (NDEF Reference Applications) como los posters inteligentes o facilitar la conexión de otras tecnologías (handover), las aplicaciones de terceros que también

se basan en NDEF (NDEF Applications) como la lectura de información en botes de medicina o aplicaciones que no usan NDEF (Non-NDEF Applications) que se comunican con tarjetas sin contacto como un lector de tickets sin contacto.

- **Modo emulación de tarjetas inteligentes**: este modo permite que el dispositivo se comporte como una tarjeta inteligente estándar. Gracias al elemento seguro la transferencia de datos es segura y se pueden realizar acciones con el banco, compras rápidas, accesos seguros, etc.

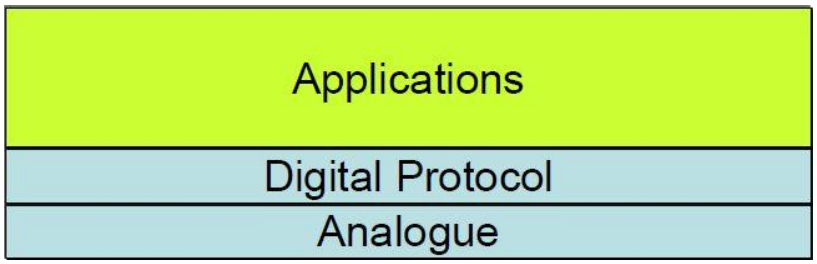


Figura 2.17. Torre de modo emulación de tarjetas

Además de los modelos de comunicación se muestra la **capa de RF** (radiofrecuencia) cuyas especificaciones analógicas definen las características de un dispositivo NFC Forum. En estas especificaciones se definen por ejemplo la forma y fuerza de los campos de radiofrecuencia y el rango operativo de los dispositivos. Respecto a los aspectos digitales de la norma ISO/IEC 18092 e ISO/IEC 14443, define los elementos básicos para la comunicación.

2.1.4 Modos de funcionamiento

El estándar NFC define dos tipos de dispositivos en un diálogo, son los denominados “Iniciador” y “Destino”. La comunicación se realiza a través de un diálogo entre ambos, o incluso podría haber más de un Destino. Como sus propios nombres indican, el Iniciador es el dispositivo encargado de iniciar la comunicación y controlar el intercambio de datos mientras que el Destino responde a las peticiones del Iniciador. Estos roles son intercambiables durante la comunicación.

Existen dos **modos de establecer la comunicación** entre dos dispositivos NFC que se basa en la obtención de la energía:

- **Modo pasivo**: el dispositivo Iniciador genera el campo electromagnético y el dispositivo destino se comunica con éste modulando la señal recibida. En este modo, el dispositivo destino obtiene la energía necesaria para funcionar del campo electromagnético generado por el Iniciador.

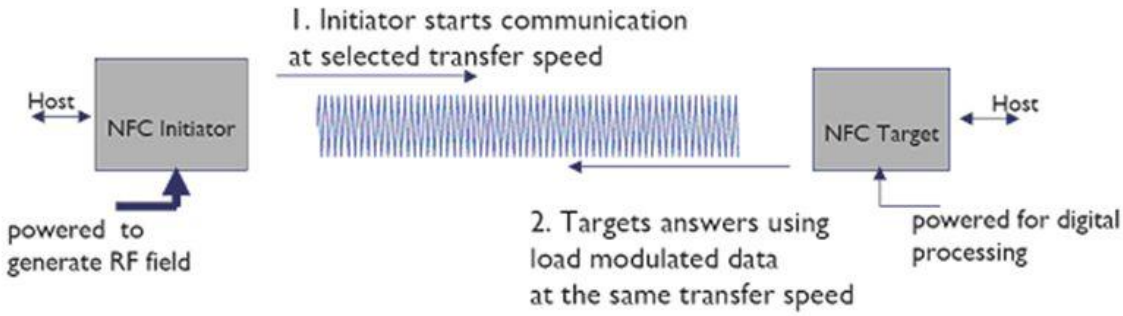


Figura 2.18. Modo de funcionamiento NFC pasivo

- **Modo activo**: tanto el dispositivo Iniciador como el destino se comunican generando su propio campo electromagnético. En este modo, ambos dispositivos requieren de una fuente de alimentación para funcionar.

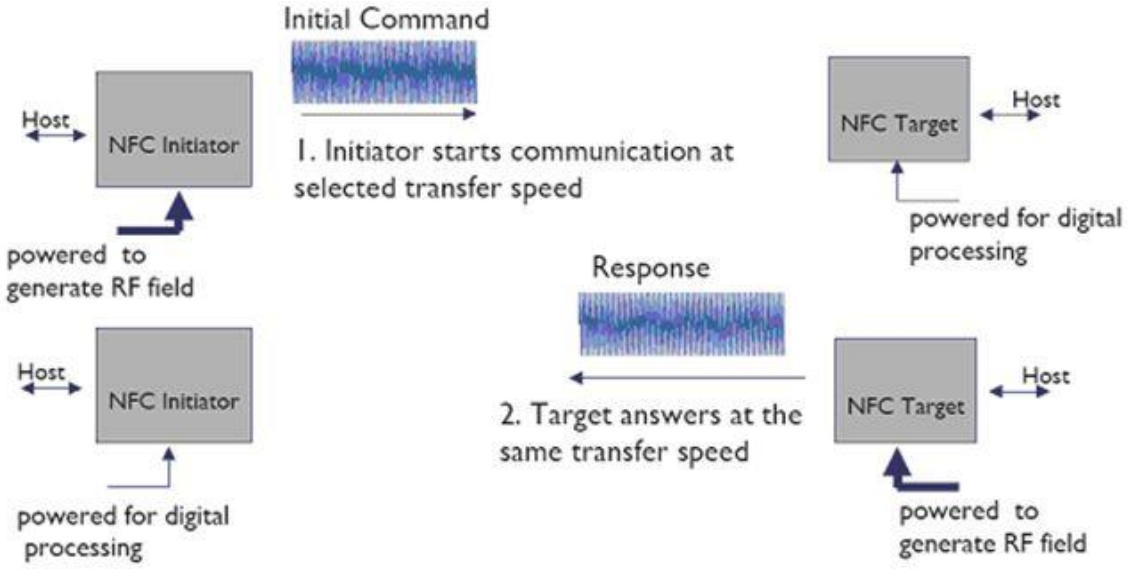


Figura 2.19. Modo de funcionamiento NFC activo

Cuando el dispositivo funciona en modo pasivo, el receptor sólo se utiliza para establecer la comunicación y confirmar la recepción de los datos. Sin embargo, en modo activo, se requiere que ambos nodos negocien el intercambio de datos. Aunque muchas aplicaciones requieren que los dispositivos involucrados sean activos, la combinación de uso activo/pasivo puede ser útil para comunicarse con elementos sin batería, como pueden ser las tarjetas sin contactos o las etiquetas RFID que no dispongan de fuente de alimentación propia.

Toda la comunicación que se produce entre dos dispositivos, ya sean de naturaleza activa o pasiva, sigue siempre una misma secuencia de operación, la **transacción NFC**, que consta de los siguientes pasos o fases:

- Descubrimiento de dispositivos NFC
- Autenticación. A bajo nivel, el protocolo NFC incluye un procedimiento para la autenticación segura y mecanismos anti-colisión para evitar la escucha del canal de comunicación.
- Negociación. Se establecen parámetros que definen las características de la comunicación como la velocidad de transmisión, el identificador del dispositivo, el tipo de aplicación, el tamaño de la transferencia y la acción solicitada.
- Transferencia de información
- Reconocimiento

La seguridad es uno de los puntos fuertes a tener en cuenta durante la transacción y para aplicaciones donde es especialmente importante, como es el caso de su uso como medio de pago, es posible utilizar cifrado AES y triple DES equiparando la seguridad a la ofrecida por las tarjetas inteligentes bancarias.

2.1.5 Etiquetas

Las etiquetas NFC son elementos pasivos que se pueden utilizar para la comunicación con cualquier dispositivo NFC (un dispositivo activo de lectura/escritura NFC).

El modo de operación que sigue una etiqueta es bastante simple: cuando un usuario con un tag (una persona con su móvil) toca un dispositivo NFC, la etiqueta toma una pequeña cantidad de energía del dispositivo para alimentar su electrónica, de esta manera la etiqueta se activa y puede transferir la información al dispositivo de lectura/escritura. Se pueden encontrar etiquetas en lugares tales como carteles, y otras áreas donde se pueda almacenar pequeñas cantidades de información como por ejemplo direcciones URL.

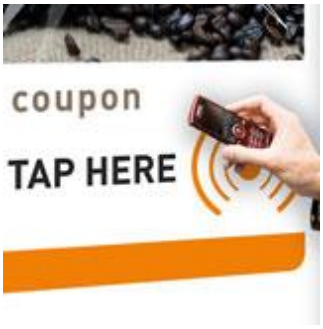


Figura 2.20. Cartel NFC

El NFC Forum presentó su primera arquitectura de la tecnología estandarizada y las normas para los dispositivos compatibles con NFC en junio de 2006. Esto incluyó el *NFC Data Exchange Format (NDEF)* y tres *Record Type Definitions (RTD)*. Se

utilizan para los carteles inteligentes, texto y aplicaciones de lectura de recursos en Internet.

Se han definido cuatro tipos básicos de etiquetas, se designan con los números del 1 al 4 tomando cada uno un formato y capacidad diferentes. Estos formatos de etiquetas NFC se basan en ISO 14443 tipo A y B (estándar internacional para tarjetas inteligentes sin contacto) y Sony FeliCa que se ajusta a la norma ISO 18092 (el modo de comunicación pasiva) [8]:

- **Tag tipo 1**: esta etiqueta se basa en el estándar ISO14443A, es de lectura y escritura (también se puede sobreescribir) y los usuarios pueden configurarla para que sea de solo lectura. La memoria es de 96 bytes que es más que suficiente para almacenar una URL de un sitio web u otra cantidad pequeña de datos, con la posibilidad de extenderse hasta los 2 kbytes. La velocidad de comunicación de esta etiqueta es de 106 kbit/s. Como resultado de su simplicidad, este tipo de variable es rentable e ideal para muchas aplicaciones NFC.

- **Tag tipo 2**: esta etiqueta tiene características comunes a las del tipo 1 ya que se basa en ISO14443A y es de lectura/escritura con posibilidad de reescribir en ella o configurarla para un uso de solo lectura. El tamaño de la memoria básica de este tipo de tag es sólo de 48 bytes, ampliable a 2 kbytes. La velocidad también es de 106 kbit/s.

- **Tag tipo 3**: en este caso varía el sistema en el que se basa la etiqueta ya que se trata del Estándar Industrial Japonés (JIS) X 6319-4, también conocido como Sony FeliCa. La capacidad de memoria es variable llegando hasta 1 Mbyte y la velocidad de comunicación de datos es de 212 kbit/s o 424 kbits/s. Debido a estas características, este tipo de tag NFC es más utilizado en aplicaciones complejas, aunque las etiquetas tienen un mayor coste.

- **Tag tipo 4**: esta etiqueta es compatible con las normas ISO14443A y B. Está configurada para ser de lectura/reescritura o sólo lectura. La capacidad de memoria es variable y su máximo valor son 32 kbytes, la velocidad de comunicación es de 424 kbit/s.

Especificaciones tags NFC			
Tipo de Tag	Norma	Memoria	Velocidad
Tipo 1	ISO14443A	96 bytes a 2 kbytes	106 kbits/s
Tipo 2	ISO14443A	48 bytes a 2 kbytes	106 kbits/s
Tipo 3	JIS X 6319-4 (Sony Felica)	1 Mbyte	212 kbits/s o 424 kbits/s
Tipo 4	ISO14443A y B	32 kbytes	424 kbits/s

Tabla 2.2. Resumen especificaciones tags NFC

A partir de las diferentes definiciones de los tipos de etiquetas NFC se deduce que el tipo de etiquetas 1 y 2 son muy diferentes de las 3 y 4 ya que tienen distinta capacidad y estructura. Por lo tanto, los usos de cada clase apenas se solaparán y será sencillo utilizar un tipo de tag para cada aplicación.

2.1.6 NFC Forum

Near Fear Communication Forum es una organización sin ánimo de lucro que promueve el uso de la tecnología inalámbrica de corto alcance "NFC" en dispositivos móviles, electrónica de consumo y ordenadores [9].

Como se ha comentado, su **misión principal** es promover el uso de NFC en la sociedad, para ello han desarrollado unas especificaciones para asegurar la interoperabilidad entre diferentes dispositivos y servicios además de ayudar y enseñar al mercado sobre la tecnología NFC.

El Forum se creó en 2004 y cuenta ya con 150 miembros entre los que se encuentran fabricantes, desarrolladores de aplicaciones e instituciones de servicios financieros entre muchos otros. Todos ellos trabajan juntos en el ecosistema NFC en sus diferentes variantes para hacer del uso de la tecnología NFC algo más fácil y cercano [10].

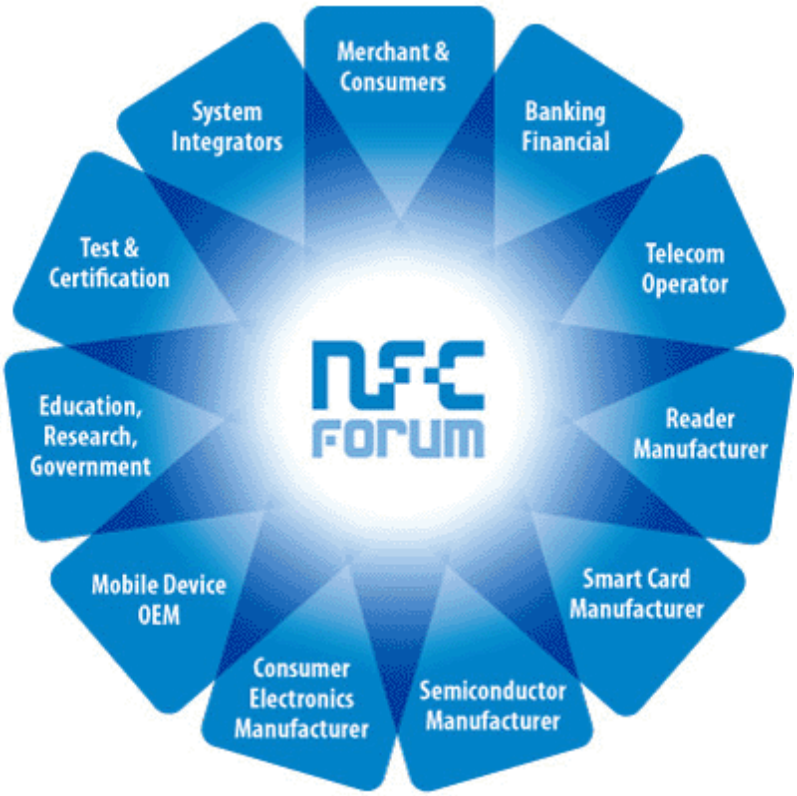


Figura 2.21. El ecosistema NFC

Los miembros del Forum cubren todas las partes del ecosistema NFC. Estas empresas comparten su experiencia en el desarrollo, aplicación y comercialización para desarrollar las mejores soluciones posibles. Todas las decisiones son aprobadas por los socios en votación para asegurar que se incluyen todos los puntos de vista con el fin de construir una base firme y flexible para el futuro crecimiento del mercado NFC. Los principales miembros que integran el NFC Forum son: *American Express, LG, Rogers, at&t, Marvell, sequent, csr, Motorola, Sony Ericsson, DNP, PayPal, Texas Instruments, dentsu, Qualcomm, Toppan Forms, Google* y *RIM*.

Los **objetivos** que persigue el NFC Forum son:

- Desarrollar estándares basados en las especificaciones de NFC que definan una arquitectura modular y parámetros de interoperabilidad para los dispositivos NFC y los protocolos.
- Fomentar el desarrollo de productos con las especificaciones NFC Forum.
- Trabajar para garantizar que los productos que demandan capacidades NFC cumplan con las especificaciones NFC Forum.
- Educar a los consumidores y empresas a nivel mundial acerca de NFC.

El NFC Forum ofrece un marco muy estable para el extenso desarrollo de aplicaciones, con soluciones interoperables y seguridad en las transacciones con NFC. El Forum ha organizado los esfuerzos de docenas de organizaciones miembros mediante la creación de Comités y Grupos de Trabajo.

En junio de 2006, sólo 18 meses después de su fundación, se perfila formalmente la arquitectura de la tecnología NFC. Hasta ahora se han dado a conocer 15 especificaciones las cuales proporcionan una “hoja de ruta” que permite a todas las partes interesadas crear nuevos y potentes productos dirigidos al consumidor. Estas especificaciones se engloban en los siguientes grupos: *Data Exchange Format (NDEF) Technical Specification, NFC Forum Tag Type Technical Specifications, Record Type Definition (RTD) Technical Specifications, Reference Application Technical Specifications, Protocol Technical Specifications, NFC Device Internal Technical Specifications* y *Certification Documents.* En el punto *2.1.8* se dará más información respecto NDEF y RTD.

2.1.7 Contactless Communication API

El API de Comunicación sin contactos (*Contactless Communication API*) está basado en la especificación JSR-257 de JavaME [11]. El desarrollo ha sido dirigido por *Nokia* con el objetivo de implementar su uso dentro de sus equipos.

La implementación de esta especificación permite el descubrimiento e intercambio de datos entre etiquetas y dispositivos NFC, etiquetas RFID y Smart Cards mediante el uso de los paquetes y librerías definidos.

El API consta de un conjunto de paquetes que permite manejar, controlar y comunicarse mediante una interfaz entre la aplicación y el módulo NFC, permitiendo así la salida o entrada de información entre dispositivos sin contactos

Paquetes JSR-257		
Paquetes Java	Interfaces	Clases
javax.microedition.contactless	TagConnection TargetListener TargetProperties TransactionListener	DiscoveryManager TargetType
javax.microedition.contactless.ndef	NDEFRecordListener NDEFTagConnection	NDEFMessage NDEFRecord NDEFRecordType
javax.microedition.contactless.rf	PlainTagConnection	
javax.microedition.contactless.sc	ISO14443Connection	
javax.microedition.contactless.visual	ImageProperties VisualTagConnection	SymbolgyManager

Tabla 2.3. Paquetes API Java Contactless Communication (JSR-257)

Cada uno de estos paquetes realiza una función específica para la comunicación sin contactos a través de sus interfaces:

- **javax.microedition.contactless:** se trata de un paquete obligatorio, es decir, para realizar cualquier tipo de comunicación sin contacto, este paquete debe estar implementado. Permite el descubrimiento de los blancos o dispositivos y clases comunes para todos ellos. Interfaces:
  - *TagConnection* es una interfaz básica para todas las etiquetas RFID y Smart Cards. Cada conexión que quiera establecerse con una tarjeta o una SmartCard debe usar esta interfaz.
  - *TargetListener* proporciona un mecanismo para que se le notifique a la aplicación cuando un objetivo ha sido descubierto mediante el hardware del dispositivo. Esta interfaz captura el evento producido en el descubrimiento, por lo tanto las aplicaciones deben implementar esta interfaz para recibir las notificaciones.
  - *TargetProperties* recopila todas las propiedades comunes a todos los objetivos compatibles con la especificación.
  - *TransactionListener* proporciona una notificación a la aplicación sobre la actividad del elemente seguro cuando se trata del modo de funcionamiento de emulación de tarjetas. Cuando surge actividad entre el elemento seguro y un lector externo a través del hardware RFID, la notificación se envía a la aplicación registrada para recibirlo. Las aplicaciones deben implementar esta interfaz para recibir las notificaciones pero no participan en estas transacciones.

- **javax.microedition.contactless.ndef:** proporciona la funcionalidad necesaria para intercambiar datos con formato NDEF con otros objetivos o dispositivos. Interfaces:

  - *NDEFRecordListener* proporciona un mecanismo de notificación a la aplicación cuando detecta registros NDEF.en los objetivos encontrados.

  - *NDEFTagConnection* define la funcionalidad básica para el intercambio de datos formato NFC Forum con tarjetas RFID y SmartCards. Los datos reales se almacenan en el objeto NDEFMessage el cual contiene la información en registros NDEF.

- **javax.microedition.contactless.rf:** este paquete proporciona una interfaz de alto nivel para acceder a los dispositivos RF sin contacto más comunes. La interfaz proporciona un nivel de acceso físico a los dispositivos:

  - *PlainTagConnection* define el mecanismo básico para comunicarse con las diferentes etiquetas RFID que contienen datos que no son de formato NFC, el formato es el definido por el proveedor de las etiquetas con sus propias características especiales y comandos de acceso.

- **javax.microedition.contactless.sc:** define el acceso a la ISO 14443-4 compatible con SmartCards sin contacto. El descubrimiento de las tarjetas se realiza de la misma manera que otros componentes de esta especificación, la comunicación se basa en comandos APDU. Interfaz:

  - *ISO14443Connection* permite la comunicación entre la aplicación y un SmartCard sin contacto mediante los comandos APDU. Estos comandos se definen en la especificación ISO7816-4 o vienen dados por el fabricante del hardware RFID. Esta interfaz no reemplaza a la APDUConnetion de la especificación JSR-177.

- **javax.microedition.contactless.visual:** proporciona los medios para la lectura almacenada en etiquetas visuales (códigos de barras) y para generar etiquetas de imágenes visuales. Interfaces:

  - *ImageProperties* permite la lectura de la información almacenada en las etiquetas visuales (códigos de barras). Esta interfaz recoge las propiedades de la etiqueta visual que son comunes a todas las simbologías y permite también gestionar estas propiedades.

  - *VisualTagConnection* permite la conexión sin contacto a una etiqueta visual objetivo. Se puede utilizar para recuperar datos de la etiqueta y para generarlos.

Para tener un concepto más visual de cómo funciona esta especificación en el interior de un dispositivo móvil se muestra el siguiente esquema:

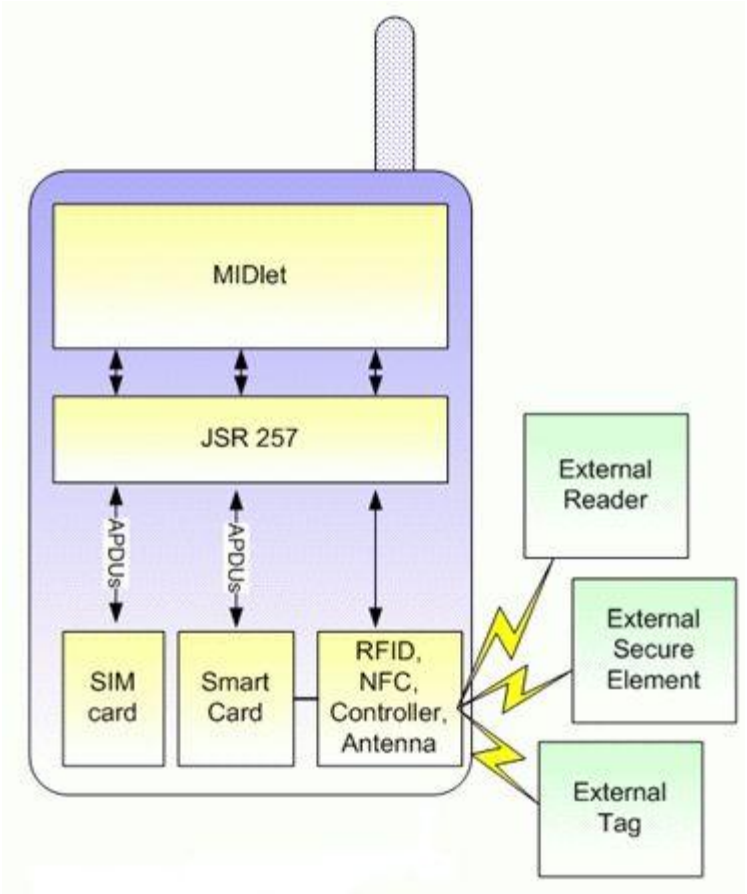


Figura 2.22. Funcionamiento interno API JSR-257 en un móvil

2.1.8 NDEF y RTD

NDEF

NDEF es un formato de datos común definido por el NFC Forum, sus siglas significan *NFC Data Exchange Format.* Se puede usar para guardar y transportar diferentes tipos de elementos, que van desde cualquier dato escrito MIME hasta documentos RTD muy pequeños como URLs.

La especificación NDEF define un formato de encapsulación de mensaje para el intercambio de datos entre dispositivos NFC o entre un dispositivo NFC y una etiqueta NFC, además de definir también las reglas para la construcción de un mensaje NDEF válido y una cadena ordenada de registros NDEF. La diferencia entre una etiqueta y un dispositivo NFC es que la primera es pasiva y necesita de un dispositivo activo para funcionar, no permite una interacción con el usuario y por sí sola no podría mostrar ninguna información. En cambio, un dispositivo NFC genera su propia energía permitiendo interacción del usuario y a través de su campo de inducción puede estimular y generar la energía para el funcionamiento de los elementos pasivos. El formato de datos de NDEF es el mismo tanto para un dispositivo

como para una etiqueta NFC, por lo que la información de NDEF es independiente del tipo de dispositivos que se estén comunicando.

NDEF es un formato binario ligero que puede encapsular una o más payloads (es la carga útil, la información que resulta útil al usuario de todo el flujo que se envía) de diferente tipo y tamaño dentro de la estructura de un solo mensaje. El payload está identificado por un tipo, una longitud y un identificador opcional. Dentro del formato NDEF se pueden enviar varios tipos de información como documentos o fragmentos XML encapsulados, datos encriptados, imágenes, cadenas de información, etc.

Existen dos conceptos relacionados entre sí que son los mensajes NDEF y los registros NDEF. Un **mensaje NDEF** está compuesto por uno o varios registros NDEF, los campos *MB* (Message Begin) y *ME* (Message End) que se muestran en la *figura 2.23*, marcan el comienzo y el final de ese mensaje, es decir, en el primer registro que componga el mensaje MB estará marcado y en el último lo estará ME; en cambio si solo hubiera un registro, estarían los dos campos marcados. Los mensajes NDEF no llevan números índices para indicar su orden, sino que está implícito en el orden en que los registros son serializados. Por ejemplo, si los registros son empaquetados por una aplicación intermedia, ésta es la responsable de asegurar que el orden de los registros sea el mismo.

Un **registro NDEF** está compuesto por una serie de campos con información sobre el envío y se dividen en 3 piezas principales: .*Type Name Format (TNF)* que indica cómo serán interpretados los bytes del registro, el *type* que define el contexto de los datos y el *payload* como se ha comentado anteriormente Los registros son de longitud variable pero todos tienen un formato común:

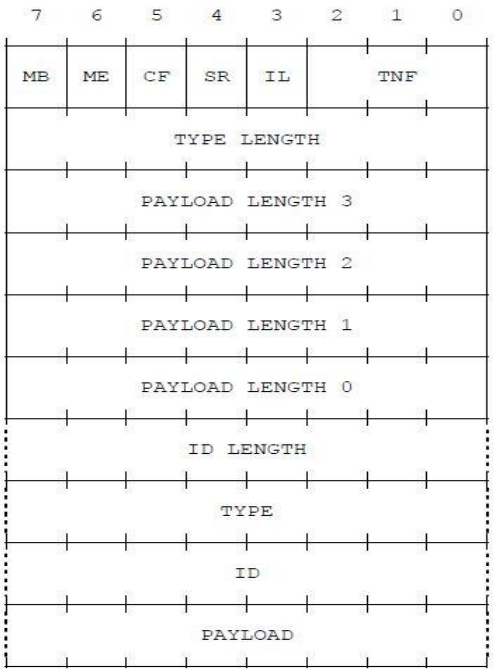


Figura 2.23. Formato de un registro NDEF

La información de los registros NDEF se presenta en nivel de octetos. El orden de transmisión es de izquierda a derecha y de arriba hacia abajo; de esta manera el bit más significativo del octeto es el bit del extremo izquierdo y para una cadena de octetos es igual, el bit más significativo es del extremo izquierdo de todo el campo de octetos y es el que se transmite primero [12].

RTD

La especificación *RTD*, por sus siglas en inglés *Record Type Definition*, provee las pautas para la especificación de los tipos de registros estándar que pueden ser incluidos en los mensajes NDEF transmitidos. En el campo de formato de los tipos de registros están contenidos los nombres del tipo de registro llamado *record type name* [13].

Cada definición de tipo de registro es identificado por su Record Type Name y pueden especificarse en distintos formatos llamados Type Name Format (TNF). Estos pueden ser:

- URIs absolutas
- Well-known, definido por NFC Forum, que a su vez incluye:
  - Texto: se trata de un registro que solo contiene datos de texto por lo que puede ser usado sin requerir mucho espacio.
  - URI: se trata de un registro contenedor compacto de URIs
  - Smart Poster: especifica la manera de incorporar datos como SMS, URIs o números de teléfono en etiquetas NFC o la manera de transportarlos entre dispositivos. El objetivo de los Smart Poster es proveer una manera simple para acceder a un servicio remoto mediante un toque. También puede contener acciones que desplieguen una aplicación en un dispositivo, por ejemplo iniciar un explorador de internet.
- Formato MIME.
- Formato definido por especificaciones externas.
- Formato vacío.

2.1.9 NFC y otras tecnologías sin contactos

Cada tipo de tecnología inalámbrica nos brinda una serie de características que la hacen apropiada para varios usos y escenarios. Sus diferentes configuraciones y modos de funcionamiento así como sus características (velocidad, alcance, tiempo de establecimiento de la comunicación, seguridades, etc.), hacen que se ajusten a las distintas necesidades.

A continuación, se muestra una gráfica comparativa del rango de alcance (también se incluyen de largo alcance) y la velocidad de transmisión de varias tecnologías inalámbricas [14]:

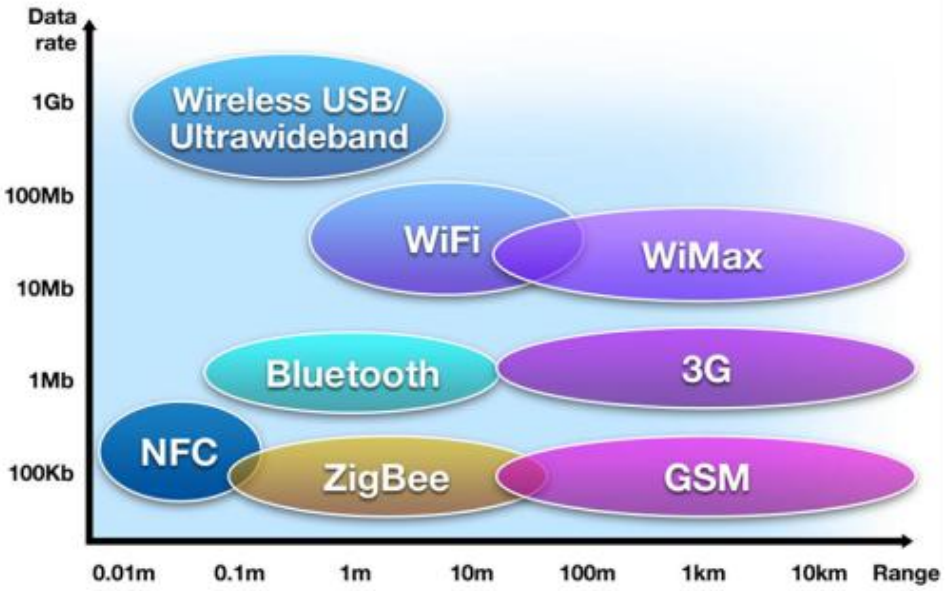


Figura 2.24. Comparación tecnologías inalámbricas

Podemos realizar una comparación más exhaustiva con algunas tecnologías de corto alcance, a continuación se nombran y se muestra una tabla comparativa con sus características básicas:

- Bluetooth: esta tecnología inalámbrica fue diseñada para reemplazar los cables entre móviles, portátiles y otros dispositivos informáticos y de comunicación en un radio de 10 metros.
- ZigBee: se trata de una tecnología inalámbrica que permite el control y la monitorización de aplicaciones industriales y de domótica en un rango de 100 metros.
- IrDA: es una tecnología inalámbrica de muy corto rango que transmite y recibe datos a través de rayos infrarrojos. Se utiliza en ordenadores y teléfonos móviles.
- RFID: es una tecnología inalámbrica de la que ya hemos hablado anteriormente. Proporciona un método de identificación automática basándose en el almacenamiento y recuperación de datos remotamente utilizando etiquetas RFID.

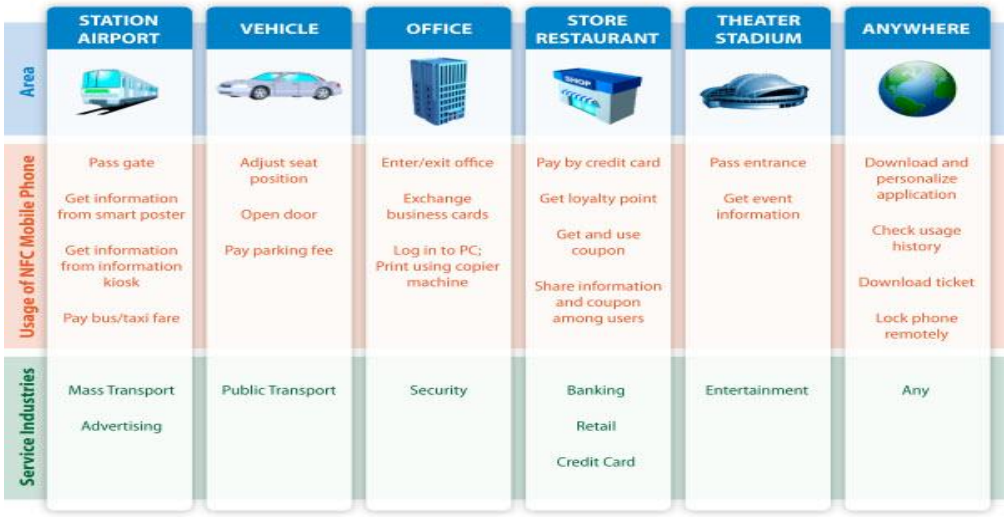
Comparativa tecnologías inalámbricas corto alcance					
	NFC	Bluetooth	ZigBee	IrDA	RFID
Tiempo de establecimiento	< 0.1s	6s	30ms	0.5s	< 0.1s
Velocidad de transmisión	424 Kbps	2.1 Mbps (versión 3.0: 24 Mbps)	250 Kbps	2400 bps a 4 Mbps	424 Kbps
Alcance	10 cm	30 m	70 m	1 m	3 m
Seguridad	Bastante alta	Alta con PIN	Alta	Visión directa	Buena
Facilidad de uso (conexión)	Muy fácil (un toque)	Configuración, emparejamiento	Sin configuración	Sin configuración	Sin configuración
Usos	Acceso edificios, pagos, obtener información, establecer conexiones, etc	Red para intercambio de datos variados (audio, imágenes, documentos)	Domótica, control industrial, monitorización pacientes	Intercambio de datos y control remoto de dispositivos	Seguimiento, identificación de productos, etc

Tabla 2.4. Comparativa tecnologías inalámbricas de corto alcance

NFC complementa muchas de las tecnologías inalámbricas más populares mediante la utilización de elementos claves existentes en los estándares de tarjetas sin contacto. Puede ser compatible con la infraestructura existente de tarjetas sin contacto y permite al consumidor utilizar un único dispositivo a través de varios sistemas. Resumiendo, con la tecnología NFC podemos conseguir conexiones sencillas, rápidas transacciones y una compartición de datos simple.

2.1.10 Casos reales de aplicación NFC

Como se ha comentado en puntos anteriores, la tecnología NFC tiene múltiples aplicaciones en la vida cotidiana como son el control de acceso físico a recintos, pagos mediante el móvil, uso de llaves electrónicas, documentación electrónica, tarjetas de transporte público, entradas para eventos, billetes de avión y transportes o el uso que se le da a NFC en este proyecto que es **simplificar y facilitar el establecimiento de la conexión a otras tecnologías como Bluetooth o WIFI**. Para entender todo lo que puede dar de sí esta tecnología, en el NFC Forum muestran un ejemplo de uso cotidiano de NFC muy interesante [15]:



7:30 – Eric coge el tren para ir a la oficina, con su teléfono habilitado para NFC toca un lector y fácilmente se abre el torniquete.

7:32 – Ve un poster que anuncia un concierto gratis esa noche. Con su teléfono toca el N-Mark** del poster y se transfiere la información detallada a su teléfono. Reserva asientos para el concierto con su móvil usando las comunicaciones móviles (SMS, Internet, etc.) y los tickets son enviados a su teléfono. Envía un mensaje de texto a su mujer para invitarla al concierto y a cenar.

8:15 – Cuando Eric llega a la oficina toca la puerta de la oficina con el móvil NFC para desbloquear el mecanismo de seguridad.

