DESARROLLO DE UN SISTEMA TRANSACCIONAL QUE PERMITE REALIZAR PAGOS ELECTRONICOS IMPLEMENTANDO NFC

UNIVERSIDAD CATÓLICA ANDRES BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA TRANSACCIONAL QUE PERMITE REALIZAR PAGOS ELECTRONICOS IMPLEMENTANDO NFC

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

Realizado por: Castillo Sánchez, Roberick E.

Mora Hung, Miguel A.

Tutor académico: Ing. José Pirrone Puma

Fecha: Mayo de 2015

UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA
ESCUELA DE TELECOMUNICACIONES

DESARROLLO DE UN SISTEMA TRANSACCIONAL QUE PERMITE REALIZAR PAGOS ELECTRÓNICOS IMPLEMENTANDO NFC.

TRABAJO ESPECIAL DE GRADO

Presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: 16 puntos

J U R A D O E X A M I N A D O R

Firma: Nombre: Ing. José Pirrone

Firma: Nombre: José Javier Baiz

Firma: Nombre: José Gregorio Castillo

REALIZADO POR Castillo Sánchez, Roberick Eduardo

Mora Hung, Miguel Angel

PROFESOR GUIA Ing. José Pirrone

FECHA Junio de 2015

RESUMEN

APLICACIÓN ANDROID PARA REALIZAR PAGOS ELECTRÓNICOS IMPLEMENTANDO NFC

Roberick Eduardo Castillo Sánchez
rkc.hades@gmail.com
Miguel Ángel Mora Hung
Mora.miguel03@gmail.com

En búsqueda de dar solución a los problemas que presentan las formas de pagos actualmente utilizadas, como el retardo que normalmente presentan las transacciones a través de puntos de ventas o el riesgo de ser víctima de robo de identidad, se concibió la idea de desarrollar un sistema que sirva como alternativa para realizar pagos electrónicos implementando la tecnología NFC, con el fin de mitigar los problemas antes descritos.

Para alcanzar los objetivos planteados, se siguieron una serie de pautas en la metodología, que estipuló primeramente una investigación documental. Posteriormente, se desarrolló un sistema constituido por tres aplicaciones: Aplicación para dispositivos móviles ANDROID, que permite la comunicación del usuario con el sistema. Aplicación servidor, programada en PHP la cual, gracias a un servicio *web*, establece la conexión del usuario con el sistema a través de una red local WIFI. Por último, una aplicación principal, desarrollada en JAVA, encargada de recibir la información del dispositivo móvil mediante el lector NFC para realizar las transacciones. El correcto funcionamiento del sistema depende de una base de datos que almacene la información del cliente y las transacciones realizadas a través del sistema, por lo tanto, se usó el lenguaje SQL para la creación de la base de datos**.**

Las pruebas realizadas al sistema demostraron el correcto funcionamiento del mismo, ofreciendo rapidez en la realización de las transacciones, gracias a la implementación de la tecnología NFC, confiabilidad, gracias a los módulos de seguridad diseñados y facilidad de uso, gracias a las interfaces gráficas.

Palabras claves: ANDROID, NFC, bases de datos, sistema automatizado.

AGRADECIMIENTOS

Agradezco a mis padres Sonia Sánchez y Néstor Castillo, a mi hermano Nestor Castillo y a Kelys Castellanos por su apoyo y cariño, gracias a ustedes hoy soy mejor persona. Por sus palabras y consejos consegui el impulso necesario para culminar este Trabajo Especial de Grado cumpliendo otro requisito y asi culminar mi carrera universitaria, mil gracias

Roberick Eduardo Castillo Sánchez

A todos mis familiares, en especial a mi padre Miguel Alberto Mora por su apoyo incondicional y los valores que me inculcó, a mi madre Magda Hung por su amor y comprensión. A mis amigos de vida por sus sabios consejos y sus palmadas en la espalda durante aquellos momentos difíciles. A mis abuelos que en paz descansen... los extraño. Y por último a Julio Gutiérrez por su paciencia en las consultas. Gracias de corazón, sin ustedes jamás habría cumplido este trabajo

Miguel Ángel Mora Hung

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El desarrollo de nuevas tecnologías es la premisa que deben tener en mente investigadores, ingenieros y organizaciones de cualquier índole, a fin de mejorar y optimizar los procesos productivos y brindar a los usuarios satisfacción al momento de realizar cualquier tipo de tareas.

Hoy en día existen a nivel mundial millones de teléfonos inteligentes, los cuales se han convertido en una herramienta indispensable para las personas gracias a la capacidad de cómputo de los dispositivos y la posibilidad de permitir conexiones a internet. Estas características, además la gran cantidad de sensores y tecnologías que disponen, han permitido la creación de distintas aplicaciones para facilitar las tareas diarias de los usuarios. Estas funcionalidades deben ser aprovechadas tanto por empresas como por desarrolladores independientes para ofrecer servicios de calidad, lo que obliga a plantear nuevos modelos de negocios o adecuar las nuevas tecnologías a los modelo de negocios existentes.

El desarrollo de nuevas tecnologías es la premisa que deben tener en mente investigadores, ingenieros y organizaciones de cualquier índole, a fin de mejorar y optimizar los procesos productivos y brindar a los usuarios gran satisfacción al momento de realizar cualquier tipo de tareas.

El principal factor a ser tomado en cuenta en la implementación de cualquier tipo de tecnologías es la economía, el ingeniero debe poseer la virtud de diseñar soluciones que permitan alcanzar altos niveles de efectividad al menor costo posible. Por esta razón en la actualidad las soluciones a través de software han tomado gran auge, al facilitar herramientas que son creadas para ser ejecutadas en el hardware disponible, esto permite reducir ampliamente los costos de implementación.

El pagos de facturas en locales comerciales, es una de las actividades que cualquier ser humano necesita efectuar a diario, por lo tanto, disminuir el tiempo necesario para realizar esta actividad ofrece una mayor calidad de vida al usuario, puesto que este tiempo puede utilizarse para cualquier otro tipo de actividad. Para alcanzar esta premisa es necesario que existan alternativas y soluciones efectivas, fáciles de utilizar y seguras con la intención de lograr este fin

En este tomo se refleja el resultado de las investigaciones y procedimientos aplicados para lograr alcanzar los objetivos propuestos, se ofrece una estructura de cinco capítulos, titulados de la siguiente manera: Planteamiento teórico, Marco teórico, Metodología, Resultados y Conclusiones y recomendaciones. Estos capítulos buscan describir de manera fiel la problemática enfrentada y la justificación del proyecto, resaltando los objetivos planteados, el contenido teórico necesario para sustentar la investigación realizada para la elaboración de este tomo, la metodología utilizado y las actividades que se realizarán para el logro de los objetivos, los resultados que arrojaron al concluir estas actividades y por último las conclusiones que se obtuvieron al finalizar el Trabajo Especial de Grado.

CAPÍTULO I

Planteamiento del proyecto

En este capítulo se mostrará al lector las bases del Trabajo Especial de Grado, se observarán las necesidades que motivaron el desarrollo de esta investigación, el objetivo que se desea alcanzar al finalizar el Trabajo de Grado y sus alcances y limitaciones.

I.1 Planteamiento del problema

En la actualidad, son numerosas las transacciones realizadas a través de los puntos de ventas (POS por sus siglas en ingles), ya sea con tarjetas de débito o de crédito, esto debido a que las personas ya no acostumbran a llevar efectivo en sus billeteras. Según GAO RESEARCH, empresa líder en sistemas embebidos que ofrece servicios para implementar puntos de venta, afirma que se han realizado aproximadamente 28 millardos de transacciones a través de puntos de ventas en Estados Unidos y el tiempo promedio de cada una es de aproximadamente 40 a 50 segundos.

Según ATUS (Encuesta Americana de Empleo del Tiempo), desde que el estudio se inició en el 2003, los ciudadanos estadounidenses han disminuido sostenidamente el tiempo usado en compras y pagos de servicios. En el 2011 el promedio de minutos por día usado en compras paso de 45 minutos en el 2010 a 43.2 minutos. En el 2003 los estadounidenses usaban en promedio 48.6 minutos en compras al día

Adicionalmente en un estudio realizado por la firma estadounidense Rich Relevance en diciembre de 2011, se demuestra la aceptación del público a realizar compras por internet a través de ordenadores o dispositivos móviles, al registrar 3.400

millones de sesiones de compras, teniendo un crecimiento del 100% en sesiones de compra a través de dispositivos móviles. La implementación de este tipo de sistemas ha permitido que los ciudadanos estadounidenses puedan reducir el tiempo usado en compras y pagos de servicios.

Igualmente para el caso de Venezuela es notable que la compra de alimentos y pagos de servicios es la responsabilidad que más resta tiempo en la cotidianidad. Desarrollar sistemas que permitan a los venezolanos disminuir el tiempo usado en estas tareas resulta de imperiosa necesidad, para otorgar a los clientes mayor comodidad y efectividad al realizar sus compras cotidianas. La implementación de sistemas de automatización de pagos o compras, también genera beneficios para los establecimientos comerciales, debido a que al disminuir la cantidad de empleados necesarios en el área de cobranza, también lo hará el gasto en sueldos de empleados, gastos por el mantenimiento de los equipos de las cajas de pago, entre otros.

I.2 Objetivos de la investigación

I.2.1 Objetivo general

Analizar, diseñar e implementar un sistema transaccional que permite realizar pagos electrónicos implementando NFC.

I.2.2 Objetivos específicos

- Diseñar una aplicación ANDROID para transmitir información desde el dispositivo móvil hacia el punto NFC
- Diseñar una aplicación que permita el registro de usuarios y almacenar la información personal de los mismos.
- Diseñar una aplicación capaz de obtener los datos recibidos a través del punto NFC.

- Implementar módulos de seguridad

I.3 Alcances y limitaciones

I.3.1 Limitaciones

ANDROID actualmente es uno de los sistemas operativos para dispositivos móviles más expandidos actualmente, esto se puede evidenciar en la figura 1.

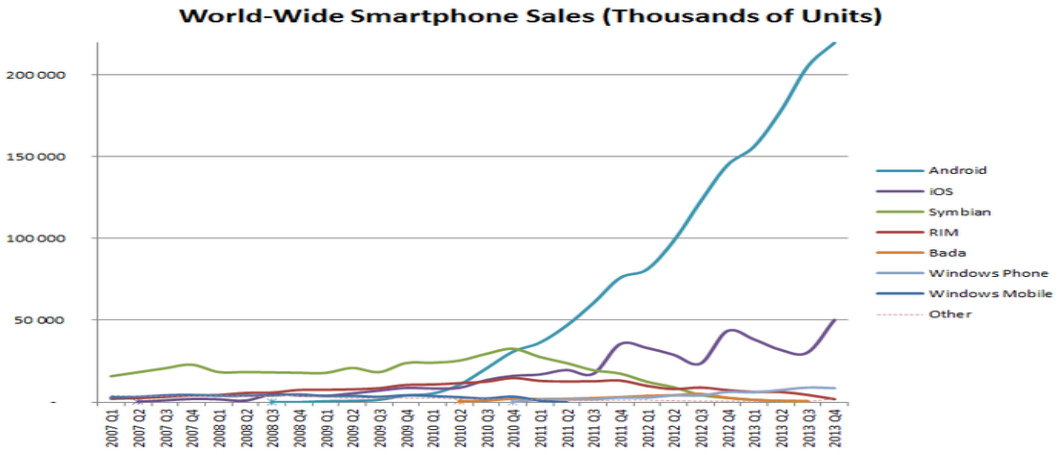


Figura 1. Ventas de teléfonos inteligentes a nivel mundial (miles de unidades)

Fuente: (Llamas, 2014)

La figura 1 nos ofrece una idea del poder de mercado que tiene ANDROID a nivel mundial, se puede observar fácilmente que hasta el último trimestre del año 2013, su más cercano competidor tiene un aproximado de un cuarto de las ventas que ANDROID cubre hasta esas fechas.

Para un sistema masivo de ventas, es ideal que funcione para cualquier dispositivo móvil, sin embargo por razones de tiempo, solo se desarrollará la

aplicación para el sistema operativo ANDROID, por ser el sistema operativo líder en ventas.

La aplicación necesita teléfonos que incluyan la tecnología NFC, sin embargo los teléfonos que ofrecen el sistema operativo ANDROID de gama baja y la mayoría de gama media no ofrecen esta plataforma, por lo cual, existe una limitación con la elección del dispositivo móvil que se requiere para la aplicación del trabajo especial de grado.

Este mismo aspecto tiene un impacto económico puesto que los teléfonos de gama alta son los más costosos del mercado, por lo cual, es necesario invertir en dispositivos móviles de estas características para el correcto funcionamiento del sistema que se requiere implementar.

NFC tiene múltiples tipos de tarjetas, en el caso de los dispositivos ANDROID disponibles, se utiliza el estándar ISO 14443 para las transacciones NFC, por lo tanto, se utilizarán solo equipos que utilicen este estándar.

I.3.2 Alcances

NFC es una tecnología que está tomando un gran auge a nivel mundial, países como España o Japón ya están utilizado esta tecnología para distintas aplicaciones que facilitan la vida cotidiana de sus pobladores, por lo tanto una de las intenciones de nuestro Trabajo Especial de Grado es implementar esta tecnología para incentivar el uso de la misma y se pueda dar a conocer dentro de nuestro país.

El alcance más importante, es ofrecer una alternativa para las formas de pago existentes para disminuir el tiempo requerido para los pagos de las compras realizadas por los clientes en cualquier establecimiento comercial.

I.4 Justificación

El tiempo es un factor importante en la vida cotidiana de cada persona, es un recurso que es imposible de recuperar. Las personas necesitan invertir parte de su tiempo para lograr adquirir distintos productos y satisfacer sus necesidades, realizar nuestros trabajos, descansar, etc.

Existen distintos tipos de pago, por lo general, el uso de efectivo es la manera más rápida de cancelar alguna factura que existe, el problema es el riesgo de llevar una alta suma de dinero, lo que amenaza la seguridad de las personas, es por eso que los medios de pago electrónicos han abierto un nuevo paradigma de cómo pagar día tras día, sin embargo, en muchas ocasiones este tipo de pagos requiere un poco más de tiempo para lograr efectuarse.

El sistema desarrollado trata de ofrecer una alternativa sencilla, efectiva, rápida y confiable para establecer una nueva forma de cancelar alguna factura, lo que reduce el tiempo invertido a la hora de pagar, todo esto gracias a la alta velocidad de transmisión que ofrece NFC y a la seguridad de la plataforma por el requerimiento de proximidad necesario para lograr la comunicación, además que NFC no solo se limita a *Tags* NFC sino que actualmente casi todos los dispositivos móviles llevan incluidos esta tecnología para que cualquier usuario la utilice según sus necesidades.

La importancia de elegir NFC como plataforma para la comunicación es que el mismo tiene distintas aplicaciones, por lo tanto, con el desarrollo necesario, es posible crear un sistema aún más robusto, que cumpla distintas tareas según las necesidades del cliente, además del sistema de pago que se requiere implementar.

El sistema operativo ANDROID, como se mostró anteriormente, es el más expandido actualmente, por lo cual, desarrollar una aplicación para el mismo asegura

que una gran cantidad de usuarios pueden disfrutar del servicio, y de los beneficios que ofrece el sistema.

Por lo tanto, el sistema desarrollado ofrece una alternativa viable para la problemática de los pagos electrónicos que existen actualmente, además de aprovechar las herramientas que se tienen a disposición para que interactúen, creando una solución sólida y confiable que puede ser tomada en cuenta para la implementación en locales comerciales.

CAPÍTULO II

Marco Teórico

Este capítulo está constituido por todas las tecnologías y herramientas que se utilizaron para desarrollar este Trabajo Especial de Grado. A través del contenido de este marco teórico se pudieron evaluar tanto las características como las ventajas y desventajas de: Entornos de desarrollo, Protocolos, Servidores *Web*, lenguajes de programación entre otras ciencias aplicadas, presentadas a continuación.

II.1 NFC (*Near Field Communication*).

Esta tecnología permite interacciones simples, seguras y bidireccionales entre dispositivos electrónicos; permitiendo a los usuarios realizar transacciones sin contacto, acceder a contenidos digitales y conectar dispositivos electrónicos con un solo toque. NFC complementa muchas tecnologías inalámbricas de consumo popular, y es compatible con la infraestructura existente de tarjetas sin contacto. NFC permite una velocidad máxima de comunicación de 424 kbps. NFC solo permite que los dispositivos estén separados por menos de 10 cm. (Ruiz, 2011)

II.1.1 Descripción General.

NFC es una tecnología inalámbrica de corto alcance basada en RFID, que permite realizar una comunicación simple, segura e intuitiva entre dispositivos electrónicos que se encuentran a una distancia de hasta de 10 centímetros en algunos casos.

Dentro de la forma de trabajo de NFC, un dispositivo genera una onda de radio de baja frecuencia que opera a 13,56 MHz. Cuando otro dispositivo NFC se acerca lo suficiente para ponerse en contacto, se genera un “acoplamiento magnético

inductivo", por medio del cual se puede realizar una transferencia de energía y de datos entre los dispositivos. (Ruiz, 2011)

Este acoplamiento inductivo funciona sólo para distancias cortas y es por esto que el uso de este tipo de acoplamiento es la principal diferencia entre NFC y otras tecnologías inalámbricas como BLUETOOTH y WiFi,

Entre los beneficios más importantes que se pueden señalar del uso y desarrollo de esta tecnología, se destacan los siguientes:

- Mejora la manipulación y la experiencia del usuario.
- Fácil acceso a servicios y contenidos ofrecidos por objetos físicos.
- Se puede compartir información digital entre dos dispositivos con tan sólo acercar el uno al otro.
- Seguridad, por ser una tecnología de corto alcance.

II.1.2 Definición Estándar de datos NFC

Para poder compartir datos entre dispositivos NFC o entre un dispositivo y las etiquetas NFC, se ha creado un estándar en el que se registra un formato común. (NFC Forum, 2012)

- **NFC Data Exchange Format (NDEF)**: Especifica un formato común y compacto para el intercambio de datos.
- **NFC Record Type Definition (RTD)**: Especifica tipos de registros estándar que pueden ser enviados en los mensajes intercambiados entre los dispositivos NFC.
- **Smart Poster RTD**: Para posters que incorporen etiquetas con datos (URLs, SMSs o números de teléfono).
- **Text RTD**: Para registros que solo contienen texto.

- **Uniform Resource Identifier (URI) RTD**: Para registros que se refieren a un recurso de Internet.

II.1.3 Especificaciones Técnicas.

Como se mencionó anteriormente, NFC opera dentro de la banda ISM (*Industrial, Scientific and Medical*) de radio frecuencia de 13,56 MHz disponible globalmente sin restricción y sin necesidad de licencia para su uso, con un ancho de banda de casi 2 MHz.

NFC es una tecnología de plataforma abierta estandarizada en la ISO/IEC 18092 y la ECMA-340. Estos estándares especifican los esquemas de modulación, codificación, velocidades de transferencia y formato de la trama de la interfaz RF de dispositivos NFC, 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 definen el protocolo de transporte, incluyendo los métodos de activación de protocolo y de intercambio de datos. (Ruiz, 2011)

La distancia de trabajo con antenas compactas estándar es aproximadamente 20 cm, aunque generalmente es efectivo cercano a los 10 cm. Las velocidades de transmisión que soporta esta tecnología son de 106, 212 y 424 kbits/s.

II.1.4 Modos de Funcionamiento.

Un dispositivo NFC con suministro interno de energía es denominado activo, y sin suministro interno de energía, como una etiqueta, es considerado pasivo. El acoplamiento inducido causa que un dispositivo pasivo absorba energía de uno activo cuando se acercan lo suficiente. Una vez encendido, el dispositivo pasivo puede comunicarse e intercambiar datos con el otro dispositivo. (Ruiz, 2011)

La habilidad de funcionar en los dos modos, activo o pasivo, hace que los dispositivos NFC sean únicos dentro de otras tecnologías de comunicación sin contacto. Esto posibilita a los dispositivos a actuar como tarjetas sin contacto o como lectores. Por tanto, un teléfono móvil habilitado con NFC puede ser usado por ejemplo, para enviar información de pago a un lector y realizar una compra o para leer información de una valla o poster publicitario con una etiqueta adherida. La descripción de ambos modos de funcionamiento se detalla a continuación:

- **Pasivo:** Sólo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.
- **Activo:** Ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos. Ambos dispositivos necesitan energía para funcionar.

Cualquier dispositivo electrónico con NFC (excepto una etiqueta NFC) puede operar de las dos formas. (Ruiz, 2011)

II.1.5 Comunicación NFC.

La comunicación NFC consta de cinco fases, estando siempre presentes. Estas etapas son:

- **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 del dispositivo, 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 ya está realizada exitosamente la comunicación y ya se puede realizar el intercambio de datos.
- **Confirmación**: El dispositivo receptor confirma el establecimiento de la comunicación y la transferencia de datos.

II.1.6 Modalidades de Uso.

Se puede decir que NFC es una tecnología única ya que puede trabajar en tres diferentes configuraciones lo que la hace más adaptable y eficiente que otras. (Ruiz, 2011)

La tecnología NFC puede trabajar en tres distintas configuraciones:

- En el modo Lector/ Escritor un dispositivo NFC lee información de una etiqueta NFC.
- En el modo Emulación de Tarjeta el dispositivo NFC se comporta como una etiqueta NFC y es leído por otro dispositivo.
- En el modo Peer-to-peer dos dispositivos se intercambian información de igual a igual.

II.1.6.1 Modo Lector/Escritor.

En este modo, el dispositivo NFC es capaz de escribir y leer los cuatro tipos de etiquetas especificados por el NFC *Forum*, que se detallan en el siguiente apartado.

En la torre de protocolos que podemos observar en la figura 2, se muestra que se puede trabajar con cualquier tipo de etiqueta, y utilizando o no, el modo de definición de datos NDEF.

En este modo, cuando el usuario toca con su dispositivo una tarjeta o etiqueta NFC, se transfiere una pequeña cantidad de información al dispositivo NFC. Esta información puede ser un texto, una dirección de una página *web* o un número de teléfono. (Ruiz, 2011)

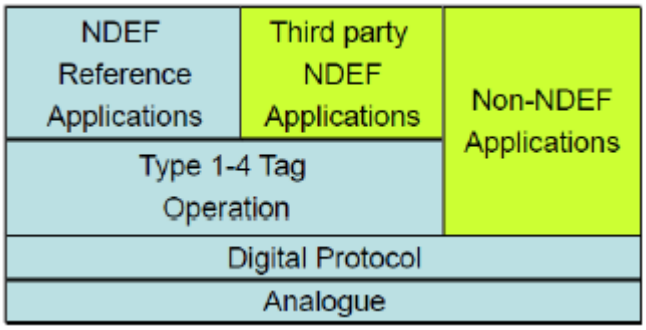


Figura 2. Torre de protocolos para el modo lectura/escritura

Fuente: (Ruiz, 2011).

II.1.6.2 Modo de Emulación de Tarjeta Inteligente NFC.

Este modo se utiliza para que el dispositivo NFC actúe como una etiqueta o una tarjeta inteligente, apareciendo ante un lector externo como si se tratase de una tarjeta sin contacto.

Con esta configuración también se puede utilizar las características de seguridad avanzadas del elemento seguro, siendo útil para, por ejemplo, transacciones bancarias, o gestión de entradas y accesos. (Ruiz, 2011)

A continuación se muestra la torre de protocolos de este modo de funcionamiento, en el que se releva la gestión de la lectura de datos para el nivel de aplicación.

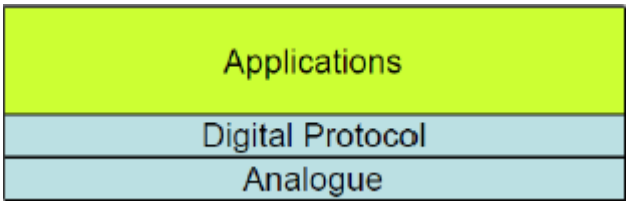


Figura 3. Torre de protocolos para el modo emulación de tarjeta inteligente NFC

Fuente: (Ruiz, 2011).

II.1.6.3 Modo de Comunicación Peer-to-Peer.

Este modo sirve para el intercambio de pequeñas cantidades de datos utilizando el mismo protocolo de NFC.

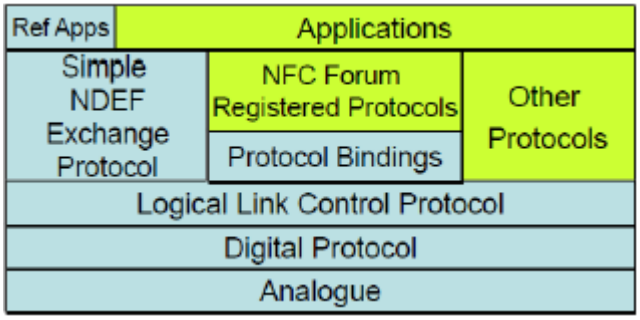


Figura 4. Torre de protocolos para el modo de comunicación Peer-to-peer.

Fuente: (Ruiz, 2011).

En la Figura 4 se muestra la torre de protocolos de este modo de funcionamiento. NFC utiliza a nivel de la capa de enlace el protocolo de control de enlace lógico (LLCP), el mismo que es usado para la activación, supervisión y desactivación de la comunicación.

El modo de transferencia se hace de modo asincrónico balanceado, es decir que cualquier dispositivo puede iniciar la transmisión sin permiso de la otra.

El protocolo de intercambio simple NDEF se utiliza para enviar mensajes con el formato NDEF en este modo.

Los protocolos de conexión "*Bindings*" proporcionan enlaces estándar para protocolos NFC registrados y permite su uso interoperable.

Los protocolos NFC registrados son aquellos para los que el *NFC Forum* ha definido un enlace con la capa LLCP (Protocolo de Control de Enlace Lógico) , por ejemplo IP, OBEX (intercambio de Objetos binarios), etc. (Ruiz, 2011)

II.1.7 Etiquetas (*tags*).

El NFC *Forum* ha creado un juego de cuatro formatos de etiquetas. Estos formatos están basados en el ISO 14443 Tipo A y B (Estándares internacionales para tarjetas inteligentes sin contacto) y FELICA (derivado el ISO 18092, modo de comunicación pasivo). (Ruiz, 2011)

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

- **Etiqueta Tipo 1**: Se basa en la norma ISO14443A. Las etiquetas pueden ser de lectura y escritura, aunque los usuarios pueden configurarlas sólo para lectura. La memoria disponible va desde *96* Bytes hasta a 2 Kbytes; la velocidad de comunicación es de 106 Kbps.

