Departamento de Telemática

Universidad Carlos III de Madrid

Proyecto de Fin de Carrera

Aplicación de evaluación basada en NFC (Near Field Communication)

Autor: Natalia Sánchez Moreno

Director: Mario Muñoz Organero

Índice de Capítulos

Índice de Figuras y Tablas

Agradecimientos

Este proyecto no sería lo que es sin la ayuda de todos cuantos han contribuido a la formalización del conocimiento que en él se condensa, particularmente la de mi profesor tutor, Mario Muñoz (por tu conocimiento, dedicación y esfuerzo hasta el final), la de mis profesores durante el curso, así como mis compañeros de clase y otros amigos, de los que me siento orgullosa y con los que siempre ha sido mas que gratificante compartir momentos, tanto en las imparticiones de clase y desarrollo de prácticas, como en la vida cotidiana.

La ilusión que mi familia ha compartido conmigo en todo momento, durante las muchas horas de dedicación al desarrollo del proyecto, ha sido suficiente aliciente para mantener la constancia e intentar aportar alguna luz sobre los nuevos avances de la tecnología en nuestro mundo actual.

Capítulo 1: Introducción y objetivos

1.1 Introducción

La existencia de móviles desde hace muchos años, ha hecho posible la comunicación, sin embargo, actualmente gracias al gran avance de las tecnologías, cabe la posibilidad de que dichos dispositivos contengan nuevas funcionalidades.

Entre estas funcionalidades implantadas en los móviles, se encuentra la estudiada en este proyecto, la cual es incorporada por Nokia (entre otros) a su repertorio. Se trata de un tipo de conexión inalámbrica que permite el intercambio de información entre dos dispositivos, esto permitirá realizar operaciones como: transmitir datos, realizar pagos, obtener entradas/tickets, entre otros.

Figura I.1: Dispositivos para el funcionamiento de una aplicación de pago.

Near Field Communication (NFC) es sin duda una tecnología con futuro. Cuenta con interesantísimas aplicaciones y por ello es altamente valorada por una ingente cantidad de compañías. Con tanto interés y respaldo será difícil no encontrar dispositivos NFC en cualquier lugar en un futuro no muy lejano. Al ser una tecnología inalámbrica de muy corto alcance, se facilita el intercambio de datos entre dispositivos a gran velocidad y permite tanto la lectura como la escritura.

Con el fin de fomentar esta tecnología y velar por la interoperabilidad entre dispositivos y servicios, surge una organización sin ánimo de lucro creada hace dos años, El NFC Forum. El foro cuenta con más de 100 miembros y entre ellos figuran empresas líderes en el campo de las comunicaciones móviles, tanto fabricantes de dispositivos como operadores de redes móviles, además de otros importantes nombres de otros sectores afines.

La principal ventaja de NFC es la seguridad, ya que las transacciones NFC sólo se pueden activar en un rango de acción muy limitado lo que limita seriamente el uso de la tecnología sin conocimiento del usuario. Los pagos seguros con el móvil son una de las principales aplicaciones de la tecnología.

Pero el principal interés de los operadores en NFC viene por el enorme potencial para fidelizar clientes mediante el desarrollo de nuevos servicios que no sean fácilmente transferibles a otros operadores. Por otro lado, los operadores no son los únicos interesados en NFC. Las compañías de marketing valoran especialmente NFC por su capacidad para desarrollar campañas personales y de fidelización, y los grandes fabricantes de electrónica de consumo lo consideran el interfaz ideal para dotar de un poderoso e intuitivo mecanismo de conectividad a cualquiera de sus dispositivos -cámaras de fotos, reproductores de música, consolas portátiles de juego, etc.

Todo apunta a que NFC será una de las tecnologías de moda que dominará el mercado, incluyendo la electrónica de consumo, en los próximos tiempos.

1.2 Objetivos del proyecto

En este proyecto, lo que se pretende es la realización de una aplicación para un móvil con la tecnología NFC (Near Field Communication).

Dicha aplicación se basa en la ejecución de un sistema de evaluación/calificación, en la cual, los alumnos no necesitan del papel y del bolígrafo tradicional para examinarse, simplemente se requiere de un dispositivo móvil que soporte dicha tecnología.

Figura O.1: Dispositivos para la prueba de la aplicación.

En esta figura podemos observar los dispositivos utilizados para la transmisión y recepción de los datos; en nuestro caso, sólo será necesario el teléfono móvil, y las tarjetas de memoria en las que guardar los datos introducidos tanto por el profesor, como por el alumno.

Sin embargo, con el fin de dar mayor fluidez al trabajo y evitar estar constantemente utilizando los dispositivos, se utilizó un simulador para realizar las pruebas oportunas. En la siguiente figura se muestra un ejemplo.

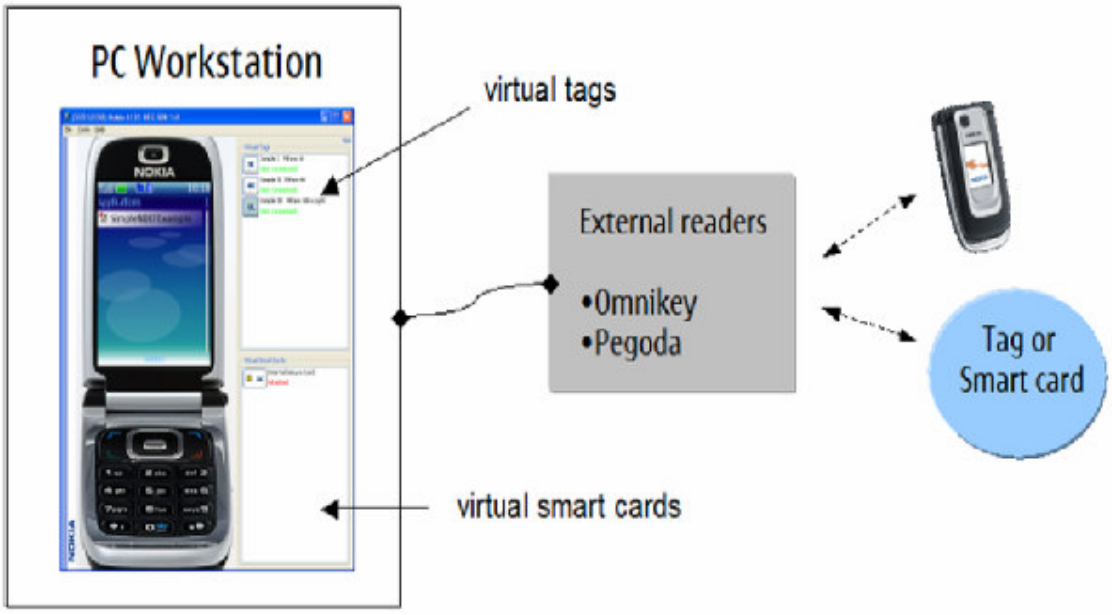


Figura O.2: Entorno de desarrollo y simulación.

Como ya se comentó en la introducción, esta tecnología soporta múltiples aplicaciones, sin embargo, el objeto de estudio, no será ninguna de ellas, sino que desarrollaremos una nueva, la cual puede suponer un gran avance en el futuro para el desarrollo de sistemas de evaluación y calificación.

Todos estos dispositivos de los que hemos hablado y mostrado mediante las imágenes, están disponibles comercialmente porque están integrados en el mercado global debido a su empleo en el entorno de las tecnologías de la información y la comunicación (*TIC*). Sin embargo podemos decir que estos dispositivos presentan un coste no demasiado elevado, por lo cual es bastante asequible para los usuarios.

En este trabajo se pretende mostrar cómo mediante la tecnología NFC se proporciona una solución económica, versátil y fácilmente escalable para los sistemas de comunicación y transmisión de datos inalámbricos, permitiendo así aumentar su uso.

Por otro lado, a través de esta tecnología lo que se pretende es reducir el riesgo de seguridad, que con otos sistemas inalámbricos pudiéramos encontrar, ya que las transacciones NFC sólo se pueden activar en un rango de acción muy limitado, lo que limita seriamente el uso de la tecnología sin conocimiento del usuario.

La documentación de este proyecto se organiza de la siguiente manera. En el capítulo 1 se analiza la tecnología NFC como medio de transmisión inalámbrico y sus estándares, para luego detallar su implantación en el mercado.

Capítulo 2: Tecnologías básicas utilizadas

2.1 La Tecnología NFC

2.1.1 Breve introducción acerca de NFC

NFC son las siglas de Near Field Communication. Una tecnología inalámbrica de alcance ultracorto de la que se está hablando mucho en el entorno móvil.

Aprobado como estándar ISO en 2003, su uso ya ha tenido recorrido en dispositivos como llaves para coche, tarjetas de identificación o tickets electrónicos.

La principal diferencia entre otras tecnologías inalámbricas es que el alcance es tan corto, que se necesita que los dos dispositivos a interaccionar estén en contacto durante un instante. Pese a que esta característica pueda parecer una limitación, es en realidad la clave.

Al contrario de lo que ocurre con los servicios por RFID o Bluetooth, basados en el descubrimiento de la presencia del dispositivo en la proximidad. Estirar el brazo para acercar el móvil con NFC hacia el detector es en sí una afirmación clara de nuestra voluntad por autentificarnos, por pagar nuestra cesta de la compra o simplemente transferir un contacto.

Near Field Communication (NFC) ofrece un potencial tremendo, no sólo porque puede ser desplegado para la adopción en masa, sino por el número de maneras diferentes en que puede ser usado para hacer la vida más fácil.

NFC en un teléfono móvil hace posible muchas más tareas del día a día a los consumidores. Basado en un pequeño rango de conectividad wireless, NFC esta diseñado para una intuitiva, simple y segura interacción entre dispositivos electrónicos.

La comunicación NFC está habilitada en dos dispositivos compatibles a pocos centímetros el uno del otro o para dos dispositivos que literalmente están tocándose el uno y el otro.

Los teléfonos móviles son vistos como herramientas poderosas para el uso de la tecnología NFC porque son capaces de "bajar" nuevas piezas de información, como cargar una tarjeta para el transporte público, escuchar nuevas canciones, obtener información sobre boletos sobre espectáculos y utilizar llaves electrónicas.

Mike Roberts, el analista principal de Informa Telecoms y Mobile, dijo que era una decisión inteligente para los operadores de telefonía móvil.

"Los operadores necesitan aumentar sus ganancias por llamados telefónicos y necesitan asegurarse que los teléfonos móviles sigan siendo útiles a la gente", explicó Roberts [2].

Como otros teléfonos, el Nokia 6131 NFC cambia la forma de interactuar con los dispositivos y servicios en sus alrededores. La comunicación básica de NFC, entre el teléfono y otro dispositivo permite a los consumidores usar el teléfono como tarjeta de viaje, tarjeta de crédito o para programas de lealtad.

Para el estudio de este proyecto se utilizará este dispositivo, el Nokia 6131 NFC, pero existen otros dispositivos que también soportan esta tecnología.

Es en este capítulo de la memoria, donde trataremos los antecedentes de NFC, incluyendo una explicación de lo que la tecnología puede lograr, cómo funciona, y en qué podría ser utilizado. Se destaca también el papel de Nokia en NFC y las compañías de apoyo para su adopción en masa.

2.1.2 Qué es Near Field Communication (NFC)

NFC es una tecnología inalámbrica de corto alcance que permite la comunicación que tendrá lugar entre dispositivos que, o bien se tocan o están juntos momentáneamente.

La tecnología trabaja a través de la inducción magnética y opera en una banda de radiofrecuencia sin licencia. Las tarjetas (Tags) son insertadas en el dispositivo (estos pueden ser dispositivos móviles como, los teléfonos móviles, o PDA's, o estaciones NFC como un ticket o tarjeta de pago). NFC permite a los dispositivos mantenerse juntos para compartir información, ya sea en una dirección o en otra.

NFC está basado en la tecnología de Identificación por radiofrecuencia (RFID), que es compatible con más de los sistemas de transporte sin contacto y soluciones de venta de entradas, que son cómodamente usados alrededor del mundo para permitir de manera rápida y sin problemas circulación de personas dentro de los sistemas de transporte público o compra de entradas. NFC es una tecnología de plataforma abierta y fue aprobada como un estándar global ISO/IEC en Diciembre de 2003.

Figura 2.1.1: Funcionamiento del sistema NFC

La comunicación NFC se realiza entre dos entidades *(peer-to-peer)*. Este protocolo trabaja en la banda de los 13,56 MHz, esto provoca que no se aplique ninguna restricción y no requiera ninguna licencia para su uso. El alcance de funcionamiento de este estándar está por debajo de los 20 cm.

NFC soporta dos modos de funcionamiento, todos los dispositivos del estándar NFCIP-1 deben soportar ambos modos:

Estos dos modos de funcionamiento son [6]:

- ✓ Activo
- ✓ Pasivo

Pasivo: Sólo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.

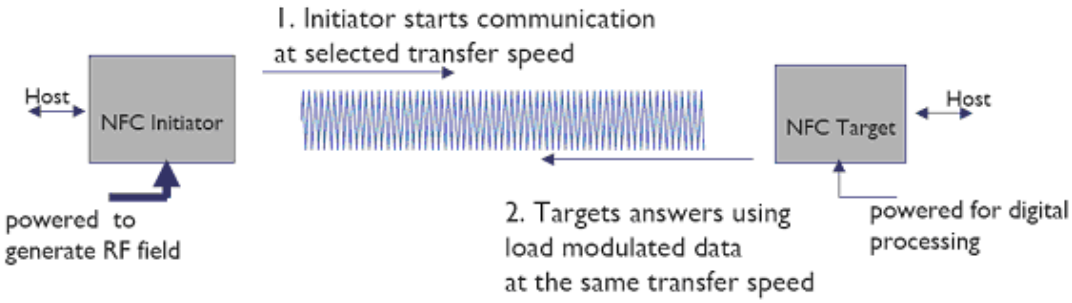


Figura 2.1.2: Esquema del modo de funcionamiento pasivo

Activo: Ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos. Ambos dispositivos necesitan energía para funcionar.

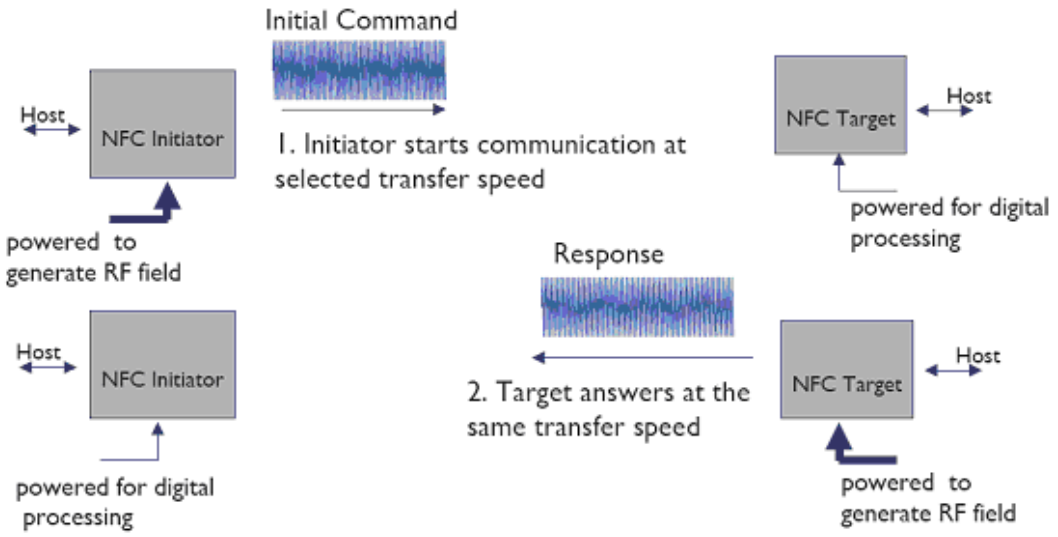


Figura 2.1.3: Esquema del modo de funcionamiento activo

NFC se caracteriza porque a la hora de realizar una comunicación entre un dispositivo y otro, se siguen una serie de fases:

- Descubrimiento
- Autenticación
- Negociación
- Transferencia
- Reconocimiento

Con un móvil equipado con la tecnología NFC, los usuarios pueden acceder fácilmente a servicios o realizar operaciones en las distintas funciones de su dispositivo.

2.1.3 Por qué es importante NFC

NFC es una importante tecnología por un número de razones:

1. **Alcance y disponibilidad:** NFC tiene el potencial para ser integrado en cada uno de los teléfonos del mundo. Esto daría a la tecnología un potencial amplio como teléfono móvil. Mediante la integración de la tecnología NFC en los móviles, los usuarios pueden obtener acceso a un número de nuevos servicios a través de su teléfono.
2. **Variedad de uso:** NFC puede ser usado para varias tareas, desde pago de bienes, hasta compra de billetes y emparejamiento de los dispositivos para el intercambio de información o el descubrimiento de nuevos servicios. Ejemplos de estas aplicaciones se comentarán mas adelante en el documento.
3. **Fácil de usar:** Porque NFC sólo requiere que dos dispositivos se toquen con el fin de comunicarse, NFC puede simplificar muchas cosas, desde abrir un navegador Web en un teléfono móvil, hasta emparejar dos dispositivos de Bluetooth automáticamente para acceder de manera inalámbrica simple y fácilmente.
4. **Seguridad:** NFC requiere de un usuario para activarlo manualmente o mantener su dispositivo móvil junto a otro móvil o junto a la estación NFC para activar un servicio o para compartir información. Al hacerlo la tecnología requiere del usuario para hacer una acción positiva que confirme la transacción o el intercambio. Además es posible construir múltiples niveles de seguridad en un dispositivo NFC.
5. **Servicios de valor añadido:** NFC permite a los usuarios acceder a los servicios de valor añadido que de otro modo no están disponibles en una obtención de billetes o en un pago con tarjeta. Como el caso de los usuarios de los servicios móviles de prepago, que pueden acceder a su saldo actual a través del sistema de menú del teléfono, por lo que los usuarios de un teléfono NFC podrán ser capaces de acceder a información similar a través de su dispositivo. Además los

dispositivos NFC pueden acceder a la red móvil para añadir crédito al dispositivo cuando se agota o esta bajo, o alternativamente, establecer una fecha determinada para añadirlo cada semana o cada mes.

6. **Infraestructura:** NFC es compatible con la estructura sin contactos, usada como una plataforma para la obtención de billetes, transporte y pago en todo el mundo. Los dispositivos móviles NFC pueden fácilmente ser fabricados con los principales sistemas de transporte que no usa contactos para acceder al servicio, por ejemplo, los que se basan en el sistema MIFARE. Esto es también compatible con el popular modo de pago con tarjeta de crédito y debito que están siendo implantadas en muchos países. El despliegue de NFC en los actuales entornos de contactos es muy sencillo. Los usuarios conocen cómo es el sistema de trabajo y mucha de la infraestructura ya está en su lugar. El despliegue de NFC es una extensión de los servicios que ya existen, pero es mayor con el elemento adicional de un interfaz de usuario del teléfono móvil y una conexión a Internet.

2.1.4 El papel de Nokia en NFC

Nokia reconoce que su inversión en investigación y desarrollo es uno de sus factores de éxito. La compañía se compromete al desarrollo y obliga a las tecnologías que permiten transformación y crecimiento, la convergencia de Internet y las industrias de telecomunicaciones.

Como el mayor fabricante mundial de teléfonos móviles, el ADN de Nokia está intrínsecamente conectado a la simple facilidad de uso de sus dispositivos. Nokia tiene un interés en la NFC como una tecnología que puede añadir valor y funcionalidad, a través del móvil a la vista de las personas.

La compañía reconoce que con el fin de obtener éxito, NFC requerirá de una asociación de un número de miembros diferentes para crear una infraestructura sin que NFC pueda florecer. La clave de la asociación necesitará incluir órganos de pago, bancos, operadores móviles, tarjetas SIM manufacturadas, sistemas integrados, la comunidad de desarrolladores, etc. Todas estas partes tienen un papel para conseguir el éxito comercial de NFC.

Para terminar, Nokia ha sido una de las co-fundadoras del Forum NFC en 2004. El Forum es una asociación de industria no lucrativa que promueve la tecnología NFC. A día de hoy el Forum tiene más de 140 miembros. Más información sobre el Forum NFC se puede encontrar en www.nfc-forum.org.

2.1.5 Aplicaciones de NFC

NFC puede utilizarse de varias maneras. Algunas de estas aplicaciones se describen a continuación [3].

2.1.5.1 Tocar para conectarse

Con NFC seremos capaces de recoger información de nuestro entorno. NFC permite a los dispositivos móviles leer información cargada en las tarjetas NFC. Por ejemplo carteles, señales de las paradas de bus, medicinas, certificados, alimentos envasados y muchos más.

Añadiendo tarjetas NFC a los carteles y anuncios de revistas, los lectores pueden tener acceso a los servicios móviles ya existentes, como las líneas directas, sms y red o Internet basado en contenidos y servicios con teléfonos NFC.

Una nueva aplicación de NFC se aprovecha de la capacidad del intercambio de los datos a fin de que la simple transferencia de datos de un dispositivo a otro se consiga a través de que los teléfonos se toquen.