Mediodía – Paga la comida con una de las tarjetas almacenadas en su móvil.

13:00 – Después del almuerzo Eric visita la oficina de su nuevo socio para una reunión. Los asistentes a la reunión intercambian sus tarjetas de visita electrónicas que están almacenadas en sus móviles NFC, mediante un toque entre los teléfonos.

18:00 – Eric queda con su esposa y van a la sala de conciertos. En la entrada, el toca con su teléfono el torniquete, las entradas se confirman y son admitidos.

20:00 – Después del concierto, van a un centro comercial donde hacen algunas compras y cenan, utilizan sus teléfonos NFC para pagar todo.

22:00 – Cuando llegan a casa, Eric se da cuenta de que se ha dejado el móvil en el tren. Llama inmediatamente a su operadora y cancela todos los servicios activos NFC que tenía. Si recupera el teléfono más tarde, el podrá reactivar todos los servicios cancelados.

**N-Mark: es el símbolo universal de NFC que permite saber donde se ofrecen los servicios o si un dispositivo está habilitado .

Figura 2.25. Usos cotidianos de NFC

Aparte de las aplicaciones genéricas de NFC, existen muchas empresas que llevan a cabo proyectos comerciales utilizando NFC. Hoy en día, una de las aplicaciones más extendidas de la NFC es como medio de pago, Japón es uno de los países más avanzados en este sentido ya que el uso de la tecnología en los teléfonos móviles está muy extendido.

Algo curioso de la utilización de NFC en Japón es que existe una fragmentación del sistema respecto con el resto del mundo, lo que podría llevar a una confusión a los usuarios con este tipo de servicios. El estándar utilizado en Japón está basado en las tarjetas Felica de Sony y es incompatible con lo utilizado en el resto del mundo (NFC tipo A y B). Todo esto ha forzado a las tres grandes empresas de telecomunicaciones del país (NTT DoCoMo, KDDI y Softbank) a formar el denominado *Japan Mobile NFC Consortium*, esta asociación intentará hacer compatible los sistemas nacionales con los del resto del mundo para que así cualquier usuario pueda utilizar los beneficios de NFC en cualquier lugar.

En la actualidad no hay una gran variedad de terminales que cuenten con la tecnología NFC nativa, esta es una de las razones por las que NFC no ha terminado de despegar. Existen varias soluciones para habilitar NFC en un móvil sin que venga integrado de serie como por ejemplo unas pegatinas que se adhieren al teléfono dotándole de capacidades NFC y que se comunica con este mediante bluetooth (*Twinlinx MyMax NFC*). Aunque la solución que podría tener mejor acogida es la integración de NFC en las tarjetas SIM, esto conlleva la utilización de una antena muy pequeña permitiendo comunicaciones de 4 cm pero que son más que suficiente para realizar pagos u otras transacciones. Algunos fabricantes que tienen dispositivos o han dejado ver sus intenciones respecto a NFC son Apple (aunque finalmente no ha incluido NFC en el lanzamiento del Iphone 4S), Google (cuyo modelo de móvil Nexus One si lo trae integrado) y por supuesto Nokia que desde el comienzo ha tenido móviles compatibles como el Nokia 6131 NFC y que ahora continúa con varios modelos que lo integran.

Cada vez se ven más proyectos que hacen uso de NFC, ya sean pruebas piloto para ir introduciendo la tecnología poco a poco o aplicaciones totalmente en funcionamiento. Para la realización de estos se unen empresas de ámbitos diferentes como son bancos tradicionales y sistemas de pago online, empresas del sector tecnológico y telecomunicaciones, empresas de transporte, ayuntamientos de ciudades, etc. A continuación se listan varios proyectos que hacen uso de NFC:

- *Mobile Shoping Sitges*: se realizó esta prueba piloto en el último semestre de 2010 que consistía en que 1500 clientes efectuaran micropagos en 500 establecimientos mediante móviles dotados con NFC. La mayoría de los usuarios declararon que era un sistema rápido, fiable y sencillo. En esta prueba participaron La Caixa, Visa, Telefónica, Samsung y el ayuntamiento de Sitges.

- Renfe junto con Vodafone pretenden integrar el sistema de pago en tarjetas SIM para comprar billetes. De momento solo es para que los

trabajadores de Renfe accedan a sus puestos de trabajo mediante la SIM, se han implantado 300 máquinas lectoras para ello.

- Paypal ha actualizado su aplicación para Android a finales de 2011 para que tenga soporte para pagos NFC.
- El ayuntamiento de Cáceres provee de información sobre la ciudad a los turistas mediante NFC. Para ello, debido a la escasez de terminales compatibles, los presta a los turistas para que puedan tener acceso a este servicio. Los turistas acercarán el teléfono a unos receptores preparados y podrán obtener información sobre los monumentos, entradas para museos, descuentos especiales, etc.
- *Distrito NFC* es un proyecto de Telefónica en el que comenzará a utilizar NFC dentro de su área empresarial, el distrito C, a las afueras de Madrid. Los empleados podrán realizar tareas cotidianas dentro de la ciudad empresarial como los pagos de las comidas o acceder a los tornos.
- *Google Wallet* es un proyecto de Google que de momento está implantado en Nueva York y San Francisco para realizar compras con el móvil. Es compatible con cualquier dispositivo NFC y consiste en una asociación de la tarjeta Master Card con el chip NFC. El sistema es totalmente seguro ya que para cualquier compra pide el número PIN.
- Samsung y Visa han llegado a un acuerdo para que durante los Juegos Olímpicos de Londres 2012 se utilice NFC lo máximo posible. Visa ha comenzado a repartir tarjetas SIM para poder realizar compras en los comercios y Samsung aportará el teléfono para realizar los pagos.

2.2 Bluetooth

2.2.1 Introducción y descripción

Bluetooth es una tecnología de corto alcance simple, segura y que podemos encontrar en cualquier lugar. Las integraciones de la tecnología en dispositivos van desde teléfonos móviles, ordenadores y productos de entretenimiento para el hogar hasta dispositivos médicos. Su objetivo principal es poder llegar a sustituir los cables que conectan los dispositivos pero sin perder los altos niveles de seguridad.

Figura 2.26. Logo Oficial Bluetooth

Su definición técnica es que se trata de una especificación industrial para *Redes Inalámbricas de Área Personal (WPAN)* 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. La transmisión de datos y voz se puede realizar simultáneamente por lo que se pueden llegar a ver soluciones Bluetooth muy interesantes como auriculares manos libres o capacidades de impresión y fax [16].

Bluetooth se ha consolidado en el mercado debido a una serie de características positivas como que no tiene un coste muy elevado, se puede integrar en un gran número de dispositivos, elimina cables y conectores, ofrece la posibilidad de crear pequeñas redes personales y permite la sincronización entre equipos. Por todo esto, es una tecnología universalmente utilizada y presente en nuestro día a día en las tareas más cotidianas.

2.2.2 Historia y evolución

En 1994, la empresa sueca *Ericsson* inició un estudio para investigar la viabilidad de una interfaz vía radio, de bajo coste y consumo, para la interconexión entre teléfonos móviles y otros accesorios con el objetivo de eliminar cables entre aparatos. El estudio partía de un largo proyecto que investigaba sobre unos multicomunicadores conectados a una red móvil, hasta que se llegó a un enlace de radio de corto alcance llamado MC link. Con el avance del proyecto quedó claro que este tipo de enlace podía ser utilizado en un gran número de aplicaciones, pues poseía como ventaja principal el hecho de basarse en un chip de radio relativamente económico.

A principios de 1997, se despertó el interés por el proyecto MC link en otros fabricantes de equipos portátiles y, en febrero de 1998, se formó un grupo llamado **Bluetooth Special Interest Group (Bluetooth SIG)** con más de 200 compañías. La

idea era lograr un conjunto adecuado de áreas de negocio, ya que se hallaban en el grupo dos líderes del mercado de las telecomunicaciones, dos del mercado de los PCs portátiles y un líder de la fabricación de chips. Este grupo estaba formado por compañías como *Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia* y *Toshiba* y su objetivo era desarrollar las especificaciones para **Bluetooth 1.0**, que se publicaron en julio de 1999.

A partir de la primera publicación de Bluetooth 1.0 se han ido creando nuevas versiones del protocolo hasta la 4.0, aunque actualmente en el mercado conviven las versiones 2.0 y 3.0 [17].

Versiones Bluetooth		
Versión Estándar Bluetooth	Fecha de la versión	Características claves
1.0	Julio 1999	Versión borrador del estándar Bluetooth
1.0a	Julio 1999	Publicada primera versión del estándar Bluetooth
1.0b	Diciembre 1999	Pequeñas actualizaciones para solventar problemas y asuntos menores
1.0b + CE	Noviembre 2000	Se añade fe de erratas para publicar la versión 1.0 del estándar Bluetooth
1.1	Febrero 2001	Primera versión utilizable. Fue usada por el IEE para su estándar IEEE 802.15.1 - 2002
1.2	Noviembre 2003	Esta versión del estándar añadió nuevas facilidades incluyendo saltos de frecuencia y eSCO para mejorar el rendimiento de la voz. Fue publicada por el IEEE como IEEE 802.15.1 - 2005. Fue la última versión publicada por el IEEE.
2.0 + EDR	Noviembre 2004	Esta versión añade la mejora en la velocidad de los datos (EDR) para aumentar el rendimiento a 3.0 Mbps
2.1	Julio 2007	En esta versión se añade el emparejamiento simple seguro para mejorar la seguridad
3.0 + HS	Abril 2009	Bluetooth 3 añade IEEE 802.11 como canal de alta velocidad para incrementar la tasa de envío a 10+ Mbps
4.0	Diciembre 2009	Se actualizó el estándar Bluetooth para incluir Bluetooth Low Energy antes conocido como Wibree

Tabla 2.5. Evolución de las versiones del estándar Bluetooth

La versión 1.2, a diferencia de la 1.1, provee una solución inalámbrica complementaria para coexistir Bluetooth y WiFi en el espectro de los 2.4 GHz, sin que se produzcan interferencias entre ellos. Usa la técnica *Adaptive Frequency Hopping (AFH)*, que ejecuta una transmisión más eficiente y un cifrado más seguro. Para

mejorar las experiencias de los usuarios, la 1.2 ofrece una calidad de voz (Voice *Quality – Enhanced Voice Processing*) con menor ruido ambiente, y provee una configuración más rápida de la comunicación con los otros dispositivos bluetooth dentro del rango del alcance, como pueden ser PDAs, HIDs (Human Interface Devices), ordenadores portátiles, ordenadores de escritorio, headsets o auriculares, impresoras y móviles.

La versión 2.0, creada para ser una especificación separada, incorpora la técnica "Enhanced Data Rate" (EDR) que le permite mejorar las velocidades de transmisión hasta 3Mbps a la vez que intenta solucionar algunos errores de la especificación 1.2.

La versión 2.1, simplifica los pasos para crear la conexión entre dispositivos, además el consumo de potencia es 5 veces menor. En esta versión la tasa llega a los 3 Mbps, en cambio el gran aporte de las versiones 3.0 y 4.0 es que pueden llegar a 24 Mbps.

El nombre de la tecnología procede del rey danés y noruego Harald Blåtand, cuyo nombre traducido al inglés sería *Harold Bluetooth* (Diente Azul, aunque en lengua danesa significa 'de tez oscura'), conocido por buen comunicador y por unificar las tribus noruegas, suecas y danesas. De la misma manera, Bluetooth intenta unir diferentes tecnologías como las de los ordenadores, teléfonos móviles y el resto de periféricos. El símbolo de Bluetooth es la unión de las runas nórdicas análogas a las letras B y H: ᚼ(Hagall) y ᛒ(Berkanan).

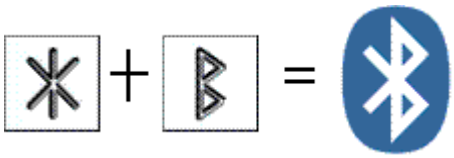


Figura 2.27. Origen del logo Bluetooth

2.2.3 Características y especificaciones

La tecnología Bluetooth tiene una serie de **características genéricas** que la definen y hacen muy atractivo su uso:

- Se utiliza para comunicaciones inalámbricas de **datos y voz**. Esta es su característica general ya que es la que compone la definición de Bluetooth.
- Es una tecnología de **corto alcance, bajo consumo y bajo coste.** Para lograr alcanzar el objetivo de bajo consumo y bajo coste, se ideó una solución que se puede implementar en un solo chip utilizando circuitos CMOS. De esta manera, se logró crear una solución de 9x9mm y que consume aproximadamente un 97% menos energía que un teléfono móvil común.

- Soporta conexiones **punto a punto y multipunto**. Se puede establecer conexiones con un solo dispositivo o con varios.
- Incorpora mecanismos de **seguridad**. Utiliza el mecanismo de emparejamiento para el establecimiento de la comunicación y la transmisión de datos.
- Es **omnidireccional** por lo que no necesita visión directa para transmitir y es capaz de atravesar obstáculos sólidos no metálicos.
- Es una tecnología regulada por organismos mundiales y **universalmente utilizada**.

Además de las características generales, Bluetooth tiene una serie de **especificaciones técnicas** que se muestran a continuación:

Especificaciones básicas Bluetooth	
Alcance	10 m (puede llegar a alcanzar 100 m)
Tiempo de conexión	6 s
Frecuencia	2.4 GHz
Velocidad de transmisión	1 Mbps (1.0) - 24 Mbps (4.0)
Estándares	IEEE 802.15.X

Tabla 2.6. Especificaciones técnicas básicas Bluetooth

La especificación de Bluetooth define un canal de comunicación de máximo 720Kbps con rango óptimo de 10m (opcionalmente 100m utilizando amplificadores). La frecuencia de radio con la que trabaja está en el rango de 2.4 a 2.48 GHz con amplio espectro y saltos de frecuencia con posibilidad de transmitir en full duplex con un máximo de 1600 saltos/seg. Los saltos de frecuencia se dan entre un total de 79 frecuencias con intervalos de 1Mhz; esto permite brindar seguridad y robustez. La potencia de salida para transmitir a una distancia máxima de 10m es de 0dBM (1 mW), mientras que la versión de largo alcance transmite entre -30 y 20dBM (100 mW).

Hay tres canales de datos síncronos (voz), o un canal de datos síncrono y uno asíncrono que pueden ser soportados en un solo canal. Cada canal de voz puede soportar una tasa de transferencia de 64 Kb/s en cada sentido, la cual es suficientemente adecuada para la transmisión de voz. Un canal asíncrono puede transmitir como mucho 721 Kb/s en una dirección y 56 Kb/s en la dirección opuesta, sin embargo, para una conexión asíncrona es posible soportar 432,6 Kb/s en ambas direcciones si el enlace es simétrico.

2.2.4 Arquitectura y funcionamiento Bluetooth

La especificación de Bluetooth pretende que todas las aplicaciones sean capaces de operar entre sí. Para conseguir esta interoperabilidad, las aplicaciones en dispositivos remotos deben ejecutarse sobre una pila de protocolos idénticos.

Para comunicarse con otros dispositivos Bluetooth, se requiere un hardware específico para Bluetooth, que incluye un **módulo de banda base**, así como otro **módulo de radio** y una **antena**. Además, deberá haber un software encargado de controlar la conexión entre dos dispositivos Bluetooth; este software (**Link Manager**) por lo general correrá en un microprocesador dedicado. Los Link Managers de diferentes dispositivos Bluetooth se comunicarán mediante el protocolo **LMP (Link Manager Protocol)**. Además, habrá otros módulos de software, que constituirán la pila de protocolos, y garantizarán la interoperabilidad entre aplicaciones alojadas en diferentes dispositivos Bluetooth.

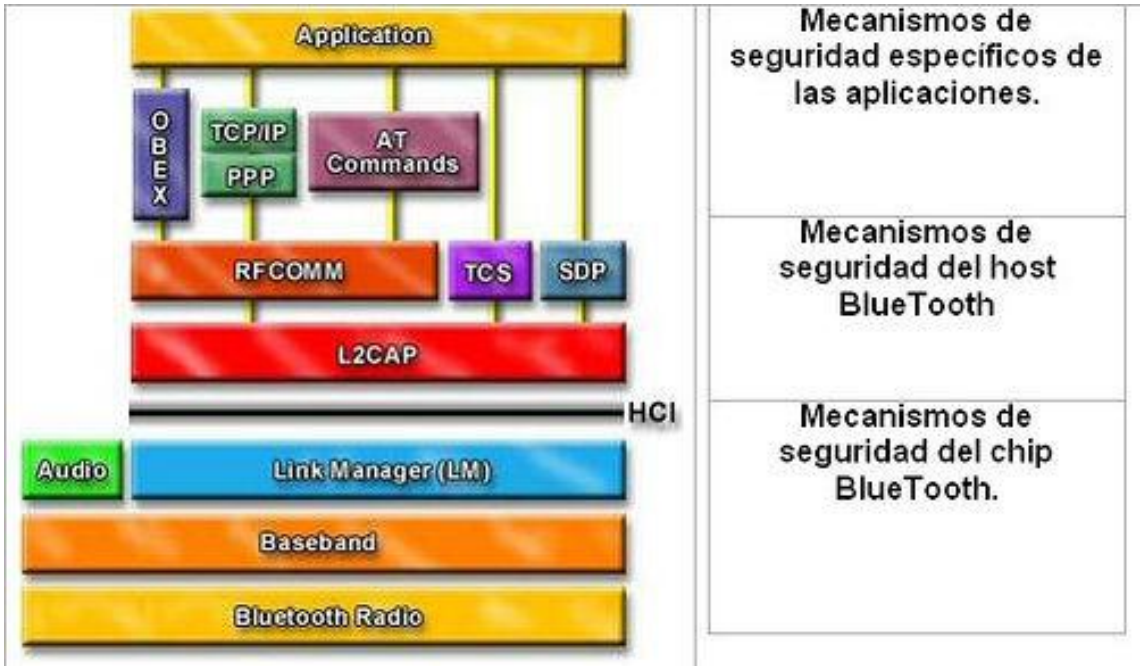


Figura 2.28. Pila de protocolos Bluetooth y hardware

Redes Bluetooth

Antes de comenzar a profundizar tanto en los componentes hardware como en los protocolos, hay que definir varios conceptos sobre la nomenclatura que adquieren los elementos en una comunicación Bluetooth en función de sus roles: la unidad básica de un sistema Bluetooth es una **piconet,** que consta de un **nodo maestro** y hasta siete **nodos esclavos** activos a una distancia de 10 metros.

En una misma sala grande, pueden encontrarse varias piconets y se pueden conectar mediante un **nodo puente**, como se muestra en la siguiente figura. Un conjunto de piconets interconectadas se denomina **scatternet.**

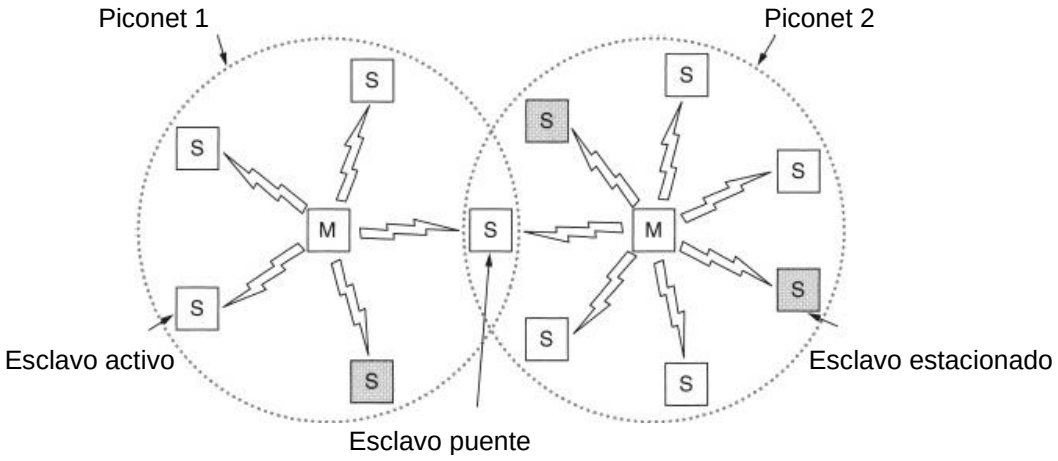


Figura 2.29. Dos piconets conectados formando un scatternet

Además de los siete nodos esclavos **activos** de una piconet, puede haber hasta 255 **nodos estacionados** (park) en la red. Éstos son dispositivos que el nodo maestro ha cambiado a un estado de bajo consumo de energía para reducir el desgaste innecesario de sus pilas. Lo único que un dispositivo en estado estacionado puede hacer es responder a una señal de activación por parte del maestro. También hay dos estados intermedios de ahorro de energía, **hold** (funciona con un contador interno cuyo valor previamente se ha acordado entre maestro y esclavo y vuelve a la actividad automáticamente) y **sniff** (escucha a la piconet a intervalos reducidos, este intervalo es programable y depende de la aplicación) [18].

Definición de un paquete

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 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, y chequeo de errores de campos de cabecera. Finalmente, el paquete que contiene la información, tiene una longitud de 0 a 2745 bits. En cualquier caso, cada paquete que se intercambia en el canal está precedido por el código de acceso.

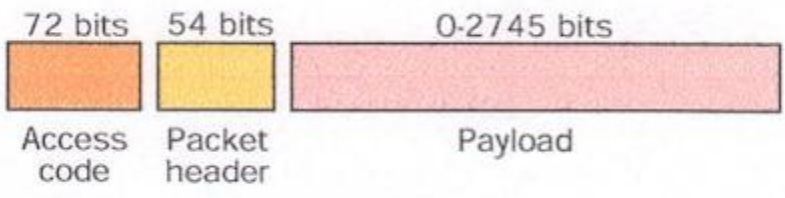


Figura 2.30. Paquete Bluetooth

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.

Arquitectura de Hardware

Un dispositivo Bluetooth se divide en dos módulos hardware principales:

- **Módulo de radio**, se encarga de modular y transmitir la señal (traslada los bits de maestro a esclavo o viceversa). La interfaz de radio soporta un gran número de canales y niveles de frecuencia diferentes así como usa fiables modos de modulación:

  - Como se ha comentado en las especificaciones técnicas, Bluetooth funciona en la banda ISM de 2,4 GHz y emplea técnicas de salto de frecuencia. Una transmisión no permanece mucho tiempo en la misma frecuencia y si se encuentra alguna interferencia, los datos serán reenviados más tarde cuando la señal esté en un canal diferente. La velocidad de saltos es de 1600 por segundo y esto permite operar en un mayor rango dinámico.

  - Los canales Bluetooth tienen una separación de 1 MHz, desde 2402 MHz hasta 2480 MHZ. En algunos países no se puede utilizar la banda completa, en Francia, Japón y España la secuencia de salto está limitada a 23 frecuencias. La frecuencia central de transmisión inicial debe estar dentro de ± 75 KHz de la frecuencia central del receptor. Para llevar a cabo una comunicación efectiva, cada dispositivo inmerso en la comunicación tiene su propio identificador, una dirección de 48 bits.

  - Se definen dos modos de modulación. Un modo obligatorio, llamado modo de transferencia básica, que usa una modulación de frecuencia binaria para reducir al mínimo la complejidad del transmisor/receptor. Y un modo opcional, llamado de transferencia de datos mejorada, que usa modulación PSK y cuenta con dos variantes: π/4-DQPSK y 8DPSK.

  - El nivel de potencia del dispositivo Bluetooth es bastante bajo, aunque hay tres clases diferentes, el valor depende del uso previsto y del rango requerido.

Potencias Bluetooth			
Clase	Máxima Potencia (dBm)	Alcance	Control de potencia
Clase1	20	Largo alcance	Obligatorio
Clase 2	4	10 m	Opcional
Clase 3	0	10 cm	Opcional

Tabla 2.7. Niveles de potencias Bluetooth

- **Módulo Baseband:** esta capa es lo más parecido a una subcapa MAC. Se convierte el flujo de bits puros en tramas y define algunos formatos clave.

  En la forma más sencilla, el maestro de cada piconet define una serie de ranuras de tiempo de 625 μseg y las transmisiones del maestro empiezan en las ranuras pares, y las de los esclavos, en las ranuras impares. Ésta es la tradicional multiplexión por división de tiempo, en la cual el maestro acapara la mitad de las ranuras y los esclavos comparten la otra mitad. Las tramas pueden tener 1, 3 o 5 ranuras de longitud.

  La sincronización de saltos de frecuencia permite un tiempo de asentamiento de 250-260 μseg por salto para que los circuitos de radio se estabilicen. Es posible un asentamiento más rápido, pero a un mayor coste. Para una trama de una sola ranura, después del asentamiento, se desechan 366 de los 625 bits. De éstos, 126 se utilizan para un código de acceso y el encabezado, y 240 para los datos. Cuando se enlazan cinco ranuras, sólo se necesita un periodo de asentamiento y se utiliza uno ligeramente más corto, de tal manera que de los 5×625=3125 bits de las cinco ranuras de tiempo, 2781 se encuentran disponibles para la capa de banda base. Así, las tramas más grandes son mucho más eficientes que las de una sola ranura.

- Cada trama se transmite por un canal lógico llamado **enlace** que está entre el maestro y un esclavo. Hay dos tipos de enlaces: *ACL* y *SCO*:

  - ACL (Asíncrono no Orientado a la Conexión), se utiliza para datos conmutados en paquetes disponibles a intervalos irregulares. Estos datos provienen de la capa L2CAP en el nodo emisor y se entregan en la capa L2CAP en el nodo receptor. El tráfico ACL se entrega sobre la base de mejor esfuerzo, no hay garantías. Las tramas se pueden perder y tienen que retransmitirse. Un esclavo puede tener sólo un enlace ACL con su maestro.

  - SCO (Síncrono Orientado a la Conexión) es para datos en tiempo real, como ocurre en las conexiones telefónicas. A este tipo de canal se le asigna una ranura fija en cada dirección. Por la importancia del tiempo en los enlaces SCO, las tramas que se envían a través de ellos nunca se retransmiten. En vez de ello, se puede utilizar la corrección de errores hacia delante (o corrección de errores sin canal de retorno) para proporcionar una confiabilidad alta. Un esclavo puede establecer hasta tres enlaces SCO con su maestro. Cada enlace de este tipo puede transmitir un canal de audio PCM de 64,000 bps.

Pila de protocolos Bluetooth

La pila de protocolos se compone tanto de protocolos específicos de Bluetooth, como LM y L2CAP, como de protocolos no específicos de Bluetooth como son OBEX (Objects Exchange Protocol), UDP (User Datagram Protocol), TCP, IP, etc. El objetivo principal a la hora de diseñar la torre de protocolos ha sido maximizar el número de protocolos existentes para que se puedan reutilizar en las capas más altas para diferentes propósitos. La especificación es abierta, lo que permite el desarrollo de nuevos protocolos de aplicación en las capas superiores, lo cual se traduce en el desarrollo de una gran variedades de servicios por parte de las casas fabricantes.

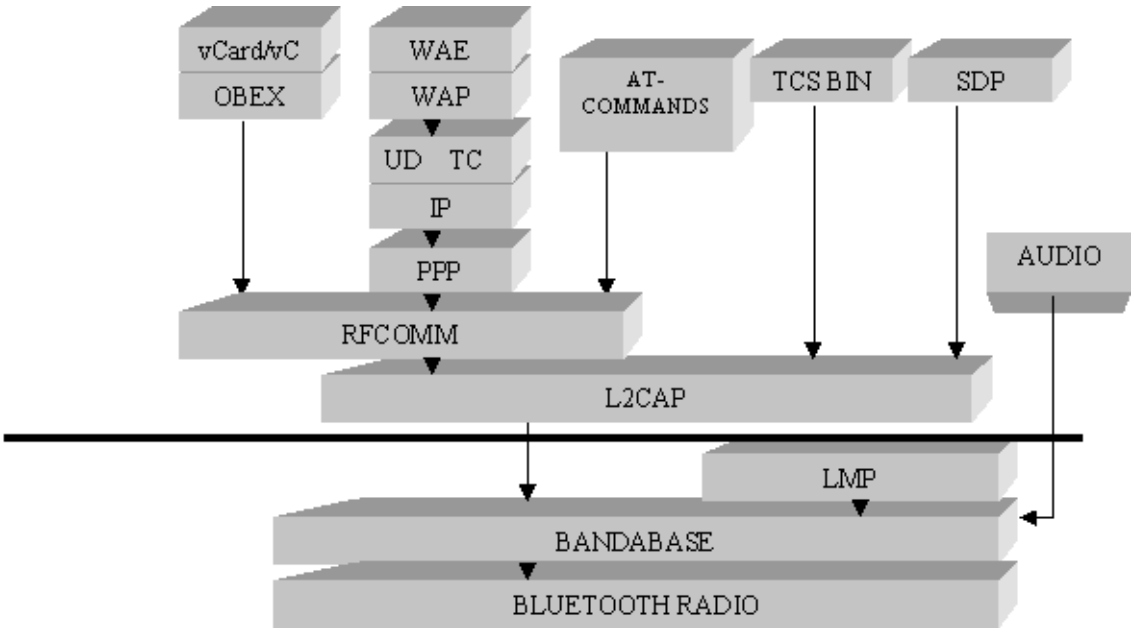


Figura 2.31. Pila de protocolos Bluetooth detallada

Cada aplicación puede operar bajo una estructura de protocolos definida por cada columna en la *figura 2.31*, o por un conjunto de ellas. Algunas columnas son usadas solo como soporte de la aplicación principal, como lo son el SDP (Service Discovery Protocol) y el TCS Binary (Telephony Control Specification).

Los protocolos pueden ser divididos en cuatro capas:

1. Protocolos Bluetooth Centrales (Bluetooth Core Protocols): BaseBand, LMP, L2CAP y SDP.

2. Protocolos de Reemplazo de Cable (Cable Replacement Protocols): RFCOMM.

3. Protocolos de control de Telefonía (Telephony Control Protocols): TCS Binary y AT-Commands.

4. Protocolos Adaptados (Adapted Protocols): PPP, UDP/TCP/IP, OBEX, WAP, vCard, vCal, IrMC y WAE.

El Grupo Bluetooth SIG, ha desarrollado los protocolos de la primera capa, los cuales son usados por la mayoría de los dispositivos Bluetooth. Por otra parte, el RFCOMM y el TCS Binary fueron desarrollados por el SIG, basándose en las especificaciones ETSI-TS 07.10 y la ITU-T Q.931, respectivamente.

Las capas de Reemplazo de Cable, Control de Telefonía, y de Protocolos adaptados conforman los llamados protocolos orientados a la aplicación. Dichos protocolos son abiertos, y permiten la inclusión de nuevos, por ejemplo HTTP o FTP, lo que hace al estándar muy flexible.

- **Link Manager y Link Manager Protocol**:

  El Link Manager es el sistema que consigue establecer la conexión entre dispositivos. Se encarga del establecimiento, la autentificación y la configuración del enlace. Localiza a otros gestores y se comunica con ellos gracias al protocolo de gestión del enlace LMP. Para poder realizar su función de proveedor de servicio, el LM utiliza los servicios incluidos en el controlador de enlace (LC -Link Controller-).

  El Link Manager Protocol básicamente consiste en un número de *PDUs (Protocol Data Units)* que son enviadas de un dispositivo a otro.

  A continuación se enuncian los servicios soportados:

  - Transmisión y recepción de datos
  - Petición de nombre: el LC tiene un eficiente método para solicitar y reportar el ID de un dispositivo con una longitud máxima de 16 caracteres.
  - Petición de las direcciones de enlace.
  - Establecimiento de la conexión
  - Autenticación
  - Negociación del modo de enlace y establecimiento, por ejemplo, modo datos o modo voz/datos. Esto puede cambiarse durante la conexión.

- **Protocolo de Adaptación y de Control de Enlace a nivel Lógico (L2CAP)**:

  El protocolo L2CAP (Logical Link Control and Adaptation Protocol) proporciona servicios de datos tanto orientados a conexión como no orientados a conexión a los protocolos de las capas superiores, junto con facilidades de multiplexación y de segmentacion y reensamblaje. L2CAP permite que los protocolos de capas superiores puedan transmitir y recibir paquetes de datos L2CAP de hasta 64 kilobytes de longitud.

Se basa en el concepto de canales (un canal es una conexión lógica que se sitúa sobre la conexión de banda base). Cada canal se asocia a un único protocolo y cada paquete L2CAP que se recibe en un canal se redirige al protocolo superior correspondiente. Varios canales pueden operar sobre la misma conexión de banda base, pero un canal no puede tener asociados más de un protocolo de alto nivel [19].

- **RFCOMM (Radio Frecuency Communication)**:

  El protocolo RFCOMM proporciona emulación de puertos serie a través del protocolo L2CAP. Este protocolo se basa en el estándar de la ETSI denominado TS 07.10. RFCOMM es un protocolo de transporte sencillo, con soporte para hasta 9 puertos serie RS-232 (EIATIA-232-E). El protocolo RFCOMM permite hasta 60 conexiones simultáneas (canales RFCOMM) entre dos dispositivos Bluetooth.

  Para los propósitos de RFCOMM, un camino de comunicación involucra siempre a dos aplicaciones que se ejecutan en dos dispositivos distintos (los extremos de la comunicación). Entre ellos existe un segmento que los comunica. RFCOMM pretende cubrir aquellas aplicaciones que utilizan los puertos serie de las máquinas donde se ejecutan. El segmento de comunicación es un enlace Bluetooth desde un dispositivo al otro (conexión directa).

  RFCOMM trata únicamente con la conexión de dispositivos directamente, y también con conexiones entre el dispositivo y el modem para realizar conexiones de red. Puede soportar otras configuraciones, tales como módulos que se comunican vía Bluetooth por un lado y que proporcionan una interfaz de red cableada por el otro [20].

- **OBEX (Object Exchange)**:

  OBEX es un protocolo muy utilizado para transferencias de ficheros sencillas entre dispositivos móviles. Su uso más importante se produce en comunicaciones por infrarrojos, donde se utiliza para transferencia de ficheros genéricos entre portátiles o dispositivos Palm y para enviar tarjetas de visita o entradas de la agenda entre teléfonos celulares y otros dispositivos con aplicaciones PIM.

  El cliente OBEX se utiliza para introducir y para recuperar objetos del servidor OBEX. Un objeto puede por ejemplo ser una tarjeta de visita o una cita. El cliente OBEX puede obtener un número de canal RFCOMM del dispositivo remoto utilizando SDP. Esto se hace especificando el nombre del servicio en lugar del número de canal RFCOMM. Los nombres de servicios soportados son: IrMC, FTRN y OPUSH. Es posible especificar el canal RFCOMM como un número [21].

- **SDP (Protocolo de Descubrimiento de Servicios)**:

  El Service Discovery Protocol o SDP permite a las aplicaciones cliente descubrir la existencia de diversos servicios proporcionados por uno o varios servidores de aplicaciones, junto con los atributos y propiedades de los servicios que se ofrecen.

  Estos atributos de servicio incluyen el tipo o clase de servicio ofrecido y el mecanismo o la información necesaria para utilizar dichos servicios. SDP se basa en una determinada comunicación entre un servidor SDP y un cliente SDP.

  El servidor mantiene una lista de registros de servicios, los cuales describen las características de los servicios ofrecidos. Cada registro contiene información sobre un determinado servicio. Un cliente puede recuperar la información de un registro de servicio almacenado en un servidor SDP lanzando una petición SDP. Si el cliente o la aplicación asociada con el cliente decide utilizar un determinado servicio, debe establecer una conexión independiente con el servicio en cuestión. SDP proporciona un mecanismo para el descubrimiento de servicios y sus atributos asociados, pero no proporciona ningún mecanismo ni protocolo para utilizar dichos servicios.

  Normalmente, un cliente SDP realiza una búsqueda de servicios acotada por determinadas características. No obstante hay momentos en los que resulta deseable descubrir todos los servicios ofrecidos por un servidor SDP sin que pueda existir ningún conocimiento previo sobre los registros que pueda contener. Este proceso de búsqueda de cualquier servicio ofrecido se denomina navegación o browsing [22].

Interfaz HCI

A parte de todos los protocolos que componen la pila, la especificación define el HCI (Host Controller Interface), que se encarga de proporcionar una interfaz de comandos al controlador BaseBand, al LC, y nos da acceso al estado del hardware y a los registros de control.

Esta interfaz proporciona una capa de acceso homogénea para todos los dispositivos Bluetooth de banda base. La capa HCI de la máquina intercambia comandos y datos con el firmware del HCI presente en el dispositivo Bluetooth. El driver de la capa de transporte de la controladora de la máquina (es decir, el driver del bus físico) proporciona ambas capas de HCI la posibilidad de intercambiar información entre ellas.