- **Etiqueta Tipo 2**: Se basa en la norma ISO14443A. Puede ser de lectura y escritura, aunque los usuarios pueden configurarla sólo para lectura. La memoria disponible va desde *48* Bytes hasta 2 Kbytes; la velocidad de comunicación es de 106 Kbps.
- **Etiqueta Tipo 3**: Se basa en la *Japanese Industrial Standard (JIS) X 6319-4*, también conocido como FELICA. Se encuentran ya configuradas desde la fabricación, ya sea para escritura, o sólo para lectura. La memoria disponible es variable, pero teóricamente el límite de memoria es 1 MB por servicio; la velocidad de comunicación es de 212 Kbps o 424 Kbps.
- **Etiqueta Tipo 4**: Es plenamente compatible con ISO14443A y B. Se encuentran ya configuradas desde la fabricación, ya sea para lectura y escritura, o de sólo lectura. La memoria disponible es variable, de hasta 32 KB por servicio; la velocidad de comunicación es de hasta 424 Kbps.

II.1.8 Seguridad en NFC.

NFC por sí sola no proporciona comunicaciones seguras, no ofrece protección contra los que se dedican a escuchar comunicaciones y es también vulnerable a modificación de datos. Las aplicaciones deben usar protocolos criptográficos para establecer un canal seguro.

Sin embargo, esto se contrarresta con la distancia de operación necesaria de NFC, 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. (Ruiz, 2011)

Además, al ser de corto alcance, evita los errores de comunicación, asegura una mayor eficacia en la transmisión de datos y hace que el posible atacante deba estar dentro de ese corto rango, en el que el usuario podría darse cuenta fácilmente.

II.2 Comparación de Tecnologías.

En la tabla 1 se presentan las características de NFC y BLUETOOTH, demostrando que NFC es la tecnología indicada para la aplicación de la transacción debido a que se adapta adecuadamente a los requerimientos necesarios.

	NFC	Bluetooth
Compatibilidad RFID	ISO 18000-3	no
Estándar	ISO/IEC	Bluetooth SIG
Estándar de red	ISO 13157, etc.	IEEE 802.15.1
Tipo de red	Punto a punto	Punto a multipunto
Rango	~ 10 cm	~10 m (clase 2) ~100m (clase 3)
Frecuencia	13.56 MHz	2.4-2.5 GHz
Tasa de transferencia	424 Kbit/s	2.1 Mbit/s (v2.1) (mayor a 721 Kbit/s)
Tiempo de inicialización	< 0.1 s	~ 6 s
Seguridad	Sí, a nivel hardware y protocolo	Sí, a nivel protocolo
Modos de comunicación	Activo-pasivo Activo-activo	Activo-activo

Tabla 1. Comparativa entre las tecnologías NFC y BLUETOOTH.

Fuente: (Chavarría , 2011)

La tasa de transferencia máxima de NFC (424 kbit/s), es menor que la de BLUETOOTH v2.1 (2.1 Mbit/s), cosa que no afecta la transacciones de pago puesto que los paquetes de datos que se envían son pequeños. El rango de trabajo en NFC es menor a 20cm, lo que reduce la probabilidad de intercepciones fraudulentas, puesto que el rango es corto, por lo que no hay oportunidad que un tercer dispositivo intercepte la comunicación, además de ser una comunicación punto a punto, por estas características NFC garantiza la seguridad a nivel de *hardware*. En contraste con BLUETOOTH, NFC es compatible con infraestructuras RFID, lo que ofrece posibilidades de escalabilidad y requiere menor potencia.

El tiempo de inicialización es otro punto fuerte de NFC para ser una plataforma que ofrezca soluciones para los pagos electrónicos, puesto que este tiempo en NFC es menor que BLUETOOTH, garantiza que el envío de la información para la

transacción sea rápida y ofrezca una ventaja sobre las formas de pago electrónicas ya existentes.

II.3 Lenguajes de Programación.

II.3.1 JAVA.

JAVA es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. JAVA es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de *web,* con unos 10 millones de usuarios reportados. (Azpitarte, Jordá, & Llinares, 2009)

Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones de JAVA son generalmente compiladas a *bytecode* (clase JAVA) que puede ejecutarse en cualquier máquina virtual JAVA (JVM) sin importar la arquitectura de la computadora subyacente.

JAVA constituye una plataforma completa para el desarrollo de *software*, debido a que posee una gran biblioteca con facilidades para el desarrollador y aplicaciones con numerosos códigos reutilizables y un entorno de programación que proporciona servicios tales como seguridad y portabilidad entre sistemas operativos. (Deitel & Deitel, 2008)

II.3.1.1 Características de JAVA.

- Arquitectura Neutral

A diferencia de lenguajes que le precedieron como C, C++ o Pascal, el compilador de JAVA no traduce el código fuente a lenguaje máquina, al finalizar la compilación es generado un código que es llamado *Bytecode*; para interpretar este código en los dispositivos finales, es necesario la instalación previa de la Máquina Virtual de JAVA (JVM), esto permite que JAVA sea un lenguaje independiente de la arquitectura del dispositivo. (Joyanes & Fernández, 2003)

La máquina Virtual de JAVA más el conjunto de clases necesarias para ejecutar los programas conforma el entorno de ejecución de JAVA o JRE (*JAVA Runtime Environment*).

Como se mencionó anteriormente para ejecutar el código JAVA solo es necesario la instalación del JRE; si se desea escribir y compilar código fuente, será necesario la instalación del Kit de Desarrollo de JAVA (JDK). En la figura 5 se muestra la jerarquía simplificada del *software* que constituye la plataforma de JAVA.

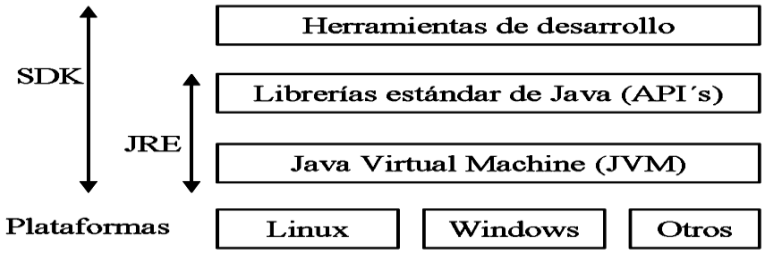


Figura 5. Plataforma JAVA

Fuente: (Azpitarte R. , y otros, 2009)

- Orientado a objetos

La programación orientada a objetos es la base de JAVA y representa una forma de organización del conocimiento, en la que las entidades centrales son los objetos. En un objeto se agrupan una serie de datos con una relación lógica entre ellos: estados y acciones, a los estados se les denomina Variables de Instancia, y las acciones que vendrían a ser las rutinas necesarias para la manipulación de dichos estados, se les conoce como métodos.

La programación orientada a objetos modela el mundo real, debido a que poseen al igual que cualquier objeto real ciertos atributos que lo identifican. JAVA es denominado orientado a objetos debido a que la programación se centra en la creación, manipulación y construcción de objetos.

Cuando se desarrolla un programa con el uso de programación orientada a objetos, no se definen verdaderos objetos, sino clases. Una clase se puede definir como una plantilla con el que pueden construir varios objetos con características similares. En las clases pueden existir métodos especiales. (Azpitarte, Jordá, & Llinares, 2009).

II.3.2 ANDROID.

El lanzamiento de ANDROID como sistema operativo para teléfonos móviles ha causado un gran impacto en nuestra sociedad ya que es un *software* libre que les otorga a los usuarios la posibilidad de crear aplicaciones que se adapten a sus rutinas diarias y por esta razón esta nueva herramienta ha tenido una excelente aceptación en el mercado, tanto de los usuarios como de las industrias. Convirtiéndose de esta manera en uno de los sistemas preferidos por el público en general y una gran

competencia frente a otras plataformas como IPHONE, WINDOWS PHONE o BLACKBERRY. (Dornin, Mednieks, Meike, & Nakamura, 2012)

Con ANDROID se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wi-Fi, BLUETOOTH, NFC, aplicaciones ofimáticas, videojuegos, etc.), así como poder crear aplicaciones que sean verdaderamente portables, reutilizables y de rápido desarrollo.

II.3.2.1 Característica de ANDROID.

ANDROID es un sistema operativo para dispositivos móviles como teléfonos inteligentes y tabletas basado en el núcleo LINUX. Es desarrollado por la OPEN HANDSET ALLIANCE, la cual es liderada por GOOGLE, usando diversos conjuntos de herramientas de *software* de código abierto para dispositivos móviles. (Tomas, 2011)

Fue construido para permitir a los desarrolladores la creación de aplicaciones móviles que aprovechan al máximo el uso de todas las herramientas que un dispositivo como este puede ofrecer.

Implementa una arquitectura en la que cualquier aplicación puede obtener acceso a las capacidades del teléfono móvil. Por ejemplo, una aplicación puede llamar una o varias de las funcionalidades básicas de los dispositivos móviles, tales como realizar llamadas, enviar mensajes de texto, o utilizar la cámara, facilitando a los desarrolladores crear experiencias más ricas y con más coherencia para los usuarios.

Está construido sobre el KERNEL de LINUX. Además, se utiliza una máquina personalizada virtual que fue diseñada para optimizarlos recursos de memoria y de *hardware* en un entorno móvil. ANDROID es de código abierto, y además puede ser libremente ampliado para incorporar nuevas tecnologías de vanguardia que van surgiendo. La plataforma continuará evolucionando a medida que la comunidad de desarrolladores trabajando juntos puedan crear aplicaciones móviles innovadoras.

La siguiente lista destaca las características funcionales más importantes de ANDROID:

- Amplia variedad de diseños (VGA, librerías de gráficos 2D y 3D, etc.).
- Almacenamiento de datos en BBDD SQLite.
- Conectividad (GSM/EDGE, CDMA, EV-DO, UMTS, BLUETOOTH, NFC y Wi-Fi).
- Mensajería (SMS y MMS).
- Navegador *Web*.
- Máquina virtual de JAVA.
- Las aplicaciones escritas en JAVA pueden ser compiladas y ejecutadas en la máquina virtual DALVIK, la cual es una máquina virtual especialmente diseñada para uso en dispositivos móviles.
- Soporte de formatos (MPEG-4, H.264, MP3, AAC, OGG, AMR, JPEG, PNG, GIF).
- Soporte para *hardware* adicional (cámaras de vídeo, pantallas táctiles, GPS, acelerómetros, etc.).
- Entorno de desarrollo (emulador, herramientas de depuración, perfiles de memoria y funcionamiento, *plugin* para ECLIPSE IDE, ANDROID STUDIO v1.0).

Además de las características más puramente funcionales, existen otro tipo de características que es necesario nombrar y hacen de ANDROID un sistema operativo especial.

- **Plataforma realmente abierta**: Es una plataforma de desarrollo libre basada en LINUX y de código libre.
- **Portabilidad asegurada**: Al desarrollar las aplicaciones en JAVA, y gracias al concepto de máquina virtual, las aplicaciones podrán ser ejecutadas en gran variedad de dispositivos tanto presentes como futuros.
- **Arquitectura basada en componentes inspirados en internet**: Por ejemplo, las interfaces se hacen en XML, lo que permite el uso de una misma interfaz en dispositivos de pantallas dispares.
- Filosofía de dispositivo siempre conectado a internet.
- **Gran cantidad de servicios incorporados:** Reconocimiento y síntesis de voz, localización basada en GPS y torres de comunicación, potentes bases de datos, NFC, etc.
- **Alto nivel de seguridad:** Los programas se encuentran aislados unos de otros. Cada aplicación dispone de una serie de permisos que limitan su rango de actuación.
- **Optimización para baja potencia y baja memoria:** Por ejemplo el uso de la máquina virtual DALVIK.

II.5.2.2 Arquitectura.

A continuación se dará una visión global por capas de la arquitectura empleada en ANDROID. Cada una de estas capas utiliza servicios ofrecidos por las capas anteriores, y ofrece a su vez los suyos propios a las capas de niveles superiores. (Aranaz Tudela , 2009)

La capa más inmediata es la que corresponde al núcleo o KERNEL de ANDROID, el cual está constituido por el núcleo de LINUX 2.6. Esta capa contiene los controladores necesarios para que cualquier *hardware* pueda ser utilizado mediante las llamadas correspondientes. Esto amerita que cada vez que el fabricante incluya un nuevo elemento de *hardware*, debe crear las librerías de control o *drivers* necesarios dentro del núcleo o KERNEL para que pueda ser utilizado desde ANDROID.

La elección de LINUX 2.6 se ha debido principalmente a dos razones: la primera, su naturaleza de código abierto y libre se ajusta al tipo de distribución que se buscaba para ANDROID; la segunda es que este KERNEL de LINUX incluye de por sí numerosos *drivers*, además de contemplar la gestión de memoria, gestión de procesos, módulos de seguridad, comunicación en red y otras muchas responsabilidades propias de un sistema operativo. (Aranaz Tudela , 2009)

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

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

- La librería *libc* incluye todas las cabeceras y funciones según el estándar del lenguaje C. Todas las demás librerías se definen en este lenguaje.
- La librería *Surface Manager* es la encargada de componer los diferentes elementos de navegación de pantalla. Gestiona también las ventanas pertenecientes a las distintas aplicaciones activas en cada momento.
- *OpenGL/SL* y *SGL* representan las librerías gráficas y, por tanto, sustentan la capacidad gráfica de ANDROID. *OpenGL/SL* maneja gráficos en 3D y

permite utilizar, en caso de que esté disponible en el propio dispositivo móvil, el *hardware* encargado de proporcionar gráficos 3D. Por otro lado, SGL proporciona gráficos en 2D, por lo que será la librería más habitualmente utilizada por la mayoría de las aplicaciones. Una característica importante de la capacidad gráfica de ANDROID es que es posible desarrollar aplicaciones que combinen gráficos en 3D y 2D.

- La librería *Media Libraries* proporciona todos los códec necesarios para el contenido multimedia soportado en ANDROID (vídeo, audio, imágenes estáticas y animadas, etc.)
- *FreeType*, permite trabajar de forma rápida y sencilla con distintos tipos de fuentes.
- La librería SSL posibilita la utilización de dicho protocolo para establecer comunicaciones seguras.
- A través de la librería SQLite, ANDROID ofrece la creación y gestión de bases de datos relacionales, pudiendo transformar estructuras de datos en objetos fáciles de manejar por las aplicaciones.
- La librería *WebKit* proporciona un motor para las aplicaciones de tipo navegador, y forma el núcleo del actual navegador incluido por defecto en la plataforma ANDROID.

Al mismo nivel que las librerías de ANDROID se sitúa el entorno de ejecución. Éste lo constituyen las *Core Libraries*, que son librerías con multitud de clases de JAVA, y la máquina virtual DALVIK. (Vacas, 2014)

Los dos últimos niveles de la arquitectura de ANDROID están escritos enteramente en JAVA. El *framework* de aplicaciones representa fundamentalmente el conjunto de herramientas de desarrollo de cualquier aplicación.

Toda aplicación que se desarrolle para ANDROID, ya sean las propias del dispositivo, las desarrolladas por GOOGLE o terceras compañías, o incluso las que el propio usuario cree, utilizan el mismo conjunto de API's y el mismo *framework*, representado por este nivel.

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

- ***Activity Manager***: importante conjunto de la API que gestiona el ciclo de vida de las aplicaciones en ANDROID (del que se hablará más adelante).
- ***Window Manager***: gestiona las ventanas de las aplicaciones y utiliza la librería ya vista *Surface Manager*.
- ***Telephone Manager***: incluye todas las API's vinculadas a las funcionalidades propias del teléfono (llamadas, mensajes, etc.).
- ***Content Providers***: permite a cualquier aplicación compartir sus datos con las demás aplicaciones de ANDROID. Por ejemplo, gracias a esta API la información de contactos, agenda, mensajes, etc. será accesible para otras aplicaciones.
- ***View System***: proporciona un gran número de elementos para poder construir interfaces de usuario (GUI), como listas, mosaicos, botones, *check-boxes*, tamaño de ventanas, control de las interfaces mediante tacto o teclado, etc.

Incluye también algunas vistas estándar para las funcionalidades más frecuentes:

- ***Location Manager*:** posibilita a las aplicaciones la obtención de información de localización y posicionamiento.
- ***Notification Manager*:** mediante el cual las aplicaciones, usando un mismo formato, comunican al usuario eventos que ocurran durante su ejecución: una

llamada entrante, un mensaje recibido, conexión Wi-Fi disponible, ubicación en un punto determinado, etc. Si llevan asociada alguna acción, en ANDROID denominada *Intent*, (por ejemplo, atender una llamada recibida) ésta se activa mediante un simple clic.

- ***XMPP Service*:** colección de API's para utilizar este protocolo de intercambio de mensajes basado en XML.

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

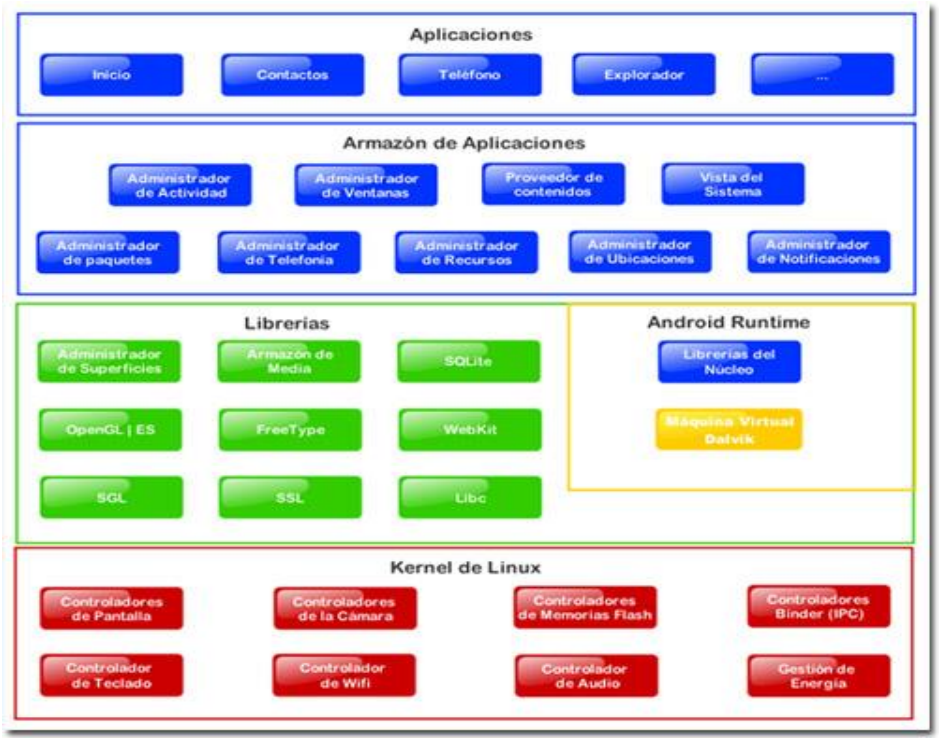


Figura 6. Arquitectura del sistema operativo ANDROID.

Fuente: (Aranaz Tudela , 2009)

II.3.3 XML (*Extensible Markup Language*).

XML no es más que un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. (Melián Montalvo, 2008)

Este lenguaje es abierto, derivado del SGML, optimizado para su uso en la *Web*, y que va a permitirnos describir el sentido o la semántica de los datos.

II.5.3.1 Características de XML.

- Es un estándar internacionalmente reconocido.
- No pertenece a ninguna compañía, y su utilización es libre.
- Permitirá la utilización efectiva de Internet en diferentes alfabetos, por personas con minusvalías físicas, y en diferentes *hardwares* (teléfonos celulares, PDAs, terminales Braille, etc).

II.5.3.2 Ventajas de XML.

- Fácilmente procesable tanto por humanos como por *software*.
- Separa radicalmente la información o el contenido de su presentación o formato.
- Diseñado para ser utilizado en cualquier lenguaje o alfabeto.
- Su análisis sintáctico es fácil debido a las estrictas reglas que rigen la composición de un documento.
- Estructura Jerárquica.
- El número de marcas es ilimitado.
- Poderosos enlaces (XLL).

II.5.3.3 XML en ANDROID.

En ANDROID se utilizan archivos XML para la declaración de *layouts* y otros elementos de los que hará uso una aplicación para su correcto funcionamiento. De esta forma, características como la instanciación dinámica se reserva para escenarios más complejos en dónde los elementos de UI (*User Interface*) no se conocen en tiempo de compilación (por ejemplo seleccionar una opción de un combo *box* que se ha llenado de acuerdo a datos consultados en Internet).

En ANDROID los archivos de *layout* codificados en XML se consideran recursos y se guardan dentro del directorio *res/layout* del proyecto. (Condesa, 2011)

Cada archivo XML está formado por un árbol de elementos que especifican la manera en la que los elementos de la UI (*User Interface*) y contenedores se acomodarán para definir la parte visual de un objeto *View*. Los atributos de cada elemento en el XML se llaman propiedades y describen cómo es que deben verse los elementos y cómo debe comportarse un contenedor.

II.5.3.4 *Layouts*

En la programación ANDROID todos los *layouts* se pueden definir vía XML pero también directamente desde código JAVA. Las razones por las cuales mayormente se escoge XML es que cuenta con herramientas visuales para la creación de *layouts* y además es mucho más fácil.

Otros de los grandes beneficios, es el hecho de tener separado el código de diseño y el código de funcionalidad, esto aplica por ejemplo, cuando se necesita editar la interfaz de usuario sin afectar el código de funcionalidad, garantizando mayor orden al programador. (Condesa, 2011)

II.3.4 PHP (*Hypertext Preprocessor*)

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo *web* y que puede ser incrustado en HTML. (php, s.f.)

Está enfocado principalmente a la programación de *scripts* del lado del servidor, por lo que permite realizar funciones similares a programas que operen con CGI (*Common Gateway Interface*), como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies.

Existen principalmente tres campos principales donde se usan scripts de PHP:

- ***Scripts* del lado del servidor:** Este es el campo más tradicional y el foco principal. Son necesarias tres cosas para que esto funcione. El analizador de PHP (módulo CGI o servidor), un servidor *web* y un navegador *web*. Es necesario ejecutar el servidor con una instalación de PHP conectada. Se puede acceder al resultado del programa de PHP con un navegador, viendo la página de PHP a través del servidor.
- ***Scripts* desde la línea de comandos:** Se puede crear un *script* de PHP y ejecutarlo sin necesidad de un servidor o navegador. Solamente es necesario el analizador de PHP para utilizarlo de esta manera. Este tipo de uso es ideal para *scripts* que se ejecuten con regularidad empleando Cron (en UNIX o LINUX) o el Planificador de tareas (en WINDOWS). Estos *scripts* también pueden usarse para tareas simples de procesamiento de texto.
- **Escribir aplicaciones de escritorio:** Probablemente PHP no sea el lenguaje más apropiado para crear aplicaciones de escritorio con una interfaz gráfica de usuario, pero si se conoce bien PHP, y se quisiera utilizar algunas características avanzadas de PHP en aplicaciones del lado del cliente, se puede utilizar PHP-GTK (*GIMP Toolkit*) para escribir dichos programas.

También es posible de esta manera escribir aplicaciones independientes de una plataforma. PHP-GTK (*GIMP Toolkit*) es una extensión de PHP, no disponible en la distribución principal.

PHP puede emplearse en todos los sistemas operativos principales, incluyendo LINUX, muchas variantes de UNIX (incluyendo HP-UX, SOLARIS y OpenBSD), MICROSOFT WINDOWS, Mac OS X, RISC OS y probablemente otros más. PHP admite la mayoría de servidores *web* de hoy en día, incluyendo APACHE, IIS, y muchos otros. Esto incluye cualquier servidor *web* que pueda utilizar el binario de PHP FastCGI, como *lighttpd* y *nginx*. PHP funciona tanto como módulo como procesador de CGI (*Common Gateway Interface*).

Una de las características más potentes y destacables de PHP es su soporte para un amplio abanico de bases de datos. Escribir una página *web* con acceso a una base de datos es increíblemente simple utilizando una de las extensiones específicas de bases de datos (p.ej., para MYSQL), o utilizar una capa de abstracción como PDO (*PHP Data Objects*), o conectarse a cualquier base de datos que admita el estándar de Conexión Abierta a Bases de Datos por medio de la extensión ODBC (*Open DataBase Connectivity*). (php, s.f.)

II.5 API (*Application Programming Interface*).

Son un conjunto de reglas que definen como se accede a un servicio de un ordenador por medio de un programa de aplicación.

II.5.1 API JAVA.

El API JAVA es una Interfaz de Programación de Aplicaciones provista por los creadores del lenguaje JAVA, y que da a los programadores los medios para desarrollar aplicaciones JAVA.

Como el lenguaje JAVA es un Lenguaje Orientado a Objetos, la API de JAVA provee de un conjunto de clases utilitarias para efectuar toda clase de tareas necesarias dentro de un programa. (Muñoz, 2010)

II.5.2 API ANDROID.

A medida que se desarrolla una aplicación en ANDROID, es útil comprender el enfoque general de la plataforma para la gestión del cambio de la API. También es importante entender el identificador del nivel de la API y el papel que desempeña en la consecución de la compatibilidad de la aplicación con los dispositivos en los que puedan estar instalados.

El nivel API es un valor entero que identifica de forma exclusiva la revisión de la API, marco que ofrece una versión de la plataforma ANDROID. (Camacho, 2012)

La plataforma ANDROID ofrece una API de marco, en donde las aplicaciones pueden utilizarlas para interactuar con el sistema ANDROID subyacente. La API de marco consiste en:

- Un conjunto básico de paquetes y clases.
- Un conjunto de elementos y atributos XML para la declaración de un archivo de manifiesto.
- Un conjunto de elementos y atributos XML para la declaración y el acceso a los recursos.
- Un conjunto de Intenciones.
- Un conjunto de permisos que pueden solicitar aplicaciones, así como refuerzos de permiso incluido en el sistema

Cada versión sucesiva de la plataforma ANDROID puede incluir actualizaciones de la aplicación de ANDROID API de marco de trabajo que se ofrece.

La tabla 2 especifica el nivel de API con el apoyo de cada versión de la plataforma ANDROID.

VERSIÒN DE LA PLATAFORMA	API de nivel	VERSION_CODE	Notas
Android 4.0.3	15	ICE_CREAM_SANDWICH_MR1	Aspectos destacados de la plataforma
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	
Android 3.2	13	HONEYCOMB_MR2	
Android 3.1.x	12	HONEYCOMB_MR1	Aspectos destacados de la plataforma
Android 3.0.x	11	PANAL	Aspectos destacados de la plataforma
Android 2.3.4 para Android 2.3.3	10	GINGERBREAD_MR1	Aspectos destacados de la plataforma
Android 2.3.2 para Android 2.3.1 Android 2.3	9	GINGERBREAD	
Android 2.2.x	8	FROYO	Aspectos destacados de la plataforma
Android 2.1.x	7	ECLAIR_MR1	Aspectos destacados de la plataforma
Android 2.0.1	6	ECLAIR_0_1	
Android 2.0	5	ECLAIR	
Android 1.6	4	DONUT	Aspectos destacados de la plataforma
Android 1.5	3	MAGDALENA	Aspectos destacados de la plataforma
Android 1.1	2	BASE_1_1	
Android 1.0	1	BASE	

Tabla 2. Historial de versiones ANDROID.

