DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA IDENTIFICACIÓN DE ANIMALES DOMÉSTICOS MEDIANTE LA ESCRITURA Y LECTURA DE ETIQUETAS NFC A TRAVÉS DE DISPOSITIVOS PORTÁTILES

NÉSTOR ADOLFO NIÑO CARDOZO

Proyecto de Grado en modalidad de Monografía para optar al título de Ingeniero Electrónico

DIRECTOR:
PhD ROBERTO FERRO ESCOBAR

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
PROYECTO CURRICULAR DE INGENERÍA ELECTRÓNICA
BOGOTÁ D.C.
2016

RESUMEN

El presente trabajo es una recopilación del desarrollo de un sistema para la identificación de animales domésticos mediante el uso de tecnologías NFC, presentes en dispositivos portátiles. El uso de esta tecnología para implementar el sistema propuesto, no solo es interesante en cuanto a su facilidad de uso por parte del usuario, sino que además hace uso de una característica cada vez más común en los celulares y dispositivos de hoy en día para resolver el problema de la pérdida de mascotas, el cual se sigue presentando con mucha frecuencia en la actualidad. Cualquier persona que posea un *Smartphone* con tecnología NFC podría saber los datos de una mascota extraviada y ver el registro de vacunación del animal, además de los datos de su propietario. Este sistema no es invasivo y permitiría actualizar los datos en cualquier momento, desde cualquier *Smartphone* con la tecnología adecuada, y luego de un control de acceso al propietario de la mascota.

Palabras clave:

NFC, NDEF, Android, Java, XML.

TABLA DE CONTENIDO

Capítulo 1. Preliminares

1.1 INTRODUCCIÓN

NFC (Near Field Communication o comunicación de campo cercano, en español) es una tecnología de comunicación inalámbrica de corto alcance y alta frecuencia[1], que aunque no es una novedad, ha tomado una relevancia significativa, pues, cada vez más teléfonos "inteligentes" o Smartphones han incorporado entre sus características esta tecnología, que aún tiene mucho potencial en cuanto a sus posibles aplicaciones.

Por esta razón, se ha propuesto una aplicación, que a futuro, cuando la mayoría de dispositivos portátiles incorporen esta tecnología, solucionaría un problema que es crítico para las personas que comparten su vida con mascotas, el cual es la pérdida o extravío de éstas. Una mascota es un animal domesticado que se conserva con el propósito de brindar compañía. A diferencia de los animales de laboratorio, animales de crianza (o ganado), animales para el transporte o animales para el deporte; los animales de compañía no son conservados para traer beneficios económicos o alimenticios, son usados para traer un beneficio personal a su propietario[2].

En la actualidad, cuando se pierde una mascota, es muy difícil que su propietario vuelva a localizarla, y aunque existen algunas soluciones, tales como collares con una placa en la que se incluyen los datos personales del propietario, estas placas tienen la desventaja de poseer una cantidad de información muy limitada en cuanto a la mascota y en cuanto a la información de contacto incluida en ellas. Además el propietario no tiene la capacidad de actualizar sus datos ni la información de su mascota.

Otra alternativa consiste en implantar un microchip subdérmico[3]. En estos microchips se graba un número único de identificación animal, similar al número de cédula de una persona. A través de este número es posible localizar al propietario de una mascota perdida. Este método tiene la desventaja de que la persona que encuentra a la mascota extraviada, no podría saber si ésta posee tal microchip, por lo que tendría que dirigirse con la mascota hacía una veterinaria. Allí el veterinario a través de un lector especial RFID podría verificar si la mascota posee este microchip y leer su ID. Posteriormente utilizaría una conexión a internet para poder consultar en una base de datos quién es el propietario de la mascota extraviada. Este método además presenta el inconveniente de ser invasivo, ya que el microchip necesita implantarse por medio de una aguja debajo de la piel del animal por un veterinario profesional.

[1] https://es.wikipedia.org/wiki/Near_field_communication
[2] https://es.wikipedia.org/wiki/Animal_de_compa%C3%B1%C3%ADa
[3] https://es.wikipedia.org/wiki/Chip_subcut%C3%A1neo

1.2 PLANTEAMIENTO DEL PROBLEMA

En la actualidad un gran porcentaje de la población mundial posee un animal de compañía. Se estima que en Colombia, seis de cada diez familias cuentan con alguna mascota en su casa, siendo los perros los animales preferidos, seguidos por los gatos[4].

Ante estas cifras es evidente la necesidad de implementar algún sistema de identificación para mascotas, este sistema no solo debería permitir la ubicación del animal en caso de pérdida sino que además podría servir para determinar de manera rápida las características y el estado de vacunación del animal, en caso de mordeduras o en caso de que se presente algún síntoma de enfermedad en el animal.

1.3 JUSTIFICACIÓN

Día a día vemos en los postes carteles con fotos de mascotas desaparecidas. En estos carteles los desesperados dueños describen las características físicas de sus mascotas, incluso llegan a ofrecer jugosas recompensas con el fin de que algún transeúnte dé aviso rápidamente sobre el paradero de su mascota.

La implementación de un sistema para la identificación de animales domésticos, es una aplicación práctica y novedosa de la tecnología NFC, la cuál ha sido integrada ampliamente en muchos dispositivos de uso cotidiano y cada vez encuentra un espectro de aplicaciones más amplio. El uso de esta tecnología para implementar el sistema propuesto en este proyecto no solo es interesante en cuanto a la facilidad de uso por parte del usuario, sino que además hace uso de una característica cada vez más común en los celulares y dispositivos de hoy en día. Así que no habría necesidad de adquirir dispositivos especiales de lectura o escritura costosos para poder interpretar la información contenida en una etiqueta NFC.

Cualquier persona que posea un dispositivo con tecnología NFC podría obtener los datos de una mascota extraviada, tales como su nombre, raza, y su registro de vacunación; además de los datos de su propietario. Este sistema no es invasivo y permitiría actualizar los datos en cualquier momento, desde cualquier dispositivo portátil con la tecnología adecuada, y luego de un control de escritura al propietario de la mascota.

Se espera que con el tiempo, la masificación de esta tecnología en un número cada vez mayor de dispositivos portátiles haga posible que se reduzca significativamente el número de mascotas extraviadas.

[4] http://www.elespectador.com/noticias/economia/seis-de-cada-10-hogares-colombianos-hay-mascotas-articulo-540449

1.4 ESTADO DEL ARTE Y ANTECEDENTES

A pesar de que hoy en día existen múltiples métodos para identificar mascotas extraviadas, aún se sigue presentando esta situación con mucha frecuencia. Los métodos para evitar la pérdida de mascotas pueden variar desde una placa con un número de teléfono fijo o celular del propietario de la mascota, hasta elementos más sofisticados, tales como placas con un código QR, con el que se puede encontrar la información de la mascota y del propietario escaneándola y luego accediendo a una base de datos a través de internet. También existen implantes subcutáneos, los cuales consisten en una pequeña cápsula de cristal del tamaño de un grano de arroz, en cuyo interior se halla un microchip con un código único que permite la identificación del animal.

Los dispositivos mencionados en el párrafo anterior son dispositivos de tipo pasivo, es decir, no incluyen ni necesitan de una batería para su funcionamiento. Sin embargo, además de estos, también existen localizadores GPS, los cuales permiten hallar la ubicación de la mascota en un mapa satelital. Este tipo de dispositivos, aunque es más eficaz, también es mucho más costoso[5], debido a que emplea baterías para su funcionamiento.

Con el desarrollo y masificación de tecnologías inalámbricas, se han desarrollado una gran variedad de aplicaciones de identificación. Aunque la mayoría de estas aplicaciones ha sido desarrollada en el entorno de la tecnología predecesora del NFC, la tecnología RFID.

En general, existen varias aplicaciones biomédicas de la tecnología RFID. Existen trabajos en los cuales se propone la integración de dichos sistemas con sensores de bajo costo para el control de temperatura a distancia de un paciente (Catarinucci, Cappelli, Colella, Di Bari, & Tarricone, 2008).Estas propuestas de aplicación conducen a una mejora real en términos de eficacia y costos si se compara con los sistemas de códigos de barras antiguos u otros sistemas de identificación.

En los desarrollos más específicos consultados, en cuanto al tema de identificación de animales, se encuentra una propuesta de uso de etiquetas UHF RFID para hacer la trazabilidad de animales destinados al consumo humano (Pongpaibool, 2008). Esta propuesta está enfocada a identificar el lugar de origen de estos animales en caso de que lleguen a presentar enfermedades que se puedan transmitir a los seres humanos.

Siendo más específicos en el tema de identificación de animales domésticos, en un artículo publicado en el año 2007 se propone un sistema de monitoreo a través de dispositivos móviles basado en etiquetas RFID (Ting, Kwok, Lee, Tsang, & Cheung, 2007). Este sistema ayuda a los dueños de las mascotas a supervisar en tiempo real el estado y la localización de su mascota empleando para ello una red inalámbrica. Sin embargo este sistema requiere que a la mascota le sea implantado el microchip subdérmico convencional.

También se encontraron estudios que hacen uso de etiquetas RFID para relacionara todos los actores involucrados en el manejo de animales domésticos (tales como entidades ambientales del gobierno, vendedores de mascotas, la industria de fabricación de alimentos

[5] http://www.ocio.net/estilo-de-vida/mascotas/collar-localizador-gps-para-perros-locator/

para mascotas, clínicas de animales, asociaciones protectoras de animales, entre otros). El uso de etiquetas RFID se emplea para organizar la información en un modelo estructural que agrupa y recoge la información de cada sector en una base de datos disponible vía web para su intercambio con los otros sectores interesados (Liu & Shao, 2010). En un estudio similar se propone una solución distribuida y descentralizada a través de una infraestructura que elimina las demoras en la actualización y el intercambio de la información relacionada con las mascotas (Liu, 2007).

Inclusive, se han propuesto aplicaciones sociales a través del uso de etiquetas RFID en mascotas. En la edición de enero de 2008 de la revista IEEE Spectrum, se publicó un artículo (Upson, 2008), en el cual los ingenieros de la compañía MIT's Media proponían el lanzamiento de un producto que combinaba una etiqueta RFID para mascotas con un acelerómetro. La idea consistía en que, cuando el perro diera un paseo y se encontrara con otros perros que poseían el mismo producto, sus etiquetas intercambiarían sus identificadores. Al regresar cada perro a su casa, esta etiqueta transmitiría a una estación base cualquier código recogido durante este paseo. Así el dueño del perro podría obtener más información sobre los perros que su mascota encontró en el camino, junto con información sobre los propietarios de estos perros. De esta forma los propietarios se podrían relacionar, sin tener que charlar en persona.

Cabe destacar que en la bibliografía encontrada, todos los proyectos relacionados empleaban tecnología RFID, así que investigando un poco acerca de las desventajas de la implantación de estos microchips subcutáneos en las mascotas, se encontró algo sorpresivo y aterrador. Eneste estudio (Albrecht, 2007) se hacía una revisión exhaustiva de la literatura publicada en revistas de oncología y toxicología entre los años 1990 y 2006 frente a los efectos de la implantación de microchips RFID en roedores de laboratorio y perros. En ocho de los once artículos revisados, se encontró que la aparición de sarcomas malignos y otros tipos de cáncer se habían formado alrededor de la zona en la cual habían sido implantados los microchips RFID. De ésta forma, los investigadores concluyeron que los microchips, en casi todos los casos, eran los causantes de inducir estos tipos de cáncer.

Ante este devastador panorama, es evidente la necesidad de innovar en el uso de tecnologías más eficientes, pero sobre todo menos invasivas y dañinas para la salud, tales como la tecnología NFC planteada en este proyecto. Ya que se ha demostrado a través de varios experimentos en cadáveres humanos que el uso de etiquetas NFC, por sus características técnicas de operación (en la banda de los 13,56 MHz) tienen una mínima interacción con los tejidos humanos y animales (Freudenthal et al., 2007). Además el uso de esta tecnología también posee ventajas en cuanto al costo de implementación en sistemas de lectura/escritura, pues se prevé que ésta será incorporada en las nuevas generaciones de teléfonos celulares.

1.5 OBJETIVOS

El objetivo principal de este proyecto es diseñar e implementar un sistema para la identificación de animales domésticos mediante la lectura y escritura de etiquetas NFC a través de dispositivos portátiles. Para alcanzar este objetivo general, se deben lograr los siguientes objetivos específicos:

- Analizar y comparar las diferentes etiquetas NFC, su funcionamiento, características y ventajas a partir de los requerimientos para la implementación de este proyecto en específico, evaluando las diferente alternativas con el fin de implementar la solución más conveniente, tanto económica como tecnológicamente.

- Realizar la implementación de un sistema para la identificación de mascotas, para que en caso de extravío, ésta pueda ser fácilmente devuelta a su propietario a través de la relación de los datos de la mascota con los datos de localización de su propietario.

- Realizar pruebas comparativas de lectura y escritura de las etiquetas NFC para asegurar el correcto funcionamiento del sistema implementado y elaborar la documentación correspondiente a la implementación del sistema.

1.6 METODOLOGÍA

Para la realización de este proyecto se han identificado varias etapas de desarrollo, entre las cuales, la fase inicial consiste en la búsqueda de información acerca de los diferentes tipos de etiquetas NFC (NFC Tags) disponibles comercialmente y sus características técnicas. Para el desarrollo de este proyecto se debe seleccionar un tipo de etiqueta adecuada a la cantidad de información a almacenar, que posea algún mecanismo de seguridad capaz de controlar su reescritura y además posea una presentación adecuada para su adaptación al collar de la mascota (tipo llavero o moneda), y que de ser posible sea leída por la mayoría de dispositivos portátiles con tecnología NFC disponibles actualmente en el mercado.

A partir de la identificación del tipo de etiqueta a utilizar, se profundiza en sus características técnicas realizando para ello una investigación acerca de su estructura de registros y su funcionamiento interno. Al mismo tiempo se examinarán los programas y el software necesario para la elaboración del sistema de lectura y escritura de las etiquetas NFC para luego proceder a programar y elaborar una solución integrada que cubra todos los requerimientos expuestos en este proyecto.

Posteriormente se realizarán pruebas de lectura y escritura de las etiquetas a través de dispositivos apropiados con el hardware NFC requerido. A través de estas pruebas se corregirán errores y fallas en el funcionamiento esperado de la aplicación. Posteriormente a esta fase se implementará el sistema de control de escritura de las etiquetas NFC mediante

el establecimiento de una clave de seguridad, ya sea de forma explícita o implícita dentro de la aplicación desarrollada.

1.7 ESTRUCTURA DEL DOCUMENTO

En el Capítulo 1 se expuso la problemática que motivó el desarrollo de este trabajo, junto al estado actual de las tecnologías empleadas para la identificación de mascotas, se hizo un análisis de sus ventajas y desventajas. También se describieron los objetivos que se propone cumplir el presente trabajo.

En el Capítulo 2 se muestran los conceptos básicos sobre la tecnología NFC y los aspectos relativos al funcionamiento de esta tecnología, su seguridad y sus posibles campos de aplicación. Esto proporcionará al lector un resumen general de los conceptos necesarios para comprender el desarrollo de este proyecto.

En el Capítulo 3 se definen varios criterios de selección para el diseño de la aplicación, así como el software utilizado y los conceptos básicos para comprender la programación y el funcionamiento de la aplicación. También se hace una descripción detallada de los componentes que integran una aplicación en el sistema operativo seleccionado.