Una de las tareas más importantes de HCI que se deben realizar es el descubrimiento automático de otros dispositivos Bluetooth que se encuentren dentro del radio de cobertura. Esta operación se denomina en inglés inquiry (consulta). Tenga

siempre presente que un dispositivo remoto sólo contesta a la consulta si se encuentra configurado en modo visible (discoverable mode).

Cada dispositivo Bluetooth tiene una dirección identificativa, similar a las direcciones MAC de las tarjetas Ethernet. Esta dirección se necesita para transmitir otro tipo de información a otros dispositivos. Si se realiza una consulta sobre el dispositivo Bluetooth remoto, dicho dispositivo identificará nuestro computador como "nombre.de.su.sistema (ubt0)'', este nombre del dispositivo local se puede modificar en cualquier momento.

El sistema Bluetooth proporciona una conexión punto a punto (con sólo dos unidades Bluetooth involucradas) o también una conexión punto multipunto. En el último caso, la conexión se comparte entre varios dispositivos Bluetooth.

2.2.5 Seguridad

Los temas relacionados con la seguridad Bluetooth soy muy importantes en cualquier dispositivo o sistema. Hay una serie de medidas de seguridad Bluetooth que pueden ser incorporados en los dispositivos para evitar diversas amenazas de seguridad que podrían ocurrir.

Uno de los principales requisitos de Bluetooth es que su conexión con otros equipos debe ser fácil y sencilla, sin embargo, tiene que haber un equilibrio entre esa facilidad de uso y la seguridad [23].

Seguridad Bluetooth básica

Existen varias posibles amenazas como son ataques de denegación de servicio, espionaje, los ataques Man-in-the-middle, la modificación del mensaje, y la apropiación indebida de recursos. La información a la que podrían acceder los hackers es muy personal y sensible ya que podrían tener acceso por ejemplo a las listas de teléfono

Hay tres formas básicas de garantizar la seguridad de Bluetooth:

- Autenticación: En este proceso la identidad de los dispositivos de comunicación es verificada. La autenticación del usuario no es parte de los elementos de seguridad principales de la especificación Bluetooth.
- Confidencialidad: Este proceso evita que alguien no permitido pueda acceder a los datos. Solo los dispositivos autorizados pueden.
- Autorización: Este proceso impide el acceso asegurando que un dispositivo está autorizado para usar un servicio antes de permitirle hacerlo.

Medidas de seguridad proporcionadas por la especificación Bluetooth

Las diversas versiones de las especificaciones Bluetooth detallan cuatro modos de seguridad. Cada dispositivo Bluetooth debe funcionar en uno de los cuatro modos:

- **Modo 1**: Este modo no es seguro. La funcionalidad de autenticación y el cifrado se omite y el dispositivo es susceptible a la piratería. Los dispositivos que integran este modo no emplean mecanismos para prevenir el establecimiento de conexión de otros dispositivos Bluetooth. Puede ser aplicable a dispositivos de corto alcance que operan en un área donde otros dispositivos no pueden estar presentes. Este modo sólo se admite hasta Bluetooth 2.0 + EDR.

- **Modo 2**: en este modo hay un administrador centralizado de seguridad que controla el acceso a los servicios y dispositivos. El administrador mantiene las políticas de control de acceso e interfaces con otros protocolos y usuarios de dispositivos. Para aplicaciones de diferentes requisitos de seguridad se pueden aplicar diferentes niveles de confianza y políticas restrictivas, se puede permitir el acceso a unos servicios y a otros no. Todos los dispositivos Bluetooth pueden soportar el modo 2, sin embargo, los dispositivos v2.1 + EDR solo lo soportan para que haya compatibilidad con versiones anteriores.

- **Modo 3**: en este modo el dispositivo inicia procedimientos de seguridad antes de que se establezca la conexión física. La autenticación y el cifrado se utilizan para todas las conexiones desde y hacia el dispositivo. Los procesos de autenticación y cifrado, utilizan una clave secreta de enlace por separado que se comparte por los dispositivos vinculados una vez que el emparejamiento se ha establecido. El modo 3 solo se admite en equipos Bluetooth 2.0 + EDR o anteriores.

- **Modo 4**: se introdujo en la versión Bluetooth v2.1 + EDR. Los procedimientos de seguridad se inician después de la instalación enlace. El Emparejamiento Simple Seguro (Secure Simple Pairing) usa lo que se denomina *Elliptic Curve Diffie Hellman (ECDH)* que son técnicas para el intercambio de claves y la generación de clave de enlace. Los algoritmos para la autenticación de dispositivos y algoritmos de encriptación son los mismos que los definidos en Bluetooth v2.0 + EDR. Los requisitos de seguridad de los servicios protegidos por el modo 4 son los siguientes:

  - Clave de enlace de autenticación necesaria
  - Clave de enlace no autenticado necesaria
  - No se requiere seguridad

  El hecho de que una clave de enlace sea autenticada depende del Secure Simple Pairing. El modo de seguridad 4 es obligatorio para la comunicación entre los dispositivos que soporten la versión 2.1 + EDR.

2.2.6 Bluetooth API J2ME

El API necesario en J2ME para manejar un dispositivo con comunicaciones Bluetooth es el JSR-82. Se compone de dos paquetes: *javax.bluetooth* y *javax.obex*, a continuación se muestra una tabla con información sobre el contenido básico para establecer la comunicación [24]:

Interfaces	
Nombre	Descripción
DiscoveryListener	Permite a una aplicación especificar un event listener que responda a eventos de descubrimiento de servicios y dispositivos
ServiceRecord	Define un ID uqe es un elemento sin signo de 16 bits y un valor que es un DataElement. También se pueden recuperar atributos de los servicios deseados mediante métodos que implementa

Tabla 2.8. Interfaces básicas JSR 82

Clases	
Nombre	Descripción
DiscoveryAgent	Proporciona métodos para el descubrimiento de servicios y dispositivos. Puede detener el descubrimiento.
RemoteDevice	Proporciona información básica sobre un dispositivo remoto, incluyendo la dirección bluetooth del dispositivo y si nombre
LocalDevice	Proporciona acceso y control sobre el dispositivo bluetooth local.

Tabla 2.9. Clases básicas JSR 82

2.2.7 Usos y aplicaciones actuales

Como se ha comentado anteriormente, Bluetooth es un protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo. Gracias a Bluetooth, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance sin necesidad de cableado, por lo que existe una gran variedad de aplicaciones que hacen uso de esta tecnología:

- Conexión sin cables entre los móviles y equipos de manos libres y kit para vehículos. Estos sistemas permiten la realización de llamadas así como el acceso a la agenda y otras opciones del teléfono móvil desde el coche.
- Red inalámbrica en espacios reducidos donde no sea tan importante un ancho de banda grande.
- Comunicación sin cables entre el ordenador y dispositivos de entrada y salida. Ejemplos de dispositivos son impresoras, teclados y ratones.

- Transferencia de archivos. Los más comunes son audio (como mp3), imágenes (jpg) y vídeo. Pero aparte se pueden enviar archivos de texto, presentaciones, etc.
- Enviar pequeñas publicidades o información turística desde anunciantes a dispositivos con Bluetooth. Un negocio o lugar turístico podría enviar publicidad a teléfonos móviles cuyo Bluetooth estuviera activado al pasar cerca.
- Uso de mandos inalámbricos en consolas como Las Sony PlayStation 3 y Nintendo Wii.
- Sincronización automática entre dispositivos. Se sincroniza la información PIM (Personal Information Management) entre distintos dispositivos. Esta información puede ser la agenda, el calendario, notas, mensajes, etc.

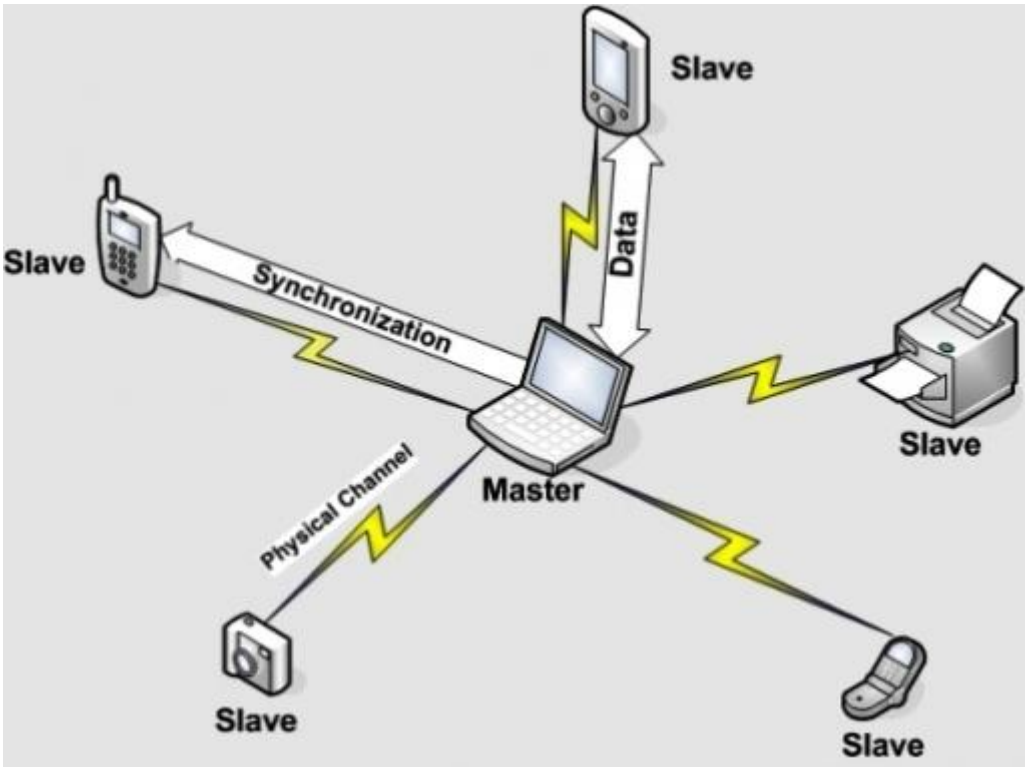


Figura 2.32. Comunicación PC con distintos dispositivos Bluetooth

2.3 Java

2.3.1 Introducción

Java es toda una tecnología orientada al desarrollo de software con el cual podemos realizar cualquier tipo de programa. Hoy en día, la tecnología Java ha cobrado mucha importancia en el ámbito de Internet gracias a su plataforma J2EE. Pero Java no se queda ahí, ya que en la industria para dispositivos móviles también hay una gran acogida para este lenguaje [26].

La tecnología Java está compuesta básicamente por dos elementos: el **lenguaje Java** y su plataforma, la **máquina virtual de Java** (Java Virtual Machine). Aparte de esos dos elementos principales, hay que destacar también su biblioteca estándar para el lenguaje, el **API Java**.

Java también es un lenguaje de programación. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía *Sun Microsystems* con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más punteras.

El *JRE (Java Runtime Environment, o Entorno en Tiempo de Ejecución de Java)* es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final usa el JRE como parte de paquetes software o plugins (o conectores) en un navegador Web. Sun ofrece también el *SDK de Java 2, o JDK (Java Development Kit)* en cuyo seno reside el JRE, e incluye herramientas como el compilador de Java, Javadoc para generar documentación o el depurador. Puede también obtenerse como un paquete independiente, y puede considerarse como el entorno necesario para ejecutar una aplicación Java, mientras que un desarrollador debe además contar con otras facilidades que ofrece el JDK.

Existen varias ediciones de Java, cada una de ellas diseñada para cierto ambiente en particular:

- **Java Estandar Edition (Java SE)**: es la edición que se emplea en los ordenadores personales. Es la que necesitas instalar para poder programar Java en el ordenador aunque tus programas estén destinados a otras ediciones. Hasta la versión 5 de Java se la ha conocido como J2SE, a partir de esta son Java SE 6 y Java SE 7.

- **Java Micro Edition (Java ME)**: esta edición se emplea en dispositivos móviles tales como teléfonos móviles, PDAs, electrodomésticos etc. Se trata de una versión de Java SE reducida con ciertas extensiones enfocadas a las necesidades particulares de esos tipos de dispositivos.

- **Java Enterprise Edition (Java EE)**: esta edición se emplea para hacer aplicaciones empresariales como acceso a bases de datos (JDBC),

aplicaciones Web (JSP y Servlets). Incluye la edición SE completa además de muchas otras funcionalidades.

- **Java Card**: se trata de una edición muy limitada de Java enfocada a aplicaciones que se ejecutan en las tarjetas de crédito con chip.

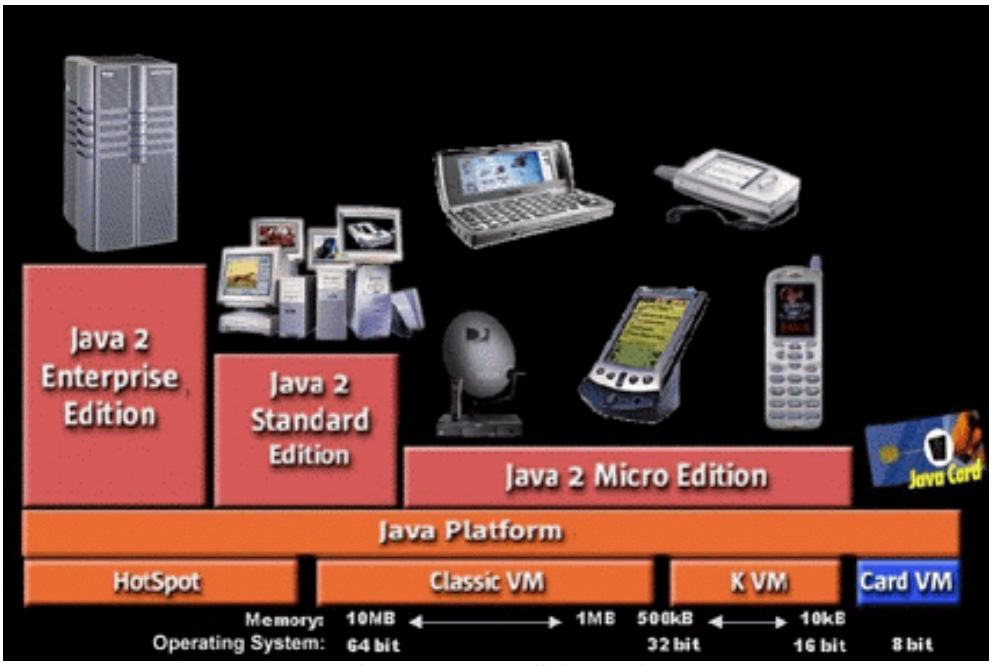


Figura 2.33. Ediciones de Java

2.3.2 Historia

Java fue diseñado en 1990 por *James Gosling*, trabajador de *Sun Microsystems*. En principio se diseñó como software para dispositivos electrónicos de consumo, como calculadoras y microondas. Inicialmente se llamó *Oak* (roble en inglés), aunque tuvo que cambiar debido a que dicho nombre ya estaba registrado por otra empresa y pasó a llamarse *Green* antes de adquirir su nombre definitivo, **Java**.

Gosling observó que muchas de las características que ofrecían C o C++ para este tipo de dispositivos aumentaban de forma alarmante el coste de pruebas y depuración. Por ello, en los sus ratos libres creó un lenguaje de programación donde intentaba solucionar los fallos que encontraba en C++. Es decir, en lugar de tratar únicamente de optimizar las técnicas de desarrollo y dar por sentado la utilización de C o C++, el equipo de Gosling se planteó que tal vez estos lenguajes eran demasiado complicados como para conseguir reducir de forma apreciable la complejidad asociada a este campo. Por este motivo, su primera propuesta fue idear un nuevo lenguaje de programación lo más sencillo posible, con el objeto de que se pudiese adaptar con facilidad a cualquier entorno de ejecución. Basándose en el conocimiento y estudio de gran cantidad de lenguajes, este grupo decidió recoger las características esenciales que debía tener un lenguaje de programación moderno y potente, pero eliminando todas aquellas funciones que no eran absolutamente imprescindibles.

El fracaso comercial de *FirstPerson*, la filial creada por Sun para este mercado, llevó al lenguaje al olvido. Tuvo que ser *Bill Joy*, cofundador de Sun y uno de los desarrolladores principales del sistema operativo Unix de Berkeley, quien lo sacara de él, ya que juzgó que Internet podría llegar a ser el campo de juego adecuado para disputar a Microsoft su primacía casi absoluta en el terreno del software, y vio en Oak el instrumento idóneo para llevar a cabo estos planes. Para poderlo presentar en sociedad se tuvo que modificar el nombre de este lenguaje de programación y se tuvo que realizar una serie de modificaciones de diseño para poderlo adaptar al propósito mencionado. Y así Java fue presentado en sociedad en agosto de 1995.

El término Java fue acuñado en una cafetería frecuentada por algunos de los miembros del equipo. Pero no está claro si es un acrónimo o no, aunque algunas fuentes señalan que podría tratarse de las iniciales de sus creadores: ***J**ames Gosling, **A**rthur **V**an Hoff, y **A**ndy Bechtolsheim*. Otros abogan por el siguiente acrónimo, ***J**ust **A**nother **V**ague **A**cronym* ("sólo otro acrónimo ambiguo más"). La hipótesis que más fuerza tiene es la que Java debe su nombre a un tipo de café disponible en la cafetería cercana, de ahí que el icono de java sea una taza de café caliente. Un pequeño signo que da fuerza a esta teoría es que los 4 primeros bytes (el *número mágico*) de los archivos *.class* que genera el compilador, son en hexadecimal, 0xCAFEBABE. A pesar de todas estas teorías, el nombre fue sacado al parecer de una lista aleatoria de palabras.

Figura 2.34. Logo de java

En abril de 2009 *Oracle* adquirió Sun Microsystems, lo que generó temor en la comunidad ante la posible mercantilización del lenguaje de programación a objetos más popular actualmente. Por ahora Oracle ha seguido manteniendo Java, siendo las versiones posteriores a la 6 bajo su control [27].

2.3.3 Características

Una de las principales características por las que Java se ha hecho tan famoso es que es un **lenguaje independiente de la plataforma**. Eso quiere decir que si hacemos un programa en Java podrá funcionar en cualquier ordenador del mercado.

Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina Virtual de Java (**JMV**) para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente. A parte de esa portabilidad entre sistemas operativos java se define por las siguientes características [28]:

- **Lenguaje simple**. Java posee una curva de aprendizaje muy rápida, resulta relativamente sencillo escribir applets desde el principio. Debido a su semejanza con C y C++, y dado que mucha gente los conoce aunque sea de forma elemental, resulta muy fácil aprender Java. Los programadores experimentados en C++ pueden migrar muy rápidamente a Java y ser productivos en poco tiempo.

- **Orientado a objetos**. Java fue diseñado como un lenguaje orientado a objetos desde el principio. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos o funciones que manipulan esos datos. La tendencia apunta cada vez más a la programación orientada a objetos, especialmente en entornos cada vez más complejos y basados en red.

- **Distribuido**. Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

- **Interpretado y compilado a la vez**. Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (*run-time*).

- **Robusto**. Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

- **Indiferente a la arquitectura**. Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples

plataformas hardware y software. El resto de problemas los soluciona el intérprete de Java.

- **Multihilos**. Las aplicaciones que solo pueden ejecutar una acción a la vez están muy limitadas. Java soporta sincronización de múltiples hilos de ejecución (multithreading) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas.

- **Dinámico**. El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

2.3.4 Java SE

Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. Esta edición de Java es la que en cierta forma recoge la iniciativa original del lenguaje Java y ha avanzado a lo largo de los años tanto en funcionalidad como en logros partiendo de la versión JDK 1.0 hasta la última lanzada a mediados de 2011 Java SE 7.

Esta edición de Java contiene el conjunto básico de herramientas usadas para desarrollar Java Applets, así como las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de usuario, multimedia, redes de comunicación, etc.

Arquitectura

En la *figura 2.35* se muestran los paquetes por los que está compuesta la Plataforma Java Standard Edition en su versión 5.0. Como se observa, las máquinas virtuales se apoyan sobre la base de los diferentes sistemas operativos, sin pertenecer a uno concretamente y a su vez sobre las máquinas se encuentran las librerías que conforman la plataforma.

Toda la información contenida en JRE se encuentra englobada en la JDK, además de que JDK añade funcionalidades extra para el desarrollo de aplicaciones:

- **JRE (Java Runtime Environment)**: se trata de un entorno para ejecutar (runtime) programas Java. Esta situación se da cuando empresas de Software diseñan alguna aplicación en Java para su producto. Cabe mencionar que muchos productos que utilizan Java ya incluyen un JRE para evitarle la molestia de instalarlo, uno de estos es Oracle; sin embargo, muchos otros requieren de la instalación por parte del usuario.

- **JDK (Java Development Kit) o SDK (Standard Development Kit)**: este componente incluye el API de Java, el JRE (JVM), el compilador de Java y otras funcionalidades definidas por Sun. El API de Java es un conjunto de clases que es utilizado para generar programas básicos en el lenguaje; Partiendo de estas clases se generan todos los programas, interfaces y elementos programados en Java, inclusive a partir de estas clases se pueden definir otras clases específicas que serán utilizadas por el programa o producto. Una vez que estan definidos los programas/clases en Java, aún es necesario compilarlas para producir el denominado byte-code o class files (este byte-code puede ser comparado con un binario), y es este byte-code el que interpreta el JRE [29].

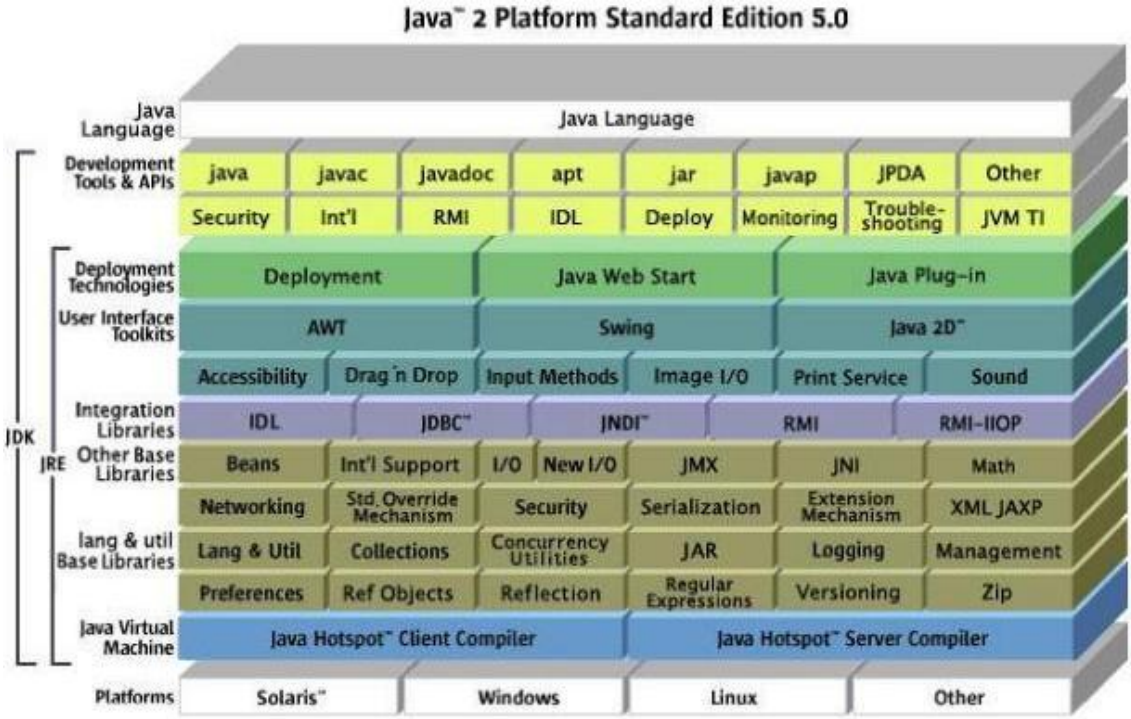


Figura 2.35. Arquitectura Plataforma Java Standard Edition 5.0

En la *figura 2.35* se muestra la arquitectura de la versión 5.0 que se compone de varios paquetes y librerías, a continuación vemos algunos de ellos:

- *java.applet*: Creado para soportar la creación de applet Java, el paquete java.applet permite a las aplicaciones ser descargadas sobre una red y ejecutarse dentro de una sandbox. Las restricciones de seguridad son impuestas fácilmente en la sandbox. Un desarrollador, por ejemplo, puede aplicar una firma digital a un applet, etiquetándola como segura. Haciéndolo permite al usuario conceder permiso al applet para realizar operaciones restringidas (tales como acceder al disco duro local), y elimina alguna o todas las restricciones de la sandbox.

- *java.beans*: contiene varias clases para desarrollar y manipular beans, componentes reutilizables definidos por la arquitectura JavaBeans. La arquitectura suministra mecanismos para manipular propiedades de componentes y lanzar eventos cuando esas propiedades cambian.

- *java.awt*: La *Abstract Window Toolkit* contiene rutinas para soportar operaciones básicas GUI (Graphic User Interface) y utiliza ventanas básicas desde el sistema nativo subyacente. Muchas implementaciones independientes de la API Java implementan todo excepto AWT, el cual no es usado por la mayoría de las aplicaciones de lado de servidor.

- *java.rmi*: este paquete suministra la invocación a métodos remotos Java para soportar llamadas a procedimientos remotos entre dos aplicaciones Java que se ejecuten en diferentes JVM. Esto es esencial para tener en cuenta en la certificación.

- *java.security*. Soporte para seguridad, incluyendo el algoritmo de resumen de mensaje.

- *java.sql*: Una implementación de la API JDBC (usada para acceder a bases de datos SQL) se agrupa en este paquete.

- *javax.swing*: swing es una colección de rutinas que se construyen sobre java.awt para suministrar un toolkit de widgets independiente de plataforma. Swing usa las rutinas de dibujado 2D para renderizar los componentes de interfaz de usuario en lugar de confiar en el soporte GUI nativo subyacente del Sistema operativo.

- *javax.swing.text.html.parser*: Suministra el parser de HTML tolerante a errores que se usa para escribir varios navegadores web.

JVM (Máquina Virtual de Java)

El lenguaje Java es a la vez compilado e interpretado. Con el compilador se convierte el código fuente que reside en archivos cuya extensión es **.java**, a un conjunto de instrucciones que recibe el nombre de *bytecodes* que se guardan en un archivo cuya extensión es **.class**. Estas instrucciones son independientes del tipo de ordenador. El intérprete ejecuta cada una de estas instrucciones en un ordenador específico (Windows, Macintosh, etc). Solamente es necesario, por tanto, compilar una vez el programa, pero se interpreta cada vez que se ejecuta en un ordenador (*figura 2.36*) [30].

Cada intérprete Java es una implementación de la *Máquina Virtual Java (JVM)*. Los bytecodes posibilitan el objetivo de "write once, run anywhere", de escribir el programa una vez y que se pueda correr en cualquier plataforma que disponga de una implementación de la JVM. Por ejemplo, el mismo programa Java puede correr en Windows 7, Linux, Macintosh, etc.

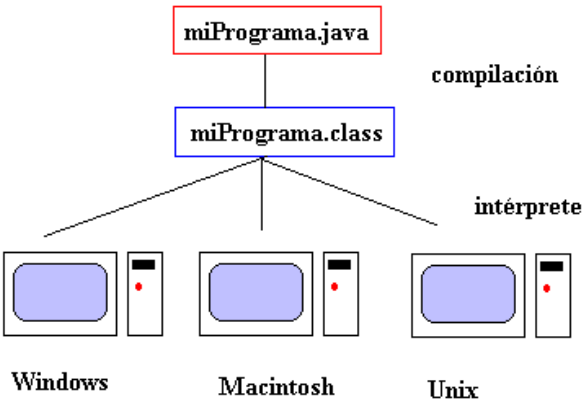


Figura 2.36. Compilación Java

Java es, por tanto, algo más que un lenguaje, ya que la palabra Java se refiere a dos cosas inseparables: el lenguaje que nos sirve para crear programas y la Máquina Virtual Java que sirve para ejecutarlos. Como vemos en la figura, el API de Java y la Máquina Virtual Java forman una capa intermedia (Java platform) que aísla el programa Java de las especificidades del hardware (hardware-based platform).

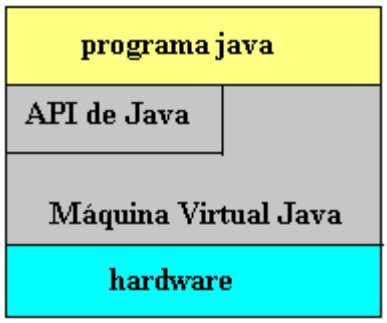


Figura 2.37. Situación de la máquina virtual

La Máquina Virtual Java (JVM) es el entorno en el que se ejecutan los programas Java, su misión principal es la de garantizar la portabilidad de las aplicaciones Java. Define esencialmente un ordenador abstracto y especifica las instrucciones bytecodes que este ordenador puede ejecutar. El intérprete Java específico ejecuta las instrucciones que se guardan en los archivos cuya extensión es .class. Las tareas principales de la JVM son reservar espacio en memoria para los objetos creados, liberar la memoria no usada (garbage collection), asignar variables a registros y pilas, llamar al sistema huésped para ciertas funciones, como los accesos a los dispositivos y vigilar el cumplimiento de las normas de seguridad de las aplicaciones Java.

El JDK proporciona dos implementaciones de la máquina virtual:

- **Java HotSpot Client VM:** el JDK viene con este tipo de máquina virtual en las plataformas que normalmente utilizan aplicaciones cliente. La máquina virtual se ajusta para reducir el tiempo de inicio y la memoria.

- **Java HotSpot Server VM:** esta máquina virtual está diseñada para una máxima velocidad de ejecución del programa.

Versiones

En la *figura 2.35* se muestra la arquitectura de la versión 5.0 pero con el paso del tiempo se han ido creando nuevas versiones añadiendo mejoras a las características de versiones anteriores:

Versiones Java 1		
Versión	Fecha de la versión	Descripción
Java 1 Java 1.0 - JDK 1.0	Enero 1996	8 paquetes, 212 clases Primera versión pública. La presión hizo que se hiciera pública demasiado pronto, lo cual significa que el diseño del lenguaje no es demasiado bueno y hay montones de errores. Respecto a seguridad, es restrictivo por defecto,no dejando hacer demasiado al código no fiable
Java 1 Java 1.1 - JDK 1.1	Febrero 1997	23 paquetes, 504 clases. Mejoras de rendimiento en la JVM, nuevo modelo de eventos en AWT, clases anidadas, serialización de objetos,API de JavaBeans, archivos jar, internacionalización, API Reflection (Reflexión), JDBC (Java Data baseConnectivity), RMI (Remote Method Invocation). Se añade la firma del código y la autentificación. Es la primeraversión lo suficientemente estable y robusta

Tabla 2.10. Versiones Java 1

Versiones Java 2		
Versión	Fecha de la versión	Descripción
Java 2 Java 1.2 - J2SE 1.2	Diciembre 1998	59 paquetes, 1520 clases. JFC (Swing), Drag and Drop, Java2D, Corba, API Collections. Se producen notables mejoras a todos los niveles.Para enfatizar esto Sun lo renombra como "Java 2". El JDK (Java Development Kit) se renombra como SDK(Software Development Kit). Se divide en J2SE, J2EE y J2ME
Java 2 Java 1.3 - J2SE 1.3	Mayo 2000	77 paquetes, 1595 clases. Orientada sobre todo a la resolución de errores y a la mejora del rendimiento; se producen algunos cambios menores como la inclusión de JNDI (Java Naming and Directory Interface) y la API Java Sound. También incluye un nuevo compilador de alto rendimiento JIT (Just In Time)

Java 2 Java 1.4 - J2SE 1.4	Febrero 2002	103 paquetes, 2175 clases. También conocido como Merlin. Mejora notablemente el rendimiento y añade entre otros soporte de expresiones regulares, una nueva API de entrada/salida de bajo nivel (NIO, New I/O), clases para el trabajo con Collections, procesado de XML; y mejoras de seguridad como el soporte para la criptografía mediantelas Java Cryptography Extension (JCE), la inclusión de la Java Secure Socket Extension (JSSE) y el Java Authentication and Authorization Service (JAAS)
Java 2 Java 1.5 - J2SE 5.0	Octubre 2004	131 paquetes, 2656 clases. También conocido como Tiger, renombrado por motivos de marketing como Java 5.0.Incluye como principales novedades: tipos genéricos (generics), autoboxing/unboxing conversiones impliticas entre tipos primitivos y los wrappers correspondientes, Enumerados, Bucles simplificados, printf, Funciones con número de parámetros variable, Metadatos en clases y métodos

Tabla 2.11. Versiones Java 2

Versiones Java modernas		
Versión	Fecha de la versión	Descripción
Java SE 6	Diciembre 2006	También conocido como Mustang. Estuvo en desarrollo bajo la JSR 270. En esta versión, Sun cambió el nombre"J2SE" por Java SE y eliminó el ".0" del número de versión.Los cambios más importantes introducidos en esta versión son: Incluye un nuevo marco de trabajo y APIs quehacen posible la combinación de Java con lenguajes dinámicos como PHP, Python, Ruby y JavaScript. Incluye elmotor Rhino, de Mozilla, una implementación de Javascript en Java. Incluye un cliente completo de Servicios Web ysoporta las últimas especificaciones para Servicios Web, como JAX-WS 2.0, JAXB 2.0, STAX y JAXP. Mejoras enla interfaz gráfica y en el rendimiento
Java SE 7	Julio 2011	Nombre clave Dolphin. Soporte para XML dentro del propio lenguaje. Un nuevo concepto de superpaquete. Soporte para closures. Introducción de anotaciones estándar para detectar fallos en el software

Tabla 2.12. Versiones Java modernas

2.3.5 J2ME

Esta versión de Java está enfocada a la aplicación de la tecnología Java en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes. Esta edición tiene unos componentes básicos que la diferencian de las otras versiones, entre ellos

están el uso de una máquina virtual denominada *KVM* (Kilo Virtual Machine, debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en vez del uso de la JVM clásica y la inclusión de un pequeño y rápido recolector de basura.

Hay varios componentes que forman parte de esta tecnología: las **máquinas virtuales Java** con diferentes requisitos para pequeños dispositivos, las **configuraciones** que son un conjunto de clases básicas orientadas a conformar el corazón de las implementaciones, los **perfiles** que también son bibliotecas de clases pero orientadas a implementar funcionalidades más específicas y los **paquetes opcionales** que amplían a las configuraciones y perfiles con funcionalidades extra fruto de los requerimientos del mercado [31].

Máquinas virtuales

Las implementaciones tradicionales de JVM son, en general, muy pesadas en cuanto a memoria ocupada y requerimientos computacionales. J2ME define varias JVMs de referencia adecuadas al ámbito de los dispositivos electrónicos que, en algunos casos, suprimen algunas características con el fin de obtener una implementación menos exigente.

Como veremos en el siguiente punto, existen dos configuraciones diferentes (CLDC y CDC), por lo que cada una de ellas necesita su propia máquina virtual [32]:

- **KVM**: es para CLDC y se corresponde con la Máquina Virtual más pequeña desarrollada por Sun. Su nombre KVM proviene de Kilobyte (haciendo referencia a la baja ocupación de memoria, entre 40Kb y 80Kb). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria. La KVM está escrita en lenguaje C, aproximadamente unas 24000 líneas de código, y fue diseñada para ser:

  - Pequeña, con una carga de memoria entre los 40Kb y los 80 Kb, dependiendo de la plataforma y las opciones de compilación.
  - Alta portabilidad.
  - Modulable.
  - Lo más completa y rápida posible y sin sacrificar características para las que fue diseñada.

  Sin embargo, esta baja ocupación de memoria hace que posea algunas limitaciones con respecto a la clásica Java Virtual Machine (JVM):

  - No hay soporte para tipos en coma flotante. No existen por tanto los tipos double ni float. Esta limitación está presente porque los

dispositivos carecen del hardware necesario para estas operaciones.

- No existe soporte para JNI (Java Native Interface) debido a los recursos limitados de memoria.
- No existen cargadores de clases (class loaders) definidos por el usuario. Sólo existen los predefinidos.
- No se permiten los grupos de hilos o hilos daemon. Cuando queramos utilizar grupos de hilos utilizaremos los objetos Colección para almacenar cada hilo en el ámbito de la aplicación.
- No existe la finalización de instancias de clases. No existe el método Object.finalize().
- No hay referencias débiles.
- Limitada capacidad para el manejo de excepciones debido a que el manejo de éstas depende en gran parte de las APIs de cada dispositivo por lo que son éstos los que controlan la mayoría de las excepciones.
- Reflexión

Cabe destacar que el verificador de clases estándar de Java (es el encargado de rechazar las clases no válidas en tiempo de ejecución) es demasiado grande para la KVM. Por esta razón, los dispositivos que usen la configuración CDLC y KVM introducen un algoritmo de verificación en dos pasos, uno mediante el desarrollo y otro en el dispositivo cliente final.