Un número de actividades asociadas con la transferencia de datos entre dispositivos, requiere cierto grado de interacción con el usuario para que están se realicen. Por ejemplo, muchos dispositivos de Bluetooth requieren de un proceso de emparejamiento que tendrá lugar antes de que los dispositivos se puedan utilizar. Mientras que esto es relativamente sencillo, la funcionalidad puede no estar inmediatamente accesible sin el menú del sistema y el proceso de emparejamiento puede inhibir en el uso de la tecnología.

La más actual especificación básica para el estándar de Bluetooth incluye la capacidad de emparejar dispositivos vía NFC. Simplemente activando el Bluetooth en ambas partes, búsqueda, espera, emparejamiento y autorización en ambos lados que puede ser reemplazado tocando los dos dispositivos que están juntos. Esto proporciona al usuario una manera simple y participativa para vincular los dispositivos con Bluetooth.

De manera similar pueden los usuarios de NFC obtener acceso a la red inalámbrica. En lugar de la larga duración del proceso de búsqueda del punto de acceso, un usuario puede simplemente tocar un punto de red LAN inalámbrica compatible con NFC y todo el proceso podría ser automatizado, incluyendo el pago de cualquier coste de la billetera virtual del dispositivo.

Como el dispositivo móvil cada vez más se convierte en el hogar de los contenidos digitales, se hará más apremiante la posibilidad de compartir fácilmente este contenido.

NFC puede permitir un entorno en el que la gente pueda tocar los dispositivos para compartir tarjetas de visita, toque para descargar sus fotografías a una impresora, o contacto para compartir su música con un amigo.

La tecnología NFC tiene el potencial de impactar en la comercialización y las promociones de la industria, desde que al tocar para recibir información de los usuarios de manera activa tiene un interés en un producto o servicio [1].

2.1.5.2 Tocar para emitir billetes

Las tarjetas de contacto comienzan una nueva era para el transporte y venta de entradas con velocidad y flexibilidad. Con la habilitación de NFC en los teléfonos móviles, los usuarios pueden comprar tickets, recibirlos en su celular e ir a través de la vía rápida, mientras otros esperan. Un balance puede ser chequeado online o los tickets pueden ser cargados remotamente.

Típicamente, un usuario compra una tarjeta de plástico con un cierto valor monetario en un chip incrustado en la tarjeta. A medida que el usuario accede al sistema de transporte público, el costo del pasaje está tomado de la tarjeta, lo que deja un nuevo balance de la tarjeta. Una vez que la tarjeta no tiene valor, el usuario puede optar entre dejarla o recargar el saldo mediante la adición de más dinero a la tarjeta para permitir el viaje.

Este planteamiento tiene grandes beneficios en términos de facilidad de uso y la velocidad de acceso a los sistemas de transporte. No es necesario comprar una tarjeta cada día. Mediante la recarga online y tarifas de acceso mensual, también se garantizan menos colas en las cabinas de venta de billetes.

2.1.5.3 Tocar para pagar

NFC proporciona tickets y tarjetas sin contacto para ser sostenidos en los teléfonos como Nokia 6131 NFC. En lugar de llevar los billetes de transporte, lealtad y tarjetas de crédito por separado, los usuarios pueden elegir almacenar varias tarjetas en sus teléfonos NFC. El Nokia 6131 NFC tiene una funcionalidad similar a las tarjetas inteligentes estándar de contactos que son usadas en todo el mundo en tarjetas de crédito y tickets para sistemas de transporte público.

Una vez que una aplicación, como por ejemplo una tarjeta de crédito, ha sido aprovisionada de seguridad para un teléfono NFC, los clientes pueden pagar simplemente apoyando su teléfono en el lector de venta. En ese mismo momento el celular ofrece la mayor seguridad, como en cualquier transacción se requiere una confirmación de usuario. Para mayor comodidad un historial de la transacción está fácilmente alcanzable y los consumidores pueden cogerlo mediante algunos programas sin ocupar espacio en su billetera.

Las aplicaciones de pago y expedición de billetes se almacenarán en elemento seguro del dispositivo NFC. El elemento seguro es un pequeño chip capaz de almacenar múltiples aplicaciones, como por ejemplo la tarjeta SIM, tarjeta de memoria segura, u otras adicionales tarjetas inteligentes incluidas en el dispositivo NFC.

Muchas de las principales ciudades del mundo usan el sistema de pago sin contacto dentro de la infraestructura de transporte. Estos sistemas se basan en las tarjetas del sistema de Identificación por Radiofrecuencia (RFID) para facilitar el acceso a los servicios de transporte sostenible y la rapidez y comodidad de pago.

Mientras que el servicio funciona bien, las tarjetas inteligentes utilizadas en el sistema no son realmente inteligentes. Aquí es donde NFC puede añadir valor a esta aplicación existente.

Mediante la sustitución de una tarjeta inteligente con un dispositivo móvil NFC, los usuarios pueden acceder a todos los servicios que tienen con una tarjeta inteligente, con la funcionalidad añadida de una interfaz de usuario de proporcionar información adicional, así como el acceso a través de la red móvil para recargar la tarjeta mediante la facilidad de tocar un botón. El teléfono también puede utilizar la tecnología actual de redes de telefonía móvil para acceder a la última información sobre el tráfico o información en el mapa.

El usuario de un dispositivo móvil habilitado con NFC no se limita necesariamente a la recarga de una tarjeta. Es posible añadir a la tarjeta de crédito un elemento NFC, que permite al usuario una forma de pago compatibles en cualquier estación o punto de venta.

También es posible añadir varias tarjetas de crédito o débito a un dispositivo móvil NFC. En este escenario, el dispositivo móvil se convierte en un monedero virtual, llevando un número de tarjetas diferentes, algunas de crédito, algunas de débito, algunas de lealtad en el dispositivo [1].

En última instancia, el dispositivo móvil NFC puede sustituir la necesidad de un usuario para ejecutar una cartera a todos los efectos -la creación de un servicio central para la transacción en efectivo, tarjeta de débito y tarjeta de crédito para las compras-, todo desde un dispositivo móvil.

2.1.5.4 Gestión de identidad y procesos de negocios

Casi cada oficina o fábrica de la base trabajadora está obligada a llevar una etiqueta de identidad para acceder a los locales de trabajo. Como las empresas se hacen más complejas y de carácter mundial, muchos trabajadores necesitan tener acceso a múltiples locales.

La gestión de este proceso puede ser complicado, más aún en ambientes donde hay diferentes niveles de seguridad para los diferentes trabajadores.

NFC puede permitir que la gestión de la identidad que se añade a un dispositivo móvil, proporcione una única solución integrada para la gestión de la identidad. El dispositivo móvil se puede utilizar para proporcionar acceso a determinados lugares y por supuesto, negarlo a otros. Importante, el acceso se puede actualizar a través de la red inalámbrica o móvil, es decir, los trabajadores no están obligados a visitar un sitio físicamente para cambiar su perfil de usuario.

Un guardia de seguridad podría entrar en un contacto existente en la habitación, proporcionando un sendero digital de sus movimientos durante una patrulla. Un mensajero podría tocar en la entrega de un paquete local para recibir información sobre el tráfico y orientaciones del siguiente punto de recogida [1].

2.1.5.5 Nuevas aplicaciones

Muchas aplicaciones en el campo de la NFC son extensiones para actualizar soluciones. Pago y soluciones de venta de entradas están ampliamente disponibles en todo el mundo y, sobre todo, son compatibles con NFC.

Extender estas aplicaciones a un dispositivo móvil debidamente equipado serán los primeros pasos en el camino de la NFC.

Sin embargo, para limitar a la tecnología actual de los escenarios es a la vez poco realista y carece de imaginación. Más probable es que un número de solicitudes se desarrollarán a medida que la tecnología esté más implantada. A medida que más personas utilizan NFC y que se vuelve más ubicuo, de igual manera se generarán aún más las aplicaciones. Para lograrlo, necesita un entorno que se creará dentro de los cuales puede tener éxito NFC [1].

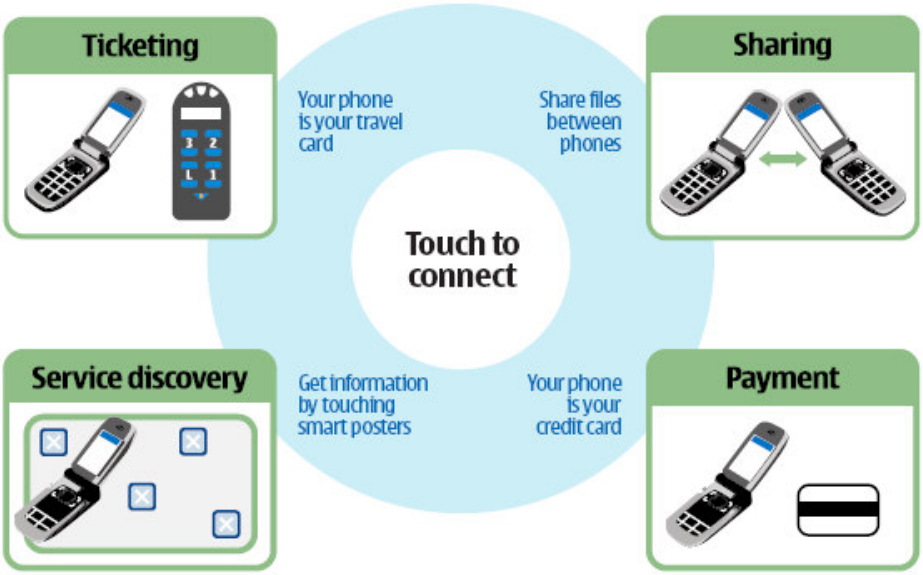


Figura 2.1.4: Aplicaciones de NFC (Fuente: [1] p.4)

2.1.6 Qué se requiere para que NFC tenga éxito

1. Factores clave para el éxito de NFC

El éxito de NFC depende de la creación de un ya complejo ambiente interoperable soportado por un número de diferentes partes.

Primero es necesario que haya dispositivos móviles que soportan el sistema. Esto se basa en la producción de los fabricantes de teléfonos móviles compatibles con los dispositivos NFC. Estos también tendrán que provenir de distintos vendedores, que ofrecen la opción de mercado y la diferenciación. El primer dispositivo móvil con NFC está disponible ahora en el mercado.

Operadores de red móvil necesitarán soporte NFC. Acceso a servicios adicionales de datos y la posibilidad de que estos ingresos se puedan aportar a través de la red móvil, es un valor crítico de la NFC y el apoyo de la comunidad de operadores de telefonía móvil tiene la obligación de facilitar esto.

Bancos y compañías de tarjetas de crédito, junto con los operadores de transporte también tendrán que comprometerse con NFC. Otorgando un alto nivel de seguridad real y percibida, será esencial para el éxito de NFC y los bancos de crédito y las empresas tener un papel fundamental que desempeñar, garantizando el despliegue de los servicios de pago a los dispositivos NFC.

Los minoristas, desde tiendas a los restaurantes, desde la prensa y hasta las tiendas de café, tendrán que apoyar y mejorar su actual oferta con un elemento NFC. De la misma manera que se desarrolló la implantación de soluciones de chips y PIN en equipos del comercio minorista, se hará el desarrollo de NFC en el nuevo hardware.

El despliegue de la tecnología de NFC al por menor es probable que tenga lugar en el tiempo. Los minoristas cerca de las estaciones de transporte

público puede ser uno de los primeros en adoptar la tecnología para prestar un servicio a sus clientes que ya están utilizando las soluciones de transporte sin contacto. La adopción en masa se lleva a cabo, con un número creciente de dispositivos disponibles en el comercio, el despliegue de NFC en los puntos de venta se convertirán en ubicua.

Finalmente, la comunidad de desarrolladores es también de importancia crítica para el desarrollo de la NFC. Una activa comunidad de desarrolladores con las herramientas adecuadas y de manera rápida y eficaz pueden aportar nuevas soluciones al mercado puede añadir impulso a la tecnología y ofrecer fácil acceso NFC en las aplicaciones de los usuarios finales [1].

2. Seguridad

La seguridad es una condición decisiva para el pago mediante NFC y la expedición de billetes y, como tal, es importante para entender la seguridad sin un entorno habilitado con NFC.

Los teléfonos móviles tienden a venir equipados con un código que puede ser activado por el usuario. Mientras que muchos usuarios lo activan para garantizar seguridad en su teléfono, un dispositivo móvil que permite NFC, especialmente uno que podría incorporar una serie de tarjetas de crédito y débito, debe tener un mayor nivel de seguridad para tranquilizar a los usuarios contra el mal uso.

Inherentemente, la tecnología NFC es construida con un alto nivel de seguridad. Los usuarios pueden activar una serie de opciones con el fin de almacenar sus datos en un entorno seguro.

Un usuario puede establecer más allá del límite financiero con un código requerido para autorizar el pago, o también un usuario puede disponer de los elementos de la billetera del teléfono que son encendidos durante tan solo dos minutos. Esto permite a un usuario hacer la compra y así estar seguro de que el dispositivo es desactivado poco después. Para la transferencia de dinero de un elemento del dispositivo a otro, un usuario también debería ser capaz de limitar la cantidad de riesgos asociados con la perdida de un teléfono con relativa pequeña cantidad. Por lo cual las aplicaciones que requieren tarjetas de crédito y/o débito podrán ser activadas sólo cuando el usuario introduzca el código.

Una tarjeta de debito/crédito habilitada con NFC o una aplicación de emisión de billetes es sostenida con un elemento de seguridad en los dispositivos móviles. Usando la misma tecnología que en un chip o en una tarjeta PIN, estos elementos de seguridad están certificados y apoyados por la industria de pagos, proporcionando tan alto nivel de garantía a los usuarios finales como en las tarjetas de crédito tradicionales.

El elemento seguro puede o bien colocarse en un chip en el dispositivo móvil, o en una tarjeta SIM, o en un dispositivo con tarjeta de memoria. Potencialmente, podría incluso ser un elemento de seguridad tanto en la tarjeta SIM como en el teléfono. Esto aseguraría que los derechos del teléfono y la tarjeta SIM se usaran para cualquier transacción, añadiendo más nivel de seguridad.

Finalmente, a diferencia de una pérdida de la billetera, un teléfono móvil se comunica regularmente con la red móvil para que pueda funcionar. En un momento dado, la red móvil sabrá donde está el móvil con un grado de precisión. Lo que es más, el teléfono puede ser deshabilitado en pocos momentos, moviendo el conjunto de la tarjeta de crédito a un nuevo dispositivo. [1]

2.1.7 NFC Forum

El Near Field Communication (NFC) Forum es una asociación industrial sin animo de lucro fundada por NXP Semiconductors, Sony Corporation y Nokia para regular el uso de la interacción inalámbrica de corto alcance en la electrónica de consumo, dispositivos móviles y los PCs.

El NFC Forum promueve la implantación y la estandarización de la Tecnología NFC como mecanismo para la interoperabilidad entre dispositivos y servicios.

Para conseguir esto, se encarga de:

- ✓ Desarrollar especificaciones basadas en estándares.
- ✓ Asegurarse del uso de las especificaciones del NFC Forum.
- ✓ Trabajar para que los productos con tecnología NFC cumplan con las especificaciones del NFC Forum.
- ✓ Educar a los consumidores y las empresas respecto de la Tecnología NFC.

El NFC Forum ha establecido un estándar en la que se registra un formato común para poder compartir datos entre los dispositivos NFC entre sí y/o entre los dispositivos y las etiquetas NFC.

- **NFC Data Exchange Format (NDEF)**
  Especifica un formato común y compacto para el intercambio de datos.

- **NFC Record Type Definition (RTD)**

Especifica tipos de registros estándar que pueden ser enviados en los mensajes intercambiados entre los dispositivos NFC.

- **Smart Poster RTD**

  Para posters que incorporen etiquetas con datos (URLs, SMSs o números de teléfono).

- **Text RTD**

  Para registros que solo contienen texto.

- **Uniform Resource Identifier (URI) RTD**

  Para registros que se refieren a un recurso de Internet

	Type 1	Type 2	Type 3	Type 4
RF Interface	ISO 14443 A-2	ISO 14443 A-2	FeliCa (ISO 18092, passive communication mode at 212 kbits/sec)	ISO 14443-2
Initialization	ISO 14443 A-3	ISO 14443 A-3	FeliCa (ISO 18092, passive communication mode at 212 kbits/sec)	ISO 14443-3
Speed	106 kbits/sec	106 kbits/sec	212 kbits/sec	106-424 kbits/sec
Protocol	Specific Command set	Specific Command Set	FeliCa protocol	ISO 14443-4 ISO 7816-4 commands
Memory Size	Up to 1 KB	Up to 2 KB	Up to 1 MB	Up to 64KB
Cost (memory dependent)	Low	Low	Moderate	Moderate
Use cases	Tags with small memory for single application		Flexible tags with larger memory offering multi-application capabilities	

Figura 2.1.5: Estándares soportados por NFC para compartir datos.

2.2 Otros contenidos de Nokia 6131 NFC

2.2.1 Información general

El teléfono dispone de numerosas funciones muy prácticas para el día a día, como los mensajes de texto y multimedia, una agenda, un reloj, una alarma, una radio, un reproductor de música y una cámara integrada. El teléfono también admite las siguientes funciones [8]:

- ✓ Servicio en línea plug and play para obtener los ajustes de configuración.
- ✓ Efectuar pagos y obtener billetes con NFC.
- ✓ NFC para leer y transmitir la información a las etiquetas de servicio.
- ✓ Pulsar para hablar.
- ✓ Tarjeta de memoria microSD para ampliar la capacidad de memoria del teléfono.
- ✓ Espera activa.
- ✓ Mensajería de audio.
- ✓ Mensajería instantánea.
- ✓ Aplicación de correo electrónico.
- ✓ Marcación mediante voz mejorada.
- ✓ Contactos con información de presencia.
- ✓ Plataforma Java 2 Micro Edition (J2METM).
- ✓ Descarga de contenido
- ✓ Actualizaciones de software
- ✓ Códigos de acceso
    - o Código de seguridad.
    - o Códigos PIN.
    - o Códigos PUK.
    - o Contraseñas de restricciones.
    - o Clave de acceso para elementos de seguridad.

2.2.2 Aspectos Genéricos

Una vez hemos comprobado la gran funcionalidad que nos proporciona el sistema de comunicación NFC, comentaremos a continuación algunos contenidos genéricos que se necesitan conocer.

- ***¿Cómo activar y desactivar la detección de tarjetas?***

  Para activar o desactivar la detección de etiquetas de servicio, seleccione: **Menú** > *NFC* > *Detección etiquetas*.

  Para ahorrar carga de la batería, el dispositivo apaga automáticamente la iluminación de la pantalla y establece la detección de etiquetas en modo de espera. La iluminación de la pantalla y la detección de etiquetas vuelven a activarse cuando se realiza alguna acción sobre el dispositivo.

- ***¿Dónde está localizada la antena en Nokia 6131 NFC?***

  En la parte superior de la carcasa trasera del teléfono. No está alrededor de la pantalla, pero ligeramente por encima de ella cuando se sujeta el teléfono en posición vertical.

Figura2.2.1: Localización de antena en el Nokia 6131 NFC

- ***¿Es más corta la distancia de lectura en Nokia 6131 NFC que en Nokia 3229 NFC?***

  Esto depende de algunos factores, y por lo tanto no hay una simple respuesta a esto. No debería haber mucha diferencia, pero en el mundo real las mediciones dirán la verdad.

- ***¿Cómo se maneja la administración de energía y cómo compararlo con el Nokia 6131 NFC?***

  La administración de energía en el 6131 NFC esta considerablemente mas avanzada que en el 3220 NFC, por lo cual, en general, se puede esperar un mejor rendimiento. No obstante, esto depende de sus patrones de uso.

- ***¿Qué tipo de tarjetas soporta JSR-257 implementado en Nokia 6131 NFC?***

  Los tipos de tarjeta soportados son:

  - ✓ TargetType.ISO14443_CARD para ISO 14443-4 que complementa el acceso de las tarjetas inteligentes usadas en comandos APDU.
  - ✓ TargetType.NDEF_TAG para una tag que contiene datos formateados de NFC Forum.
  - ✓ TargetType.RFID_TAG para tarjetas RFID en general.

Figura 2.2.2: Ejemplo de tarjetas RFID en general [7]

- ***¿Pueden terceras partes añadir cosas al menú de configuración de NFC?***

  No. Esto es en el teléfono S40, el cual no tiene flexibilidad para códigos nativos como es S60.

- ***¿Puede el midlet ser puesto en un área segura y con ello no ser borrado por el usuario?***

  El usuario tiene control sobre los midlets en el teléfono, incluyendo la habilidad de borrarlos.

2.2.3 Elementos seguros y tarjetas inteligentes

En este apartado se comentan los elementos de seguridad de los que dispone el Nokia 6131 que soporto la tecnología NFC.

- ***¿Cuales son los detalles técnicos del elemento de seguridad integrado en Nokia 6131 NFC?***

  El sistema operativo es Giesecke & Devrient`s (G&D) SmartCafé Expert 3.1.

  El elemento seguro consiste en una tarjeta inteligente Java y un área Mifare 4K para la emulación de la tarjeta. El área de la tarjeta Java es compatible con Global Platform 2.1.2 y compatible con Java Card 2.2.1.