Fuente: (Tomas, 2011)

El identificador de nivel API desempeña un papel clave para garantizar la mejor experiencia posible para los usuarios y desarrolladores de aplicaciones:

- Permite que la plataforma ANDROID describa el marco máximo de la revisión de la API que soporta.
- Se permite a las aplicaciones describir la revisión de la API marco de trabajo que requieren.

- Permite negociar la instalación de aplicaciones en el dispositivo del usuario, de tal manera que la versión de las aplicaciones incompatibles no están instalados.
- Cada versión de la plataforma ANDROID almacena su identificador de nivel API internamente, en el sistema ANDROID.

II.5.3 APIs y Sistemas.

En la comunidad de desarrollo JAVA, se suelen identificar cada una de las diferentes bibliotecas existentes como API's JAVA. Cuando se construye un sistema informático este suele emplear diversas API`s. (Muñoz, 2010)

II.5.4 API y Protocolos.

Una API puede ser también una implementación de un protocolo. En general, la diferencia entre una API y un protocolo es que el protocolo define una manera estándar a las peticiones de cambio y las respuestas basados en un transporte común y ponerse de acuerdo sobre un conjunto de datos o el formato de intercambio de mensajes, mientras que una API se implementa normalmente como una biblioteca para ser utilizado directamente. (Muñoz, 2010)

II.5.5 Ejemplos de API.

- ASPI (*Advanced SCSI Programming Interface*) para SCSI (*Small Computers System Interface*) interfaz de dispositivo
- De CARBON y el COCOA para el MACINTOSH
- DirectX de MICROSOFT WINDOWS
- EHLLAPI
- API de JAVA

- ODBC (*Open DataBase Connectivity*) para MICROSOFT WINDOWS
- OpenAL (*Open Audio Library*) multi-plataforma de sonido API
- OpenCL (*Open Computing Language*) API multiplataforma para la computación de propósito general para las CPU y GPU
- OpenGL(*Open Graphics Library*) multi-plataforma API de gráficos
- OpenMP (*Open Multi-Processing*) API que soporta multi-plataforma de programación de memoria compartida de multiproceso en C, C + + y FORTRAN en muchas arquitecturas, incluyendo UNIX y plataformas de MICROSOFT WINDOWS.
- *Simple DirectMedia Layer* (SDL)
- TALEND para integrar la gestión de datos con el BPM (*Business Process Management*) de BONITA *Open Solution*.

II.6 IDE (*Integrated Development Environment*).

Un Entorno de desarrollo integrado es una aplicación de *software*, que proporciona servicios integrales para facilitarle al programador de computadora el desarrollo de *software*. Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDEs tienen auto-completado inteligente de código.

A continuación se darán a conocer los IDEs que fueron utilizados en el proceso de aprendizaje y fueron seleccionados para la culminación de la aplicación los cuales fueron: ANDROID STUDIO v1.2 y NETBEANS.

II.6.1 NETBEANS.

NETBEANS es un ambiente de desarrollo integrado escrito en lenguaje JAVA, el cual puede ser usado como un entorno genérico para construir cualquier tipo de

aplicaciones. NETBEANS IDE es un producto libre y sin restricciones de uso, fundado por SUN MICROSYSTEMS en Junio del año 2000, quienes fueron los principales patrocinadores del proyecto hasta Enero del año 2010. (NetBeans, 2013)

Soporta muchos lenguajes, desde JAVA, C/C++, XML, HTML, PHP, GROOVY, JAVADOC, JAVASCRIPT y JSP. También proporciona diferentes vistas de los datos a partir de múltiples ventanas y módulos útiles para la creación y gestión de aplicaciones, lo que permite que las aplicaciones basadas en NETBEANS puedan ser comprendidas por otros desarrolladores rápidamente.

II.6.2 ANDROID STUDIO.

ANDROID STUDIO es un entorno de desarrollo integrado (IDE), basado en INTELLIJ IDEA de la compañía JETBRAINS, que proporciona varias mejoras con respecto al *plugin* ADT (*ANDROID Developer Tools*) para ECLIPSE. ANDROID STUDIO utiliza una licencia de *software* libre APACHE 2.0, está programado en JAVA y es multiplataforma. (Androcode, 2013)

Fue presentado por GOOGLE el 16 de mayo del 2013 en el congreso de desarrolladores GOOGLE I/O, con el objetivo de crear un entorno dedicado en exclusiva a la programación de aplicaciones para dispositivos ANDROID, proporcionando a GOOGLE un mayor control sobre el proceso de producción. Se trata pues de una alternativa real a ECLIPSE, el IDE recomendado por GOOGLE hasta la fecha, pero que presentaba problemas debido a su lentitud en el desarrollo de versiones que solucionaran las carencias actuales (es indispensable recordar que ECLIPSE es una plataforma de desarrollo, diseñada para ser extendida a través de *plugins*).

ANDROID STUDIO se ha mantenido durante todo este tiempo en versión beta, pero desde el 8 de diciembre de 2014, en que se liberó la versión estable de

ANDROID STUDIO 1.0, GOOGLE ha pasado a recomendarlo como el IDE para desarrollar aplicaciones para su sistema operativo, dejando el *plugin* ADT para ECLIPSE de estar en desarrollo activo. (Vacas, 2014)

II.6.2.1 Principales Características.

- Soporte para programar aplicaciones para ANDROID *Wear* (sistema operativo para dispositivos corporales como por ejemplo un reloj).
- Herramientas LINT (detecta código no compatible entre arquitecturas diferentes o código confuso que no es capaz de controlar el compilador) para detectar problemas de rendimiento, usabilidad y compatibilidad de versiones.
- Utiliza PROGUARD para optimizar y reducir el código del proyecto al exportar a APK (*Application Package File*). Muy útil para dispositivos de gama baja con limitaciones de memoria interna.
- Integración de la herramienta GRADLE encargada de gestionar y automatizar la construcción de proyectos, como pueden ser las tareas de *testing*, compilación o empaquetado.
- Nueva interfaz específica para el desarrollo en ANDROID.
- Permite la importación de proyectos realizados en el entorno ECLIPSE, que a diferencia de ANDROID STUDIO (GRADLE) utiliza ANT (en español: hormiga).
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar: MERCURIAL, GIT, GITHUB o SUBVERSION.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.
- Vista previa en diferentes dispositivos y resoluciones.
- Editor de diseño que muestra una vista previa de los cambios realizados directamente en el archivo XML.

II.6.2.2 Ventajas de ANDROID STUDIO.

- ANDROID STUDIO ha pasado a ser el entorno recomendado para el desarrollo de aplicaciones en ANDROID, al tratarse de un IDE oficial de GOOGLE en colaboración con JETBRAINS (compañía de desarrollo *software* especializada en diseño de IDEs).
- ANDROID STUDIO permite la creación de nuevos módulos dentro de un mismo proyecto, sin necesidad de estar cambiando de espacio de trabajo para el manejo de proyectos, algo habitual en ECLIPSE.
- Con la simple descarga de ANDROID STUDIO se disponen de todas las herramientas necesarias para el desarrollo de aplicaciones para la plataforma ANDROID.
- Su nueva forma de construir los paquetes .APK, mediante el uso de GRADLE, proporciona una serie de ventajas más acorde a un proyecto JAVA:
- Facilita la distribución de código, y por lo tanto el trabajo en equipo.
- Reutilización de código y recursos.
- Permite compilar desde línea de comandos, para aquellas situaciones en las que no esté disponible un entorno de desarrollo.

II.6.2.3 Desventajas de ANDROID STUDIO.

- Aunque ya ha se ha lanzado la primera versión estable, la v1.0, al estar en una fase inicial, siempre es susceptible a introducir cambios que puedan provocar inestabilidad entre proyectos de diferentes versiones.
- Curva de aprendizaje más lenta para nuevos desarrolladores de ANDROID.

II. 6.2.4 Comparación entre ANDROID STUDIO y ADT ECLIPSE.

Para una mayor comprensión de las diferencias y novedades que presenta ANDROID STUDIO con respecto al IDE ECLIPSE, y más concretamente con el ADT para ANDROID, se presenta la tabla 3 que compara ambas opciones:

Características	**Android STUDIO**	**ADT**
Sistema de construcción	GRADLE	ANT
Construcción y gestión de proyectos basado en Maven (herramienta de *software* para la gestión y construcción de proyectos Java, similar a APACHE ANT, pero su modelo es más simple ya que está basado en XML)	Si	No (es necesario instalar un *plugin* auxiliar)
Construir variantes y generación de múltiples APK (muy útil para *Android Wear*)	Si	No
Refactorización y completado avanzado de código ANDROID	Si	No
Diseño del editor gráfico	Si	Si
Firma APK y gestión de almacén de claves	Si	Si
Soporte para NDK (*Native Development Kit*: herramientas para implementar código nativo escrito en C y C++)	Próximas versiones	Si
Soporte para *GOOGLE Cloud Platform*	Si	No
Vista en tiempo real de renderizado de *layouts*	Si	No
Nuevos módulos en proyecto.	Si	No
Editor de navegación.	Si	No
Generador de *assets.*	Si	No
Datos de ejemplo en diseño de *layout.*	Si	No
Visualización de recursos desde editor de código	Si (a la izquierda de la línea de asignación del recurso)	No

Tabla 3. Comparación entre ANDROID STUDIO y ADT ECLIPSE

Fuente: (Vacas, 2014)

II.7 BBDD (Base de Datos)

Según la base de datos que se requiere para llevar a cabo el proyecto, se ha obtenido la información necesaria de las características que esta debe tener. Esta base de datos debe ser:

- Base de datos dinámica.

En estas bases de datos la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, eliminación y edición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado.

- Base de datos Transaccionales

Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, éstas son poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de producción y datos industriales. Es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto no hay redundancia ni duplicación de información como en las demás bases de datos.

Un ejemplo habitual de transacción es el traspaso de una cantidad de dinero entre cuentas bancarias. Normalmente se realiza mediante dos operaciones distintas, una en la que se decrementa el saldo de la cuenta origen y otra en la que se incrementa el saldo de la cuenta destino. Para garantizar la atomicidad del sistema (es decir, para que no aparezca o desaparezca dinero), el sistema debe garantizar que, bajo cualquier circunstancia (incluso una caída del sistema), el resultado final es que: o bien se han realizado las dos operaciones, o no se ha realizado ninguna. (Hernandez, 2003)

II.7.1 Modelo de base de datos relacional

Esta tipo de base de datos almacena información tomando en consideración la relación entre cada uno de los datos; esto lo percibe el diseñador como tablas. El

orden físico de los datos registrados o campos de la tabla son inmateriales, ya que cada registro en la tabla es identificado por un campo que contiene un valor único.

Las relaciones son importantes, ya que establecen una conexión entre un par de tablas que lógicamente no están relacionadas y que la información que estas contienen si se encuentra relacionada de alguna forma.

Las relaciones categorizadas en este modelo son: *one to one* (uno a uno), *one to many* (uno a muchos) y *many to many* (muchos a muchos). (Hernandez, 2003)

II.7.1.2 Relaciones *one to one*

Un par de tablas tiene una relación *one to one* cuando un solo registro de la primera tabla se relaciona a un solo registro de la segunda tabla y viceversa.

II.7.1.3 Relaciones *one to many*

Un par de tablas tiene una relación *one to many* cuando un registro de la primera tabla puede relacionarse con uno o varios registros de la segunda tabla. En el caso de la segunda tabla, solo un registro de la misma puede relacionarse a un registro de la primera tabla.

II.7.1.4 Relaciones *many to many*

Un par de tablas tienen una relación many to many cuando un registro de la primera tabla puede relacionarse con uno o varios registros de la segunda tabla y viceversa.

Existe otro tipo de relación denominado *Self-Referencing*, en donde no se toman en consideración dos tablas, sino que se refiere a la relación que existe entre los datos

de una misma tabla. De igual forma estas relaciones en una sola tabla pueden ser *one to one, one to many y many to many*.

El lenguaje utilizado para crear, modificar y mantener las bases de datos relacionales es el SQL (*Structured Query Language*).

II.8.2 MySQL

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. (MySQL, 1997)

Existen varias interfaces de programación de aplicaciones que permiten, a aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySQL, incluyendo C, C++, C#, PASCAL, DELPHI (vía DBEXPRESS), EIFFEL, SMALLTALK, JAVA (con una implementación nativa del *driver* de JAVA), LISP, PERL, PHP, PYTHON, RUBY, GAMBAS Y REALBASIC (MAC y LINUX), cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL.

MySQL es muy utilizado en aplicaciones *web*, como DRUPAL o PHPBB, en plataformas (LINUX/WINDOWS-APACHE-MySQL-PHP/PERL/PYTHON), y por herramientas de seguimiento de errores como BUGZILLA. Su popularidad como aplicación *web* está muy ligada a PHP, que a menudo aparece en combinación con MySQL. (MySQL, 1997)

II.9 Seguridad

II.9.1 Encriptación.

Es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible ésta no pueda ser obtenida con facilidad por terceros. Opcionalmente puede existir un proceso de desencriptación a través del cual la información puede ser interpretada de nuevo a su estado original, aunque existen métodos de encriptación que no pueden ser revertidos. El término encriptación es traducción literal del inglés y no existe en el idioma español. La forma más correcta de utilizar este término sería cifrado. (TANENBAUM & WETHERALL, 2012)

II.9.1.1 Métodos de encriptación

Para poder encriptar un dato, se pueden utilizar tres procesos matemáticos diferentes: los algoritmos HASH, los simétricos y los asimétricos.

- Algoritmo HASH.

Este algoritmo efectúa un cálculo matemático sobre los datos que constituyen el documento y da como resultado un número único llamado MAC (*Message Authentication Code*). Un mismo documento dará siempre un mismo MAC (*Message Authentication Code*).

- Criptografía de Clave Secreta o Simétrica

Utilizan una clave con la cual se encripta y desencripta el documento. Todo documento encriptado con una clave, deberá desencriptarse, en el proceso inverso,

con la misma clave. Es importante destacar que la clave debería viajar con los datos, lo que hace arriesgada la operación, imposible de utilizar en ambientes donde interactúan varios interlocutores.

Los criptosistemas de clave simétrica se caracterizan porque la clave de cifrado y la de descifrado es la misma, por tanto la robustez del algoritmo recae en mantener el secreto de la misma.

Sus principales características son:

- Rápidos y fáciles de implementar.
- Clave de cifrado y descifrado son la misma.
- Cada par de usuarios tiene que tener una clave secreta compartida.
- Una comunicación en la que intervengan múltiples usuarios requiere muchas claves secretas distintas.

Actualmente existen dos métodos de cifrado para criptografía de clave secreta, el cifrado de flujo y el cifrado en bloques. (TANENBAUM & WETHERALL, 2012)

- Cifrado de flujo.

El emisor A, con una clave secreta y un algoritmo determinístico RKG (*Random Key Generator*), genera una secuencia binaria (s) cuyos elementos se suman módulo 2 con los correspondientes bits de texto claro m, dando lugar a los bits de texto cifrado c, Esta secuencia (c) es la que se envía a través del canal. En recepción, B, con la misma clave y el mismo algoritmo determinístico, genera la misma secuencia cifrante (s), que se suma módulo 2 con la secuencia cifrada (c), dando lugar a los bits de texto claro m. Los tamaños de las claves oscilan entre 120 y 250 bits.

- Cifrado en bloque.

Los cifrados en bloque se componen de cuatro elementos:

- Transformación inicial por permutación.
- Una función criptográfica débil (no compleja) iterada r veces o "vueltas".
- Transformación final para que las operaciones de encriptación y desencriptación sean simétricas.
- Uso de un algoritmo de expansión de claves que tiene como objeto convertir la clave de usuario, normalmente de longitud limitada entre 32 y 256 bits, en un conjunto de subclaves que puedan estar constituidas por varios cientos de bits en total.

- Algoritmos Asimétricos (RSA).

Requieren dos Claves, una Privada (única y personal, solo conocida por su dueño) y la otra llamada Pública, ambas relacionadas por una fórmula matemática compleja imposible de reproducir. El concepto de criptografía de clave pública fue introducido por Whitfield Diffie y Martin Hellman a fin de solucionar la distribución de claves secretas de los sistemas tradicionales, mediante un canal inseguro. El usuario, ingresando su PIN genera la clave Pública y Privada necesarias. La clave Pública podrá ser distribuida sin ningún inconveniente entre todos los interlocutores. La Privada deberá ser celosamente guardada. Cuando se requiera verificar la autenticidad de un documento enviado por una persona se utiliza la Clave Publica porque el utilizó su Clave Privada. (TANENBAUM & WETHERALL, 2012)

II.10.1.2 Encriptación con AES (*Advanced Encryption Standard*).

Al realizar una comunicación con el servidor *web* desde la aplicación ANDROID es indispensable la implementación de algún algoritmo de encriptación robusta donde se oculten las contraseñas de los usuarios para que los atacantes no puedan detectarlas. Es por esto que se utilizó AES para encriptar solamente las contraseñas mediante una clase, la cual solamente conoce el programador.

AES es un algoritmo de cifrado por bloques, inicialmente fue diseñado para tener longitud de bloque variable pero el estándar define un tamaño de bloque de 128 bits, por lo tanto los datos a ser encriptados se dividen en segmentos de 16 bytes (128 bits) y cada segmento se lo puede ver como un bloque o matriz de 4x4 bytes al que se lo llama estado.

Por ser simétrico, se utiliza la misma clave para encriptar como para desencriptar, la longitud de la clave puede ser de 128, 192 o 256 bits según especifica el estándar, esto permite tres implementaciones conocidas como AES-128, AES-192 y AES-256, el presente trabajo está basado en AES-128.

Partiendo de una clave inicial de 16 bytes (128 bits), que también se la puede ver como un bloque o matriz de 4x4 bytes, se generan 10 claves, estas claves resultantes junto con la clave inicial son denominadas subclaves. (Forouzan, 2006)

II.10 Servidor APACHE

El Servidor APACHE es un servidor *web* HTTP de código abierto para plataformas UNIX (BSD, GNU/LINUX, etc.), MICROSOFT WINDOWS, MACINTOSH y otras, que implementa el protocolo HTTP/1.1.

APACHE es usado principalmente para enviar páginas web estáticas y dinámicas en la *World Wide Web*. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación APACHE, o que utilizarán características propias de este servidor *web*.

APACHE es el componente de servidor *web* en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/PERL/PYTHON.

Los programadores de aplicaciones *web* a veces utilizan una versión local de APACHE con el fin de previsualizar y probar el código mientras éste es desarrollado.

MICROSOFT Internet Information Services (IIS) es el principal competidor de APACHE, así como SUN JAVA *SYSTEM Web Server* de SUN MICROSYSTEMS y un anfitrión de otras aplicaciones como ZEUS *Web Server*. Algunos de los más grandes sitios *web* del mundo están ejecutándose sobre APACHE. La capa frontal (*front end*) del motor de búsqueda GOOGLE está basado en una versión modificada de APACHE, denominada GOOGLE *Web Server* (GWS). Muchos proyectos de WIKIMEDIA también se ejecutan sobre servidores *web* APACHE. (Maldonado, 2008)

II.10.1 Ventajas del servidor APACHE.

- Modular.
- Código abierto.
- Multi-plataforma.
- Extensible.
- Popular (fácil conseguir ayuda/soporte)
- APACHE tiene amplia aceptación en la red.

Desde el año 1996, APACHE, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios *web* en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Cabrera, 2014)

II.10.2 Módulos.

La arquitectura del servidor APACHE es muy modular. El servidor consta de una sección *core* y diversos módulos que aportan mucha funcionalidad, que podría considerarse básica para un servidor *web*. Algunos de estos módulos son:

- **mod_ssl**: Comunicaciones Seguras vía TLS.
- **mod_rewrite**: reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como PHP en páginas estáticas HTML para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- **mod_dav**: Soporte del protocolo *WebDAV* (RFC 2518).
- **mod_deflate:** Compresión transparente con el algoritmo *deflate* del contenido enviado al cliente.
- **mod_auth_ldap**: Permite autentificar usuarios contra un servidor LDAP (*Lightweight Directory Access Protocol*).
- **mod_proxy_ajp**: Conector para enlazar con el servidor JAKARTA TOMCAT de páginas dinámicas en JAVA (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- **mod_cband:** Control de tráfico y limitador de ancho de banda.
- **mod_perl**: Páginas dinámicas en Perl.

- **mod_php**: Páginas dinámicas en PHP.
- **mod_python**: Páginas dinámicas en Python.
- **mod_rexx:** Páginas dinámicas en REXX y *Object* REXX.
- **mod_ruby:** Páginas dinámicas en Ruby.
- **mod_aspdotnet:** Páginas dinámicas en .NET de MICROSOFT (Módulo retirado).
- **mod_mono:** Páginas dinámicas en Mono
- **mod_security**: Filtrado a nivel de aplicación, para seguridad.

Entre los módulos que se utilizaron en el servidor APACHE 2.4.4 de este trabajo se encuentran principalmente los siguientes: mod_php, mod_security, mod_ssl y mod_rewrite, entre otros.

II.11. WAMP

WAMP *Server* es un paquete de herramientas que provee a los desarrolladores con los elementos necesarios para un servidor *web,* equipado con: un manejador de base de datos (MySQL), un *software* para servidor *web* (APACHE) y un *software* de programación *script Web* (PHP (generalmente), PYTHON o PERL), debiendo su nombre a dichas herramientas. Lo mejor de todo es que WAMP es completamente gratuito. WAMP incluye, además de las últimas versiones de APACHE, PHP y MySQL, versiones anteriores de las mismas, para el caso de que se quiera probar en un entorno de desarrollo particular. También incluye una aplicación *web* llamada PHPMYADMIN que permite crear y administrar bases de datos MySQL, y otra aplicación llamada SQLitemanager que tiene el mismo propósito que la anterior. (Ortega Martinez, 2013)

II.12 HTTP (*HyperText Transfer Protocol*).

Es un protocolo cliente/servidor el cual se orienta a las transacciones; de igual forma es un protocolo sin estados lo cual nos indica que cada transacción se evalúa de forma independiente, lo que se traduce en que al momento en que se realiza una transacción se crea una conexión nueva y la misma culmina una vez se completa de forma efectiva dicha transacción; esto se repetirá cada vez que se realice una transacción. (TANENBAUM & WETHERALL, 2012)

La Figura 7 muestra tres ejemplos de funcionamiento de HTTP: el primer ejemplo ilustrado es cuando el cliente (agente de usuario), establece una conexión directa con el servidor de origen donde se encuentran los recursos de interés. En este caso es el cliente el que realiza la solicitud HTTP, la cual está compuesta por una orden completa, denominada método, una URL (*Uniform Resource Locator*) y un mensaje de tipo MIME (*Multipurpose Internet Mail Extensions*) que contiene todas las características de la solicitud, tales como, información acerca del cliente, información adicional del contenido, entre otros. Una vez el servidor recibe la solicitud, éste intenta realizar con éxito la acción solicitada y regresa al cliente con una respuesta HTTP

El segundo ejemplo que podemos ver en la Figura 7 se aprecia un caso en el que no existe una conexión TCP entre el cliente y el servidor de origen. En su lugar, existen uno o más sistemas intermedios con conexiones TCP entre sistemas lógicamente adyacentes. De manera que la solicitud del cliente se transmite a través de dichos sistemas intermedios al servidor de origen al igual que la respuesta.

La tercera imagen de la Figura 7 es un ejemplo de una caché. Una caché es un servicio que puede almacenar solicitudes y respuestas para responder nuevas solicitudes. Por lo tanto si se realiza una solicitud que se ha hecho previamente, entonces la caché puede dar respuesta a dicha solicitud sin acceder al recurso

indicado en la URL.. Es importante destacar que no todas las transacciones pueden almacenarse o dicho almacenaje puede ser por un tiempo determinado.

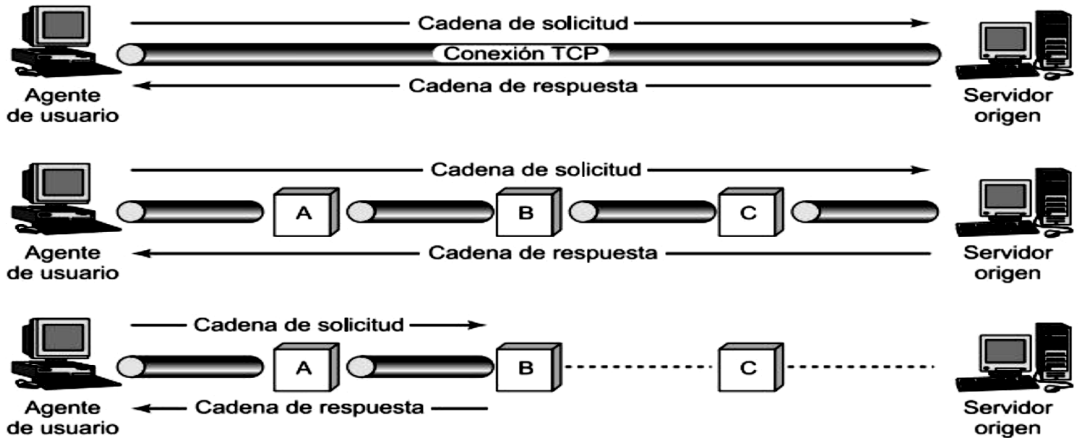


Figura 7. Ejemplo del funcionamiento de HTTP

Fuente: (TANENBAUM & WETHERALL, 2012).

El mensaje HTTP está compuesto por dos tipos de mensajes: las solicitudes que realiza el cliente al servidor y las respuestas que este último proporciona. En la Figura 8 se encuentra la estructura general de los mensajes HTTP. (TANENBAUM & WETHERALL, 2012)

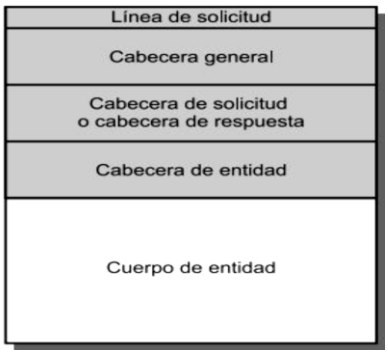


Figura 8. Estructura general de los mensajes HTTP

Fuente: (TANENBAUM & WETHERALL, 2012).

CAPITULO III

METODOLOGÍA

Este capítulo presenta al lector las diferentes fases en las que se dividió la metodología, estas fases son necesarias para lograr los objetivos planteados en este Trabajo Especial de Grado.

La presente investigación se realizó bajo la modalidad de Proyecto factible, con la intención de desarrollar una aplicación para dispositivos móviles, que sirviera como alternativa para solventar distintos inconvenientes que tienen los usuarios a la hora de pagar facturas dentro de distintos establecimientos que pudieron ser constatados a través de observación directa

III.1 PRIMERA FASE: Investigación documental

Esta fase comprende el estudio y tratamiento de la información requerida para la investigación realizada. Se procesaron, clasificaron y se recuperaron distintos tipos de información referidas a los tópicos y temas asociadas a este trabajo de investigación.

Se requiere de esta fase para lograr una revisión documental que permitiría evaluar y posteriormente seleccionar las herramientas, estrategias y tecnologías que se aplicarían para lograr los siguientes objetivos.