Finalmente, en el Capítulo 4 se describe el proceso de diseño de esta aplicación en particular, realizando una descripción detallada de los comandos específicos para la interacción de la aplicación con el hardware NFC del dispositivo, así como la programación del código y la creación de las pantallas de la aplicación. Finalmente se exponen las conclusiones de este trabajo, junto con propuestas de posibles características que se podrían añadir a la aplicación, que la hagan aún más interesante y funcional para el público en general.

Capítulo 2. Marco teórico

2.1 DESARROLLO DE LA TECNOLOGÍA NFC

NFC es una tecnología inalámbrica que permite el intercambio de datos entre dispositivos con elementos físicos pasivos y otros dispositivos móviles activos interactuando en la banda de los 13.56 MHz. Esta tecnología se basa en la identificación por radiofrecuencia (RFID[6]). NFC es un conjunto de tecnologías con las cuales los dispositivos y tarjetas "inteligentes" pueden interactuar sin tener contacto físico entre sí, lo que permite "leer" los datos almacenados a una distancia limitada a un máximo de 10 cm (Caballero Nadales, 2012).

Figura 1. Logo de la tecnología NFC

La tecnología NFC comenzó a desarrollarse en el año 2002 en una acción conjunta de Philips y Sony, con el fin de conseguir un protocolo compatible con las tecnologías propietarias existentes en el mercado: MIFARE de Philips y FeliCa de Sony. Finalmente, NFC fue aprobado como el estándar ISO 18092 en diciembre de 2003 y posteriormente, en marzo de 2004, Philips, Sony y Nokia formaron el NFC Forum para avanzar en el desarrollo de las especificaciones NFC (Madlmayr, Kantner, & Grechenig, 2014).En 2006 salió al mercado el primer dispositivo con NFC, el Nokia 6131, y en 2010 Samsung lanzó el primer dispositivo con Android y NFC, el Samsung Nexus S.

Esta tecnología cubre estándares que cubren protocolos de comunicación y formatos de intercambio de datos. ISO 14443 es un estándar internacional relacionado con las tarjetas de identificación electrónicas, en especial las tarjetas de proximidad, gestionado conjuntamente por la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica Internacional (IEC)[7].En ISO/IEC 18092 y ECMA-340se especifican los esquemas de modulación, codificación, velocidades de transferencia y el formato de la trama, así como los esquemas de inicialización y condiciones requeridas para el control de colisión de datos durante la inicialización para ambos modos de comunicación, activo y pasivo. También se define el protocolo de transporte, y se incluyen los protocolos de activación e intercambio de datos.

[6] https://es.wikipedia.org/wiki/RFID
[7] https://es.wikipedia.org/wiki/ISO_14443

2.2 FUNCIONAMIENTO

La tecnología NFC soporta tres modos de operación (Cavoukian, 2011):

- Lectura/Escritura (Read/Write): Este modo permite a los dispositivos móviles NFC leer los datos almacenados en etiquetas RFID pasivas. El dispositivo móvil genera una señal de radio de corto alcance, que el dispositivo pasivo utiliza para alimentar su circuito de alimentación interno e iniciarla transmisión de información a la velocidad establecida por el dispositivo iniciador. Este modo permite actuar sobre los datos contenidos en forma de URL (Identificador de recursos uniforme), con instrucciones de acceso hacia una dirección Web. También puede contener instrucciones para hacer una llamada o enviar un mensaje de texto (SMS). Este modo NFC además permite a los dispositivos móviles la escritura de datos en algunos tipos de etiquetas.

- Emulación de tarjeta (Card Emulation): Este modo permite a los propietarios de dispositivos móviles realizar transacciones comerciales sin realizar ningún tipo de contacto, de la misma manera que las tarjetas inteligentes se utilizan hoy en día. Este modo de operación permite que los dispositivos móviles se utilicen para aplicaciones de identificación, de pago y como control de acceso.

- Intercambio punto a punto (Peer to peer): Este modo permite a los dispositivos móviles interactuar más fácilmente con otros teléfonos al compartir datos entre sí (Cada teléfono tiene que estar equipado con NFC y las aplicaciones necesarias para establecer el intercambio), ya sea para el intercambio de tarjetas de presentación, fotos, documentos u otro tipo de información personal a través de la transferencia de datos de manera bidireccional.

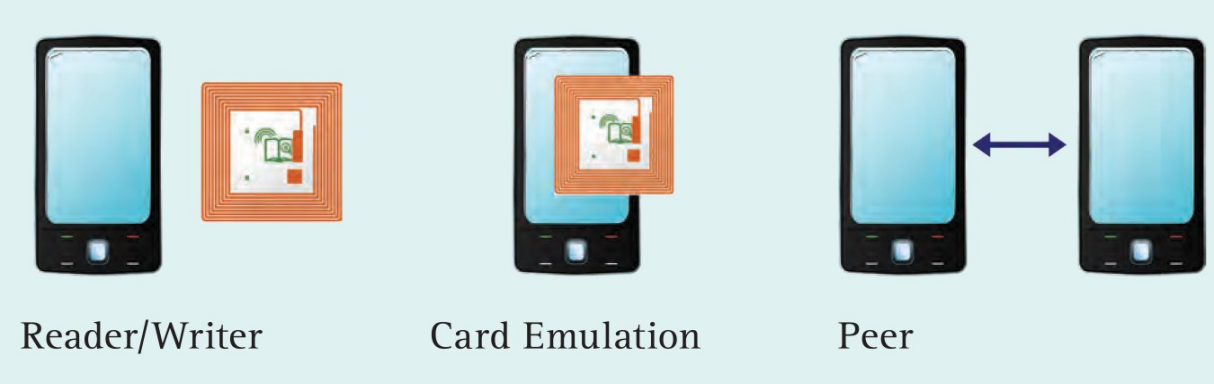


Figura 2. Modos de operación de la tecnología NFC(Cavoukian, 2011)

2.3 TRANSACCIÓN NFC

La comunicación NFC consta de cinco fases o etapas (Campa Ruiz, 2011):

- Descubrimiento: En esta fase los dispositivos inician la etapa de rastrearse el uno al otro y posteriormente su reconocimiento.

- Autenticación: En esta parte los dispositivos verifican si el otro dispositivo está autorizado o si deben establecer algún tipo de cifrado para la comunicación.

- Negociación: En esta parte del establecimiento, los dispositivos definen parámetros como la velocidad de transmisión, la identificación delos dispositivos, el tipo de aplicación, su tamaño, y si es el caso también definen la acción a ser solicitada.

- Transferencia: Una vez negociados los parámetros para la comunicación, se puede decir que se establecieron exitosamente los protocolos de comunicación entre los dispositivos y ya se puede realizar el intercambio de datos.

- Confirmación: El dispositivo receptor confirma el establecimiento de la comunicación y la transferencia de datos.

La tecnología NFC no está destinada para la transferencia masiva de datos, pero se puede utilizar para la configuración de otras tecnologías inalámbricas de mayor ancho de banda como Bluetooth o Wi-Fi con la ventaja de que al utilizar NFC el tiempo de establecimiento de la comunicación es muy inferior que si se utilizaran estas tecnologías por sí solas para efectuar el enlace.

2.4 ETIQUETAS NFC

Una etiqueta NFC está compuesta por una antena, un transductor de radio y un material encapsulado o chip. Las hay de diferentes tipos, y se pueden clasificar en pasivas y activas, en etiquetas de solo lectura, o etiquetas de lectura y escritura (Campa Ruiz, 2011).

Para las etiquetas activas, la fuente de alimentación es propia mediante baterías de larga duración. Por otra parte las etiquetas pasivas no poseen alimentación eléctrica, la señal que les llega de los lectores les induce una pequeña corriente eléctrica suficiente para que comience a operar el circuito integrado CMOS dentro de la etiqueta, de forma que éste pueda generar y transmitir una respuesta.

Las etiquetas cuya funcionalidad es de “sólo lectura” (Read Only) o “escritura única” WORM (Write Once, Read Many) son numeradas previamente y necesitan de un banco de datos que reciba su información, una vez programada, la etiqueta no puede ser modificada. Las etiquetas de lectura-escritura (Read-Write) tienen una capacidad de memoria más grande y se pueden modificar o actualizar siempre que se necesite. Las velocidades de transmisión que soporta esta tecnología son de 106, 212, 424 y 848 Kbits/s.

Figura 3. Múltiples formas de presentación de las etiquetas RFID(Campa Ruiz, 2011)

2.5 TIPOS DE ETIQUETAS NFC

El NFC Forum ha creado cuatro formatos de etiquetas para todos los dispositivos acordes con el NFC Forum. Estos formatos están basados en el ISO 14443 tipo A y B (Estándares internacionales para tarjetas inteligentes sin contacto) y FeliCa (derivado del ISO 18092):

- Etiqueta Tipo 1: Se basa en la norma ISO14443A. Estas etiquetas permiten ser leídas y reescritas, aunque los usuarios pueden configurarlas sólo para lectura. La memoria disponible es de 96 bytes, expandible a 2 kilobytes; la velocidad de comunicación es de 106 Kbps.

- Etiqueta Tipo 2: Se basa en la norma ISO14443A. Permiten ser leídas y reescritas, aunque los usuarios pueden configurarlas sólo para lectura. La memoria disponible es de 48 bytes, expandible a 2 kilobytes; la velocidad de comunicación es de 106 Kbps.

- Etiqueta Tipo 3: Estas etiquetas se basan en el Japanese Industrial Standard (JIS) X 6319-4, también conocido como FeliCa. Se encuentran ya configuradas desde su fabricación, ya sea para escritura, o sólo para lectura. La memoria disponible es variable, pero teóricamente el límite de memoria es de 1 megabyte por intercambio; la velocidad de comunicación es de 212 Kbps o 424 Kbps.

- Etiqueta Tipo 4: Este tipo de etiquetas es plenamente compatible con ISO14443 tipo A y B. Se encuentran ya configuradas desde su fabricación, ya sea para su lectura y escritura o sólo para lectura. La memoria disponible es variable, pero el máximo es de 32 kilobytes por intercambio; la velocidad de comunicación es de hasta 424 Kbps.

Las etiquetas NFC tienen un número de identificación único (con un tamaño de 4 a 10 bytes), que se da por la combinación entre el código de la empresa fabricante y el tipo de tecnología de la etiqueta (MIFARE, FeliCa, etc.).

2.6 DEFINICIÓN ESTÁNDAR DE DATOS NFC

Para poder compartir datos entre sí y/o entre los dispositivos y etiquetas NFC, se ha creado un estándar en el que se especifica un formato común para el intercambio de datos. (Campa Ruiz, 2011). A este formato se le conoce como NDEF (NFC Data Exchange Format). Al adherirse a este formato de intercambio de datos, dispositivos con distinto hardware o diferente sistema operativo son capaces de compartir datos a través de un lenguaje común, de manera mutuamente comprensible y organizada.

El formato de intercambio de datos NFC (NDEF) está compuesto por mensajes y registros NDEF. Un mensaje NDEF puede estar contenido en uno o más registros NDEF. El primer registro de un mensaje NDEF determina cómo interpretar la carga útil del mensaje. Un registro NDEF contiene los siguientes campos:

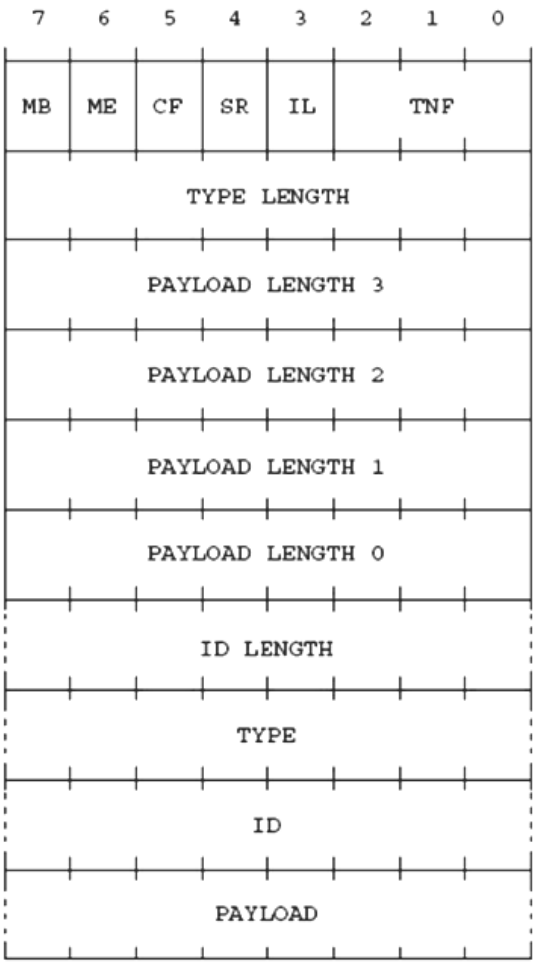


Figura 4. Estructura de un registro NDEF (Coskun, Ok, & Ozdenizci, 2013)

- MB (Message Begin): Es un bit que indica el comienzo del mensaje NDEF.

- ME (Message End): Es un bit que indica si es el último registro del mensaje NDEF.

- CF (Chunk Flag): Es un bit que indica si este registro es el primer fragmento o el fragmento de en medio en un mensaje con carga útil fragmentada.

- SR (Short Record): Indica si el campo PAYLOAD LENGTH está compuesto por un solo byte (8 bits) o menos. Esto permite registros más compactos.

- IL: Es un bit que indica que el campo ID LENGTH está presente en el encabezado del registro. Si este es cero, el campo ID LENGTH es omitido en el registro.

- TNF (Type Name Format): Es un campo de 3 bits que describe el tipo de información contenida en el registro.

- TYPE LENGTH: Este campo indica la longitud en bytes del campo TYPE. Este valor siempre es cero para registros con campo TNF definido.

- PAYLOAD LENGTH: Especifica la longitud en bytes del campo PAYLOAD. El tamaño de este campo está determinado por el bit SR. Si el valor de SR se establece como cero, este valor ocupará 4 bytes (32 bits).

- ID LENGTH: Especifica la longitud en bytes del campo ID. Este campo solo está presente si el bit IL es uno.

- TYPE: Este valor describe el tipo de carga útil del registro. Los valores de este campo deben corresponder con el valor introducido en los bits TNF del encabezado del registro.

- ID: Es el valor de identificación único del registro. Si el valor de IL se establece en cero, este campo es omitido.

- PAYLOAD: Este campo lleva la carga útil del registro NDEF, y tendrá exactamente la longitud descrita en el campo PAYLOAD LENGTH

Dentro de este formato también se especifican tres tipos de registros estándar RTD (NFC Record Type Definition) que pueden ser intercambiados entre los dispositivos NFC:

- RTD_SMART_POSTER (“Sp”): Para posters que incorporan etiquetas con datos, tales como URLs, SMSs o números de teléfono.

- RTD_TEXT (“T”): Para registros que solo contienen texto.

- RTD_URI (“U”): Una URI (Uniform Resource Identifier) es una cadena única que representa un recurso de Internet.

Valor TNF	Tipo de registro
0x00 TNF_EMPTY	Vacío. No contiene ninguna carga útil
0x01 TNF_WELL_KNOWN	Bien definido. Significa que el mensaje NDEF tiene alguno de los siguientes tipos: RTD_TEXT (Texto plano con ISO del idioma correspondiente), RTD_URI (Contiene una URI), RTD_SMART_POSTER (Contiene varios registros como URI, acciones, etc.).
0x02 TNF_MIME_MEDIA	Registro del tipo MIME. Un tipo de medio de Internet tal como se define en el RFC 2046.
0x03 TNF_ABSOLUTE_URI	URI absoluto. Un URI tal como se define en el RFC 3986.
0x04 TNF_EXTERNAL_TYPE	Registro de tipo externo. Un valor definido por el usuario, basado en reglas específicas.
0x05 TNF_UNKNOWN	Desconocido. Indica que el tipo de carga útil es desconocido. El campo Type Length debe ser 0.
0x06 TNF_UNCHANGED	Sin cambios. Sólo para registros con carga útil fragmentada. El campo Type Length debe ser 0.
0x07 TNF_RESERVED	Reservado. Reservado por el NFC Forum para usos futuros.