  Los applets en el área de Java Card pueden acceder al área Mifare 4K con librerías especificas de G&D proporcionadas por SmartCafe Professional Toolkit.

- ***¿Cuál es el tamaño de memoria del elemento de seguridad?***

  Aproximadamente 65kB. El tamaño total de la memoria es 72kB, no obstante, algunos espacios son requeridos para aplicaciones de productos específicos y áreas Mifare 4K.

- ***¿Cómo el área interna Mifare 4k del Nokia 6131 NFC o el área de la tarjeta inteligente interna puede ser utilizada para conservar la información crítica?***

  Esto depende totalmente del tipo de aplicación y las infraestructuras técnicas del entorno, donde la aplicación vaya a ser usada. No obstante, las siguientes cosas serán consideradas:

  - ✓ Área Mifare 4K es ya una memoria con control de acceso, y típicamente es fácil de implementar.
  - ✓ Java Card proporciona alta seguridad en el entorno y puede ejecutar un código, lo que significa que puede ser usado par más aplicaciones complementarias.
  - ✓ Gestionar algunas aplicaciones en Mifare esta cambiando.
  - ✓ En Java Card, la mayoría de las aplicaciones se pueden colocar en las memorias Java Card.

- ***¿Qué emisores de la tercera parte pueden gestionar las aplicaciones en el elemento seguro?***

  Por ejemplo Venyon, Cassis, ViVOtech

- ***¿Cuáles son las claves usadas con Mifare NDEFTagConnection?***

  Hay esencialmente 2 claves usadas para leer NDEF tags:

  - ✓ Claves usadas para el área MAD, que son: A0A1A2A3A4A5
  - ✓ Claves usadas para el área NDEF, que son: D3F7D3F7D3F7

- ***¿Cómo se desarrollarán las aplicaciones de seguridad en el futuro, cuando todos los teléfonos tengan diferentes claves de configuración?***

  Cada elemento seguro –no importa donde esté localizado: en el teléfono, en la tarjeta SIM, en el disco duro- tiene un único número de serie. Cada uno de los elementos de seguridad en Nokia 6131 NFC tiene sus propios chips específicos ISD.

  Lo que necesitas es la clave ISD del elemento seguro. Esta clave permite el libre acceso al elemento seguro y por lo tanto, debe mantenerse en secreto en todo momento. Esto permite, por ejemplo, ser instalado en el elemento seguro de las tarjetas de crédito, de esta manera, nadie podrá tener acceso sin autorización.

  Habrá un servicio disponible, que construya la conexión entre el elemento seguro y el servidor final. El servidor iniciado calcula la clave específica del chip ISD y estabiliza un protocolo seguro de comunicación entre el elemento seguro y la parte final.

  Habrá un servicio que permitirá desbloquear el elemento seguro para desarrollar tu trabajo.

2.2.4 Funcionalidad SDK

Es esta parte se cuestionan aspectos relacionados con la funcionalidad del SDK como pueden ser las siguientes:

- ***¿Es el emulador capaz de simular la funcionalidad del elemento de seguridad interno del teléfono?***

  El elemento de seguridad interno es simulado en el emulador. La implementación de Nokia 6131 NFC SDK no contiene funcionalidad actual en el elemento seguro real. Además, la implementación de simulación del elemento seguro no es valida para las actuales tarjetas inteligentes.

- ***¿Están limitadas las conexiones a cierto dominio? ¿Hay algunos permisos que necesitan ser puestos en orden para usar las conexiones del elemento seguro interno?***

  No necesitas establecer ningún permiso para acceder al elemento seguro o para usar la conexión ISO14443. No obstante, si tu quieres acceder al elemento seguro del teléfono, el MIDlet tiene que estar en la Trusted 3rd party del domino de seguridad.

  Si el MIDlet no está marcado en la tercera parte del certificado, conseguirás una excepción de seguridad cuando estés arrancando el MIDlet.

2.2.5 APIs SDK

A continuación se tienen en cuenta una serie de consideraciones relacionadas con los APIs de SDK.

- ***¿Como puedo usar el estándar Mifare del API?***

  En fin, utilizar el estándar Mifare del API es obligatorio para actualizar el software del teléfono a la ultima versión, 5.11.

  El software de Nokia 6131 NFC puede ser actualizado a al versión 5.11 en el punto de servicio de Nokia.

  El software de Nokia 6131 NFC no está disponible a través del actualizador de Nokia Software.

- ***¿Cómo puedo conseguir una conexión ISO14443 en el elemento de seguridad interno del teléfono?***

  Nokia 6131 permite a los MIDlets acceder al elemento de seguridad interno del teléfono. Las conexiones internas están abiertas directamente llamando Connector.open() y usando la “internal.se.url” del sistema de propiedad.

  Aquí hay un ejemplo de código de cómo abrir la conexión ISO14443 usando el interfaz javax.microedition.contactless.sc.ISO1443Connection:

```
String uri =System.getProperty("internal.se.url");
ISO14443Connection iseConn = (ISO14443Connection) Connector.open(uri);
```

- ***¿Como puedo conseguir la conexión en el estándar MiFare del elemento de seguridad interno del teléfono?***

  Nokia 6131 NFC permite a los MIDlets acceder al lado MiFare del elemento de seguridad interno del teléfono. Las conexiones internas del lado de MiFare están abiertas directamente llamando a Connector.open() y usando "internal.mf.url".

  Aquí se muestra un código de ejemplo de como se abre la conexión estándar de MiFare usando el interfaz:
  com.nokia.nfc.nxp.mfstd.MFStandardConnection

  ```
  String uri=System.getProperty("internal.mf.url"); MFStandardConnection
  mfStdConn = (MFStandardConnection) Connector.open(uri);
  ```

- ***¿Puede el Nokia 6131 NFC SDK soportar la conexión NFCIP-1? Si es así, ¿Cómo puede usarlo?***

  JSR 257 proporciona una extensión del API para las conexiones punto a punto de NFC. El paquete com.nokia.nfc.p2p contiene el interfaz NFCIPConnection para la comunicación entre dos dispositivos NFCIP.

  El modo de conexión es decidido cuando la conexión se abre usando la clase Connector. Un iniciador del modo de conexión puede ser abierto con la URL nfc:rf; type=nfcip; mode=initiator y una tarjeta de modo de conexión con nfc:rf; type=nfcip; mode=target.

  Hay que tener en cuenta que el método Connector.open(java.lang.String) llama a los bloques hasta que un dispositivo NFCIP es encontrado. Notificaciones sobre un dispositivo NFCIP no pueden ser recibidas al registrar una TargetListener en el DiscoveryManager.

Uno puede definir un valor de tiempo adicional en la URL de conexión. El tiempo está definido en milisegundos. Por ejemplo nfc:rf; type=nfcip; mode=initiator; timeout=5000 se esperará 5 segundos. El valor 0 significa que la llamada abierta se bloqueará indefinidamente. El valor 0 es tambien el valor por defecto.

En el modo iniciador uno tiene primero que enviar dato, entonces recibe datos, envía datos, recibe datos,…
Uno puede hacer tantas secuencias de enviar-recibir como sea necesario. Por cada llamada de envío debe haber una llamada de recibir.

En el modo de tarjeta las comunicaciones es similar pero uno primero debe recibir datos y luego enviar.

- ***¿Cómo se iniciará automáticamente una aplicación al tocar una etiqueta RFID?***
  Nokia 6131 NFC soporta conexiones del JSR-257 PushRegistry definido como JSR-257 en la especificación 1.0 Apéndice B.

  En pocos palabras, si un MIDlet es lanzado tocando una tarjeta NDEF y una NDEFRecordListener es registrada en DiscoveryManager, entonces el escuchador notificará y el parámetro recordDetected() contendrá el grabador NDEF que provocará el lanzamiento.

  Con el fin de lanzar el MIDlet tocando una tarjeta inteligente y además conseguir la información en la tarjeta después de que el MIDlet haya sido lanzado, debes implementar NDEFRecordListener y su método recordDetected y también, en el constructor añadir el NDEFRecordListener al DiscoveryManager con el correspondiente NDEFRecordType.

- ***¿Como puedo lanzar automáticamente una aplicación cuando tocas otro Nokia 6131 NFC?***

  De manera similar al lanzamiento de la aplicación tocando la tarjeta PushRegistry puede ser usado para lanzar una aplicación cuando se toca otro Nokia 6131 NFC. Necesitas usar "nfc:undefined_format" como una conexión URL como parámetro para usar esta funcionalidad.

  Tenga en cuenta que cuando "nfc:undefined_format" se ha registrado una aplicación es lanzada cuando las etiquetas no contienen NDEF o los datos NTIP son tocados.

- ***¿Puede el sondeo ser cambiado con un midlet?***

  No realmente. Un midlet activo hará incluso un sondeo del teléfono mientras es cerrado, pero no en todos, la administración de energía la realiza el propio teléfono. Es posible apagar el sondeo desde el menú de configuración de NFC.

- ***¿Qué tipos de tarjetas son soportadas por JSR-257 implementado en Nokia 6131 NFC SDK?***

  Los tipos de tarjeta soportados son:

  - ✓ Tipo de tarjeta ISO14443_CARD para ISO 14443-4 compatibles con las tarjetas inteligentes accedidas usando comandos APDU.
  - ✓ Tipo de tarjeta NDEF_TAG para etiquetas que contienen datos del NFC Forum.
  - ✓ Tipo de tarjeta RFID_TAG para tarjetas RFID en general

- ***¿Cuales son los NDEF RTDs soportados por Nokia 6131 NFC?***

  Los siguientes NDEF RTDs son soportados por aplicaciones nativas de NFC y JSR-257 implementado en Nokia 6131 NFC.

Type name format	Type name	Description
MIME	text/x-vCard	Business cards
MIME	text/x-vCalendar	Calendar notes
NFC Forum RTD	urn:nfc:wkt:Sp	Smartposters
NFC Forum RTD	urn:nfc:wkt:U	URI records
NFC Forum Ext Type	urn:nfc:ext:nokia.com:bt	Bluetooth record (for printing/image frame)

El siguiente NDEF RTDs está reservado para aplicaciones nativas y no pueden ser usadas para registros PushRegistry:

Type name format	Type name	Description
MIME	text/x-vCard	Business cards
MIME	text/x-vCalendar	Calendar notes
NFC Forum RTD	urn:nfc:wkt:Sp	Smartposters

- ***¿Cual es el formato de la tarjeta del Nokia 6131 NFC con un dispositivo de bluetooth?***

  El tipo de grabador NDEF puede ser EXTERNAL_RTD y el nombre "urn:nfc:ext:nokia.com:bt". Para la carga útil se necesita hacer el siguiente array de bytes:

Name	Size	Description
Configuration type	1 byte	0x00 = Discovery only 0x01 = PIN 0x02 = Public key
Buetooth address	6 bytes	Bluetooth address.
Class of Device(CoD)	3 bytes	Only imaging tags are supported by Nokia 6131 NFC, so bytes should in following bit format: xxxxxxxx xxx00110 1xxxxxxx. For example 0x00 0x06 0x80
Authentication info	16 bytes	Depends on the configuration type.
Short name length	1 byte	Length of short name in bytes.
Short name	n bytes	Text displayed on screen upon discovery

2.3 Introducción a Java

2.3.1 Introducción

La tecnología Java consiste en un lenguaje de programación y una plataforma software. El núcleo de beneficio de la plataforma Java es que puede ejecutarse sobre diferentes sistemas operativos, ocultando la complejidad del dispositivo de las aplicaciones y desarrolladores de aplicaciones.

Nokia está usando la tecnología Java para proporcionar una plataforma de aplicaciones abierta a los desarrolladores, permitiéndoles crear aplicaciones para dispositivos que soportan la tecnología. Una plataforma estándar da un amplio margen para la creatividad y libera a los desarrolladores de preocuparse por las particularidades de cada dispositivo diferente.

La interoperatividad que proporciona la plataforma Java es también muy útil para los usuarios de dispositivos móviles. Las aplicaciones creadas con los APIs estándar de Java deben ejecutarse en todos los dispositivos compatibles, no importa quien los fabricó.

La plataforma Java, Micro Edition (Java ME) está orientada para dispositivos mas pequeños, como dispositivos móviles, comunicadores, y PDAs. La plataforma Java ME no es una especificación simple para una pieza del software. En este caso, es una colección de tecnologías y especificaciones diseñadas para diferentes partes del mercado de los pequeños dispositivos.

La plataforma Java ME define tanto una Configuración de Conexión Limitada de Dispositivos (CLDC) basada en el Perfil de información de Dispositivo móvil (MIDP) como una Configuración de Conexión de Dispositivo (CDC) basada en un Perfil Personal. La plataforma J2ME define tanto CLDC como CDC.

MIDP es un perfil para dispositivos portátiles basados en CLDC, que tienen capacidad de comunicación, como la telefonía móvil. Esto define funcionalidad como el uso del interfaz de usuario, persistencia de almacenamiento, creación de redes, y modelos de aplicación. Casi todos los dispositivos de Nokia soportan el perfil MIDP. MIDP 2.0 es soportado por los nuevos dispositivos, considerando que los viejos dispositivos soportan MIDP 1.0. En la parte superior de MIDP, los dispositivos de Nokia soportan varios APIs adicionales.

2.3.2 Objetivos de diseño de los creadores de JAVA

2.3.2.1 Lenguaje Familiar:

Java no sería un lenguaje totalmente nuevo, se parecería a lo que conocemos como C++, así que no le sería tan complicado recalar en los programadores escépticos.

2.3.2.2 Lenguaje Orientado a objetos:

Para que un lenguaje pueda considerarse orientado a objetos debe soportar como mínimo las características de:

- Encapsulación.
- Herencia.
- Polimorfismo.
- Enlace dinámico.

2.3.2.3 Lenguaje Robusto:

Uno de los problemas más comunes en los lenguajes de programación es la posibilidad de escribir programas que pueden bloquear el sistema. Algunas veces este bloqueo puede ser inmediato, pero en otras ocasiones llega a aparecer inesperadamente porque, por ejemplo, la aplicación accede a zonas de memoria que no estaban siendo ocupadas por otros programas hasta ese momento. Un ejemplo claro de lenguaje no robusto es C. Al escribir código en C o C++ el programador debe hacerse cargo de la gestión de memoria de una forma explícita, solicitando la asignación de bloques a punteros y liberándolos cuando ya no son necesarios.

En Java, los punteros, la aritmética de punteros y las funciones de asignación y liberación de memoria (malloc() y free()) no existen. En lugar de los punteros se emplean referencias a objetos, los cuales son identificadores simbólicos. El gestor de memoria de Java lleva una contabilidad de las referencias a los objetos. Cuando ya no existe una referencia a un objeto, éste se convierte en candidato para la recogida de basura (garbage collection).

2.3.2.4 Lenguaje de alto rendimiento (multiples Threads):

Una de las características del lenguaje es que soporta la concurrencia a través de threads. En ocasiones puede interesarnos dividir una aplicación en varios flujos de control independientes, cada uno de los cuales lleva a cabo sus funciones de manera concurrente. Cuando los distintos flujos de control comparten un mismo espacio lógico de direcciones, se denominan threads.

2.3.2.5 Lenguaje portable:

El principal objetivo de los diseñadores de Java, y dado el gran crecimiento de las redes en los últimos años, fue el de desarrollar un lenguaje cuyas aplicaciones una vez compiladas pudiesen ser inmediatamente ejecutables en cualquier máquina y sobre cualquier sistema operativo. Por ejemplo, un programa desarrollado en Java en una estación de trabajo Sun que emplea el sistema operativo Solaris, debería poderse llevar a un PC que utilice sistema operativo Windows NT.

2.3.2.6 Lenguaje lo más simple posible:

Los diseñadores de Java trataron de mantener las facilidades básicas del lenguaje en un mínimo y proporcionar un gran número de extras con las librerías de clases.

2.3.2.7 Lenguaje seguro:

Se pretendía construir un lenguaje de programación que fuese seguro, esto es, que no pudiera acceder a los recursos del sistema de manera incontrolada. Por este motivo se eliminó la posibilidad de manipular la memoria mediante el uso de punteros y la capacidad de transformación de números en direcciones de memoria (tal y como se hace en C) evitando así todo acceso ilegal a la memoria. Esto se asegura porque el compilador Java efectúa una verificación sistemática de conversiones.

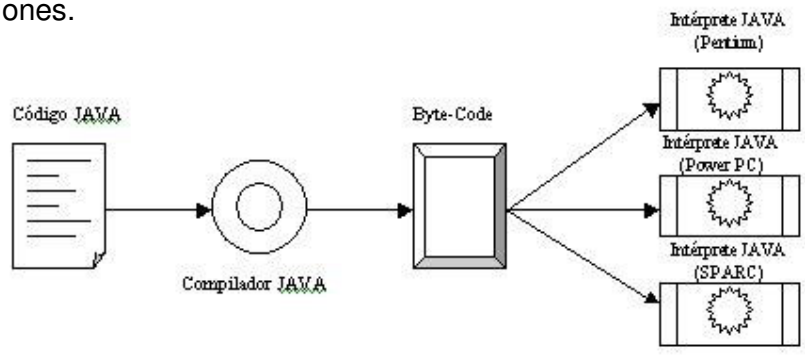


Figura 2.3.1: Ejecución en Java [9]

2.3.3 Java API Access Permissions

Ciertamente el método de llamadas seguras y APIs de MIDlets tiene algunas restricciones. Es posible que en esos casos el usuario consiga enviarlo por confirmación para permitir el método de llamada segura o el acceso puede ser bloqueado completamente, que dará como resultado un SecurytyException para ser lanzada.

Hacer que estas instrucciones aparezcan menos frecuentemente requiere del desarrollador para firmar el MIDlet. Solo la firma del operador o fabricante eliminarán las instrucciones completamente, también esto, realmente requiere una estrecha colaboración con las partes.

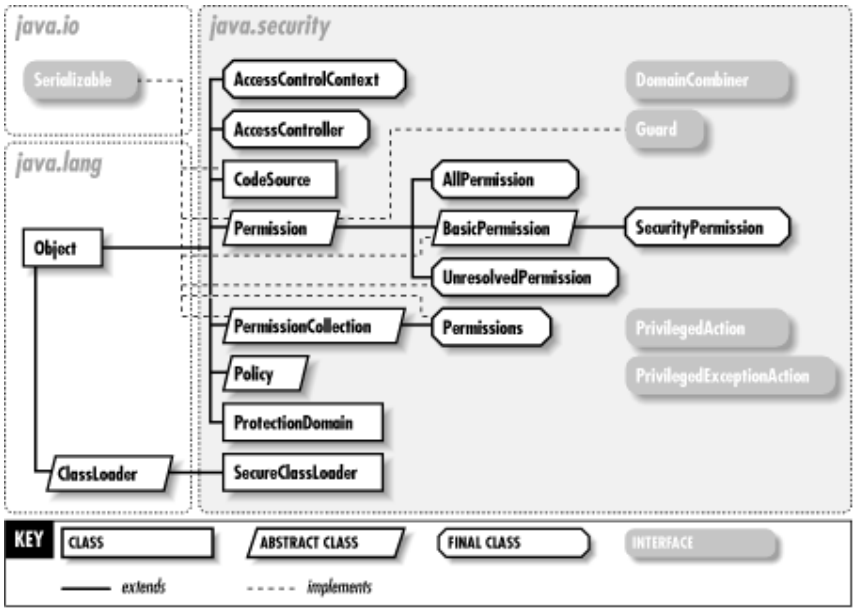


Figura 2.3.2: Clases del control de acceso del paquete java.security

2.3.3.1 Dominios de seguridad:

La especificación de MIDP 2.0 define 4 dominios de seguridad en los que el MIDlet puede ser instalado:

- Tercera parte de protección del domino
- Identificación de la tercera parte de protección del dominio
- Operador del protección del dominio
- Fabricante de la protección del dominio

2.3.3.2 API de protección de los grupos

Cada uno de los dominios de protección tiene cierto nivel de acceso para la protección. Los derechos de acceso están organizados en grupos de funciones:

- Acceso a la red (la especificación de MIDP también define el nivel bajo de red)
- Mensajes
- Aplicación de inicio automático
- Conectividad local
- Grabación multimedia
- Lectura de datos de usuario (incluye ficheros y PIM)
- Escribir/editar datos de usuario (incluye ficheros y PIM)
- Localización
- Comunicación mediante tarjeta inteligente
- Autenticación
- Control de llamadas
- Control de teléfono

El MIDlet tendrá acceso a cada una de las configuraciones definidas en los grupos de funciones anteriores, que están soportadas por el teléfono.

La configuración puede ser una de las definidas a continuación por el dominio de políticas de seguridad del teléfono.

- Siempre permitido / acceso bloqueado
- Preguntar la primera vez / preguntar una vez por sesión
- Preguntar todas las veces
- No permitido

Uno no puede cambiar la configuración por defecto del teléfono, pero después de instalar MIDlet, es posible cambiar la configuración del API de acceso de aquellas en las que esta permitido.

2.3.3.3 API de acceso definido en los estándares de Java ME

Las especificaciones de Java incluyen un número de versiones disponibles en el API de derechos de acceso.