- Identificar cuales tecnologías son las que mejor se adaptaban a los requerimientos del sistema para su correcto funcionamiento.
- Almacenar la información del cliente para poder realizar el pago de las facturas

- Establecer la conexión de las distintas tecnologías involucradas en el sistema
- Elegir los lenguajes de programación adecuados para alcanzar las metas anteriormente mencionadas

III.2 SEGUNDA FASE: Selección de tecnologías

Durante esta fase se realizó el análisis de la información suministrada al finalizar la fase anterior. Este análisis se dividió en dos bloques, en el primer bloque se compararon los distintos sistemas operativos móviles para seleccionar el sistema operativo en el que se desarrollaría la aplicación móvil y el segundo bloque definió cuales serían las tecnologías de comunicación entre las aplicaciones y la base de datos.

Sistema operativo para dispositivos móviles	Cantidad vendidas a nivel mundial
ANDROID	255.3
IOS	32.5
WINDOWS PHONE	7.4
BLACKBERRY	1.5
Otros	1.9

Tabla 4. Comparación en ventas de sistemas operativos para teléfonos móviles (cifras expresadas en millones) en el año 2014

Fuente: (Llamas, 2014)

En la tabla 4, se puede observar que el mercado a nivel mundial está totalmente dominado por ANDROID como sistema operativo para teléfonos celulares, por lo tanto sería la primera opción para el desarrollo de una aplicación de uso masivo, puesto que podría llegar a un mayor número de personas, esto sumado al hecho de ser un sistema operativo con un modelo de desarrollo de código abierto, fueron razones suficientes para preferir este sistema operativo como el ideal para el desarrollo de la aplicación.

Para el segundo bloque se realizó sobre la base de un análisis en los beneficios y características de cada una de las opciones que se tenían al alcance para comunicar el dispositivo móvil a la base de datos y hacer un intercambio de datos, a continuación se observa en la tabla 5 la información recopilada.

Alternativas	Características
UDP	Protocolo no orientado a conexión, por lo tanto, no ofrece confiabilidad a la comunicación. Este protocolo es ideal para transmisión de video y voz en tiempo real.
TCP	Protocolo orientado a conexión, por lo tanto, ofrece por lo tanto ofrece confiabilidad en la comunicación. La cabecera normalmente es de unos 20 bytes lo aporta pocos datos al *overhead*
Web service Protocolo HTTP	Ofrece confiabilidad en la comunicación ya que está basado en TCP. Es ideal cuando se requieren múltiples consultadas a la base de datos. Normalmente se genera un aumento del *overhead* ya que la cabecera es de gran tamaño

Tabla 5. Alternativas para la comunicación entre el dispositivo móvil y la base de datos.

Fuente: propia

En esta fase también se definieron los lenguajes de programación ideales para lograr la conexión entre el punto NFC y la computadora que haría las funciones de servidor, tomando en cuenta que la aplicación no podía ser excesivamente pesada y no podía consumir demasiados recursos, así como también la plataforma ideal para manejar las bases de datos. Se tomó en cuenta el hecho que la aplicación principal

para el servidor preferiblemente debía ser multiplataforma, de esta manera se garantizaba que la misma funcionara de manera correcta en cualquier sistema operativo para computadoras.

Una vez definidas las plataformas donde se desarrollarían las aplicaciones y las herramientas de programación, así como también la manera en que se comunicarían cada uno de los elementos del sistema, se definieron las aplicaciones necesarias para el correcto funcionamiento del proyecto

III.3 TERCERA FASE: Diseño de la base de datos

Para la elaboración de la base de datos del sistema, primero se procedió a hacer un modelo conceptual, con la intención de identificar que datos eran necesarios y la forma que tomaría la base de datos finalmente. Esto se llevó a cabo tomando en cuenta los actores que estaban involucrados en el proceso. Estos actores se les conocen como entidades los cuales cumplen una función dentro del sistema

Para la elaboración de la base de datos se utilizó la herramienta WORKBENCH de MySQL. Esta herramienta facilita enormemente la creación y diseño de las bases de datos, ya que cuenta con una interfaz gráfica donde se visualizan fácilmente las tablas, datos y relaciones que existen entre las entidades. Este proceso es conocido como "Generación de entidad – relación". Una vez que culmina este proceso, WORKBENCH posee un proceso que construye la base de datos para posteriormente utilizarla en el servidor MySQL.

Tomando en cuenta lo anterior, se observó que eran necesarias tres entidades:

- **Entidad usuario**: Esta entidad se encarga de identificar a la persona que realizará el pago de la factura, también tiene sus datos para poder ingresar al sistema. Sus características son: nombre, apellido, *email*, clave.

- **Entidad tarjeta**: Esta entidad se encarga de mantener el control sobre las tarjetas de los usuarios. Sus características son: *email*, número de la tarjeta, el monto asignado a las mismas y el saldo actual.

- **Entidad control**: Esta entidad se encarga de identificar cada una de las operaciones monetarias que se realizan dentro del sistema, llevando un control estricto para posteriormente consultar cualquier movimiento financiero. Las características de esta entidad son: número de tarjeta, monto de la transacción y numero de referencia.

Posteriormente se procedió a la creación de la base de datos, se utilizó la herramienta WORKBENCH de MySQL para la generación de las siguientes tablas:

- **Usuario:** Contiene los datos que se nombraron anteriormente, además de tener una relación *uno a uno* con la tabla tarjeta, de esta manera cada usuario se le asignará una sola tarjeta la cual podrá utilizar en el sistema.

- **Tarjeta:** Posee los datos que se nombraron anteriormente, esta tabla tiene una relación *uno a muchos* con la tabla control, puesto que la misma tarjeta puede tener múltiples movimientos financieros.

- **Control:** Contiene los datos necesarios para el control de las transacciones bancarias.

Estos *scripts* generados en WORKBENCH MySQL fueron utilizados posteriormente para la creación de la aplicación servidor.

III.4 CUARTA FASE: Desarrollo de la aplicación servidor

La aplicación servidor fue desarrollada en el lenguaje de programación PHP, el cual fue diseñado para el desarrollo *web* dinámico y se utilizó el servidor HTTP APACHE para la comunicación *web*.

Esta aplicación permite la comunicación del dispositivo móvil a la base de datos para el registro de usuarios nuevos y el *login* de usuarios ya registrados, permitiendo obtener información que usará el dispositivo posteriormente en la aplicación que se encargará de las transacciones bancarias.

El servidor se encargará de modificar la tabla de usuario para la creación de nuevos usuarios. En el caso de que el dispositivo móvil requiera un *login* el servidor, verificará que el usuario ya ha sido registrado y devolverá la información de la tabla tarjeta asociada a dicho usuario.

III.5 QUINTA FASE: Desarrollo de la aplicación principal

Durante esta fase se procedió al desarrollo de la aplicación que se encargaría de manejar las transacciones bancarias que se realicen dentro del sistema, esta aplicación fue desarrollada en el IDE (Entorno de desarrollo por sus siglas en inglés) NETBEANS.

El lenguaje de programación seleccionado fue JAVA, puesto que al ser multiplataforma es independiente del sistema operativo alojado dentro del computador servidor.

La aplicación banco intervendrá en distintas tareas dentro del sistema, la primera de ellas es asociar cada usuario nuevo con una tarjeta nueva, esto se hace

gracias a la tabla tarjeta, tabla que contiene información de distintas tarjetas destinadas para la asignarse a los usuarios para poder utilizar el sistema.

Esta aplicación se encargará de escuchar continuamente el puerto USB en el cual esté conectado el punto NFC, al detectar un intercambio de información la misma capturará estos datos para enviarlos a la base de datos.

Estos datos modificarán dos tablas dentro de la base de datos, la primera en modificarse será la tabla de tarjetas de la siguiente manera:

- Se verificará el número de la tarjeta para localizar el registro de la misma dentro de la base de datos

- Una vez encontrado este campo, se restará el saldo anterior con el monto enviado a través del punto NFC para obtener un nuevo saldo actual

- Por último, con el resultado de esta operación se modificará los datos dentro de la tabla tarjeta y se colocará un nuevo saldo actual.

La segunda tabla a modificarse será la tabla de control, esto ocurrirá de la siguiente manera:

- Con la información del número de la tarjeta se procede a llenar un registro vacío dentro de la tabla control.

- Se incluirá la información del monto de la operación para registrarlo en la tabla control.

- Por último se asignará un número de referencia automático para posteriormente consultar los movimientos bancarios en caso de ser necesario.

III.6 SEXTA FASE: Desarrollo de la aplicación ANDROID

La aplicación para los dispositivos móviles fue desarrollada para el sistema operativo ANDROID, por motivos explicados anteriormente. La aplicación fue hecha en el IDE (Entorno de desarrollo, por sus siglas en inglés) ANDROID STUDIO.

Esta aplicación se desarrolló con el fin de crear el cliente para cumplir los requerimientos del sistema, esto con la intención de poder lograr que el sistema funcione y lograr los requerimientos tanto de la aplicación servidor como de la aplicación bancaria.

La principal función de esta aplicación es lograr que el dispositivo se comporte como una *tag* NFC y envié la información de la tarjeta y el monto de la misma a través de esta plataforma, conectándose con la aplicación bancaria a través de un punto NFC.

La segunda función es conectarse con la aplicación servidor tanto para registrar usuarios nuevos como hacer el *login* para obtener los datos de la tarjeta asociada al usuario y proceder a hacer las tareas de la función principal.

III.7 sÉPTIMA FASE: Pruebas del Sistema.

Una vez finalizadas las fases anteriores, se procedió a realizar pruebas pilotos en cada una de las aplicaciones, de esta manera se observa el correcto funcionamiento del sistema, las pruebas se realizaron con la intención de verificar los siguientes puntos de interés:

- Conexión del cliente a la aplicación servidor.
- Respuesta del servidor al cliente.
- Transferencia desde el dispositivo móvil al punto NFC.
- Conectividad del punto NFC a la aplicación principal.

- Consultas y modificaciones automáticas de las bases de datos.
- Descuento final de saldo al cliente.

III.8 OCTAVA FASE: Conclusiones y recomendaciones

En fase se demuestran las soluciones encontradas al problema planteado, posteriormente se muestra al lector las conclusiones obtenidas al concluir el Trabajo Especial de Gado. También se ofrecen múltiples recomendaciones, obteniendo una descripción de las acciones a tomar para la mejora del trabajo realizado.

III.9 NOVENA FASE: Elaboración del tomo

En esta fase se consideran los mecanismos y procedimientos formales determinados por la Universidad Católica Andrés Bello para realización del Tomo; documentando de forma cronológica los momentos que conforman el proceso de investigación y de pruebas realizadas para comprobar el buen funcionamiento del sistema. Adicionalmente, en esta fase se deja constancia del cumplimiento a los objetivos planteados al inicio de la investigación.

CAPITULO IV

RESULTADOS

En este capítulo se muestra al lector los resultados obtenidos en la investigación, describiendo y analizando los resultados obtenidos en cada una de las fases desarrolladas.

El propósito de la investigación, sin entrar en mayores detalles, fue crear una aplicación que ofrezca a distintos clientes una alternativa para realizar pagos a través de sus dispositivos móviles de manera rápida, confiable y sencilla apoyándose en la plataforma NFC. Esta aplicación disminuye el tiempo que invierte cada persona a la hora de pagar sus facturas en cualquier local comercial.

IV.1. Selección de tecnologías

Una vez culminada la investigación documental, se llevó a cabo un proceso de comparación y selección de tecnologías que estarían involucradas en el sistema.

IV.1.1. Conexión a la aplicación servidor

En esta instancia del sistema se manejaron dos alternativas posibles:

- Conexión a través de *sockets.*
- Conexión mediante servicios *web.*

Los *Sockets* permiten la conexión a dispositivos remotos o locales, haciendo uso de direcciones IP públicas o privadas y habilitando el puerto en el cual el servidor atenderá las peticiones de los clientes.

Los servicios *web* permiten a dispositivos finales realizar consultas a aplicaciones alojadas en servidores remotos haciendo uso del protocolo, resulta realmente útil cuando múltiples consultas y los datos de aplicación enviado son de gran tamaño.

Puesto que la aplicación se realizó para su futura implementación, con la intención de que pueda funcionar como una alternativa de pago cotidiana; lo ideal sería que hubiera una comunicación segura y que el servidor recibiera múltiples consultas de múltiples usuarios. Por lo cual, la elección del servicio *web* es la más apropiada para el sistema de transacción inalámbrico.

IV.1.2. Herramientas y lenguajes de programación

Para lograr la comunicación entre el servicio *web* y la aplicación ANDROID, se seleccionó PHP como lenguaje de programación para el lado del servidor, el mismo se encarga de registrar los datos en la lista de la base de datos destinada a la información del usuario, también se encarga del inicio de sesión por parte del cliente previamente registrado, para el correcto funcionamiento del servicio *web* se utilizó la herramienta WAMPSERVER.

Para el desarrollo de la aplicación principal se tomó JAVA como lenguaje de programación, puesto que el lenguaje es multiplataforma lo que permite el correcto funcionamiento de la aplicación, independientemente del sistema operativo. Otro punto fuerte de JAVA es que existen múltiples herramientas que permiten el desarrollo de aplicaciones, entre ellas podemos nombrar a ECLIPSE y NETBEANS. La aplicación principal fue desarrolla en NETBEANS puesto que facilita la creación de la interfaz gráfica necesaria para la aplicación principal.

Para la base de datos se necesitó una herramienta para la gestión de la base de datos, preferiblemente de código abierto, para evitar pagar por alguna licencia de

código propietario. Las alternativas fueron MySQL y PostgreSQL, actualmente MySQL es más utilizada que PostgreSQL puesto que posee una mayor velocidad a la hora de realizar las operaciones, por lo tanto es un gestor de base de datos que presenta un gran rendimiento, además MySQL por su bajo consumo puede ser usado en computadoras con escasos recursos. Todas estas ventajas llevaron a la elección de MySQL como el gestor de base de datos para el sistema.

IV.2. Diseño de la base de datos

Durante el desarrollo del sistema, se generó una base de datos en la cual se almacena la información del usuario, de su tarjeta y un registro de las transacciones realizadas usando las aplicaciones diseñadas, por lo tanto haciendo uso MySQL WORKBENCH se generaron los scripts necesarios para usarlos dentro del servidor *web*. Para el funcionamiento del proyecto se necesitaron tres tablas, que se pueden observar en la figura 9, las cuales permiten al sistema:

- Almacenar los datos de los usuarios nuevos al momento de registrarse para posteriormente permitir el ingreso de los mismos y disfrutar de los beneficios del sistema
- Registrar el serial de la tarjeta NFC del dispositivo móvil para registrarlo como número de tarjeta, para posteriormente hacer los descuentos necesarios.
- Guardar un comprobante de todas las operaciones monetarias que se hacen en el sistema para futuras consultas.

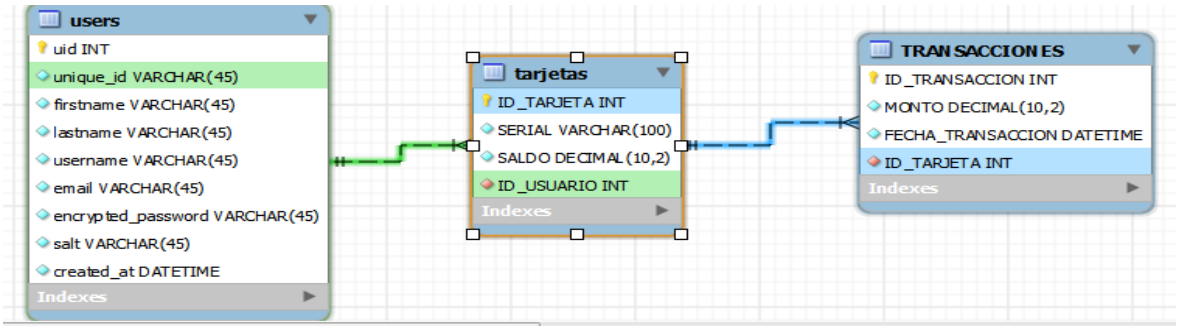


Figura 9. Diagrama Entidad-Relación de la base de datos del sistema.

Fuente: Propia.

IV.3 Desarrollo de la aplicación servidor

Esta parte del sistema es la encargada recibir y enviar toda la información requerida a través del servicio *web*. Esta comunicación se realiza gracias al intercambio del servicio *web* a través del código PHP con el dispositivo móvil, este intercambio de información se lleva a cabo a través de un JSON (*JavaScript Object Notation*) que emite una petición desde la aplicación ANDROID para que el mismo sea respondido por el servicio *web*, devolviendo información, dependiendo de la necesidad del usuario. Principalmente la aplicación servidor responde las siguientes peticiones:

- Registro de nuevo usuario: El servicio *web* al escuchar una petición para el ingreso de un nuevo usuario, usará la información suministrada por el dispositivo móvil para agregar estos datos a la lista *users*. Cuando se procede a almacenar la información de la clave o *password* del cliente la misma procede a ser encriptada usando un Hash para ofrecer un nivel más alto de seguridad en la base de datos. Una vez que el registro es finalizado, se procede a enviar un correo electrónico informando al usuario que el registro fue exitoso, como se observa en la figura 10.

Figura 10. Correo electrónico enviado por la aplicación servidor al momento del registro

Fuente: Propia

- *Login* o ingreso de usuario: Para permitir el ingreso de usuarios previamente registrados se requiere el correo electrónico y su contraseña para verificar que dichos datos se encuentren almacenados en la base de datos. Posteriormente responderá a la aplicación móvil para que se permita la interacción con el sistema de pagos.
- Cambio de contraseña: Si el usuario en cualquier momento olvida su contraseña, esta función permite que ingresando su correo electrónico pueda crear una nueva contraseña. Al ejecutar estas acciones la aplicación envía un correo electrónico informando que el cambio de contraseña fue exitoso, esto se puede evidenciar en la figura 11.

Figura 11. Correo electrónico enviado desde la aplicación servidor al cambiar contraseña del usuario

Fuente propia

Las características nombradas anteriormente son las funciones que ofrece la aplicación servidor antes de que el usuario acceda al sistema, por lo tanto a continuación se explicaran el resto de las tareas que realiza el servidor:

- Información de saldo disponible: Permite consultar el saldo que actualmente el usuario posee a su favor, el servidor *web* responderá de manera tal de mantener informado a los clientes sobre la capacidad de pago que el mismo posea.
- Información de cuenta: Regresa la información personal suministrada por el usuario al momento de crear su cuenta en sistema. Esta información se mostrará en la pantalla de la aplicación y la misma debe ser verificada por la persona encargada de recibir el pago para constatar la identidad del cliente
- Información de monto de transacción: Presenta al usuario el último monto que se le fue debitado a la hora de realizar una transacción monetaria a través del sistema, esto con la intención de que el cliente pueda asegurarse que el saldo debitado fue el correcto.

En cualquiera de los casos es evidente que el código PHP está diseñado para la comunicación continua con la base de datos diseñada para el sistema, las consultas se

hace de manera automática para obtener un correcto funcionamiento, esta aplicación no cuenta con una interfaz gráfica, puesto que las respuestas se observan directamente en la aplicación ANDROID.

IV.4 Desarrollo de la aplicación principal

La aplicación principal tiene múltiples funciones dentro del sistema de pagos automatizadas. Aunque solo se diseñó una aplicación en JAVA, la misma utiliza múltiples clases para lograr los objetivos necesarios dentro del sistema. Esta aplicación permite la comunicación entre el dispositivo móvil, el terminal NFC y la computadora en la cual está alojada la aplicación como se puede observar en la figura 12, también la tiene la capacidad de interactuar con la base de datos para el envío de información por correo electrónico y modificar los campos que están involucrados dentro de la transacción bancaria. A continuación se explicarán las distintas funciones que cumple esta aplicación, la cual cuenta con una interfaz gráfica amigable y sencilla, para que sea de fácil uso para cualquier persona.

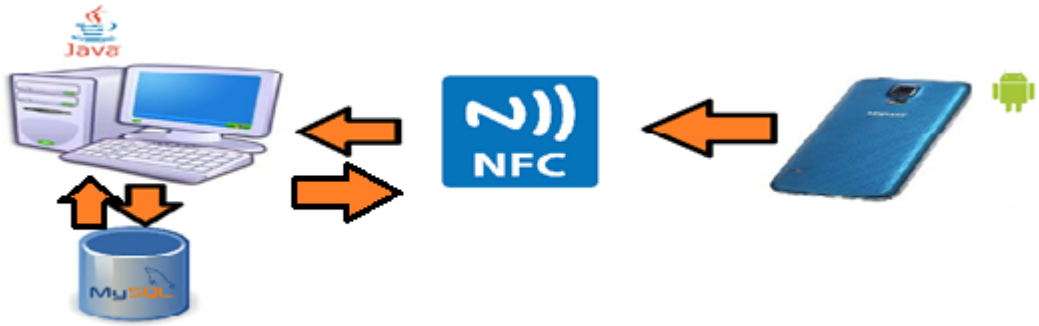


Figura 12. Esquema del funcionamiento de la aplicación principal

Fuente: Propia

Para el diseño y correcto funcionamiento de la aplicación se requirió que el sistema gestionara la base de datos. Las siguientes clases se encargan de buscar,

insertar, modificar y eliminar datos dentro de las listas de la base de datos destinadas para almacenar la información del usuario, su tarjeta y de las transacciones realizadas. Las clases involucradas para estos fines son las siguientes:

- *ConeccionBD.java*: Esta clase se encarga de iniciar la conexión con base de datos y cerrar la conexión con la misma. Está Diseñada para funcionar con el nombre de la base de datos *comprasnfc* el cual contiene la información de todos los clientes utiliza los datos necesarios para acceder a la misma y obtener los permisos para las modificaciones. Todas las clases cuyos nombres terminan en DAO (*Data Access Object*) extienden de esta clase y permite realizar todas las conexiones que los objetos DAO (*Data Access Object*) necesitan.
- *ClienteDAO.java*: Esta clase se encarga en obtener toda la información referente al usuario, interactúa directamente con la tabla *users*. esta clase puede buscar a un usuario ya sea por el nombre de usuario, serial de su tarjeta o correo electrónico, estas búsquedas son necesarias para interactuar con el usuario y ofrecer información al operador. Esta clase además puede insertar, actualizar y eliminar datos dentro su tabla correspondiente
- *TarjetaDAO.java*: Esta clase está clase se encarga de obtener la información referente a la tarjetas del usuario interactuando con la tabla tarjetas. Al igual que la clase *ClienteDAO.java* tiene métodos para buscar insertar, actualizar y eliminar datos de la tabla tarjetas. Su uso es primordial para almacenar los datos del serial NFC que contiene el dispositivo móvil y hacer una diferenciación entre los mismos para asignarlas a un usuario en específico.
- *TransaccionDAO.java:* Esta clase es la encargada de la gestión de la tabla Transacciones, como las clases anteriores DAO (*Data Access Object*) tiene distintos métodos para permitir la búsqueda, actualización, insertar o eliminar datos dentro de los registros de su tabla asignada. La idea para la creación de esta tabla es permitir almacenar un registro de transacciones dentro del

sistema para posteriormente el cliente o el operador del *software* puedan consultar de manera sencilla cualquiera de estos datos almacenados.

Es importante recalcar que algunas funciones DAO interactúan entre sí para ofrecer un mejor funcionamiento del sistema, por lo tanto algunas de estas clases pueden obtener información de otras tablas además de su tabla asignada.

Las siguientes clases son las encargadas de interactuar con el usuario. Estas clases se encuentran incluidas en el paquete Ventanas, dichas clases están basadas en el paquete *Swing* de JAVA. Define parte de la funcionalidad del sistema, puesto que se encarga de validaciones básicas de formato de texto, por ejemplo: campo vacío. Otra de las grandes funciones de estas clases es mostrar al usuario los datos suministrados por las capas de modelo y controlador, de esta forma se puede apreciar el correcto funcionamiento del sistema.

- Clase *Inicial.java:* Esta clase se encarga de mostrar la primera pantalla de la aplicación, la cual solicita una clave de acceso para ingresar a la aplicación como se observa en la figura 13.

Figura 13. Primera pantalla de la aplicación principal

Fuente: Propia

- Clase *Principal.java*: Esta clase muestra la pantalla principal de la aplicación, donde se visualizan las opciones principales del sistema, se aprecia en la figura 14.

Figura 14. Vista de la clase Principal.java

Fuente: Propia

- Paquete *com.nfc.ventana.clientes*: Dentro de este paquete se encuentran las clases visuales que permiten la interacción con los datos del cliente, ya sea para consultar, registrar serial o recargar el saldo. La vista de estas clases se puede observar en la figura 15

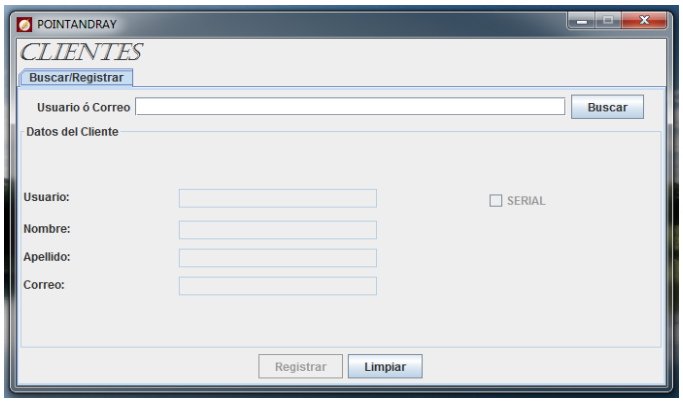


Figura 15. Vista de las clases que conforman la vista Clientes

Fuente: Propia

- Paquete *Pagos*: En este paquete, se encuentran todas las clases necesarias para la interacción del usuario con la capa controlador para realizar la transacción monetaria.

- Paquete *Transacciones*: En este paquete, se encuentran todas las clases que muestran las transacciones realizadas dentro del sistema para la consulta del mismo en cualquier momento.

  A continuación se muestrantodas las clases encargadas de realizar las funciones principales de la aplicación, se definen toda las funciones necesarias para la buena comunicación entre las dos capas descritas anteriormente por lo tanto es la capa que define cuando es necesario llamar algunas de las clases DAO y cuáles son los atributos que está solicitando el usuario.

Entre estas clases de negocio, se encuentran en el paquete *com.nfc.negocio* y está conformado por las siguientes clases:

- Clase *Cliente.java*: Es la encargada de todas las acciones referentes al cliente, como búsqueda, registro del serial NFC y recarga de saldo. Esta clase define de manera precisa los pasos necesarios para cada operación además de cuales pantallas y mensaje se deben mostrar al usuario

- Clase *Paquetes.java*: Se encarga de toda la lógica para la realización de pago a través del sistema, validando diferentes situaciones como conexión con el punto NFC, hasta casos más simple como el saldo del usuario para cubrir una transacción.