- **CVM**: La Compact Virtual Machine ha sido tomada como Máquina Virtual Java de referencia para la configuración CDC y soporta las mismas características que la Máquina Virtual de J2SE. Está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y en torno a 2Mb o más de memoria RAM. Las características que presenta esta Máquina Virtual son:
  - Sistema de memoria avanzado.
  - Tiempo de espera bajo para el recolector de basura.
  - Separación completa de la VM del sistema de memoria.
  - Recolector de basura modularizado.
  - Portabilidad.
  - Rápida sincronización.

- Ejecución de las clases Java fuera de la memoria de sólo lectura (ROM).
- Soporte nativo de hilos.
- Baja ocupación en memoria de las clases.
- Proporciona soporte e interfaces para servicios en Sistemas Operativos de
- Tiempo Real.
- Conversión de hilos Java a hilos nativos.
- Soporte para todas las características de Java2 v1.3 y librerías de seguridad, referencias débiles, Interfaz Nativa de Java (JNI), invocación remota de métodos (RMI), Interfaz de depuración de la Máquina Virtual (JVMDI).

Configuración

Una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Éstas APIs describen las características básicas, comunes a todos los dispositivos, que son: características soportadas del lenguaje de programación Java, características soportadas por la Máquina Virtual Java y bibliotecas básicas de Java y APIs soportadas [32].

Como ya hemos visto con anterioridad, existen dos configuraciones en J2ME:

- **Configuración de dispositivos con conexión, CDC (Connected Device Configuration)**: orientada a dispositivos con cierta capacidad computacional y de memoria. Por ejemplo, decodificadores de televisión digital, televisores con internet, algunos electrodomésticos y sistemas de navegación en automóviles. CDC usa una Máquina Virtual Java similar en sus características a una de J2SE, pero con limitaciones en el apartado gráfico y de memoria del dispositivo. Ésta Máquina Virtual es la que hemos visto como CVM (Compact Virtual Machine). La CDC está enfocada a dispositivos con las siguientes capacidades: procesador de 32 bits, 2 Mb o más de memoria total, poseer la funcionalidad completa de la Máquina Virtual de Java 2 y conectividad a algún tipo de red.

  La *tabla 2.13* nos muestra las librerías incluidas en la CDC.

Librerías CDC	
Nombre de paquete CDC	Descripción
java.io	Clases e interfaces estándar de E/S.
java.lang	Clases básicas del lenguaje.
java.lang.ref	Clases de referencia.
java.lang.reflect	Clases e interfaces de reflection.
java.math	Paquete de matemáticas.
java.net	Clases e interfaces de red.
java.security	Clases e interfaces de seguridad
java.security.cert	Clases de certificados de seguridad.
java.text	Paquete de texto.
java.util	Clases de utilidades estándar.
java.util.jar	Clases y utilidades para archivos JAR.
java.util.zip	Clases y utilidades para archivos ZIP y comprimidos
javax.microedition.io	Clases e interfaces para conexión genérica CDC.

Tabla 2.13. Librerías de la configuración CDC

- **Configuración de dispositivos limitados con conexión, CLDC (Connected Limited Device Configuration):** La CLDC está orientada a dispositivos dotados de conexión y con limitaciones en cuanto a capacidad gráfica, cómputo y memoria. Un ejemplo de estos dispositivos son: teléfonos móviles, buscapersonas (pagers), PDAs, organizadores personales, etc. Ya hemos dicho que CLDC está orientado a dispositivos con ciertas restricciones. Algunas de estas restricciones vienen dadas por el uso de la KVM, necesaria al trabajar con la CLDC debido a su pequeño tamaño. Los dispositivos que usan CLDC deben cumplir los siguientes requisitos: disponer como mínimo entre 160 Kb y 512 Kb de memoria total disponible, 128 Kb de memoria no volátil para la Máquina Virtual Java y las bibliotecas CLDC, procesador de 16 o 32 bits con al menos 25 Mhz de velocidad, ofrecer bajo consumo (debido a las baterías), tener conexión a algún tipo de red, normalmente sin cable, con conexión intermitente y ancho de banda limitado (unos 9600 bps).

  La t*abla 2.14* nos muestra las librerías incluidas en la CLDC.

Librerías CLDC	
Nombre de paquete CLDC	Descripción
java.io	Clases y paquetes estándar de E/S. Subconjunto de J2SE.
java.lang	Clases e interfaces de la Máquina Virtual. Subconjunto de J2SE.
java.util	Clases, interfaces y utilidades estándar. Subconjunto de J2SE.
javax.microedition.io	Clases e interfaces para conexión genérica CDC.

Tabla 2.14. Librerías de la configuración CLDC

Perfiles

Para conformar un entorno de ejecución completo orientado a una categoría de dispositivos, las configuraciones se han de combinar con un conjunto de APIs de un nivel más alto, llamadas perfiles, que van un paso más allá en la definición del modelo de ciclo de vida de las aplicaciones, la interfaz de usuario y acceso a las propiedades específicas de los dispositivos.

En la actualidad existen los siguientes perfiles asociados a J2ME: Mobile Information Device Profile (MIDP), Foundation Profile, Personal Profile, Personal Basis Profile [32].

- **Mobile Information Device Profile (MIDP)**: está diseñado para teléfonos móviles y PDAs con capacidades básicas. Ofrece la funcionalidad básica para las aplicaciones móviles, incluyendo la interfaz de usuario, conectividad a redes, almacenamiento local de datos y gestión del ciclo de vida de las aplicaciones. Al combinarlo con la configuración CLDC, MIDP proporciona un entorno de ejecución Java completo que incrementa la capacidad de los dispositivos móviles y que reduce el consumo de memoria y energía. Las aplicaciones que realizamos utilizando MIDP reciben el nombre de MIDlets (por simpatía con APPlets). Decimos así, que un **MIDlet** es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC.

- **Foundation Profile (FP)**: los perfiles CDC están organizados en capas de forma que permitan la agregación según se precise para proporcionar funcionalidad a las aplicaciones para distintos tipos de dispositivos. El FP es el perfil de más bajo nivel para el CDC. Proporciona una implementación lista para el trabajo en red que se puede emplear en implementaciones embebidas que carecen de interfaz de usuario. También se puede combinar con los perfiles Personal Basis y Personal para los dispositivos que precisan de una interfaz gráfica de usuario (IGU).

- **Personal Profile (PP)**: el perfil Personal, es el perfil para CDC orientado a dispositivos que requieren una IGU completa o capacidad de ejecutar applets de Internet, como por ejemplo PDAs de gama alta, consolas de juegos, etc. Incluye todas las bibliotecas de funciones de la Java Abstract Window Toolkit (AWT) y ofrece fidelidad Web, permitienddo la ejecución de applets diseñados para utilización en entornos de sobremesa. PP reemplaza la tecnología PersonalJavaTM.

- **Personal Basis Profile (PBP)**: el perfil Personal Basis es un subconjunto del perfil Personal y proporciona un entorno de aplicación para dispositivos con conexión que toleran un nivel de presentación gráfica básico o que precisan de conjuntos de herramientas (toolkits) gráficas especializadas para aplicaciones específicas. Al igual que el

perfil Personal, está pensado para ejecutarse sobre la configuración CDC.

Paquetes opcionales

La plataforma J2ME se puede ampliar combinando varios paquetes opcionales con CLDC y CDC junto con sus perfiles. Estos paquetes se han creado para responder a requisitos concretos de mercado y ofrecen un conjunto de APIs estándares para utilizar tanto tecnologías existentes como emergentes; entre estas se incluyen Bluetooth, servicios Web, mensajeríaa wireless, capacidades multimedia o conectividad a bases de datos. Dado que son modulares, los fabricantes de dispositivos pueden incorporarlos según los vayan necesitando para mejorar las características soportadas [32]:

- **Information Module Profile (IMP), JSR 195**: es un paquete que se puede combinar con CLDC y MIDP. Proporciona un entorno de aplicación para dispositivos embebidos que no tiene grandes capacidades gráficas o con recursos limitados de alguna otra manera como paneles de emergencia, parquímetros, sistemas de alarma domésticos y similares. IMP proporciona la funcionalidad de aplicación básica para aplicaciones máquina-máquina, incluyendo conectividad de red, almacenamiento local y gestión del ciclo de vida de la aplicación.

- **Wireless Messaging API (WMA); JSR 120, JSR 205**: es un paquete que se puede utilizar sobre CLDC y MIDP (1.0 y 2.0) y CDC y sus perfiles. El API para mensajería sin cables (Wireless Messaging API, WMA) proporciona acceso independiente de plataforma a recursos de comunicación sin cable como la mensajería SMS (Short Message Service, SMS). Existen dos especificaciones:

  - WMA 1.0, la especificación original a partir de JSR 120.

  - WMA 1.1, especificación que incluye cambios para considerar el framework de seguridad y la arquitectura de comunicación de MIDP 2.0

- **Mobile Media API (MMAPI); JSR 135:** es un paquete que se puede utilizar sobre CLDC y MIDP (1.0 y 2.0) y CDC y sus perfiles. Este paquete extiende la funcionalidad de la plataforma J2ME incorporando soporte de audio, video y otros tipos de datos multimedia basados en tiempo a dispositivos de recursos limitados. Existen dos especificaciones:

  - WMA 1.0, la especificación original a partir de JSR 135.

  - WMA MMAPI 1.1, especificación que incluye cambios para considerar el framework de seguridad de MIDP 2.0

- **Location API for J2ME (JSR-179)**: es un paquete que se puede utilizar sobre CLDC 1.1 y CDC. Esta especificación permite la localización de dispositivos móviles para dispositivos con recursos limitados. El API se ha diseñado para generar información sobre la localización geográfica actual del terminal para las aplicaciones Java. El API cubre la obtención de la localización geográfica presente, la orientación del terminal y acceder a una base de datos de mapas almacenados en el terminal.

- **SIP API for J2ME (JSR-180)**: es un paquete que se puede utilizar sobre CLDC. El protocolo Session Initiation Protocol (SIP) se utiliza para establecer y gestionar sesiones IP multimedia. El mismo mecanismo se puede utilizar para proporcionar mensajería instantánea, presencia y servicios de juego. El API se ha diseñado para permitir que las aplicaciones Java envíen y reciban mensajes SIP.

- **Security and Trust Services API for J2ME (JSR-177)**: es un paquete que se puede utilizar sobre CLDC. Este paquete amplía las características de seguridad para la plataforma J2ME añadiendo APIs de cifrado, servicio de firma digital y gestión de credenciales de usuario.

- **Mobile 3D Graphics (JSR-184)**: es un paquete que se puede utilizar sobre CLDC y MIDP (1.0 y 2.0) El paquete Mobile 3D Graphics API (M3G) permite generar gráficos tridimensionales a frecuencias de imagen interactivas en dispositivos móviles de recursos restringidos. También incluye utilidades para la gestión de escenas 3D y animaciones así como un formato de archivo para despliegue eficaz OTA para contenido 3D.

- **J2ME Web Services APIs (WSA), JSR 172**: es un paquete que se puede utilizar sobre CLDC y MIDP (1.0 y 2.0). El paquete J2ME Web Services APIs (WSA) amplía la plataforma de servicios web para incluir J2ME. Estas APIs permiten que los dispositivos J2ME puedan ser clientes de servicios web mediante un modelo de programación consistente con la plataforma estándar de servicios web.

- **Bluetooth API (JSR-82, Motorola, Java Partner Site)**: es un paquete que se puede utilizar sobre CLDC y MIDP (1.0 y 2.0). Proporciona un estándar para la creación de aplicaciones Bluetooth, de forma que las aplicaciones desarrolladas con el paquete opcional puedan ejecutarse utilizando esta tecnología.

- **J2ME RMI Optional Package, (RMI OP); JSR 66**: es un paquete que se puede utilizar sobre CDC. EL paquete RMI Optional (RMI OP) permite a dispositivos de consumo y aplicaciones embebidas interactuar como y con aplicaciones distribuidas. Se amplían las características de RMI existentes para que pequeños dispositivos interactúen entre sí. RMI OP es un subconjunto del API RMI de J2SE.

- **JDBC Optional Package for CDC/Foundation Profile API (JSR-169)**: es un paquete que se puede utilizar sobre CDC. Es un paquete que define un subconjunto del API JDBC 3.0 que se puede utilizar en J2ME para procesar datos de repositorios, habitualmente BBDD relacionales, mediante SQL y para manipular datos tabulares como si fueran JavaBeans

2.4 FTP

2.4.1 Introducción

El acrónimo de FTP es protocolo de transferencia de ficheros (*File Transfer Protocol*) y es un protocolo de red que permite a usuarios transferir ficheros entre ordenadores en una **red TCP/IP**, basándose en el modelo **cliente/servidor**.

FTP tiene sus orígenes en 1971, y aunque ha evolucionado con el paso de los años, es uno de los protocolos más antiguos que todavía están en uso. Hoy en día se usa principalmente en redes corporativas y la red más grande que existe, Internet.

El funcionamiento es sencillo. Una persona desde su ordenador invoca un programa cliente FTP para conectar con otro ordenador, que a su vez tiene instalado el programa servidor FTP. Una vez establecida la conexión y debidamente autenticado el usuario con su contraseña, se pueden empezar a intercambiar archivos de todo tipo.

El protocolo FTP es el sistema de transferir archivos más estable y fiable que hay en Internet. Esto significa que la descarga y subida de archivos que hagas tendrán más opciones de completarse sin errores de transferencia, y quedarán intactos después del envío. El servicio FTP es ofrecido por la capa de aplicación del modelo de capas de red TCP/IP al usuario, utilizando normalmente el puerto de red 20 y el 21. Un problema básico de FTP es que está pensado para ofrecer la máxima velocidad en la conexión, pero no la máxima seguridad, ya que todo el intercambio de información, desde el login y password del usuario en el servidor hasta la transferencia de cualquier archivo, se realiza en texto plano sin ningún tipo de cifrado, con lo que un posible atacante puede capturar este tráfico, acceder al servidor y/o apropiarse de los archivos transferidos. Para solucionar este problema son de gran utilidad aplicaciones como *scp* y *sftp*, incluidas en el paquete *SSH*, que permiten transferir archivos pero cifrando todo el tráfico [33].

2.4.2 Esquema de funcionamiento FTP

FTP es un servicio orientado a conexión concurrente que funciona sobre dos puertos 21 para el intercambio de comandos y 20 para los datos. La estructura general de funcionamiento es la que se muestra la siguiente *figura 2.38*.

El **Servidor FTP** es la máquina a la que el usuario se quiere conectar y debe aceptar las sesiones, debe tener activo el servicio FTP. Los componentes que aparecen en el esquema por parte del servidor son [34]:

- **Servidor PI (Protocol Interpreter)**: El intérprete de protocolo del servidor escucha en el puerto 21 los comandos que le envía el intérprete de protocolo del cliente y controla el proceso de transferencia de datos del servidor.
- **Servidor DTP (Data Transfer Protocol)**: El protocolo de transferencia de datos del servidor se utiliza para transmitir los datos entre el servidor

y el protocolo de transferencia de datos del cliente. Puede estar en modo pasivo a la escucha de conexiones en el puerto 20 de datos.

El **Cliente FTP** es la máquina con la que nos conectamos al servidor FTP. Está compuesta por los siguientes elementos:

- **Interfaz de usuario**: conjunto de comandos de alto nivel que el usuario puede memorizar más fácilmente que los comandos FTP que se envían entre cliente y servidor.
- **Cliente PI**: el intérprete de protocolo de usuario inicia el control de la conexión a través del puerto 21 con el servidor FTP, envía los comandos FTP una vez codificados por la interfaz de usuario y los envía al intérprete de protocolo del servidor, y controla el proceso de transferencia de los archivos (DTP).
- **Cliente DTP**: el proceso de transferencia de datos "escucha" el puerto de datos (20) aceptando conexiones para la transferencia de ficheros.

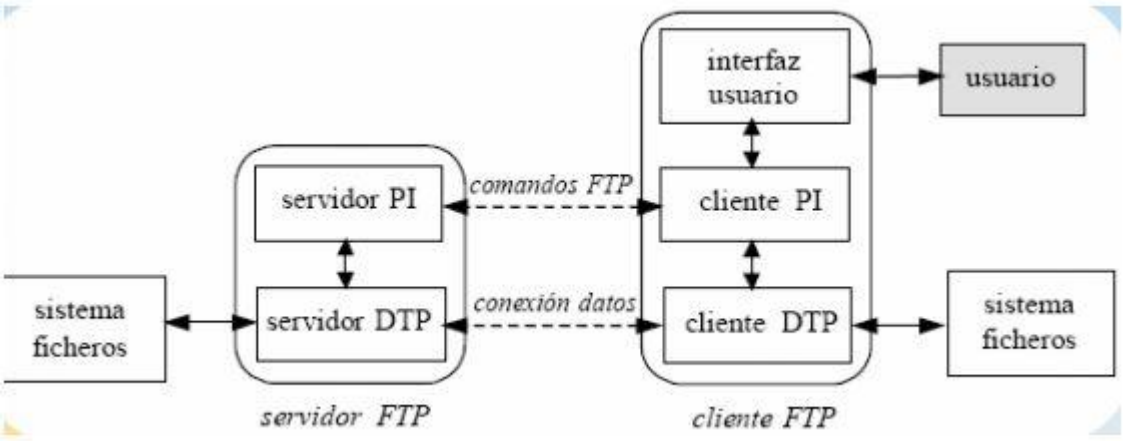


Figura 2.38. Esquema de funcionamiento FTP

En el modelo descrito en la *figura 2.38*, el PI del cliente inicia la conexión TCP por el puerto 21. Al iniciarse, se envían los comandos u órdenes mediante dicho PI al PI del servidor. Estas órdenes FTP especifican parámetros para la conexión de datos (puerto de datos, modo de transferencia, tipo de representación y estructura) y la naturaleza de la operación sobre el sistema de archivos (almacenar, recuperar, añadir, borrar, etc.). Si el servidor acepta la conexión, solicita una identificación al usuario, pudiéndose realizar un acceso anónimo (no aceptado por todos los servidores). Cuando se solicita un archivo del servidor, se establece una conexión TCP por el puerto 20 entre el DTP del cliente y el servidor para la transmisión de datos.

La comunicación entre cliente y servidor es independiente del sistema de archivos utilizado en cada ordenador, de manera que no importa que sus sistemas operativos sean distintos, porque las entidades que se comunican entre sí son los PI y los DTP, que usan el mismo protocolo estandarizado: el FTP.

También hay que destacar que la conexión de datos es bidireccional, es decir, se puede usar simultáneamente para enviar y para recibir, y no tiene por qué existir todo el tiempo que dura la conexión FTP.

2.4.3 Software Servidor y Cliente FTP

Para llevar a cabo la comunicación FTP y gestión de los archivos y ficheros en un servidor remoto, se necesita la instalación de un software tanto en el servidor como en el cliente. Este software puede ser básico, integrado en el sistema operativo a modo de consola y para cuyo uso existen una serie de comandos que se utilizan en la máquina remota y son muy semejantes a sus equivalentes en Unix. Por ejemplo ***open*** *servidor,* ***delete*** *archivo* o ***rename*** *archivo.* Aparte, existen múltiples programas con interfaz gráfica que permiten la comunicación FTP de una manera más sencilla [35].

Servidor FTP

Un servidor FTP es un programa especial que se ejecuta en un equipo servidor normalmente conectado a Internet (aunque puede estar conectado a otros tipos de redes, LAN, MAN, etc.). Su función es permitir el intercambio de datos entre diferentes servidores/ordenadores. Por lo general, los programas servidores FTP no suelen encontrarse en los ordenadores personales, por lo que un usuario normalmente utilizará el FTP para conectarse remotamente a uno y así intercambiar información con él.

Las aplicaciones más comunes de los servidores FTP suelen ser el alojamiento web, en el que sus clientes utilizan el servicio para subir sus páginas web y sus archivos correspondientes; o como servidor de backup (copia de seguridad) de los archivos importantes que pueda tener una empresa. Para ello, existen protocolos de comunicación FTP para que los datos se transmitan cifrados, como el *SFTP (Secure File Transfer Protocol).*

Cliente FTP

Cuando un navegador no está equipado con la función FTP, o si se quiere cargar archivos en un ordenador remoto, se necesitará utilizar un programa cliente FTP. Un cliente FTP es un programa que se instala en el ordenador del usuario, y que emplea el protocolo FTP para conectarse a un servidor FTP y transferir archivos, ya sea para descargarlos o para subirlos.

Para utilizar un cliente FTP, se necesita conocer el nombre del archivo, el ordenador en que reside (servidor, en el caso de descarga de archivos), el ordenador al que se quiere transferir el archivo (en caso de querer subirlo nosotros al servidor), y la carpeta en la que se encuentra.

Algunos clientes de FTP básicos en modo consola vienen integrados en los sistemas operativos, incluyendo Microsoft Windows, DOS, GNU/Linux y Unix. Sin embargo, hay disponibles clientes con opciones añadidas e interfaz gráfica. Aunque muchos navegadores tienen ya integrado FTP, es más confiable a la hora de conectarse con servidores FTP no anónimos utilizar un programa cliente.

2.4.4 Modos de conexión

A diferencia de HTTP y otros protocolos utilizados en Internet, el protocolo FTP utiliza un mínimo de dos conexiones durante una sesión: una conexión dúplex media para el control y una conexión dúplex completa para la transferencia de datos. De manera predeterminada, se utiliza el puerto TCP 21 del servidor para la conexión de control, pero la conexión de datos queda determinada por el método que el cliente utiliza para establecer conexión con el servidor [36].

Las conexiones de FTP **en modo activo** se denominan a veces "administradas por el cliente" porque el cliente envía un comando **PORT** al servidor a través de la conexión de control. El comando solicita al servidor el establecimiento de una conexión de datos del puerto TCP 20 del servidor al cliente mediante el puerto TCP especificado en el comando PORT, el cual ha sido antes aleatoriamente escogido por el cliente siendo mayor que 1024. Teniendo en cuenta esta aleatoriedad, la máquina cliente siempre tiene que estar dispuesta a recibir conexiones entrantes por puertos mayores al 1024, lo que supone un problema de seguridad, por esto se creó el modo pasivo.

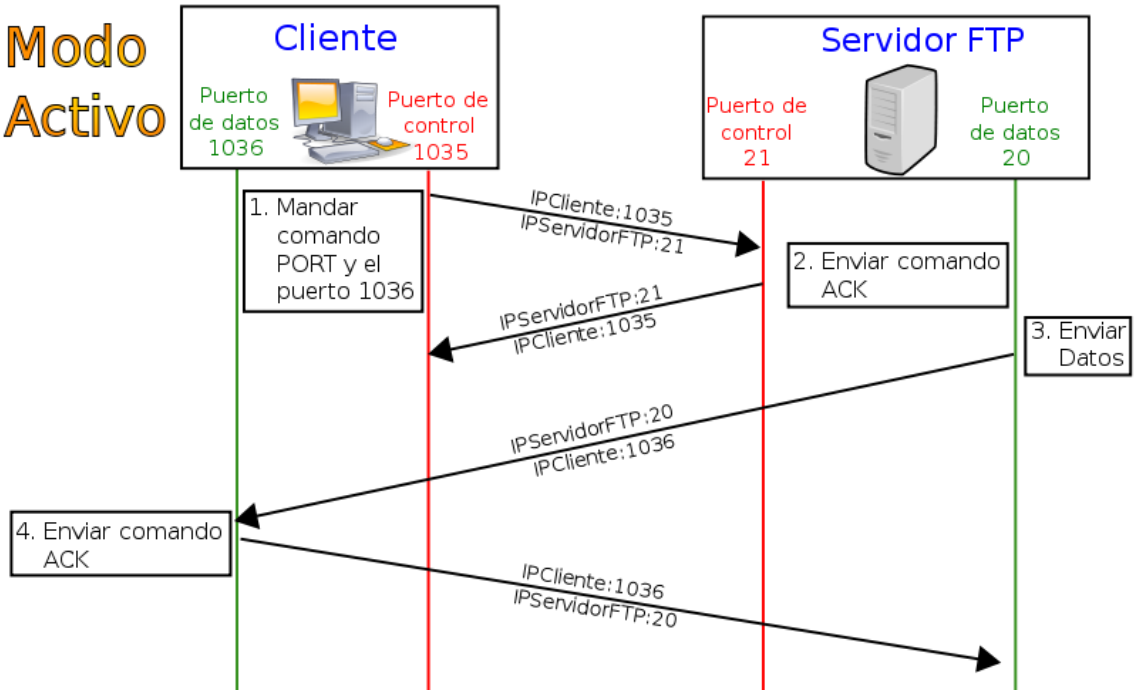


Figura 2.39. Ejemplo de comunicación FTP modo activo

Las conexiones FTP **en modo pasivo** se denominan a veces "administradas por el servidor", porque después de que el cliente emite un comando **PASV**, el servidor responde con uno de los puertos transitorios utilizados como puerto del servidor de la conexión de datos. Después de que el cliente emite un comando de conexión de datos, el servidor se conecta con el cliente utilizando el puerto inmediatamente superior al puerto del cliente de la conexión de control.

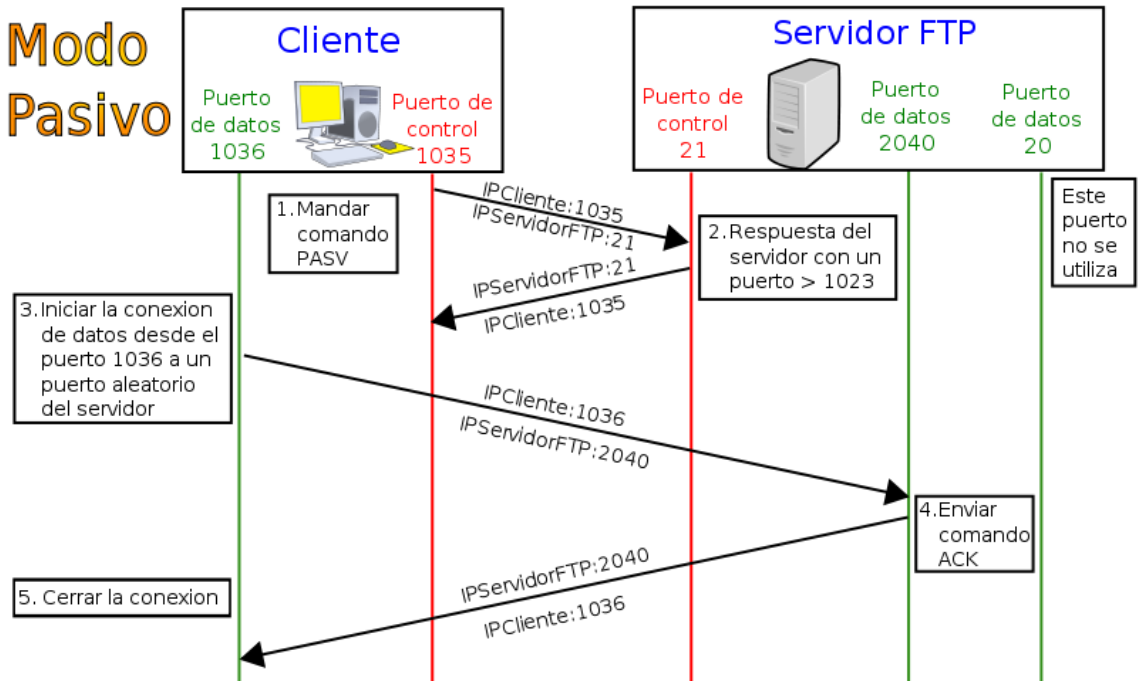


Figura 2.40. Ejemplo de comunicación FTP en modo pasivo

2.5 Tecnologías de desarrollo Web

2.5.1 Introducción

La **Web**, también conocida como **World Wide Web** (www o w3) es una idea que se construyo sobre Internet. Las conexiones físicas son sobre Internet, pero introduce una serie de ideas nuevas, heredando las ya existentes. Antes de la Web, la manera de obtener los datos por la Internet era caótica, había un sinfín de maneras posibles y con ello había que conocer múltiples programas y sistemas operativos.

La Web introduce un concepto fundamental: la posibilidad de lectura universal, que consiste en que una vez que la información esté disponible, se pueda acceder a ella desde cualquier ordenador, desde cualquier país, por cualquier persona autorizada, usando un único y simple programa. Para que esto fuese posible, se utilizan una serie de conceptos, el más conocido es el hipertexto. Con Web los usuarios novatos podrían tener un tremendo poder para hallar y tener acceso a la riqueza de información localizada en sistemas de cómputos en todo el mundo.

Empezó a principios de 1990, en Suiza en el centro de investigación CERN (centro de Estudios para la Investigación Nuclear) y la idea fue de Tim Berners-Lee, que se gestó observando una libreta que él usaba para añadir y mantener referencias de cómo funcionaban los ordenadores en el CERN. Este solo hecho llevó un avance tremendo de Internet, un ímpetu tan grande que en 1993 World Wide Web creció un sorprendente 341000%, hasta nuestros días que tiene un alcance inimaginable.

La World Wide Web consiste en ofrecer una interface simple y consistente para acceder a la inmensidad de los recursos de Internet. Es la forma más moderna de ofrecer información, el medio más potente. La información se ofrece en forma de páginas electrónicas.

Las tecnologías Web sirven para acceder a los recursos de conocimiento disponibles en Internet o en las intranets utilizando un navegador. Están muy extendidas por muchas razones: facilitan el desarrollo de sistemas de Gestión del Conocimiento (en lo adelante GC), su flexibilidad en términos de escalabilidad, es decir, a la hora de expandir el sistema; su sencillez de uso y que imitan la forma de relacionarse de las personas, al poner a disposición de todos el conocimiento de los demás, por encima de jerarquías, barreras formales u otras cuestiones. Estas tecnologías pueden llegar a proporcionar recursos estratégicos, pero, evidentemente, no por la tecnología en sí misma, que está disponible ampliamente, sino por lo fácil que es personalizarla y construir con ella sistemas de GC propietarios de la empresa. Internet o intranet permiten a los usuarios el acceso a una gran cantidad de información: leer publicaciones periódicas, buscar referencias en bibliotecas, realizar paseos virtuales por museos, compras electrónicas y otras muchas funciones. Gracias a la forma en que está organizada la Web, los usuarios pueden saltar de un recurso a otro con facilidad. Hoy en día, la mayoría de las aplicaciones están en red. Por esto, hay un gran desarrollo de proyectos, aplicaciones y servicios basados en tecnologías web para diferentes ámbitos.

Actualmente, existen varias tecnologías web que permiten obtener resultados parecidos a la hora de realizar un proyecto web. Sin embargo, es interesante conocer con cierto nivel de detalle las características principales de cada una de ellas para elegir siempre la opción más conveniente para un proyecto. Seguramente con todas ellas podemos llegar a obtener un resultado parecido pero, aun así, algunas de estas tecnologías pueden ofrecer facilidades, tanto al usuario final como al desarrollador, que faciliten su uso para determinados proyectos.

¿Cómo elegir una tecnología? Los aspectos más importantes a la hora de elegir una u otra tecnología son el número de usuarios y la cantidad de información que manejaremos. Además, otros aspectos que pueden condicionar nuestra elección son, por ejemplo, el presupuesto del proyecto (hay tecnologías que son más caras que otras simplemente por el tipo de servidor o las licencias a utilizar), o también si ya disponemos de un servidor, en cuyo caso debemos saber que tecnologías soporta.

A continuación se enumeran varias herramientas web de diferentes ámbitos de actualidad:

- **Navegadores Web**: Mozilla Firefox, Google Chrome, Internet Explorer, Konqueror, Netscape Navigator, Opera, Safari, etc.
- **Servidores Web**: Servidor HTTP Apache, ISS, Cherokee, Tomcat, etc.
- **Lenguajes de programación (lado servidor)**: PHP, ASP, Perl, Python, .NET, JSP, etc.
- **Otros lenguajes o herramientas**: XHTML, CSS, JavaScript, Flash, etc.

En los siguientes puntos se describen varias tecnologías de las comentadas anteriormente relevantes en el presente proyecto.

2.5.2 Introducción a PHP

PHP es el acrónimo de *Hipertext Preprocesor*. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación [37].

Un lenguaje del lado del servidor es aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

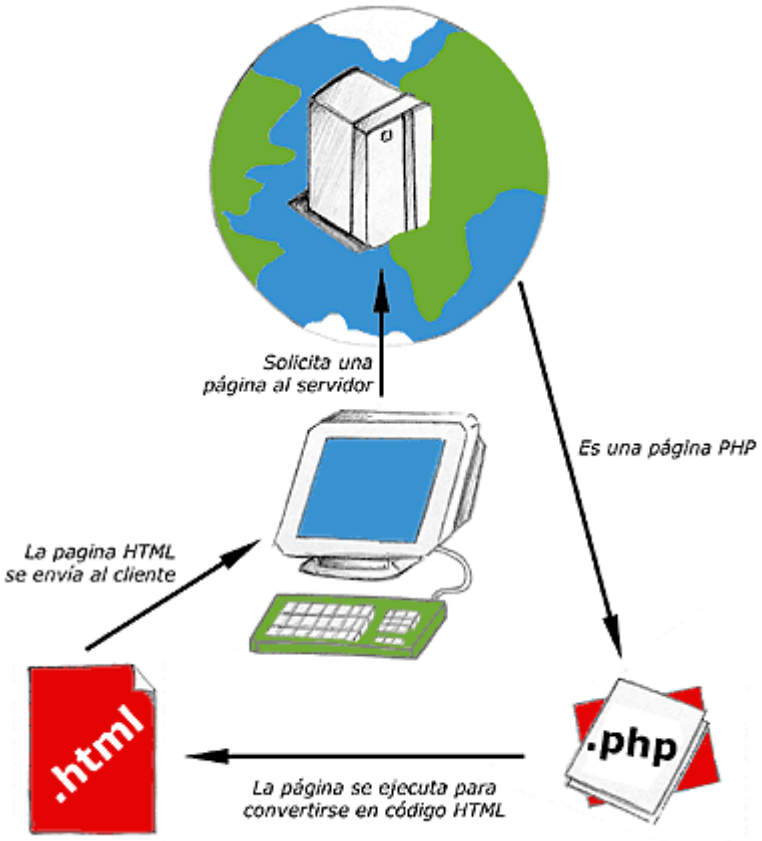


Figura 2.41. Esquema de funcionamiento de las páginas PHP

Una vez que ya conocemos el concepto de lenguaje de programación de scripts del lado del servidor, podemos hablar de PHP. PHP se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar, al igual que ocurre con el popular ASP de Microsoft, pero con algunas ventajas como su gratuidad, independencia de plataforma, rapidez y seguridad. Cualquiera puede descargar a través de la página principal de PHP (www.php.net) y de manera gratuita, un módulo que hace que nuestro servidor web comprenda los scripts realizados en este lenguaje. Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante, ya que permite portar el sitio desarrollado en PHP de un sistema a otro sin prácticamente ningún trabajo.

Por último señalábamos la seguridad, en este punto también es importante el hecho de que en muchas ocasiones PHP se encuentra instalado sobre servidores Unix o Linux, que son de sobra conocidos como más veloces y seguros que el sistema operativo donde se ejecuta ASP. Además, PHP permite configurar el servidor de modo que se permita o rechacen diferentes usos, lo que puede hacer al lenguaje más o menos seguro dependiendo de las necesidades de cada cual.

Fue creado originalmente en 1994 por Rasmus Lerdorf, pero como PHP está desarrollado en política de código abierto, a lo largo de su historia ha tenido muchas contribuciones de otros desarrolladores, actualmente se encuentra en su versión 5.

Este lenguaje de programación está preparado para realizar muchos tipos de aplicaciones web gracias a la extensa librería de funciones con la que está dotado. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por poner dos ejemplos.

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC, por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

2.5.3 Introducción a JavasScript

Javascript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento, no requiere compilación. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado [38].

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de *Mocha*, el cuál fue renombrado posteriormente a *LiveScript*, para finalmente quedar como JavaScript. El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó soporte para la tecnología Java en su navegador web Netscape Navigator en la versión 2.0B3 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha caracterizado por muchos como una estrategia de mercadotecnia de Netscape para obtener prestigio e innovar en lo que eran los nuevos lenguajes de programación web. JavaScript es una marca registrada de Oracle Corporation. Es usada con licencia por los productos creados por Netscape Communications y entidades actuales como la Fundación Mozilla.

Muchos confunden el Javascript con el Java pero ambos lenguajes son diferentes y tienes sus características singulares. Javascript tiene la ventaja de ser incorporado en cualquier página web, puede ser ejecutado sin la necesidad de instalar otro programa para ser visualizado. En cambio Java tiene como principal característica ser un lenguaje independiente de la plataforma. Se puede crear todo tipo de programa que podrá ser ejecutado en cualquier ordenador del mercado y debido a sus características también es muy utilizado para Internet.