MIDP 2.0 API access rights
MIDP 2.0.1 API access rights
MIDP 2.1 API access rights (igual que en MSA)
JTWI API access rights

Un MIDlet que no tiene firma, será colocado en el dominio inseguro, que tiene más restricciones para acceder a los APIs. Si el MIDlet ha sido firmado y el correspondiente certificado está cargado en el certificado del teléfono, el MIDlet será colocado en el dominio de protección para que el certificado sea vinculado.

2.3.3.4. Políticas del dominio de seguridad para un número de portadores, desviación del estándar

Como la especificación de las políticas del dominio seguridad de MIDP es ya una recomendación, algunos operadores han definido sus propios dominios de seguridad y derechos de acceso al API. Estos incluyen:

AT&T Java security domains (Cingular)
China Unicom Java security domains
Hutchinson 3G security domains
Sprint Java security domains
T-Mobile U.S. Java security domains

2.3.3.5 API de acceso a la configuración real de los móviles

También, los móviles genéricos tienen diferentes versiones implementadas del API de derechos de acceso:

API access rights on phones, S60 2nd FP2, on generic 6630 (2.39.126)
API access rights on phones, S60 2nd FP2 ver2, on generic 6680, 6630 (6.03.40)
API access rights on phones, S60 2nd FP3, on generic N72
API access rights on phones, S60 3rd, on generic E61i
API access rights on phones, S60 3rd FP1, on generic N95
API access rights on phones, Series 40 3rd FP1, on generic 6131
API access rights on phones, Series 40 3rd FP2, on generic Nokia 5300, 6300, 7373
API access rights on phones, Series 40 5th FP1, on generic Nokia 6500 slide

2.4 Introducción a Java Micro Edition

2.4.1 Introducción

En los últimos años el uso de dispositivos portátiles tales como teléfonos celulares y asistentes de datos se ha popularizado a nivel mundial. Este tipo de dispositivos tiene fines y características que son en esencia diferentes de los que tienen, por ejemplo, las computadoras portátiles o de escritorio. En el desarrollo de sistemas y aplicaciones para dispositivos móviles se hace necesario, por lo tanto, el uso de una tecnología especializada que tenga en consideración aspectos tan disímiles como las limitaciones de los recursos computacionales, la heterogeneidad de las plataformas de hardware, las comunicaciones móviles y, en especial, la preservación de la confidencialidad e integridad de los datos personales.

La mayor parte de los dispositivos portátiles se basa en la edición de la tecnología Java para sistemas integrados – embedded systems – denominada Java 2 Micro Edition (J2ME). La característica fundamental de esta edición, al igual que toda tecnología Java, es el uso de una máquina virtual, especialmente diseñada y adaptada para aprovechar al máximo los escasos recursos con los que cuentan los dispositivos integrados.

La tecnología J2ME proporciona mecanismos integrales para garantizar propiedades de seguridad de los dispositivos; en particular, para dispositivos móviles, define un modelo de seguridad a nivel de aplicación que restringe el acceso de las aplicaciones a funciones consideradas potencialmente peligrosas [10].

2.4.2 Breve historia

El objetivo de Java ME es el desarrollo de aplicaciones multi-plataforma para dispositivos móviles portables.

- Historia de Java
  - Oak (Proyecto Green) (1990)
    - Software para dispositivos electrónicos de consumo
  - Java 1 → 1.0 (96) 1.1(97)
  - Java 2 → 1.2 (98), 1.3 (2000), 1.4 (2002), 1.5 (2004), 1.6 (2006)

- Sun ha estructurado Java 2 dirigiéndose a sectores distintos (1999):
  - Java 2 Enterprise Edition (J2EE):
    - Soluciones de empresas e-commerce, e-bussines
  - Java 2 Standard Edition (J2SE):
    - Soluciones de PCs de sobremesas: applets, aplicaciones.
  - Java 2 Micro Edition (J2ME):
    - Dispositivos móviles, dispositivos de consumo y embebidos

- También Java Card (1996)
  - Tarjetas inteligentes (“smart cards”)
  - CPU: 8-16 bits; 1-5MHz
  - Memoria: 1.2K RAM, 32K memoria no volátil.

- Historia de J2ME
  - Personal Java (1997)
    - Dispositivos conectados con interfaces de usuario (set-top boxes, etc)
    - Basados en el jdk 1.1.8
    - Incorporado en el Personal Profile de J2ME
  - Embedded Java (1998)
    - Dispositivos embebidos con funcionalidad dedicada y restricciones de memoria (control automóvil)
    - Incorporado en un perfil CDC

2.4.3 Arquitectura de la plataforma Java 2

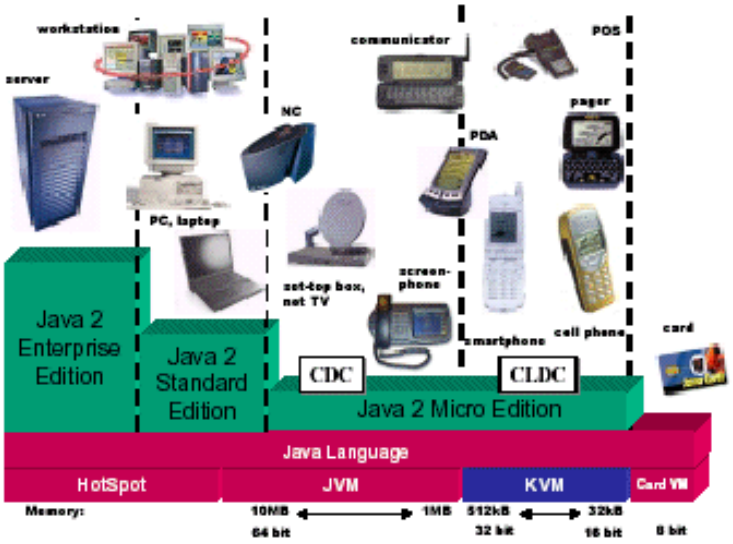


Figura 2.4.1: Arquitectura de la plataforma Java 2

Diferentes APIs y VMs, pero siempre el lenguaje de programación es Java

- Java proporciona:
  - Una plataforma estándar para el desarrollo de aplicaciones
  - Capacidades gráficas para diseñar interfaces de interacción con el usuario
  - Gran numero de programadores Java: facilidad y rapidez en el desarrollo de aplicaciones
  - Portabilidad de las aplicaciones entre diferentes dispositivos y distintos fabricantes.

2.4.4 Generalidades de J2ME

- Nueva plataforma para la programación de aplicaciones Java en dispositivos limitados

- Abarca un gran tipo de dispositivos limitados no sólo teléfonos móviles
  - PDAs, buscas, electrodomésticos inteligentes, etc.

- En el mundo de los sistemas móviles:
  - J2ME es complementaria, no es una alternativa a: WAP, iMode,…
  - J2ME añade:
    - Mayor riqueza de contenidos
    - Descarga de software en dispositivos móviles: personalización de servicios proporcionados por terceras partes.

- Versión muy simplificada de J2SE

- Estandarización bajo el Java Community Process (JCP)
  - JSR 68: J2ME Platform Specification
    - Arquitectura de la plataforma
    - Actividades de estandarización
  - JSR 185: Java Technology for Wireless Industry (JTWI)
    - Especifico para teléfonos móviles de siguiente generación
    - Como trabajan de forma conjunta varias tecnologías asociadas con MIDP para proporcionar una solución servicios.

- Java Specification Reports separados para los J2ME

2.4.5 Arquitectura de Java Micro Edition

Para conseguir flexibilidad y adaptación, J2ME se estructura en tres capas:

- Maquina virtual
- Configuración
  - Mínimo conjunto de clases disponibles
  - Engloba un segmento horizontal de mercado
- Perfiles
  - Clases adicionales para un segmento vertical de mercado

Un dispositivo puede soportar múltiples perfiles.

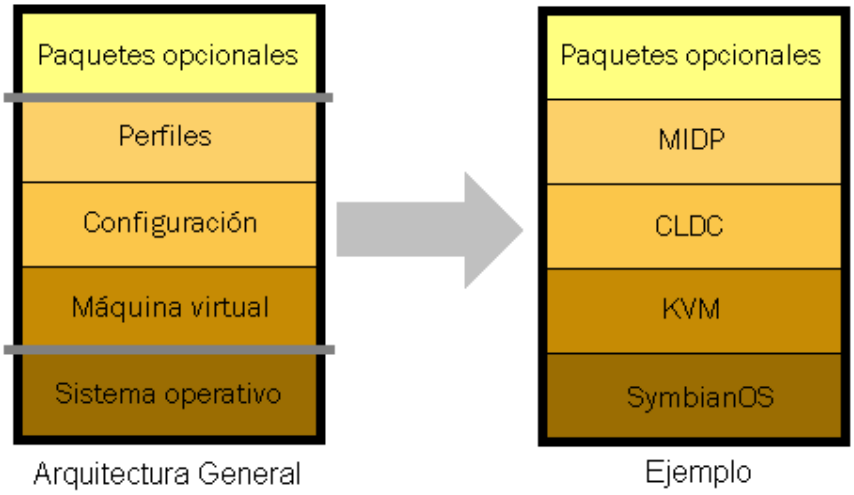


Figura 2.4.2: Arquitectura de Java Micro Edition

2.4.6 Maquina virtual

Una JVM:

- Interpreta código intermedio de los programas Java precompilados a código máquina ejecutable por la plataforma
- Efectúa las llamadas pertinentes al sistema operativo
- Observa las reglas de seguridad
- Está ligada a una configuración

Existen dos VM en la actualidad:

- CVM: Compact Virtual Machine, C Virtual Machine
- KVM: "Kilo" Virtual Machine, K Virtual Machine

1. CVM

- Orientada a dispositivos embebidos y electrónica de consumo (set-top box, TV digital, electrodomésticos…)
- Misma funcionalidad que JVM con:
  - Mejor uso de la memoria (=2MB)
  - Procesadores de 32 bits
- Ligada a la configuración CDC

2. KVM

- Antecedentes: Spotless (VM para PalmOs)
- Dispositivos con poca memoria, capacidad de proceso limitada y con conexión a red intermitente:
  - Memoria mínima 128KB
  - Procesadores de 16 ó 32 bits RISC o CISC
- Acepta el mismo conjunto de bytecodes y formato de ficheros de clase que la JVM
- Ocupa entre 40 y 80 KB
- Ligada a la configuración CLDC → más pequeña

2.4.7 Configuraciones

¿Qué es una configuración?

Mínimo conjunto de clases disponibles para un grupo de dispositivos. Los grupos se establecen según requisitos similares de memoria y procesamiento.

¿Qué define?

- Características soportadas del lenguaje de programación Java.
- Características soportadas por la Máquina Virtual Java.
- Bibliotecas básicas de Java y APIs soportados

Las configuraciones se especifican vía la iniciativa JCP que genera los correspondientes JSR.
Existen dos configuraciones actualmente: CDC y CLDC

2.4.7.1 Connected Device Configuration (CDC)

Orientado a dispositivos con:

- 512 KB de ROM
- 256 KB de RAM
- Conexión a red (fija)
- Soporte complete a la especificación de JVM
- Interfaz de usuario relativamente limitado
- Basado en J2SE v1.3

Especificado en JSR 36 (CDC1.0) y JSR 218 (CDC 1.1)
Ejemplos: Internet screen phones, DTV set-top boxes y sistemas telemáticos de automóviles.

Librerías incluidas:

- java.io → Clases en interfaces estándar de E/S
- java.lang → Clases básicas del lenguaje
- java.math → Paquete de matemáticas
- java.net → Clases en 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 utilidad estándar
- javax.microedition.io → Clases e interfaces para conexión CDC

2.4.7.2 Connected Limited Device Configuration (CLDC)

Orientado a dispositivos con:

- 160KB a 512KB de memoria disponible para Java
- Procesador de 16 o 32 bits, velocidad 8-32MHz
- Limitaciones de consumo (baterías)
- Conectividad a red (inalámbrica)
- Restricciones importantes en el interfaz de usuario

Especificado en JSR 30 (CLDC 1.0) y JSR 139 (CLDC 1.1)

Especificación CLDC 1.0/1.1 disponible:

- Sun proporciona una implementación de referencia de CLDC sobre KVM, para Linux, Windows y Solaris.
- Principales fabricantes de móviles la implementan en la mayoría de sus modelos (Nokia, Siemens, Samsung,…)

Librerías incluidas:

- java.io → Clases en interfaces estándar E/S. Subconjunto J2SE
- java.lang → Clases e interfaces de la VM. Subconjunto de J2SE.
- java.util → Clases e interfaces y utilidades estándar.
- javax.microedition.io → Clases e interfaces para conexión CLDC

2.4.8 Perfiles

Conjunto de clases Java que complementan una configuración para un conjunto específico de dispositivos ("segmento vertical").

¿Qué definen?

- APIs que controlan el ciclo de vida de la aplicación
- Interfaz de usuario, etc.

Los perfiles permiten la portabilidad de aplicaciones J2ME entre diferentes dispositivos.

Los perfiles se especifican vía la iniciativa JCP que genera los correspondientes JSR.

Perfiles sobre CDC:

- Foundation Profile (JSR 46, JSR 219): Perfil básico para dispositivos sin interfaz gráfico.
- Personal Basis Specification (JSR 129): Perfil gráfico para dispositivos con interfaz gráfico básico.
- Personal Prifile (JSR 62, JSR 216): Perfil gráfico basado en AWT (dispositivos con interfaz gráfico), evolución de Personal Java.

Perfiles sobre CLDC:

- Mobile Information Device Profile (JSR 37, 118 (2), 271 (3)): Perfil para dispositivos inalámbricos: móviles, PDAs,…
- Information Module Profile (JSR 195): Perfil para dispositivos con interfaz gráfica limitada: parquímetros, alarmas,…

2.4.9 Paquetes Opcionales

Los paquetes opcionales son un conjunto de APIs adicionales que pueden ser añadidos de forma flexible sobre diferentes perfiles.

Son utilizadas en una multitud de dispositivos y familias de dispositivos.

Un paquete opcional contiene una funcionalidad que es independiente del segmento vertical: Bluetooth, gestión de contenido multimedia, localización,…

Un dispositivo puede soportar múltiples paquetes opcionales.

Paquetes opcionales sobre CDC:

- JSR 66: RMI Optional Package → Subconjunto de J2SE RMI.
- JSR 169: JDBC Optional Package → Soporte JDBC en dispositivos CDC.
- JSR 209: Advanced Graphics and User Interface Optional Package → Facilidades de migración para interfaces de usuario y gráficos avanzados de J2SE a J2ME.

Paquetes opcionales sobre CLDC:

- JSR 75: PDA Optional Packages → Acceso a ficheros y datos personales
- JSR 82: Bluetooth API → Desarrollo de aplicaciones que usan Bluetooth
- JSR 120, JSR 205 (2.0): Wireless Messaging API → Acceso a sistemas de envío de mensajes (SMS, CBS Cell Broadcast Service)
- JSR 135: Mobile Media API (MMAPI) → Acceso a reproducción de recursos multimedia (audio y video) y funcionalidades multimedia avanzadas.

- JSR 172: Web Services APIs → Desarrollo de clientes Web en dispositivos móviles.
- JSR 177: Security and Trust Services → Mejora la seguridad añadiendo almacenamiento seguro, APIs criptográficas, firmas digitales, gestión de credenciales.
- JSR 179: API de Localización → Acceso a la información de localización física.

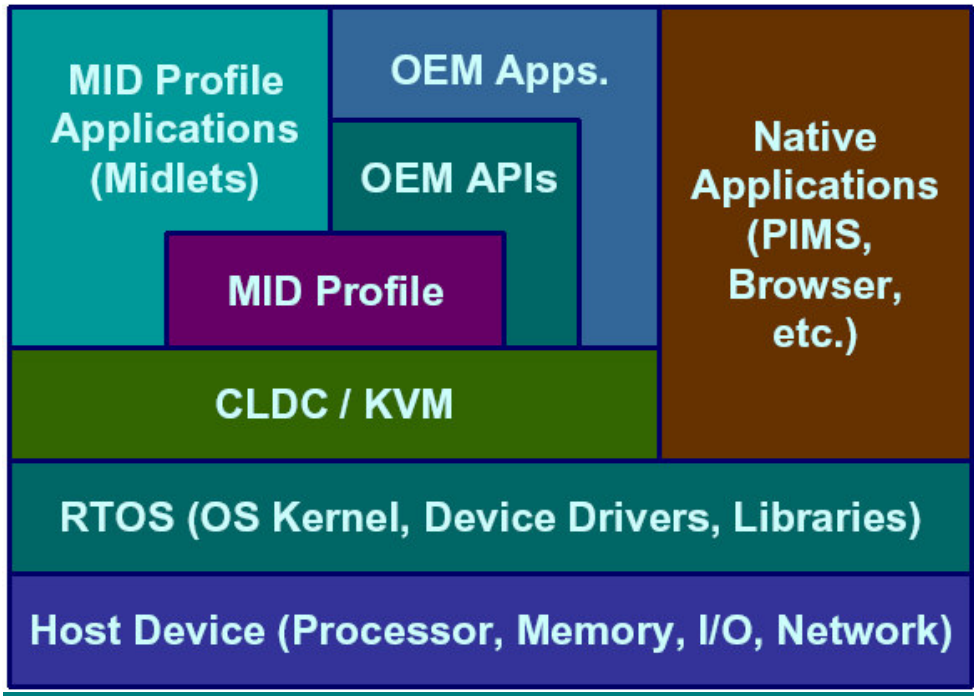


Figura 2.4.3: Arquitectura MIDP/CLDC/KVM

2.5 Introducción a MIDP

MIDP es el acrónimo de Perfil para dispositivos de información móvil (Mobile Information Device Profile) y nos proporciona un perfil que se apoya en CLDC y que nos va a proporcionar los paquetes y clases necesarios para el desarrollo de nuestras aplicaciones. MIDP está orientado principalmente a teléfonos móviles, aunque existe una implementación para PalmOS (versión 3.5 y superiores) y PocketPC, por lo que es también utilizable en casi cualquier PDA.

Es una versión de J2ME (Java 2 Micro Edition) integrada en el hardware de celulares relativamente modernos que permite el uso de applets en estos, tales como juegos, aplicaciones u otros.

2.5.1 Versiones

- MIDP 1
  - JSR 30
  - Final Release: Sep 2000
- MIDP 2
  - JSR 118
  - Final Release: Nov 2002
  - Final Release 2: Jun 2006
- MIDP 3
  - JSR 271
  - Estado: Early Draft Review (Feb 2007)
  - MIDlets en CLDC, CDC, y OSGi

2.5.2 Generalidades

- Los Requisitos hardware mínimos que exige MIDP 1.0 son los siguientes:

    - Memoria:
        - 256 KB de memoria no volátil para los componentes MIDP
        - 8 KB de memoria no volátil para creación de datos persistentes
        - 128 KB de memoria volátil para la ejecución de Java
    - Pantalla:
        - Tamaño: 96x54
        - Profundidad: 1 bit
        - Aspecto píxel 1:1
    - Entrada, uno o más de los siguientes mecanismos:
        - Teclado “one-handed” o “two-handed”
        - Pantalla táctil
    - Conectividad:
        - Limitada, tipicamente wireless
    - Sonido
        - Tonos, vía hardware dedicado o algoritmo software

La arquitectura de las aplicaciones desarrolladas sobre dispositivos que incorporan la arquitectura MIDP coexiste con terceras aplicaciones que se desarrollan sobre las distintas capas de aplicación que existen sobre estos dispositivos, dando lugar a la posible coexistencia de distintos tipos de aplicaciones sobre un mismo dispositivo que se aprovechan de la tecnología existente.

2.5.3 Alcance

Define el conjunto de APIs disponibles para el desarrollo de aplicaciones portables entre dispositivos móviles.

MIDP cubre:

- Ciclo de vida de la aplicación
- Interfaz de usuario
- Soporte de red
- Almacenamiento persistente
- Sonidos
- Juegos en 2D
- Seguridad extremo a extremo
- Timers, excepciones,…
- …

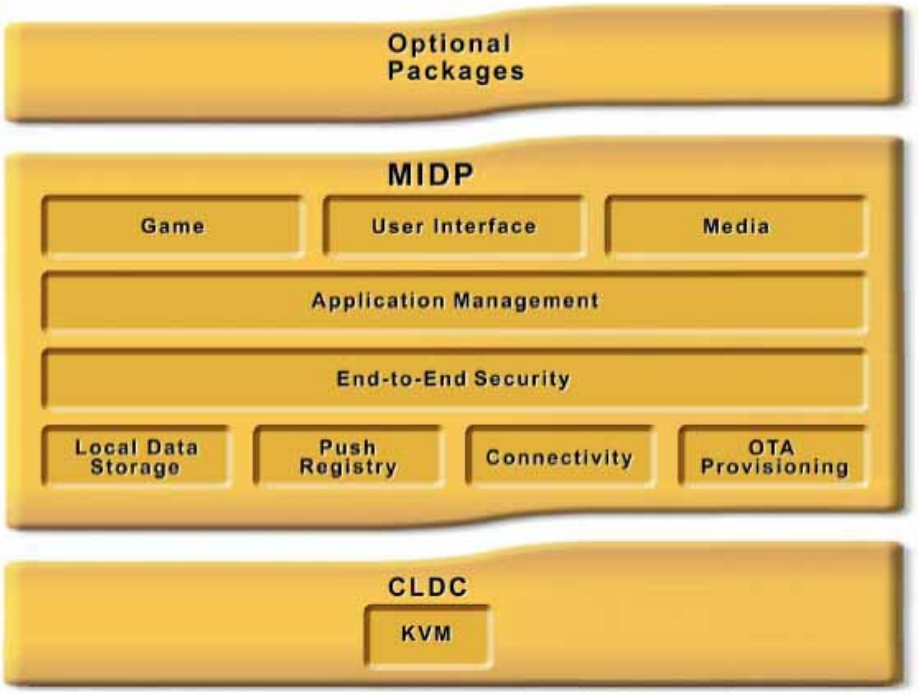


Figura 2.5.1: Localización de MIDP en la arquitectura.