Tabla 1. Tipos de registro NDEF

2.7 SEGURIDAD EN NFC

La tecnología NFC por sí sola no asegura comunicaciones seguras, no ofrece protección contra escucha y es vulnerable a modificación de datos. Por esta razón es muy recomendable que las aplicaciones usen protocolos criptográficos para establecer un canal seguro (Campa Ruiz, 2011). Sin embargo, y a pesar de estas dificultades graves en materia de seguridad, este aspecto se contrarresta con la distancia de operación necesaria para su funcionamiento, ya que las transacciones 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 sencillez y simplicidad de la interacción por aproximación del dispositivo lleva inherentemente características de confianza y seguridad. Además, al ser de corto alcance, se

evitan los errores de comunicación, se asegura una mayor eficacia en la transmisión de datos y se hace necesario que un posible atacante deba estar dentro de ese corto rango, en el cual el usuario podría darse cuenta fácilmente.

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Interfaz RF	ISO 14443 A-2	ISO 14443 A-2	FeliCa (ISO 18092, modo de comunicación pasivo a 212 kbits/sec)	ISO 14443 A-2
Inicialización	ISO 14443 A-3	ISO 14443 A-3	FeliCa (ISO 18092, modo de comunicación pasivo a 212 kbits/sec)	ISO 14443 A-3
Velocidad	106 kbits/sec	106 kbits/sec	212 kbits/sec	106-424 kbits/sec
Protocolo	Conjunto de comandos específicos	Conjunto de comandos específicos	Protocolo FeliCa	Comandos ISO 14443 A-4 ISO 7816-4
Tamaño de memoria	Mayor de 1KB	Mayor de 2 KB	Mayor de 1 MB	Mayor de 64 KB
Coste (dependiendo de la memoria)	Bajo	Bajo	Moderado	Moderado
Casos de uso	Etiquetas con tamaño de memoria pequeño para aplicaciones sencillas		Etiquetas flexibles con tamaño de memoria grande para aplicaciones con múltiples capacidades	

Figura 5. Relación de los distintos tipos de etiquetas y sus características(Campa Ruiz, 2011)

2.8 TIPOS DE APLICACIONES

Actualmente, los usos de NFC están ligados a los teléfonos móviles, debido al hecho de que es el único dispositivo que necesitamos llevar a todas partes. Los estudios llevados a cabo por el NFC Forum señalan que en un periodo de tres a cinco años, la tercera parte de los teléfonos móviles a nivel mundial estarán equipados con tecnología NFC(Campa Ruiz, 2011).Formalmente, el NFC Forum ha especificado tres aplicaciones centrales:

- Conexión NFC: Permite la transferencia de datos y archivos entre dispositivos.
- Acceso NFC: Permite acceder a información "on-the-move" (en movimiento) con sólo acercar el dispositivo.
- Transacciones NFC: Permite el pago a través de dispositivos móviles y tickets.

Capítulo 3. Marco de referencia

3.1 DISPOSITIVOS PORTÁTILES

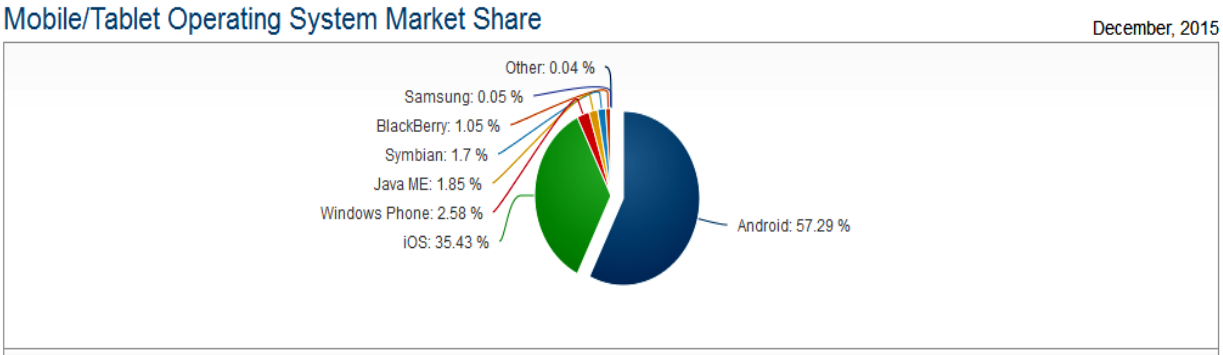
En este punto es necesario definir las pautas sobre las cuales se desarrollará este proyecto. Dado que se estipuló que el sistema se desarrollará para dispositivos portátiles, es necesario definir con más detalle a que hacemos referencia. En primer lugar, el término portátil "hace referencia a aquello que resulta sencillo de trasladar y que, por lo tanto, se lo puede calificar como móvil". En este sentido se ha proyectado el desarrollo de este proyecto especialmente para teléfonos móviles, debido a la utilización generalizada de estos, no solo para la comunicación con otras personas, sino también como una herramienta poderosa que incluye múltiples sensores, que aumentan sus funcionalidades. Si bien, existen otro tipo de dispositivos que se han denominado como "portátiles", tales como los ordenadores portátiles (notebooks), tablets, entre otros; es imprescindible que estos dispositivos tengan incluido de fábrica el hardware NFC, ya que es indispensable para el funcionamiento del sistema propuesto.

En cuanto a las tablets, estas funcionan con un sistema operativo muy similar (en muchos casos igual) al de los teléfonos móviles, y algunos fabricantes han incluido el hardware NFC en sus tabletas. Por lo tanto, estas también están incluidas en los "dispositivos portátiles" a los que hace referencia este proyecto.

Actualmente, no existe en el mercado ningún computador portátil que disponga de hardware NFC implementado de fábrica, ya que es mucho más práctico conectar a través de un puerto USB el hardware NFC como un periférico. Aunque la funcionalidad de este conjunto es similar ala de un teléfono móvil con NFC, el funcionamiento de un computador portátil es mucho más complejo que el de un teléfono móvil, por consiguiente, los programas y los sistemas operativos utilizados en cada dispositivo difieren bastante, tanto en su funcionamiento, como en la programación de sus aplicaciones (o programas, en el caso de un computador).

3.2 SELECCIÓN DEL SISTEMA OPERATIVO

Como se dijo anteriormente, el proyecto está enfocado a dispositivos portátiles en los cuales esté incluido de fábrica el hardware NFC necesario para el funcionamiento del sistema propuesto. Los dispositivos, a los cuales hacemos alusión son algunos teléfonos móviles y tablets. La gran mayoría de estos dispositivos utilizan el sistema operativo Android, según lo muestra el siguiente gráfico comparativo sobre la participación en el mercado para los diferentes sistemas operativos móviles, actualizado hasta diciembre de 2015:



OPERATING SYSTEM	TOTAL MARKET SHARE
Android	57.29%
iOS	35.43%
Windows Phone	2.58%
Java ME	1.85%
Symbian	1.70%
BlackBerry	1.05%
Samsung	0.05%
Kindle	0.02%
Bada	0.01%
Windows Mobile	0.00%
LG	0.00%

Figura 6. Distribución de mercado para los distintos sistemas operativos móviles a diciembre de 2015[8]

En este gráfico podemos apreciar que hay dos grandes sistemas operativos, Android e iOS. Sin embargo, el sistema operativo iOS está diseñado para funcionar únicamente en dispositivos de la marca Apple®. Esta empresa ha desarrollado teléfonos móviles, los cuales son más conocidos en el mercado como iPhone's, también ha desarrollado tablets, denominados iPad's. Sin embargo, si nos fijamos en el listado de dispositivos móviles existentes en el mercado que integran la tecnología NFC[9], podemos encontrar, que hasta la fecha, esta tecnología no ha sido incorporada en ningún dispositivo de la marca Apple®, lo cual excluye definitivamente al sistema operativo iOS y a sus dispositivos portátiles de utilizar el desarrollo aquí expuesto.

3.3 VERSIÓN DEL SISTEMA OPERATIVO

Como definimos en el punto anterior, la aplicación se desarrollará en el entorno Android. Android es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes (Smartphones) y tablets. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró[10].

[8] http://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=M&qpct=3&qpf=1
[9] http://www.nfcworld.com/nfc-phones-list/
[10] https://es.wikipedia.org/wiki/Android

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan sobre un núcleo de bibliotecas Java en una máquina virtual. El sistema operativo Android ha visto numerosas actualizaciones desde su liberación inicial. Estas actualizaciones al sistema operativo base típicamente arreglan bugs y agregan nuevas funciones al sistema. Generalmente cada actualización del sistema operativo Android recibe, el nombre en inglés de un postre o dulce diferente. En cada versión el postre o dulce elegido empieza por una letra distinta, de acuerdo a un orden alfabético:

- A: Apple Pie (1.0): Tarta de manzana
- B: Banana Bread (1.1): Pan de plátano
- C: Cupcake (1.5): Pastelillo
- D: Donut (v1.6): Rosquilla
- E: Éclair (v2.0/v2.1): Pepito
- F: Froyo (v2.2): Helado de yogurt
- G: Gingerbread (v2.3): Pan de jengibre
- H: Honeycomb (v3.0/v3.1/v3.2): Panal de miel
- I: Ice Cream Sandwich (v4.0): Sandwich de helado
- J: JellyBean (v4.1/v4.2/v4.3): Goma de gelatina
- K: KitKat (v4.4): Galleta recubierta de chocolate *KitKat*
- L: Lollipop (v5.0/v5.1): Chupeta
- M: Marshmallow (v6.0) Malvavisco

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.7%
4.1.x	Jelly Bean	16	9.0%
4.2.x		17	12.2%
4.3		18	3.5%
4.4	KitKat	19	36.1%
5.0	Lollipop	21	16.9%
5.1		22	15.7%
6.0	Marshmallow	23	0.7%

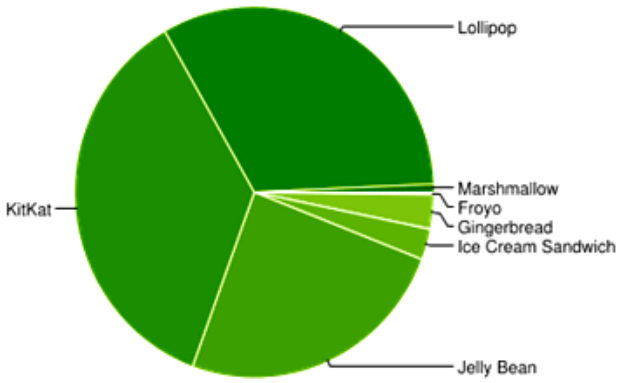


Figura 7. Distribución porcentual de dispositivos según la versión de su sistema operativo Android a comienzos de enero de 2016[11]

[11] http://developer.android.com/intl/es/about/dashboards/index.html#

Actualmente, según los datos obtenidos de la página oficial para desarrolladores de Android, existen tres versiones que acaparan el 93.4% de los teléfonos móviles que tienen el sistema operativo Android, los cuales son en su orden KitKat, Lollipop y JellyBean, las cuales corresponden a los niveles de interfaz de programación de aplicaciones(API) 16 a 22. Una API es el conjunto de subrutinas, funciones y procedimientos (métodos) que ofrece acceso a ciertos servicios y representa un método de abstracción en la programación, generalmente entre los niveles o capas inferiores y superiores del software[12].

Esta aplicación, al requerir de manera obligatoria de la tecnología NFC, no solo está limitada por el hardware del dispositivo, también está limitada por la implementación de los controladores diseñados por el fabricante en cada dispositivo. Estos controladores pueden interactuar con el sistema operativo subyacente a través de las nuevas características agregadas en cada versión del sistema operativo. A partir de la API 10 (Gingerbread), se agregan al sistema operativo los comandos necesarios para que los desarrolladores puedan crear aplicaciones que interactúen con el hardware NFC del dispositivo[13]. En la API 14, se introduce la característica Android Beam, la cual permite el envío de mensajes NDEF. En este nivel también se introducen los filtros de intención con el fin de determinar que aplicación será lanzada automáticamente cuando se detecte una etiqueta NFC[14]. Finalmente, en el nivel 19 de la API (KitKat), se implementa la emulación de tarjeta, y se establece un nuevo modo de operación que permite restringir la actividad en primer plano solamente al detectar alguna etiqueta de tipo predeterminado[15].

3.4 SOFTWARE DE DESARROLLO ANDROID STUDIO

El desarrollo de aplicaciones en Android requiere de un entorno basado en tecnologías integradas. En primera instancia necesitamos las librerías y herramientas oficiales desarrolladas por la marca, es decir, el SDK (Software Delevelopment Kit) de Android. Esta herramienta nos informa que complementos están instalados en el computador y cuáles no.

Como lenguaje de desarrollo para la construcción de aplicaciones Android se utiliza Java. Además se usa el metalenguaje XML para el diseño de las interfaces. También se usa un sistema de construcción llamado Gradle, para automatizar la compilación, depuración y despliegue de las aplicaciones. Esta herramienta emplea el compilador de Java (javac) en una máquina virtual Java (JDK) para programar de manera modular la aplicación mediante "Scripting".

Para la edición y construcción, Google ha creado un software que reúne todo el conjunto de herramientas requeridas por el programador en un IDE (Ambiente de desarrollo integrado) llamado Android Studio. Un IDE es una aplicación informática que proporciona servicios integrales para facilitar el desarrollo de software. Los IDE están diseñados para reducir la

[12] https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones
[13] http://developer.android.com/intl/es/about/versions/android-2.3.3.html
[14] http://developer.android.com/intl/es/about/versions/android-4.0.html
[15] http://developer.android.com/intl/es/about/versions/android-4.4.html

configuración necesaria para reconstruir múltiples utilidades de desarrollo y para maximizar la productividad del programador, proporcionando componentes unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo, generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software[16].

3.5 CONCEPTOS BÁSICOS DE JAVA

La estructura del sistema operativo Android está compuesta de aplicaciones que se ejecutan sobre un núcleo de bibliotecas Java. Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. A continuación vamos a describir algunos de los componentes que son fundamentales para entender este lenguaje de programación:

3.5.1 Clases

Una clase es la representación abstracta de las características de un objeto en particular. Dentro de ésta se definen las características (atributos) que posee y los comportamientos que tendrá (métodos).Para declarar una clase se usa la palabra reservada class. Los modificadores de acceso determinan el alcance de sus atributos y métodos hacia otras clases. Existen 4 modificadores, los cuales son:

- public: Permite que una pieza de código sea visible en todos los niveles.
- protected: Con este modificador solo las clases que se encuentren en el mismo paquete pueden ver y acceder a este código.
- private: Solo la clase donde está contenido el código, puede tener acceso.
- sin modificador: Permite que solo las clases del paquete accedan al código. A diferencia de protected, este modificador no permite que las subclases tengan privilegios del código de su superclase.

A través de la palabra reservada extends se declara que una clase hereda de otra. La herencia es el proceso de transmitir atributos y métodos de una clase a otra. Donde la clase que se toma como base se le denomina superclase y la que hereda se le llama subclase. La referencia superes utilizada para representar a la superclase.