Entre las acciones típicas que se pueden realizar en Javascript tenemos dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, Javascript nos permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que podemos crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

Javascript es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas, etc. Además, Javascript pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente.

Con Javascript el programador, que se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página cuando la está visualizando el cliente.

2.5.4 Introducción a HTML

HTML es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web.

El HTML es un lenguaje de marcación de elementos para la creación de documentos hipertexto, muy fácil de aprender, lo que permite que cualquier persona, aunque no haya programado nunca, pueda enfrentarse a la tarea de crear una web. HTML es fácil y se puede dominar fácilmente. Se puede incorporar otro tipo de lenguajes para definir el formato con el que se tienen que presentar las webs, como *CSS*, que proporciona el estilo a la web.

El HTML se creó en un principio con objetivos divulgativos de información con texto y algunas imágenes. No se pensó que llegara a ser utilizado para crear área de ocio y consulta con carácter multimedia (lo que es actualmente la web), de modo que, el HTML se creó sin dar respuesta a todos los posibles usos que se le iba a dar y a todos los colectivos de gente que lo utilizarían en un futuro. Sin embargo, pese a esta deficiente planificación, si que se han ido incorporando modificaciones con el tiempo, estos son los estándares del HTML [39].

El HTML fue desarrollado originalmente por Tim Bernes-Lee en el CERN (Centro Europeo de Física de Partículas), y fue popularizado por el navegador Mosaic desarrollado por la NCSA (National Center for Supercomputing Applications). Pero debido al rápido crecimiento de la web, surgió la necesidad de crear un estándar para que tanto los autores como los navegadores pudieran reconocer cualquier versión de HTML [40]:

- **HTML 1.0 - 2.0 (**1989 – 1991): estos son los primeros pasos del HTML, las páginas no eran muy vistosas pero tenían hipertexto. Estas versiones no fueron oficiales.
- **HTML 3.0 – 3.1 – 3.2** (1995): en esta época Netscape y Microsoft competían por tener un navegador con más funciones (y así ganar mercado), hasta inventaban sus propias etiquetas HTML. Esto perjudicaba al desarrollador web que tenía que estar a tanto de los dos navegadores. Esta versión incluía nuevas funciones como tablas y texto alrededor de figuras

- **HTML 4** (1998): por fin terminó la guerra de los navegadores y llegó al rescate el World Wide Web Consortium (W3) creando una sola versión de HTML. La idea era separar la estructura y presentación de las páginas web en 2 lenguajes. HTML 4 para estructura y CSS para presentación, y convencer a las compañías que creaban navegadores que era necesario adoptar esos estándares. Aún así te tuvieron que hacer algunos cambios en versiones posteriores.

- **HTML 4.01** (1999): se podía escribir tranquilamente un sólo código estando seguro que la mayoría de navegadores lo interpretaría bien.

- **XHTML 1.0** (2000): las cosas cambiaron. HTML y otro lenguaje de marcado conocido como XML se juntaron y nació el XHTML 1.0. Combina la popularidad y la capacidad de verse correctamente en todos los navegadores del HTML con la capacidad de extensión del XML.

- **XHTML 1.1** (2001): es similar a la anterior pero parte a la especificación en módulos.

- **HTML 5:** es la versión más actual de HTML que cada vez va siendo más utilizada. A continuación se explica con mayor detalle.

HTML 5

HTML 5 no es simplemente una nueva versión del lenguaje de marcación HTML, sino una agrupación de diversas especificaciones concernientes a el desarrollo web. Es decir, HTML 5 no se limita sólo a crear nuevas etiquetas, atributos y eliminar aquellas marcas que están en desuso o se utilizan inadecuadamente, sino que va mucho más allá.

Así pues, HTML 5 es una nueva versión de diversas especificaciones, entre las que se encuentran: HTML 4, XHTML 1, DOM Nivel 2 (DOM = Document Objetc Model).

A la par, HTML 5 pretende proporcionar una plataforma con la que desarrollar aplicaciones web más parecidas a las aplicaciones de escritorio, donde su ejecución dentro de un navegador no implique falta de recursos o facilidades para resolver las necesidades reales de los desarrolladores. Para ello se están creando unas APIs que permitan trabajar con cualquiera de los elementos de la página y realizar acciones que hasta hoy era necesario realizar por medio de tecnologías accesorias.

Estas API, que tendrán que ser implementadas por los distintos navegadores del mercado, se están documentando con minuciosidad, para que todos los navegadores, creados por cualquier compañía, las soporten tal cual se han diseñado. Esto se hace con la intención que no ocurra lo que viene sucediendo en el pasado, que cada navegador hace la guerra por su parte y los que acaban pagándolo son los desarrolladores y a la postre los usuarios, que tienen muchas posibilidades de acceder a webs que no son compatibles con su navegador preferido.

Durante este año 2012 debería ser cuando despegara totalmente HTML aunque muchos navegadores ya son compatibles con sus novedades entre las que destacan [41]:

- **Estructura del cuerpo**: La mayoría de las webs tienen un formato común, formado por elementos como cabecera, pie, navegadores, etc. HTML 5 permite agrupar todas estas partes de una web en nuevas etiquetas que representarán cada uno de las partes típicas de una página.

- **Etiquetas para contenido específico**: Hasta ahora se utilizaba una única etiqueta para incorporar diversos tipos de contenido enriquecido, como animaciones Flash o vídeo. Ahora se utilizarán etiquetas específicas para cada tipo de contenido en particular, como audio, vídeo, etc.

- **Canvas**: es un nuevo componente que permitirá dibujar, por medio de las funciones de un API, en la página todo tipo de formas, que podrán estar animadas y responder a interacción del usuario. Es algo así como las posibilidades que nos ofrece Flash, pero dentro de la especificación del HTML y sin la necesidad de tener instalado ningún plugin.

- **Bases de datos locales**: el navegador permitirá el uso de una base de datos local, con la que se podrá trabajar en una página web por medio del cliente y a través de un API. Es algo así como las Cookies, pero pensadas para almacenar grandes cantidades de información, lo que permitirá la creación de aplicaciones web que funcionen sin necesidad de estar conectados a Internet.

- **Web Workers**: son procesos que requieren bastante tiempo de procesamiento por parte del navegador, pero que se podrán realizar en un segundo plano, para que el usuario no tenga que esperar que se terminen para empezar a usar la página. Para ello se dispondrá también de un API para el trabajo con los Web Workers.

- **Aplicaciones web Offline**: Existirá otro API para el trabajo con aplicaciones web, que se podrán desarrollar de modo que funcionen también en local y sin estar conectados a Internet.

- **Geolocalización**: Las páginas web se podrán localizar geográficamente por medio de un API que permita la Geolocalización.

- **Nuevas APIs para interfaz de usuario**: temas tan utilizados como el "drag & drop" (arrastrar y soltar) en las interfaces de usuario de los programas convencionales, serán incorporadas al HTML 5 por medio de un API.

- **Fin de las etiquetas de presentación**: todas las etiquetas que tienen que ver con la presentación del documento, es decir, que modifican

estilos de la página, serán eliminadas. La responsabilidad de definir el aspecto de una web correrá a cargo únicamente de CSS.

2.5.5 Introducción a CSS

El acrónimo de CSS es *Hojas de Estilo en Cascada (Cascading Style Sheets)*, se trata de un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser presentada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

CSS se ha creado en varios niveles y perfiles. Cada nivel se construye sobre el anterior, generalmente añadiendo funciones al previo. Los perfiles son, generalmente, parte de uno o varios niveles de de CSS definidos para un dispositivo o interfaz particular. Actualmente, pueden usarse perfiles para dispositivos móviles, impresoras o televisiones. A continuación se muestran las distintas especificaciones [42]:

- **CSS 1** (1996) - primera especificación oficial de CSS. Algunas de las funcionalidades que ofrece son:
  - Propiedades de las fuentes, como tipo, tamaño, énfasis...
  - Color de texto, fondos, bordes u otros elementos
  - Atributos del texto, como espaciado entre palabras, letras, líneas, etcétera
  - Alineación de textos, imágenes, tablas u otros
  - Propiedades de caja, como margen, borde, relleno o espaciado
  - Propiedades de identificación y presentación de listas
- **CSS 2** (1998) - Como ampliación de CSS1, se ofrecieron, entre otras:
  - Las funcionalidades propias de las capas ( ) como de posicionamiento relativo/absoluto/fijo, niveles (z-index), etcétera
  - El concepto de "media types"
  - Soporte para las hojas de estilo auditivas
  - Texto bidireccional, sombras, etcétera

- **CSS 2.1** (Junio 2011): es la primera revisión de CSS2 que corrige algunos errores encontrados en CSS2, elimina funcionalidades poco soportadas o inoperables en los navegadores y añade alguna nueva especificación.

- **CSS 3** (parcialmente, finales de 2011): en diferencia a CSS2, que fue una gran especificación que definía varias funcionalidades, CSS3 está dividida en varios documentos separados, llamados módulos. Cada módulo añade nuevas funcionalidades a las definidas en CSS2, de manera que se preservan las anteriores para mantener la compatibilidad. Los trabajos en el CSS3, comenzaron a la vez que se publicó la recomendación oficial de CSS2, y los primeros borradores de CSS3 fueron liberados en junio de 1999. Debido a la modularización del CSS3, diferentes módulos pueden encontrarse en diferentes estadios de su desarrollo, de forma que a fechas de noviembre de 2011, hay alrededor de cincuenta módulos publicados, tres de ellos se convirtieron en recomendaciones oficiales de la W3C en 2011: Selectores, Espacios de nombres y Color.

Capítulo 3

Descripción de la aplicación

Para la compresión de un proyecto siempre es necesario poner en situación al lector y presentarle una visión general del mismo. De esta manera comienza el presente capítulo, con una descripción del proyecto general mostrando el entorno en el que se va a llevar a cabo, y continúa mostrando más detalles de bajo nivel.

Se describirá la arquitectura del proyecto dando a conocer los medios utilizados para el desarrollo.

3.1 Escenario de la aplicación

Antes de profundizar sobre cómo se han llevado a cabo las aplicaciones que componen el proyecto, conviene hacer un pequeño resumen para tener una visión general.

El ámbito en el que se va a dar uso a la aplicación es una empresa de transportistas. Cada día los conductores tienen una ruta diferente que llevar a cabo, por lo que necesitan una manera sencilla de informase sobre el destino y el itinerario a seguir, con las posibles paradas intermedias u otras indicaciones.

Para dar solución al caso planteado, se pensó en el envío de unos archivos (imágenes) con las indicaciones y un pequeño mapa orientativo por medio de la tecnología Bluetooth. Pero debido a la fase de emparejamiento previa necesaria en Bluetooth o la petición de la ruta que requiere mayor interacción entre el usuario y la máquina, la tarea perdía automatismo, así que aquí es donde entra en juego NFC. Como se ha visto en el capítulo dedicado a NFC, esta tecnología puede entrar en funcionamiento simplemente con un toque. En nuestro caso, gracias a ese toque se podrá realizar automáticamente la conexión Bluetooth, es una pasarela ahorrando los pasos previos.

El envío de las imágenes se hace desde un ordenador que hace la función de servidor Bluetooth. Cuando recibe las peticiones de los móviles de los conductores, se conecta a su vez a un servidor de almacenamiento remoto y obtiene la imagen de la ruta adecuada a cada usuario. A continuación se la envía al usuario por Bluetooth.

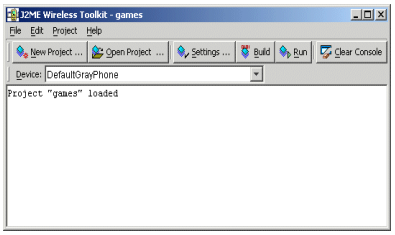
Lo anterior es la funcionalidad principal del proyecto, pero como anexo, se ha creado una aplicación Web que ayuda a la gestión y creación de las imágenes que recibirá cada conductor. Esta aplicación no influye en el funcionamiento de lo comentado hasta ahora y será utilizada por el personal administrativo de la empresa de transportes.

3.2 Arquitectura de la aplicación

3.2.1 Recursos utilizados

Para la realización del proyecto han sido necesarios varios componentes hardware y software que han permitido llevar a cabo el desarrollo y las pruebas.

Recursos software

Recursos software		
Recurso	Descripción	Imagen
IDE Eclipse	Es la plataforma donde se ha desarrollado parte del proyecto, concretamente lo relacionado con Java.	
Editor de texto Notepad ++	Todo lo que no se ha llevado a cabo en Eclipse se ha editado aquí, concretamente lo relacionado con el desarrollo del código para la aplicación Web.	
Wireless Toolkit	Herramienta utilizada para la compilación, verificación y simulación en emuladores de dispositivos móviles. No es para edición de código. Fue necesario integrar el emulador del Nokia 6131 NFC con el que se ha realizado el proyecto.	
JDK Java	> JDK Java SE: contiene todas las librerías y herramientas necesarias para el desarrollo de aplicaciones Java. > Librería Bluecove: Se trata de una librería externa para Java pasada en la especificación de JSR-82 Bluetooth. Es necesario porque el paquete de Java estándar no incluye librerías Bluetooth como la versión Micro Edition.	 
SDK Nokia 6131 NFC 1.1	Contiene las librerías necesarias Java para la ejecución de NFC, múltiples ejemplos, el emulador del móvil, etc.	

Herramientas de Nokia: Nokia PC Suite y Nokia Connectivity Framework	Software de Nokia para interactuar con sus dispositivos conectados al ordenador y para emular conectividad con el framework	
Herramientas Mozilla: Navegador Firefox y cliente FTP filezilla	Utilizado para la visualización de la aplicación Web y para el acceso de una manera más sencilla al servidor real donde se almacenan las imágenes de las rutas.	 
VMWare (sólo si no trabajas con Windows XP)	Algunos programas o software mencionados anteriormente solo son compatible con Windows XP por lo que se hace necesario trabajar con máquina virtual si no se tiene.	

Tabla 3.1. Recursos Software

Recursos hardware

Recursos hardware		
Recurso	Descripción	Imagen
Ordenador	Se utilizará para todo el desarrollo, tiene que contener el software descrito en el punto anterior. También será herramienta para la simulación y como servidor Bluetooth.	
Bluetooth	Se necesita una antena Bluetooth USB o que venga integrado en el ordenador	
Un móvil Nokia 6131	Contiene NFC y Bluetooth. Se utilizará para la ejecución de los MIDlets.	
Tag NFC	Almacena el identificador Bluetooth. Podría ser cualquier Tag NFC que siguiera la norma.	

Tabla 3.2. Recursos Hardware

3.2.2 Definición de las aplicaciones

Anteriormente se ha definido el escenario de la aplicación, ahora se va a mostrar con más detalle los elementos que lo componen, los roles de cada uno y las comunicaciones que se crean entre ellos.

Se pueden diferenciar dos partes en todo el desarrollo: la implementación principal de la **funcionalidad NFC-Bluetooth** y la creación de la **aplicación Web** para la gestión de los contenidos. Se compone de las siguientes aplicaciones:

- **NFC-Bluetooth**: se compone de tres aplicaciones desarrolladas en el lenguaje de programación Java.

  - ***WriterNFC:*** aplicación desarrollada en Java ME para el dispositivo móvil. Es un MIDlet cuya misión es escribir en el tag NFC el identificador Bluetooth. El tag NFC se encuentra en el interior de un puesto de información habilitado para los trabajadores, es el mismo concepto que los poster NFC descritos en el estado del arte.

    Para ello, se abre el MIDlet WriterNFC en el teléfono (Nokia 6131) y pide que se inserte mediante el teclado el texto a grabar en el tag, a continuación se acerca a la tarjeta y se almacena la información en ella (se trata de un mensaje NDEF). Esta aplicación sólo se utilizará cuando el identificador Bluetooth del servidor Bluetooth cambie por alguna razón y haya que actualizar los datos en el tag NFC.

  - ***ReaderNFCBluetooth*****:** aplicación desarrollada en Java ME para el dispositivo móvil. Es un MIDlet con dos funciones.

    La primera es leer la información contenida en el tag NFC, es decir el identificador Bluetooth. Una idea para la total automatización sería que el MIDlet se ejecutara solo al detectar la presencia del tag NFC mediante Push Registry, pero está implementado para que sea el usuario el que abra la aplicación en su móvil. Una vez hecho esto ocurre lo mismo que en el caso de la escritura, el móvil se sitúa cercano al tag y comienza la transmisión vía NFC de la información, se transmite el mensaje NDEF con los datos, que en este caso almacena simplemente texto.

    Una vez que el móvil tiene el identificador comienza la ejecución de la segunda función, la comunicación Bluetooth con el dispositivo cuyo identificador es el obtenido. Cuando el móvil conecta con el servidor automáticamente le está realizando la petición de la imagen con la ruta de ese día. En este punto el móvil queda a la espera del envío de la imagen por parte del servidor.

    El envío de la imagen se realiza mediante el protocolo RFCOMM. Este protocolo provee múltiples emulaciones de los puertos serie RS-232 entre dos dispositivos Bluetooth. Las direcciones Bluetooth

de los dos puntos definen una sesión RFCOMM y una sesión puede tener más de una conexión, el número de conexiones dependerá de la implementación pero en nuestro caso solo se permite una simultánea. Una aplicación que ofrece un servicio basado en el perfil puerto serie (SPP) se trata de un servidor SPP, lo que en nuestro caso es el servidor Bluetooth; por consiguiente, la aplicación readerNFCBluetooth es el cliente SPP: la conexión entre el cliente y el servidor se realiza mediante una url de conexión, se trata de un string con el siguiente formato: *btspp://dirección:identificador_canal* (por ejemplo, btspp://001A891791F6:3). La dirección es el identificador que obtenemos del tag NFC y el canal se obtiene en la búsqueda de los servicios que proporciona el servidor SPP (en la explicación del servidor Bluetooth se verá más información).

Este protocolo se utiliza para la transmisión de pequeñas cantidades de datos como texto por lo que inicialmente se pensó y preparó para que la comunicación fuera mediante OBEX, ya que es más adecuado para el envío de archivos de mayor tamaño como es una imagen. Sin embargo, el sistema operativo del Nokia 6131 es *Symbian serie 40 3rd Edition,* en cuyas características no incluye el soporte OBEX (fue a partir de la Edición 5). Se realizaron pruebas con otro dispositivo Nokia que si era compatible con OBEX pero sin NFC y se comprobó que la transmisión de la imagen era mucho más rápida. Aún así la transmisión mediante RFCOMM es posible y se llega a realizar perfectamente aunque en un tiempo mayor.

Finalmente el móvil recibe la imagen que es almacenada en una carpeta creada especialmente y si aún no existe el MIDlet se encarga de crearlo. A parte, la ruta se visualiza en pantalla en el momento de su recepción.

- ***BluetoothServer:*** aplicación desarrollada en Java SE con la integración de la librería externa Bluecove [25] para el soporte Bluetooth. Esto es necesario ya que Java Estándar no tiene soporte para Bluetooth en aplicaciones de escritorio, que no sean para dispositivos móviles. La aplicación se va a ejecutar en un ordenador con Bluetooth integrado que estará lo suficientemente cercano al puesto NFC, donde estará el usuario con su móvil en el momento de la petición de la ruta.

  La función de esta aplicación es la de servidor Bluetooth (Bt). Como se ha comentado en la aplicación anterior, cliente y servidor residen en los extremos de una sesión RFCOMM. El servidor SPP (BluetoothServer) inicializa y registra sus servicios en el *SDDB (Service Discovery Data Base*), y como parte del proceso de registro, se añade un identificador de canal (channel identifier) al ServiceRecord por la implementación. La forma de la url de

conexión por parte del servidor es la siguiente: *btspp://localhost:uuid;name=nombre* (por ejemplo, btspp://localhost:1101;name=SPPServ). El canal se añade automáticamente en el registro de servicio, el *uuid* define el tipo de servicio y el name simplemente da nombre al servicio.

La aplicación se está ejecutando constantemente, siempre está a la espera o a la escucha de nuevos dispositivos, aunque no realiza ninguna acción hasta que un dispositivo se conecta. Lo que hace al arrancar es lanzar el servicio Bluetooth que ofrece para que los dispositivos interesados puedan encontrarlo.

Cuando recibe la petición por parte de un usuario, se pone en marcha para obtener la ruta que le pertenece. Como referencia utiliza el identificador Bluetooth del móvil del usuario ya que tiene una relación directa con el nombre de la ruta. Las rutas de todos los conductores se encuentran almacenadas en un servidor remoto (se ha utilizado espacio gratuito de un servidor existente en la red, *000webhost.com*), por lo tanto la aplicación se conecta a este servidor para descargarse la ruta específica. Esta conexión y la descarga del fichero se realizan mediante el protocolo FTP. Una vez el servidor Bt tiene la ruta totalmente descargada, procede al envío de la misma mediante Bluetooth al usuario que había quedado a la espera y el servidor continúa con su función.

Como medidas de apoyo, la aplicación tiene un sistema de Logs donde se van almacenando en documentos de texto los pasos y las instrucciones que van sucediendo para así tener un mayor control y un registro de la actividad del servidor Bluetooth. Aparte de esto, la aplicación controla las rutas de almacenamiento de estos Logs y de las rutas de los conductores en directorios clasificados por año, mes y día en el que nos encontremos.

- **Aplicación Web** (*RouteDriver*): aplicación desarrollada en **PHP, Javascript, HTML y CSS**. Para el alojamiento de la aplicación se ha utilizado el mismo espacio comentado anteriormente, 000webhost.com. Se trata de un espacio de hosting gratuito Apache con soporte para php y ftp, por lo que cumple los requisitos para la aplicación Web. Se ha hecho de esta manera para tener mayor realismo en el montaje de todos los componentes del proyecto, trabajando realmente con comunicaciones remotas.

  El objetivo de esta aplicación es gestionar los conductores de la empresa, relacionarlos con sus identificadores Bluetooth en una base de datos xml y crear las rutas para cada uno de ellos de una forma más rápida. Gracias a que se trata de un entorno Web se puede acceder desde cualquier lugar con conexión a internet por lo que no es

necesario encontrarse en el lugar de trabajo para utilizarla. Se llevó a cabo la elección de una pequeña base de datos en un fichero xml debido a que la carga de datos a tratar es pequeña, simplemente se almacenan el nombre del trabajador y el identificador Bluetooth correspondiente, además de tratarse de una empresa pequeña que no tiene muchos empleados.

Respecto a la creación de rutas, la aplicación se divide en tres módulos: en el primero se escriben las indicaciones a seguir, en el segundo se puede adjuntar opcionalmente un mapa de la ruta como apoyo a las indicaciones y finalmente en el tercero se crea la imagen final asociándola al nombre del conductor correspondiente. Por otro lado se puede configurar los conductores, añadiendo nuevos a la base de datos o eliminando los ya existentes.

3.2.3 Esquema de funcionamiento

Las aplicaciones anteriores son ejecutadas por los siguientes componentes hardware:

- **Nokia 6131 NFC**: sobre este móvil se ejecutan en diferentes momentos WriterNFC y ReaderNFCBluetooth.
- **Ordenador**: simula un servidor Bluetooth ejecutando la aplicación BluetoothServer. Descarga las rutas por FTP.
- **Servidor remoto Web/FTP**: aloja la aplicación Web RouteDriver y las rutas de los usuarios. Una persona en posesión de las credenciales puede acceder a la aplicación en cualquier ordenador.

A continuación se muestra visualmente la interconexión que se produce entre los diferentes dispositivos:

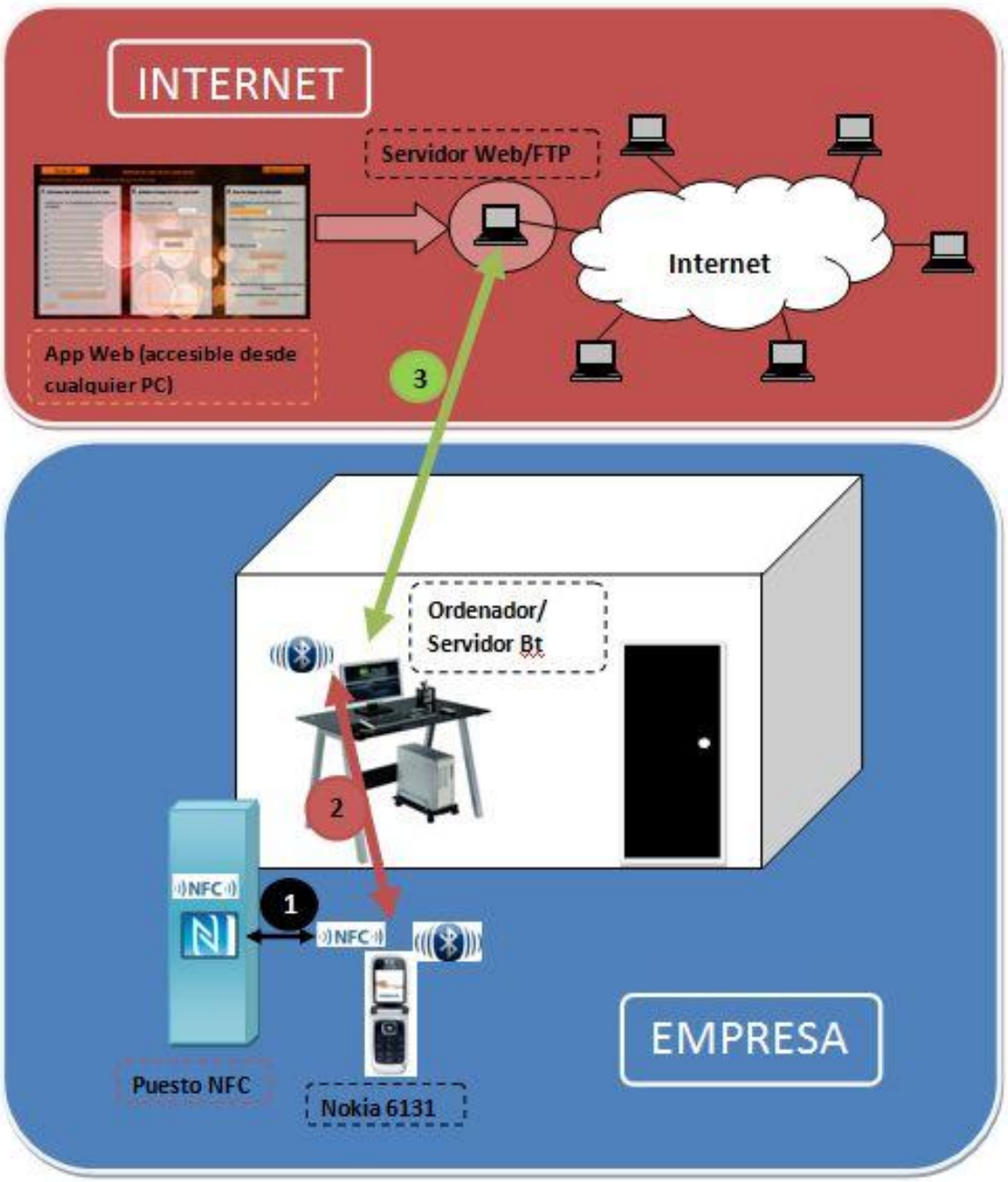


Figura 3.1. Esquema de funcionamiento e interconexión

Las conexiones producidas son:

- El trabajador abre el MIDlet ReaderNFCBluetooth y acerca el *Nokia 6131* al *Puesto NFC,* puede ser con un toque hasta unos 4 cm. El tag pasivo NFC que se encuentra en el puesto se activa y el lector obtiene el mensaje grabado en el tag, el Bluetooth id (**número 1** en la figura). Aquí finaliza la breve pero importante funcionalidad NFC.

- A continuación, el MIDlet continúa ejecutándose estableciendo la conexión con el Servidor Bluetooth y se queda a la espera de recibir su contenido (**número 2**). El servidor recibe la petición, almacena la

dirección Bluetooth del dispositivo conectado en ese momento y a través de la red establece la conexión mediante FTP con el servidor remoto. Busca la imagen de ruta concreta cuyo nombre viene definido por la dirección Bluetooth del usuario y se la descarga a local (**número 3**).

- Para finalizar, se envía al usuario la imagen recién descargada haciendo uso de la misma conexión Bluetooth que se había establecido con anterioridad (**número 2**). El usuario la recibe y da por concluida la conexión.

3.3 Diseño software

A continuación se va a mostrar la estructura de las aplicaciones que componen el proyecto a través de sus diagramas UML de clases.

3.3.1 Diagramas de clases

Módulo WriterNFC

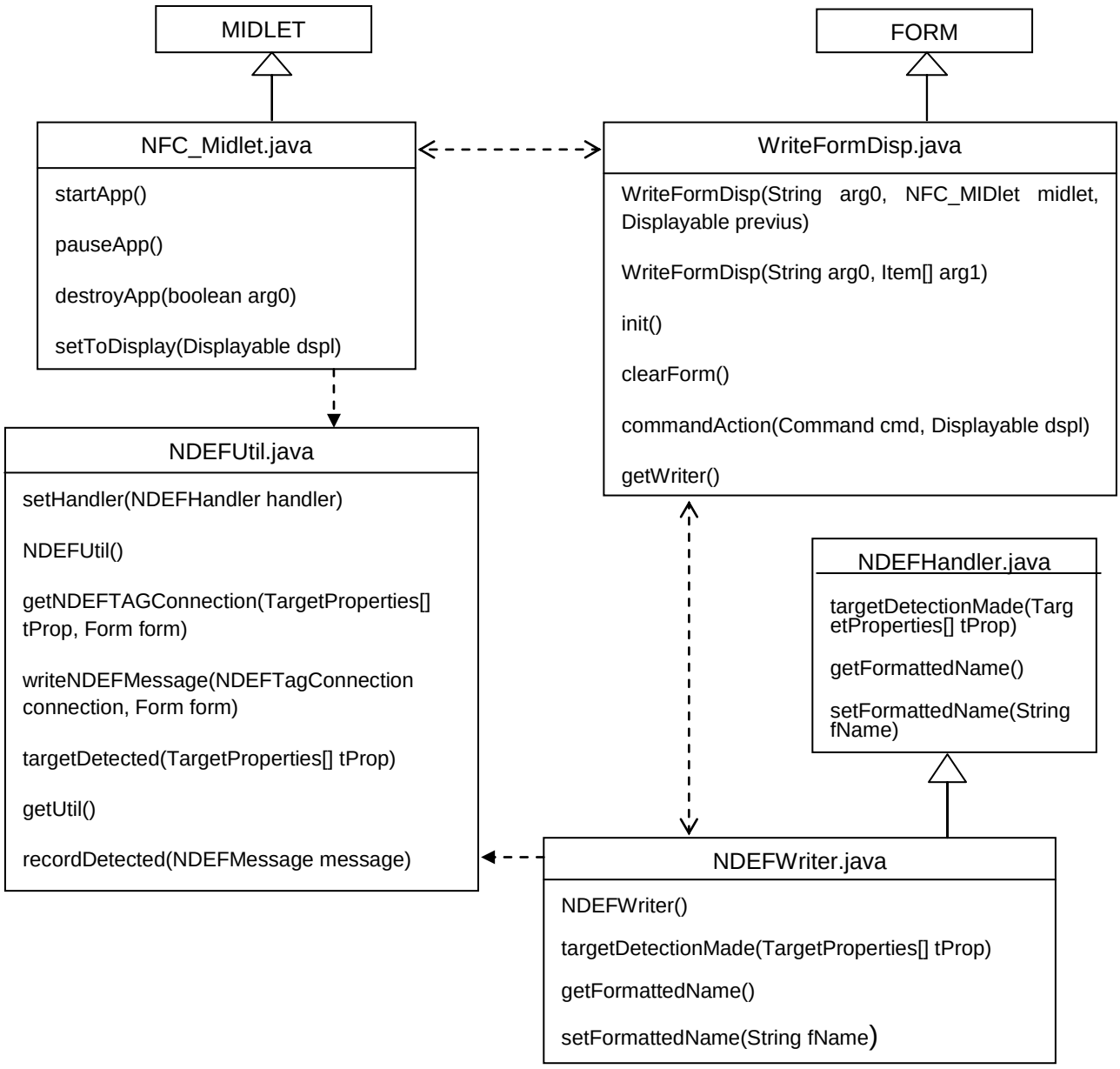


Figura 3.2. Diagrama de clases MIDlet WriterNFC

Módulo ReaderNFCBluetooth

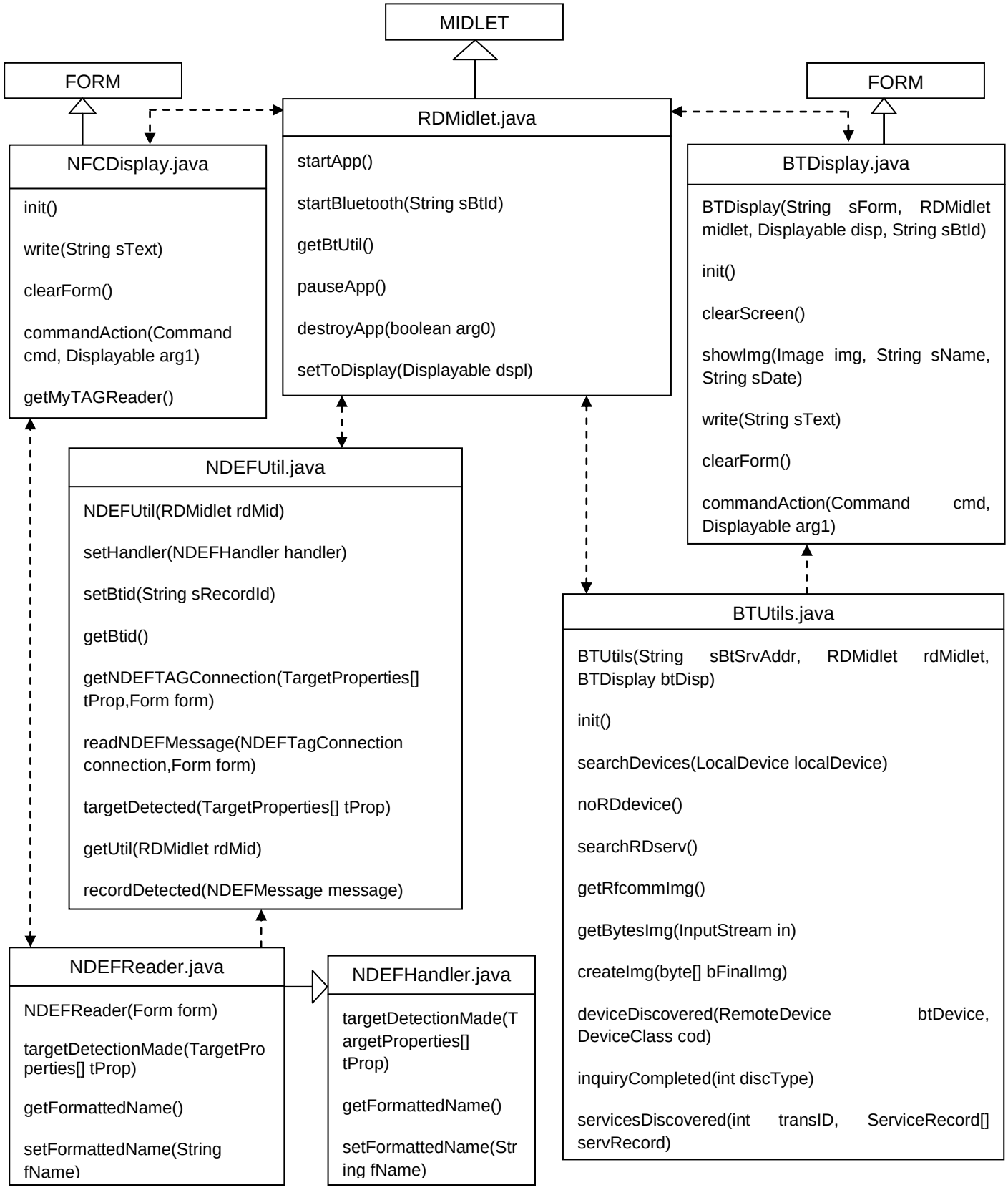


Figura 3.3. Diagrama de clases MIDlet ReaderNFCBluetooth

Módulo BluetoothServer

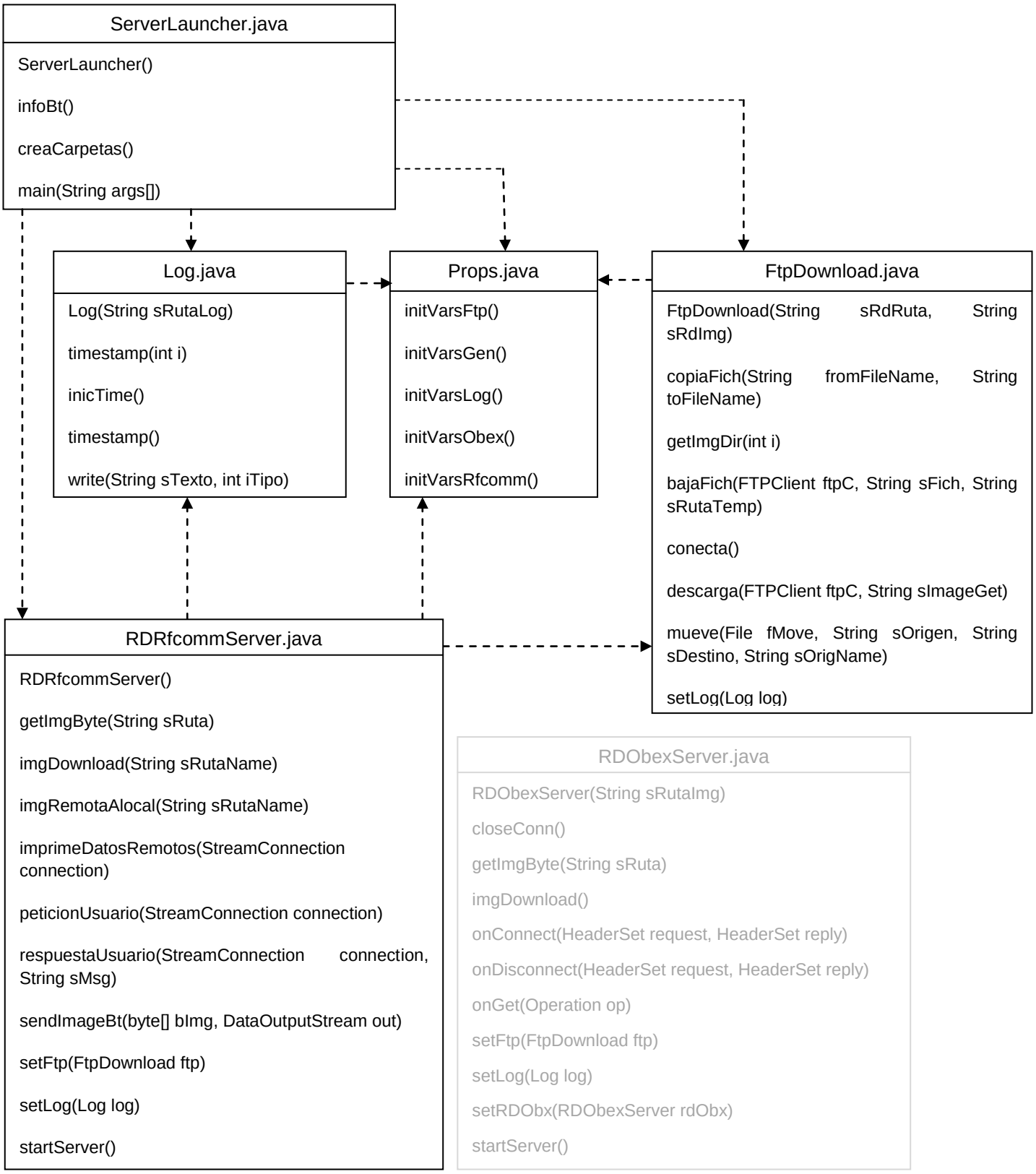


Figura 3.4. Diagrama de clases Java BluetoothServer

Módulo aplicación Web

En el caso de la aplicación Web es diferente a los anteriores. No se ha hecho con programación orientada a objetos, sino que se ha incrustado código php en el documento html o se han referenciado páginas php externas desde el documento. Aún asi, se muestra un diagrama de bloques con la estructura:

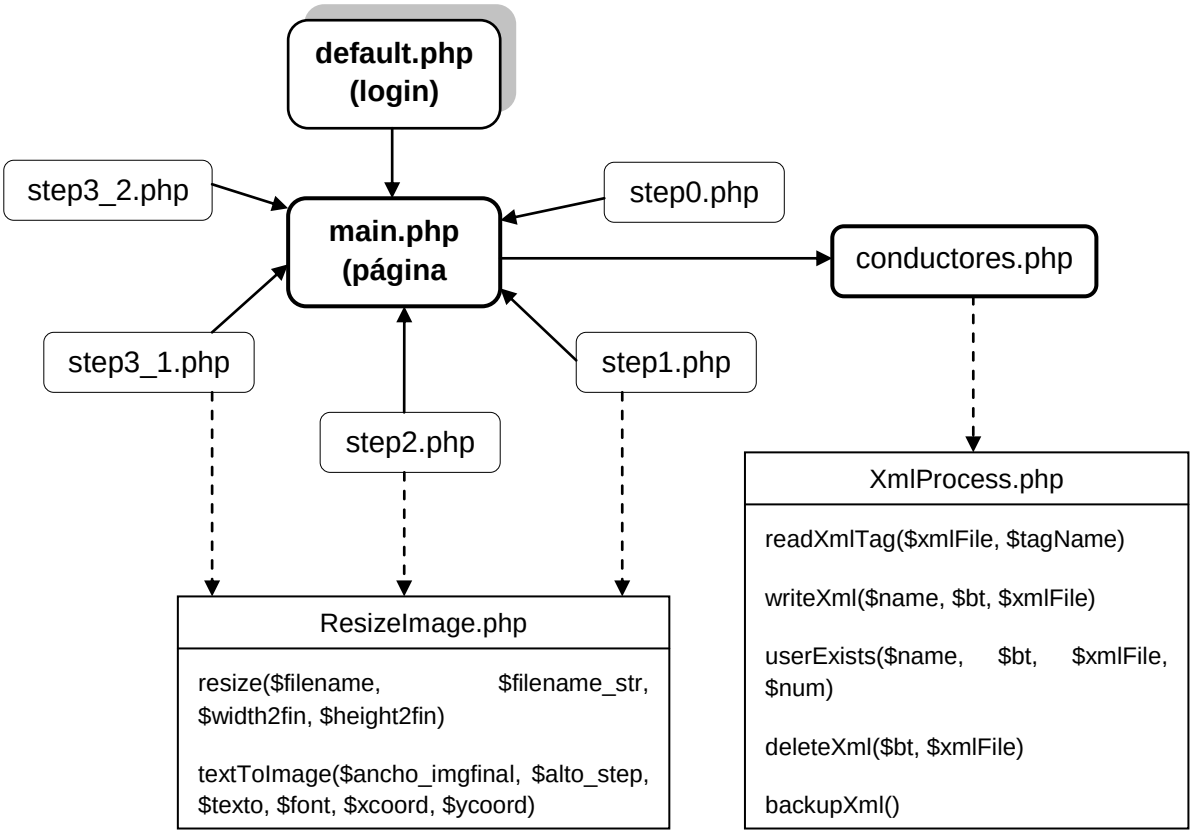


Figura 3.5. Diagrama de bloques aplicación Web

Como se observa en la *figura 3.5*, hay una pequeña parte de la aplicación Web que si hace uso de POO. Hay ciertas funciones que son utilizadas desde varios archivos y se crearón clases con las funciones aparte.

3.3.2 Clases y métodos de los MIDlets

En este punto se va a realizar una descripción detallada de las clases y métodos que las componen, aparte se mostrarán algunos fragmentos de código importantes. Antes de comenzar, cabe destacar como es el funcionamiento de un MIDlet, su ciclo de vida.

Un MIDlet siempre heredará de la clase *javax.microedition.midlet.MIDlet*. Los métodos de esta clase permiten a nuestra aplicación crear, empezar, parar y destruir un MIDlet, son la interfaz del Midlet.

El ciclo de vida de un MIDlet se compone de lo siguientes estados: pausado, activo o destruido. Sólo puede estar en un estado a la vez. Cuando un MIDlet se carga en memoria, inicialmente pasa al estado pausado, entonces se realiza la inicialización de la clase (método *startApp()*). Si el MIDlet lanza una excepción durante la ejecución de su constructor, se destruye.

El MIDlet puede pasar de activo a pausado, cuando, por ejemplo, recibimos una llamada en nuestro móvil; es el sistema quien pasa nuestro MIDlet de activo a pausado y viceversa. Un MIDlet puede ser lanzado y parado todas las veces que queramos, pero sólo puede ser destruido una vez.

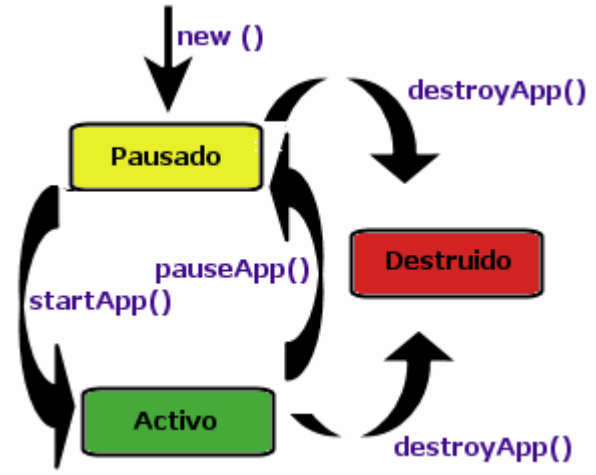


Figura 3.6. Estados del ciclo de vida de un MIDlet

Aplicación WriterNFC

Se trata de la aplicación más simple del proyecto. Es un MIDlet que implemente la funcionalidad NFC de escritura en un tag.

- Clase *NFC_Midlet*: es la clase principal y se encarga de gestionar el MIDlet. Contiene el método *startApp* por lo que es la encargada de ejecutar el arranque además de todos los demás estados del MIDlet. Métodos:
  - *startApp():* primer método que se ejecuta al arrancar la aplicación, lanza el display con las consiguientes impresiones por pantalla y se queda a la espera del tag NFC.
  - *pauseApp:* método para detener el midlet.
  - *destroyApp:* destruye el midlet. Pone las variables a null.
  - *setToDisplay:* establece el nuevo display que se le pasa como argumento.

- Clase *NDEFHandler*: clase abstracta que define una serie de métodos (*targetDetectionMade, getFormattedName, setFormattedName*) que serán implementados en otras clases.

- Clase *NDEFUtil*: se trata de una clase muy importante ya que implementa gran parte de la utilidad NFC con las interfaces *TargetListener* y *NDEFRecordListener.* Realiza las operaciones de detección y escritura.

  - *setHandler:* establece el manejador de la tarjeta cuando es llamado.

  - *getNDEFTAGConnection:* gestiona las propiedades de la tarjeta para encontrar el NDEFTagConnection.

```
public static NDEFTagConnection getNDEFTAGConnection
(TargetProperties[] tProp,Form form)
{
    for (int j = 0; j < tProp.length; j++)
    {
        Class[] connections = tProp[j].getConnectionNames();
        if(connections != null)
        {
            for (int i = 0; i < connections.length; i++)
            {
                if (connections[i].getName().equals(
                "javax.microedition.contactless.ndef.NDEFTagConnection"))
                {
                    try
                    {
                        return (NDEFTagConnection) Connector.open(tProp[0]
                            .getUrl(connections[i]));
                    }
                    catch (Exception e)
                    {
                        form.append(e.toString()+"\n");
                        System.out.println(e.toString() + "\n");
                    }
                }
            }
        }
    }
    return null;
}
```

Figura 3.7. Implementación método de detección de tarjetas

- *writeNDEFMessage:* con este método es posible escribir en el tag.

```
public static void writeNDEFMessage(NDEFTagConnection connection,Form form)
{
    try {
        form.append("***Creando un nuevo record NDEF (BT ID).\n");
        NDEFRecord[] records = new NDEFRecord[1];
        String empty = null;
        NDEFRecord fName = new NDEFRecord(new NDEFRecordType(NDEFRecordType.UNKNOWN,empty),
        new String(FORMATTED_NAME_ID).getBytes(),
        new String(util.handler.getFormattedName()).getBytes());

        form.append("***BT ID creado.\n");
        records[0] = fName;
        NDEFMessage message = new NDEFMessage(records);
        form.append("***Mensaje NDEF creado.Comienza de escritura.\n");
        connection.writeNDEF(message);
        form.append("=== MENSAJE NDEF ESCRITO ===\n");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 3.8. Implementación método de escritura NDEF

- *getUtils:* obtiene la instancia de NDEFUtil con todos los útiles necesarios para gestionar la tarjeta. Añade los listeners *TargetListener* y *NDEFRecordListener*.
- *targetDetected:* detecta las propiedades de la tarjeta.
- *recordDetected:* detecta si hay algún dato grabado en la tarjeta.

- Clase *NDEFWriter:* esta clase implementa los métodos definidos en NDEFHandler.
  - *NDEFWriter:* constructor de la clase que proporciona la instancia del display anterior.
  - *targetDetectionMade:* detecta las propiedades de la tarjeta y permite la conexión con ella.
  - *getFormattedName:* devuelve un vector con todos los datos grabados en la tarjeta.
  - *setFormattedName(String fName):* añade al vector de datos de la tarjeta nueva información.
- Clase *WriteFormDisplay:* se encarga de mostrar por pantalla el contenido y de la funcionalidad de la misma.
  - *WriteFormDisp:* método constructor que inicializa las variables del display anterior y el midlet.
  - *init:* escribe las sentencias que se visualizarán por pantalla y obtiene el objeto de escritura NFC en caso de ser necesario.

- *clearForm:* limpia el formulario, la pantalla.
- *commandAction:* permite asignar acciones a los botones.
- *getWriter:* devuelve el objeto de escritura NFC.

Aplicación ReaderNFCBluetooth

Esta aplicación es un MIDlet que se encarga de la comunicación NFC con el tag, concretamente lee los datos grabados en ella. Esta parte de la funcionalidad es muy similar a la vista en WriterNFC. Por otro lado, cuando ha obtenido la información de la tarjeta, abre una conexión Bluetooth con la dirección obtenida y solicita su ruta.

- Clase *RDMidlet*: se trata de la clase principal del MIDlet que permite la ejecución del mismo. Contiene los métodos del ciclo de vida.
  - *startApp():* primer método que se ejecuta al arrancar la aplicación, lanza el display con las consiguientes impresiones por pantalla y obtiene el lector NFC quedándose a la espera del contacto con tag NFC y la consiguiente información.
  - *startBluetooth:* este método contiene el Bluetooth id y despierta al hilo principal, que estaba a la espera del id, para comenzar la conexión Bluetooth.
  - *getBtUtil:* devuelve la instancia de btUtils.
  - *pauseApp:* método del ciclo de vida del MIDlet que lo detiene.
  - *destroyApp:* método del ciclo de vida del MIDlet que lo destruye y pone las variables a null.
  - *setToDisplay:* establece el nuevo display que se le pasa como argumento.
- Clase *NDEFUtil*: clase muy importante que implementa la funcionalidad NFC con las interfaces *TargetListener* y *NDEFRecordListener.* Realiza las operaciones de detección y lectura.
  - *NDEFUtil:* método constructor que inicializa variables.
  - *setHandler:* establece el manejador de la tarjeta cuando es llamado.
  - *getNDEFTAGConnection:* gestiona las propiedades de la tarjeta para encontrar el NDEFTagConnection.
  - *readNDEFMessage:* con este método se escriben los datos en la tarjeta.
  - *targetDetected:* detecta las propiedades de la tarjeta

- *getUtil:* obtiene la instancia de NDEFUtil con todos los útiles necesarios para gestionar la tarjeta. Añade los listeners *TargetListener* y *NDEFRecordListener*.
- *recordDetected:* detecta si hay algún dato grabado en la tarjeta

- Clase *NDEFHandler*: clase abstracta que define una serie de métodos (*targetDetectionMade, getFormattedName, setFormattedName*) que serán implementados en otras clases.
- Clase *NDEFReader*: esta clase implementa los métodos definidos en NDEFHandler.
  - *NDEFReader:* constructor de la clase que proporciona la instancia del display anterior.
  - *targetDetectionMade:* detecta las propiedades de la tarjeta y permite la conexión con ella.
  - *getFormattedName:* devuelve un vector con todos los datos grabados en la tarjeta.
  - *setFormattedName(String fName):* añade al vector de datos de la tarjeta nueva información.
- Clase *NFCDisplay*: se encarga de mostrar por pantalla el contenido relacionado con la parte NFC y de la funcionalidad de los botones.
  - *NFCDisplay:* método constructor que inicializa las variables display anterior y midlet.
  - *init:* escribe las primeras sentencias que se visualizarán por pantalla, obtiene el objeto de escritura NFC en caso de ser necesario y añade los botones.
  - *write:* imprime por pantalla.
  - *clearForm:* borra el contenido de la pantalla.
  - *commandAction:* añade funcionalidad a los botones.
  - *getMyTAGReader:* devuelve el objeto de lectura NFC.
- Clase *BTUtils*: en esta clase se definen todos los métodos necesarios para la funcionalidad Bluetooth. Desde el descubrimiento de dispositivos hasta la recepción de la imagen en el dispositivo local.
  - *BTUtils:* método constructor que inicializa variables y muestra la información Bluetooth del dispositivo local.

- *init:* comienza la búsqueda de dispositivos Bluetooth, y si encuentra su objetivo, continúa con la búsqueda de servicios.

- *searchDevices:* método que se encarga de la búsqueda de dispositivos. Obtiene el DiscoveryAgent del dispositivo local y comienza la búsqueda.

```
//método que comienza la búsqueda de devices
public void searchDevices(LocalDevice localDevice)
{
    btDisp.write("Buscando devices...");

    //obtiene el discoveryagent
    agent = localDevice.getDiscoveryAgent();

    //Comienza la búsqueda
    try{
        agent.startInquiry(DiscoveryAgent.GIAC, rdMidlet.getBtUtil());
    }catch(Exception e){
        btDisp.write("Excepción en searchDevices(startInquiry): " + e.toString());
        System.err.println("Excepción en searchDevices(startInquiry): " + e.toString());
    }
   //espera a que termine la búsqueda y inquiryCompleted le despierte
    try {
        synchronized(lock){
            lock.wait();
        }
    }
    catch (InterruptedException e) {
        e.printStackTrace();
    }

    btDisp.write("Busqueda de devices completa.");
}
```

Figura 3.9. Implementación del método para la búsqeda de dispositivos Bt

- *noRDdevice:* método utilizado cuando no se ha encontrado el servidor Bluetooth en la búsqueda. Imprime por pantalla los dispositivos resultantes de la búsqueda.

- *searchRDserv:* este método se ejecuta si ha encontrado el dispositivo correcto. Realiza la búsqueda de los servicios disponibles con la consiguiente obtención de la url de conexión. Si la búsqueda es positiva hace la petición de la imagen al servidor.

```
public void searchRDserv()
{
    try{
        //check for spp service
        RemoteDevice remoteDevice = RDremDev;
        UUID[] uuidSet = new UUID[1];
        //*********for RFCOMM***********
        uuidSet[0] = new UUID("1101", false);
        //********for OBEX*******
        //uuidSet[0] = new UUID("1106",false);

        btDisp.write("Searching for service...");
        agent.searchServices(null,uuidSet,remoteDevice,rdMidlet.getBtUtil());

        try {
            synchronized(lock){
                lock.wait();
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        if(connectionURL == null){
            //******for RFCOMM*******
            btDisp.write("Device does not support Simple SPP Service.");
            //*******for OBEX*********
            //btDisp.write("Device does not support OBEX Service.");
            btDisp.write("No se ha encontrado el sevcio.");

            //rdMidlet.cerrarApp();
        } else{
            //esperamos un par de segundos para ver la pantalla
            try{
                Thread.sleep(2000);
            } catch(Exception e){
                e.printStackTrace();
            }
            btDisp.clearScreen();
            //******for RFCOMM*******
            btDisp.write("-- 2. Obtencion imagen por RFCOMM --");
            byte[] bImg = getObexImg();
            Thread.sleep(5000);
            btDisp.clearScreen();
            btDisp.write("-- 3. Creacion imagen en dispositivo --");
            if(bImg == null){
                btDisp.write("ERROR, los bytes dela imagen estan vacios");
            } else{
                createImg(bImg);
            }
        }
    }catch(Exception e){
        btDisp.write("Excepción en searchRDserv: " + e.toString());
        System.err.println("Excepción en searchRDserv: " + e.toString());
    }
}
```

Figura 3.10. Implementación del método de búsqueda de servicios Bt

- *getRfcommImg:* método encargado de la recepción de los bytes de la imagen.

- *getBytesImg:* se encarga de almacenar los bytes recibidos de la imagen en un ByteArrayOutputStream para finalmente incluirlos en un array de bytes.
- *createImg:* este método se encarga de hacer la imagen visible en el sistema del móvil. Crea una ruta local en el dispositivo con un directorio para almacenar la imagen además de mostrarla por pantalla.
- *deviceDiscovered:* método de la interfaz *Device Discovery*. Es llamado cada vez que encuentra un dispositivo, si es el nuestro para la búsqueda para continuar con el proceso siguiente.
- *inquiryCompleted:* método de la interfaz *Device Discovery*. Método llamado cuando la búsqueda de dispositivos a finalizado ya sea porque no hay más, porque se ha cancelado o porque ha surgido algún error.
- *servicesDiscovered:* método de la interfaz *Device Discovery*. Se le llama cuando se descubre algún servicio. Obtiene la url de conexión del service record.
- *serviceSearchCompleted:* método de la interfaz *Device Discovery*. Se le llama cuando ha terminado la búsqueda de servicios ya sea porque se han encontrado todos porque ha sido cancelada, por algún error, porque no se encontraron grabaciones o no se encontró el servicio.

▪ *Clase BTDisplay*: clase encargada de gestionar todo el contenido mostrado por pantalla referente a Bluetooth y la funcionalidad de los botones.

- *BTDisplay:* método constructor que inicializa variables.
- *init:* añade los botones y muestra las primeras sentencias en la pantalla.
- *clearScreen:* elimina todo el contenido de la pantalla.
- *showImg:* muestra la ruta por pantalla al usuario.
- *write:* método de escritura por pantalla.
- *clearForm:* borra todo el contenido del form.
- *commandAction:* añade funcionalidad a los botones.

3.3.3 Clases y métodos de la aplicación de escritorio

La aplicación *BluetoothServer* está desarrollada en Java SE y su función principal es la de ejercer como servidor Bluetooth, más concretamente como servidor

SPP. La aplicación registra un servicio RFCOMM para que sea visible por los posibles clientes que estén interesados en utilizarlo. Posteriormente, recibe la petición de un cliente interesado que solicita su ruta. Mediante una conexión FTP, la aplicación se conecta al servidor remoto, localiza la imagen específica y se la descarga a local para poder enviarla por Bluetooth de nuevo al cliente.

Antes de que el servicio fuera RFCOMM se creó un servicio OBEX, aunque por razones que ya se han comentado se tuvo que cambiar. Por esto en el diagrama de clases de la *figura 3.5* aparece la clase aunque en principio no se haga uso de ella.

- Clase *ServerLauncher*: clase principal del proyecto que contiene el método main. Se encarga de la preparación del entorno creando los directorios básicos de almacenamiento para el contenido en caso de ser necesario y de lanzar el servidor adecuado (RFCOMM u OBEX) según se le pase por parámetro.
  - *infoBt:* establece el dispositivo como visible para poder ser encontrado por otros dispositivos y muestra por pantalla y en el log la información (nombre y dirección Bt).
  - *creaCarpetas:* Si no existen, crea los directorios base donde se almacenarán las imágenes y los ficheros de log.
  - *main:* ejecuta los métodos descritos anteriormente, pone en funcionamiento el servicio Bluetooth correspondiente y establece el valor de diferentes variables de otras clases.
- Clase *RDRfcommServer:* la función principal de esta clase es establecer el servicio Bluetooth para poder ser descubierto. Gestiona las peticiones y las respuestas de los usuarios.
  - *setLog:* inicializa la variable log.
  - *setFtp:* inicializa la variable ftp.
  - *startServer:* crea la url de conexión para el servicio obteniendo varios de sus parámetros del fichero *properties.* Lanza el servicio y se queda a la espera de conexiones entrantes. Desde este método también se gestionan las peticiones y las descargas de imágenes a través de otras funciones.

```
//open server url
StreamConnectionNotifier streamConnNotifier =
    (StreamConnectionNotifier)Connector.open(sConnURL);
StreamConnection connection = streamConnNotifier.acceptAndOpen();
```

Figura 3.11. Registro del servicio SPP

- *imprimeDatosRemotos:* imprime por pantalla y en log los datos del dispositivo remoto que ha realizado la petición.
- *peticionUsuario:* maneja la petición del usuario abriendo el stream que envía y leyendo los datos.
- *respuestaUsuario:* método encargado de dar respuesta a la petición del usuario. Envía un stream con información sobre el resultado de la descarga de su ruta.
- *imgremotaAlocal:* este método se encarga de controlar si la descarga de la imagen ha sido correcta. Si es así, devuelve los bytes descargados.
- *imgDownload:* conecta con la clase FtpDownload para gestionar la conexión y descarga de la ruta. Devuelve la ruta en la que se ha almacenado la imagen en local.
- *getImgByte:* obtiene los bytes de la imagen descargada para prepararla para su envío Bt.
- *sendImgBt:* envía los bytes de la imagen por Bluetooth como un stream.

▪ Clase *FtpDownload:* esta clase contiene todos los métodos necesarios para la comunicación con el servidor remoto mediante FTP. Se encarga de la conexión mediante credenciales, la descarga, cambio de directorio del fichero, etc. Los métodos necesarios para la conexión FTP con el servidor Apache se encuentran en una librería externa, *commons-net*.

- *FtpDownload:* método constructor que inicializa variables.
- *setLog:* incializa la variable log.
- *conecta:* obtiene los datos de acceso como dirección del servidor y usuario y el login del fichero properties

```
String sDomain = p.DOMAIN;
String sDirip = p.IPFTP;
String sLogin = p.USER;
String sPasswd = p.PASSWORD;

log.write("Conectando con ftp " + sDomain +
        " (" + sDirip + "), user: " + sLogin + " y password: " + sPasswd + " ...", 2);
FTPClient ftpC = new FTPClient();
try{
    //intenta la conexion con el user y el password
    ftpC.connect(sDomain);
    ftpC.login(sLogin, sPasswd);

    //comprueba si el resultado de la conexion ha sido positivo
    int reply = ftpC.getReplyCode();
    if(!FTPReply.isPositiveCompletion(reply)) {
       ftpC.disconnect();
       log.write("ERROR: FTP no conecta", 0);
       return null;
    }

    log.write("Conexión OK, conectado con " + sDomain, 2);
}catch(IOException e) {
    System.out.println("error excepcion");
    throw new IOException();
}
```

Figura 3.12. Implementación de un fragmento de método conecta

- *descarga:* busca el archivo concreto en el ftp y si lo encuentra, lo descarga a un directorio local temporal. Cuando termina lo pasa al directorio definitivo local. Al finalizar cierra la sesión FTP con el servidor. El método devuelve el directorio final donde está el fichero.
- *bajaFich:* realiza la acción de descargar el fichero del método anterior.
- *mueve:* mueve los ficheros entre directorios.
- *copiaFich:* copia ficheros entre directorios.
- *getImgDir:* devuelve la información de la fecha actual para la creación de las carpetas.

- Clase *Log:* esta clase define métodos para la gestión de los ficheros de log de la aplicación, principalmente su creación y escritura.
  - *Log:* método constructor que inicializa variables.
  - *timestamp:* método que devuelve diferentes formatos de fechas y tiempos para la creación de ficheros y carpetas.
  - *inicTime:* crea las carpetas necesarias de año y mes (si han cambiado respecto a las ya creadas).

- *write*: método llamado cada vez que se quiere escribir algo en el fichero log, aquí también se crea o se sobreescribe el que ya existía. Aparte imprime por pantalla.

```
//escribe en el fichero de log
try {
    OutputStream fout= new FileOutputStream(ruta_fich_log, bAppend);
    OutputStream bout= new BufferedOutputStream(fout);
    while (bBloqueo){
        //System.out.println("Bloqueo");
    }
    bBloqueo = true;
    out = new OutputStreamWriter(bout);
    if(sTexto != "\n") {
        out.write(">> " + sTimeLine + sTexto + "\n");
    } else{
        out.write("\n");
    }
    out.flush();
    out.close();
    bBloqueo = false;
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

Figura 3.13. Porción del método write de escritura en el fichero log

- Clase *Props:* se trata de una clase cuyo objetivo es obtener las variables que se almacenan en el fichero externo *RouteDriver_Server.properties.* La clase proporciona métodos para ser utilizados por el resto de clases y que obtengan sus variables. Las ventajas de utilizar un fichero externo de almacenamiento de variables es que no es necesario hacer cambios en el código y volver a compilar el proyecto, simplemente con editar el fichero y hacer los cambios necesarios es suficiente.

  - *Props:* método constructor que carga el fichero de properties externo y llama a los métodos correspondientes para obtener las variables. Al constructor se le pasa como parámetro un valor

  - *initVarsGen:* obtiene las variables generales del fichero properties.

  - *initVarsObex:* obtiene las variables de la clase RDObex.

  - *initVarsRfcomm:* obtiene las variables de la clase RDRfcomm.

  - *initVarsLog:* obtiene las variables de la clase Log.

- *initVarsFtp:* obtiene las variables de la clase FtpDownload.

```
//obtiene las variables para "FtpDownload"
private void initVarsFtp()
{
    DOMAIN = properties.getProperty("DOMAIN").trim();
    IPFTP = properties.getProperty("IPFTP").trim();
    USER = properties.getProperty("USER").trim();
    PASSWORD = properties.getProperty("PASSWORD").trim();
    FTP_IMAGE_FOLDER = properties.getProperty("FTP_IMAGE_FOLDER").trim();
    TEMP_FOLDER = properties.getProperty("TEMP_FOLDER").trim();
}
```

Figura 3.14. Implementación del método de obtención de variables

3.3.4 Programación de la aplicación Web

Debido a que en esta aplicación no se ha utilizado programación orientada a objetos, se van a ir definiendo los ficheros de los que consta el proyecto web, cual es su cometido y sus principales características.

Se pueden diferenciar tres tipos de ficheros: páginas html, php y javascript, páginas php y hojas de estilo. Comenzamos con el primer grupo:

- *default.php*: es la página de acceso a la web. Es muy simple ya que solo consta de un acceso de usuario con login y contraseña. Se basa un formulario html que envía los datos de acceso por POST (es el modo de envío de los datos del formulario, los datos se envían codificados y después se pueden obtener por medio de php) para conseguir el acceso. A continuación se muestra el formulario de acceso y la obtención de los datos de acceso para comprobar que son correctos:

```
<form method="POST" action="main.php">
    <label title = "usuario" for="usuario" class = "textfield">Usuario  </label>
    <input type="text" name="usuario" size="20" class = "textfieldpss_color" /><br /><br />
    <label title = "password" for="password" class = "textfield">Password  </label>
    <input type="password" name="password" size="20" class = "textfieldpss_color"><br /><br />
    <br /><br />
    <div class = "center_button">
        <input type="submit" value="Acceso" name="privado" class = "color_bigbutton">
    </div>
</form>
```

Figura 3.15. Ejemplo formulario de acceso a usuario HTML, método POST

```
$usuario = $_POST['usuario'];
$password = $_POST['password'];
```

Figura 3.16. Obtención variables método POST

- *main.php*: este archivo es la página central del proyecto web. Solo existen dos pantallas de visualización en la aplicación y esta es la

principal. Consta de tres módulos que hay que realizar para la obtención de la ruta final y aparte, tiene el botón de acceso para gestionar los conductores de la base de datos.

Esta página está desarrollada en html, php y javascript:

- Con **html** se crea el contenido, la estructura y el marcado del texto. Define los tres módulos mediante 3 * * alineados horizontalmente con formularios en su interior que envían el contenido o los datos insertados en cada módulo a las páginas php que se encargan de su procesamiento.

  Cada módulo tiene un cometido, el primero se encarga de la inserción de las instrucciones de la ruta. Está creado con un formulario con campos de tipo texto y con las opciones de enviarlos o limpiar los campos. Cuando son guardadas las instrucciones se almacenan momentáneamente en pequeños archivos jpg en una carpeta del servidor.

```
<label title = "Indicación 1" for="i1">I1.  </label>
<input type="text" name = "i1" size="45" maxlength="86" />
```

Figura 3.17. Fragmento formulario html: input tipo text

  El segundo módulo es opcional y su cometido es agregar un mapa a las instrucciones. No es un paso obligatorio ya que el mapa puede hacer que el archivo a enviar aumente considerablemente en tamaño. Se realiza mediante otro *<form>* pero en este caso de tipo *file.* Con ese formulario es posible escoger una imagen del ordenador del usuario y subirla al servidor. Al igual que en el paso anterior, la imagen escogida es almacenada en una carpeta del servidor.

```
<p>Escoge el mapa (formato .jpg):</p>
<input type="file" size = "35" name="archivo" id = "archivo" accept="image/jpeg" />
```

Figura 3.18. Fragmento formulario html: input tipo file

  Para finalizar encontramos el módulo más importante ya que es el encargado de unir todo el contenido y formar el archivo de ruta final. En este módulo igualmente formado por un formulario, se escoge el nombre del usuario al que pertenece la ruta mediante un desplegable (*<select>*) con todos los trabajadores de la empresa (se obtienen desde la base de datos xml). Aparte se añade un título a la ruta y si se desea o no adjuntar el mapa que se haya escogido previamente. Se puede visualizar el resultado final y en caso de no estar satisfechos puede eliminarse y comenzar una ruta nueva.

```
<select name="ruta_name" id="ruta_name" class = "textfield_color" disabled="disabled">
    <option value = ""> Elige el nombre del conductor</option>
    <?php
        include("utils/xml.php");
        $xmlProc = new XmlProccess();
        $arrayNames = $xmlProc -> readXmlTag($xmlFile, $tagNombre);
        $arrayBt = $xmlProc -> readXmlTag($xmlFile, $tagBt);

        $arrayNamesLength = count($arrayNames);
        for($r = 0; $r < $arrayNamesLength; $r++){
            $value = str_replace(" ", "", $arrayBt[$r]);
            echo "<option value=$value> $arrayNames[$r]</option>";
        }
    ?>
</select>
```

Figura 3.19. Fragmento formulario html: select

Los módulos que se comentan son de orden secuencial, por lo tanto, sin que se haya pasado por el paso número uno no se puede acceder al dos ni al tres. Esto se controla mediante variables php que indican si se ha pasado por un módulo concreto o no. Cuando se envía el formulario concreto, y pasa por la página php para realizar el procesamiento, se establece el valor en las variables que indican que módulos se han completado. Al cargar de nuevo la página html se comprueba, y junto a javascript, se habilitan unos contenidos u otros de la página para continuar con el paso adecuado.

```
    <?php
        if($newruta){
    ?>
            <script type="text/javascript">
                setVar('newruta');
                enableBtnStep(0);
            </script>
    <?php
        }elseif($save){
    ?>
            <script type="text/javascript">
                setVar('save');
                enableBtnStep(0);
                enableBtnStep(1);
            </script>
    <?
        }
    ?>
```

Figura 3.20. Uso de php y javascript combinados

- Con **javascript** se consigue dotar de mayor dinamismo a la web y mostrar u ocultar elementos de la misma en distintas situaciones, cambiar colores, habilitar/deshabilitar botones, validar los campos

introducidos antes de enviarlos (así se evita la comprobación antes de llegar al servidor), etc. Esto se consigue mediante los scripts insertados en la página principal. En nuestro caso, se inserta un gran script con funciones javascript definidas que serán usadas a lo largo de la página.

En la figura mostrada a continuación se puede ver una función definida en javascript que es una de las que hace referencia la *figura anterior 3.20*.

```
//habilita botones en cada paso
function enableBtnStep(boton){
    if(boton == 0){
        document.form1.guardarInd.disabled=false;
        document.form1.limpiar.disabled=false;
        for(h=1; h<=11; h++){
            document.getElementById('i'+h).disabled=false;
        }
    }else if(boton == 1){
        document.form2.gMapa.disabled=false;
        document.form2.archivo.disabled=false;
        document.form3.creaImg.disabled=false;
        document.form3.ruta_name.disabled=false;
        document.form3.ruta_title.disabled=false;
    } else if(boton == 2){
        document.form2.vPrevia.disabled=false;
        document.form3.checkmapa.disabled=false;
    } else if(boton == 3){
        document.form3.eliminar.disabled=false;
        document.form3.vPrevia2.disabled=false;
    }
}
```

Figura 3.21. Ejemplo de una función javascript

- *conductores*.php: esta página es accesible desde la página principal main.php. Se encarga de gestionar los datos de los trabajadores, se pueden añadir nuevos trabajadores (sólo se añade el nombre y el Bt id correspondiente) o eliminar los ya existentes.