MIDP no cubre:

- Descarga y gestión de aplicaciones en los dispositivos
- Seguridad a bajo nivel
- Gestión de baterías, codificadores de voz,…

Se asume la existencia de Application Management System (AMS)

- Dependiente del dispositivo
- Instala, interacciona con y borra MIDlets.

Las aplicaciones MIDP permiten tener aplicaciones intuitivas y gráficas. La IGU se ha optimizado para las pequeñas pantallas, mecanismos de introducción de datos y otras características de los dispositivos móviles. Las aplicaciones MIDP se pueden instalar y ejecutar en local, trabajar en red o de forma desconectada y puede almacenar y gestionar de forma segura datos en local.

Las pantallas y elementos son relativos al dispositivo, con soporte incorporado en los mismos para el tamaño de la pantalla, mecanismos de navegación e introducción de datos. Las aplicaciones creadas, gracias a la portablilidad proporcionada por J2ME, se adaptarán a la disposición y mecanismos de navegación propios de cada dispositivo.

2.5.4 Propiedades

MIDP debe proporcionar al menos las siguientes propiedades

- microedition.locale
- microedition.profiles

MIDP 2.0 especifica dos propiedades más dependientes de la implementación

- microedition.commports
- microedition.hostname

2.5.5 Librerías

Interfaz de usuario:

- javax.microedition.lcdui

Ciclo de vida de la aplicación (MIDlet):

- javax.microedition.midlet

Memoria persistente:

- javax.microedition.rms

Conectividad:

- javax.microedition.io

Núcleo:

- java.io
- java.lang
- java.util

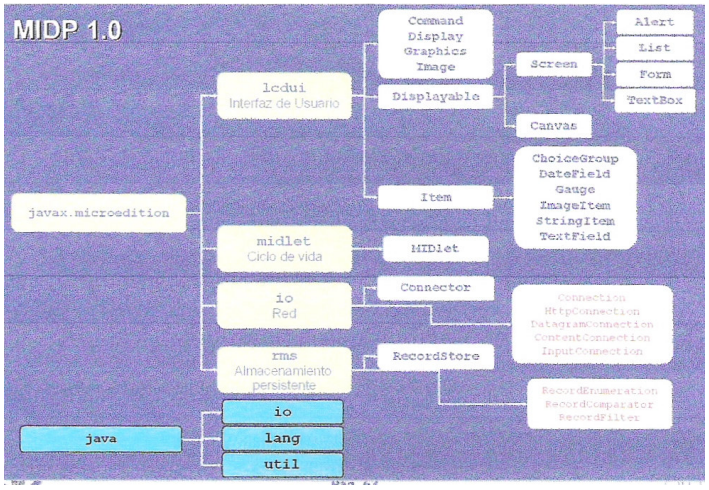


Figura 2.5.2: Librerías MIDP 1.0

2.5.6 Librerías añadidas MIDP 2.0

Interfaz de usuario:

- javax.microedition.lcdui.game

Seguridad: clave pública

- javax.microedition.pki

Sonidos:

- javax.microedition.media
- javax.microedition.media.control

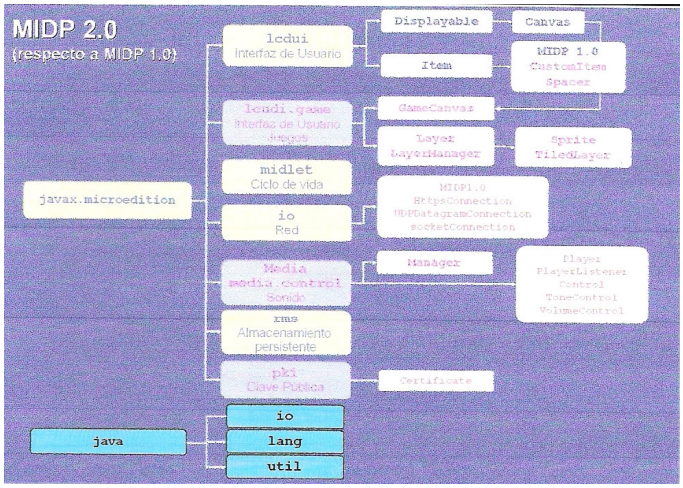


Figura 2.5.3: Librerías añadidas MIDP 2.0

2.5.7 Interfaz de usuario

El API de usuario aísla a los desarrolladores de la complejidad de la construcción de aplicaciones portables.

La funcionalidad de la interfaz de usuario incluye pantallas predefinidas para selección de listas, edición de texto, cuadros de diálogo emergentes para alertas y anuncios deslizantes. Se incluyen pantallas de formulario que pueden incluir una serie de ítems predefinidos:

- imágenes
- campos de texto de sólo lectura
- campos de texto editables
- campos de fecha
- campos de hora
- gráficos
- botones de selección
- elementos personalizados

API de alto nivel:

- Muy portable
- Orientada a screen y widget
- Las aplicaciones que usan este API deberían funcionar en todos los dispositivos
- No hay acceso a todas las funciones del dispositivo
- Mas sencillo y menos potente que AWT

API de bajo nivel:

- Primitivas de dibujo
- Eventos de teclado

Menos portabilidad, mejor experiencia del usuario.

2.5.8 Memoria Persistente

- API independiente del dispositivo
- Base de datos sencilla orientada a registro (RMS)
    - Registro (record) son array de bytes
    - Los registros se guardan en almacenes de registro (record stores)
    - Los almacenes de registros se comparten entre MIDlets de un mismo MIDlet suite
- Soporta numeración, ordenación y filtrado
- Actualización atómica de registros
- Definido en el paquete javax.microedition.rms

2.5.9 Conectividad

- Implementan el Generis Connection Framework definido en el paquete javax.microedition.io:
    - Requiere soporte de conexiones http como cliente

- Añade e implementa el interfaz HttpConnection, hereda directamente del interfaz ContentConnection

- La implementación del interfaz DatagramConnection, definido en CLDC es opcional, pero recomendable.

2.5.10 MIDlets

2.5.10.1 Introducción

- ✓ Un MIDlet es la unidad básica de ejecución en MIDP
  - Tiene un ciclo de vida bien definido
  - Da información descrita sobre sí mismo
  - Extiende javax.microedition.midlet.MIDlet

- ✓ Existe el concepto de MIDlet permanente:
  - Reside, al menos en parte, en memoria no volátil
  - Puede descargarse de la red y grabarse en memoria persistente.
  - Pueden ser ejecutados repetidas veces por el usuario sin necesidad de volver a descargarlos.

- ✓ MIDlet Suite:
  - Conjunto de aplicaciones (MIDlets) que comparten recursos en el contexto de una única maquina virtual

Para definir de manera mas concreta, un MIDlet es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC.

Están pensados para ser descargados a través de una conexión a internet. El medio empleado para garantizar esta descarga recibe el nombre de OTA (Over the Air)

Una aplicación J2ME está formada por:

- Archivo JAR (Java Archive): contiene a la aplicación en sí
- Archivo JAD (Java Archive Descriptor): contiene información sobre la aplicación.

Los midlets tienen algunos requisitos funcionales:

- Dispositivo MID (Mobile Information Device): Dispositivo que sigue la especificación MIDP

- Todo dispositivo MID dispone de un software residente llamado AMS (Application Management Software o Gestor de Aplicaciones). El AMS gestiona el ciclo de vida de los MIDLets.

- Etapas del ciclo de vida de un MIDLet
  - Localización
  - Instalación
  - Ejecución
  - Actualización
  - Borrado

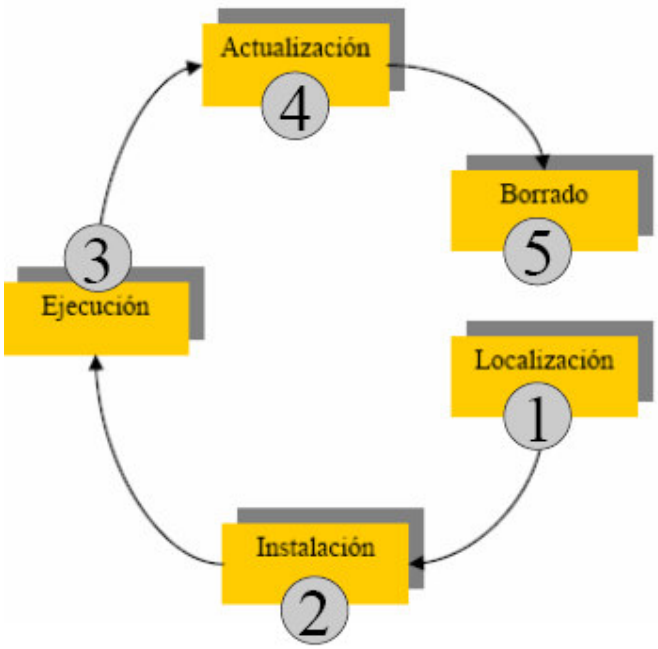


Figura 2.5.4: Etapas del ciclo de vida un MIDLet

2.5.10.2 Etapas del ciclo de vida de un MIDLet

1. Localización o Descubrimiento

- ✓ Selección a través del AMS de la aplicación a descargar

- ✓ Según las capacidades del dispositivo, descarga mediante:
  - Cable conectado al ordenador (serie, USB,…)
  - Wi-Fi, BlueTooth, GPRS, UMTS, …

- ✓ Protocolo descarga
  - HTTP 1.1 u otro protocolo con funcionalidad similar

- ✓ El usuario localiza un MIDLet con su dispositivo, típicamente, a través de hiperenlaces

- ✓ Si el destino del hiperenlace es un archivo JAR: El archivo JAR y su URL son enviados al AMS del dispositivo

- ✓ Si el destino del hiperenlace es un archivo JAD:
  - Transferencia del JAD y de su JAR asociado
  - Archivo JAD convertido a Unicode antes de ser usado
  - El JAD debe contener los atributos obligatorios de la especificación MIDP
  - Atributos JAD comprensibles según la sintaxis de la especificación MIDP
  - El usuario debe poder confirmar la instalación del MIDlet y debería haber un control de versiones.

2. Instalación

- ✓ Una vez descargada la aplicación
- ✓ El AMS controla el proceso, informando al usuario de la evolución del mismo y de posibles problemas
- ✓ Una vez instalado, todas sus clases, archivos y almacenamiento persistente están preparados para su uso.
- ✓ Después de la fase de instalación, el MIDlet queda almacenado en una zona de memoria persistente del dispositivo MID (hasta que el usuario utilice el AMS para borrarlo)

3. Ejecución

- ✓ Mediante el AMS, se inicia la ejecución de los MIDlets.
  - o En esta fase, el AMS gestiona los 3 posibles estados de ejecución del MIDlet, según los eventos que se produzcan durante su ejecución
  - o Estados: pausa, activo y destruido
  - o Transición de estados: mediante la invocación de métodos de la clase MIDLet (new, startApp, pauseApp, destroyApp)

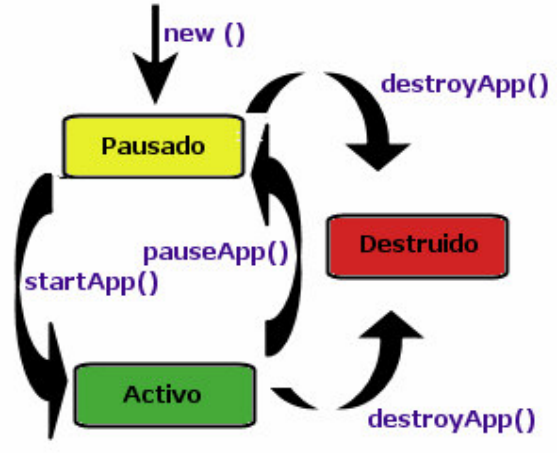


Figura 2.5.5: Estados de un MIDLet

- ✓ El dispositivo debe invocar a las clases CLDC y MIDP requeridas por la especificación MIDP.
- ✓ Si existen varios MIDlets presentes, posibilidad de seleccionar el MIDlet que se desea ejecutar

4. Actualización

- ✓ El AMS detecta, tras una descarga, si el MIDlet descargado es una actualización de un MIDlet ya presente en el dispositivo.

- ✓ Si es así, debe de informar acerca de esto, además de ofrecer la oportunidad de decidir si interesa o no dicha actualización.

5. Borrado

- ✓ El AMS debe permitir al usuario eliminar MIDlets

- ✓ Antes de eliminar una aplicación, el usuario debe dar su confirmación

- ✓ El dispositivo debería advertir al usuario de cualquier circunstancia especial durante la eliminación del MIDlet.
  - o Por ejemplo, el MIDlet a borrar podría contener a otros MIDlets, y el usuario debería de ser alertado ya que todos ellos quedarían eliminados.

Capítulo 3: Funcionalidad Requerida

Este capítulo explica el conjunto de requisitos y objetivos propuestos que debe cumplir la aplicación.

Por lo cual, se quiere realizar una aplicación que con el uso de la telefonía móvil y la tecnología NFC integradas en el espacio tecnológico universitario, permita el desarrollo de nuevas aplicaciones que consigan la implantación no sólo de un nuevo modelo de enseñanza, sino un nuevo modelo universitario.

Estas nuevas soluciones podrán estar orientadas a muchos y diferentes aspectos: aportar solución a problemas de identificación, control de presencia/asistencia, puntos de información, envío de información, pago electrónico, fidelización, uso de servicios universitarios, etc.

Sin embargo, nuestro trabajo está orientado al servicio y aplicaciones docentes, mas concretamente se centra en el estudio y desarrollo de una aplicación que permita a los alumnos y profesores, la realización y corrección de exámenes a través de un dispositivo móvil, y tarjetas Mifare. Ya que si nos centramos en el entorno Universitario, es difícil pensar que alumnos o profesores no dispongan de un terminal móvil y que éste le acompañe en todo momento. Además junto con la tecnología NFC permitirá el intercambio de datos.

Esta transmisión de datos no es que sea masiva, como el caso de bluetooth o WLan, pero si suficiente como lector de etiquetas. Además, garantiza seguridad, debido a que por su corto alcance los dispositivos tienen prácticamente que tocarse, al igual que proporciona un servicio intuitivo, y sin necesidad de configuración.

Por ello, ante el intento del desarrollo de una aplicación que cumpla con todos estos requisitos comentados, se lleva a cabo el estudio de las posibles soluciones a implantar en el entorno universitario, lo cual nos lleva a realizar este proyecto, que se compone de dos módulos diferenciados, uno por parte del profesor, y otro por parte del alumno.

En cada una de estos módulos, las tareas a realizar son diferentes. En la parte del profesor se querrán realizar unas determinadas tareas distintas a las del alumno. Es por ello, que en nuestro desarrollo tendremos lo siguiente:

- **Módulo del profesor**

En este caso, queremos que entre las tareas a realizar estén:

- Redactar el examen con el propio dispositivo móvil.

  Esto consiste en que el profesor podrá escribir manualmente con su dispositivo móvil, el conjunto de preguntas y respuestas que conformarán el examen. Para ello, se le irán solicitando paso a paso cada uno de los elementos que componen el examen.

  Lo primero que se deberá solicitar, es que introduzca el enunciado de la pregunta y el número de respuestas posibles. Entonces, en función de este número, se le solicitará al profesor que introduzca que posibles opciones de respuestas existen, entre las cuales estará la correcta y que también deberá marcar el profesor, para cuando proceda a corregir.

  Una vez ha escrito una pregunta, podrá seguir introduciendo nuevas preguntas si lo desea, o por el contrario, podrá guardarlas en la tarjeta que entregará al alumno posteriormente para que este pueda responderlas.

- Cargar directamente un examen que ya ha sido redactado.

  Con esta opción, no será necesario que el profesor introduzca una a una las preguntas en el dispositivo móvil, ya que el examen está redactado con otra aplicación y por lo cual lo único que tiene que hacer, es cargarlo.

  Una vez observe que todo es correcto y las preguntas que contiene el examen son las deseadas, al igual que en el caso anterior, podrá guardarlas en una tarjeta que se entregará al alumno, para que las conteste.

- Corregir un examen.

  Un sistema de evaluación no cumpliría con los requisitos si no existe la opción de corregir.

  Con esta opción el profesor podrá determinar cuantas preguntas contestó correctamente el alumno y cuáles no, y a partir de ello asignar una nota.

  Por lo cual, esta aplicación debe leer la tarjeta que contiene las respuestas que introdujo el alumno, y a continuación compararlas con las que almacenó el profesor que son correctas.

  De esta manera, obtenemos los fallos y los aciertos, y por medio de un cálculo matemático obtenemos la calificación.

❖ **Módulo del alumno**.

- Responder preguntas.

  Esta es la única opción que tendremos por parte del alumno, ya que la única tarea de este será responder el examen.

  Para ello, en primer lugar se deberá realizar la lectura de la tarjeta que le entregó el profesor, y que contiene las preguntas para la realización del examen.

Estas preguntas se deberán ir mostrando una a una, para que el alumno pueda responderlas poco a poco.

Además tendrá la opción de volver atrás para poder modificarla, antes de almacenar los datos en la tarjeta. La cual, será la que se entregue al profesor para que este pueda corregir el examen.

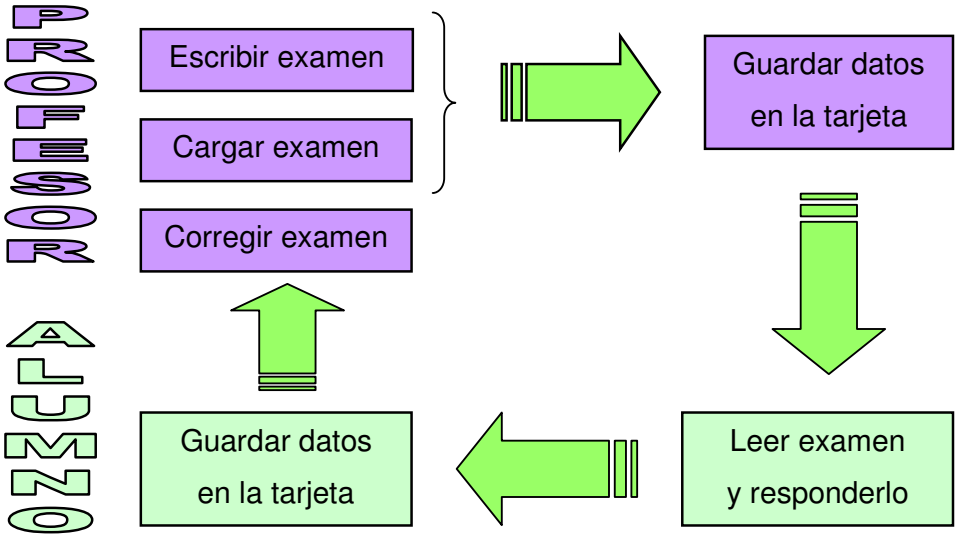


Figura 3.1: Funcionalidad Requerida que debe contener la aplicación

Capítulo 4: Entorno de desarrollo

Este capítulo se centra principalmente en explicar en detalle el conjunto de herramientas y elementos utilizados para la realización del proyecto.

En primer lugar se explica la aplicación utilizada para escribir el código necesario que permite desarrollar la aplicación y obtener la ejecución deseada.

Esta aplicación es la denominada, "**pcGRASP**", que simplemente se trata de un entorno de desarrollo, creado para una mejor comprensión de código escrito de lenguajes como Java, estas distintas sentencias java son ejecutadas por el Wireless Toolkit para mostrar la aplicación.