El modificador final declara que un fragmento de código no puede cambiar. Este modificador se puede usar en clases, atributos, métodos, parámetros y variables locales. Declarar una clase con este modificador asegura que esta no tenga subclases. Al hacerlo sobre un método, significa que las subclases de una superclase no pueden sobrescribir este método. Usar final con un atributo obliga a que se asigne el valor en la declaración o en todos sus constructores.

[16] https://es.wikipedia.org/wiki/Ambiente_de_desarrollo_integrado

A través del modificador static es posible declarar atributos y métodos como parte de la clase, en vez de ser parte de las instancias. Cuando un atributo es marcado con static dentro de una clase significa que será común para todos los objetos y tendrá una posición fija de memoria. Cualquier instancia de una clase puede cambiar el valor de un atributo estático, sin embargo también es posible hacerlo sin tener que crear una instancia. Para declarar una constante en Java se usa la combinación de los modificadores final y static. Esta constante podrá usarse en todo el proyecto.

En Android podemos encontrar varios tipos de clases. Estas se utilizan con diversos propósitos para ciertas funciones específicas[17]:

- Clases genéricas: Los Generics permiten crear implementaciones de clases, métodos e interfaces basados en parámetros. Al crear, por ejemplo una lista de un tipo específico, dicha lista debe estar contenida en cuñas "<>". A este "contenedor" se le denomina parámetro. Al usar un parámetro, podemos referirnos a un tipo genérico que tendrá elementos con cosas en común, así al momento de acceder a estos elementos se producirá el mismo resultado sin importar cuál de estos se use.

- Clases anónimas: Una clase anónima es un tipo especial de clase interna que se declara y se instancia al mismo tiempo. Esta puede ser declarada dentro de los bloques de código como si fuera un método. Su importancia está en que, en vez de ser tratadas como una declaración, se consideran como una expresión. Esto significa que es posible asignarlas a otra expresión existente dentro de un bloque de código. El ejemplo más común se da al momento de usar "escuchas" para manejar los eventos en algún botón de la interfaz de Android a través del método setOnClickListener(). Este método recibe una instancia del tipo OnClickListener. Justo en ese parámetro se puede crear una clase anónima que satisfaga este requerimiento con el operador new. Al hacer esto, estamos definiendo el cuerpo de una nueva clase sin nombre (clase anónima).

- Clases abstractas: Una clase abstracta es aquella de la cual no se pueden crear instancias directamente. Sin embargo, se pueden crear subclases a partir de ellas. Definir una clase de este tipo requiere usar la sentencia abstract en la declaración. Este concepto permite organizar las clases semejantes y obliga a los programadores a sobrescribir métodos importantes que garanticen el correcto funcionamiento de un patrón. Dentro de una clase abstracta también pueden existir métodos abstractos. Un método abstracto es aquel que se declara pero no se implementa, con el fin de que las subclases si lo hagan.

- Clases anidadas: El lenguaje Java permite que los desarrolladores declaren clases dentro de otras clases. Ellas reciben el nombre de clases anidadas, donde a la clase contenedora se le denomina clase externa. Una clase anidada puede ser declarada como estática o como no estática, y se comporta como un miembro más de la clase externa. La primera es llamada clase anidada estática y la segunda clase interna. Las

[17] http://www.hermosaprogramacion.com/2015/08/introduccion-a-java-para-desarrollo-android/

clases internas pueden acceder a otros miembros de la clase externa, incluso si estos son privados. En cambio una clase anidada estática no tiene esta oportunidad.

3.5.2 Excepciones

Una excepción es un evento inusual que ocurre en la ejecución de un programa y que interrumpe el flujo de instrucciones. Cuando esto sucede, se crea un objeto del tipo Exception que contiene los datos del error. Las excepciones se basan en la clase Throwable, la cual tiene implementado el comportamiento para interrumpir el flujo normal de la aplicación en caso de que se presenten condiciones específicas. Cuando ocurren problemas graves en el proceso de la máquina virtual de Java, esta lanza objetos de tipo Error. Si el problema no es tan grave para el sistema, se lanzan objetos del tipo Exception.

Los dos tipos principales de excepciones son:

- RuntimeException: Errores del programador, como una división por cero o el acceso fuera de los límites de un arreglo. No es necesario declarar este tipo de excepciones en la cláusula throws.

- IOException: Errores que no puede evitar el programador, generalmente relacionados con la entrada/salida del programa.

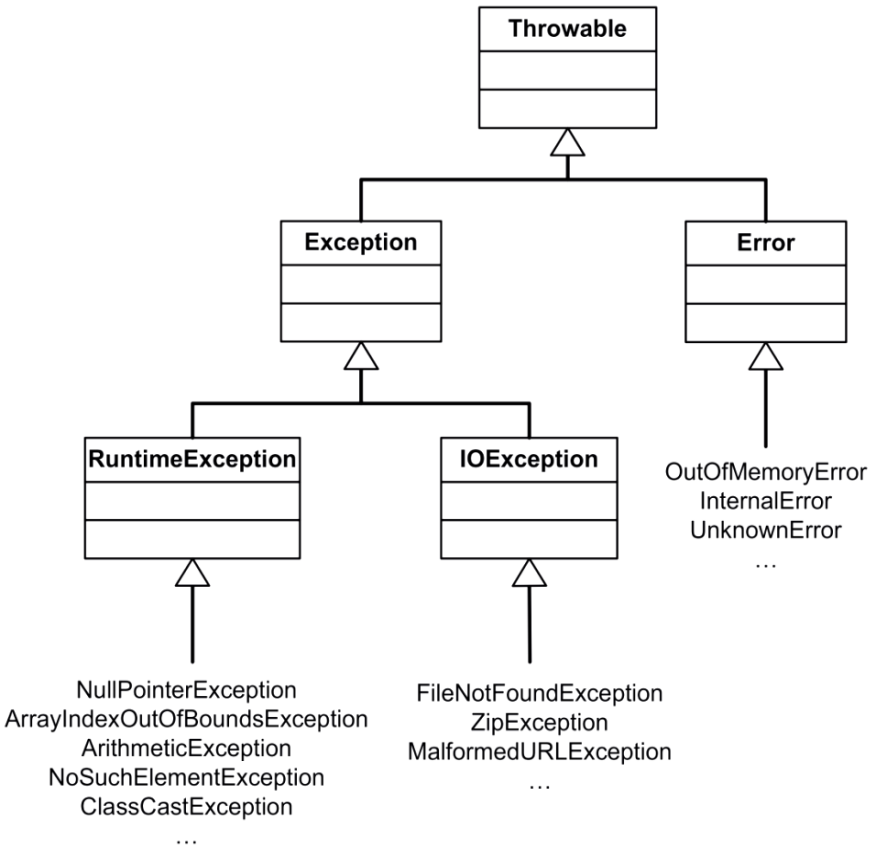


Figura 8. Jerarquía de clases para el manejo de excepciones en Java (Berzal Galiano, 2006)

Por medio del método throws se lanzan las excepciones, y para manejarlas se usa un bloque try-catch, donde try recibe las instrucciones a procesar y catch realiza las acciones en caso de que se llegue a presentar una excepción. También es posible incluir dentro de este bloque de instrucciones el comando finally, que ejecutará un fragmento de código independientemente de si se produce o no una excepción. Es usual emplear la instrucción printStackTrace() para imprimir la pila de seguimiento sobre la excepción, esto es útil para indicar las capas sucesivas en dónde ocurrió el inconveniente.

3.6 CONCEPTOS BÁSICOS DE XML

XML significa extensible markup language, que en español significa lenguaje de marcado extensible. Es un lenguaje basado en etiquetas descriptivas, cuyo fin es representar información de texto en forma jerárquica, y a su vez darle formato. XML hace posible la interoperabilidad entre dos sistemas de información, ya que propone un puente de intercambio entre dos fuentes de datos distintas.

Inicialmente XML se originó para manejar y estructurar opciones de configuración para sistemas operativos y aplicaciones. Posteriormente se escogió para intercambiar información entre distintas tecnologías, debido a que establece un solo formato para la obtención de datos. XML propone una sintaxis cómoda para las máquinas y los humanos, siendo un lenguaje amigable para representar jerarquías de datos. Esta característica lo hace muy útil para la gestión de recursos de un proyecto en Android Studio. Los archivos XML se analizan e interpretan por una herramienta llamada parser (Analizador de sintaxis). Los parsers ayudan a obtener la información de un documento XML de forma automática en un contexto. Para ello interpreta su jerarquía y aplica el formato establecido para cada dato.

3.6.1 Estructura de un documento XML

Un archivo XML está compuesto por los siguientes elementos básicos: prólogo, comentarios y elementos[18]:

- El prólogo es una declaración XML no obligatoria que especifica algunas características del documento, como por ejemplo la versión de XML que se usa o el tipo de encriptado. Esta expresión usa como inicio la palabra reservada “xml” y un signo de interrogación “?”. Luego está la palabra “versión”, que hace referencia al estándar XML utilizado y seguido a esto, la palabra “encoding”, que indica el estándar de encriptado o la forma de traducir los caracteres presentes en el documento en forma binaria.

- Los comentarios son mensajes destinados al entendimiento del código por parte del programador. Solo tiene fin informativo, así que serán ignorados por el parser a la hora del analizar el archivo XML. Para la apertura del comentario se utiliza la expresión “<!–” y para el cierre se emplea “–>”.

[18] http://www.hermosaprogramacion.com/2014/08/xml-lenguaje/

- Los elementos son la base de un documento XML. Un elemento es descrito por una etiqueta inicial envuelta en paréntesis angulares <elemento> y una etiqueta de cierre con una barra inclinada antecedida al nombre del elemento </elemento>. Al igual que en HTML. Al texto que se encuentra en el interior del elemento se le llama contenido. En casos donde el elemento no describe contenido alguno, es posible abreviar la etiqueta de la siguiente forma <elemento/>. Con esto se reduce a una sola expresión la apertura y cierre del bloque de construcción.

3.6.2 Atributos de un elemento en XML

Los atributos son propiedades que caracterizan y representan datos relacionados a un elemento que describen sus funciones informativas. Es muy recomendable usar atributos cuando la información no se repite y además no requiere de un orden lógico. Para todos los demás casos, es recomendable emplear elementos e identificar de manera secuencial cada elemento para conseguir procesar la información en orden.

3.6.3 XPath para extraer datos de un documento XML

Para extraer información de un archivo XML es necesario usar el lenguaje XPath. Este lenguaje funciona con un estilo de navegación a través de barras laterales y expresiones regulares para encontrar el elemento requerido. En Android Studio, es muy frecuente pedir el valor de los nodos <string> de la forma:"android:text="@string/name"". Esa sentencia indica que el atributo texto de las vistas se encontrará almacenado en el archivo strings.xml cuyo nombre es name.

3.7 ANDROIDMANIFEST

Un proyecto en Android Studio está conformado básicamente por un descriptor de la aplicación (AndroidManifest.xml), por los archivos de código fuente en Java y por una serie de ficheros con recursos, además por los archivos de configuración y los archivos de construcción de Gradle.

El archivo AndroidManifest es un archivo XML que contiene nodos descriptivos sobre las características de la aplicación Android. En él se indican las actividades, las intenciones, los servicios y los proveedores de contenido de la aplicación. También se declaran los permisos requeridos al instalar la aplicación, la versión mínima de Android para poder ejecutarla, etc.

Todas las aplicaciones deben contener este archivo por convención. El nombre debe permanecer intacto, ya que es usado como referencia para el parsing de la aplicación. El nodo raíz de este documento se representa con la etiqueta <manifest> y debe contener un hijo de tipo <application>. El campo manifest posee dos atributos, xmlns:android y package. El primero es el namespace del archivo, que nunca se debe cambiar, y el segundo indica el nombre del paquete Java que soporta la aplicación. El nombre del paquete debe ser único y es un diferenciador a largo plazo.

Dentro del archivo AndroidManifest, se define la versión mínima y la versión para la cual se ha diseñado la aplicación Android dentro del campo <uses-sdk>. Teniendo en cuenta las características incluidas en cada versión de Android, la distribución porcentual de estas versiones en la actualidad, y su proyección a futuro, se decidió que la API 19 será la versión mínima (minSdkVersion) que soportará la aplicación desarrollada, y la versión empleada para su desarrollo (targetSdkVersion), la API 23, que corresponde al último nivel de la API en la actualidad.

A través del campo<uses-permission> se definen los permisos necesarios para que la aplicación tenga acceso al hardware NFC y se solicita el permiso para acceder a las funciones telefónicas del dispositivo con el fin de obtener un identificador con el que posteriormente se pueda establecer algún tipo de validación entre el dispositivo y cada una de las etiquetas que se intentarán escribir.

3.8 COMPONENTES DE UNA APLICACIÓN ANDROID

Una aplicación en Android está compuesta por varios componentes que permiten estructurar de manera lógica el funcionamiento integral de la aplicación. Estos componentes se interrelacionan de manera conjunta para obtener el correcto funcionamiento de la aplicación. A continuación, describiremos de manera breve cada uno de los componentes:

3.8.1 Actividades

Una actividad es la representación visual e interactiva de una aplicación. Esta se puede interpretar como una máquina de estados que está permanentemente pendiente de las acciones del usuario. Aunque el programador no puede controlar la forma en que se iniciará, si puede decidir qué acciones se ejecutarán en cada estado.

Las actividades son representadas por la clase Activity, y en ella se encuentran todos los métodos que representan su ciclo de vida. Cada transición entre estados representa un método de retrollamada (callback) sobre la actividad. La siguiente ilustración representa las respectivas transiciones entre estados a través de estos métodos(Revelo Urrea, n.d.):

- Creación: Cuando el usuario presiona sobre el icono de la aplicación en su dispositivo, se dispara el método onCreate() es ejecutado inmediatamente para cargar el layout de la actividad principal en memoria.

- Ejecución-Reanudación: Después de haber sido cargada la actividad se hace la transición hacia su ejecución de manera secuencial a través de los métodos onStart() y onResume(). Aunque onStart() hace visible la actividad, es onResume() quien le transfiere el foco para que interactúe con el usuario.

- Pausa: Una actividad está en pausa cuando se encuentra parcialmente visible en la pantalla. Un ejemplo sería cuando se abren diálogos que toman el foco

superponiéndose a la actividad. El método llamado para la transición hacia la pausa es onPause().

- Detención: Una actividad está detenida cuando no es visible en la pantalla, pero aún se encuentra en memoria y en cualquier momento puede ser reanudada. Cuando una aplicación es enviada a segundo plano se ejecuta el método onStop(). Al reanudar la actividad, se pasa por el método onRestart() hasta llegar al estado de ejecución y luego al de reanudación.

- Destrucción: Cuando la actividad ya no existe en memoria se encuentra en estado de destrucción. Antes de pasar a destruir la aplicación se ejecuta el método onDestroy(). Es común que la mayoría de actividades no implementen este método, a menos que deban destruir procesos en segundo plano (como servicios).