- Clase *ConexionNFC.java*: Esta clase tiene la responsabilidad de todo el proceso para la comunicación con el punto NFC (conexión, solicitud de data y existencia de un dispositivo en contacto con el punto NFC). *ConexionNFC.java* trabaja de la mano con las dos clases antes mencionadas para asegurar una ejecución perfecta del flujo del aplicación

Todas las capas antes descritas necesitan estar comunicadas en términos sencillos para evitar errores dentro de la ejecución del programa, por este motivo se crearon una serie de clases denominadas TO (incluidas en el paquete TO), las cuales tiene las mismas estructuras de las tablas de Base de Datos y sus relaciones, para un mejor manejo de la data a través de las capas

- Paquete *Utils*: Dentro de este paquete se encuentran todas las clases utilitarias, las cuales son llamadas por la aplicación en cualquiera de las capas y en cualquier momento. Estas clases tienen métodos específicos para realizar funciones específicas en un momento dado, sin necesidad de utilizar el objeto completo. Ejemplo de esto es la clase *InsertarImagen.java*, que es la encargada de insertar todas las imágenes e iconos de la aplicación para hacerla más atractiva para el usuario.

- Clase *NFCGeneralException*: Esta clase se encarga del manejo de todos los casos atípicos del sistema y la captura de todos los errores. La intención de diseñar esta clase fue para manejar de manera más cómoda los casos especiales para minimizar el uso de validaciones a banderas.

- *Util.java*: Es clase contiene métodos de validaciones, como los métodos para el envío de correo. Estos métodos solo se usan en casos muy específicos durante la ejecución de la aplicación.

IV.5 Desarrollo de aplicación ANDROID

Es necesario que el usuario pueda comunicarse tanto con el servicio *Web,* para iniciar su registro dentro del sistema y acceder a su cuenta, así como también para poder realizar los pagos de manera efectiva. Por lo tanto la aplicación para dispositivos móviles ANDROID (teléfonos inteligentes y tabletas) es quien permite esta interacción del cliente con el sistema. Ofrece la posibilidad del registro, cambio de contraseña, acceso a la cuenta, acceso a la información almacenada y la posibilidad de pago. La comunicación entre el servicio *web* y la Aplicación ANDROID se realiza usando el formato JSON, para tener la ventaja y la facilidad de escribir el analizador sintáctico (*parser*). A continuación para facilitar el entendimiento del lector se mostraran las *Activity* de la aplicación ANDROID y se explicaran de forma detallada las tareas que las mismas cumplen.

IV.5.1 *Activity* para inicio de sesión

Esta *Activity* permite que el usuario pueda acceder a su cuenta una vez que esté registrado en el sistema. Se requiere el correo y la contraseña del usuario para posteriormente enviarlos al servicio *web* para verificar si estos datos están almacenados en la lista *users* de la Base de Datos (*comprasnfc*). Si se encuentra una coincidencia, el usuario es dirigido a la pantalla principal, en caso contrario se muestra un *Toast* de error en la aplicación indicando al usuario que no hubo coincidencias.

Esta *Activity* utiliza la clase *Asynctask* para comprobar la conexión a internet y a su vez obtener los datos del servidor MySQL y posee los botones para permitir el registro de un nuevo usuario y la opción de recuperar la contraseña.

El proceso de inicio de sesión se lleva a cabo por medio de una petición POST que se realiza desde la aplicación ANDROID hacia el servidor *Web* donde primero se

envían los parámetros: *email* y contraseña en formato JSON, hacia el método *loginUser* de la clase *UserFunctions* que se encuentra en el paquete de librerías, el cual se encarga de colocar en un *array* todos los parámetros que van a ser enviados al servidor *web*. Posteriormente este *array* va al método *getJSONFromUrl* de la clase *JSONParser* junto con el URL del servidor *web*, para que esta clase se encargue de enviar la información al servidor *web*, que en este caso es el *email* y la contraseña, para posteriormente recibir una respuesta verificando si el usuario existe o no en la base de datos.

Cabe destacar que al enviarse la contraseña esta va encriptada con el algoritmo AES para que evitar que alguna persona no deseada la capture.

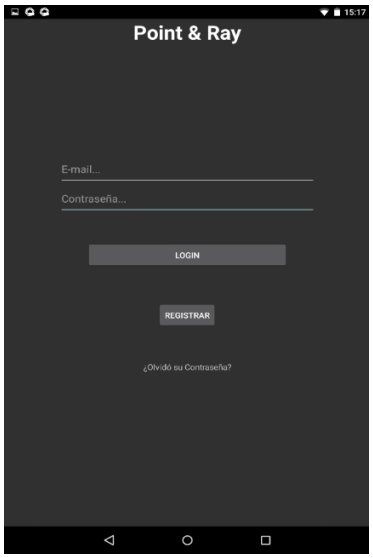


Figura 16. Vista de la ActivityTwo, destinada al inicio de sesión

Fuente: Propia

IV.5.2 *Activity* para registro

Permite al usuario ingresar sus datos personales para la creación de una cuenta en el sistema para que posteriormente pueda iniciar sesión. La *Activity* utiliza la

clase *Asynctask* para realizar una prueba de conexión a internet, una vez culminada la prueba, se envían los datos al servicio *web*, para almacenarlos dentro de la Base de Datos (*comprasnfc*), siempre y cuando no exista un usuario con el mismo correo electrónico. Si el usuario pudo ser registrado, se muestra un *Toast* y además se envía un correo electrónico indicando que el registro fue exitoso, en caso de que el registro no pudo ser completado, alguno de los campos están vacíos o la contraseña no tiene el número de caracteres mínimos se mostrará en la aplicación un *Toast* indicando alguno de estos errores. Esta *Activity* permite a través de la clase *DatabaseHandler*, almacenar los datos enviados en la base de datos interna del dispositivo (*SQLite Database*). La visualización de esta *Activity* se puede apreciar en la figura 17.

El proceso de interacción con la aplicación ANDROID y la base de datos es similar al de inicio de sesión, sólo que en este caso se envían los parámetros del registro: nombre, apellido, nombre de usuario, contraseña y *email*. Después de que el servidor *web* reciba la petición POST este se encarga de almacenar en la base de datos (*comprasnfc*) todos los datos del usuario enviados en formato JSON para luego enviar una respuesta a la aplicación de ANDROID con el mismo formato indicando que el registro fue exitoso.

Al igual que el inicio de sesión en esta etapa también se encripta la contraseña al ser enviada a través de la intranet inalámbrica para evitar que sea vista por terceros.

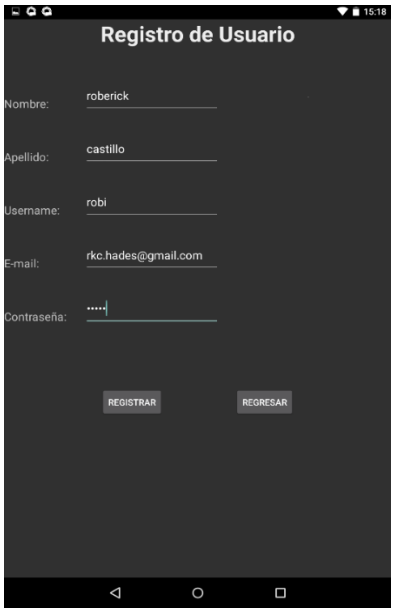


Figura 17. Vista de la ActivityThree, destinada al registro de usuario

Fuente: propia

IV.5.3 *Activity* detalles de registro

Si el registro fue exitoso, en la clase de registro existe una llamada al método (*addUser*) de la clase *DatabaseHandler*, que se encarga de almacenar la información del usuario en la base de datos del dispositivo móvil (*SQLite database*). La *Activity* de detalles de registro muestra la información almacenada en la base de datos del dispositivo móvil del último usuario registrado como se puede observar en la figura 18

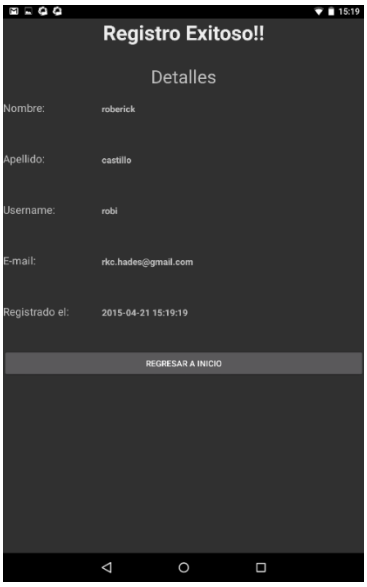


Figura 18. Vista de la ActivityFour, destinada a mostrar los detalles de registro

Fuente: Propia

IV.5.4 *Activity* Olvido de contraseña.

En esta fase el usuario tiene la posibilidad de recuperar su contraseña en caso de olvido para que pueda ingresar de nuevo en su cuenta iniciando sesión con una contraseña válida. Esta etapa consta de dos *Activitys*:

- Introducción del correo: en esta *Activity* es necesario que el usuario ingrese su correo electrónico para verificar que efectivamente posee una cuenta registrada en la base de datos. Se envía una petición POST en la cual se envía el *email* en formato JSON para que se realice la consulta en la base de datos (*comprasnfc*) verificando si el usuario existe. Luego se envía una respuesta en formato JSON al dispositivo mostrando el *Activity* siguiente, en caso que no exista el correo electrónico en la base de datos se envía un mensaje, el cual se muestra en pantalla al usuario indicando que el *email* no existe.

- Introducción de nueva contraseña: luego de verificar la existencia del usuario a través de su correo electrónico se muestra la segunda *Activity* donde el usuario puede ingresar su nueva contraseña para posteriormente iniciar sesión. Al realizar este proceso el servidor *web* envía un correo electrónico al usuario indicando una advertencia que realizó un cambio de contraseña. En este proceso se envía la contraseña encriptada con AES desde la clase *ActivityTwelve* donde luego es enviada al método de *NewPass* de la clase *UserFunction* donde se construye un *array* con la nueva contraseña, el *email* y el URL del servidor *web* para que posteriormente sea enviada a la clase *JSONParser* donde los parámetros son enviados en formato JSON y do nde también se recibe la respuesta del servidor *web* con un mensaje de éxito o de fracaso.

Figura 19. Vista de la ActivityFive, para enviar correo en caso de olvido de contraseña

Fuente: Propia

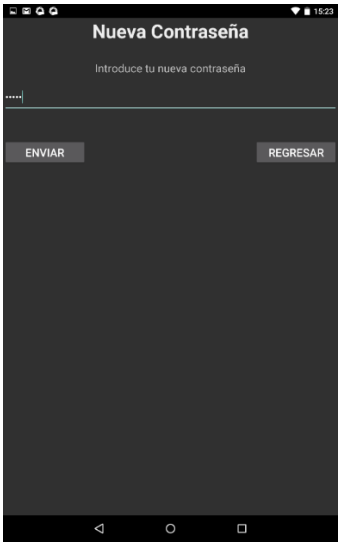


Figura 20. Vista de la ActivityTwelve para enviar correo en caso de olvido de contraseña.

Fuente: Propia

IV.5.5 *Activity* Pantalla principal.

Esta *Activity* muestra la vista principal luego de que el usuario inicie sesión. Esta pantalla consta de 6 botonos y un *TextView* donde muestra el nombre de usuario de la persona que ingresó. Cada uno de estos botones conlleva a una interacción entre la base de datos (*comprasnfc*) y el dispositivo móvil, exceptuando el botón de salir. A continuación se mostrarán los botones y sus funciones:

- Saldo disponible: Muestra una pantalla donde muestra el saldo actual del usuario.
- Factura: En este botón se muestra la última transacción realizada, para que el usuario tenga un mayor control de sus pagos.
- Pagar: Este botón es utilizado al momento de efectuar un pago, donde el dispositivo pasa a emular una etiqueta NFC con la característica que ellas poseen.

- Consultar datos: Este botón muestra una pantalla en la cual se despliega la información de registro del usuario.
- Configuración: Permite realizar cambio de contraseña para brindarle al usuario la libertad de cambiarla cuando lo desee.
- Salir: Retrocede a la pantalla de inicio de sesión.

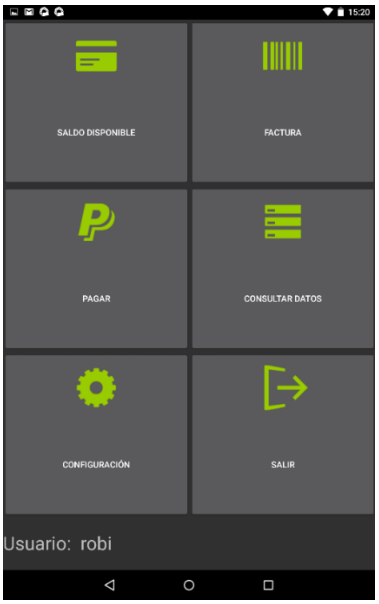


Figura 21.Vista de la ActivitySix, donde se muestra la pantalla principal.

Fuente: Propia

IV.5.6 *Activity* Cambio de contraseña.

Esta *Activity* se creó con la finalidad de que el usuario pueda cambiar su contraseña cuando lo desee, una vez que logre ingresar a la pantalla principal. Es similar a la función del *Activity* de olvido de contraseña sólo que esta no requiere verificar el correo electrónico del usuario debido a que, para que el usuario pueda cambiar su contraseña en este caso, es necesario que inicie sesión. Para lograr esto en la clase *ActivitySeven* se envía una petición POST en formato JSON al servidor *web*, donde se envían los parámetros de la nueva contraseña y el correo del usuario. Luego

la función de cambio de contraseña en PHP se encarga de manejar la consulta en la base datos teniendo como referencia el *email* del usuario y reemplaza el valor de su contraseña actual por la nueva. Para finalizar el servidor envía una respuesta con el mismo formato JSON donde se comprueba a través de un mensaje que el cambio fue exitoso, además de esto también se envía un correo al usuario para mejor administración de su cuenta. En la figura 22 se puede observar el *Layout*:

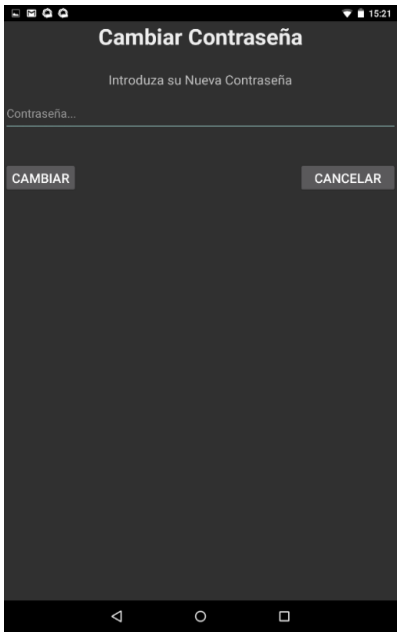


Figura 22. Vista de la ActivitySeven para cambio de contraseña

Fuente: Propia

IV.5.7 *Activity* para Consultar los datos del usuario.

En esta *Activity* se muestran los datos de registro del usuario una vez que inició sesión con su cuenta correspondiente. Para poder llevar a cabo esto hay que destacar que el usuario al iniciar sesión recibe en un formato JSON la respuesta del servidor *web* junto con todos los datos del usuario que agregó al momento de registrarse: nombre, apellido, nombre de usuario, *email* y la hora y fecha del registro. Luego estos

son almacenados en una base de datos (contactos) creada en la clase *DatabaseHandler*, esto permite que estos datos sean extraídos y se pueden consultar posteriormente llamando al método *getUserDetails* el cual retorna un *HashMap* donde se encuentran todos estos datos.

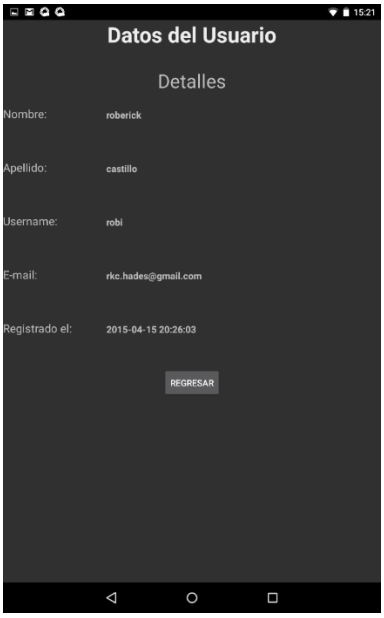


Figura 23. Vista de la ActivityEight para visualizar los datos personales.

Fuente: Propia

IV.5.8 *Activity* para pagar.

Esta *Activity* tiene la finalidad de permitir que el dispositivo emule una etiqueta NFC utilizando las clases *MyHostApduService* y *utils* las cuales pertenecen al paquete de librerías y son utilizadas específicamente para la comunicación entre el dispositivo y el lector NFC ACR122U. En esta etapa del proceso es donde se realizan los pagos enviando el serial NFC asociado al dispositivo.

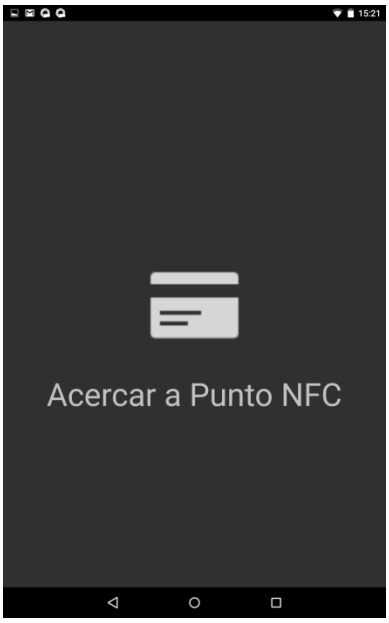


Figura 24. Vista de la ActivityNine para realizar los pagos.

Fuente: Propia

IV.5.9 *Activity* para ver la última transacción.

En esta *Activity* se muestra al usuario la última transacción realizada la cual se logra a través de una petición POST hacia el servidor *Web*, donde se envía como parámetro el número de tarjeta asociada al usuario en un formato JSON para poder realizar la consulta en la base de datos y poder extraer el ultimo monto que registró. Posteriormente es recibido en la clase *JSONParser* para que luego sea enviado al llamar al método *getJSONObject* desde la *ActivitySix* para que finalmente la información sea trasladada por medio de un *bundle* a la *ActivityTen* donde se mostrará el número de la última transacción.

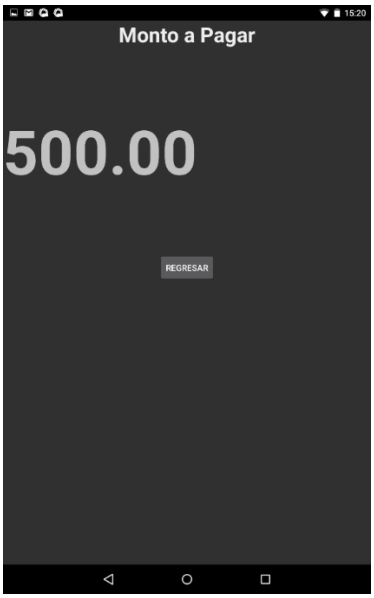


Figura 25. Vista de la ActivityTen para mostrar la última transacción.

Fuente: Propia.

IV.5.10 *Activity* para ver el saldo disponible.

Esta *Activity* muestra el saldo disponible que posee el usuario. El proceso de obtención del dato es similar al anterior, realizando una petición POST al servidor *Web* para que posteriormente haga la consulta en la base de datos pero esta vez con el parámetro id para poder encontrar la fila correspondiente al usuario. Éste retorna el valor del saldo en formato JSON para que posteriormente sea desplegado en la pantalla llamando igualmente que el caso anterior al método *getJSONObject* de la clase *JSONParser* desde la *ActivitySix* para que posteriormente la información sea enviada a la *ActivityEleven* a través de un *bundle* donde dicha información del saldo será desplegada en la pantalla del dispositivo del usuario.

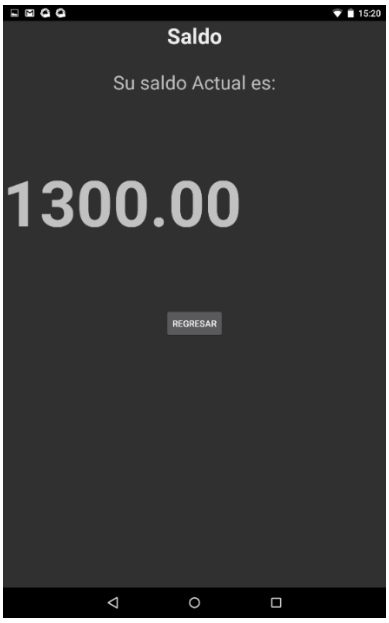


Figura 26. Vista de la ActivityEleven para mostrar el saldo.

Fuente: Propia.

IV.5.11 Paquete de librerías.

Este paquete de librerías contiene una serie de clases que fueron indispensables para la realización de alguna de las funciones mencionadas anteriormente, desde el envío y recepción de datos desde y hacia el servidor *web* mediante el formato JSON, el manejo de la base de datos interna del dispositivo (*SQLite Database*) para introducir y extraer los datos del usuario como también la realización de los pagos y la seguridad implementada para encriptar las contraseñas.

Las clases encargadas de todos estos procesos son los siguientes:

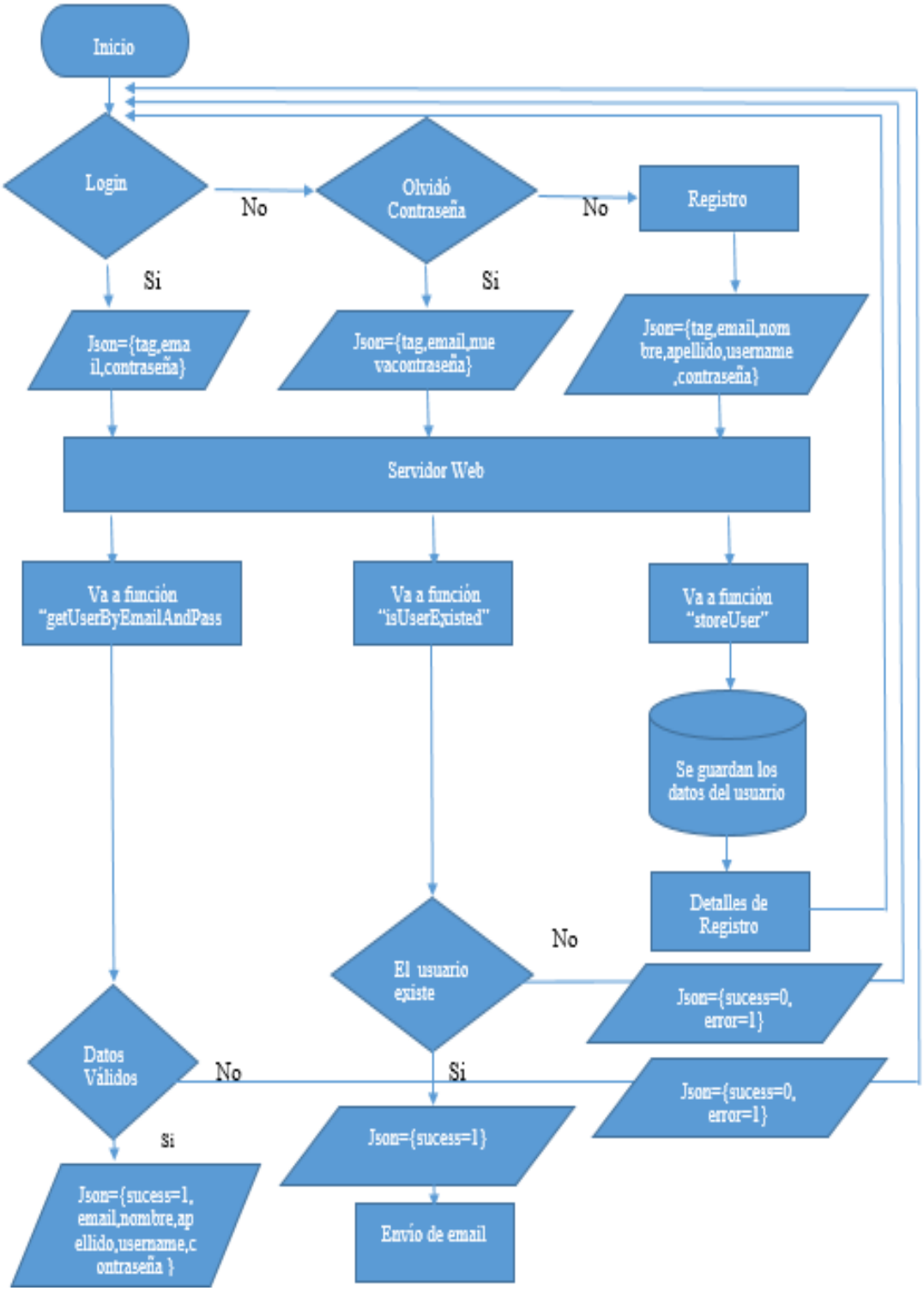
- *DatabaseHandler.java*
- *JSONParser.java*
- *MyHostApduService.java*
- *Seguridad.java*

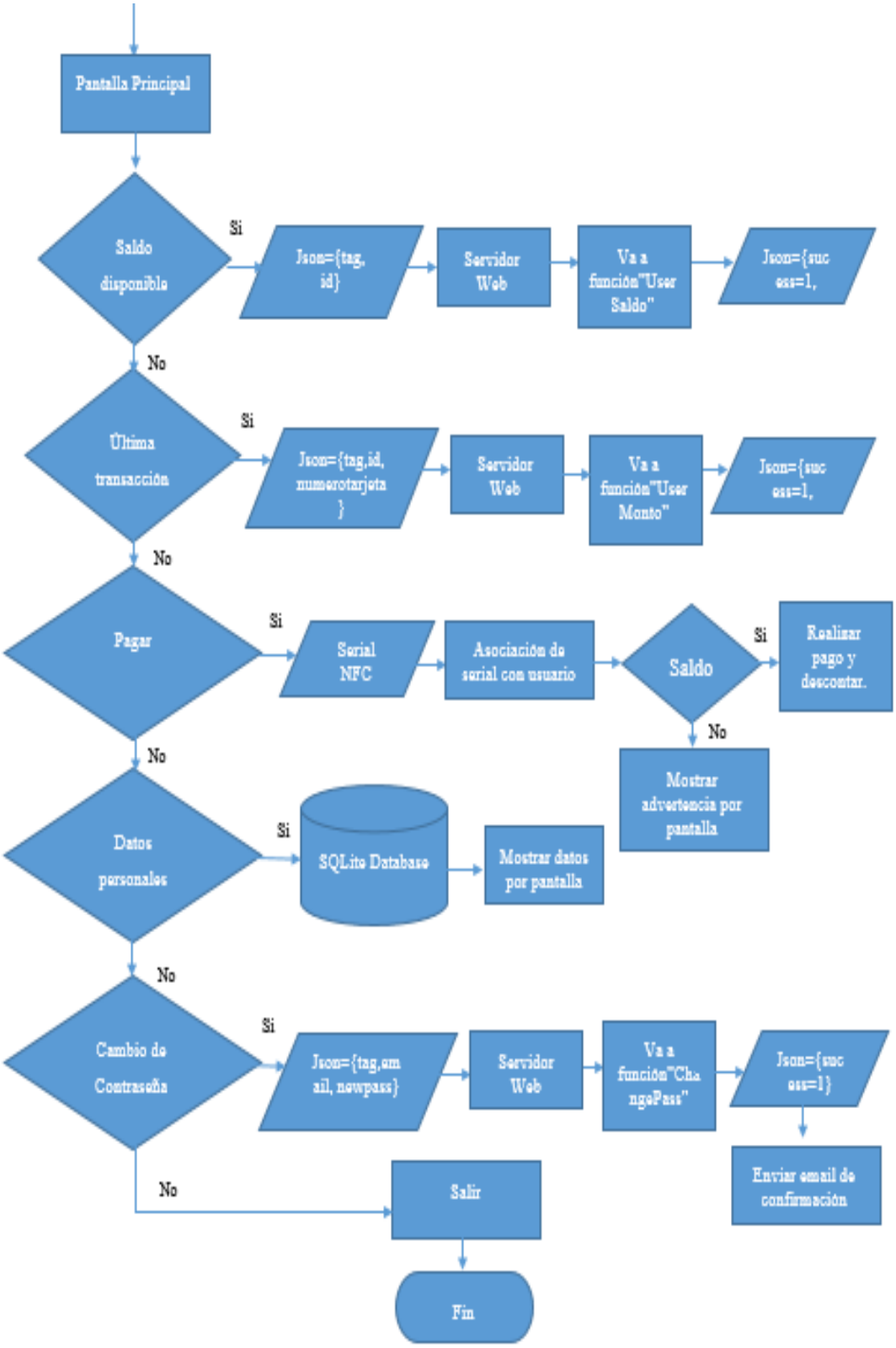
- *UserFunctions.java*
- *Utils.java*

IV. 6 Pruebas del sistema

En este apartado se muestran las pruebas que se hicieron al sistema para verificar el correcto funcionamiento del mismo, se anexan capturas de pantallas del funcionamiento tanto de la aplicación, base de datos y de la aplicación ANDROID trabajando en conjunto, con la intención de demostrar al lector el logro de los objetivos establecidos.