En la siguiente imagen se muestra un ejemplo de dicho entorno:

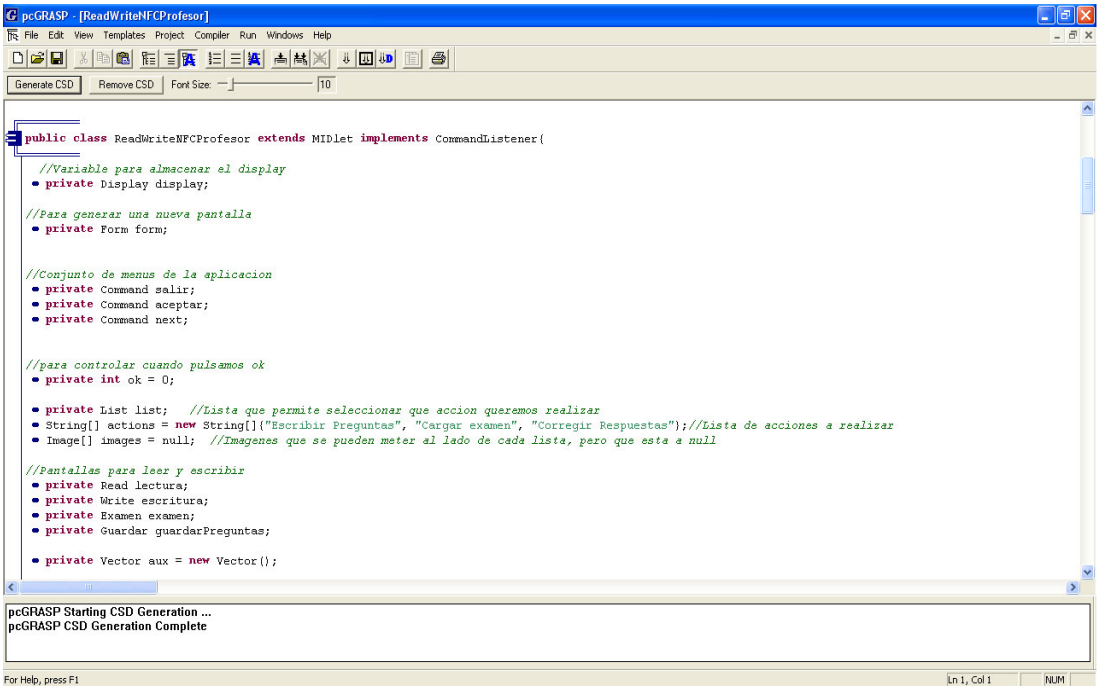


Figura 4.1: Ejemplo del entorno de desarrollo pcGRASP

Para almacenar las sentencias java se generan unos ficheros que se denominan clases, las cuales realizan una tarea específica dentro de la aplicación. Por ejemplo, las clases "ReadWriteNFCAlumno.java" y "ReadWriteNFCProfesor.java" son la que lanzan el midlet, la clase "Read.java" lee de la tarjeta, la clase "Write.java" escribe en la tarjeta, "NFCUtil.java" para gestionar las conexiones, y así con cada una de las clases generadas.

Para que todo este código java escrito pueda compilarse, garantizando que la escritura es correcta, se requiere del **jdk** (Java Development Kit). Esto

no es más que un software que provee herramientas de desarrollo para la creación de programas en java.

Los programas más importantes que se incluyen son:

- **Appletviewer**: es un visor de applet para generar sus vistas previas, ya que un applet carece de método *main* y no se puede ejecutar con el programa java.
- **Javac**: es el compilador de JAVA.
- **java**: es el intérprete de JAVA.
- **javadoc**: genera la documentación de las clases java de un programa.

En nuestro caso, de estos programas que se incluyen, son necesarios el compilador Javac, que nos permite detectar posibles errores de sintaxis, y el intérprete Java, que como el propio nombre indica, interpreta las sentencias java y las ejecuta. También de carácter opcional es el javadoc, que genera la documentación de las clases y de esta manera cualquiera puede obtener información acerca del funcionamiento y características de las mismas.

Una vez las clases java han sido creadas, podemos comenzar a utilizar otra de las herramientas que se requieren para la realización de este proyecto.

Este otro elemento necesario para el desarrollo de la aplicación, es la herramienta **Java Wireless Toolkit**, emulador de aplicaciones para los celulares.

Esta herramienta permite simular la aplicación que se ejecutará en el teléfono móvil, y con la cual se han realizado las pruebas de la aplicación, antes que utilizar el dispositivo físico Nokia 6131 NFC.

El jdk junto al Wireless Toolkit, interpretan y ejecutan el código java desarrollado mediante pcGRASP o cualquier otro editor. Sin embargo, para que la ejecución sea factible, hay que crear en primer lugar los archivos .jar y .jad, generados tras la compilación mediante build.

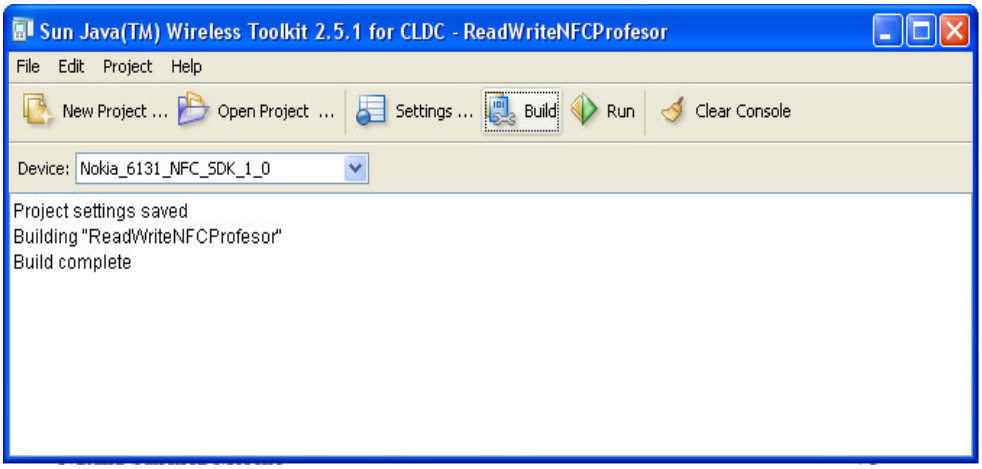


Figura 4.2: Compilación y ejecución de la aplicación con Wireless Toolkit

Una vez han sido generados dichos archivos, ya se puede simular la aplicación con esta herramienta utilizando el botón run. Esto muestra en la pantalla la ejecución de la aplicación NFC en un teléfono móvil como simulador, tal como se muestra en la siguiente imagen:

Figura 4.3: Simulación de ejecución

Una vez aparece esta pantalla, ya está todo listo para comenzar la simulación. Sin embargo, para que en ella se puedan utilizar las herramientas de NFC, tales como tarjetas y conexiones, se necesita un API específico para probar dichas características, en este caso se trata del **API de Nokia 6131 NFC**. Este API debe ser añadido a la herramienta Wireless Toolkit para poder ver en la simulación todas las operaciones que se pueden realizar con esta tecnología inalámbrica.

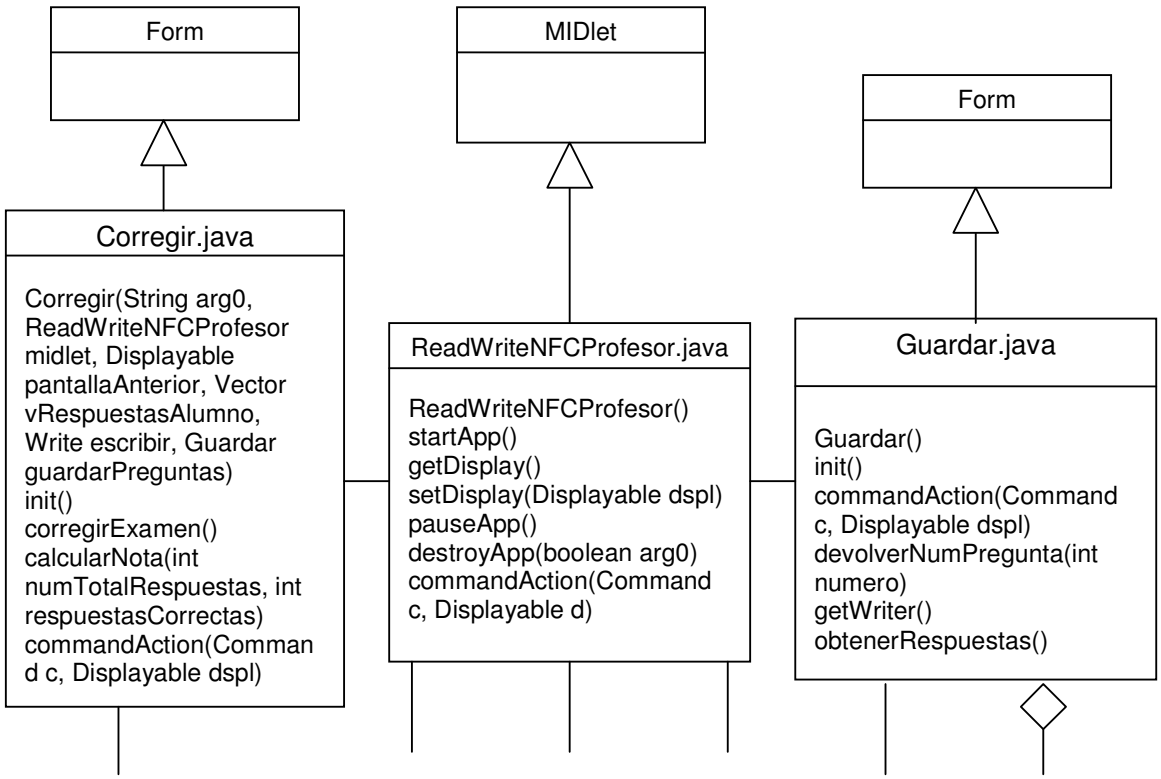
Capitulo 5. Diseño software

A lo largo de esta memoria, hemos explicado, por un lado los conceptos de la tecnología estudiada; y por otro se han definido el entorno de desarrollo y el conjunto de requisitos que requiere la aplicación, por lo cual, es el momento de explicar el conjunto de clases y métodos que la componen.

5.1 Diagrama UML

Con este diagrama se muestra una primera visión del conjunto de clases que componen la aplicación, y la relación existente entre ellas.

5.1.1 Relación para el módulo del profesor



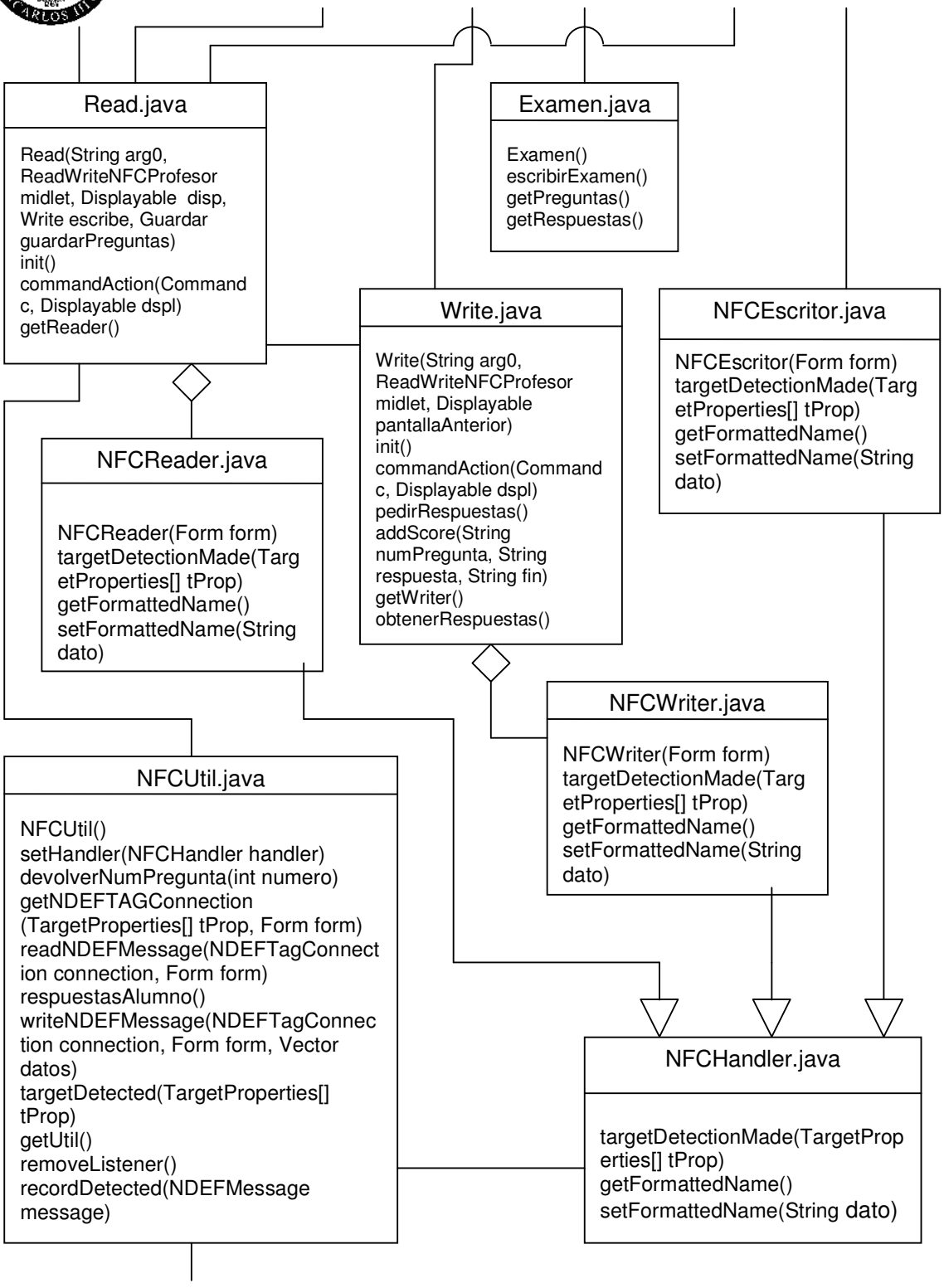


Figura 5.1: Relación UML de las clases del Profesor

5.1.2 Relación para el módulo del alumno

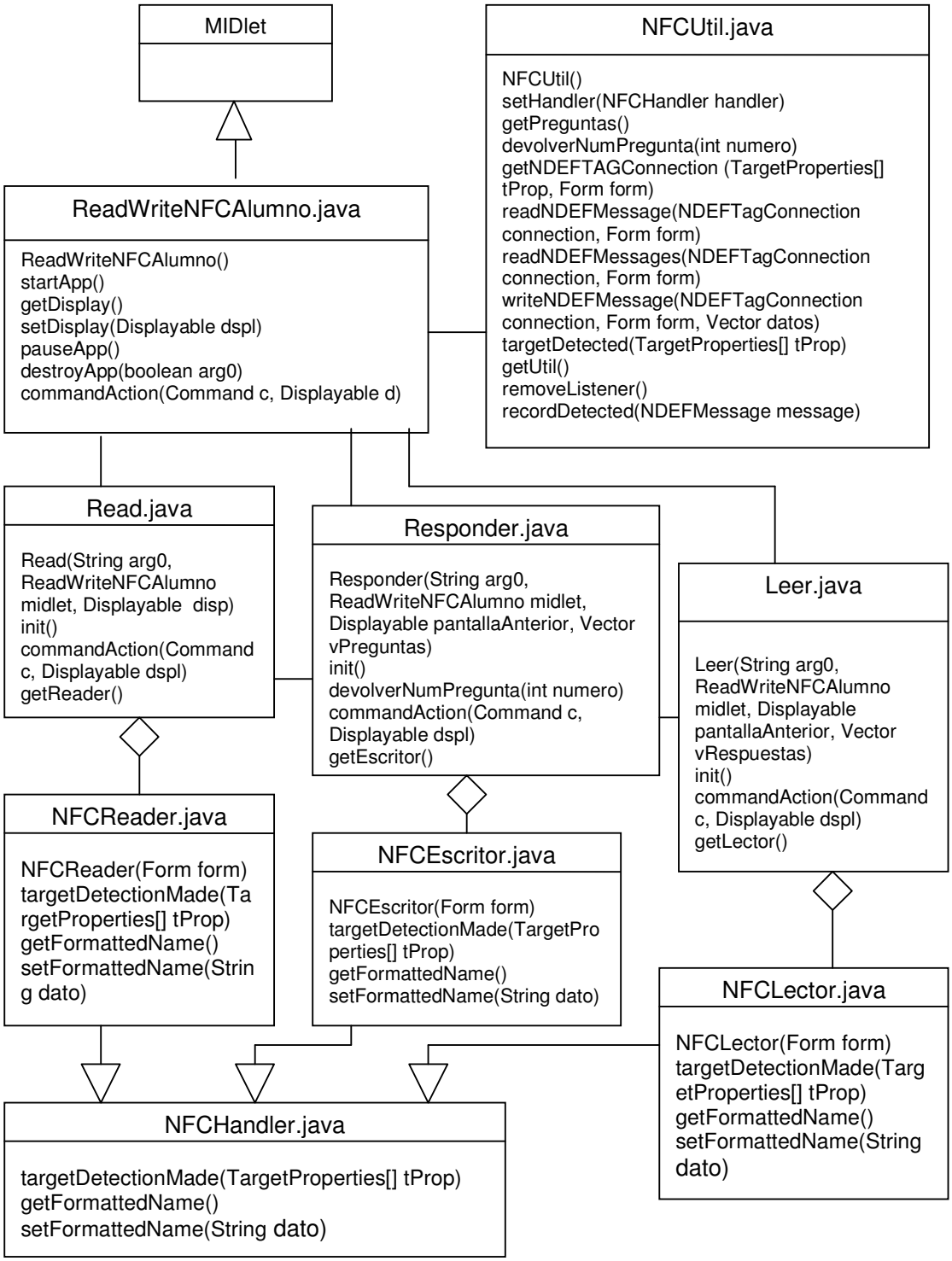


Figura 5.2: Relación UML de las clases del alumno

5.2 Clases y métodos

Una vez definida la relación entre las distintas clases, es importante explicar que hace cada una de ellas y los métodos que la forman

5.2.1 Clases y métodos para el módulo del profesor

- Clase *ReadWriteNFCProfesor*:

  Esta es la clase principal de este módulo. Es la que hace arrancar la aplicación profesor, ya que extiende de un Midlet.

  Por lo cual, los principales métodos que contiene son los que gestionan un midlet, en este caso:

  - *ReadWriteNFCProfesor ( )*: Método constructor que inicializa la pantalla (form), donde añadiremos los botones, imágenes y los ítems para seleccionar la tarea.
  - *startApp( ):* Método que se ejecuta cuando arranca la aplicación, por lo cual, es en este método donde debemos indicar el display a lanzar.
  - *getDisplay( ):* Método que permite obtener el display que se está lanzando.
  - *setDisplay(Displayable dspl):* Método para modificar el display que se va lanzar.
  - *pauseApp( ):* Método para detener el midlet.
  - *destroyApp(boolean arg0):* Método que permite poner a null todas las variables una vez que el midlet se ha parado.
  - *commandAction(Command c, Displayable d):* Método que permite detectar que opción ha sido seleccionada o el botón que se ha pulsado, y en función de ello realizar una tarea determinada.

- Clase *Examen*:

  Esta clase es la que se utiliza para generar un examen sin la utilización del móvil, y el cual podrá cargarse con una opción de la aplicación.

  - *Examen( ):* Método constructor en el cual se inicializan los atributos con las preguntas del examen.
  - *escribirExamen( ):* Este método permite rellenar el vector de preguntas y respuestas definidas en el constructor
  - *getPreguntas( ):* Método que devuelve el vector que contiene las preguntas y posibles respuestas.
  - *getRespuestas( ):* Método que devuelve el vector que contiene las respuestas correctas introducidas por el profesor.

- Clase *Guardar*:

  La clase Guardar es la que permite guardar las preguntas que se cargan de un examen ya definido, a una tarjeta mifare que se entregará a alumno.

  - *Guardar( ):* Constructor donde se inicializan los parámetros.
  - *init( ):* Este método permite mostrar los datos en pantalla para que el usuario los visualice antes de guardarlos en la tarjeta.
  - *commandAction(Command c, Displayable dspl):* Método que permite detectar que botón que se ha pulsado, y en función de ello realizar una tarea determinada.
  - devolverNumPregunta(int numero): Método que devuelve un numero con formato String a partir de otro que es entero.
  - getWriter( ): Este método devuelve el objeto escritor que permite escribir en la tarjeta.
  - obtenerRespuestas( ): Método que devuelve un Vector con las respuestas correctas escritas por el profesor.

- Clase *NFCEscritor*:

  Esta clase es la que detecta las propiedades de la tarjeta para establecer la conexión con ella y poder así escribir y/o leer en/de la misma.

  - *NFCEscritor(Form form):* Constructor que inicializa las variables, en este caso, permite inicializar la variable form, para controlar la pantalla anterior.
  - *targetDetectionMade(TargetProperties[ ] tProp):* Método que detecta las propiedades de la tarjeta y realiza la conexión con ella y de esta manera poder llamar a los métodos de lectura y escritura.
  - *getFormattedName( ):* Método que devuelve un vector con los datos escritos en la tarjeta.
  - *setFormattedName(String dato):* Método que permite añadir al vector de datos uno nuevo que se escribirá en la tarjeta.

- Clase *NFCUtil*:

  Esta clase es la que realiza las operaciones de, detección de propiedades de la tarjeta, así como la lectura y escritura de/en la misma.

  - *NFCUtil()*: Constructor donde se inicializan las variables.
  - *setHandler(NFCHandler handler):* Método que permite modificar el manejador de la tarjeta.
  - *respuestasAlumno():* Este método devuelve un vector que contiene las respuestas del alumno que han sido leídas de la tarjeta.
  - *devolverNumPregunta(int numero):* Método que, a partir de un número entero, devuelve un número en formato string que identifica la respuesta.