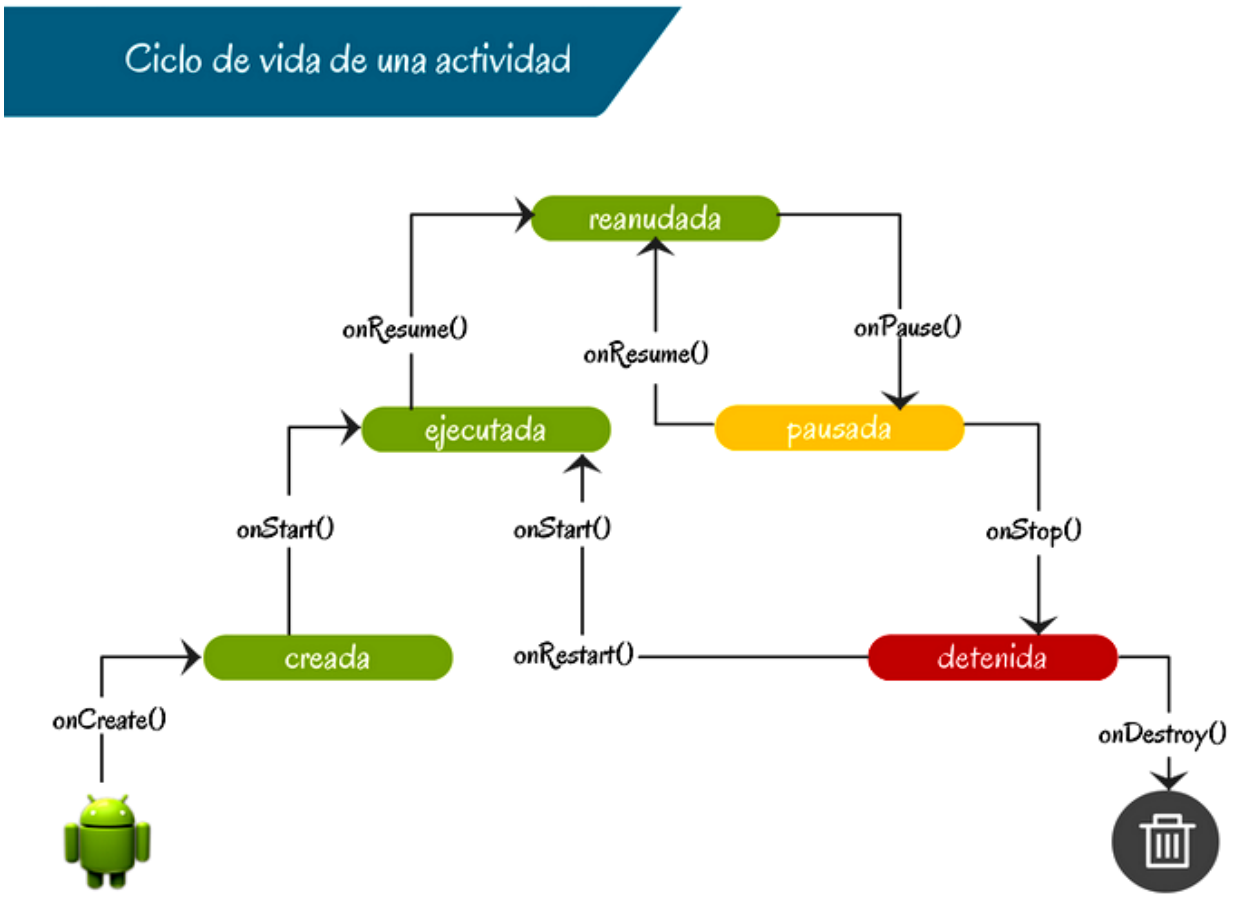


Figura 9. Ciclo de vida de una actividad en Android (Revelo Urrea, n.d.)

Aunque la aplicación puede tener varias actividades en su estructura, es necesario definir una actividad principal. Para hacerlo se debe especificar en el AndroidManifest un componente <activity>, con un componente hijo <intent-filter>. Dentro de este componente se deben declarar dos nuevos componentes, <action> y <category>. El primero tendrá asignado el elemento enumerado MAIN y el segundo, el elemento LAUNCHER.

3.8.2 Intents

Un Intent es el elemento básico de comunicación entre los distintos componentes Android. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones para comunicarse entre sí y compartir datos. Estos se pueden asemejar a los links entre páginas web.

Los Intents pueden ser implícitos o explícitos. Un ejemplo de Intent implícito se presenta cuando se desea compartir un sitio web en alguna red social. En ese momento, Android ofrece una lista para elegir que aplicación abrir entre todas las aplicaciones sociales disponibles que están instaladas en el teléfono, en este caso no se especificó una aplicación, simplemente se mostraron todas las aplicaciones que podrían responder a este tipo de acción. Contrario a esto, un Intent explicito se da cuando se invoca una actividad con un tipo específico de clase para una acción determinada.

3.8.3 Servicios

Un servicio es una entidad que ejecuta instrucciones en segundo plano sin que el usuario lo note en la interfaz. Son muy utilizados para realizar acciones de larga duración mientras las actividades muestran otro tipo de información. Por ejemplo, guardar información en una base de datos, escuchar música mientras se ejecuta la aplicación, administrar conexiones de red, etc. Los Servicios extienden de la clase base Service y no son afectados por las transiciones en el ciclo de vida de una actividad.

Debido a que las actividades tienden por defecto a ejecutarse en un mismo hilo llamado UI thread, esto hace que una instrucción no se pueda ejecutar hasta que se termine el proceso actual. Es por esta razón que se requieren los servicios, para permitir ejecutar instrucciones de larga duración en otro hilo simultáneo y así no alterar las respuestas de la interfaz.

3.8.4 Content Providers

Un Content Provider es una interfaz que permite intercambiar información persistente y estructurada entre dos aplicaciones. Cada aplicación tiene protegida su información, de modo que están aisladas de los contextos de otras aplicaciones. Aunque los Intents permiten intercambiar pequeños datos entre aplicaciones, un Content Provider extiende esta funcionalidad para que un determinado grupo de datos esté disponible para distintas aplicaciones.

Los Content Providers extienden la clase ContentProvider para implementar los métodos de la interfaz, las aplicaciones no acceden a este interfaz directamente, sino que lo hacen a través de la clase ContentResolver. Este sistema permite acceder a datos almacenados en el sistema de ficheros, bases de datos SQLite o cualquier otra fuente de datos a través de un sistema unificado.

Los Content Providers son excelentes para trabajar en segundo plano cargando información. Esta ventaja permite proporcionar una excelente experiencia al usuario debido

a que no habrá interferencia entre las consultas a la base de datos y las respuestas de la interfaz.

3.8.5 Broadcast Receivers

Los Broadcast Receivers son disparadores (triggers) implementados para activarse cuando un Intent se envía entre componentes. Dichos eventos se relacionan con el cambio de características en un componente o el cambio de estado. Los Broadcast Receivers extienden la clase BroadcastReceiver, y no tienen interfaz de usuario, sin embargo son capaces de lanzar una actividad en respuesta a un evento o usar una notificación para alertar al usuario de alguna acción o tarea.

Los Broadcasts Receivers son muy útiles para monitorear los estados de la aplicación, lo que permite ejecutar instrucciones si en algún momento sucede alguna acción prevista. Aunque un Broadcast Receiver no se carga en memoria inicialmente, siempre está a la espera de algún evento para ejecutarse. Por ejemplo, es común hacer seguimiento al uso de la batería con Broadcasts Receivers para optimizar su rendimiento, o para monitorear cuando llega un correo nuevo en la bandeja de entrada.

3.9 LOS LAYOUTS

Un layout es el contenedor principal que define las dimensiones, el orden y la secuencia en que se organizarán los elementos que componen la interfaz de usuario en las diferentes pantallas de la aplicación. La construcción de los layouts se hace a través de nodos XML. Dentro de estos elementos se pueden incluir y combinar diversas estructuras de diseño, para así personalizar la apariencia de cada pantalla.

Declarar los elementos de la interfaz de usuario en archivos XML es útil para crear todo el aspecto visual que se usará en la aplicación de forma estática. Si los elementos deben cambiar por algún motivo cuando la aplicación está en funcionamiento, entonces se requiere la implementación de código Java para modificar dinámicamente los elementos de la interfaz. Ambos métodos no son excluyentes.

Adicionalmente Android Studio proporciona un panel de diseño estilo “Drag and Drop” (arrastrar y soltar), que facilita la creación de las interfaces de usuario. Con este diseñador se ubican los componentes sobre la pantalla seleccionada y estos automáticamente son generados como nodos en el archivo XML.

Algunos de los layout más utilizados por los diseñadores de aplicaciones son[19]:

- Linear Layout: Este tipo de layout distribuye uno tras otro todos sus elementos hijos en una sola dimensión, ya sea en una sola fila, de forma horizontal o en una sola columna, de forma vertical según se establezca en el atributo “orientation”.

[19] http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/

- Relative Layout: Permite alinear la posición de cada elemento de forma relativa a su elemento padre y con los bordes de otros elementos incluidos dentro del layout. A través de los Relative Layouts es posible formar un diseño irregular con una serie de elementos.

- Table Layout: Organiza los elementos en filas y columnas de forma tabular.

- Grid Layout: Este layout permite alinear sus elementos hijos en una cuadrícula (grilla o grid). Nace con el fin de evitar anidar linear layouts para crear diseños complejos. Su funcionamiento se basa en un sistema de índices con inicio en cero.

En ocasiones los layout creados suelen desajustarse al momento de rotar la pantalla. Esto afecta la experiencia de usuario, ya que desaprovecha el espacio de acción disponible y es posible que los elementos pierdan congruencia. Esto se soluciona creando un layout especial llamado layout-land, para mostrar la actividad cuando la pantalla se encuentre de forma horizontal.

Entre los principales elementos de diseño o views se encuentran los siguientes:

- Botones: Un botón es un elemento simple que sirve fundamentalmente para realizar una transición entre actividades. Android proporciona tres tipos de botones: Button: Es el típico botón normal, contiene un texto y puede contener un icono. ToggleButton/RadioButton: Este tipo de botón se utiliza como un switch, contiene un indicador visual, en forma de raya o un círculo concéntrico (respectivamente), que indican su estado, pulsado o no pulsado. ImageButton: Contiene una imagen, a la cual se accede como un recurso. Es muy útil cuando se quieren hacer cosas vistosas o diseñar botones con formas raras, ya que es posible usar cualquier forma poniendo el fondo transparente.

- TextView y EditText: Son componentes indispensables de casi toda aplicación. Los TextView son campos que se utilizan como medio de salida para mostrar un determinado texto al usuario, mientras que los EditText son cajas de texto que tiene el usuario para introducir datos a la aplicación.

- CheckBoxes, Spinners y Pickers: A través de un CheckBox es posible seleccionar y des-seleccionar varios elementos de un listado. Con un Spinner es posible desplegar una lista de opciones y seleccionar una sola opción de entre el listado. Esto es muy ventajoso debido a que el listado se encuentra comprimido ocupando un espacio reducido en la pantalla y solamente es desplegado al presionar sobre el Spinner. Mediante un Picker el usuario puede escoger una fecha o una hora a través de un selector que asegura una fecha u hora válidas, en el formato correcto y ajustada a la configuración regional del usuario.

- ListView: Muestra una lista de elementos seleccionables con scroll integrado (si la lista es muy grande). Esta lista se puede llenar de dos formas: A través de un

adaptador que extrae el contenido de una fuente, tal como una matriz, o a través de una base de datos volcando el objeto cursor sobre ella.

- AlertDialogs, Toast y Fragments: Un AlertDialog es una ventana sencilla con un título, un mensaje y un botón para cerrarla. Es muy útil para informar al usuario de alguna situación, obligándolo a tomar una decisión para que la ventana desaparezca. Un Toast nos permite mostrar un mensaje superpuesto por un breve período de tiempo, pero sin congelar la aplicación. Este mensaje no permite interactuar con el usuario (no se pueden ingresar datos, seleccionar botones ni obligar a que se cierre). Por otra parte, un Fragment permite extender las características de un AlertDialog para mostrar un conjunto organizado de elementos superpuestos en la pantalla original, tales como botones, listas, CheckBoxes, Pickers, etc.

Capítulo 4. Desarrollo de la aplicación

4.1 ESPECIFICACIONES TÉCNICAS DE LAS ETIQUETAS NFC

Para realizar correctamente esta aplicación, se debe tener en cuenta que además del dispositivo con tecnología NFC, las etiquetas NFC a utilizar también deben cumplir con ciertos requisitos de memoria y seguridad necesarios para el desarrollo de esta aplicación. Las etiquetas NFC deben contar con la capacidad de memoria suficiente para poder almacenar todos los datos relevantes de la mascota, tales como:

- Nombre de la mascota
- Raza
- Color y descripción del pelaje de la mascota
- Género de la mascota
- Vacunas y desparasitación
- Esterilización
- Discapacidades o enfermedades

E información relevante sobre el propietario de la mascota, como:

- Nombre del propietario
- Ciudad de residencia
- Número de teléfono
- Correo electrónico de contacto

Teniendo en cuenta la cantidad de datos a escribir, es necesario que las etiquetas NFC tengan una capacidad mínima aproximada de 290 bytes[20]. Esto excluye muchos tipos de etiqueta disponibles en el mercado. Si además tenemos en cuenta que las etiquetas deben contar con algún sistema de control de acceso que permita modificar o actualizar los registros solamente al dueño de la mascota, las opciones se restringen a unas pocas. Entre las etiquetas más populares ofrecidas actualmente en el mercado, que cumplen con los requisitos de memoria mínimos se encuentran las etiquetas NTAG 21x, MIFARE Classic, Topaz 512, MIFARE PLUS y MIFARE DESFire.

De acuerdo a la tabla hecha a partir de las especificaciones de los fabricantes, podemos descartar la etiqueta Topaz 512, ya que no cuenta con ningún sistema de control de escritura, por lo tanto no es una opción a considerar para el desarrollo de este proyecto. Las etiquetas del tipo MIFARE Plusestán diseñadas especialmente para aplicaciones de pago por lo que poseen un sistema de criptografía avanzada, tipo AES-128 (Advanced

[20] Estimado a partir de los datos a almacenar

Encryption Standard). Las etiquetas MIFARE DESFire, que también poseen una amplia capacidad de memoria, (incluso mucha más memoria de lo que la aplicación requiere) poseen además varios métodos de encriptación configurables, incluidos 2KTDES, 3KTDES y AES-128. Los dos tipos de etiqueta mencionados anteriormente, aunque cumplen con los requisitos de memoria requeridos, se han descartado para este proyecto, ya que debido a sus características avanzadas de encriptación de datos son demasiado costosas para la implementación del sistema.

	Tamaños de memoria disponibles (bytes)	Memoria disponible para el usuario (bytes)	Velocidades de transmisión de datos (Kbit/s)	Compatibilidad con las especificaciones del NFC Forum	Seguridad
NTAG 21x	540 924	504 888	106	Si	Contraseña de 32 bits
MIFARE Classic	1024 4096	720 3360	106	No	Crypto-1 de 48 bits
Topaz 512	512	454	106	Si	No
MIFARE Plus	2048 4096	1440 3360	106 / 212 / 424 / 848	No	AES-128 / Crypto-1 de 48 bits
MIFARE DESFire	2528 5088 8192	2304 4864 7936	106/848	Si	2KTDES / 3KTDES / AES-128

Tabla 2. Algunos tipos de etiqueta NFC disponibles en el mercado y sus especificaciones

Finalmente quedan dos tipos de etiqueta, las etiquetas NTAG 21x (en este proyecto, nos referimos solamente a las etiquetas NTAG 215 y NTAG 216) y las MIFARE Classic. Estos tipos de etiqueta poseen sistemas de seguridad diferentes, en las etiquetas del tipo NTAG 21x, el sistema de seguridad consiste en una contraseña de 32 bits que protege una estructura interna de 16 páginas a través de bytes de bloqueo dinámicos(NXP Semiconductors, 2015); mientras que las etiquetas del tipo MIFARE Classic poseen un sistema de encriptación asimétrico de 48 bits, con el cual es posible establecer dos claves, una clave pública (key A) y una clave privada o secreta (key B). A través de ellas es

posible bloquear determinados sectores de la estructura interna de la etiqueta (NXP Semiconductors, 2012a, 2012b).