  El html consta de dos formularios, uno tipo texto para insertar el nuevo usuario y el otro es un desplegable con todos los trabajadores para eliminar el que sea necesario. Al igual que en la página principal, se hace uso de php para las operaciones con la base de datos xml (*xml.php*) y de javascript para validar los datos y habilitar botones.

A continuación se definen los archivos php auxiliares que se ejecutan desde los mencionados anteriormente:

- *step0.php:* este fichero se encarga de eliminar los archivos utilizados para la creación de la ruta previa que no sean el archivo de ruta final. Esto sucede cada vez que se comienza a crear una nueva ruta.

```
function delete($dir){
    $del = false;
    // Abre un directorio conocido, y elimina todo el contenido
        if ($dh = opendir($dir)) {
            while (($file = readdir($dh)) !== false) {
                if(is_file($dir.$file)){
                    if(unlink($dir.$file)){
                        $del = true;
                    }
                }
            }
            closedir($dh);
        }
    return $del;
}
```

Figura 3.22. Función php para eliminar ficheros

- *step1.php:* este fichero se encarga de procesar las instrucciones introducidas en el paso número 1. Obtiene el texto del formulario que se envía por POST al fichero php y crea imágenes con los campos de texto. Las imágenes se almacenan en el servidor hasta que se forme la ruta final.

- *step2.php:* en este fichero se maneja el mapa recién subido al servidor. Se cambia el nombre del mismo, se escala a un tamaño menor si es necesario y se guarda en el directorio correcto hasta que finalmente sea utilizado para crear la imagen de ruta final.

- *step3_1.php:* dentro del módulo tres se realizan dos acciones: crear la ruta final y eliminar la ruta recién creada si se necesita. Este fichero php se encarga de la primera opción, une todas las imágenes creadas en el paso 1, y si se ha marcado, agrega el mapa a la imagen de la ruta final. En este paso también se introduce el nombre del conductor al que se va a asignar la ruta y el título de la ruta. El nombre del archivo no será el del conductor, si no que se hace una consulta al xml donde está asociado el nombre del conductor y el Bluetooth id, este último será finalmente el nombre del archivo.

- *step3_2.php:* este php se encarga de la segunda parte del módulo 3, no es obligatorio ya que es para eliminar la ruta recién creada por si no estamos acorde con ella. Existe la opción de visualizar la ruta con la función de vista previa.

Los ficheros php siguientes son simplemente definiciones de funciones que van a ser utilizados desde los ficheros php anteriores. Se engloban en estos ya que serán utilizadas en más de una ocasión.

- *xml.php:* en este php se definen funciones relacionadas con la inserción y obtención de información de la base de datos xml. En el caso de un nuevo trabajador se inserta un nuevo elemento xml con la información, en cambio si se trata de eliminar un usuario existente se parsea el documento en su busca y se elimina.

- *resizeImage.php:* en este caso se definen funciones relacionadas con el tratamiento de las imágenes utilizadas. Las funciones que lo componen son la de escalado tanto de imagen vertical como horizontal y la de convertir texto en imagen jpg utilizado en el paso número 1 de la aplicación.

Capítulo 4

Descripción funcional

Ya se ha analizado el proyecto en profundidad, desde una visión general de la misma hasta el análisis de clases de las aplicaciones. Sin embargo, aunque si exista una idea de en que ámbito va a utilizarse, no se ha visto la aplicación en funcionamiento.

El propósito del proyecto es implantar un sistema de envío de rutas a los conductores de una empresa de transportes mediante NFC y Bluetooth. Esto se dará en un lugar concreto de la empresa donde esté situado el puesto NFC, que será próximo a la sala del servidor Bluetooth. El momento de la petición de la ruta será el elegido por el trabajador. Teniendo en cuenta esto, en los siguientes puntos se va a realizar la descripción del uso que se le puede dar al proyecto

4.1 Uso para el trabajador

Cada conductor de la empresa tendrá instalado en su teléfono móvil corporativo la aplicación ReaderNFCBluetooth. Con esta aplicación el móvil obtendrá el identificador Bt al acercar el móvil al puesto NFC y automáticamente el móvil establecerá comunicación con el servidor Bluetooth haciendo la petición de su ruta.

El proyecto se ha realizado con el Nokia 6131 NFC pero el móvil corporativo podría ser cualquier dispositivo habilitado con NFC y compatible con el MIDlet creado. Las pruebas se han realizado tanto con emuladores como con dispositivos reales. Para la funcionalidad NFC se pueden usar ambos recursos, aunque se utilizan capturas de pantalla del emulador ya que proporcionan mayor calidad de imagen, Sin embargo, el caso de Bluetooth es diferente ya que no se puede emular (aún utilizando la herramienta Nokia Conectivity Framework, nunca llega a la búsqueda de servicios) y las imágenes serán tomadas de los dispositivos reales. Los pasos a seguir por parte del trabajador son los siguientes:

1. El trabajador abre la aplicación desde su móvil

Figura 4.1. Apertura MIDlet ReaderNFCBluetooth

2. El trabajador acerca el móvil al puesto NFC. El lector del móvil activa el tag pasivo del puesto y lee la información grabada en la tarjeta, el identificador Bluetooth.

3. Una vez obtenida la información del id Bt, el móvil abre la conexión Bluetooth con el servidor (*figura 4.3*). Envía la petición de ruta y se queda a la espera de la respuesta del servidor.

4. El móvil recibe la respuesta del servidor confirmando la recepción de la petición e informa al usuario que está obteniendo su ruta. El usuario queda a la espera del envío.

Figura 4.3. Comunicación Bluetooth en MIDlet

5. La imagen llega al dispositivo del usuario, se almacena en el mismo y se muestra por pantalla. Dependiendo de la cantidad de instrucciones incluidas en la ruta, la vista instantánea en la pantalla será más o menos útil, ya que viene limitada por el tamaño de la pantalla del dispositivo. Esta imagen se muestra dentro del display del MIDlet que se está ejecutando, por lo que el tamaño es muy limitado. Por esto, en ocasiones en que las rutas no tienen buena visualización, es muy recomendable acceder a la ruta desde la tarjeta de memoria donde se ha almacenado (se indica por pantalla como acceder), aquí tiene la opción de realizar zoom si es necesario y visualizarla con un mayor detalle.

Hay que destacar que las rutas enviadas a los trabajadores no son indicaciones pormenorizadas de las indicaciones a seguir, ya que para eso podrían utilizar un GPS. Se trata de guías que indican donde tendrán que recoger la mercancía esa jornada y las paradas que debe hacer durante su ruta.

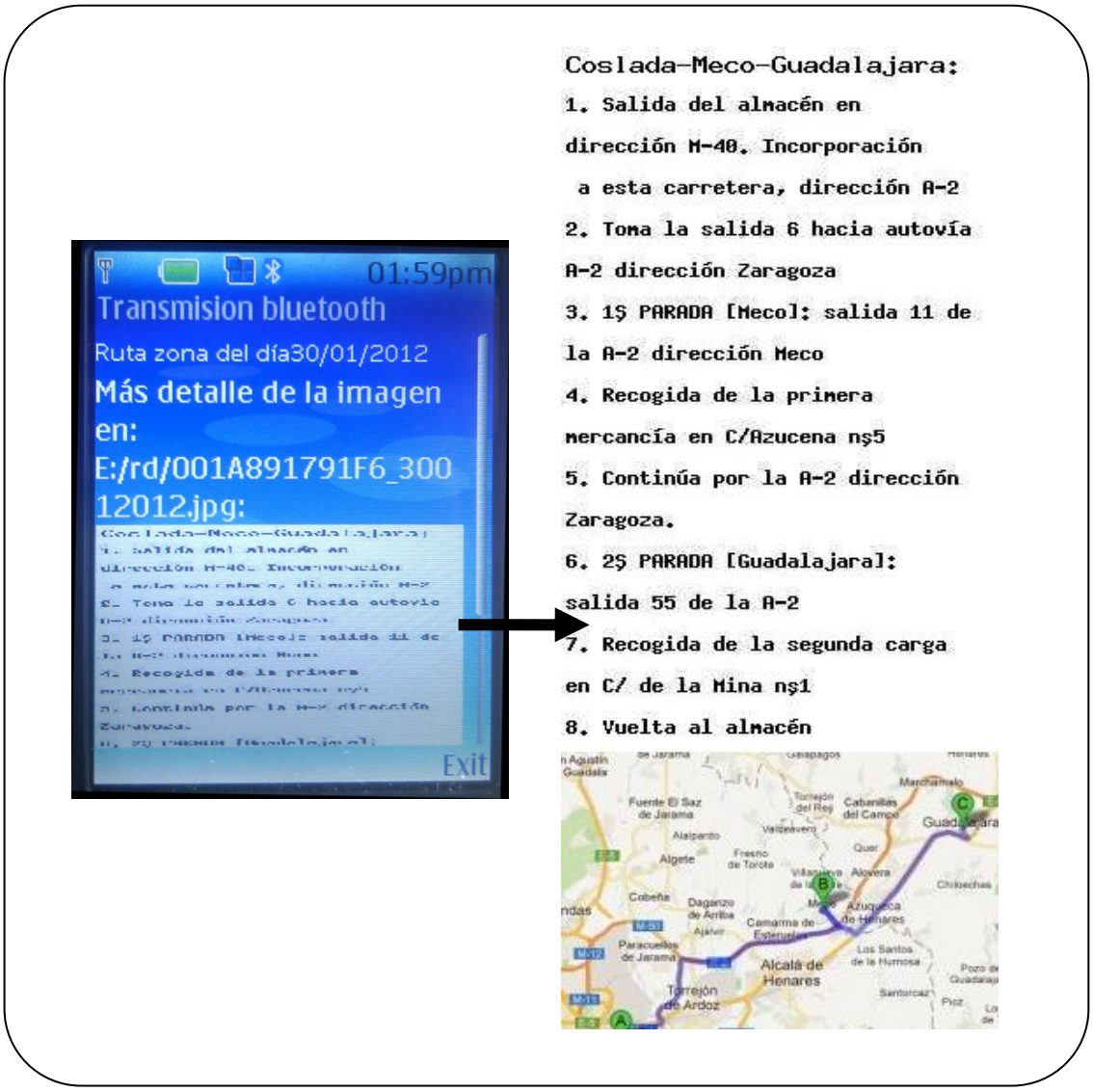


Figura 4.4. Vistas de la ruta: en pantalla y desde tarjeta

4.2 Uso administrativo

4.2.1 Escritura en tag NFC

Se trata de otro MIDlet que se instalará en el Nokia 6131 NFC. En este caso no será utilizado por los conductores si no que lo utilizarán para grabar la información en el tag del poster NFC. El uso de esta aplicación está dirigido para el personal administrativo de la empresa, aunque será utilizada en contadas ocasiones, cuando por alguna razón cambie el Bt id del servidor Bluetooth. La utilización de la aplicación es la siguiente:

1. El usuario abre la aplicación y se muestra la pantalla principal para introducir el identificador.

Figura 4.5. Pantalla principal MIDlet WriterNFC

2. Se introduce el Bt id y se acepta si es correcto.

Figura 4.6. Introducción Bt id con MIDlet NFCWriter

3. Se acerca el móvil al poster NFC (en la zona concreta donde se encuentra el tag) y se comienza la grabación. Si todo es correcto aparece el mensaje *Mensaje NDEF escrito*.

Figura 4.7. Mensaje NDEF grabado en tarjeta NFC

4.2.2 Servidor Bluetooth

En este caso contamos con una aplicación Java de escritorio pero no tiene interfaz gráfica. La aplicación se lanza y se ejecuta en segundo plano, al lanzarse registra el servicio Bluetooth permitiendo así a los dispositivos clientes encontrar el servidor Bluetooth y el uso al que está destinado (servicio).

Figura 4.8. Lanzamiento de la aplicación java RouteDriver (servidor Bluetooth)

Cuando la aplicación se lanza, antes de registrar el servicio tienen lugar dos acciones: la creación de las carpetas de los directorios de almacenamiento (si no estaban creados con anterioridad) y la creación e inicio de los ficheros de log donde consta la actividad del servidor y los clientes conectados. Las rutas creadas son las siguientes: *C:\RouteDriver\img, C:\RouteDriver\log* y *C:\RouteDriver\temp.*

Figura 4.9. Directorios de la aplicación RouteDriver (Servidor Bluetooth)

En la carpeta *log* se almacenan los ficheros de log en función del año y el mes en el que nos encontramos, la nomenclatura utilizada para los archivos es: *logdíamesaño.txt* (por ejemplo log30012012.txt). En *temp* se almacena temporalmente la imagen que se descarga del servidor FTP para posteriormente almacenarse en la carpeta *img*, en el año, mes y día correspondientes (el nombre del archivo es el identificador Bluetooth del usuario).

```
>> *** Comienza ServerLauncher de RouteDriver_Client ***
>> Dirección Bluetooth local (server): 000DF048454B
>> Nombre Bluetooth local (server): ARUAL
>> Lanzamiento de RFCOMM server

>> 12:48:02 >> ------------------- INICIO ESPERA CONEXION ---------------------
>> 12:48:02 >> Servidor iniciado. Esperando conexión de cliente...
>> 12:48:38 >> Se ha recibido una conexión RFCOMM....
>> Dirección Bluetooth remota: 001A891791F6
>> Nombre Bluetooth remoto: Nokia6131NFC
>> 12:48:45 >> Ruta pedida por el usuario: 001A891791F6.jpg
>> Preparando conexión con el servidor para descargar la imagen
>> 12:48:45 >> Conectando con ftp routedriver.site50.net (64.191.114.182),
              user: a9831754 y password: 123456p@ssword ...
>> 12:48:46 >> Conexión OK, conectado con routedriver.site50.net
>> Busqueda de imagen a descargar en: /public_html/imgs
>> 12:48:47 >> Escogemos el fichero 001A891791F6.jpg del FTP dentro de la
              carpeta /public_html/imgs
>> 12:48:47 >> Descargando el fichero...
>> 12:48:49 >> Fichero bajado OK (001A891791F6.jpg)
>> 12:48:49 >> Descargado a "C:\RouteDriver\temp\001A891791F6.jpg"
>> Eliminado fichero descargado del servidor
>> 12:48:50 >> Conexion cerrada con el servidor
>> 12:48:50 >> Cambiamos la imagen de "temp" a
              "C:\RouteDriver\img\2012\01\30\001A891791F6.jpg"
>> El tamaño de la imagen original es: 113582
>> El tamaño del array de bytes es: 113582
>> 12:48:50 >> Imagen descargada ok
>> Envía mensaje confirmación descarga al usuario
>> 12:48:50 >> Enviamos los bits por bluetooth al cliente...
>> 12:48:50 >> ------------------------- FIN CONEXION ----------------------------
```

Figura 4.10. Fragmento de log sobre la actividad del servidor

Como se observa en la *figura 4.10* el servidor comienza su actividad lanzando el servicio y se queda a la espera de conexiones. Cuando se conecta un dispositivo, conecta con el servidor FTP y descarga la imagen que posteriormente envía al dispositivo conectado.

4.2.3 Creación de rutas

En este caso utilizamos una aplicación Web por lo que necesitamos un navegador para ejecutarla. En la barra de direcciones introducimos la URL de la aplicación (*http://www.routedriver.site50.net*) y lo primero que aparece es la pantalla para introducir usuario y contraseña.

Figura 4.11. Pantalla de acceso de aplicación Web

Los trabajadores del departamento administrativo pueden acceder desde cualquier lugar con acceso a internet ya que disponen de usuario y contraseña. Una vez se introducen los datos requeridos, se accede a la página principal donde se puede comenzar la creación de las rutas y la gestión de los trabajadores.

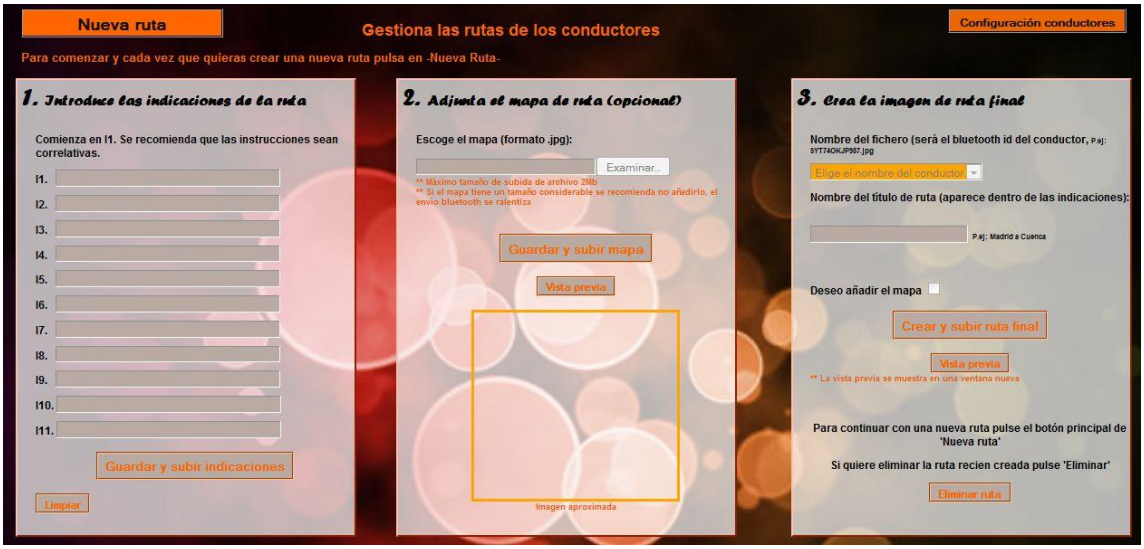


Figura 4.12. Página principal de la aplicación Web

Se observa que la página principal consta de 3 módulos: el primero permite introducir las instrucciones, el segundo adjuntar un mapa y en el tercero se forma la ruta final. Para comenzar la creación de una ruta, se pulsa sobre el botón *Nueva ruta* de la esquina superior izquierda y de esta manera se habilita el módulo 1 para el comienzo (*figura 4.13 izq*). Cuando se introducen todas las instrucciones, se guardan y así se habilita el módulo 2 donde, opcionalmente, se puede adjuntar un mapa que acompañará a las instrucciones en la ruta final y ver su vista previa (*figura 4.13 dcha*).

Figura 4.13. Paso 1 y paso 2 de la aplicación Web

Además, teniendo en cuenta que el paso 2 es opcional, se podría pasar directamente al 3 por lo que este también se habilita. En este tercer módulo es donde se crea la ruta definitiva y para ello hay que introducir una serie de datos: el conductor al que está asociada esa ruta que se elige en el desplegable, un título descriptivo para la ruta y si se desea o no añadir el mapa (esta opción solo estará disponible en caso de adjuntar un mapa en el paso 2). Cuando se ha guardado y subido el mapa se habilita la opción de ver la vista previa del mismo la cual se abre en otra ventana aparte, si no se está conforme con el resultado se puede eliminar y comenzar de nuevo.

Figura 4.14. Paso 3 app. Web

Al finalizar con el último paso se habrá almacenado en el servidor FTP la imagen de la ruta recién creada, con el nombre correspondiente al Bt id del trabajador que se ha seleccionado y estará disponible para su descarga y envío al cliente cuando realice la petición.

4.2.4 Gestión de trabajadores

La aplicación Web tiene una funcionalidad extra para manejar los conductores de la empresa que se encuentran en la base de datos xml. El acceso a este módulo se encuentra en la página principal de la aplicación, en forma de botón, en la esquina superior derecha (*Configuración conductores*).

Cuando se pulsa sobre el botón se abre una nueva ventana que contiene dos opciones: *añadir conductor* y *eliminar conductor*. En función de lo que se quiera hacer se introducen los datos en uno u otro cuadro de texto. Para añadir se pide el nombre del conductor y su Bt id, en cambio para eliminar a un conductor de la base de datos basta con seleccionarlo en el desplegable. Cuando se ha llevado a cabo la opción deseada, se podrá seguir con el proceso en la página principal.

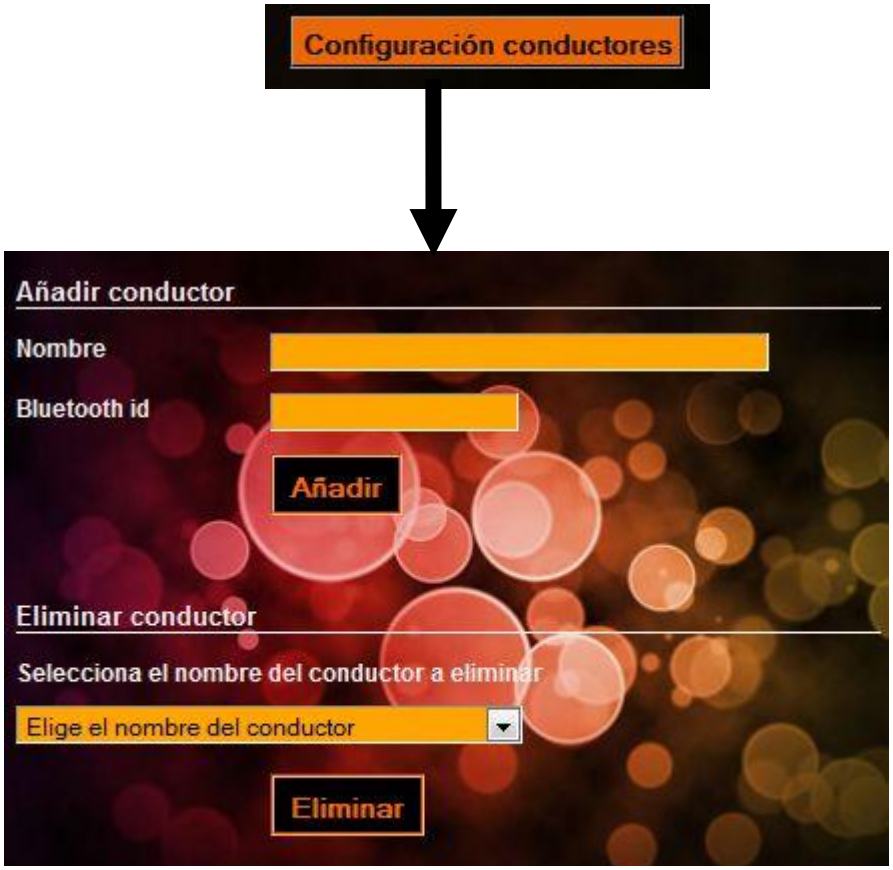


Figura 4.15. Página de configuración de conductores de la app. Web

Capítulo 5

Pruebas

En este capítulo se describen la batería de pruebas que se han ido realizando a lo largo del proyecto para garantizar su correcto funcionamiento.

Las pruebas de software son una parte esencial en el desarrollo de una aplicación y se integran en el ciclo de vida del desarrollo del software. En este caso, las pruebas seleccionadas realizadas son acciones o elementos claves de la funcionalidad del proyecto. Una vez se verificó el comportamiento del sistema mediante los tests, se procedió a integrar los diferentes módulos del proyecto y así lograr la funcionalidad global descrita en el *capítulo 4*.

5.1 Pruebas del módulo NFC-Bluetooth

Para garantizar el funcionamiento final de las aplicaciones que componen este módulo, se fueron realizando pruebas a las partes más importantes y con mayor impacto. Se dividieron en aplicaciones más sencillas previamente creadas, y finalmente se integraron las necesarias para lograr el funcionamiento global.

5.1.1 Lectura/escritura en tarjeta NFC

Para comprobar el correcto funcionamiento de las aplicaciones con tecnología NFC se utilizaron dos aplicaciones complementarias, una de escritura y otra de lectura. La diferencia con el resultado final implementado en el proyecto es que la aplicación de lectura no incluye la funcionalidad Bluetooth, simplemente lee el mensaje NDEF del tag NFC y lo muestra por pantalla.

Las pruebas se llevaron a cabo en el emulador del Nokia 6131 antes de su instalación en el dispositivo real. La aplicación de escritura se ejecuta exactamente igual que la utilizada en la solución final, por lo tanto remito al punto *4.2.1* para obtener información sobre ella. Sin embargo, para poder realizar la simulación correctamente, es necesario que el valor almacenado en el tag se mantenga para la futura simulación de lectura. Para lograrlo, se sitúa el ratón sobre el tag NFC virtual del emulador, que se ha utilizado para grabar el mensaje NDEF, y con el botón derecho se selecciona *Save data.*

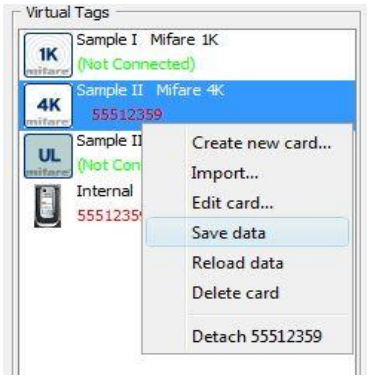


Figura 5.1. Guardar datos en tag virtual NFC

Para la lectura del tag se utilizó otra aplicación muy similar a la de escritura, que establece el contacto entre tarjeta y lector/escritor de la misma manera, pero en vez de grabar información en ella, lee la que ya tenía. Al igual que en el caso de escritura, se escoge el mismo tag virtual para simular el contacto entre dispositivo y tag (que es donde previamente se ha almacenado la información) y comprobamos que la lee sin problemas. La simulación del acercamiento entre móvil y tag se realiza haciendo doble click en el tag o arrastrándolo y soltándolo sobre el móvil.

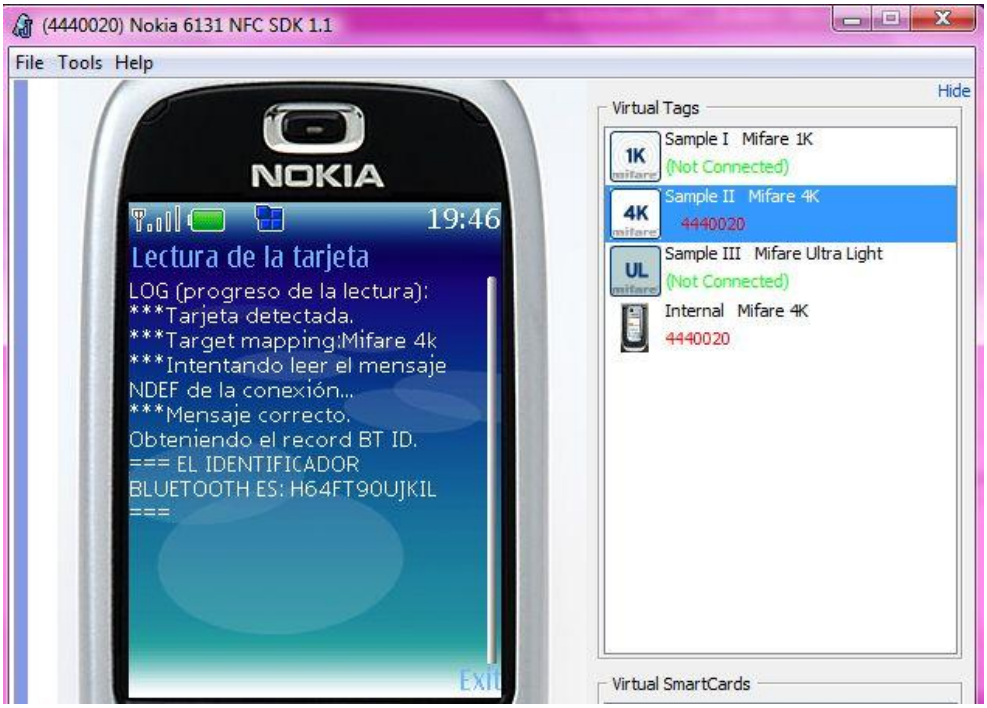


Figura 5.2. Lectura de mensaje NDEF

5.1.2 Búsqueda de dispositivos y servicios Bluetooth

Hay una única aplicación para la funcionalidad NFC y Bluetooth, pero para asegurarnos de su funcionamiento se hicieron pruebas por separado. La parte de lectura se ha explicado en el punto anterior, ahora vamos a revisar la parte Bluetooth.

La idea principal del proyecto ha sido la utilización del emulador y dispositivo real Nokia 6131, por lo tanto todas las pruebas se han realizado siempre con él.

Teniendo en cuenta esto, surgieron algunos problemas en el momento de la simulación con Bluetooth que se explican a continuación.

Como sabemos, para la comprobación y verificación de los MIDlets se ha utilizado Wireless Toolkit (WTK). Esta herramienta trae unos emuladores por defecto pero no comprende el Nokia 6131, por esto el emulador se obtuvo de la SDK previamente descargada en *Nokia Forum* y se integró con la herramienta. Los emuladores que trae por defecto WTK tienen la posibilidad de emular conexiones Bluetooth entre ellos y encontrar dispositivos y servicios, por lo que lo intentamos con nuestro emulador y lo conseguimos también. El problema es que el funcionamiento ideal del proyecto es que se produzca la comunicación Bluetooth entre el móvil y un ordenador, no entre dos móviles, y aquí es donde entra el Nokia Connectivity Framework (NCF).

NCF permite simular conexiones en los emuladores, pero además, se les puede dotar de mayor realismo ya que en vez de utilizar la conexión Bt simulada, se le puede asociar antenas reales que estén integradas en el ordenador donde se ejecuta el emulador, de esta manera se obtiene una conexión real. Gracias a esta herramienta se montó la siguiente simulación: en un ordenador con Bluetooth se ejecutaba el emulador Nokia 6131 con conexión Bt real y en otro se instaló la aplicación de escritorio creada como servidor Bluetooth y se ejecutaba ofreciendo su servicio a los posibles dispositivos clientes. Todo funcionaba correctamente hasta el momento de la búsqueda de servicios, que nunca llegaba a ejecutarse debido a alguna limitación de la herramienta NCF, y por esto se tomó la decisión de realizar todas las pruebas con los dispositivos reales (sustituir el emulador por un dispositivo físico Nokia 6131).

Finalmente, realizando dichas pruebas se comprobó el correcto funcionamiento del módulo Bluetooth. Encontraba y listaba los dispositivos Bt a su alcance y eligiendo el dispositivo servidor Bt encontraba el servicio sin problemas. Todo esto, al producirse la unión entre el módulo NFC y Bluetooth desaparece ya que sabemos a qué dispositivo queremos conectarnos y no es necesario listarlo ni seleccionarlo, se ejecuta en segundo plano para comprobar que el servidor está dentro del alcance del móvil.

5.1.3 Lanzamiento de servicio Bluetooth

En este caso se trata de una comprobación muy simple. La aplicación java de escritorio se encarga de publicar el servicio Bluetooth entre otras cosas, y para estar seguros de que esto lo ha realizado correctamente, se utilizó un programa de terceros llamado *Medieval Bluetooth Network Scanner*. Este programa proporciona información de los servicios Bluetooth de un dispositivo conectado a nosotros o los servicios locales, en este caso de nuestro servidor Bluetooth.

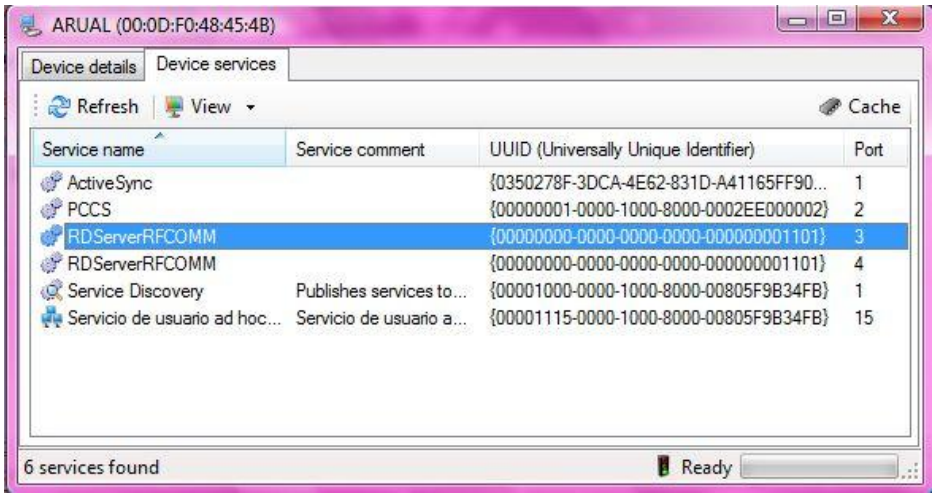


Figura 5.3. Medieval Bluetooth Network Scanner

En la imagen anterior se observa como si está publicado el servicio del servidor Bluetooth (RDServerRFCOMM), e incluso se puede ver el UUID y el puerto utilizado. Así, si hubiera algún problema de búsqueda de servicios se podría descartar que fuera por parte del servidor Bluetooth porque estaba realizando su cometido.

5.2 Pruebas de la aplicación Web

Como se ha visto, la aplicación Web consta de varios módulos que se van ejecutando hasta llegar al resultado final. Debido a esto, las pruebas realizadas en esta aplicación se han basado en la comprobación de cada uno de los pasos que la componen.

5.2.1 Subida de ficheros

En los dos primeros módulos se introducen las instrucciones o indicaciones para el conductor y se adjunta un mapa. El contenido se almacena provisionalmente en un directorio del servidor para posteriormente unirlas y formar la ruta final. Las pruebas realizadas consistían en introducir diferentes números de instrucciones (desde 1 hasta 11) y comprobar en el servidor si se estaban almacenando todas ellas, además de comprobar si el fichero del mapa también se encuentra almacenado.

Como vemos en la siguiente figura, se muestran los directorios que componen el servidor y en *imgs/temp* se encuentran los ficheros subidos. Al abrirlos se comprueba que tienen el contenido correcto que previamente se ha introducido en la aplicación. Para visualizar los directorios y ficheros se ha utilizado el programa *Filezilla* que permite navegar por los directorios del servidor mediante un usuario y contraseña FTP.

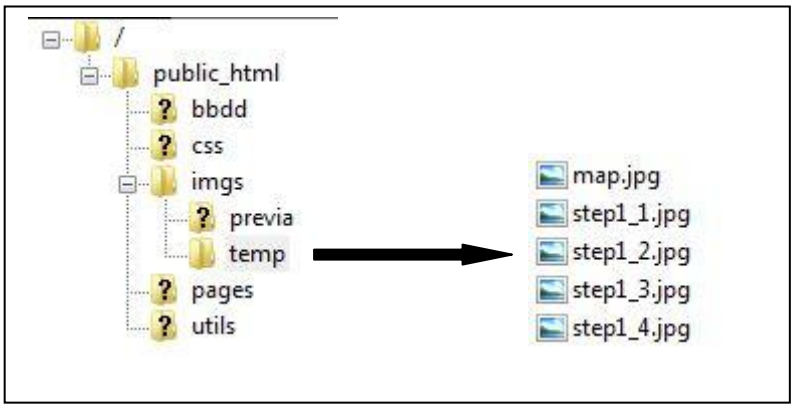


Figura 5.4. Estructura de carpetas aplicación Web

Además, en la propia aplicación Web existen unas reglas de validación que tienen que cumplirse para que el contenido sea válido para la subida, por ejemplo, que al menos haya una indicación (de lo contrario aparece un mensaje advirtiendo que es obligatorio) y que hay un número máximo de caracteres por indicación, cuando se llega a él, no es posible introducir más valores en el campo de texto.

5.2.2 Creación de ruta final

En el último paso de la aplicación no es necesario subir nada al servidor ya que el contenido a utilizar ya lo tiene, simplemente se procesa para unirlo, añadirle el título y nombrarlo correctamente. Las pruebas que se realizan en este caso son comprobar

que en el directorio *imgs* del servidor se ha almacenado la imagen de ruta final con el nombre del id Bt (que es el del trabajador seleccionado) y que su contenido es el correcto, resultado de la unión de los elementos anteriores.

Figura 5.5. Fichero de la ruta final en el servidor

Cuando el servidor Bluetooth se conecta al servidor FTP y se descarga la imagen concreta, esta desaparece del servidor porque ya ha sido pedida y pasa a almacenarse en local.

5.2.3 Actualización de la base de datos

La base de datos xml es exclusiva para su uso mediante la aplicación Web ya que no es demasiado cómodo manejar directamente un archivo xml, sobre todo si es de grandes dimensiones. Por esta razón, había que comprobar que los cambios que se hacían sobre los conductores en la aplicación, tenían su efecto en el fichero xml, (que se añadían o eliminaban). El fichero xml se encuentra en el servidor en el directorio *bbdd.*

Figura 5.6. Actualización de la base de datos de conductores

Aparte de lo anterior, cuando se realiza algún cambio en la base de datos, inmediatamente después se puede contar que ese nuevo usuario para la creación de su ruta si es necesario, por lo que la gestión de los conductores puede utilizarse en cualquier momento. Esto se comprueba realizando algún cambio, como añadir un conductor, y en la página principal del conductor ver que ya aparece el mismo en el desplegable para nombrar el fichero de ruta.