- *getNDEFTAGConnection(TargetProperties[] tProp,Form form):* Método que gestiona las propiedades de la tarjeta para encontrar NDEFTagConnection.
- *readNDEFMessage(NDEFTagConnection connection, Form form*): Método que permite leer de la tarjeta.
- *writeNDEFMessage(NDEFTagConnection connection, Form form, Vector datos):* Método que permite escribir en la tarjeta.
- *targetDetected(TargetProperties[] tProp):* Método para detectar las propiedades de la tarjeta.
- *getUtil():* Método para obtener los útiles que gestionan la tarjeta.
- *removeListener():* Método que permite borrar el TargetListener, el cual es el escuchador de la tarjeta para añadir o leer registros de ella.
- *recordDetected(NDEFMessage message):* Método que permite detectar si hay datos o no en la tarjeta

- Interfaz *NFCHandler*:

  Este interfaz es implementado en las clases NFCEscritor, NFCWriter y NFCReader. Por lo cual, la funcionalidad de cada uno de los métodos que la componen, están explicados en dichas clases.

- Clase *Write*:

  Clase que permite mostrar una nueva pantalla en la aplicación, donde el profesor puede introducir las preguntas y posibles respuestas que conformaran el examen.

  - *Write(String arg0, ReadWriteNFCProfesor midlet, Displayable pantallaAnterior):* Método constructor donde inicializamos las variables.

- *init():* Método con el que iniciamos esta pantalla. Muestra al usuario campos de texto donde introducir los datos del examen.
- *commandAction(Command c, Displayable dspl):* Método escuchador de eventos. Permite controlar el botón pulsado y en función de ello, realizar una determinada tarea.
- *pedirRespuestas():* Método que solicita al usuario que introduzca las respectivas respuestas (a, b, c, ó d) y cual de ellas es la correcta.
- *addScore(String numPregunta, String respuesta, String fin):* Método que añade las respuestas correctas del profesor al recordStore.
- *getWriter():* Método que devuelve el objeto escritor necesario para escribir en la tarjeta.
- *obtenerRespuestas():* Método que nos permite obtener el vector que contiene el conjunto de respuestas correctas introducidas por el profesor

o Clase *NFCWriter*:

Esta clase es la que permite obtener las propiedades y manejadores de la tarjeta para poder escribir en ella.

- *NFCWriter(Form form):* Constructor que inicializa las variables, en este caso, permite inicializar la variable form, para controlar la pantalla anterior.
- *targetDetectionMade(TargetProperties[ ] tProp):* Método que detecta las propiedades de la tarjeta y realiza la conexión con ella y de esta manera poder llamar a los métodos de escritura.
- *getFormattedName( ):* Método que devuelve un vector con los datos escritos en la tarjeta.
- *setFormattedName(String dato):* Método que permite añadir al vector de datos uno nuevo que se escribirá en la tarjeta.

- Clase *Read*:

  Esta es la clase que permite mostrar una nueva pantalla donde, se solicita que se acerque la tarjeta para leerla. Es esta clase la que se encarga de obtener el lector de tarjeta, para pasar a una nueva clase donde leerla.

  - *Read(String arg0, ReadWriteNFCProfesor midlet, Displayable disp, Write escribe, Guardar guardarPreguntas):* Método constructor que inicializa las variables.
  - *init():* Método inicial de esta pantalla, que utilizamos para agregar botones a la nueva pantalla y para obtener el lector NFC en caso de que sea null.
  - *commandAction(Command cmd, Displayable arg1):* Método que permite dar funcionalidad a los botones.
  - *getReader():* Método que devuelve el lector NFC

- Clase *NFCReader*:

  Esta es la clase que hace las operaciones de lectura de la tarjeta.

  - *NFCReader(Form form):* Método constructor que permite inicializar las variables, en este caso, el form que gestiona las pantallas.
  - *targetDetectionMade(TargetProperties[ ] tProp):* Método que detecta las propiedades de la tarjeta y realiza la conexión con ella y de esta manera poder llamar a los métodos de lectura.
  - *getFormattedName( ):* Método que devuelve un vector con los datos escritos en la tarjeta.
  - *setFormattedName(String dato):* Método que permite añadir al vector de datos uno nuevo que se escribirá en la tarjeta. Este método no será utilizado porque solo estamos leyendo, pero al extender del interfaz NFCHandler, debe aparecer.

- Clase *Corregir*:

  Esta es la clase que permite corregir un examen.

  - *Corregir(String arg0, ReadWriteNFCProfesor midlet, Displayable pantallaAnterior, Vector vRespuestasAlumno, Write escribir, Guardar guardarPreguntas):* Método constructor que inicializa las variables.
  - *init():* Método que llamamos para agregar botones a la nueva pantalla y para comenzar a corregir el examen.
  - *corregirExamen():* Método que compara el vector de respuestas del alumno con el vector de respuestas del profesor y determina cuantas preguntas fueron acertadas y cuantas falladas.
  - calcularNota (int numTotalRespuestas, int respuestasCorrectas): Método que calcula la nota del examen, en función de las respuestas acertadas.
  - *commandAction(Command c, Displayable dspl):* Método que permite controlar que botón se ha pulsado.

5.2.2 Clases y métodos para el módulo del alumno

- Clase *ReadWriteNFCAlumno*:

  Esta es la clase principal para leer o escribir de la tarjeta, permite arrancar el midlet de la aplicación.

  - *ReadWriteNFCAlumno( ):* Método constructor donde inicializamos las variables y creamos la pantalla donde seleccionar operación.
  - *startApp( ):* Método que se ejecuta cuando arranca la aplicación, por lo cual, es en este método donde debemos indicar el display a lanzar.

- *getDisplay( ):* Método que permite obtener el display que se está lanzando.
- *setDisplay(Displayable dspl):* Método para modificar el display que se va lanzar.
- *pauseApp( ):* Método para detener el midlet.
- *destroyApp(boolean arg0):* Método que permite poner a null todas las variables una vez que el midlet se ha parado.
- *commandAction(Command c, Displayable d):* Método que permite detectar que opción ha sido seleccionada o el botón que se ha pulsado, y en función de ello realizar una tarea determinada.

- Clase ***Read***:

  Esta es la clase que permite mostrar una nueva pantalla, donde, se solicita que se acerque la tarjeta para leerla. Es esta clase la que se encarga de obtener el lector de tarjeta, para pasar a una nueva clase donde leerla.
  - *Read(String arg0, ReadWriteNFCAlumno midlet, Displayable disp):* Método constructor que inicializa las variables.
  - *init( ):* Método que llamamos para agregar botones a la nueva pantalla y para obtener el lector NFC en caso de que sea null.
  - *commandAction(Command cmd, Displayable d):* Método que permite dar funcionalidad a los botones.
  - *getReader( ):* Método que devuelve el lector NFC

- Clase ***NFCReader***:

  Esta es la clase que hace las operaciones de lectura de la tarjeta.
  - *NFCReader(Form form):* Método constructor que permite inicializar las variables.
  - *targetDetectionMade(TargetProperties[ ] tProp):* Método que permite detectar una tarjeta a partir de sus propiedades.

- *getFormattedName( ):* Método que permite devolver el vector que contiene los datos de la tarjeta.
- *setFormattedName(String dato):* Método que permite almacenar un string al vector de datos, el cual se almacenará en la tarjeta. Sin embargo, no se implementa este método porque solo se realizan lecturas, pero como extiende de NFCHandler hay que mostrarlo.

- Clase *NFCUtil*:

  Esta es la clase que maneja los utilities para realizar las operaciones de lectura y escritura de la tarjeta.
  - *NFCUtil()*: Constructor donde se inicializan las variables.
  - *setHandler(NFCHandler handler):* Método que permite modificar el manejador de la tarjeta.
  - *getPreguntas():* Método que devuelve un vector con las preguntas del profesor, leídas de la tarjeta.
  - *devolverNumPregunta(int numero):* Método que, a partir de un numero entero, devuelve un número en formato string que identifica la respuesta.
  - *getNDEFTAGConnection(TargetProperties[] tProp,Form form):* Método que gestiona las propiedades de la tarjeta para encontrar NDEFTagConnection.
  - *readNDEFMessage(NDEFTagConnection connection, Form form*): Método que permite leer de la tarjeta las preguntas del profesor.
  - *readNDEFMessages(NDEFTagConnection connection, Form form):* Método que permite leer de la tarjeta las respuestas almacenadas por parte del alumno.
  - *writeNDEFMessage(NDEFTagConnection connection, Form form, Vector datos):* Método que permite escribir en la tarjeta.

- *targetDetected(TargetProperties[] tProp):* Método para detectar las propiedades de la tarjeta.
- *getUtil():* Método para obtener los útiles que gestionan la tarjeta.
- *removeListener():* Método que permite borrar el TargetListener, el cual es el escuchador de la tarjeta para añadir o leer registros de ella.
- *recordDetected(NDEFMessage message):* Método que permite detectar si hay datos o no en la tarjeta

- Interfaz *NFCHandler*:

  Este interfaz es implementado en las clases NFCEscritor, NFCLector y NFCReader. Por lo cual, la funcionalidad de cada uno de los métodos que la componen, están explicados en dichas clases.

- Clase *Responder:*

  Clase que permite responder las preguntas de un examen.
  - *Responder(String arg0, ReadWriteNFCAlumno midlet, Displayable pantallaAnterior, Vector vPreguntas):* Método constructor que inicializa las variables.
  - *init( ):* Método inicial de la pantalla, que permite mostrar una a una las preguntas del examen para que el alumno las pueda contestar.
  - *devolverNumPregunta(int numero):* Método que, a partir de un numero entero, devuelve un número en formato string que identifica la respuesta.
  - *commandAction(Command c, Displayable dspl):* Método que permite controlar que botón se ha pulsado.
  - *getEscritor():* Método que devuelve el escritor para poder escribir en tarjeta.

- Clase *NFCEscritor:*

  La clase NFCEscritor permite obtener las propiedades y manejadores de la tarjeta para poder escribir en ella.

  - NFCEscritor(Form form): Método constructor que inicializa las variables.
  - *targetDetectionMade(TargetProperties[ ] tProp):* Método que detecta las propiedades de la tarjeta y establece la conexión con ella.
  - *getFormattedName( ):* Método que permite devolver el vector de datos escritos en la tarjeta.
  - *setFormattedName(String dato):* Método que permite escribir en el vector de datos que serán escritos en la tarjeta.

- Clase *Leer*:

  Esta es la clase que hace las operaciones de lectura de la tarjeta, y muestra al alumno lo que a respondido.

  - *Leer(String arg0,ReadWriteNFCAlumno midlet,Displayable pantallaAnterior, Vector vRespuestas):* Método constructor que inicializa las variables.
  - init( ): Método inicial de la pantalla, que pide mostrar la tarjeta donde guardó las respuestas y donde se añaden los botones que conforman las pantallas.
  - *commandAction(Command c, Displayable dspl):* Método que permite controlar que botón se ha pulsado.
  - getLector( ): Método que devuelve el lector NFC.

- Clase *NFCLector*:

  Esta es la clase que hace las operaciones de lectura de la tarjeta. Principalmente, obtener sus propiedades para luego poder hacer la lectura.

- *NFCLector(Form form):* Método constructor que inicializa las variables.
- *targetDetectionMade(TargetProperties[ ] tProp):* Método que detecta las propiedades de la tarjeta y establece la conexión con ella.
- *getFormattedName( ):* Método que permite devolver el vector de datos escritos en la tarjeta.
- *setFormattedName(String dato):* Método que permite almacenar un string al vector de datos, el cual se almacenará en la tarjeta. Sin embargo, no se implementa este método, porque solo se realizan lecturas, pero como extiende de NFCHandler hay que mostrarlo.

Capitulo 6. Descripción funcional

Tras haber introducido la explicación de todos y cada una de los elementos necesarios para la elaboración de este proyecto, en este capitulo se va a proceder a explicar cual es la funcionalidad del mismo.

El propósito de este proyecto es el estudio de nuevas aplicaciones, que haciendo uso de los últimos avances en las TICs, puedan ser implantadas en el entorno universitario con la finalidad de ofrecer a alumnos y profesores información que pueda ser utilizada para la mejora del nuevo modelo de enseñanza.

Estas soluciones, basadas en el uso de la telefonía móvil y la tecnología NFC (Near Field Communication), la primera de uso común en la sociedad y, por lo tanto, en cualquier estamento universitario, y la segunda de nuevo desarrollo y futura y amplia implantación, pretenden aportar un nuevo sistema de evaluación distinto al que se ha venido realizando.

El estudio se basa en una aplicación profesor y una aplicación alumno que simula la realización y corrección de un examen. En ambos escenarios son muy pocos los nuevos elementos que debemos de incorporar, tan solo un teléfono móvil que soporte la tecnología y las tarjetas que almacenan la información, pero por el contrario, es totalmente revolucionario el conjunto de ventajas que aporta este nuevo concepto, que en todo momento están relacionadas con hacer más sencillas la realización de estas operaciones cotidianas, y lo que es más importante una total convivencia con la tecnología actual.

Únicamente contaremos con nuestro teléfono móvil que poseerá la tecnología NFC, al que incorporaremos una pequeña aplicación realizada en J2ME.

Para el caso del profesor, esta aplicación J2ME, lo que hará será mostrar las posibles opciones de creación del examen que el profesor puede elegir, en este caso, podrá cargar un examen ya editado con anterioridad o crearlo directamente con el teléfono móvil.

Figura 6.1: Carga de examen en el móvil

Figura 6.2: Edición de examen en el móvil

Una vez el profesor ha cargado o editado el examen, lo que se debe realizar en esta ocasión es escribir los datos en la tarjeta, sin embargo, antes de ello, la propia aplicación te ofrece la posibilidad de ver las preguntas editadas y en caso de no estar conforme reeditarlas de nuevo, o ya proceder al paso de escritura, la cual se realizará gracias a que la propia aplicación J2ME dispone de un botón que permite decidir cual va a ser el momento de escribir los datos en la tarjeta, la cual será entregada al alumno para que las preguntas del examen sean respondidas.

Para realizar la operación en que el alumno responde las preguntas que con anterioridad el profesor ha editado, existe otra aplicación J2ME.

Esta otra aplicación del alumno consiste en leer de la tarjeta las preguntas, las cuales se muestran al alumno en la pantalla del móvil. A continuación, el alumno responderá a las preguntas y guarda en su tarjeta dichas respuestas, que entregará al profesor para su posterior corrección.

Figura 6.3: Aplicación de respuestas del alumno

Al igual que en el caso del profesor, antes de que el alumno acerque la tarjeta para escribir en ella las respuestas, puede ver lo que ha contestado, y en caso de no estar conforme, puede volverlas a editar.

Una vez el profesor tiene las tarjetas de los alumnos con las respuestas, otra de las opciones que el profesor podrá seleccionar es la de corregir el examen, lo que permitirá obtener cuantas respuestas contestó el alumno correctamente y cuales no.

Figura 6.4: Corrección de examen en el móvil

Como se puede observar, aprovechando la tecnología NFC, se pueden ofrecer unos servicios innovadores y complementarios a los servicios disponibles en la actualidad. Esta nueva tecnología integrada en el teléfono móvil no se concibe como una tecnología que elimine a los dispositivos que se tienen montados, ya que puede convivir perfectamente con la tecnología ya existente, haciendo que los servicios que ya se ofrecen se puedan extender, facilitando así el día a día de los usuarios en sus actividades.

Capítulo 7. Pruebas

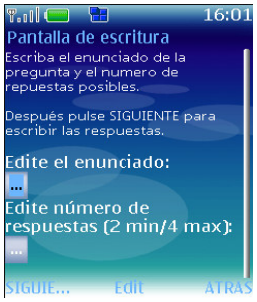
Una vez hemos explicado los elementos que componen la aplicación, la funcionalidad requerida, el desarrollo software y demás características, es importante realizar una fase de pruebas, en la que comprobar las prestaciones de la aplicación y principalmente, ver que todo funciona como debería.

Es por todo ello, por lo que es necesario realizar este apartado de pruebas, las cuales se explican, una a una, a continuación.

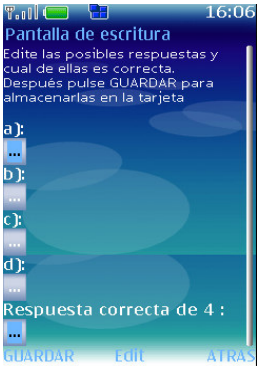
7.1 Pruebas de funcionalidad

7.1.1 Funcionalidad del módulo del profesor

- Escritura de examen:
  - Simulador: Se prueba la funcionalidad de escritura manual del examen en el simulador y conseguimos realizar la funcionalidad esperada.



La cual consiste en primer lugar solicitar al usuario, en este caso el profesor, que edite la el enunciado de la pregunta y el número de posibles respuestas entre las que tendrá que elegir el alumno.



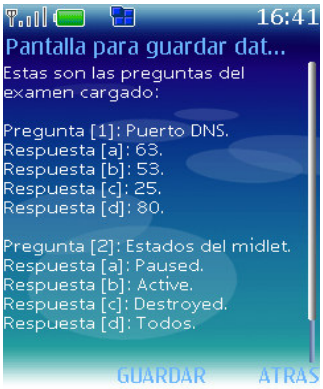
A continuación, una vez pulsamos siguiente, pasamos a la pantalla de introducir las posibles respuestas, la cual también funciona correctamente ya está en concordancia con el número de respuestas que le hemos introducido.

Y finalmente comprobamos que escribe correctamente los datos en la tarjeta, cuando la acercamos al móvil.

- Teléfono móvil: Las mismas pruebas que se realizaron para el simulador, fueron realizas con el dispositivo móvil, sin embargo, encontramos una variabilidad.
  Con esto nos referimos a que a la hora de almacenar en la tarjeta real, comienzan los problemas de tamaño, ya que en el simulador no hay límite, en contraposición a la tarjeta real, la cual solo nos permite un tamaño máximo de 96 bytes. Motivo por el cual se redujo el número de preguntas. Por lo demás, todo funciona correctamente.

- Carga de examen:

  - Simulador: En este caso, la funcionalidad requerida se consigue completamente.
    Se solicita al profesor que cargue un examen de una aplicación anteriormente desarrollada, y tras ver que las preguntas cargadas son las deseadas, las almacena en la tarjeta Mifare.

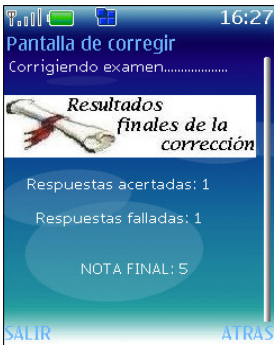


  - Teléfono móvil: Al igual que ocurría en el caso anterior, el problema surgido a la hora de ejecutar la aplicación en el dispositivo móvil, encontramos el problema del tamaño máximo de almacenaje permitido por la tarjeta, por lo cual, era necesario reducir el número de preguntas generadas por la aplicación que cargaba el examen.

- Corrección de examen:

  - Simulador: Si probamos la funcionalidad de este apartado, obtenemos los objetivos planteados.

    En este caso lo que se plantea es leer de la tarjeta las respuestas introducidas por el alumno, para que así, el profesor las pueda corregir, mostrando en la pantalla el número de fallos y aciertos, así como la calificación.



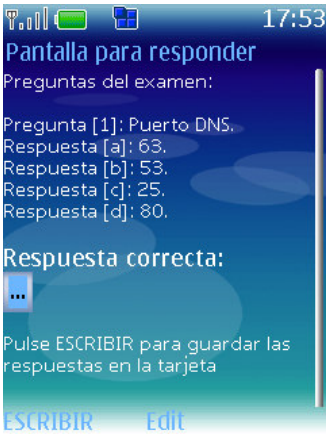
- Teléfono móvil: Como se ha comentado en la mayoría del resto de casos, la funcionalidad probada en los dispositivos reales, se consigue perfectamente, pero sigue apareciendo el problema de la capacidad de las tarjetas. Aunque en este caso, este problema no afecte directamente porque solo se lee de ella, afecta en el alumno, cuando intente escribir demasiadas respuestas y superen los 96bytes de tamaño máximo permitido.

7.1.2 Funcionalidad del módulo del alumno

- Lectura y contestación de examen:
  - Simulador: Al igual que ocurre en la aplicación del profesor, en el caso del alumno, sucede lo mismo, la funcionalidad requerida cumple con los objetivos.

    Ya que cuando se solicita en la aplicación que se lea de la tarjeta y muestre las preguntas, lo realiza correctamente, y va mostrando una a una cada pregunta, para que el alumno pueda responderlas.

    Finalmente, esas respuestas son las que se almacenan en la tarjeta para que el profesor pueda corregirlas.



  - Teléfono móvil: Toda esta funcionalidad probada en el simulador, se probó de manera real en el teléfono móvil y se comprobó que todo iba correctamente, a excepción de la capacidad de almacenamiento de la tarjeta Mifare que se ha comentado en apartados anteriores, que imposibilita almacenar grandes cantidades de datos.

7.2 Pruebas de capacidad

7.2.1 Capacidad de la tarjeta

Sería redundante volver a comentar el principal inconveniente que conlleva el utilizar tarjetas Mifare en el desarrollo de esta aplicación.