Debido a queel sistema de seguridad de las etiquetas MIFARE Classic es relativamente más robusto que el de las etiquetas NTAG 21x,y debido a que MIFARE es la tecnología de chips sin contacto más utilizada en todo el mundo[21], se ha decidido desarrollar el proyecto específicamente para este tipo de etiquetas, aunque no se descarta un desarrollo posterior que abarque un mayor tipo de etiquetas, debido principalmente a que las etiquetas MIFARE Classic no son universalmente compatibles con todos los dispositivos según las especificaciones del NFC Forum, ya que la compañía NXP es propietaria de varias tecnologías y protocolos de comunicación patentados bajo la marca MIFARE.

4.2 PLANIFICACIÓN DEL FUNCIONAMIENTO

Para desarrollar correctamente la aplicación, lo primero que se debe realizar es un "bosquejo" mental de las pantallas necesarias para que la aplicación cumpla con la funcionalidad requerida y al mismo tiempo sea sencilla de manejar para el usuario.

Hay dos propósitos fundamentales que debe cumplir la aplicación, leer una etiqueta NFC que pueda contener información relevante acerca de una mascota extraviada y de su propietario y permitir la escritura de datos en la etiqueta solamente por el propietario de la mascota. Teniendo en cuenta esto, debe haber una pantalla que permita escoger entre las dos actividades.

Luego de esa pantalla, el flujo de la actividad se divide, si el usuario escoge la opción de lectura, se debe desplegar una pantalla que muestre el contenido de la etiqueta de forma adecuada al contenido. Por otro lado, si el usuario escoge la opción de escritura, se debe realizar un proceso de autentificación o validación, en el cual el sistema sea capaz de identificar si el usuario efectivamente está habilitado para escribir la etiqueta NFC.

A continuación de que el usuario haya escogido la opción de escritura, y suponiendo que no ha habido problemas con el proceso de validación, el usuario debe tener la opción de escoger el tipo de mascota que posee, y de acuerdo a ello variará nuevamente la pantalla que se mostrará.

Independientemente del tipo de mascota escogido, estas pantallas deben contener un formulario específico para cada tipo de mascota. Cuando el usuario haya llenado todos los campos requeridos se pasará a otra pantalla en la cual el propietario llenará sus datos personales y de contacto.

Si el proceso de escritura ha sido correcto, se mostrará una nueva pantalla indicando que la escritura de la etiqueta NFC se ha realizado con éxito, de lo contrario se debe mostrar un mensaje que le indicará al usuario que debe volver a intentarlo, porque ha habido una falla en la escritura. Si el proceso de validación de la etiqueta no ha sido correcto, se debe

[21] https://www.mifare.net/en/

mostrar un mensaje indicando al usuario que no es posible escribir esa etiqueta, ya que no está habilitado para ello, posiblemente debido a que no es el dueño de la mascota quién está intentando escribirla.

Teniendo en cuenta que para el proceso de validación, el usuario debe colocar la etiqueta cerca del teléfono, y que el proceso de escritura de la etiqueta también requiere que el usuario acerque la etiqueta al teléfono, se ha decidido que primero se mostrarán los formularios de datos de la mascota y del propietario, y luego se hará la validación y la escritura como tal, en un mismo proceso, así el usuario se evita tener que acercar dos veces la etiqueta al teléfono.

El proceso de autentificación para realizar la escritura de la etiqueta también se ha pensado para que el usuario no tenga que establecer contraseñas que harían más difícil e incómodo el manejo de la aplicación. Para ello, se partió de unas suposiciones razonables, en las cuales el propietario de la mascota también es el propietario del dispositivo con tecnología NFC y de la etiqueta NFC que pretende escribir. Al asumir esto, el sistema de autentificación se haría a través del teléfono directamente por medio de la aplicación, de tal manera que ningún otro dispositivo ni ninguna otra aplicación serían capaces de escribir una etiqueta que fue previamente escrita mediante esta aplicación.

Adicional a las pantallas descritas anteriormente, es posible incluir en el desarrollo de la aplicación, pantallas superpuestas para la inserción de datos específicos, tales como las fechas de vacunación o algún otro dato en específico.

4.3 CREACIÓN DE LOS LAYOUTS

Después de definir el número de pantallas que tendrá la aplicación, se procede a estructurar de forma definitiva los detalles de cada una.

4.3.1 Presentación

Muchas de las aplicaciones que usamos a diario en nuestros dispositivos muestran una pantalla de presentación al inicio durante unos pocos segundos. A este tipo de pantallas se les denomina SplashScreen.

En esta pantalla se planea colocar el logotipo de la aplicación, para ello primero se configura la pantalla para que permanezca siempre en sentido vertical y para que esta se muestre completamente (ocultando la barra de título o la Action Bar dependiendo de la versión de Android usada) utilizando para ello los métodos setRequestedOrientation() y requestWindowFeature(). A continuación definimos una cuenta atrás usando las clases Timer y TimerTask, mientras se está ejecutando esta cuenta atrás se mostrará el SplashScreen, una vez se haya cumplido la cuenta atrás se lanzará la siguiente actividad y el SplashScreen dejará de ser visible[22].

[22] https://amatellanes.wordpress.com/2013/08/27/android-crear-un-splash-screen-en-android/

Figura 10. Logotipo de la aplicación

4.3.2 Leer / Escribir

A través de este layout el usuario es capaz de escoger entre los dos modos de funcionamiento de la aplicación. A través de esta pantalla, también se detecta si el dispositivo tiene el hardware NFC necesario para el funcionamiento de la aplicación. Si el dispositivo no lo posee, se redirecciona hacia la pantalla "leer", mostrando al usuario el mensaje: "Este dispositivo no soporta NFC". De igual forma, si está desactivada la conexión de intercambio de datos a través de los ajustes del dispositivo, se mostrará el mensaje: "NFC deshabilitado. Por favor habilítelo en los ajustes del dispositivo e inténtelo nuevamente".

4.3.3 Leer

Cuando el usuario escoge ésta actividad se despliega inmediatamente una pantalla en blanco y la aplicación comienza a interactuar con el hardware del dispositivo a través de la clase NfcAdapter. Cuando el dispositivo Android detecta alguna etiqueta, ésta se procesa a través de un mecanismo conocido como "intención". A través de los filtros de intención Android determina que actividad poner en marcha. Para las etiquetas NFC, el sistema operativo dispone de tres filtros de intención ordenados de mayor a menor prioridad:

1. ACTION_NDEF_DISCOVERED: Si el sistema detecta una etiqueta con formato NDEF y ésta contiene una carga útil bien definida (TNF_WELL_KNOWN), el sistema intentará iniciar una actividad determinada.

2. ACTION_TECH_DISCOVERED: Esta intención es lanzada directamente si la etiqueta escaneada contiene datos NDEF, pero éstos no son de tipo MIME o URI; si no se ha registrado ninguna actividad para manejar mensajes NDEF o si la etiqueta no contiene datos NDEF pero esta es de una tecnología conocida (especificada en una lista de tecnologías <tech-list> en el AndroidManifest).

3. ACTION_TAG_DISCOVERED: Esta intención se inicia si no existe ninguna actividad que maneje las intenciones anteriores.

En resumen, esto significa que el sistema al detectar una etiqueta NFC, trata de iniciar una actividad, ya sea ACTION_NDEF_DISCOVERED o ACTION_TECH_DISCOVERED. Si no hay ningún filtro para este propósito, intenta iniciar una actividad con la siguiente intención de menor prioridad (ya sea ACTION_TECH_DISCOVERED o ACTION_TAG_DISCOVERED) mostrando una selección de posibles aplicaciones para el manejo de esa intención. Finalmente si no hay aplicaciones que manejen alguno de estos intentos, el sistema no hace nada.

Cada aplicación necesita filtrar alguna de estas intenciones con la prioridad más alta de acuerdo a sus requerimientos. Para esta aplicación, se ha hecho un filtro de intención en el AndroidManifest para detectar etiquetas con formato NDEF, y en la actividad "leer" a través de la clase onNewIntent. La información de la etiqueta se obtiene asignando un objeto tipo "Tag" a la etiqueta detectada por medio de la instrucción EXTRA_TAG. Debido a esto, incluso cuando no se haya iniciado la aplicación, al acercar una etiqueta que maneje el estándar para intercambio de mensajes NDEF, y que además contenga en su carga útil un mensaje en formato de texto plano (MIME_TEXT_PLAIN) se iniciará esta actividad mostrando el contenido de la etiqueta.

Es importante tener en cuenta que cuando la aplicación está abierta mostrando el contenido de una etiqueta previamente acercada, y se acerca una nueva etiqueta al dispositivo, no se debería reiniciar la aplicación, sino que se debería mostrar directamente el contenido de la nueva etiqueta. Esto se soluciona empleando un "foreground dispatch" (sistema de despacho en primer plano) y posteriormente creando un PendingIntent al cual le pasamos la indicación FLAG_ACTIVITY_SINGLE_TOP para que la actividad no se cree de nuevo y se reutilice si ya está presente en la pila de actividades.

El sistema de lectura de esta actividad se maneja a través de tareas asíncronas[23], esto permite realizar operaciones en segundo plano y publicar los resultados sobre la interfaz de usuario sin tener que manipular el hilo principal de la actividad. Se recomienda el uso de esta clase abstracta para operaciones de corto plazo (algunos segundos), tal como la lectura de datos de la etiqueta NFC. El uso de AsyncTask permite ejecutar el proceso de lectura en un subproceso de fondo, y al obtener el resultado de la lectura publicarlo en la interfaz de usuario.

Durante el transcurso de ésta se utilizan tres tipos de datos:

1. Parámetros: Son enviados a la tarea en ejecución, para que esta se pueda ejecutar.

2. Progreso: Hace referencia al tipo de unidades publicadas de fondo durante el progreso de la tarea.

3. Resultado: Es el tipo del resultado arrojado por el cálculo de fondo.

[23] http://developer.android.com/intl/es/reference/android/os/AsyncTask.html

Si alguno de estos tipos no se utiliza, se debe marcar como Void (vacío).

A la vez, una tarea asíncrona está compuesta por cuatro pasos:

1. OnPreExecute: Es invocado en el subproceso de interfaz de usuario antes de ejecutar la tarea. Este paso se utiliza normalmente para configurar la tarea, por ejemplo, mostrando una barra de progreso en la interfaz de usuario.

2. doInBackground: Invoca al subproceso de fondo inmediatamente después de que OnPreExecute termina de ejecutarse. Este paso se utiliza para realizar el cálculo de fondo. En este paso también se puede utilizar publishProgress para hacer el cálculo de una o más unidades de progreso.

3. onProgressUpdate: Este método se utiliza para mostrar cualquier forma de progreso en la interfaz de usuario, mientras que el cálculo de fondo se sigue ejecutando. Por ejemplo, se puede utilizar una barra de progreso o mostrar registros en un campo de texto.

4. onPostExecute: Este método es invocado en el subproceso de interfaz de usuario una vez finalizado el cálculo de fondo. El resultado de este cálculo se pasa a la interfaz de usuario como un parámetro.

Si el dispositivo detecta una etiqueta que contiene en su interior un registro NDEF bien conocido (TNF_WELL_KNOWN) tipo texto RTD_TEXT ("T" o 54 en hexadecimal según la codificación UTF), la carga útil se debe interpretar de acuerdo a la especificación técnica sobre registros de texto (NFC Forum, 2006), en la cual se establece la composición típica que debe encabezar la carga útil(PAYLOAD) de un registro NDEF.

Disposición	Contenido	Explicación	Información de la sintaxis
0	N/A	Bit IL=0 (No está presente el campo ID), SR=1 (Registro corto)	Encabezado del registro NDEF
1	0x01	Longitud del tipo de registro	
2	0x10	Longitud de la carga útil (16 bytes)	
3	"T"	Codificación binaria del tipo de registro	
4	0x02	Byte de estado: Codificado en UTF-8, y con dos bytes de código de lenguaje	Carga útil
5	"en"	Código ISO para el idioma	
7	"Hello, world!"	String codificado en UTF-8	Cuerpo de texto actual

Figura 11. Composición típica de un registro de texto(NFC Forum, 2006)

En esta especificación se establecen dos posibles codificaciones para el texto, UTF-8 y UTF-16, también se establece un bit llamado RFU (reservado para usos futuros) y un campo de 6 bits para establecer el código de lenguaje según la IANA. Este código debe estar enmascarado bajo el valor hexadecimal 3F (63 decimal), que corresponde al valor máximo que se puede representar con 6 bits.

Número de bit (0 es el bit menos significativo)	Contenido
7	0: El texto está codificado en UTF-8 1: El texto está codificado en UTF-16
6	RFU (Debe ser establecido en cero)
5..0	Longitud del código de lenguaje IANA

Figura 12. Codificación del byte de estado(NFC Forum, 2006)

4.3.4 Tipo de mascota

Si el usuario escoge la opción de escribir, se abrirá una pantalla mostrando al usuario tres opciones diferentes de tipo de mascota para escoger: un canino, un felino y otro tipo de mascota. Si el usuario escoge la opción "Otro", se desplegará un formulario básico, en el cuál se pueden llenar datos comunes para varios tipos de mascota no específicos, tales como nombre, raza, color y descripción del pelaje y discapacidades o enfermedades. En este tipo de campos se usó una combinación de TextViews para mostrar el enunciado y EditTexts para introducir los datos. En esta pantalla básica también se utilizaron dos Spinners para que el usuario tenga la posibilidad de escoger el género de la mascota y si el animal está esterilizado o no. Al final se colocó un botón a través del cual el usuario accede a otra pantalla en la cual puede introducir sus datos personales, pero se ha colocado una condición para que pueda pasar a esta pantalla, y es que ninguno de los EditText debe estar vacío. Si el usuario presiona este botón sin haber llenado los campos de texto anteriores se mostrará un mensaje mediante un Toast.

Si el usuario escoge la opción "Un canino" o "Un felino" se mostrarán los mismos campos que en el caso anterior, pero adicionalmente también se mostrarán varios Checkboxes, en los cuales el usuario tiene la opción de elegir las vacunas que posee actualmente el animal y si éste está desparasitado. La diferencia entre ambas pantallas radica únicamente en los registros de vacunación mostrados, ya que al ser dos especies de animales diferentes, también serán diferentes las vacunas que cada uno requiere, sin embargo tienen en común dos campos:

- La vacuna antirrábica, que en muchos países es de obligatoriedad, ya que es un virus que puede ser trasmitido a cualquier mamífero, incluyendo al ser humano[24].

[24] https://es.wikipedia.org/wiki/Rabia

- La desparasitación, que previene parásitos tanto internos como externos, que pueden afectar negativamente la salud de la mascota, e incluso ser transmitidos a los seres humanos o a otros animales.