A continuación se muestra un diagrama de flujo del sistema desarrollado. Iniciado desde la aplicación en ANDROID


Inicio
Login
No
Olvidó Contraseña
No
Registro
Si
Si
Json={tag,email,nombre,apellido,username,contraseña}
Json={tag,email,nuevacontraseña}
Json={tag,email,contraseña}
Servidor Web
Va a función "getUserByEmailAndPass
Va a función "isUserExisted"
Va a función "storeUser"
Se guardan los datos del usuario
Detalles de Registro
No
El usuario existe
Json={sucess=0, error=1}
Datos Válidos
No
Si
Json={sucess=0, error=1}
Si
Json={sucess=1}
Json={sucess=1, email,nombre,apellido,username,contraseña }
Envio de email


Pantalla Principal
Saldo disponible
Si
Json={tag, id}
Servidor Web
Va a función"User Saldo"
Json={success=1,
No
Última transacción
Si
Json={tag,id, numerotarjeta}
Servidor Web
Va a función"User Monto"
Json={success=1,
No
Pagar
Si
Serial NFC
Asociación de serial con usuario
Saldo
Si
Realizar pago y descontar.
No
Mostrar advertencia por pantalla
No
Datos personales
Si
SQLite Database
Mostrar datos por pantalla
No
Cambio de Contraseña
Si
Json={tag,email, newpass}
Servidor Web
Va a función"ChangePass"
Json={success=1}
Enviar email de confirmación
No
Salir
Fin

El funcionamiento del sistema es realmente sencillo, como se observa en el diagrama anterior, el *login* del usuario es necesario para permitir el acceso al menú principal en la aplicación ANDROID. Los usuarios nuevos, tienen la posibilidad de registrar sus datos para crear una cuenta personalizada dentro del sistema. Una vez realizada esta acción, el usuario recibe un correo electrónico confirmando esta operación. También se ofrece la posibilidad de cambiar la contraseña en caso de olvido.

Esta comunicación se logra vía Wi-fi, permitiendo la comunicación del dispositivo móvil con el servicio *web*, el cual tiene acceso a la base de datos y puede identificar a los distintos usuarios registrados en el sistema, de esta manera se verifica la información recibida y se envía un mensaje de confirmación al dispositivo móvil, permitiendo el ingreso del usuario a la pantalla principal. El servicio *web* también actúa al momento del registro de un usuario nuevo y en los cambios de contraseña como se explicó en el apartado Desarrollo de la aplicación servidor.

Una vez que el usuario ha accedido a la pantalla principal, se requiere que el usuario acerque el dispositivo móvil al lector NFC, permitiendo la captura del serial NFC y posteriormente asociarla al usuario. Una vez finalizado el registro, se ofrece la posibilidad de consultar información personal de la cuenta, el saldo disponible y la última transacción realizada, esta información también es recibida gracias al servicio *web*.

Para pagar a través del sistema es realmente sencillo, se debe verificar el monto a cancelar en la aplicación principal y se debe acercar el teléfono inteligente al lector NFC cuando sea indicado por parte de las aplicaciones, una vez cumplido estos pasos, la transacción habrá finalizado y se envía la información de la factura al usuario a través de un correo electrónico.

A continuación se muestran los resultados obtenidos de las pruebas realizadas para cumplir con los objetivos planteados.

IV. 6.1 Prueba de conexión del cliente al servidor

Para demostrar esta etapa del sistema, se requirió el registro de un usuario nuevo por lo tanto en la figura 27 se aprecia los datos de un cliente registrando sus datos desde la aplicación ANDROID. Es importante mencionar que la comunicación es local, por lo tanto es necesario que los dispositivos involucrados estén dentro de la misma red del servidor.

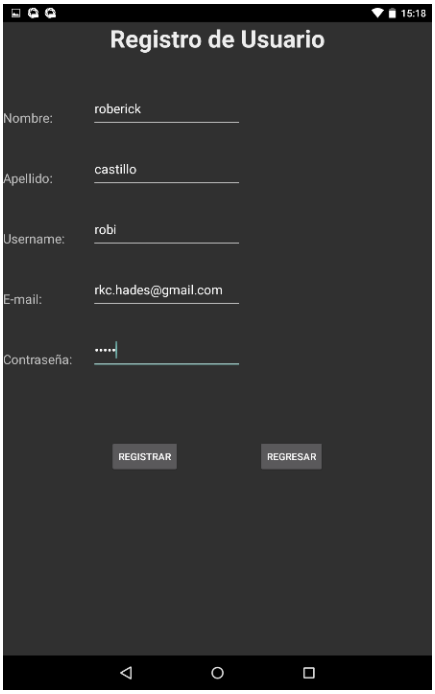


Figura 27. Registro desde ANDROID.

Fuente: Propia

En la figura 28, se demuestra la captura de los paquetes que se intercambiaron entre el servidor y la aplicación ANDROID.

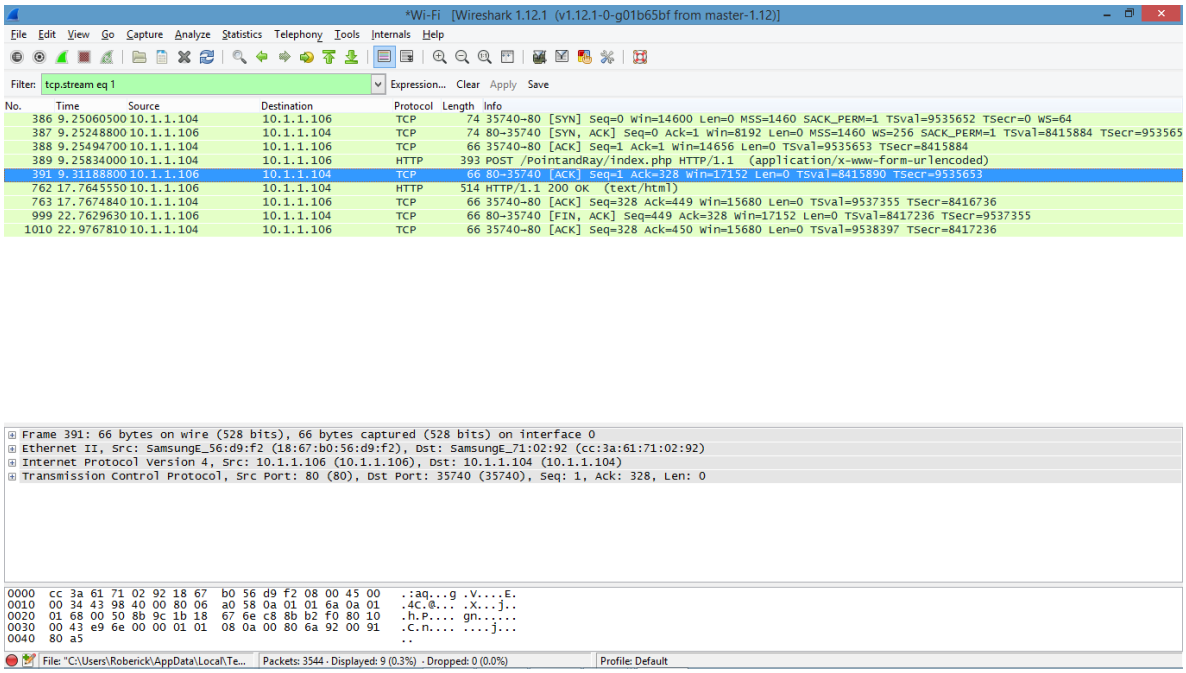


Figura 28. Captura de Wireshark. Intercambio de paquetes en registro

Fuente: Propia

Como se mencionó en apartados anteriores, el servicio *web* utiliza el protocolo TCP, en la figura 28 se observa claramente que dicho protocolo es el que opera durante esta comunicación, y en la figura 29 se observan los datos que el cliente envía al servidor, el mensaje se ve en texto plano, sin embargo, tomando en cuenta la seguridad del sistema, la contraseña está encriptada. Esta encriptación es válida para todo intercambio de información donde la contraseña del usuario esté involucrada, es decir, para el inicio de sesión y el cambio de contraseña. En las figuras 30 y 31 se aprecia esta afirmación.

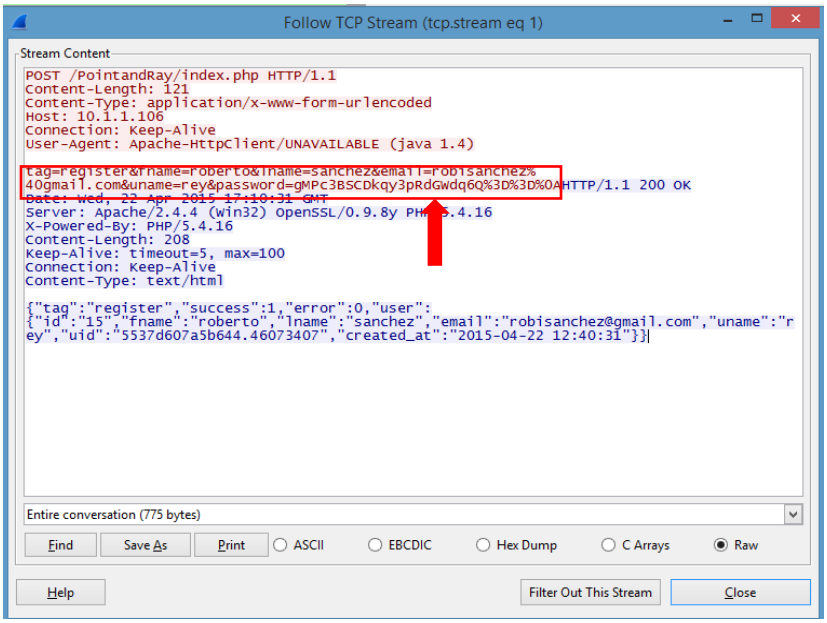


Figura 29. Datos encriptados de registro.

Fuente: Propia

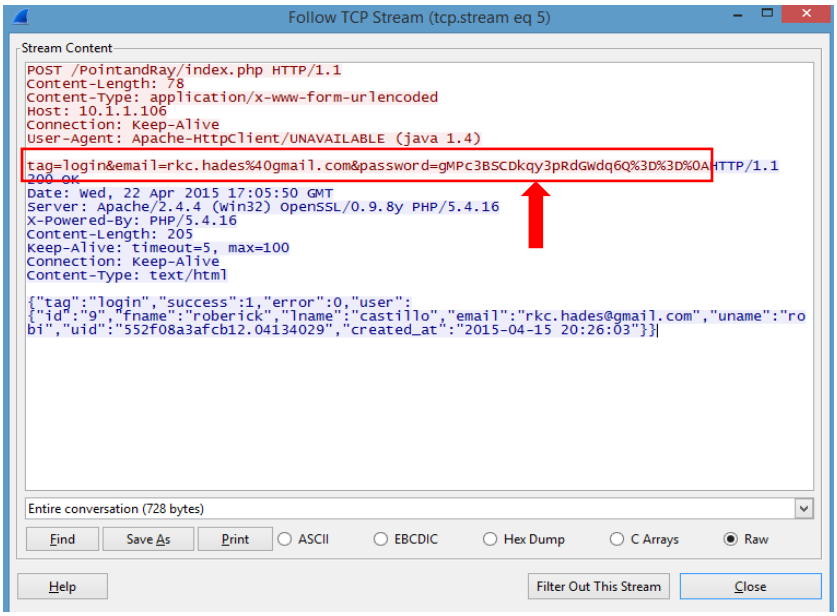


Figura 30. Datos encriptados en el login.

Fuente: Propia

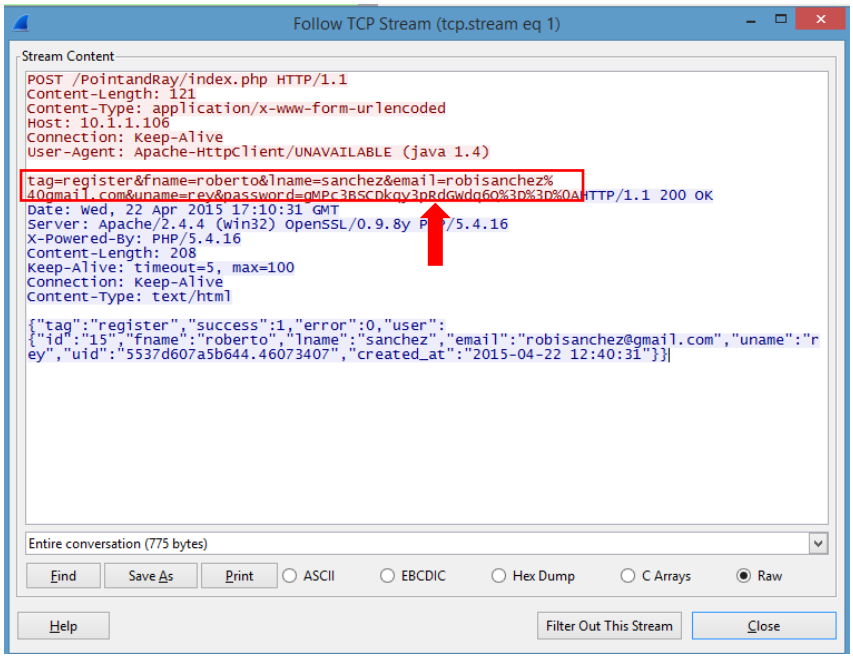


Figura 31. Datos encriptados en el cambio de contraseña

Fuente: Propia

IV. 6.2 Respuestas del servidor al cliente

Una vez que el servidor recibe cualquier petición del cliente es necesario una respuesta para indicar el correcto funcionamiento del sistema, por lo tanto en las figura 27 se observa como la *ActivityFour*, es la encargada de demostrar que el registro es exitoso, en las capturas de WIRESHARK de la figura 28, se observa que el servidor (con dirección IP: 10.1.1.106) responde al dispositivo móvil, por lo tanto se puede afirmar que las respuestas del servidor al cliente fueron exitosas. Este intercambio de información es vital para el sistema, puesto que si el dispositivo móvil no recibe respuesta alguna, la aplicación ANDROID no permite que el usuario interactúe dentro de la aplicación principal.

IV 6.3 Transferencia de datos a través de NFC.

En este apartado se demostrarán los resultados a través de la plataforma NFC. Esta comunicación en necesaria para capturar el serial NFC del dispositivo móvil. En la figura 32, se aprecia claramente como la aplicación principal captura el serial NFC del dispositivo móvil. La comunicación con el lector, tanto para recibir instrucciones como para responder a las mismas, es en formato hexadecimal. La conexión entre el lector NFC y la computadora donde se está ejecutando la aplicación es por vía USB, por lo tanto es necesario que la aplicación escuche el puerto y capture los datos solo cuando sea necesario. La comunicación entre el dispositivo móvil y el punto NFC es vía inalámbrica, a una distancia de aproximadamente 3 centímetros.

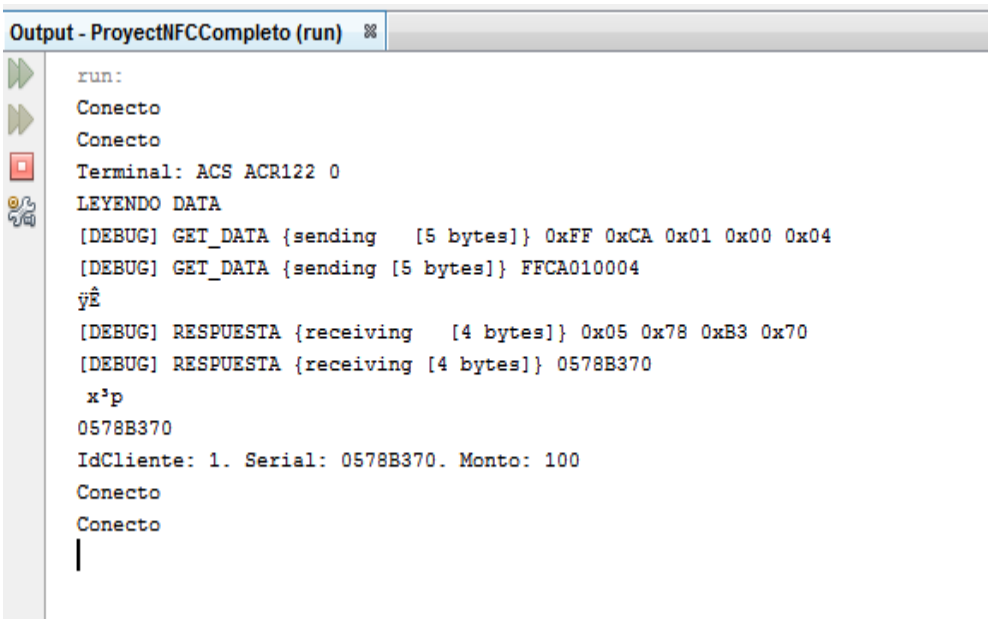


Figura 32. Captura de datos por parte de la aplicación principal a través de NFC. Ejemplo especifico de recarga de saldo a un usuario

Fuente: propia

IV.6.4 Consultas y modificaciones automáticas de la base de datos.

Para el correcto funcionamiento de la aplicación se requiere que la base de datos esté completamente actualizada. Por los tanto, las modificaciones a los registros de las tablas deben efectuarse a gran velocidad puesto que, en caso de que el usuario o la aplicación procedan a efectuar una consulta, no existan errores en los datos insertados.

Por este motivo se hicieron registros y pagos para constatar que estos datos realmente eran insertados dentro de la base de datos y que la aplicación puede lograr consultar las tablas de manera correcta, en las figuras 33 y 34, se observan los datos dentro de la lista usuarios y una consulta desde la aplicación de principal para verificar que ambos coincidan.

En la figura 35 y en la figura 36, se evidencian los registros de las transacciones durante un rango de fechas determinado desde la aplicación principal y los datos almacenados en la tabla transacciones de la base de datos.

Por lo tanto, se puede constatar el correcto funcionamiento de las aplicaciones al momento de realizar consultas y modificaciones a la base de datos, de esta manera se asegura, que el sistema no tiene error alguno cuando se refiere a la comunicación con la base de datos.

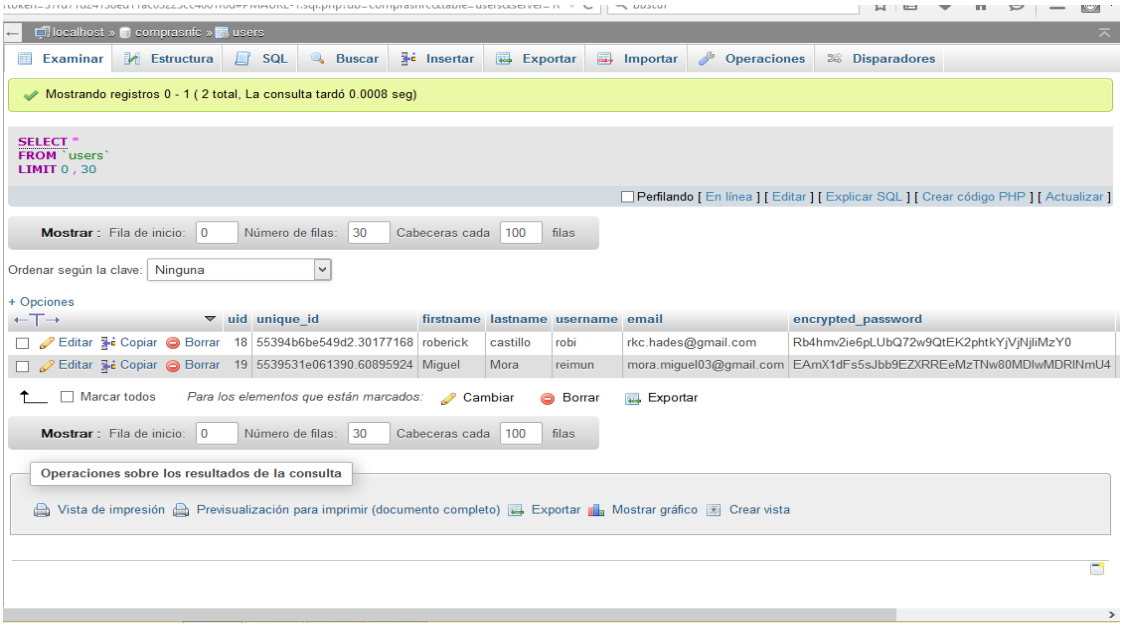


Figura 33. Tabla users de la Base de datos comprasnfc.

Fuente: Propia

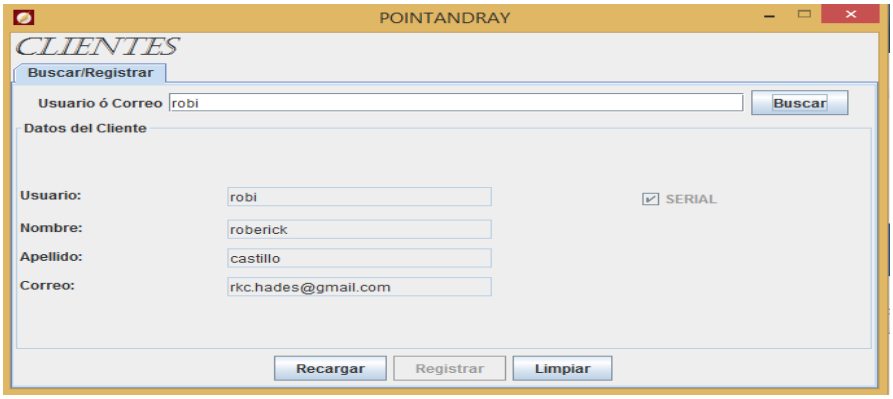


Figura 34. Consulta a los datos de un usuario desde la aplicación principal.

Fuente: Propia.

Figura 35. Tabla transacciones de la base de datos comprasnfc.

Fuente: propia

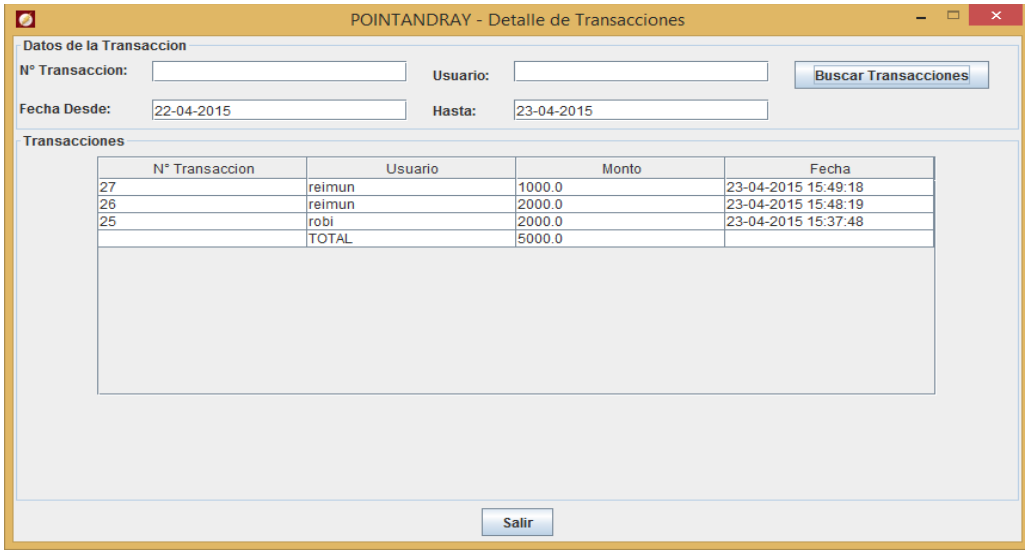


Figura 36. Consulta de transacciones realizadas en un rango de tiempo desde la aplicación principal

Fuente: propia

IV.6.5 Descuento final al cliente

Al realizar una transacción nueva la aplicación de ANDROID a través del punto NFC se comunica automáticamente, por lo tanto esta es la prueba total del sistema. Para esta prueba se usó al usuario *robi*, que se encuentra previamente registrado en el sistema con un saldo de 9000 como se observa en la figura 37:

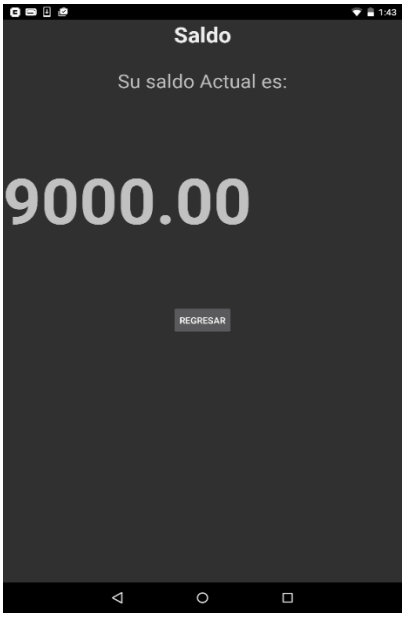


Figura 37.Saldo disponible del usuario robi. Consulta desde la aplicación ANDROID

Fuente: Propia

Desde la aplicación principal, se procede al cobro de una factura por un monto de 2000, como se observa en la figura 38:

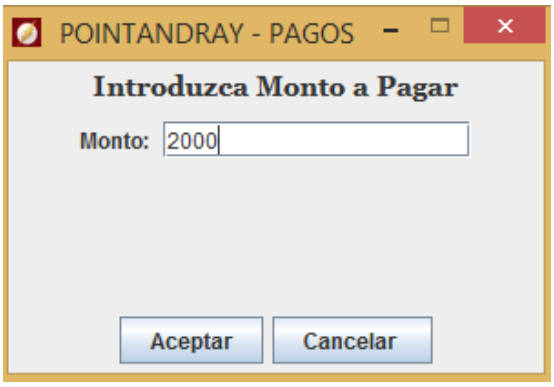


Figura 38. Ventana para introducir el monto de la factura a cobrar

Fuente: Propia

La aplicación espera hasta que se acerque el dispositivo NFC al lector NFC y detecte la conexión para el intercambio de datos, la venta que indica este proceso se muestra en la figura 39:

Figura 39. Ventana para informar la espera de la conexión del dispositivo móvil con el lector NFC

Fuente: Propia

Posteriormente cuando el descuento ha sido éxito se informa a través de una ventana para indicar el fin de la transacción, como se observa en la figura 40.

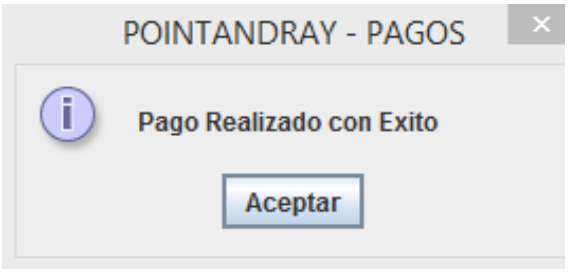


Figura 40. Ventana para indicar el fin de la transacción

Fuente: Propia

Para que el usuario pueda constatar el monto de la factura debitada, el mismo puede pulsar el botón de *factura* en la aplicación ANDROID para observar el monto de la última transacción. Se verifica que el dígito que arroja esta *Activity* es el mismo que se cobró en la aplicación principal de JAVA, como se muestra en la figura 41.

Figura 41.Consulta del monto cancelado en última transacción desde la aplicación ANDROID.

Fuente: Propia

Por último, se consulta el saldo disponible del usuario una vez realizada la operación anterior, para esto se presiona el botón de *Saldo,* destinado para esta función desde la aplicación ANDROID. Ver figura 42.

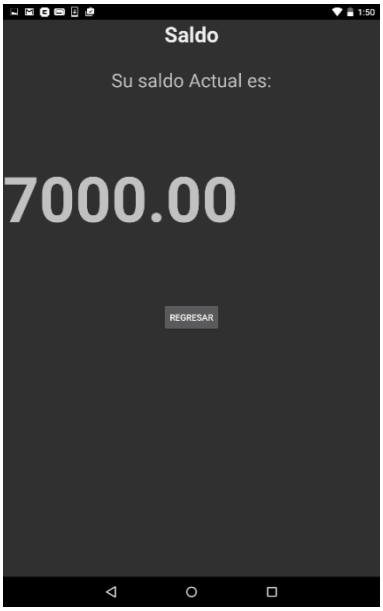


Figura 42. Consulta del nuevo saldo disponible después de finalizar la transacción.

Fuente: Propia

Desde la aplicación principal JAVA se puede hacer una consulta a las transacciones para constatar el funcionamiento del sistema por completo. La figura 43 muestra una consulta de la aplicación principal, para verificar el resultado final de la transacción.

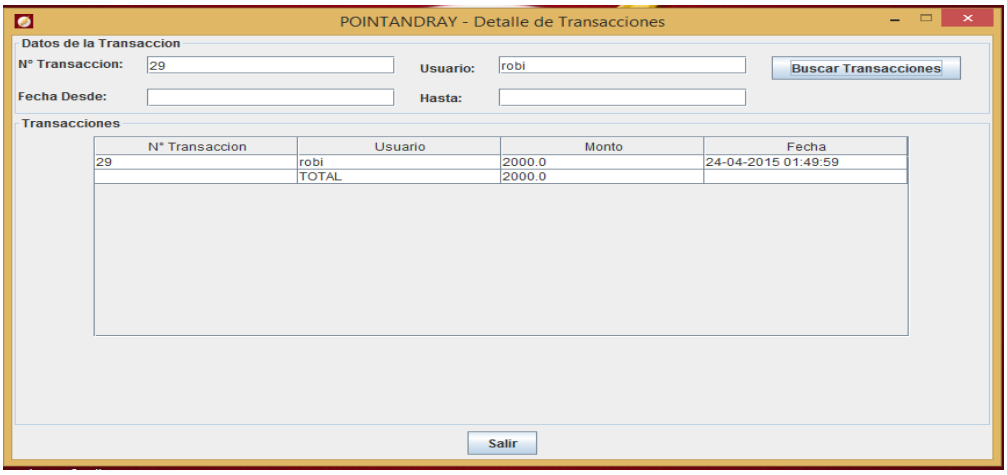


Figura 43. Consulta de la transacción que se realizó como ejemplo

Fuente: Propia

El tiempo requerido para que un usuario nuevo se registre y proceda a un pago durante las pruebas experimentales, nunca fue superior a un minuto y si el usuario se encuentra previamente registrado e ingresó al menú principal de la aplicación ANDROID el tiempo para las transacciones no superó los 20 segundos, sin embargo .este tiempo puede ser mayor o menor dependiendo de la velocidad de la conexión a internet, puesto que el envío de los correos electrónicos que ofrece el sistema como información al usuario genera un leve retraso para finalizar la transacción por completo, sin embargo las modificaciones e inserciones de datos a la base de datos, ya sea a través del servicio *web* o a través del lector NFC son extremadamente rápidas.

Estas pruebas demuestran que el sistema funciona como se planificó en un principio, ofreciendo resolver el problema planteado, por lo tanto, esta alternativa para pagos electrónicos es una propuesta interesante para ser desarrollada y tomada en cuenta para su masificación e inclusión en locales comerciales.

CAPÍTULO V

Conclusiones y Recomendaciones

Una vez elaboradas todas las actividades programadas para cumplir con los objetivos planteados, se presentan a continuación las conclusiones y recomendaciones obtenidas al finalizar el Trabajo Especial de Grado.

V.1 Conclusiones.

Al culminar el Trabajo Especial de Grado, se puede concluir en primera instancia que a pesar de que la tecnología NFC tiene aproximadamente 11 años de haberse inmerso en el mundo de las Telecomunicaciones aún no posee suficiente popularidad en el mercado, y más aún en Venezuela. Sin embargo, gracias a la investigación realizada se pudo demostrar que la tecnología NFC posee muchas aplicaciones en la actualidad, las cuales se pueden implementar bajo cualquier circunstancia y en cualquier lugar mediante el uso de etiquetas y algún lector NFC que sea capaz de interactuar con dichas etiquetas.

Para cumplir con el primer objetivo de diseñar una aplicación ANDROID para transmitir información desde el dispositivo móvil hacia el punto NFC, se realizaron 12 interfaces gráficas o *layouts que van desde activity_activity_one hasta activity_activity_twelve, donde cada una posee una clase en JAVA: ActivityOne, ActivityTwo, ActivityThree, ActivityFour, ActivityFive, ActivitySix, ActivitySeven, ActivityEight, ActivityNine, ActivityTen, ActivityEleven y la ActivityTwelve.* El usuario puede interactuar con cada uno de los *layouts* a través de botones, *EditTexts, TextViews, ImageButtons, ProgressBar y ProgressDialog*. En particular, la *activity_activity_one es* un *Splash* mostrando un logo por 8 segundos utilizando la clase *AsyncTask*, lo cual le da mayor calidad de diseño y buena presentación a la aplicación. Además, la aplicación también cuenta con 6 clases de librerías:

DatabaseHandler, JSONParser, MyHostApduService, seguridad, UserFunctions, utils; las cuales son de suma importancia debido que a través de ellas se logran actividades como: seguridad al enviar contraseñas vía WIFI, almacenamiento de datos del usuario en base de datos en *SQLite Database*, emulación del dispositivo como tarjeta NFC, etiquetar y colocar en un *array* todos los parámetros que son enviados al servidor *Web* en un *JSONObject* y el envío y recepción de datos hacia el servidor *Web* en formato JSON.

Este Trabajo Especial de Grado se centró en la tecnología NFC aplicada hacia pagos electrónicos, donde gracias a la API 19 de ANDROID se pudo utilizar la *Tablet* "NEXUS 7" y el móvil "SAMSUNG GALAXY S4" como emuladores de etiquetas, para poder establecer una comunicación entre los dispositivos y el Lector NFC ACR122U. Sin embargo, hay que destacar que las comunicaciones realizadas en el sistema no son exclusivamente vía NFC, es por esto que dichas comunicaciones se dividen en dos partes: Usuario/WIFI y Usuario/NFC.

Para la comunicación Usuario/WIFI se utiliza la red inalámbrica WIFI para el intercambio de datos entre la Aplicación ANDROID y el Servidor *Web* APACHE, siendo los datos que viajan a través de esta vía: *email*, contraseña, nombre, apellido, nombre de usuario, fecha y hora de registro, saldo disponible, último pago realizado y un identificador denominado "tag". El objetivo de utilizar esta red es mantener un control de todos los datos suministrados por el usuario en una base de datos, donde el servidor *web* se encarga de realizar las consultas. En cambio, la comunicación Usuario/NFC se encarga solamente del pago que realiza el usuario, en dicho proceso se ven involucrados la aplicación principal, el lector NFC y el dispositivo móvil o *Tablet*. Para lograr esta comunicación se envía un mensaje a través del terminal NFC al dispositivo móvil para captura su serial NFC, de esta manera se identifica al usuario que está realizando la transacción, posteriormente se verifica si posee saldo suficiente para cancelar la factura. Si el proceso se realiza de manera exitosa se envía

un correo electrónico al usuario enviando los datos de la transacción, en caso contrario, se muestra un mensaje de error por pantalla indicando: Saldo insuficiente, Serial no asociado a ningún usuario o el serial no puedo ser capturado.

Para lograr el segundo objetivo de diseñar una aplicación que permita el registro de usuarios y almacenar la información personal de los mismos, se realizaron cuatro *scripts* basados en el lenguaje PHP, los cuales son: *index.php, config.php, DB_Connect.php y DB_Functions.php*. Estos *scripts* realizan las funciones de conexión, consulta y cierre de la base de datos *comprasnfc* para poder registrar a los usuarios una vez que hayan creado enviado su información desde el dispositivo móvil.

Para cumplir el objetivo de diseñar una aplicación capaz de obtener los datos recibidos a través del punto NFC, se realizó una aplicación en JAVA a través del entorno de desarrollo NETBEANS, capaz de captar los parámetros transmitidos por el lector NFC en números hexadecimal al acercar el dispositivo móvil, para posteriormente extraer el serial NFC del dispositivo y asociarlo al usuario correspondiente previamente registrado desde la aplicación en ANDRIOD.

Para cumplir con el objetivo de seguridad en la comunicación con el servidor *web*, donde los datos viajan vía inalámbrica a través de la red WIFI, se encripta la contraseña del usuario con el algoritmo AES desde la aplicación ANDROID y posteriormente, es encriptada nuevamente con el algoritmo SHA1 desde el script en PHP *DB_Functions.php* para ser almacenada en la base de datos, siendo difícil identificar la contraseña del usuario por un analizador de tráfico. Esto se realizó con la finalidad de que sólo el usuario pueda saber su contraseña y ésta no sea vista por una tercera persona.

Para finalizar, la prueba realizada con el usuario de ejemplo fue completamente exitosa, y su duración total fue de aproximadamente 15 segundos. Hay que destacar

que una vez que el usuario registre sus datos y el serial del dispositivo este tiempo baja considerablemente a 7 segundos.

V.2 Recomendaciones.

Para futuros proyectos relacionados con este Trabajo Especial de Grado se recomienda, en caso de utilizar un lector NFC similar al ACR122U, obtener el SDK proporcionado por la empresa ACS, puesto que facilitará el desarrollo del sistema llevando a cabo una comunicación exclusivamente vía NFC.

Es importante llevar este sistema de pago a un nivel macro, incluyendo a empresas bancarias o financieras, que puedan brindar su apoyo para realizar transacciones reales mediante la moneda nacional, bajo la tutela de un supervisor de algún local o establecimiento.

La utilización de un Servidor *Web* local es una gran limitante, debido a que sólo se puede efectuar la comunicación en una intranet con un router inalámbrico, el cual brinda a los usuarios acceso a la red, lo que es ideal para locales pequeños. Es por esto que se recomienda implementar un Servidor *Web* público para poder ampliar el área de comunicación entre los usuarios y el establecimiento, además de poder ser aplicado en franquicias comerciales.

Se recomienda optimizar la aplicación en JAVA para que ésta pueda interactuar con distintos tipos de etiquetas NFC, además de permitir que todo el intercambio de información entre el usuario y la base de datos sea netamente vía NFC y no depender del uso de otras tecnologías inalámbricas como WIFI para llevar a cabo el proceso de transmisión. Esto conduce a lograr realizar una comunicación dúplex entre el dispositivo del usuario y el lector/escritor NFC, para que el usuario pueda obtener respuestas de las transacciones en tiempo real y todo el intercambio de datos sea solamente vía NFC.

BIBLIOGRAFÍA

Alberca Gómez, E. L. (2013). *Estudio de la Tecnología inalámbrica NFC y sus Aplicaciones en el ámbito de las telecomunicaciones* . Quito.

Alvarez. (2013). *Foxit.* Obtenido de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/alvarez_b_c/capitulo1.pdf

Alvarez, R. (2014). *Redes Inalámbricas.* Ciudad de Mexico.

Androcode. (16 de mayo de 2013). *Androcode*. Obtenido de http://www.androcode.es

Aranaz Tudela , J. (2009). *DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MOVILES SOBRE LA PLATAFORMA ANDROID DE GOOGLE.* UNIVERSIDAD CARLOS III DE MADRID .

Azpitarte, R., Jordá, P., & Llinares, J. (2009). *Introducción a la programación orientada a objetos con java.* Valencia: Universidad Politécnica de Valencia.

Azpitarte, R., Jordá, P., Llinares, J., De Elias, E., Ruiz, M., & Torres, F. (2009). *Introducción a la programación orientada a objetos con java.* Universidad Politécnica de Valencia.

Broseta Gutiérrez, R. (2012). *DISPOSITIVOS MÓVILES Y NFC APLICADOS AL CANJEO DE TICKETS: BeepVip.* Valencia.

Cabrera, H. (17 de 4 de 2014). *EcuRed*. Obtenido de http://www.ecured.cu/index.php/Servidor_HTTP_Apache

Camacho, R. (18 de abril de 2012). *Interfaz de programación de aplicaciones (API)*. Obtenido de http://rcmdispmoviles.blogspot.com

Chavarría , D. (2011). Tecnología de Comunicación de Campo Cercano (NFC) y sus Aplicaciones. Universidad de Costa Rica.

Condesa. (11 de julio de 2011). *Androideity*. Obtenido de www.androideity.com

Deitel, P. J., & Deitel, H. M. (2008). *Java cómo programar. Septima Edición.* Mexico: Pearson Education.

Dornin, L., Mednieks, Z., Meike, B., & Nakamura, M. (2012). *Programing Android.* Estados Unidos de América: O´reilly.

Evans, k. (23 de octubre de 2014). *Info Merchant*. Obtenido de http://dev.infomerchant.net/creditcardprocessing/index.html

Forouzan, B. A. (2006). *Transmisión de datos y redes comunicacionales.* New York, Estados Unids de América: McGraw Hill.

Gomez, P. M. (2008). *USB.*

Hernandez, M. J. (2003). *Database Design for Mere Mortals.* Estados Unidos de América: Addison Wesley.

Joyanes, L., & Fernández, M. (2003). *Java 2 Manual de programación.* Madrid: McGraw-Hill.

Llamas, R. (15 de febrero de 2014). *IDC Analyze the Future*. Obtenido de http://www.idc.com/prodserv/smartphone-market-share.jsp

Maldonado, D. M. (20 de Abril de 2008). *empresa&economia*. Obtenido de http://www.empresayeconomia.es

Melián Montalvo, M. (2008). *XML. El nuevo lenguaje universal*. Obtenido de www,bibliociencias.com

Microsoft. (Enero de 2005). *Microsoft*. Obtenido de https://msdn.microsoft.com/es-es/library/cc737738%28v=ws.10%29.aspx

Muñoz, A. (2010). *API de JAVA.* Ciudad de Panamá.

MySQL. (1997). *MySQL*. Recuperado el 14 de Junio de 2014, de MySQL the world´s most popular open sourse data base: http://dev.mysql.com/doc/refman/5.0/es/features.html

NetBeans. (2013). *Home: NeatBeans*. Recuperado el 18 de Junio de 2014, de NetBeans: https://netbeans.org/about/index.html

NFC Forum. (11 de Julio de 2012). NFC Analog Specifications. Estados Unidos de América.

Ortega Martinez, H. I. (19 de noviembre de 2013). *Que es Wamp Server*. Obtenido de http://ingenieros.wordpress.com

php. (s.f.). *php*. Obtenido de http://php.net/manual/es/intro-whatis.php

rafaelma. (2 de Octubre de 2010). *Postgre SQL*. Recuperado el 10 de Junio de 2014, de Postgre SQL: http://www.postgresql.org.es/sobre_postgresql

Ruiz, A. C. (2011). *DESARROLLO DE UNA APLICACIÓN DE PAGO A TRAVÉS DE LA TECNOLOGÍA NFC.* Madrid.

TANENBAUM, A., & WETHERALL, D. (2012). Redes de Computadoras. En T. Wetherall. Mexico: Pearson.

Tomas, J. (2011). *Tutoriales Android Fundamentos*. Recuperado el 21 de Marzo de 2014, de Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles por la Universidad Politécnica de Valencia: http://www.androidcurso.com/index.php/tutoriales-android-fundamentos

Vacas, J. (11 de 12 de 2014). *academia android*. Obtenido de http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/

DESARROLLO DE UN SISTEMA TRANSACCIONAL QUE PERMITE REALIZAR PAGOS ELECTRONICOS IMPLEMENTANDO NFC

Roberick E. Castillo S., Miguel A. Mora H.

Escuela de Telecomunicaciones, Universidad Católica Andrés Bello

Caracas, Venezuela.

rkc.hades@gmail.com

mora.miguel03@gmail.com

***Abstract*─** En búsqueda de dar solución a los problemas que presentan las formas de pagos actualmente utilizadas, como el retardo que normalmente presentan las transacciones a través de puntos de ventas o el riesgo de ser víctima de robo de identidad, se concibió la idea de desarrollar un sistema que sirva como alternativa para realizar pagos electrónicos implementando la tecnología NFC, con el fin de mitigar los problemas antes descritos.

Para alcanzar los objetivos planteados, se siguieron una serie de pasos pautados en la metodología, que estipuló primero una investigación documental. Posteriormente, se desarrolló un sistema constituido por tres aplicaciones: Aplicación para dispositivos móviles ANDROID, que permite la comunicación del usuario con el sistema. Aplicación servidor, programada en PHP la cual, gracias a un servicio *web*, establece la conexión del usuario con el sistema a través de una red local WIFI. Por último, una aplicación principal, desarrollada en JAVA, encargada de recibir la información del dispositivo móvil mediante el lector NFC para realizar las transacciones. El correcto funcionamiento del sistema depende de una base de datos que almacene la información del cliente y las transacciones realizadas a través del sistema, por lo tanto, se usó el lenguaje SQL para la creación de la base de datos**.**

Las pruebas realizadas al sistema demostraron el correcto funcionamiento del mismo, ofreciendo rapidez en la realización de las transacciones, gracias a la implementación de la tecnología NFC, confiabilidad, gracias a los módulos de seguridad diseñados y facilidad de uso, gracias a las interfaces gráficas. Estas pruebas son base para afirmar que el sistema puede ser usado como alternativa de pago en locales comerciales, alcanzando los objetivos trazados y solucionando la problemática planteada.

Palabras claves: ANDROID, NFC, bases de datos, sistema automatizado.

I. INTRODUCCIÓN

En este tomo se refleja el resultado de las investigaciones y procedimientos aplicados para lograr alcanzar los objetivos propuestos, se ofrece una estructura de

cinco capítulos, titulados de la siguiente manera: Planteamiento teórico, Marco teórico, Metodología, Resultados y Conclusiones y recomendaciones. Estos capítulos buscan describir de manera fiel la problemática enfrentada y la justificación del proyecto, resaltando los objetivos planteados, el contenido teórico necesario para sustentar la investigación realizada para la elaboración de este tomo, la metodología utilizado y las actividades que se realizarán para el logro de los objetivos, los resultados que arrojaron al concluir estas actividades y por último las conclusiones que se obtuvieron al finalizar el Trabajo Especial de Grado.

II. PLANTEAMIENTO DEL PROYECTO

En este capítulo se mostrará al lector las bases del Trabajo Especial de Grado, se observarán las necesidades que motivaron el desarrollo de esta investigación, el objetivo que se desea alcanzar al finalizar el Trabajo de Grado y sus alcances y limitaciones.

A. Planteamiento del problema.

En la actualidad, son numerosas las transacciones realizadas a través de los puntos de ventas (POS por sus siglas en ingles), ya sea con tarjetas de débito o de crédito, esto debido a que las personas ya no acostumbran a llevar efectivo en sus billeteras. Según GAO RESEARCH, empresa líder en sistemas embebidos que ofrece servicios para implementar puntos de venta, afirma que se han realizado aproximadamente 28 millardos de transacciones a través de puntos de ventas en Estados Unidos y el tiempo promedio de cada una es de aproximadamente 40 a 50 segundos.

B. Objetivos.

1) Objetivo General: Analizar, diseñar e implementar un sistema transaccional que permite realizar pagos electrónicos implementando NFC.

2) Objetivos Específicos:

- Diseñar una aplicación ANDROID para transmitir información desde el dispositivo móvil hacia el punto NFC.
- Diseñar una aplicación que permita el registro de usuarios y almacenar la información personal de los mismos.
- Diseñar una aplicación capaz de obtener los datos recibidos a través del punto NFC.
- Implementar módulos de seguridad

C. Alcances y limitaciones.

1) Limitaciones: La aplicación necesita teléfonos que incluyan la tecnología NFC, sin embargo los teléfonos que ofrecen el sistema operativo ANDROID de gama baja y la mayoría de gama media no ofrecen esta plataforma, por lo cual, existe una limitación con la elección del dispositivo móvil que se requiere para la aplicación del trabajo especial de grado. Este mismo aspecto tiene un impacto económico puesto que los teléfonos de gama alta son los más costosos del mercado, por lo cual, es necesario invertir en dispositivos móviles de estas características para el correcto funcionamiento del sistema que se requiere implementar.

2) Alcances: El alcance más importante, es ofrecer una alternativa para las formas de pago existentes para facilitar las compras de los clientes en cualquier establecimiento comercial.

D. Justificación.

El sistema desarrollado trata de ofrecer una alternativa sencilla, efectiva, rápida y confiable para establecer una nueva forma de cancelar alguna factura, lo que reduce el tiempo invertido a la hora de pagar, todo esto gracias a la alta velocidad de transmisión que ofrece NFC y a la seguridad de la plataforma por el requerimiento de proximidad necesario para lograr la comunicación.

III. MARCO TEÓRICO

A. NFC (Near Field Communication).

Esta tecnología permite interacciones simples, seguras y bidireccionales entre dispositivos electrónicos; permitiendo a los usuarios realizar transacciones sin contacto, acceder a contenidos digitales y conectar dispositivos electrónicos con un solo toque. NFC complementa muchas tecnologías inalámbricas de consumo popular, y es compatible con la infraestructura existente de tarjetas sin contacto. NFC permite una velocidad máxima de comunicación de 424 kbps. NFC solo permite que los dispositivos estén separados por menos de 10 cm. [1]

Dentro de la forma de trabajo de NFC, un dispositivo genera una onda de radio de baja frecuencia que opera a 13,56 MHz. Cuando otro dispositivo NFC se acerca lo suficiente para ponerse en contacto, se genera un "acoplamiento magnético inductivo", por medio del cual se puede realizar una transferencia de energía y de datos entre los dispositivos. [1]

Se puede decir que NFC es una tecnología única ya que puede trabajar en tres diferentes configuraciones lo que la hace más adaptable y eficiente que otras. [1]

La tecnología NFC puede trabajar en tres distintas configuraciones:

- En el modo Lector/ Escritor un dispositivo NFC lee información de una etiqueta NFC.
- En el modo Emulación de Tarjeta el dispositivo NFC se comporta como una etiqueta NFC y es leído por otro dispositivo.
- En el modo Peer-to-peer dos dispositivos se intercambian información de igual a igual.

B. Lenguajes de Programación.

1) JAVA: JAVA es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "*write once, run anywhere*"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. [2]

A diferencia de lenguajes que le precedieron como C, C++ o Pascal, el compilador de JAVA no traduce el código fuente a lenguaje máquina, al finalizar la compilación es generado un código que es llamado Bytecode; para interpretar este código en los dispositivos finales, es necesario la instalación previa de la Máquina Virtual de JAVA (JVM), esto permite que JAVA sea un lenguaje independiente de la arquitectura del dispositivo. [3]

La máquina Virtual de JAVA más el conjunto de clases necesarias para ejecutar los programas conforma el entorno de ejecución de JAVA o JRE (JAVA Runtime Environment).

2) *ANDROID:* Es un sistema operativo para dispositivos móviles como teléfonos inteligentes y tabletas basado en el núcleo LINUX. Con ANDROID se busca reunir en una misma plataforma todos los elementos necesarios que permitan al desarrollador controlar y aprovechar al máximo cualquier funcionalidad ofrecida por un dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wi-Fi, BLUETOOTH, NFC, aplicaciones ofimáticas, videojuegos, etc.), así como poder crear aplicaciones que sean verdaderamente portables, reutilizables y de rápido desarrollo. [4]

3) *XML (Extensible Markup Language):* Es un conjunto de reglas para definir etiquetas semánticas que organizan un documento en diferentes partes. XML es un metalenguaje que define la sintaxis utilizada para definir otros lenguajes de etiquetas estructurados. [5]

En ANDROID se utilizan archivos XML para la declaración de *layouts* y otros elementos de los que hará uso una aplicación para su correcto funcionamiento.

4) *PHP (Hypertext Preprocessor):* PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo *web* y que puede ser incrustado en HTML. [6]

Está enfocado principalmente a la programación de scripts del lado del servidor, por lo que permite realizar funciones similares a programas que operen con CGI (Common Gateway Interface), como recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies.

C. *IDE (Integrated Development Environment).*

1) *NETBEANS:* Es un ambiente de desarrollo integrado escrito en lenguaje JAVA, el cual puede ser usado como un entorno genérico para construir cualquier tipo de aplicaciones. [7]

Soporta muchos lenguajes, desde JAVA, C/C++, XML, HTML, PHP, GROOVY, JAVADOC, JAVASCRIPT y JSP. También proporciona diferentes vistas de los datos a partir de múltiples ventanas y módulos útiles para la creación y gestión de aplicaciones, lo que permite que las aplicaciones basadas en NETBEANS puedan ser comprendidas por otros desarrolladores rápidamente.

2) *ANDROID STUDIO:* Es un entorno de desarrollo integrado (IDE), basado en INTELLIJ IDEA de la compañía JETBRAINS, que proporciona varias mejoras con respecto al plugin ADT (ANDROID *Developer Tools*) para ECLIPSE. ANDROID STUDIO utiliza una licencia de software libre APACHE 2.0, está programado en JAVA y es multiplataforma. [8]

D. *BBDD (Base de Datos).*

Según la base de datos que se requiere para llevar a cabo el proyecto, se ha obtenido la información necesaria de las características que esta debe tener. Esta base de datos debe ser:

1) Base de datos dinámica: En estas bases de datos la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización, eliminación y edición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de un supermercado.

2) Base de datos transaccionales: Son bases de datos cuyo único fin es el envío y recepción de datos a grandes velocidades, éstas son poco comunes y están dirigidas por lo general al entorno de análisis de calidad, datos de

producción y datos industriales. Es importante entender que su fin único es recolectar y recuperar los datos a la mayor velocidad posible, por lo tanto no hay redundancia ni duplicación de información como en las demás bases de datos.

E. Seguridad.

1) Encriptación: Es el proceso mediante el cual cierta información o texto sin formato es cifrado de forma que el resultado sea ilegible a menos que se conozcan los datos necesarios para su interpretación. Es una medida de seguridad utilizada para que al momento de almacenar o transmitir información sensible ésta no pueda ser obtenida con facilidad por terceros. [9]

2) Métodos de encriptación: Para poder encriptar un dato, se pueden utilizar tres procesos matemáticos diferentes: los algoritmos HASH, los simétricos y los asimétricos.

- *Algoritmo HASH.*

- Criptografía de Clave Secreta o Simétrica
- Algoritmos Asimétricos (RSA).

Al realizar una comunicación con el servidor *web* desde la aplicación ANDROID es indispensable la implementación de algún algoritmo de encriptación robusta donde se oculten las contraseñas de los usuarios para que los atacantes no puedan detectarlas. Es por esto que se utilizó AES para encriptar solamente las contraseñas mediante una clase, la cual solamente conoce el programador.

F. WAMP SERVER.

Es un paquete de herramientas que provee a los desarrolladores con los elementos necesarios para un servidor web, equipado con: un manejador de base de datos (MySQL), un software para servidor *web* (APACHE) y un software de programación script *Web* (PHP (generalmente), Python o PERL), debiendo su nombre a dichas herramientas. [10]

IV. METODOLOGÍA

Este capítulo presenta al lector las diferentes fases en las que se dividió la metodología, estas fases son necesarias para lograr los objetivos planteados en este Trabajo Especial de Grado.

A. PRIMERA FASE: Investigación documental.

Esta fase comprende el estudio y tratamiento de la información requerida para la investigación realizada. Se procesaron, clasificaron y se recuperaron distintos tipos de información referidas a los tópicos y temas asociadas a este trabajo de investigación.

B. SEGUNDA FASE: Selección de tecnologías

Durante esta fase se realizó el análisis de la información suministrada al finalizar la fase anterior. Este análisis se dividió en dos bloques, en el primer bloque se compararon los distintos sistemas operativos móviles para seleccionar el sistema operativo en el que se desarrollaría la aplicación móvil y el segundo bloque definió cuales serían las tecnologías de comunicación entre las aplicaciones y la base de datos.

En esta fase también se definieron los lenguajes de programación ideales para lograr la conexión entre el punto NFC y la computadora que haría las funciones de servidor, tomando en cuenta que la aplicación no podía ser excesivamente pesada y no podía consumir demasiados recursos, así como también la plataforma ideal para manejar las bases de datos. Se

tomó en cuenta el hecho que la aplicación principal para el servidor preferiblemente debía ser multiplataforma, de esta manera se garantizaba que la misma funcionara de manera correcta en cualquier sistema operativo para computadoras.

C. *TERCERA FASE: Diseño de la base de datos.*

Para la elaboración de la base de datos se utilizó la herramienta WORKBENCH de MySQL. Esta herramienta facilita enormemente la creación y diseño de las bases de datos, ya que cuenta con una interfaz gráfica donde se visualizan fácilmente las tablas, datos y relaciones que existen entre las entidades. Este proceso es conocido como "Generación de entidad – relación". Una vez que culmina este proceso, WORKBENCH posee un proceso que construye la base de datos para posteriormente utilizarla en el servidor MySQL.

D. *CUARTA FASE: Desarrollo de la aplicación servidor.*

La aplicación servidor fue desarrollada en el lenguaje de programación PHP, el cual fue diseñado para el desarrollo *web* dinámico y se utilizó el servidor HTTP APACHE para la comunicación *web*.

Esta aplicación permite la comunicación del dispositivo móvil a la base de datos para el registro de usuarios nuevos y el *login* de usuarios ya registrados, permitiendo obtener información que usará el dispositivo posteriormente en la aplicación que se encargará de las transacciones bancarias.

E. *QUINTA FASE: Desarrollo de la aplicación principal.*

Esta aplicación se encargará de escuchar continuamente el puerto USB en el cual esté conectado el punto NFC, al detectar un intercambio de información la misma capturará estos datos para enviarlos a la base de datos.

Estos datos modificarán dos tablas dentro de la base de datos, la primera en modificarse será la tabla de tarjetas de la siguiente manera:

- Se verificará el número de la tarjeta para localizar el registro de la misma dentro de la base de datos.
- Una vez encontrado este campo, se restará el saldo anterior con el monto enviado a través del punto NFC para obtener un nuevo saldo actual.
- Por último, con el resultado de esta operación se modificará los datos dentro de la tabla tarjeta y se colocará un nuevo saldo actual.

La segunda tabla a modificarse será la tabla de control, esto ocurrirá de la siguiente manera:

- Con la información del número de la tarjeta se procede a llenar un registro vacío dentro de la tabla control.
- Se incluirá la información del monto de la operación para registrarlo en la tabla control.
- Por último se asignará un número de referencia automático para posteriormente consultar los movimientos bancarios en caso de ser necesario.

F. *SEXTA FASE: Desarrollo de la aplicación ANDROID.*

La principal función de esta aplicación es lograr que el dispositivo se comporte como una etiqueta NFC y envié la información de la tarjeta y el monto de la misma a través de esta plataforma,

conectándose con la aplicación bancaria a través de un punto NFC.

La segunda función es conectarse con el servidor *Web,* tanto para registrar usuarios nuevos como hacer el inicio de sesión, para obtener los datos de la tarjeta asociada al usuario y proceder a hacer las tareas de la función principal.

G. SÉPTIMA FASE: Pruebas del Sistema.

Una vez finalizadas las fases anteriores, se procedió a realizar pruebas pilotos en cada una de las aplicaciones, de esta manera se observa el correcto funcionamiento del sistema, las pruebas se realizaron con la intención de verificar los siguientes puntos de interés:

- Conexión del cliente a la aplicación servidor.
- Respuesta del servidor al cliente.
- Transferencia desde el dispositivo móvil al punto NFC.
- Conectividad del punto NFC a la aplicación principal.
- Consultas y modificaciones automáticas de las bases de datos.
- Descuento final de saldo al cliente.

H. OCTAVA FASE: Conclusiones y Recomendaciones.

En fase se demuestran las soluciones encontradas al problema planteado, posteriormente se muestra al lector las conclusiones obtenidas al concluir el Trabajo Especial de Gado. También se ofrecen múltiples recomendaciones, obteniendo una descripción de las acciones a tomar para la mejora del trabajo realizado.

I. NOVENA FASE: Elaboración del Tomo.

En esta fase se consideran los mecanismos y procedimientos formales determinados por la Universidad Católica Andrés Bello para realización del Tomo; documentando de forma cronológica los momentos que conforman el proceso de investigación y de pruebas realizadas para comprobar el buen funcionamiento del sistema. Adicionalmente, en esta fase se deja constancia del cumplimiento a los objetivos planteados al inicio de la investigación.

V. RESULTADOS.

En este capítulo se muestra al lector los resultados obtenidos en la investigación, describiendo y analizando los resultados obtenidos en cada una de las fases desarrolladas.

A. Selección de Tecnologías.

1) Conexión a la aplicación servidor: En esta instancia del sistema se manejaron dos alternativas posibles:

- Conexión a través de *sockets.*
- Conexión mediante servicios *web.*

Puesto que la aplicación se realizó para su futura implementación, con la intención de que pueda funcionar como una alternativa de pago cotidiana; lo ideal sería que hubiera una comunicación segura y que el servidor recibiera múltiples consultas de múltiples usuarios. Por lo cual, la elección del servicio *web* es la más apropiada para el sistema de transacción inalámbrico.

2) Herramientas y lenguajes de programación:

Para lograr la comunicación entre el servicio *web* y la aplicación ANDROID, se seleccionó PHP como lenguaje de programación para el lado del servidor, el mismo se encarga de registrar los datos en la

lista de la base de datos destinada a la información del usuario, también se encarga del inicio de sesión por parte del cliente previamente registrado, para el correcto funcionamiento del servicio *web* se utilizó la herramienta WAMPSERVER.

Para el desarrollo de la aplicación principal se tomó JAVA como lenguaje de programación, puesto que el lenguaje es multiplataforma lo que permite el correcto funcionamiento de la aplicación, independientemente del sistema operativo.

Para la base de datos se necesitó una herramienta para la gestión de la base de datos, preferiblemente de código abierto, para evitar pagar por alguna licencia de código propietario. Todas estas ventajas llevaron a la elección de MySQL como el gestor de base de datos para el sistema.

B. *Diseño de la base de datos*

Para el funcionamiento del proyecto se necesitaron tres tablas, que se pueden observar en la figura 1, las cuales permiten al sistema:

- Almacenar los datos de los usuarios nuevos al momento de registrarse para posteriormente permitir el ingreso de los mismos y disfrutar de los beneficios del sistema
- Registrar el serial de la tarjeta NFC del dispositivo móvil para registrarlo como número de tarjeta, para posteriormente hacer los descuentos necesarios.
- Guardar un comprobante de todas las operaciones monetarias que se hacen en el sistema para futuras consultas.

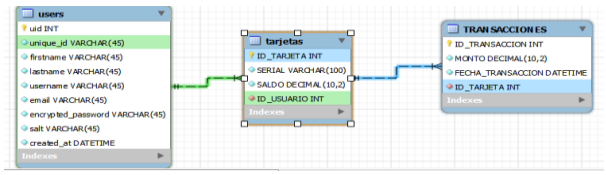


Figura 1. Diagrama Entidad-Relación de la base de datos del sistema

C. *Desarrollo de la aplicación servidor.*

Esta parte del sistema es la encargada recibir y enviar toda la información requerida a través del servicio *web*. Esta comunicación se realiza gracias al intercambio del servicio *web* a través del código PHP con el dispositivo móvil, este intercambio de información se lleva a cabo a través de un JSON (*JavaScript Object Notation*) que emite una petición desde la aplicación ANDROID para que el mismo sea respondido por el servicio *web*, devolviendo información, dependiendo de la necesidad del usuario. Principalmente la aplicación servidor responde las siguientes peticiones:

- Registro de nuevo usuario: El servicio *web* al escuchar una petición para el ingreso de un nuevo usuario, usará la información suministrada por el dispositivo móvil para agregar estos datos a la lista *users*.

- *Login* o ingreso de usuario: Para permitir el ingreso de usuarios previamente registrados se requiere el correo electrónico y su contraseña para verificar que dichos datos se encuentren almacenados en la base de datos.

- Cambio de contraseña: Si el usuario en cualquier momento olvida su contraseña, esta función permite que ingresando su correo electrónico pueda crear una nueva contraseña.

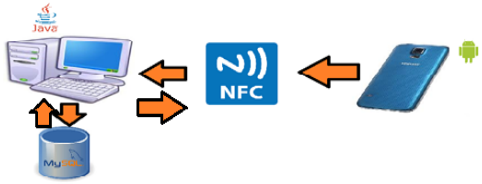
- Información de saldo disponible: Permite consultar el saldo que actualmente el usuario posee.

- Información de cuenta: Regresa la información personal suministrada por el usuario al momento de crear su cuenta en sistema.

- Información de monto de transacción: Presenta al usuario el último monto que se le fue debitado a la hora de realizar una transacción monetaria a través del sistema.

D. *Desarrollo de la aplicación principal.*

La aplicación se diseñó usando un patrón de arquitectura de *software* MVC (Modelo-Vista- Controlador) con la intención de separar los datos y la lógica de negocio de la interfaz gráfica. La elección de este patrón se debe a la facilidad que ofrece para las tareas de desarrollo del *software*, lo que reduce el tiempo necesario para finalizar la aplicación y tener una mayor organización al momento de programar. Por lo tanto la explicación del funcionamiento de la aplicación se realizará basada en este patrón.



1) *Modelo:* Esta parte de la aplicación se encarga de buscar, insertar, modificar y eliminar datos dentro de las listas de la base de datos destinadas para almacenar la información del usuario, su tarjeta y de las transacciones realizadas. Las clases involucradas para estos fines son las siguientes:

- *ConeccionBD.java*: Esta clase se encarga de iniciar la conexión con base de datos y cerrar la conexión con la misma.

- *ClienteDAO.java*: Esta clase se encarga en obtener toda la información referente al usuario, interactúa directamente con la tabla *users.*

- *TarjetaDAO.java*: Esta clase está clase se encarga de obtener la información referente a la tarjetas del usuario interactuando con la tabla tarjetas.

- *TransaccionDAO.java:* Esta clase es la encargada de la gestión de la tabla Transacciones.

2) *Vistas:* Son las clases encargadas de interactuar con el usuario. Estas vistas están separadas de la siguiente manera:

- Clase *Inicial.java:* Esta clase se encarga de mostrar la primera pantalla de la aplicación, la cual solicita una clave de acceso para ingresar a la aplicación.

- Clase *Principal.java*: Esta clase muestra la pantalla principal de la aplicación, donde se visualizan las opciones principales del sistema.

- Paquete *com.nfc.ventana.clientes*: Dentro de este paquete se encuentran las clases visuales que permiten la interacción con los datos del cliente.

- Paquete *Pagos*: En este paquete, se encuentran todas las clases necesarias para la interacción del usuario con la capa controlador para realizar la transacción monetaria.

- Paquete *Transacciones*: En este paquete, se encuentran todas las clases que muestran las transacciones realizadas dentro del sistema para la consulta del mismo en cualquier momento.

3) *Controlador:* se definen toda las funciones necesarias para la buena comunicación entre las dos capas descritas anteriormente por lo tanto es la capa que define cuando es necesario llamar algunas de las clases DAO y

cuáles son los atributos que está solicitando el usuario.

Entre estas clases de negocio, se encuentran en el paquete *com.nfc.negocio* y está conformado por las siguientes clases:

- Clase *Cliente.java*: Es la encargada de todas las acciones referentes al cliente, como búsqueda, registro del serial NFC y recarga de saldo.

- Clase *Paquetes.java*: Se encarga de toda la lógica para la realización de pago a través del sistema.

- Clase *ConexionNFC.java*: Esta clase tiene la responsabilidad de todo el proceso para la comunicación con el punto NFC (conexión, solicitud de data y existencia de un dispositivo en contacto con el punto NFC).

E. Desarrollo de la aplicación ANDROID.

La aplicación para dispositivos móviles ANDROID (teléfonos inteligentes y tabletas) es quien permite esta interacción del cliente con el sistema.

1) Activity para inicio de sesión: El proceso de inicio de sesión se lleva a cabo por medio de una petición POST que se realiza desde la aplicación ANDROID hacia el servidor *Web* donde primero se envían los parámetros: *email* y contraseña en formato JSON, hacia el método *loginUser* de la clase *UserFunctions* que se encuentra en el paquete de librerías, el cual se encarga de colocar en un *array* todos los parámetros que van a ser enviados al servidor *web*. Posteriormente este *array* va al método *getJSONFromUrl* de la clase *JSONParser* junto con el URL del servidor *web*, para que esta clase se encargue de enviar la información al servidor *web*, que en este caso es el *email* y la contraseña, para posteriormente recibir una respuesta verificando si el usuario existe o no en la base de datos.

2) Activity para registro: Permite al usuario ingresar sus datos personales para la creación de una cuenta en el sistema para que posteriormente pueda iniciar sesión. La *Activity* utiliza la clase *Asynctask* para realizar una prueba de conexión a internet, una vez culminada la prueba, se envían los datos al servicio *web*, para almacenarlos dentro de la Base de Datos (*comprasnfc*), siempre y cuando no exista un usuario con el mismo correo electrónico. Si el usuario pudo ser registrado, se muestra un *Toast* y además se envía un correo electrónico indicando que el registro fue exitoso, en caso de que el registro no pudo ser completado, alguno de los campos están vacíos o la contraseña no tiene el número de caracteres mínimos se mostrará en la aplicación un *Toast* indicando alguno de estos errores. Esta *Acitivity* permite a través de la clase *DatabaseHandler*, almacenar los datos enviados en la base de datos interna del dispositivo (*SQLite Database*).

3) Activity detalles de registro: Si el registro fue exitoso, en la clase de registro existe una llamada al método (*addUser*) de la clase *DatabaseHandler*, que se encarga de almacenar la información del usuario en la base de datos del dispositivo móvil (*SQLite database*). La *Activity* de detalles de registro muestra la información almacenada en la base de datos del dispositivo móvil del último usuario registrado.

4) Activity olvido de contraseña: En esta fase el usuario tiene la posibilidad de recuperar su contraseña en caso de olvido para que pueda ingresar de nuevo en su cuenta iniciando sesión con una

contraseña válida. Esta etapa consta de dos *Activitys*:

- Introducción del correo: en esta *Activity* es necesario que el usuario ingrese su correo electrónico para verificar que efectivamente posee una cuenta registrada en la base de datos.

- Introducción de nueva contraseña: luego de verificar la existencia del usuario a través de su correo electrónico se muestra la segunda *Activity* donde el usuario puede ingresar su nueva contraseña para posteriormente iniciar sesión.

5) *Activity pantalla principal:* Esta *Activity* muestra la vista principal luego de que el usuario inicie sesión. A continuación se mostrarán los botones y sus funciones:

- Saldo disponible: Muestra una pantalla donde muestra el saldo actual del usuario.

- Factura: En este botón se muestra la última transacción realizada, para que el usuario tenga un mayor control de sus pagos.

- Pagar: Este botón es utilizado al momento de efectuar un pago, donde el dispositivo pasa a emular una etiqueta NFC con la característica que ellas poseen.

- Consultar datos: Este botón muestra una pantalla en la cual se despliega la información de registro del usuario.

- Configuración: Permite realizar cambio de contraseña para brindarle al usuario la libertad de cambiarla cuando lo desee.

- Salir: Retrocede a la pantalla de inicio de sesión.

6) *Activity cambio de contraseña:* Esta *Activity* se creó con la finalidad de que el usuario pueda cambiar su contraseña cuando lo desee, para lograr esto, en la clase *ActivitySeven* se envía una petición POST en formato JSON al servidor *web*, donde se envían los parámetros de la nueva contraseña y el correo del usuario. Luego la función de cambio de contraseña en PHP se encarga de manejar la consulta en la base datos teniendo como referencia el *email* del usuario y reemplaza el valor de su contraseña actual por la nueva. Para finalizar el servidor envía una respuesta con el mismo formato JSON donde se comprueba a través de un mensaje que el cambio fue exitoso, además de esto también se envía un correo al usuario para mejor administración de su cuenta.

7) *Activity para consultar los datos del usuario:* el usuario, al iniciar sesión, recibe en un formato JSON la respuesta del servidor *web* junto con todos los datos del usuario que agregó al momento de registrarse: nombre, apellido, nombre de usuario, *email* y la hora y fecha del registro. Luego estos son almacenados en una base de datos (contactos) creada en la clase *DatabaseHandler*, esto permite que estos datos sean extraídos y se pueden consultar posteriormente llamando al método *getUserDetails* el cual retorna un *HashMap* donde se encuentran todos estos datos.

8) *Activity para pagar:* Esta *Activity* tiene la finalidad de permitir que el dispositivo emule una etiqueta NFC utilizando las clases *MyHostApduService* y *utils* las cuales pertenecen al paquete de librerías y son utilizadas específicamente para la comunicación entre el dispositivo y el lector NFC ACR122U. En esta etapa

del proceso es donde se realizan los pagos enviando el serial NFC asociado al dispositivo.

9) *Activity para ver la última transacción:* En esta *Activity* se muestra al usuario la última transacción realizada la cual se logra a través de una petición POST hacia el servidor *Web*, donde se envía como parámetro el número de tarjeta asociada al usuario en un formato JSON para poder realizar la consulta en la base de datos y poder extraer el ultimo monto que registró. Posteriormente es recibido en la clase *JSONParser* para que luego sea enviado al llamar al método *getJSONObject* desde la *ActivitySix* para que finalmente la información sea trasladada por medio de un *bundle* a la *ActivityTen* donde se mostrará el número de la última transacción.

10) *Activity para ver el saldo disponible:* Esta *Activity* muestra el saldo disponible que posee el usuario. Éste retorna el valor del saldo en formato JSON para que posteriormente sea desplegado en la pantalla llamando igualmente que el caso anterior al método *getJSONObject* de la clase *JSONParser* desde la *ActivitySix* para que posteriormente la información sea enviada a la *ActivityEleven* a través de un *bundle* donde dicha información del saldo será desplegada en la pantalla del dispositivo del usuario.

11) *Paquete de librerías:* Las clases que constituyen el paquete de librerías son los siguientes:

- *DatabaseHandler.java*
- *JSONParser.java*
- *MyHostApduService.java*
- *Seguridad.java*

F. Pruebas del sistema.

Al realizar las pruebas finales del sistema se utilizó un usuario de ejemplo para comprobar que la comunicación se efectuara de manera efectiva y eficaz. Al registrar al usuario se utilizó la *Tablet NEXUS 7*, la cual posee el API 21, en conjunto con una laptop SAMSUNG con 4 GBytes de RAM, procesador I3 y con una conexión a internet de aproximadamente 2 Mbps, en la cual está instalado *WAMP SERVER*. Una vez registrados los datos del usuario, se envía un *email* indicando que el proceso fue exitoso, esto se realizó con una duración de 3 segundos, luego al iniciar sesión tardó alrededor de 3 segundos en ingresar a la pantalla principal de la aplicación. Para realizar la transacción, previamente se introdujo un saldo de 10000 al usuario Roberick desde la aplicación principal en JAVA, luego se procedió a registrar el serial del dispositivo acercando la *Tablet* al lector NFC. El serial almacenado en la tabla de tarjetas de la base de datos *comprasnfc* fue 05758070, con un tiempo de duración no mayor a 5 segundos. Al momento de realizar el pago se colocó el monto de 5000 en la aplicación principal en JAVA, y posteriormente se presionó el botón pagar en la aplicación ANDROID colocando la *Tablet* encima del lector NFC, para que la aplicación principal en JAVA se encargue de asociar el serial con la cuenta del usuario, verificando si dispone de saldo suficiente o no, esta actividad se realizó en aproximadamente 4 segundos. Una vez realizada esta operación el usuario recibió un correo indicando el monto que pagó y el saldo disponible. Además de esto, se verificó desde la aplicación ANDROID si el monto y el saldo eran los correctos (Monto: 5000, Saldo: 5000) a través de los botones *Factura* y Saldo disponible.

VI. CONCLUSIONES Y RECOMENDACIONES

A. Conclusiones.

Al culminar el Trabajo Especial de Grado, se puede concluir en primera instancia que a pesar de que la tecnología NFC tiene aproximadamente 11 años de haberse inmerso en el mundo de las Telecomunicaciones aún no posee suficiente popularidad en el mercado, y más aún en Venezuela. Sin embargo, gracias a la investigación realizada se pudo demostrar que la tecnología NFC posee muchas aplicaciones en la actualidad, las cuales se pueden implementar bajo cualquier circunstancia y en cualquier lugar mediante el uso de etiquetas y algún lector NFC que sea capaz de interactuar con dichas etiquetas.

Para finalizar, la prueba realizada con el usuario de ejemplo fue completamente exitosa, y su duración total fue de aproximadamente 15 segundos. Hay que destacar que una vez que el usuario registre sus datos y el serial del dispositivo este tiempo baja considerablemente a 7 segundos.

B. Recomendaciones.

Para futuros proyectos relacionados con este Trabajo Especial de Grado se recomienda, en caso de utilizar un lector NFC similar al ACR122U, obtener el SDK proporcionado por la empresa ACS, puesto que facilitará el desarrollo del sistema llevando a cabo una comunicación exclusivamente vía NFC.

Es importante llevar este sistema de pago a un nivel macro, incluyendo a empresas bancarias o financieras, que puedan brindar su apoyo para realizar transacciones reales mediante la moneda nacional, bajo la tutela de un supervisor de algún local o establecimiento.

La utilización de un Servidor *Web* local es una gran limitante, debido a que sólo se puede efectuar la comunicación en una intranet con un Router inalámbrico, el cual brinda a los usuarios acceso a la red, lo que es ideal para locales pequeños. Es por esto que se recomienda implementar un Servidor *Web* público para poder ampliar el área de comunicación entre los usuarios y el establecimiento, además de poder ser aplicado en franquicias comerciales.

VII. BIBLIOGRAFÍA

[1] A. C. Ruiz, «DESARROLLO DE UNA APLICACIÓN DE PAGO A TRAVÉS DE LA TECNOLOGÍA NFC,» Madrid, 2011.

[2] R. Azpitarte, P. Jordá y J. Llinares, Introducción a la programación orientada a objetos con java, Valencia: Universidad Politécnica de Valencia, 2009.

[3] L. Joyanes y M. Fernández, Java 2 Manual de programación, Madrid: McGraw-Hill, 2003.

[4] J. Tomas, «Tutoriales Android Fundamentos,» 2011. [En línea]. Available: http://www.androidcurso.com/index.php/tutoriales-android-fundamentos. [Último acceso: 21 Marzo 2014].

[5] M. Melián Montalvo, «XML. El nuevo lenguaje universal,» 2008. [En línea]. Available: www,bibliociencias.com.

[6] php, «php,» [En línea]. Available: http://php.net/manual/es/intro-whatis.php.

[7] NetBeans, «Home: NeatBeans,» 2013. [En línea]. Available: https://netbeans.org/about/index.html.

[Último acceso: 18 Junio 2014].

[8] Androcode, «Androcode,» 16 mayo 2013. [En línea]. Available: http://www.androcode.es.

[9] A. TANENBAUM y D. WETHERALL, «Redes de Computadoras,» Mexico, Pearson, 2012.

[10 H. I. Ortega Martinez, «Que es Wamp Server,» 19 noviembre 2013. [En línea]. Available: http://ingenieros.wordpress.com.