Ya que como hemos comentado en cada uno de los casos anteriores, estas tarjetas, tienen una capacidad máxima de memoria de 96bytes lo que imposibilita la creación de exámenes de gran envergadura, así como minimiza el número de respuestas introducidas por el alumno.

Todo esto da lugar a un estudio mas exhaustivo de las capacidades y ventajas e inconvenientes de esta tecnología, al igual que provoca un retardo en su implantación dentro del entorno real.

Capítulo 8. Conclusiones y líneas futuras

8.1 Conclusiones

Son muchas las aplicaciones existentes en el mercado que utilizan el móvil para realizar determinadas tareas, pues porqué una de ellas no puede ser la realización de un examen, tarea muy común en el entorno estudiantil.

Sin embargo, hasta que no se está en su desarrollo, y se realizan las respectivas pruebas, no nos percatamos de las ventajas e inconvenientes que este sistema conlleva. Es a partir de estas ventajas e inconvenientes donde el desarrollador obtiene conclusiones, como es por ejemplo, el tema de la capacidad. Y en nuestro caso, ocurre que las tarjetas, no soportan cantidades de datos grandes, principal inconveniente, debido a que un examen está compuesto por un tamaño relativamente grande.

Pero por otro lado, sí podemos decir que en este trabajo se ha presentado una introducción a esta novedosa tecnología, a la vez que, mostramos una primera aproximación a su uso en grandes superficies como es la universidad, aunque todavía esté lejos de su implantación.

Como se ha mostrado en este proyecto, esta nueva tecnología integrada en el teléfono móvil no ha llegado a invadir al ser humano, como usuario de la misma. Además, tampoco se concibe como una tecnología que elimine a los dispositivos que se tienen montados en los diferentes contextos, ya que puede convivir perfectamente con la tecnología ya existente, haciendo que los servicios que ya se ofrecen se puedan extender, facilitando así el día a día de los usuarios en sus actividades.

8.2 Líneas futuras

A pesar del buen desarrollo de esta aplicación y la particularidad de la misma, existen otros puntos importantes a desarrollar que harán de esta aplicación más usual y práctica. Entre ellos cabe destacar los siguientes:

- Carga de preguntas remotamente:

  Igual que el modulo del profesor permite cargar un examen a partir de una aplicación, también podría ser cargar ese examen desde otro lugar, como puede ser otro dispositivo móvil, o incluso bajarlo de la Web.

- Escritura en un fichero de texto, del cual cargaría la aplicación el examen:

  De esta manera, un usuario (el profesor), redactaría normalmente el examen en un fichero de texto, con un formato determinado, y a la hora de seleccionar la opción de cargar un examen, lo único que tendríamos que hacer es leer mediante un sistema de ficheros.

Figura 8.1: Carga de examen de un fichero

- Capacidad de publicar las notas en la Web:

  Esta otra opción que puede ser incorporada, podría dar la posibilidad de introducir una URL tras la corrección del examen, en la cual las notas serán publicadas. De esta manera, los alumnos podrán acceder a ella desde un PC o incluso desde su dispositivo móvil.

Figura 8.2: Publicación de notas en la Web

Capitulo 9. Planificación y Presupuesto

9.1 Estimación de la planificación de tareas

En este apartado, se muestra una tabla con la estimación de la duración de las distintas tareas a realizar para la consecución de este proyecto.

Como se puede apreciar, el gráfico representa una estimación aproximada por semanas, sin embargo, en este mismo capítulo, mas adelante aparece la estimación de cada tarea por horas, y el coste que supone.

Semanas / Tareas	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A														
B														
C														
D														
E														
F														

Tabla 9.1: Planificación de tareas

Relación de Tareas:

A: Búsqueda de información
B: Toma de contacto con la tecnología
C: Ideas de desarrollo y diseño de la aplicación
D: Desarrollo de la aplicación
E: Pruebas
F: Redacción de la memoria

9.2 Costes por Equipos y Componentes Empleados

En este capítulo se muestra la relación de equipos, componentes y software empleados en la realización de este proyecto, así como el precio de éstos y el presupuesto total.

Referencia dispositivo	Fabricante	Descripción	Precio/ Unidad	Unidades	Precio total
Nokia 6131 NFC	Nokia	Teléfono móvil que contiene la tecnología NFC	190 €	2	380 €
Tarjetas inteligentes MIFARE 1k	Philips	tarjetas inteligentes sin contacto de memoria protegida	0,80 €	2	1,60 €
PC	-	Ordenador que soporte software wireless toolkit y nokia PC Suit	550€	1	550 €
Sun Java (TM) Wireless Toolkit 2.5.1 for CLDC	Java	Emulador de aplicaciones para los celulares	gratuito	1	0 €
API de Nokia 6131 NFC	Nokia	Interfaz de programación de aplicaciones que contiene las herramientas NFC	gratuito	1	0 €
Java (TM) SE Development Kit 6 Update 5	Java	Software con herramientas de desarrollo para la creación de programas en java	gratuito	1	0 €
TOTAL					**931,60 €**

Tabla 9.2: Listado de materiales y componentes

Presupuesto total para los componentes (IVA incluido): **931,60 Euros**

9.3 Costes de personal

Una vez se han detallado los costes por los equipos empleados en este Proyecto de Fin de Carrera, a continuación, se muestran los costes debidos al personal que han participado en la realización del mismo, detallados según horas estimadas de trabajo y tareas.

Principalmente, para la realización de un proyecto de esta envergadura donde se requieren conocimientos de esta tecnología, se ha necesitado: un Ingeniero y un ingeniero técnico de telecomunicaciones. A los cuales les asignamos el siguiente salario mensual neto estimado.

Trabajador	Salario Mensual (Euros)
Ingeniero Telecomunicaciones	3000
Ingeniero Técnico Telecomunicaciones	1500

Tabla 9.3: Salario mensual de cada trabajador

A partir de estos salarios (libres de impuestos) y tomando como un total de horas laborables al mes aproximadamente de 160, se muestra en la siguiente tabla el salario por hora de cada trabajador.

Trabajador	Euros/Hora
Ingeniero de telecomunicaciones	18
Ingeniero Técnico Telecomunicaciones	8.5

Tabla 9.4: Salario por hora de cada trabajador

Con los datos anteriormente mostrados, se puede estimar el coste total por personal y tarea desempeñada a lo largo de este Proyecto.

Tarea	**Trabajador**	**Horas**	**Coste Total (Euros)**
Organización y gestión del proyecto	Ingeniero de Telecomunicaciones	125	2250
Búsqueda de información	Ingeniero Técnico Telecomunicaciones	60	510
Toma de contacto con la tecnología NFC	Ingeniero Técnico Telecomunicaciones	45	382,5
Ideas de desarrollo y diseño de la aplicación	Ingeniero Técnico Telecomunicaciones	70	595
Desarrollo de la aplicación	Ingeniero Técnico Telecomunicaciones	210	1785
Pruebas	Ingeniero Técnico Telecomunicaciones	80	680
Preparación y redacción de la memoria	Ingeniero Técnico Telecomunicaciones	75	637,5
TOTAL		**665**	**6.840**

Tabla 9.5: Relación de tareas y coste estimado

Todo esto supone un coste por personal de **6.840 Euros.**

9.4 Coste total del proyecto

Sumando los costes por material y personal se obtiene el presupuesto total estimado para la realización de este proyecto.

Costes	Presupuesto (Euros)
Costes por material	931,60
Costes por personal	6.840
Total	**7.771,6**

Tabla 9.6: Presupuesto total.

El presupuesto total estimado de este Proyecto de Fin de Carrera es de: **siete mil setecientos setenta y un euros con sesenta céntimos.**

Anexo

Instalación del software de desarrollo

- ✓ En primer lugar procedemos a la instalación de Java (TM) SE Development Kit 6 Update 5 que podemos encontrarlo en: https://sdlc1d.sun.com/ECom/EComActionServlet/DownloadPage:~:com.sun.sunit.sdlc.content.DownloadPageInfo;jsessionid=2000C18D873F708F50786DA58762A9A8;jsessionid=2000C18D873F708F50786DA58762A9A8
  Esto nos permite tener la plataforma de desarrollo java en Windows.

  Para la instalación, solo es necesario seguir los pasos que la propia ejecución genera, y seleccionar los distintos elementos que queremos instalar.

- ✓ A continuación se procede a la instalación del simulador que nos permite ejecutar la aplicación. Esta aplicación es el Sun Java (TM) Wireless Toolkit 2.5.1 for CLDC, disponible en: http://java.sun.com/products/sjwtoolkit/download-2_5_1.html

  Igual que en el caso anterior, solo se necesita seguir los pasos de la propia ejecución.

- ✓ Por último para ejecutar nuestra aplicación que utiliza NFC, es necesario instalar un API de Nokia (Nokia_6131_NFC_SDK_Release_1.0.zip), disponible en:
  http://wiki.forum.nokia.com/index.php/Nokia_6131_NFC_-_FAQs

Para ello descomprimimos el archivo y lo instalamos en nuestro ordenador en el directorio por defecto.

A continuación el fichero "Nokia_6131_NFC_SDK_1_0" que se encuentra en "C:\Nokia\Devices" lo copiamos en "C:\WTK2.5.1\wtklib\devices".

Por otro lado, también es necesario copiar el archivo nfc.jar de la carpeta" C:\WTK2.5.1\wtklib\devices\Nokia_6131_NFC_SDK_1_0\lib\ext" en la carpeta 'ext' la cual se encuentra en el directorio: 'C:\WTK25\lib\ext'.

Términos y Acrónimos

Términos / Acrónimos	Término Completo	Descripción
NFC	*Near Field Communication*	Tecnología inalámbrica de corto alcance que permite a un dispositivo almacenar los datos de otros que estén próximos entre sí.
NFC Forum	*Foro NFC*	Organización sin ánimo de lucro creada con el fin de fomentar esta tecnología y velar por la interoperabilidad entre dispositivos y servicios
TIC	*Tecnologías de la información y la comunicación*	Conjunto de tecnologías ligadas a las comunicaciones, la informática y los medios de comunicación y al aspecto social de éstas
ISO	*Internacional Standards Organization*	Organización Internacional para la estandarización, encargada de promover el desarrollo de normas internacionales de fabricación, comercio y comunicación
IEC	*International Electrotechnical Commission*	Órgano responsable de la estandarización de equipos eléctricos.
RFID	*Radio Frequency IDentification*	Tecnología basada en la identificación por radiofrecuencia que permite el almacenamiento y recuperación de datos remoto de una etiqueta (tag).

Tags		Tarjeta identificativa utilizada para guardar un identificador en el caso de la tecnología RFID, o para almacenar datos en otros casos como es NFC.
PDA	*Personal Digital Assistant*	Agendas personales electrónicas que tienen la capacidad para guardar todo tipo de datos
MIFARE		Tecnología de tarjetas inteligentes sin contacto de memoria protegida
SIM	*Subscriber Identity Module*	'Módulo de Identificación del Suscriptor'. Es una tarjeta inteligente usada en teléfonos móviles que almacena de forma segura la clave de servicio del suscriptor usada para identificarse ante la red.
sms	*Short Message Service*	'Servicio de mensajes cortos' Sistema de mensajes de texto para teléfonos móviles.
PC	*Personal Computer*	Ordenador personal. Es una máquina electrónica que recibe y procesa datos para convertirlos en información útil.
URL	*Uniform Resource Locator*	Localizador uniforme de recurso. Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.
LAN	*Local Area Network*	'Red de área local' Interconexión de varios ordenadores y periféricos para compartir recursos e intercambiar datos y aplicaciones.
ISD	*Issuer Security*	Sistema de seguridad implantado en el Nokia 6131 NFC basado en una clave

	Domain	de acceso.
PIN	*Personal Identification Number*	'Número personal de identificación' Código numérico que es usado en ciertos sistemas para obtener acceso a algo, o identificarse.
NXP Semiconductors	*Next eXPerience Semiconductors*	Es el nombre de una nueva compañía de semiconductores fundada por Philips.
NDEF	*NFC Data Exchange Format*	Especifica un formato común y compacto para el intercambio de datos
SDK	*Software Development Kit*	Kit de desarrollo de software. Es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema bastante concreto
API	*Application Programming Interface*	Interfaz de Programación de Aplicaciones Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
JSR	*Java Specification Requests*	Documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la plataforma Java.
NFCIP	*NFC Interface and Protocol*	Conexión que permite la comunicación punto a punto entre dispositivos NFC.
APDU	*Application Protocol Data Unit*	Unidad de Protocolo de Aplicación de datos. Es una unidad de comunicación entre el lector y la tarjeta.

J2SE	*Java Standard Edition*	La plataforma J2SE, es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.
RMI	*Java Remote Method Invocation*	Es un mecanismo ofrecido en Java para invocar un método remotamente. Por medio de RMI, un programa Java puede exportar un objeto.
JDBC	*Java Database Connectivity*	Es un API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede
J2ME	*Java 2 Micro Edition*	La plataforma Java 2, Micro Edition, o Java ME, es una colección de APIs en Java orientadas a productos de consumo como PDAs, teléfonos móviles o electrodomésticos.
MIDP	*Mobile Information Device profile*	Versión de J2ME integrada en el hardware de celulares relativamente modernos que permite el uso en estos de aplicaciones java denominadas MIDlets, tales como juegos, aplicaciones u otros.
CLDC	*Connected Limited Device Configuration*	Especificación de un marco para aplicaciones Java ME dirigidos a los dispositivos con recursos muy limitados como teléfonos móviles y PDAs.
CDC	*Connected Device Configuration*	Especificación realizada por Sun dentro del conjunto de tecnologías J2ME para computación móvil. Define capacidades básicas que debe tener un dispositivo móvil con capacidad de conexión.

C++	*pronunciado "ce más más" o "ce plus plus"*	Lenguaje de programación que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.
OSGi	*Open Services Gateway Initiative*	Iniciativa que define las especificaciones abiertas de software que permita diseñar plataformas compatibles que puedan proporcionar múltiples servicios.
AWT	*Abstract Window Toolkit*	(en español, Kit de Herramientas de Ventana Abstracta) Es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java.
JTWI	*Java Technology for the Wireless Industry*	La especificación de la tecnología Java para la industria inalámbrica (JTWI), define la plataforma del estándar de la industria para la próxima generación de tecnología Java para teléfonos móviles.
VM	*Virtual Machine*	Es un software de aplicación de una máquina (computadora) que ejecuta programas como una máquina real.
CPU	*Central Processor Unit*	La unidad central de procesamiento o, simplemente, el procesador es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de computadora.
RAM	*Random Access Memory*	La memoria de acceso aleatorio, o memoria de acceso directo se compone de uno o más chips y se utiliza como memoria de trabajo para programas y datos.

WAP	*Wireless Application Protocol*	Es un estándar internacional abierto para aplicaciones que utilizan la comunicación inalámbrica. Su uso principal es permitir el acceso a Internet desde un teléfono móvil o PDA.
iMODE		Es un conjunto de tecnologías y protocolos diseñados para poder navegar a través de minipáginas diseñadas específicamente para dispositivos móviles como teléfonos o PDAs.
PalmOS	*Palm Operating System*	Sistema operativo hecho por PalmSource, Inc. para computadores de mano (PDAs) fabricados por varios licenciatarios.
DTV	*Digital Television Systems*	Se refiere a la transmisión y recepción de imágenes en movimiento y sonido por medio de señales discretas (digital), en contraste con las señales analógicas usadas por la TV analógica.
JCP	*Java Community Process*	Es un proceso oficial que permite a las partes interesadas a que participen en la definición de las futuras versiones y características de la plataforma Java.
CVM	*Compact Virtual Machine, C Virtual Machine*	Completa y equipada JVM diseñado con mayor efecto para la próxima generación de electrónica de consumo y dispositivos integrados. CVM se distribuye como parte de la Connected Device Configuration (CDC).
KVM	*"Kilo" Virtual Machine, K Virtual Machine*	Maquina virtual diseñado para la próxima generación de electrónica de consumo y dispositivos integrados, que implementa la configuración CDC.

RISC	*Reduced Instruction Set Computer*	Computadora con Conjunto de Instrucciones Reducido. Es un tipo de microprocesador donde las Instrucciones son de tamaño fijo y presentadas en un reducido número de formatos.
CISC	*Complex Instruction Set Computer*	Modelo de arquitectura de computadores. Tiene un conjunto de instrucciones que se caracteriza por ser amplio y permitir operaciones complejas entre operandos situados en la memoria o en los registros.
AMS	*Application Management System*	Software gestor de aplicaciones que se encarga de gestionar el ciclo de vida de los MIDLets, entre otras cosas.
IGU	*Interfaz gráfica de Usuarios*	Apariencia de la pantalla que permite tener aplicaciones intuitivas y gráficas.
RMS	*Record Management System*	Es a la vez una aplicación y un API para almacenar records persistentes en dispositivos J2ME, como son los teléfonos celulares
OTA	*Over the Air*	Medio empleado para garantizar la descarga de un MIDLet.
JAR	*Java ARchive*	Es un tipo de archivo que permite ejecutar aplicaciones escritas en lenguaje Java. Estos archivos se utilizar normalmente para aplicaciones en teléfonos celulares.
JAD	*Java Application Descriptor*	Ficheros que se suelen utilizar para aplicaciones Java o el paquete de juegos que pueden ser descargados a los teléfonos móviles.

USB	*Universal Serial Bus*	(Bus universal en serie) Es un puerto que sirve para conectar periféricos a una computadora.
GPRS	*General Packet Radio Service*	Es un servicio de datos móvil orientado a paquetes. Permite velocidades de transferencia de 56 a 114 kbps.
UMTS	*Universal Mobile Telecommunications System*	El Sistema Universal de Telecomunicaciones móviles es una de las tecnologías usadas por los móviles de tercera generación.
HTTP	*HyperText Transfer Protocol*	El protocolo de transferencia de hipertexto (HTTP) es el protocolo usado en cada transacción de la Web. Define la sintaxis y la semántica que utilizan los elementos software de la arquitectura Web

Tabla T1: Relación de términos y acrónimos empleados

Bibliografía y referencias

[1] Todo sobre Near Field Communication. ¿Qué es NFC? ¿Por qué es importante? Aplicaciones con NFC, etc. Disponible en (última visita 5/4/2008): http://wiki.forum.nokia.com/images/6/6b/Nokia_NFC_white_paper.pdf

[2] Noticias y referencias en el periódico electrónico BBC Mundo.com. Disponible en (última visita 1/4/2008): http://news.bbc.co.uk/hi/spanish/science/newsid_6171000/6171448.stm

[3] Documentación acerca de las aplicaciones de las que dispone el dispositivo móvil Nokia NFC. Disponible en (última visita 17/1/2008): http://www.nokia.com/A4305081

[4] Descripción sobre las características del teléfono móvil de Nokia modelo 6131 NFC. Disponible en (última visita 2/4/2008): http://www.forum.nokia.com/info/sw.nokia.com/id/8a5911e5-2631-4b3e-9599-0f8959c2fb43/6131_NFC.html

[5] Preguntas frecuentes acerca de los contenidos del teléfono Nokia 6131 NFC. Disponible en (última visita 1/4/2008): http://wiki.forum.nokia.com/index.php/Nokia_6131_NFC_-_FAQs

[6] WIKIPEDIA. A cerca de NFC. Disponible en (última visita 11/4/2008): http://es.wikipedia.org/wiki/Near_Field_Communication

[7] WINDFALL. Información acerca de RFID. Disponible en (última visita 15/4/2008): http://www.windfallonline.com/RFID.htm

[8] NOKIA. Guía del usuario del Nokia 6131 NFC. Disponible en (última visita 17/4/2008): http://nds1.nokia.com/phones/files/guides/Nokia_6131_NFC_UG_es.pdf

[9] Introducción al lenguaje Java. Disponible en (última visita 17/4/2008): http://www.unav.es/cti/manuales/Java/indice.html

[10] J2ME y MIDP. Especificación del Modelo de Seguridad de MIDP. Disponible en (última visita 17/4/2008): http://www-sop.inria.fr/everest/personnel/Santiago.Zanella/thesis/Zanella.2006.Thesis.pdf

[11] API JAVA. Java 2 Platform Standard Edition 5.0 - API Specification. Disponible en (última visita 15/04/2009): http://java.sun.com/j2se/1.5.0/docs/api/

[12] API MIDlet. Interfaz de programación de aplicaciones definido para Mobile Information Device Profile. Disponible en (última visita 28/04/2009): http://arcad.essi.fr/riveill/enseignement/tp/j2me_fichiers/api/midp/javax/microedition/midlet/package-summary.html

[13] API MIDP. Interfaz de programación de aplicaciones que contiene un conjunto de características para implementación de aplicaciones MIDP. Disponible en (última visita 28/04/2009): http://www.j2medev.com/api/midp/javax/microedition/lcdui/package-summary.html

[14] API RMS. Interfaz de programación de aplicaciones para almacenamiento MIDP. Disponible en (última visita 05/05/2009): http://java.sun.com/javame/reference/apis/jsr118/javax/microedition/rms/package-summary.html

[15] API NFC. Interfaz de programación de aplicaciones para el desarrollo del código de conexión NFC. Disponible en (Última visita 10/05/2009): http://mobilezoo.biz/jsr/257/index.html