A continuación se hará una breve reseña de los tipos de vacunas específicos para los caninos:

- Polivalente canina: Este tipo de vacuna engloba varios tipos de vacuna en una sola dosis, aunque no es obligatorio este tipo de vacunas, por no presentar un riesgo grave para la salud de los seres humanos, es muy recomendable vacunar al animal para prevenir posibles contagios y deterioro en su salud. Generalmente protegen contra alguna o todas estas enfermedades:

    - Moquillo (Distemper)
    - Hepatitis (Adenovirus tipo 2)
    - Parvovirosis
    - Leptospirosis
    - Coronavirus

- Traqueobronquitis: Esta vacuna protege contra enfermedades del sistema respiratorio, entre ellas se encuentran:

    - Parainfluenza canina
    - Bordetelosis o tos de las perreras

- Especial cachorros: Esta vacuna incluye enfermedades que son especialmente críticas para la salud de los cachorros. Generalmente protege a la mascota de estas dos enfermedades:

    - Moquillo (Distemper)
    - Parvovirosis

También existen en el mercado vacunas pentavalentes, hexavalentes (séxtuples) u octovalentes (óctuples), que protegen contra alguna o contra las dos enfermedades respiratorias e incluyen además protección contra otras enfermedades que afectan a los caninos, y que están especificadas dentro del campo "Polivalente canina". Es necesario que el propietario de la mascota se asesore de un veterinario para saber exactamente contra que enfermedades ha sido vacunada la mascota y tener en cuenta estas fechas de vacunación y cada cuanto tiempo es necesario revacunar.

La vacunación para los felinos comprende las siguientes vacunas:

- Triple: Este tipo de vacuna engloba varios tipos de vacuna en una sola dosis, aunque no es obligatorio este tipo de vacunas, por no presentar un riesgo grave para la salud de los seres humanos, si es muy recomendable vacunar al animal para prevenir

posibles contagios y deterioro en su salud. Generalmente protegen contra las siguientes enfermedades:

- Panleucopenia
- Rinotraqueitis
- Calicivirus

- Leucemia: Esta vacuna protege al felino contra un retrovirus que causa cáncer en las células sanguíneas y puede llegar a ser mortal.

- PIF: Esta vacuna protege al felino contra el virus de la peritonitis infecciosa felina.

Al elegir entre cualquiera de las opciones de "Vacunas y desparasitación" se abrirá una nueva pantalla en la cual el usuario podrá establecer la fecha en la cual se ha vacunado el animal a través de un DatePicker. Cabe aclarar que esta fecha no puede ser posterior a la fecha actual, de lo contrario se mostrará un mensaje para que el usuario rectifique la fecha de vacunación o desparasitación seleccionada. Después de que el usuario ha establecido una fecha correcta, la aplicación retornará al formulario anterior en el cual se encuentran los otros datos de la mascota y se habrá marcado en el Checkbox la vacuna correspondiente o la desparasitación. También es posible descartar alguna vacuna o la opción de desparasitación establecida previamente al presionar nuevamente sobre el Checkbox correspondiente.

4.3.5 Datos del propietario

Una vez que el usuario haya llenado todos los campos concernientes a los datos de su mascota podrá pasar a este formulario, en el cual podrá incluir sus datos básicos de contacto, entre los cuales se encuentran el nombre del propietario, la ciudad de residencia, su número de teléfono y un correo electrónico de contacto.

Si el usuario acerca una etiqueta válida al dispositivo, se mostrará un mensaje indicando que se ha detectado una etiqueta válida para su escritura. Por el contrario, si el usuario acerca una etiqueta de una mascota de la cual no es propietario, o una etiqueta destinada para otros usos (por ejemplo tarjetas para el pago de servicios de transporte público) se mostrará el mensaje: "Usted no está habilitado para escribir esta etiqueta".

4.3.6 Escribir

Después de que el usuario llena todos los campos solicitados, puede oprimir el botón "Escribir", si faltan campos por llenar, aparecerá un "Toast" indicando que aún faltan datos por llenar. El usuario debe mantener cerca al dispositivo la etiqueta NFC al momento de presionar este botón. Si al momento de escribir no se detecta ninguna etiqueta cerca al dispositivo se mostrará el mensaje: "Por favor acerque una etiqueta NFC tipo MIFARE Classic o inténtelo nuevamente". Si el procedimiento se ha realizado correctamente aparecerá un mensaje de confirmación y un único botón para volver al menú inicial.

En cada uno de los formularios desplegados cuando se escoge el tipo de mascota se recolectan y organizan los datos introducidos por el usuario y luego estos se pasan como parámetro a través de un Intent con la instrucción putExtra. Cuando se pasa a la siguiente pantalla (datos del propietario) se obtienen en forma de string a través de los comandos getIntent().getExtras().getString().Después de esto se codifica el mensaje en un arreglo de registros de tipo NdefRecord y se encapsula en un mensaje representado por la clase NdefMessage.

Una etiqueta del tipo MIFARE Classic se compone de varios sectores que a la vez están subdivididos en bloques. El tamaño de estos bloques siempre es de 16 bytes. La etiqueta MIFARE Classic de 1K (que es la más recomendada para este proyecto) está compuesta por 16 sectores cada uno con 4 bloques. El primer sector de la etiqueta es conocido como sector MAD (MIFARE Application Directory), y contiene los datos del fabricante del circuito integrado y el identificador único de la etiqueta (UID), también llamado número de serie. En este sector también se encuentran los identificadores de aplicación (AID), que en este caso están reservados para identificar los 15 sectores de la etiqueta con datos NDEF. Un sector con datos NDEF se conoce como un sector NFC.

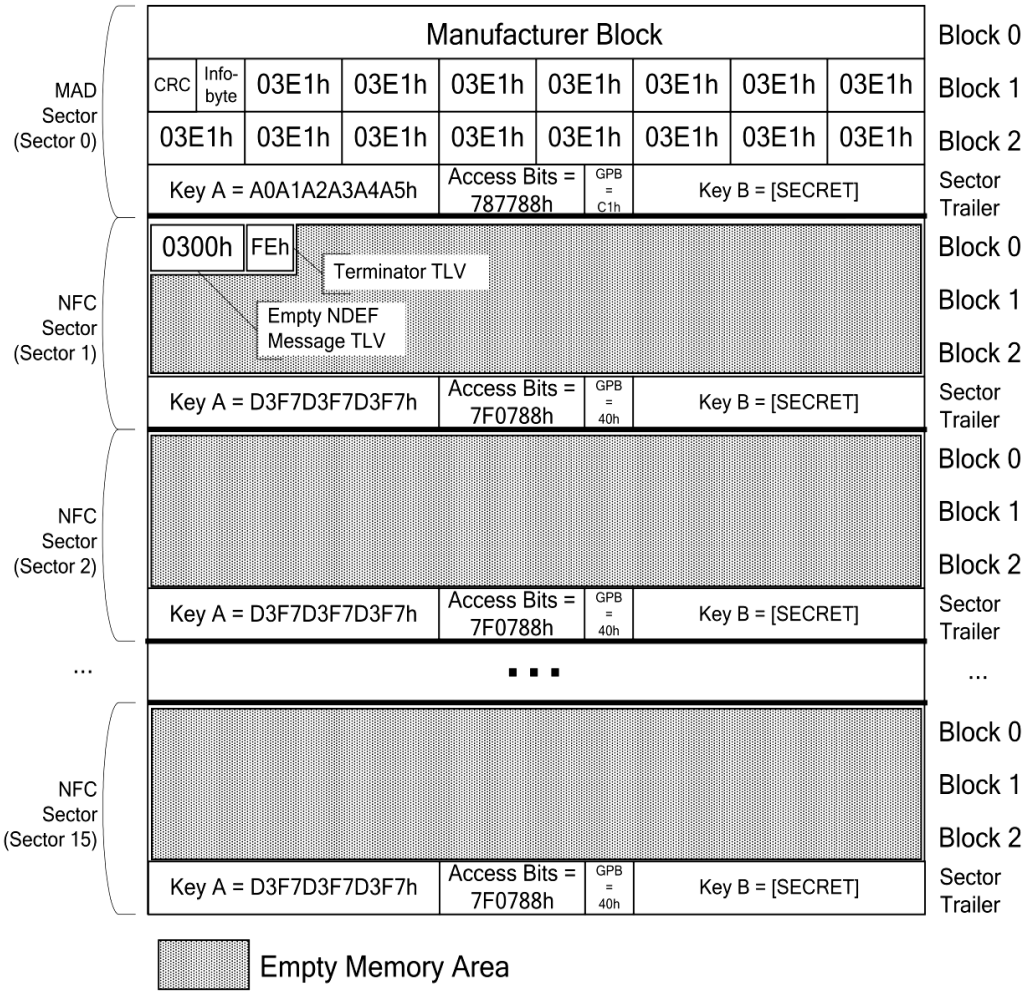


Figura 13. Composición lógica de una etiqueta MIFARE Classic de 1K (NXP Semiconductors, 2012a)

Los dos campos de un AID están establecidos de la siguiente manera: el primer byte, con el valor 03h para identificar que se trata de un sector NFC, y el siguiente byte con el valor E1h para indicar que el grupo de sectores contiene datos NDEF. Si hay más de un sector NFC, estos deben ser contiguos (NXP Semiconductors, 2012a).

Con el fin de almacenar un Mensaje NDEF en la etiqueta, el mensaje debe ser envuelto en un bloque TLV.TLV es una abreviatura de tres campos diferentes: T, que denota el tipo de bloque, L su longitud y V su valor. Un bloque TLV consiste de uno o más bytes. El campo T es el único campo obligatorio y utiliza un solo byte para identificar el tipo de bloque.

Nombre del bloque TLV	Valor del campo	Descripción corta
NULL TLV	00h	Debe ser usado para rellenar áreas de memoria, que el dispositivo lector ignorará
NDEF Message TLV	03h	Contiene un mensaje de tipo NDEF
Proprietary TLV	FDh	Información propietaria de la etiqueta
Terminator TLV	FEh	Último bloque TLV en el área de datos

Figura 14. Tipos de bloques TLV(NXP Semiconductors, 2012a)

El campo L proporciona en bytes el tamaño del bloque, y se puede representar de dos formas diferentes:

- Formato de un byte: Codifica el valor de la longitud del bloque en bytes que van desde 00h hasta FEh. Si contiene el valor FFh, el valor de la longitud del bloque está en un formato de más de un byte.

- Formato de tres bytes consecutivos: Codifica el valor de la longitud del bloqueen bytes que van desde 00FFh hasta FFFEh. El primer byte es una bandera igual a FFh, e indica que dos o más bytes deben ser interpretados como una palabra. El valor FFFFh está reservado para usos futuro (RFU).

Sin embargo, dependiendo del tipo de bloque, este campo puede no estar presente(NXP Semiconductors, 2012a).El campo V solo estará presente cuando se encuentre el campo L y este no es igual a 00h. En este caso contendrá N bytes de datos en el formato indicado por el tipo de bloque. En este campo se hallará la carga útil si se trata de un bloque del tipo NDEFMessage.

El dispositivo deberá escribir los bloques TLV en un orden específico dentro del área de datos siguiendo las siguientes reglas(NXP Semiconductors, 2012a):

- Los bloques TLV deberán ser escritos en orden, empezando por el byte cero del sector NFC con la dirección de memoria más pequeña.

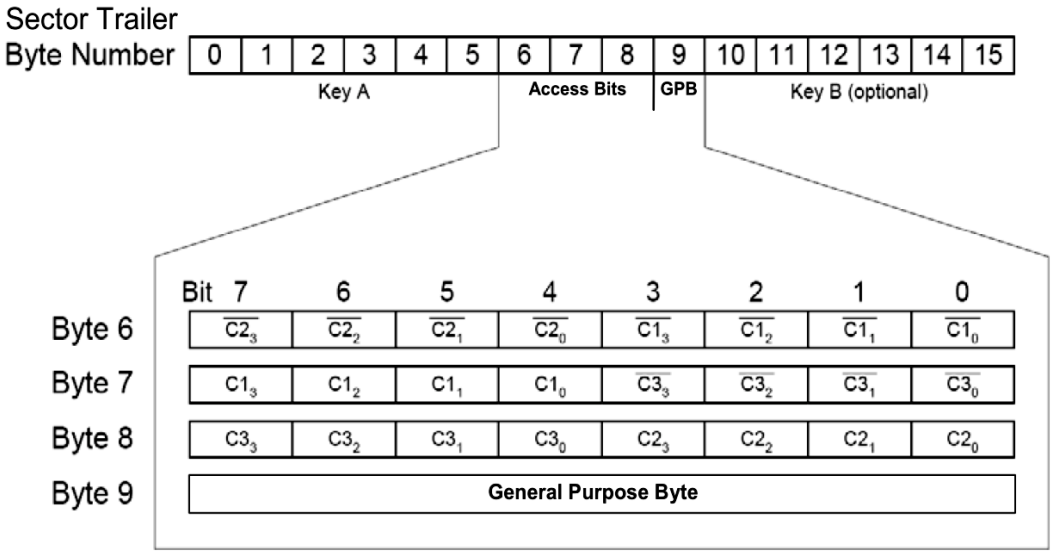
- Un bloque TLV puede ser almacenado en dos o más sectores NFC.

- Si está presente el Terminator TLV, este será el último bloque en la etiqueta.

Un mensaje NDEF típico comenzará con el byte 03h y terminará con el byte FEh.

4.3.7 Validación

Tal como se puede observar en la estructura interna de registros de la etiqueta, existe un bloque especial al final de cada sector conocido como Sector Trailer. El Sector Trailer está compuesto por la clave A (key A), la clave B (key B), los bits de acceso y el GPB (Bit de propósito general), a través de estos bits es posible establecer condiciones de acceso al sector, concediendo permisos de solo lectura o de lectura y escritura luego de un proceso de autentificación a través de las claves correspondientes.



Life Cycle State	Access Bits	MAD1 and MAD2 Sectors Access Bits Values	NFC Sectors Access Bits Values	MAD1 and MAD2 Sector Trailer Bytes Values Byte 6	Byte 7	Byte 8	NFC Sector Trailer Bytes Values Byte 6	Byte 7	Byte 8
INITIALIZED and READ/WRITE	$C1_0\ C2_0\ C3_0$	100b[i]	000b	78h	77h	88h	7Fh	07h	88h
	$C1_1\ C2_1\ C3_1$	100b	000b						
	$C1_2\ C2_2\ C3_2$	100b	000b						
	$C1_3\ C2_3\ C3_3$	011b	011b						
READ-ONLY	$C1_0\ C2_0\ C3_0$	010b[i]	010b	07h	8Fh	0Fh	07h	8Fh	0Fh
	$C1_1\ C2_1\ C3_1$	010b	010b						
	$C1_2\ C2_2\ C3_2$	010b	010b						
	$C1_3\ C2_3\ C3_3$	110b	110b						

i. This value for the access bits $C1_0\ C2_0\ C3_0$ of sector 0 (related to the manufacturer block) is suggested and it may change.

Figura 15. Estructura del Sector Trailer y codificación de los bits de acceso (NXP Semiconductors, 2012a)

El byte de propósito general no se utiliza en el mecanismo de acceso a la etiqueta, sin embargo, indica la versión de la estructura (mapa) utilizada para almacenar los datos NDEF en la etiqueta, junto con las condiciones de acceso. Generalmente el GPB se establece en 40h para indicar que se utiliza la versión 1.0 y para establecer condiciones de lectura y escritura para el sector.

Si es la primera vez que se escribe en la etiqueta es necesario darle el formato NDEF a través de la instrucción format() en Android Studio. Ésta instrucción escribe el mensaje NDEF simultáneamente (o un NULL TLV), y establece condiciones de lectura y escritura de la etiqueta por defecto, sin solicitar ninguna clave. Si la etiqueta ya ha sido escrita se utiliza la instrucción writeNdefMessage().

Para establecer la escritura de la etiqueta a través de la clave privada (key B) es necesario escribir byte a byte todo el Sector Trailer comenzando con la clave pública (key A) para un sector NDEF establecida por el NFC Forum(NXP Semiconductors, 2012a) seguida de los bits de acceso de cada sector NFC con las condiciones de acceso aconsejadas para el sector MAD y posteriormente establecer la clave privada (key B).

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
D3h	F7h	D3h	F7h	D3h	F7h

Figura 16. Clave pública (key A) establecida por el NFC Forum (NXP Semiconductors, 2012a)

El procedimiento de validación inicia en cada una de las pantallas del tipo de mascota seleccionada. En ellas se conforma una clave, de acuerdo al IMEI[25] del dispositivo. Esta será la clave secreta (key B) que se pasará como parámetro a la siguiente actividad. Si es la primera vez que se escribe en la etiqueta con la aplicación, se le coloca esta clave para evitar que cualquier otra persona modifique la información allí contenida, incluso con otra aplicación. Si se va a reescribir la etiqueta, es necesario primero validar que el dispositivo esté autorizado para tal fin a través de la clave privada (key B), posteriormente se desbloquean todos los sectores de la etiqueta estableciendo para ello las condiciones de acceso por defecto (lectura y escritura sin claves) para poder escribir el mensaje en formato NDEF y finalmente reescribir todos los Sector Trailers con las claves respectivas, junto con las condiciones de acceso originales.

4.4 CONCLUSIONES Y TRABAJOS FUTUROS

Después de analizar y comparar las diferentes alternativas de etiquetas NFC presentes en el mercado, su capacidad de memoria y sus características de seguridad en relación con los requerimientos del proyecto, se estableció que la etiqueta más indicada para tal fin era la etiqueta MIFARE Classic de 1K, con la cual se hicieron exhaustivas pruebas para detectar y corregir problemas en el funcionamiento de la aplicación. Las pruebas se realizaron con un dispositivo real, y no a través del emulador incluido en el programa Android Studio

[25] https://es.wikipedia.org/wiki/IMEI

debido principalmente a que el desenvolvimiento de la aplicación depende del hardware de cada dispositivo y del transceptor NFC específico de cada dispositivo.

Desafortunadamente las etiquetas empleadas en el desarrollo de este proyecto no son universalmente compatibles con todos los dispositivos según las especificaciones del NFC Forum. Sin embargo, como la empresa fabricante de las etiquetas MIFARE (NXP) es también la principal fabricante de transceptores NFC para dispositivos móviles, la gran mayoría de dispositivos portátiles es capaz de leer este tipo de etiquetas, solamente existen en el mercado unos pocos dispositivos que traen un transceptor de otro fabricante (Broadcom). Éstos dispositivos no reconocen la información contenida en las etiquetas MIFARE, solamente reconocen el UID.

Debido a que en el mercado existen muchas tarjetas de pago que incluyen la tecnología MIFARE, la empresa Broadcom decidió obtener la licencia MIFARE para integrarla en sus futuros transceptores[26], por esta razón se espera que cada vez existan más dispositivos móviles que puedan leer las etiquetas MIFARE Classic empleadas en este proyecto. Sin embargo también se planea como trabajo futuro ampliar la compatibilidad de la aplicación con etiquetas que cumplan con las especificaciones del NFC Forum.

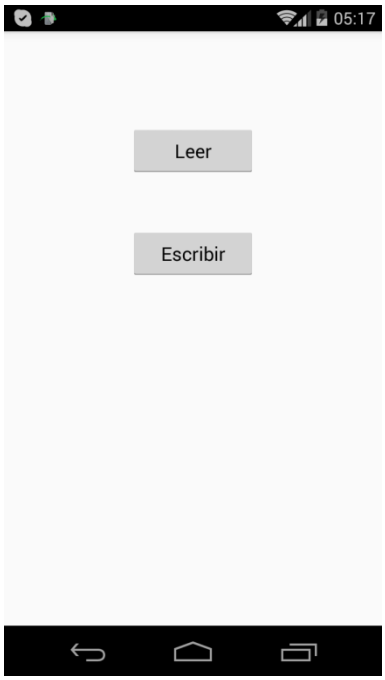
Como parte de un proyecto que va más allá de lo académico, se espera distribuir en un futuro esta aplicación en la tienda oficial para dispositivos Android, Google Play. Para este propósito, se ha incluido en la escritura de la etiqueta un registro de aplicación Android (AAR)[27], el cual es un registro NDEF de tipo externo que proporciona la certeza de que la aplicación se iniciará al escanear una etiqueta NFC que ha sido escrita con la aplicación desarrollada. Si al escanear una etiqueta se encuentra un AAR, se inicia la aplicación con el nombre del paquete incluido dentro del registro de aplicación. Si la aplicación no está presente en el dispositivo, se inicia la búsqueda dela aplicación a través de Google Play. Los AAR sólo son compatibles a nivel de aplicación y no en el nivel de actividad, debido a esto, el usuario también podrá leer esta etiqueta con cualquier otra aplicación.

Aunque esta aplicación ha sido diseñada para ayudar a que los propietarios recuperen a sus mascotas extraviadas también podría servir como un método de identificación de animales callejeros por parte de alguna entidad gubernamental o ambiental. También se sugiere a este tipo de entidades la implementación de algún mecanismo que permita constatar a cualquier persona que lea la etiqueta, cierta información susceptible de ser verificada, como por ejemplo la veracidad de las vacunas.

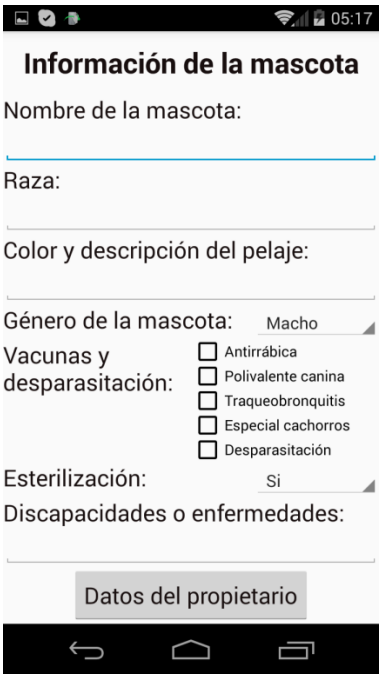
[26] https://www.broadcom.com/press/release.php?id=s916823
[27] http://developer.android.com/intl/es/guide/topics/connectivity/nfc/nfc.html

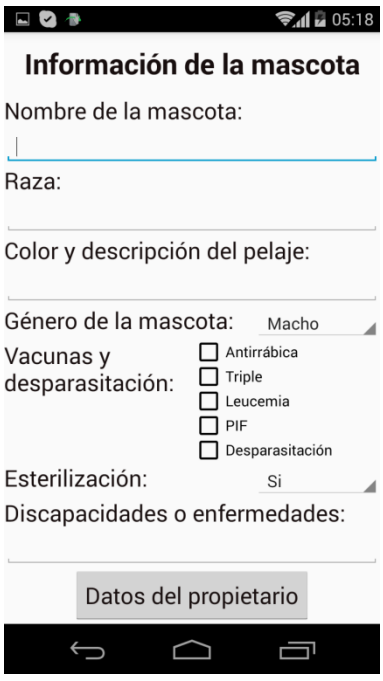
ANEXO: Pantallas de la aplicación

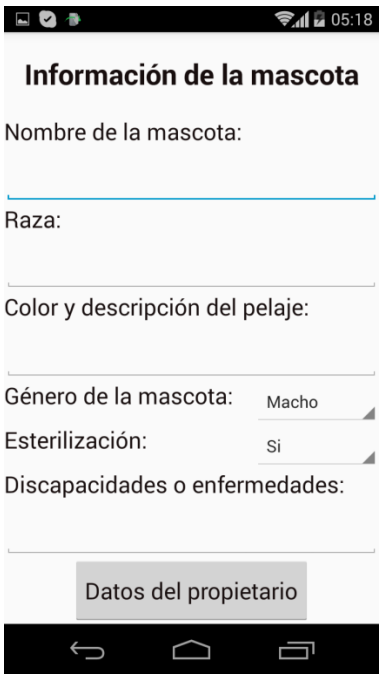
PANTALLAS DE SELECCIÓN DE OPERACIÓN (LEER / ESCRIBIR) Y SELECCIÓN DEL TIPO DE MASCOTA



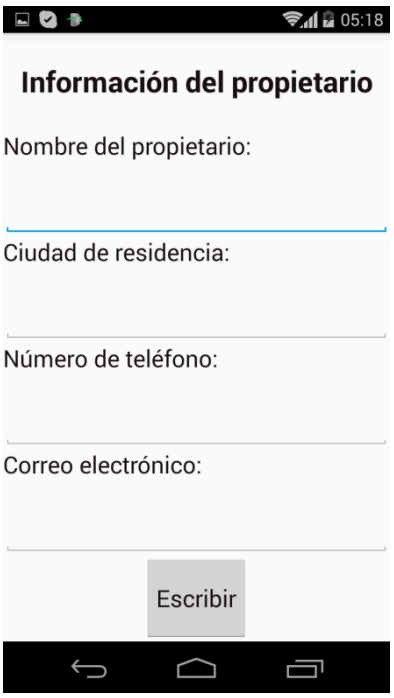
PANTALLAS DE DATOS DE LA MASCOTA (CANINO, FELINO Y OTRO TIPO DE MASCOTA RESPECTIVAMENTE)

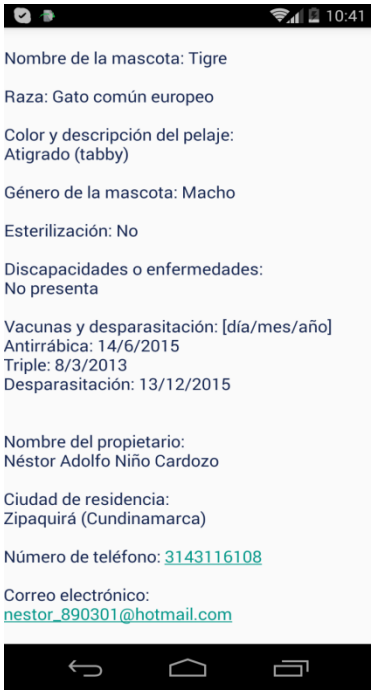




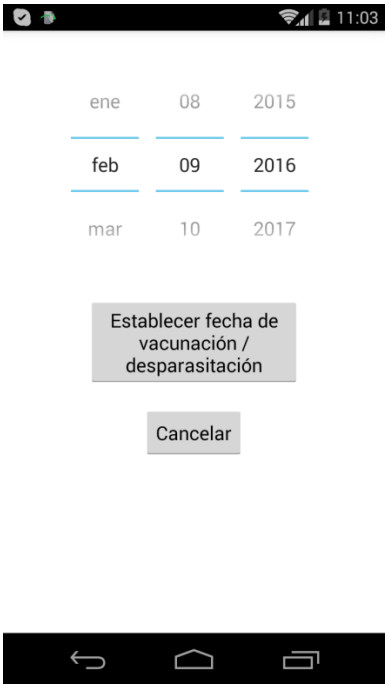


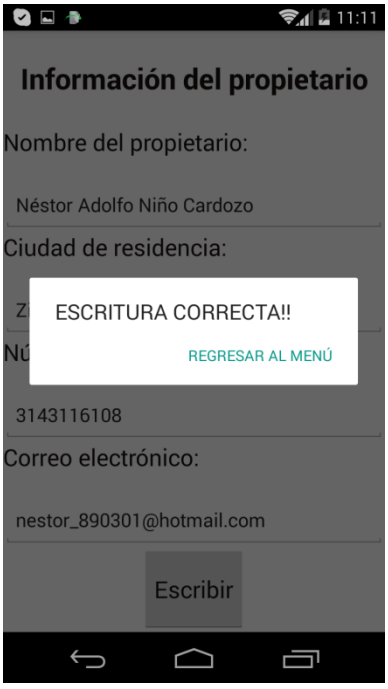
PANTALLA DE DATOS DEL PROPIETARIO Y PANTALLA DE LECTURA





PANTALLAS DE SELECCIÓN DE FECHA DE VACUNACIÓN / DESPARASITACIÓN Y CONFIRMACIÓN DE ESCRITURA





BIBLIOGRAFIA

Albrecht, K. (2007). Microchip-Induced Tumours in Laboratory Rodents and Dogs: A Review of the Literature 1990-2006. *Caspian*, (Nov 19), 337–349. Retrieved from http://www.farmtoconsumer.org/docs/O.pdf

Berzal Galiano, F. (2006). *Uso de excepciones en Java*. *Apuntes de programación orientada a objetos en Java: Fundamentos de programación y principios de diseño* (1 [a] Ed).

Caballero Nadales, L. (2012). *Informe Near Field Comunication (NFC)*. Barcelona.

Campa Ruiz, A. (2011). *Desarrollo de una aplicación de pago a través de la tecnología NFC.*

Catarinucci, L., Cappelli, M., Colella, R., Di Bari, A., & Tarricone, L. (2008). A novel and low-cost multisensor-integrated RFID tag for biomedical applications. *2008 IEEE International Symposium on Antennas and Propagation and USNC/URSI National Radio Science Meeting, APSURSI*, 4–7. http://doi.org/10.1109/APS.2008.4619694

Cavoukian, A. (2011). Mobile Near Field Communications (NFC) " Tap " n Go " Keep it Secure & Private." *PbD*.

Coskun, V., Ok, K., & Ozdenizci, B. (2013). *Professional NFC Application Development for Android*. Wiley. Retrieved from https://books.google.com.co/books?id=c4QRU17e494C

Freudenthal, E., Herrera, D., Kautz, F., Natividad, C., Ogrey, A., Sipla, J., … Estevez, L. (2007). Suitability of NFC for medical device communication and power delivery BT - 2007 IEEE Dallas Engineering in Medicine and Biology Workshop, DEMBS, November 11, 2007 - November 12, 2007, 51–54.

Liu, Y. (2007). Inter-organizational Information Sharing and Collaboration : A Case Study in the Pet ' s Information Management. *Solutions*, 6270–6272.

Liu, Y., & Shao, P. (2010). Applying RFID to the pet's information management to realize collaboration. *2010 7th International Conference on Service Systems and Service Management, Proceedings of ICSSSM' 10*, 917–922. http://doi.org/10.1109/ICSSSM.2010.5530114

Madlmayr, G., Kantner, C., & Grechenig, T. (2014). Near Field Communication, *4*(8), 351–367.

NFC Forum. (2006). Text Record Type Definition Technical Specification. Wakefield.

NXP Semiconductors. (2012a). AN1304 NFC Type MIFARE Classic Tag Operation.

NXP Semiconductors. (2012b). AN1305 MIFARE Classic as NFC Type MIFARE Classic Tag.

NXP Semiconductors. (2015). NTAG213/215/216 NFC Forum Type 2 Tag compliant IC with 144/504/888 bytes user memory.

Pongpaibool, P. (2008). A study on performance of UHF RFID tags in a package for animal traceability application. *5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON 2008*, *2*, 741–744. http://doi.org/10.1109/ECTICON.2008.4600537

Revelo Urrea, J. (n.d.). *Como crear tu primera aplicación Android desde cero*. Cali: Hermosa Programación.

Ting, J. S. L., Kwok, S. K., Lee, W. B., Tsang, A. H. C., & Cheung, B. C. F. (2007). A dynamic RFID-based mobile monitoring system in animal care management over a wireless network. *2007 International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2007*, 2085–2088. http://doi.org/10.1109/WICOM.2007.521

Upson, B. S. (2008). NETWORKING GOES TO, (January).