Capítulo 6

Conclusiones y líneas futuras

En este breve capítulo se va a hacer una puesta en común de las conclusiones que se extraen tras la realización del proyecto, los pros y los contras encontrados. Además, se describirán los posibles trabajos que se podrían realizar a raíz de este proyecto, ampliaciones o mejoras del mismo.

6.1 Conclusiones

El objetivo de este proyecto era el desarrollo de varias aplicaciones que implementaran la posibilidad de comunicarse con las tecnologías inalámbricas NFC y Bluetooth. Los desarrollos se han basado en un caso práctico concreto, una empresa de transportes cuyos trabajadores necesitaban obtener diariamente sus rutas por carretera. Los trabajadores podrían obtener la información de una manera sencilla y rápida en sus dispositivos, lo cual se ha logrado con las implementaciones llevadas a cabo.

La mayor carga del desarrollo ha recaído sobre la tecnología Bluetooth. Esta funcionalidad se ha implementado en dos aplicaciones, una móvil y otra de escritorio, permitiendo la comunicación entre dos dispositivos de distinta naturaleza (móvil y PC) y el envío de datos en forma de imágenes. Sin embargo, a pesar de la carga de trabajo de Bluetooth, sin la tecnología NFC no se habrían conseguido dos de los aspectos más importantes del proyecto: la facilidad de uso con un solo toque y la disminución del tiempo de conexión en la comunicación.

La tecnología Bluetooth fue escogida para el proyecto porque está presente en la mayoría de dispositivos móviles hoy en día, y debido a sus características, permite el envío de archivos entre dispositivos de una manera rápida y eficaz. Sin embargo, en el caso de la tecnología NFC, nos hemos encontrado con el problema contrario, aunque está en un momento de despegue, aún no es fácil encontrarla integrada en los dispositivos móviles. Aún así, se tomó la decisión de utilizarla porque proporciona grandes ventajas. En nuestro caso nos ha permitido reducir el tiempo de conexión y sobre todo ha dotado de sencillez con su uso, muy conveniente para personas que no están acostumbradas al trabajo con la tecnología. Finalmente, a pesar del inconveniente de su pobre implantación actual, hemos visto que ha sido una buena elección, ya que parece que la integración de NFC en las SIM de los móviles va a ser un punto fuerte, por lo que cualquier usuario podría obtener tecnología NFC independientemente de su modelo de móvil.

Ambas tecnologías no conllevan un gran coste, por lo que se pueden usar fácilmente a nivel empresarial debido a que su integración puede ahorrar costes y tiempo. No pueden emplearse en todas las situaciones o entornos, por esto mismo no son tecnologías que vayan a sustituir y desbancar a otras existentes, si no que son

complementarias con otras tecnologías del entorno y entre ellas mismas. En el presente proyecto, el balance de la elección de estas tecnologías es positivo ya que se complementan perfectamente, pero como inconveniente hay que destacar lo comentado anteriormente respecto la implantación de NFC en los dispositivos móviles hasta ahora.

Además de las tecnologías de comunicación inalámbricas, para la implementación de todo se ha utilizado el lenguaje de programación Java, en dos de sus versiones (J2ME y J2SE). El desarrollo con la plataforma Java ha sido relativamente sencillo ya que proporciona herramientas para el uso de las tecnologías y tienen ejemplos útiles. Sin embargo, se han encontrado un par de inconvenientes: la dificultad para simular con emuladores las tecnologías inalámbricas y la implementación de Bluetooth con J2SE ya que no tiene soporte para ello, teniendo que utilizarse una librería externa. A pesar de esto, los inconvenientes se han solventado con otras soluciones permitiendo llegar al buen resultado final.

Finalmente, no podemos olvidar la aplicación Web que se ha desarrollado. Esta ha proporcionado mayor comodidad, disminución de posibles errores a la hora de la creación de rutas y mejor control de los trabajadores. Aunque se trata de un proceso manual, ha constado de una serie de pasos guía que han permitido llegar a un resultado final óptimo, tanto para el usuario que lo ejecuta como para el usuario final que recibe el resultado.

6.2 Líneas futuras

El presente proyecto cumple con los objetivos y expectativas marcados en un principio, pero se podrían añadir nuevas funcionalidades y mejoras para conseguir una herramienta más completa. A continuación se describen algunas opciones que podrían resultar interesantes:

- **Implementar la funcionalidad NFCIP**. Se podría realizar el desarrollo para que el envío del contenido se llevara a cabo mediante NFCIP (proporciona mayor rapidez de comunicación que Bluetooth). Habría que estudiar de que manera proveer al PC de tecnología NFC y la tecnología NFCIP podría combinarse con Bluetooth, de forma que habría la opción de escoger que tecnología utilizar.

- **Realizar el desarrollo móvil para un dispositivo más moderno, con soporte OBEX**. Uno de los problemas surgidos durante la realización del proyecto ha sido la incompatibilidad del Nokia 6131 con el protocolo OBEX en la comunicación Bluetooth. Esta funcionalidad está implementada y preparada para funcionar en el proyecto pero se hicieron pruebas en un dispositivo compatible sin NFC. Por lo tanto, sería una opción implantar la aplicación en un dispositivo más moderno con soporte OBEX y NFC donde posiblemente tuviera una mejor pantalla y se podrían ver con mayor claridad las rutas.

- **Cambiar el formato de la base de datos a mySQL.** La base de datos utilizada en la aplicación web es un fichero xml. Para la extensión del proyecto y la empresa funciona sin problemas, además de que es más económico, pero si se tratara de una empresa de mayor tamaño convendría tener otro sistema de base de datos como mySQL o SQL Server (para mayor cantidad de datos). Este cambio afectaría a las consultas realizadas desde los ficheros php de la aplicación Web, pero la manera de mostrarlo al usuario continuaría siendo la misma que ahora.

- **Montar servidores Web y FTP reales.** La aplicación Web se creó como un anexo o añadido al proyecto principal. Por esta razón se utilizó un hosting gratuito para alojarla el cual ya tenía instalado php y nos permite acceso mediante FTP. Este añadido no cambiaría el funcionamiento de la aplicación pero sería una buena práctica para comprender la preparación e instalación de un servidor Web con php y un servidor FTP.

- **Push Registry**. Uno de los inconvenientes que resta automatismo a la aplicación, es que el propio conductor tiene que abrirla antes de acercarla al puesto NFC. Esto podría evitarse si se utilizara el API de Push Registry que permite la ejecución automática del MIDet sin intervención del usuario, así, se podría iniciar automáticamente el MIDlet de lectura NFC cuando se acercara al puesto NFC.

Capítulo 7

Presupuesto y planificación

En este capítulo se va a realizar un análisis del presupuesto necesario para llevar a cabo el proyecto, incurriendo tanto en gastos del trabajo personal como de los materiales utilizados. Además, se verá el diagrama Gantt de planificación del proyecto que muestra el tiempo de dedicación de las diferentes tareas.

7.1 Presupuesto

El presupuesto necesario para realizar el proyecto se desglosa en varios conceptos como los gatos personales, de material, y otros costes directos.

En el conjunto de tablas que se encuentran en la página siguiente se muestra el presupuesto completo, se trata de una ficha con todos los conceptos mencionados anteriormente. Para llegar hasta sus valores, se han tenido en cuenta los siguientes datos:

- **Costes personales**: estos costes son debidos a las personas que participan y desarrollan el proyecto, detallando las horas de trabajo. Para la realización del proyecto se ha necesitado un Ingeniero Técnico de Telecomunicaciones y un tiempo aproximado de **9 meses,** excluyendo el parón veraniego. Eliminando los fines de semana, quedan una media de 20 días laborables al mes, resultando un total de **180 días**. Teniendo en cuenta una jornada laboral de 5 horas diarias, el total de horas de trabajo realizado son **900**. Todos estos valores junto con el dato de la medida hombre mes (131,25 horas), dan lugar a los resultados mostrados en la *tabla 7.1*, con un coste de personal total de **17.940 €**.

- **Costes de equipos**: estos costes son referentes a los materiales utilizados para la realización del proyecto, tanto hardware y herramientas físicas como software. Para la realización del proyecto se han utilizado diversos programas, ya comentados en el capítulo 3 en los recursos software, que son gratuitos por lo que no se incluyen para el cálculo del presupuesto. Para el cálculo se toma un periodo de depreciación de 60 meses, excepto en el caso de Internet que se paga mes a mes, se renueva. Los recursos de material utilizados que conllevan costes se muestran en la *tabla 7.2* y resultan un total de **331,85 €.**

- Los **costes directos** relacionados con el proyecto son sobre el material de oficina y recursos de impresión necesarios, en la *tabla 7.3* se muestran los costes de **270 €**.

PRESUPUESTO DEL PROYECTO

Autora: Laura Tolsada Bris **Departamento**: Ingeniería telemática

Descripción del proyecto (título y duración): Desarrollo de una aplicación de transferencia de ficheros basada en NFC y Bluetooth | 9 meses

Tasas de costes indirectos: 20%

Presupuesto total del proyecto (valorado en €): 22.250€

Desglose presupuestario (costes directos):

Costes de personal				
Nombre	Categoría	Dedicación (hombre mes)	Coste hombre mes (€)	Coste (€)
Laura Tolsada Bris	Ingeniero Técnico de Telecomunicación: sonido e imagen	6,9	2600	**17940**

Tabla 7.1. Costes de personal del proyecto

Costes de equipo					
Descripción	Coste	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenador portátil	900 €	75	9	60	101,25 €
Nokia 6131 NFC	100 €	100	6	60	10,00 €
NFC approved tag	1 €	100	6	60	0,10 €
Internet	49 €	50	9	1	220,50 €
				Total	**331,85 €**

Tabla 7.2. Costes de equipos del proyecto

Otros costes directos	
Descripción	Costes imputables
Material oficina	20,00 €
Gastos de impresión	50,00 €
Gastos de encuadernación	200,00 €
Total	**270,00 €**

Tabla 7.3. Otros costes directos del proyecto

Resumen de costes:

Resumen de costes	
Presupuesto costes totales	Presupuesto costes
Personal	17.940,00 €
Amortización	331,85 €
Otros costes directos	270,00 €
Costes indirectos	3.708,37 €
Total	**22.250,22 €**

Tabla 7.4. Resumen de costes del proyecto

7.2 Planificación

En este apartado se muestra el diagrama Gantt con la estimación de las tareas a realizar, especificadas en el punto 1.3, y su duración.

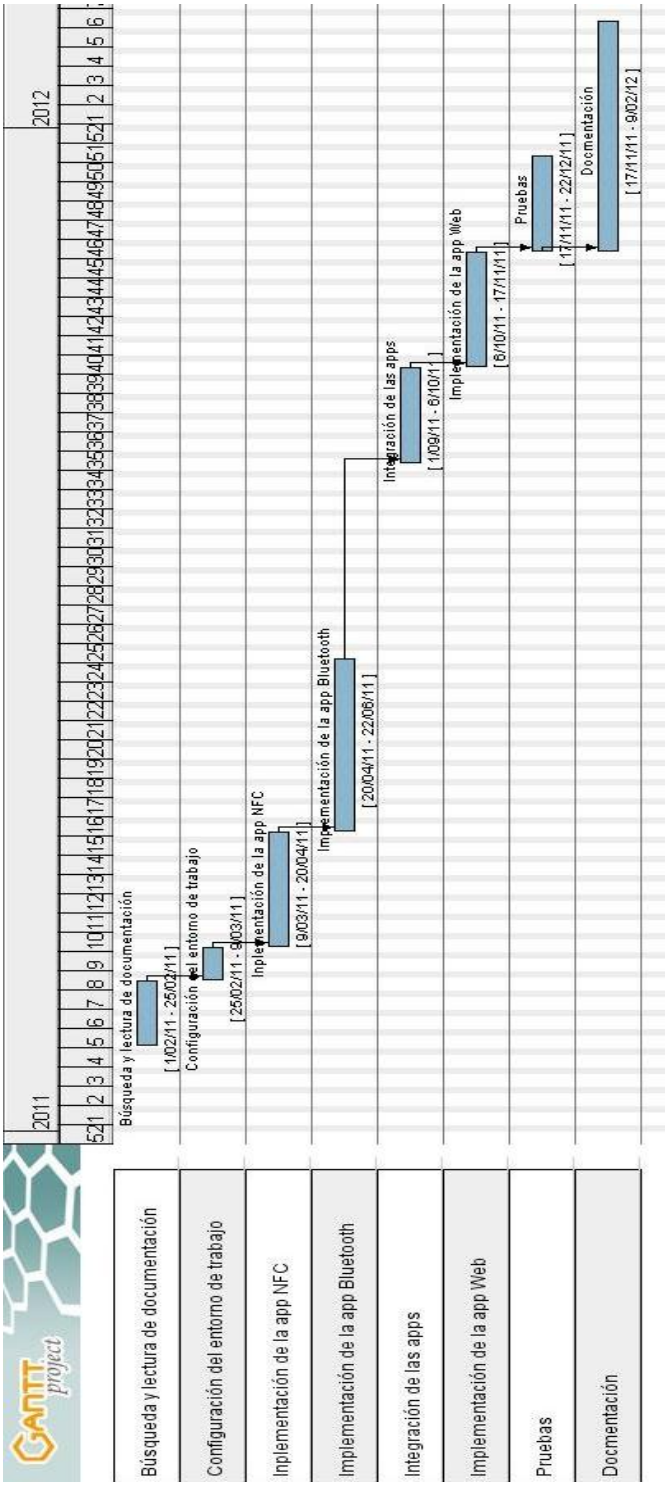


Figura 7.1. Diagrama Gantt

Anexo

Manual de instalación

En este anexo se va a explicar cómo obtener e instalar las herramientas software necesarias (listadas en el punto *3.2.1, recursos software*) para el desarrollo del proyecto.

JDK Java

Para cualquier desarrollo de Java estándar es necesaria la instalación del SDK o JDK. Para ello descargamos el JDK de la web de Oracle: http://www.oracle.com/technetwork/java/javase/downloads/jdk-7u2-download-1377129.html. La última versión es la 7, aunque nosotros hemos utilizado la 6.

Seguimos todos los pasos del asistente de instalación, eligiendo la ruta donde queremos que se instale el JDK. Cuando termina, tenemos que añadir la ruta a las **variables de entorno** de nuestro ordenador, para ello abrimos lo siguiente (para Windows Vista): *Panel del control >Sistema>Cambiar configuración>pestaña opciones Avanzadas>Variables de entorno*.

Las variables que vamos a modificar son CLASSPATH y PATH. La variable CLASSPATH permite a la máquina virtual ubicar todas las clases que la constituyen, así como los directorios donde se encuentran las librerías externas. Por otro lado, la variable PATH se utiliza para que se puedan ejecutar los programas desde cualquier parte del disco duro (sin ella, al compilar en MSDOS habría que escribir la ruta del JDK: *C:> \jdk1.3.1_<número de versión>\bin\javac MyClass.java*).

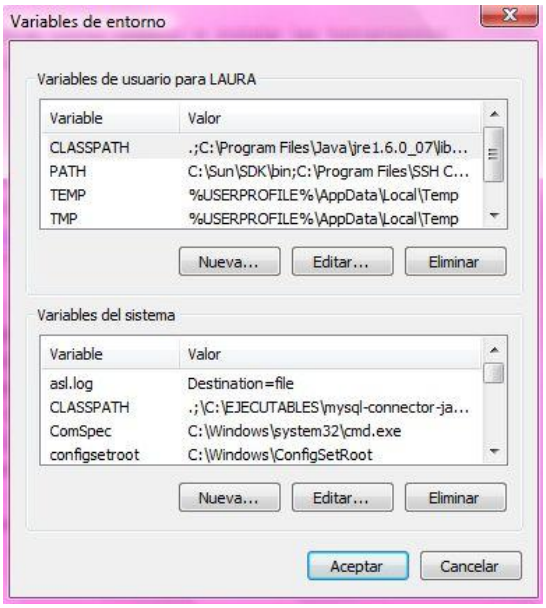


Figura A.1. Variabes de entorno en Windows Vista

Bluecove

Descargamos esta librería externa necesaria para añadir funcionalidad Bluetooth a Java estándar. Hay que descargarse esta librería desde su web oficial: http://bluecove.org/. Al igual que antes, al tratarse de una librería externa necesitará añadirse en el CLASSPATH.

Eclipse

Eclipse es un entorno de desarrollo que utilizaremos para desarrollar la aplicación J2SE. Se trata de un entorno de código abierto por lo que se puede descargar gratuitamente desde su web:

http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/europawinter

La versión utilizada en el proyecto es la 3 (Europa), pero ya hay bastantes versiones más recientes que se podrían descargar. No requiere instalación ya que lo que se descarga es el ejecutable (*eclipse.exe*), cada vez que queramos abrir la herramienta se abrirá este archivo. La primera vez que se ejecuta el programa pregunta por el espacio de trabajo o *workspace* que será el directorio donde se almacenarán los proyectos creados en Eclipse.

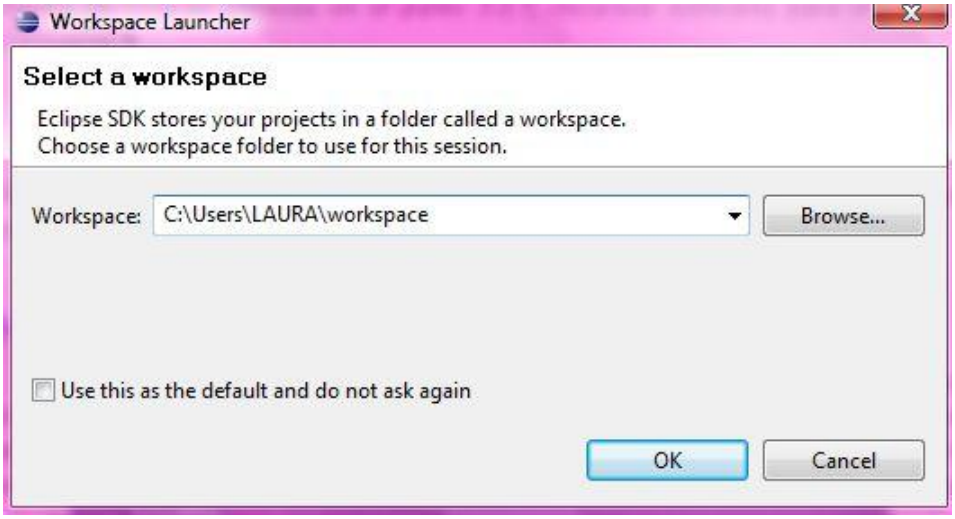


Figura A.2. Selección de workspace en Eclipse

Antes de comenzar, para nuestro proyecto es necesario utilizar un par de librerías externas, *Bluecove* y *commons-net.* Esta última también la añadiremos al classpath y podemos descargar en la siguiente web oficial de Apache: http://commons.apache.org/net/download_net.cgi. Con esta librería obtenemos funcionalidad FTP con el servidor remoto apache. Aparte, las añadimos al proyecto Eclipse en las propiedades del proyecto (*botón derecho del ratón>propiedades*) en *Java Buil Path* se añaden las librerías externas escogiendo el directorio donde están almacenados los ficheros jar de las librerías.

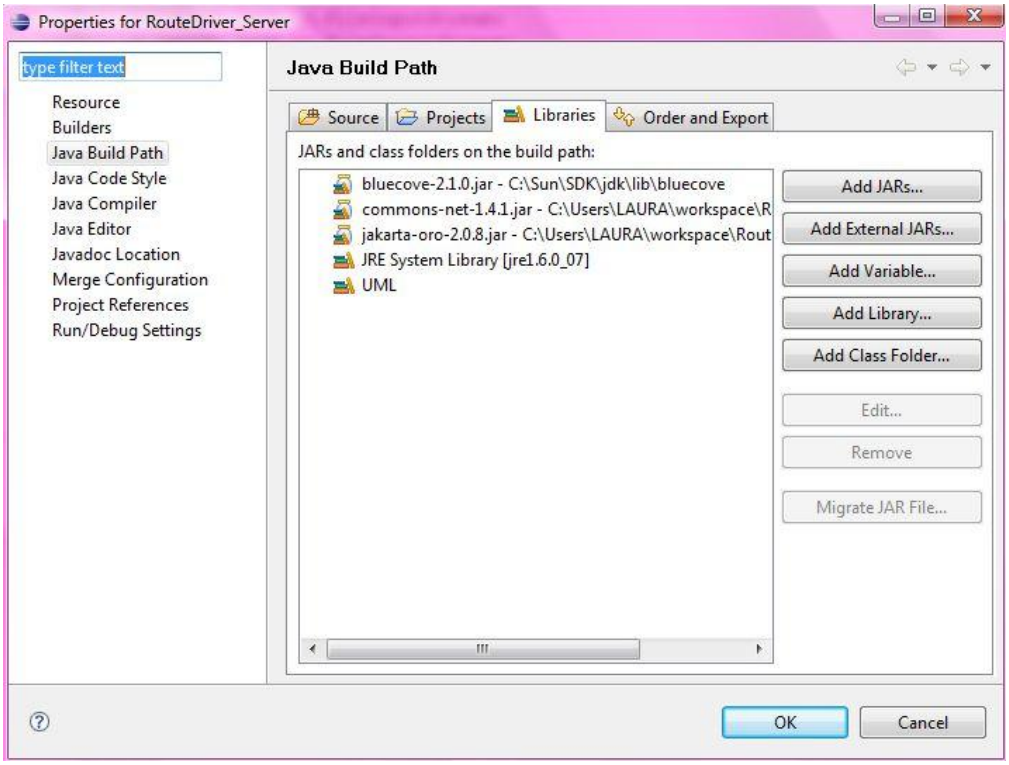


Figura A.3. Java Build Path en Eclipse

Notepad ++

Se trata de un editor para múltiples lenguajes y tareas. Simplemente lo utilizamos para escribir código, tanto Java ME como html y php.

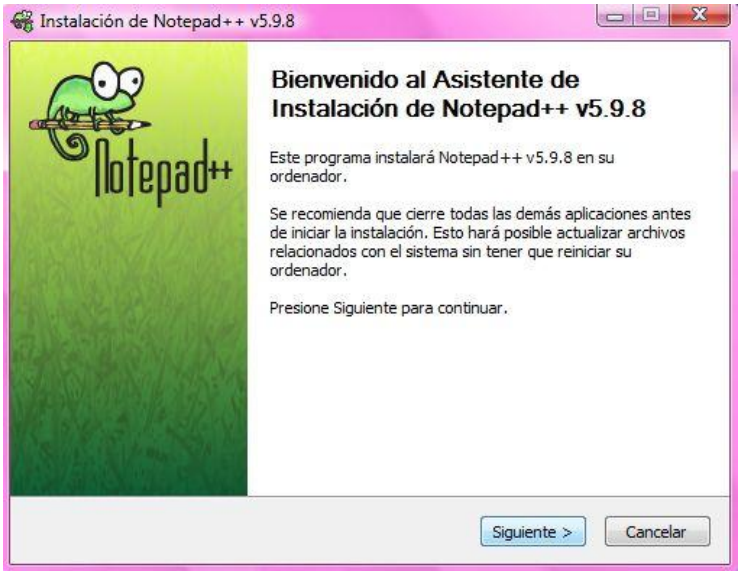


Figura A.4. Instalación Notepad ++

Su instalación es muy sencilla, es software de código libre que se puede descargar desde aquí: http://notepad-plus-plus.org/download/v5.9.8.html. Se ejecuta el instalable y se siguen los pasos de instalación. No tiene ninguna configuración previa para comenzar a utilizarlo.

Wireless Toolkit

Se trata de un software que proporciona un conjunto de herramientas para poder desarrollar aplicaciones para dispositivos reducidos, como móviles, utilizando J2ME. En ella vienen integradas librerías de Java ME SDK, pero aún así desde su web de descarga recomiendan descargarlo. Se descarga la versión **Java ME 3.0,** se trata de un ejecutable .exe (*sun_java_me_sdk-3_0-win.exe*) que nos llevará por el proceso de instalación siguiéndolo paso a paso.

La página oficial de Oracle se puede obtener el fichero de instalación e información: http://www.oracle.com/technetwork/java/download-135801.html. La versión que se ha utilizado es la 2.5.2. Una vez obtenido el ejecutable se procede a su instalación siguiendo los pasos necesarios. El paso más importante en la instalación es cuando pide la ruta del JDK instalado, si no lo encuentra automáticamente se puede seleccionar de las carpetas de nuestro equipo.

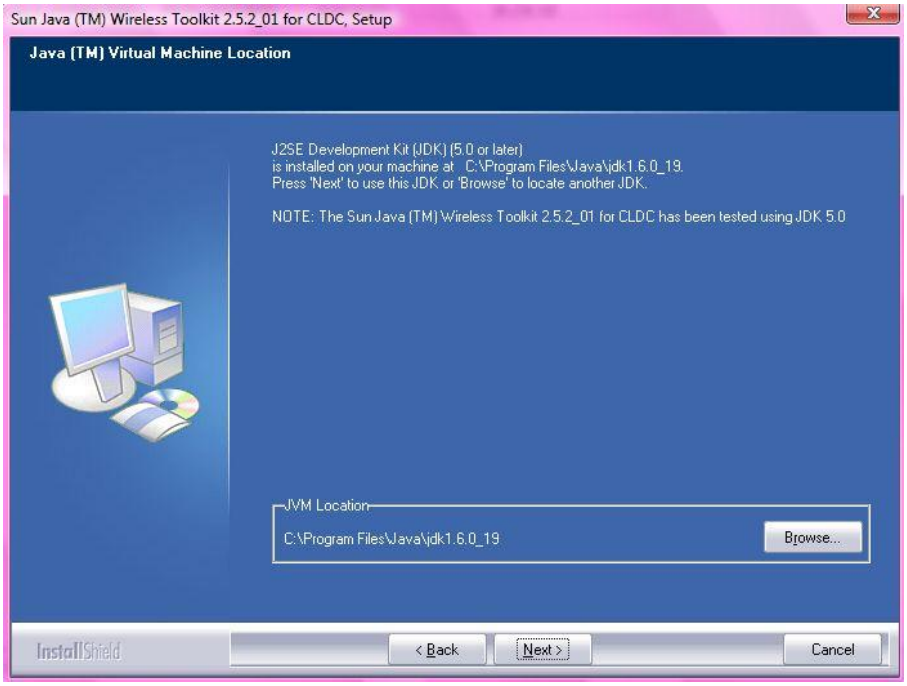


Figura A.5. Instalación Wireless toolkit

Hay que resaltar que no es una herramienta de edición, simplemente es para compilar, verificar y empaquetar las aplicaciones, con los ficheros jar y el descriptor jad. La herramienta viene con un gran número de ejemplos ilustrativos sobre varias funcionalidades de los teléfonos móviles implementadas en aplicaciones que se almacenan en el directorio *j2mewtk* creado en la instalación del programa, es en este directorio donde nosotros almacenaremos nuestras aplicaciones para compilarlas con el programa (en nuestro caso, *C:\Users\LAURA\j2mewtk\2.5.2\apps*).

SDK Nokia 6131 NFC

Para la simulación del terminal Nokia 6131 NFC podemos descargar su SDK desde la comunidad de desarrollo de Nokia. Incluye ejemplos de aplicaciones realizadas con dicha SDK, un emulador y las librerías exclusivas de uso NFC. La web de descarga es: http://www.developer.nokia.com/info/sw.nokia.com/id/ef4e1bc9-d220-400c-a41d-b3d56349e984/Nokia_6131_NFC_SDK.html.

Una vez descargado el zip, lo descomprimimos y ejecutamos el *setup.exe* para comenzar la instalación. En el asistente pregunta si queremos hacer la instalación integrada con Eclipse pero lo hacemos por separado ya que da alguno problemas esa integración.

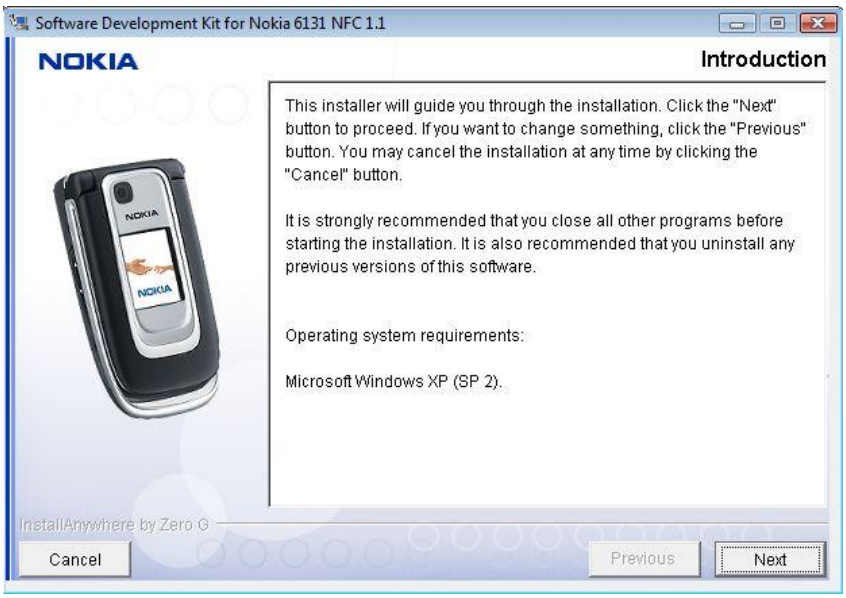


Figura A.6. Instalación Nokia 6131 SDK

Una vez concluida la instalación, tenemos el SDK en la ruta seleccionada, *C:/Nokia/Tools* en nuestro caso. Para tener mayor comodidad a la hora de realizar las pruebas, integramos el SDK con la herramienta Wireless Toolkit (WTK), de esta manera el emulador aparecerá como uno más en la lista de WTK y se podrán ejecutar las aplicaciones que vamos realizando directamente en el. Para conseguirlo, movemos la carpeta de la SDK recién instalada (*Nokia_6131_NFC_SDK_1_1*) al directorio *devices* de WTK (*C:\WTK2.5.2_01\wtklib\devices*), donde se encuentran todos los emuladores.

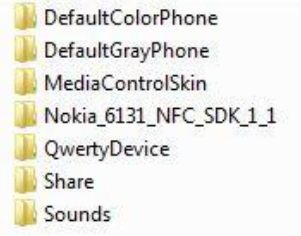


Figura A.7. Directorio emuladores WTK

PC Suite

PC Suite es una herramienta de Nokia para interactuar con el teléfono móvil conectado al ordenador. En nuestro caso lo hemos utilizado para poder instalar las aplicaciones creadas en el teléfono real. La versión instalada en nuestro caso es la 7.1, se puede descargar en la siguiente dirección: http://www.nokia.com/es-es/soporte/descargas/. Actualmente han cambiado la versión PC suite por otro software de similares características, el Nokia Suite. Para instalarlo se siguen los pasos del asistente.

Glosario

NFC	*Near Field Communication*
WIFI	*Wireless Fidelity*
PAN	*Personal Area Network*
RFID	*Radio Frequency Identification*
SIM	*Subscriber Identity Module*
J2ME	*Java 2 Micro Edition*
J2SE	*Java 2 Standard Edition*
PC	*Personal Computer*
FTP	*File Transfer Protocol*
ISO	*Organización Internacional de Estandarización*
HSDPA	*High Speed Downlink Packet Access*
Kbps	*Kilo Bits Per Second*
P2P	*Peer to Peer*
IFF	*Identification, Friend or Foe*
EAS	*Electronic Article Surveillance*
MIT	*Massachusetts Institute of Technology*
EPC	*Electronic Product Code*
ERO	*European Communications Office*
CEPT	*European Conference of Postal and Telecommunications*
ETSI	*European telecommunications Standard Institute*
OMS	*Organización Mundial de la Salud*
ISM	*Industrial, Scientific and Medical*
IP	*Internet Protocol*
OBEX	*OBject EXchange*
NDEF	*NFC Data Exchange Format*

RTD *Record Type Definitions*

JIS *Estándard Industrial Japonés*

CPU *Central Processing UNIT*

Referencias

[1] Introducción a NFC
http://es.wikipedia.org/wiki/Near_Field_Communication

[2] RFID y NFC
http://www.rfidpoint.com/noticias/¿que-es-near-field-communication-¿que-utilidad-tiene/

[3] Historia RFID
http://www.rfidpoint.com/preguntas-frecuentes/%C2%BFcual-es-el-origen-de-la-tecnologia-rfid-2/

[4] Arquitectura de RFID
http://es.wikipedia.org/wiki/RFID

[5] Aplicaciones y seguridad RFID
http://www.agpd.es/portalwebAGPD/revista_prensa/revista_prensa/2010/notas_prensa/common/julio/Guia_RFID.pdf

[6] Características de NFC
http://www.nfc-forum.org/aboutnfc/

[7] Arquitectura NFC
http://www.nfc-forum.org/news/june06_architecture_and_specs/nfc_architecture_schematic/
http://www.nfc-forum.org/events/oulu_spotlight/2009_09_01_Secure_Element_Programming.pdf

[8] Tipos de tarjetas NFC
http://www.nfc-forum.org/news/pr/view?item_key=2c0cb92de7d47bbbe7c99f13912b3307fc03c1c6

[9] NFC Forum
http://www.nfc-forum.org/aboutus/

[10] Ecosistema NFC
http://www.nfc-forum.org/aboutnfc/ecosystem/

[11] Descarga Contactless Communication API
http://jcp.org/en/jsr/detail?id=257

[12] NDEF
http://www.nfc-forum.org/specs/spec_list/#ndefts

[13] RTD
http://www.nfc-forum.org/specs/spec_list/#rtds

[14] Comparación entre tecnologías sin contactos
http://www.nfc-forum.org/aboutnfc/nfc_and_contactless/

[15] NFC en la vida real
http://www.nfc-forum.org/aboutnfc/nfc_in_action/
http://es.engadget.com/tag/Japan+Mobile+NFC+Consortium/

[16] Introducción Buetooth
http://www.bluetooth.com/Pages/Basics.aspx

[17] Historia y evolución Bluetooth
http://es.wikipedia.org/wiki/Bluetooth

[18] Arquitectura Bluetooth
http://gamersmafia.com/tutoriales/show/432
http://es.wikipedia.org/wiki/Bluetooth_%28especificaci%C3%B3n%29

[19] Pila protocolos Bluetooth: L2CAP
http://www.palowireless.com/infotooth/tutorial/l2cap.asp

[20] Pila de protocolos Bluetooth: RFCOMM
http://www.palowireless.com/infotooth/tutorial/rfcomm.asp

[21] Pila de protocolos Bluetooth: OBEX
http://es.wikipedia.org/wiki/OBEX

[22] Pila de protocolos Bluetooth: SDP
http://www.palowireless.com/infotooth/tutorial/sdp.asp

[23] Seguridad Bluetoth
http://www.radio-electronics.com/info/wireless/bluetooth/networks-networking-connections-pairing.php

[24] API J2ME Bluetooth
http://docs.oracle.com/javame/config/cldc/opt-pkgs/api/bluetooth/jsr082/index.html

[25] Bluecove
http://bluecove.org/

[26] Introducción a Java
http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29

[27] Historia y evolución de Java
http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29#Historia

[28] Características de Java
http://www.iec.csic.es/criptonomicon/java/quesjava.html

[29] J2SE
http://java.sun.com/j2se/1.5/

[30] Máquina virtual Java
http://docs.oracle.com/javase/1.5.0/docs/guide/vm/index.html
http://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java

[31] J2ME
http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

[32] Máquinas virtuales J2ME, configuraciones y perfiles
Libro “Java a Tope: J2ME” de Sergio Gálvez Rojas y Lucas Ortega Díaz. Soporte electrónico: http://www.lcc.uma.es/~galvez/ftp/libros/J2ME.pdf

[33] Introducción a FTP
http://es.wikipedia.org/wiki/File_Transfer_Protocol

[34] Esquema de funcionamiento FTP
http://www.falconmarbella.com/esigranada/dmdocuments/Punto_232_FTP.pdf

[35] Servidor/Cliente FTP
http://es.wikipedia.org/wiki/File_Transfer_Protocol#Servidor_FTP

[36] Modos de conexión FTP
http://technet.microsoft.com/es-es/library/cc771040%28WS.10%29.aspx
http://es.wikipedia.org/wiki/File_Transfer_Protocol#Modos_de_conexi.C3.B3n_del_cliente_FTP

[37] PHP
http://php.net/manual/es/intro-whatis.php
http://www.desarrolloweb.com/articulos/392.php

[38] Javascript
http://es.wikipedia.org/wiki/JavaScript

[39] HTML
http://www.desarrolloweb.com/articulos/que-es-html.html

[40] Evolución HTML
http://aprendiendoweb.com/2008/08/la-historia-del-html-%281989-2008%29

[41] HTML 5
http://www.desarrolloweb.com/articulos/que-es-html5.html

[42] CSS
http://www.w3c.es/divulgacion/guiasbreves/hojasestilo
http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada