ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PROTOTIPO DE CARRO DE COMPRAS VIRTUAL EMPLEANDO COMUNICACIONES NFC Y EL SISTEMA ANDROID

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

REYES CIFUENTES VÍCTOR HUMBERTO
victor.reyes@est.epn.edu.ec

SANTOS TORRES ANDRÉS JAVIER
andres.santosj@epn.edu.ec

DIRECTOR: ING. RAÚL DAVID MEJÍA NAVARRETE, MSc.
david.mejia@epn.edu.ec

Quito, Diciembre 2014

DECLARACIÓN

Nosotros, Víctor Humberto Reyes Cifuentes y Andrés Javier Santos Torres, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____ _____

Víctor Humberto Reyes Cifuentes Andrés Javier Santos Torres

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Víctor Humberto Reyes Cifuentes y Andrés Javier Santos Torres, bajo mi supervisión.

_____

Ing. Raúl David Mejía Navarrete, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTO

Agradezco a mis padres: Francisca Torres y Roque Santos, por todo el apoyo brindado durante esta etapa de mi vida, gracias a ellos es que he podido cumplir todas las metas que me he propuesto.

A mis hermanos: Roque Santos y Alberto Santos quienes con su apoyo y consejos me han permitido llegar hasta este punto de mi vida.

Un especial agradecimiento a Loly Pérez una gran amiga que me acompaño y apoyo durante estos años de carrera universitaria.

A mis amigos en especial al coautor de este proyecto Víctor Reyes quien con su ayuda y apoyo se logró concluir este proyecto exitosamente.

Al director de este proyecto MSc. David Mejía por su ayuda, aportes y tiempo dedicado en este proyecto.

A la EPN y a mis profesores: Ing. Pablo Hidalgo, MSc. Soraya Sinche, Ing. Fernando Flores, MSc. Xavier Calderón y a todos los profesores con los que alguna vez recibí clases, por sus conocimientos y consejos que me impartieron durante mi carrera universitaria.

A los compañeros y equipo de trabajo “los mijines”: David, Vanesa, Johanna, Jorge, Gabriel, Marlon, Telmo y todas las personas que estuvieron durante estos meses dándome ánimos para concluir exitosamente este proyecto.

Andrés Santos

DEDICATORIA

Dedico este proyecto a Dios y a mis padres a quienes se los debo todo.

Andrés Santos

,,Bilde dich selbst und dann
Wirke auf andere durch das,
was du bist.”
-Wilhelm von Humboldt

AGRADECIMIENTO

A Dios, por permitirme vivir y permanecer a mi lado en los momentos difíciles ayudándome a sobrellevarlos de la mejor manera.

A mis padres y mi familia por el apoyo y confianza brindada para alcanzar cada una de mis metas profesionales y personales.

A mí director de proyecto MSc. David Mejía por su apoyo, ayuda y todos sus conocimientos brindados para lograr este proyecto.

A mis amigos y de manera especial al coautor de este proyecto por su ayuda y apoyo durante todo el desarrollo de este proyecto.

A la Escuela Politécnica Nacional y a cada uno de sus docentes por cada una de las enseñanzas y consejos que me han brindado durante mi vida estudiantil.

A las comunidades de usuarios y desarrolladores de software que sin interés alguno comparten sus conocimientos y experiencias.

Víctor Reyes

DEDICATORIA

Todo el esfuerzo y sacrificio empleado a este proyecto va dedicado a toda mi familia quienes han sido mi apoyo incondicional en mis momentos difíciles y me han sabido guiar y apoyar durante toda mi vida estudiantil.

Víctor Reyes

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

ÍNDICE DE TABLAS

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 4

CAPÍTULO 5

ÍNDICE DE CÓDIGO

CAPÍTULO 3

CAPÍTULO 4

RESUMEN

El presente Proyecto comprende el desarrollo de un sistema prototipo de carro de compras virtual empleando comunicaciones NFC y el sistema Android, que permita automatizar la selección y compra de artículos en locales comerciales.

El sistema prototipo de carro de compras virtual está conformado por un Servidor SOA, una base de datos, una aplicación cliente para dispositivos móviles y una aplicación cliente para computadores de escritorio.

El servidor SOA, accede a la base de datos. La base de datos almacena la información de usuarios del sistema, artículos y toda la información necesaria para el proceso de compra. El servidor SOA se basa en la tecnología REST e implementa la lógica de negocio.

La aplicación móvil, desarrollada para el sistema operativo Android, implementa la interfaz de usuario necesaria para el manejo del carro de compras, además esta aplicación hace uso del lector NFC incorporado en dispositivos móviles para obtener información de los artículos acercando el dispositivo a un tag NFC.

La aplicación de escritorio, implementa la interfaz de usuario, necesaria para el manejo del inventario del sistema prototipo, así como también para gestionar la facturación.

En el primer capítulo se presentan conceptos de sistemas distribuidos, Arquitectura Orientada a Servicios (SOA), servicios WCF basados en la tecnología REST y comunicaciones inalámbricas con NFC, así como también conceptos de desarrollo de aplicaciones para la plataforma Android empleando el lenguaje de programación Java. También se presenta conceptos de desarrollo de .NET con el lenguaje de programación C# y conceptos de la metodología de programación eXtreme Programming.

En el segundo capítulo se presenta el análisis de requerimientos obtenido a partir de las aplicaciones NFC iShop y Mercado Libre. En base a los requerimientos

mínimos y a otros requerimientos, se detalla la metodología de desarrollo de todos los componentes que conforman el sistema prototipo y se especifica la función que cumple cada componente en el sistema.

En el tercer capítulo se describe el proceso de desarrollo de todos los componentes del sistema según los diseños realizados. Específicamente, se presenta el desarrollo del Servicio Web, la base de datos y las aplicaciones cliente.

En el cuarto capítulo se presentan los resultados de las pruebas realizadas al sistema; específicamente se presentan: pruebas unitarias, pruebas de carga y pruebas de funcionalidad realizadas en ambientes controlados, adicionalmente se presenta una comparación entre la aplicación de Android concluida con las aplicaciones utilizadas para elaborar el análisis de requerimientos.

En el quinto capítulo se realiza el análisis de costos del sistema prototipo propuesto, adicionalmente se realiza una comparación de costos de la aplicación móvil desarrollada, con las aplicaciones móviles Mercado Libre y NFC iShop utilizadas en el análisis de requerimientos.

Finalmente se presentan las principales conclusiones y recomendaciones que arrojan el desarrollo y la implementación del sistema.

Adicionalmente en los anexos se incluye: todos los Scripts para la creación de la base de datos; el código fuente, Instalador y el manual de instalación del Servicio Web; el código fuente, el instalador, el manual de instalación y el manual de usuario de la aplicación de escritorio; el código fuente de la biblioteca de clases utilizada en la aplicación de escritorio y el servidor; el código fuente, instalador, manual de instalación y manual de usuario de la aplicación móvil; por último se incluye las pruebas unitarias de las aplicaciones y las pruebas de carga realizadas al Servicio Web.

PRESENTACIÓN

El presente Proyecto de Titulación se ha realizado con el objetivo de dar a conocer la tecnología inalámbrica NFC (*Near Field Communications*). Las comunicaciones mediante dispositivos NFC están cambiando la forma actual de hacer distintos tipos de transacciones como por ejemplo: el uso de una tarjeta de crédito, publicidad en locales comerciales, transferencia de archivos, etc.

En el presente documento el lector tendrá a su disposición la información básica para comprender el funcionamiento de NFC, así como información básica para comprender el funcionamiento del sistema operativo Android y la programación de aplicaciones para este sistema operativo mediante el lenguaje de programación Java, por último se incluye información básica para realizar aplicaciones con el *framework* de .NET.

Dentro del proyecto de titulación se desarrollan dos aplicaciones: una aplicación móvil desarrollada para dispositivos Android en la cual se implementa el carro de compras virtual haciendo uso de la tecnología inalámbrica NFC, la segunda aplicación desarrollada para trabajar sobre el sistema operativo Windows en el cual se implementa el control de inventario y el manejo de la facturación.

El sistema prototipo se desarrolló empleando Servicios Web con transacciones HTTP. Se desarrolló un servidor SOA WCF (*Service Oriented Architecture Windows Comunication Fundation*) basado en la tecnología REST el cual realiza la petición de los datos a la base de datos y los entrega a los clientes.

Por último se realiza un análisis de costos del proyecto, en el cual se identifica los costos de todos los componentes que conforman el sistema y se determina la rentabilidad del mismo.

CAPÍTULO 1. FUNDAMENTOS TEÓRICOS

1.1 SISTEMAS DISTRIBUIDOS

1.1.1 DEFINICIÓN [1] [2]

Un Sistema Distribuido es un conjunto de computadores autónomos interconectados por una red que comparten un estado global, dando como consecuencia una visión de sistema único, que muestra sus recursos de manera homogénea.

El sistema distribuido está compuesto por varios elementos de procesamiento autónomos que cooperan para lograr una meta. Por lo que resulta útil clasificarlos en: fuertemente acoplados y sistemas débilmente acoplados. En los sistemas fuertemente acoplados, los elementos del sistema tienen acceso a una memoria común, la sincronización y comunicación pueden efectuarse mediante técnicas basadas en variables compartidas. En cambio, los sistemas débilmente acoplados no tienen acceso a una memoria común y su comunicación requiere de paso de mensajes e invocaciones remotas.

Partiendo de la variedad de los procesadores del sistema también se los puede clasificar en sistemas homogéneos en los cuales todos los procesadores son del mismo tipo y sistemas heterogéneos los cuales contienen procesadores de diferentes tipos.

1.1.2 PRINCIPALES ARQUITECTURAS EN SISTEMAS DISTRIBUIDOS

1.1.2.1 Cliente Servidor [3]

En el modelo cliente servidor se tienen dos participantes, los primeros son los clientes quienes hacen peticiones de servicio, y los segundos son los servidores quienes proveen de servicios. En algunos casos los servidores pueden actuar como clientes de otros servidores, como es el caso del buscador de Internet, que funciona como servidor al recibir una petición del usuario desde la página web y

como cliente de otros servidores de Internet, al buscar información para el usuario.

1.1.2.1.1 Arquitectura multiestrato

La funcionalidad está distribuida entre varios computadores y presenta 3 niveles que son la interfaz de usuario del sistema, la capacidad de procesamiento y la gestión de los datos. Dependiendo de cómo se sitúen estos niveles se pueden tener varios tipos de arquitecturas como las que se muestra en la **Figura 1.1**.

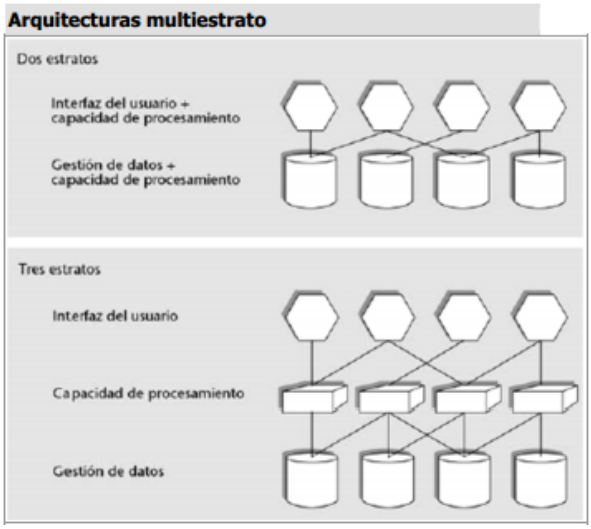

Figura 1.1 Arquitecturas multiestrato [3]

- *Arquitectura de 2 estratos:* En esta arquitectura, la gestión de la base de datos es manejada por el servidor y en el cliente se comparte la capacidad de procesamiento y la interfaz de usuario. Este tipo de funcionamiento da características como: tener poco procesamiento en el servidor, una base de datos centralizada y estática, y un mantenimiento mínimo.
- *Arquitectura de 3 estratos:* En esta arquitectura, entre el cliente y el servidor se añade un tercer estrato que va ser el encargado manejar la capacidad de procesamiento. En esta arquitectura se separa el procesamiento de la interfaz de usuario.

1.1.2.1.2 Aplicaciones basadas en web

Este tipo de aplicaciones están basadas en la arquitectura cliente servidor. El cliente envía peticiones al servidor web a través de la red, el servidor recibe la

petición y realiza el procesamiento necesario. Luego envía como respuesta al cliente el resultado de la petición con todos los datos procesados. Estos datos son mostrados directamente al usuario a través de una interfaz que puede ser un navegador, sin la necesidad de realizar ningún procesamiento extra.

1.1.2.2 Publicación-Subscripción [3]

En esta arquitectura, el que produce la información anuncia la disponibilidad de la misma, y el que esté interesado en la información se subscribe para recibirla.

Este tipo de arquitectura se caracteriza porque no hay que buscar la información pues está ya se encuentra localizada, solo se recibe la información deseada, la actualización de los datos es permanente y se reduce el trafico ya que no se hace búsqueda ni actualización de datos permanente desde el cliente, sino que la información está localizada y se la actualiza en caso de algún cambio.

1.1.2.3 Peer to Peer [3]

En este tipo de Sistema Distribuido, cada uno de los componentes o nodos tienen las mismas funciones, capacidades y la comunicación es simétrica. Estos sistemas son muy utilizados para la compartición de archivos.

1.1.2.4 Sistemas distribuidos basados en eventos [3]

En estos sistemas es posible que un objeto pueda reaccionar a cambios en otro objeto por medio del uso de eventos, cada uno de los componentes puede anunciar o difundir los eventos y los componentes se pueden registrar en los eventos que estén interesados, cuando se produce un evento se invoca a todos los componentes que estén registrados para ese evento.

Las características principales de estos eventos es que son heterogéneos y asincrónicos. Aparentemente estos sistemas son similares a los de Publicación-Subscripción pero en realidad son muy diferentes ya que en estos sistemas los roles no están definidos: la frecuencia de información que es muy variable y la cantidad de información tiene que ser mínima.

1.1.3 SERVICIOS WEB

1.1.3.1 Definición [4]

Un servicio expresa una funcionalidad en concreto que puede ser descubierta en la red, en el servicio se establece tanto lo que puede hacer como el modo de interactuar con la red. Un servicio realiza una tarea en concreto que puede corresponder a un proceso que sea este tan sencillo como extraer un dato o involucrar procesos de alto nivel y de un alto grado de complejidad que puedan involucrar varias aplicaciones de negocio.

Los Servicios Web se definen como aplicaciones que utilizan estándares para el transporte, codificación e intercambio de información. Estos servicios permiten la intercomunicación entre sistemas de cualquier plataforma.

1.1.3.2 Características [5]

- *Interoperabilidad:* Mediante los Servicios Web se pueden intercambiar datos entre aplicaciones de lenguajes de programación diferentes, que se ejecutan sobre diferentes plataformas.
- Los Servicios Web se basan en protocolos y estándares muy difundidos como son HTTP[1] (*Hypertext Transfer Protocol*), XML[2] (*eXtensible Markup*), etc.
- No se imponen restricciones sobre las aplicaciones a las que se les da acceso, ni las tecnologías que las implementan.
- Existen organizaciones que son las responsables de definir la arquitectura y los estándares, estas organizaciones son OASIS[3] (*Organization for the Advancement of Structured Information Standards*) y W3C[4] (*World Wide Web Consortium*).

[1] **HTTP:**Protocolo de capa aplicación usado en cada transacción de Internet.
[2] **XML:** Lenguaje de marcas utilizado para almacenar datos en forma legible.
[3] **OASIS:** Es una organización que trabaja para desarrollar estándares para comercio electrónico y Servicios Web.
[4] **W3C:** Es una comunidad internacional donde sus integrantes trabajan para desarrollar estándares web.

1.1.4 ARQUITECTURA ORIENTADA A SERVICIOS [4]

En este tipo de arquitectura se establece una marca de diseño para aplicaciones independientes, para que desde la red se pueda acceder a los servicios que se descomponen de aplicaciones monolíticas para implementar las funcionalidades en forma modular. La forma más común de implementar esta arquitectura es mediante Servicios Web.

Cuando se desarrolla software SOA (*Software Oriented Architecture)*, se debe orientar el software al fin de crear servicios comunes solicitados por clientes para implementar procesos. Estos servicios pueden estar residentes en el Internet o en una intranet y pueden usar diversos estándares como son: XML, HTTP, REST[5](*Representational State Transfer*), JSON[6](*Javascript Object Notation*), WSDL[7] (*Web Services Description Language*), etc.

El utilizar SOA proporciona grandes beneficios como la mejora en los tiempos de realización de cambios en los procesos, la facilidad para la integración de tecnologías, la orientación a servicios nos permite crear aplicaciones flexibles, reutilizables y adaptables y se facilita la migración ya que se reutilizan los componentes existentes.

1.1.4.1 Software como Servicio

"*Software que se pone en explotación en la modalidad de servicio gestionado y que al cual se accede a través de Internet*" [4].

SaaS (*Software as a Service)* se basa en una arquitectura con una única instancia de múltiples usuarios que permita una experiencia con funcionalidades avanzadas. Las aplicaciones de este tipo de servicio normalmente ofrecen un proveedor de forma directa o un intermediario (llamado "agregador") que empaqueta ofertas SaaS de distintos proveedores y las que se ofrecen como una plataforma unificada de aplicaciones o una suite de servicios de aplicación.

---

[5] **REST:** Arquitectura utilizada en la web que se caracteriza por no guardar estado entre las peticiones que el cliente realiza al servidor.
[6] **JSON:** Formato de mensajes ligero utilizado para el intercambio de datos.
[7] **WSDL:** Formato XML que se utiliza para describir Servicios Web.

1.2 COMPUTACIÓN UBICUA

1.2.1 Introducción [6]

Para poder entender la importancia de la aparición de la Computación Ubicua y como esta puede ser alcanzada es necesario hacer una revisión de las fases por las cuales ha pasado la computación.

En un principio se vio la aparición de la era del *Main Frame* en una época que la computación era un recurso bastante escaso y se basaba en computadoras almacenadas en habitaciones independientes, manejadas y administradas por usuarios expertos. En esta fase muchas personas utilizaban una computadora para lograr realizar múltiples tareas.

A continuación llego la era de la PC en la cual cada persona puede acceder a su propia computadora, de tal forma que los recursos computacionales ya no tenían que ser compartidos. En esta fase la relación de los usuarios con las computadoras era muy personal.

Posteriormente llego la era del Internet, en la cual gracias al avance de la tecnología y las comunicaciones, facilito que muchas personas y su información estén interconectadas. En esta fase cada usuario además de tener acceso a su propio computador para realizar sus tareas, puede navegar a través del Internet para acceder a grandes servidores y compartir su información con un sinfín de usuarios.

Finalmente llega la Computación Ubicua promovida por Mark Weiser[8], esta era tiene como característica principal que busca que muchas computadoras sirvan a una misma persona, ya que una persona va a estar rodeada de un conjunto de elementos computacionales que van a ayudar a realizar su trabajo.

[8] **Mark Weiser:** Investigador de la universidad de Michigan ideólogo de la computación ubicua.

1.2.2 Definición [7]

La computación ubicua se refiere a la integración de la informática en el entorno de la persona, gracias a lo cual se puede acceder a servicios que antes se pensaba que solo era posible en ciencia ficción, y además la computación ubicua permite acceder a estos servicios cuando sea y en cualquier lugar del planeta.

La Computación Ubicua tiene sus áreas de investigación en diversos campos como: sensores (RFID[9], NFC[10]), redes de próxima generación, sistemas distribuidos, computación móvil, servicios de posicionamiento y localización, desarrollo de sistemas ubicuos especialmente enfocados hacia la medicina, entre otros.

1.2.3 Características [8]

- *Uso eficaz de "Espacios Perspicaces":* Se basa en análisis y detección de las necesidades del usuario en cualquier lugar, un espacio perspicaz se produce cuando varios dispositivos inteligentes coinciden en un mismo espacio físico e interactúan colaborativamente para facilitar las actividades de los usuarios. Un ejemplo de esto es la domótica[11].
- *Invisibilidad:* Consiste en la completa desaparición de la tecnología de la conciencia del usuario, es decir que el uso de la tecnología sea trasparente para el usuario, llegar a un punto en que el uso de la tecnología cause una mínima distracción para el usuario.
- *Escalabilidad local:* Los usuarios disponen de capacidades dependiendo del contexto en que se encuentren, ya que cada usuario va a tener diferentes necesidades dependiendo del entorno en que se encuentre, por ejemplo una aplicación de domótica que se utilice en una vivienda, no va a tener mucho sentido en una oficina.
- *Ocultación de los desniveles de acondicionamiento:* La distribución de los servicios puede ser muy poco uniforme dependiendo del desarrollo

[9] **Radio Frequency Identification (RFID):** Tecnología inalámbrica que utiliza etiquetas para la identificación de objetos o personas.
[10] **Near Field Comunications(NFC)**: Tecnología inalámbrica de corto rango similar a la tecnología RFID
[11] **Domótica:** Conjunto de sistemas capaces de gestionar los distintos aparatos e instalaciones presentes en el domicilio.

tecnológico y la infraestructura disponible. Esta característica no se cumple en los sistemas ubicuos actuales, ya que estos sistemas están aislados, sin continuidad entre unos y otros.

1.3 COMUNICACIONES INALAMBRICAS UTILIZANDO NEAR FIELD COMUNICATION

El mercado de dispositivos móviles que tienen instalado la tecnología NFC ha crecido vertiginosamente durante los últimos años, de acuerdo a un estudio realizado por la compañía Berg Insigth especializada en telecomunicaciones estima que el crecimiento de la base instalada entre los años 2012 a 2017 será de un 65% hasta alcanzar 2.1 billones de unidades que contengan esta tecnología [9].

Actualmente la tecnología NFC está cambiando la forma de realizar distintos tipos de transacciones tales como: transacciones bancarias, sistemas de control de acceso, publicidad entre otros. El uso de este tipo de tecnologías se traduce en menores tiempos de atención a los usuarios, ya que se automatizan los procesos de atención.

1.3.1 DEFINICIÓN [10]

NFC (*Near Field Comunication*) es un tipo de comunicación inalámbrica. Este tipo de comunicaciones se da cuando el uso de conexiones cableadas es imposible o resulta impráctico su uso, la comunicación inalámbrica se puede realizar desde unos pocos centímetros como es el caso de la tecnología NFC hasta muchos kilómetros como es el caso de las redes GSM[12](*Global System for Mobile*) en telefonía celular.

La comunicación NFC implica el intercambio de información en un corto rango de manera inalámbrica, NFC trabaja a una frecuencia de 13.56 MHz y una tasa de trasmisión de no más de 424 Kbps. Esta comunicación se da entre dos

[12]**GSM:** Sistema estándar de telefonía móvil digital que ofrece a más de la función de transmisión de voz ofrece otras funciones como transmisión de datos.

dispositivos; un dispositivo necesariamente debe actuar como dispositivo activo y el otro puede ser un dispositivo activo o pasivo. La comunicación entre los dispositivos es *half-duplex*, es decir mientras un dispositivo está transmitiendo, el otro dispositivo no puede enviar información.

El usuario en la comunicación NFC interactúa con un objeto inteligente que puede ser: una etiqueta NFC, un lector NFC u otro dispositivo móvil con NFC (véase **Figura 1.2**). Esta interacción se da cuando los dos dispositivos se acercan entre sí. Después de la interacción, el dispositivo móvil del usuario obtiene la información necesaria para ejecutar un servicio tal como abrir una aplicación o establecer una conexión con un Servicio Web.

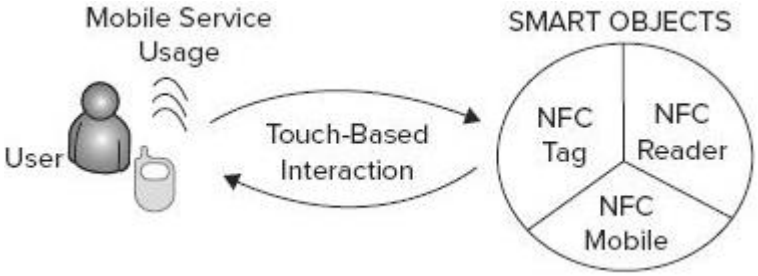

Figura 1.2 Interacción en la comunicación NFC [10]

1.3.2 DISPOSITIVOS NFC [10]

En la tecnología NFC se identifica claramente dos tipos de dispositivos dependiendo de sus características, estos son: dispositivos activos y dispositivos pasivos.

En cada comunicación NFC, el dispositivo que inicia la comunicación se denomina dispositivo *Iniciador* y el dispositivo que responde en la comunicación se denomina dispositivo *Destino*. En la **Tabla 1.1** se puede observar los distintos modos de interacción entre los dispositivos NFC de acuerdo a sus roles.

1.3.2.1 Dispositivos Activos

Un dispositivo activo es aquel que tiene embebido una fuente de poder y pueden generar su propio campo de radiofrecuencia. Este tipo de dispositivos son los

iniciadores de la comunicación y son los que controlan todo el proceso de la comunicación.

INICIADOR	DESTINO
MÓVIL NFC	NFC TAG
MÓVIL NFC	MÓVIL NFC
LECTOR NFC	MÓVIL NFC

Tabla 1.1 Posibilidades de interacción entre dispositivos NFC

Los dispositivos activos que se encuentran en la tecnología NFC son:

- *Dispositivos móviles con NFC integrado*: Son los dispositivos NFC más importantes debido a la diversidad de aplicaciones que ofrece la tecnología NFC en estos dispositivos.
- *Lector NFC*: Un lector NFC está en posibilidad de transferir información a otros módulos de un sistema NFC. El ejemplo común es un terminal de pago NFC[13].

1.3.2.2 Dispositivos Pasivos

Un dispositivo pasivo solo puede responder a las solicitudes de un dispositivo activo, no tiene fuente de alimentación de energía por lo que depende de un dispositivo activo para energizar su circuito.

Los únicos dispositivos pasivos en la tecnología NFC son los tags *NFC;* que en realidad puede ser cualquier tag que trabaje a la frecuencia de 13.56 MHz. Este tipo de dispositivos no tienen una fuente de alimentación integrada al dispositivo.

1.3.3 MODOS DE OPERACIÓN NFC [10]

La tecnología NFC opera en tres modos: lectura/escritura, P2P[14] *(Peer-to-Peer) y* emulación de tarjeta. Cada modo de operación tiene su propia interfaz de

[13] **Terminales de pago NFC:** permiten realizar pagos cuando un móvil con NFC se acerca al lector NFC.
[14] **P2P:** Modo de comunicación entre pares.

comunicación (ISO/IEC 14443[15], FeliCa[16] y NFCIP-1[17]). El modo lectura/escritura permite intercambiar información entre un móvil NFC y un tag NFC. El modo P2P permite el intercambio de información entre dos móviles NFC. En el modo de operación emulación de tarjeta, el usuario interactúa con un lector NFC para usar su móvil NFC como una tarjeta inteligente[18].

1.3.3.1 Modo lectura/escritura

En el modo de operación lectura/escritura se produce la interacción entre el móvil NFC y un tag NFC; este modo internamente define el modo de lectura y el modo de escritura. En el modo de lectura, el dispositivo iniciador lee la información que contiene el dispositivo destino. En el modo de escritura, el dispositivo iniciador escribe datos sobre el dispositivo destino. Este modo de operación trabaja únicamente a tasas de transmisión de 106 Kbps. Un esquema de este modo de operación se puede observar en la **Figura 1.3**.

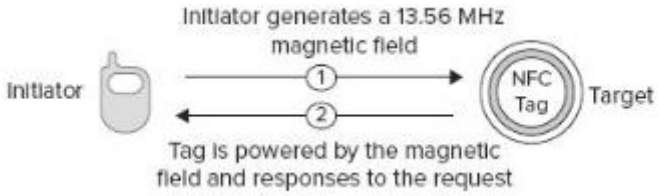

Figura 1.3 Esquema de operación del modo lectura/escritura [10]

El modelo de operación genérico sigue los pasos que se describen a continuación:

1.3.3.1.1 Petición de lectura

El usuario pide información a un tag NFC que puede ser parte de otro componente tal como un aviso, una caja u otro componente. La petición la realiza acercando su móvil NFC al tag.

---

[15] **ISO/IEC 14443:** Estándar desarrollado para tarjetas inteligentes de proximidad, operan a la frecuencia de 13.56 MHz y fue desarrollado por los comités ISO/IEC.
[16] **FeliCa:** Estándar desarrollado para tarjetas inteligentes de alta velocidad, operan a la frecuencia de 13.56 MHz y fue desarrollado por la empresa Sony
[17] **NFCIP-1:** Estándar definido en ISO/IEC 18092, define los protocolos de transporte para NFC.
[18] Una tarjeta inteligente es una tarjeta que tiene incorporada un tag NFC y lleva información del usuario, esta tarjeta suele utilizarse como tarjetas de identificación, de crédito, etc.

1.3.3.1.2 Transferencia de datos

La información que contiene el tag NFC se transfiere al móvil NFC.

1.3.3.1.3 Procesamiento de la información

El móvil NFC procesa la información que recibe y puede ser utilizada para diferentes propósitos tales como: desplegar información al usuario, ejecutar una aplicación o utilizar la información recibida en otras aplicaciones para propósitos adicionales.

1.3.3.1.4 *Procesamiento adicional de la información*

El modo de operación lectura/escritura puede ser utilizado en un modo más complejo, no solo internamente en el móvil NFC, sino también este puede comunicarse con un servidor para realizar más operaciones tales como solicitar información a una base de datos.

1.3.3.1.5 Petición de escritura

El usuario realiza una petición de escritura sobre el tag NFC; usualmente el algoritmo utilizado sobrescribe los datos del tag NFC eliminando toda la información contenida anteriormente, sin embargo se puede modificar el algoritmo para que solo modifique cierta parte de los datos y se conserve el resto de información que se encuentra almacenados en el tag.

1.3.3.1.6 Confirmación de escritura

Al finalizar la operación de escritura el tag responde con un acuse de recibo para informar el éxito de la operación.

Un esquema detallado de este modo de operación se puede observar en la **Figura 1.4**.

1.3.3.2 Modo P2P

Este modo de operación se realiza entre dos móviles NFC. En su modo más simple los únicos participantes en la comunicación son los dos dispositivos, sin embargo un tercer participante puede intervenir, como en el caso de que se requiera obtener datos de un servidor.

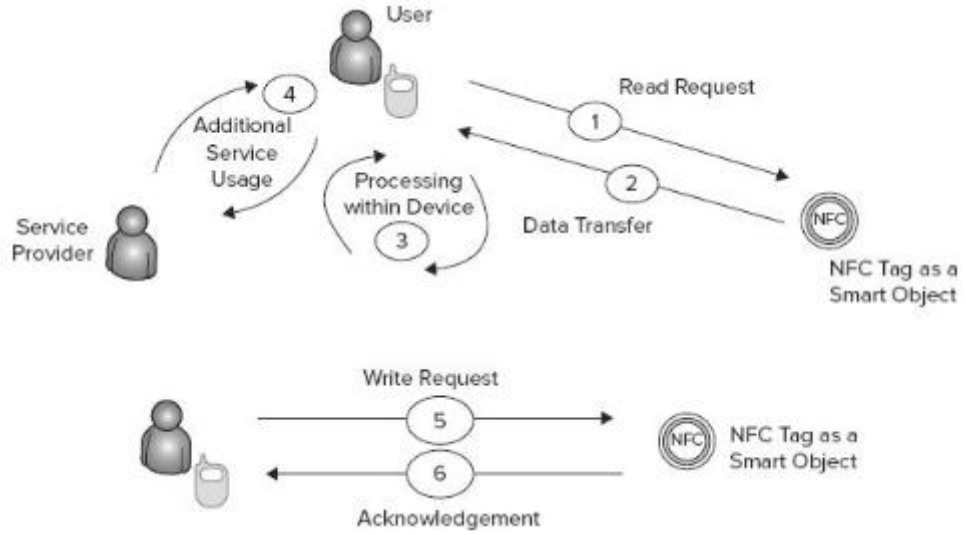

Figura 1.4 Esquema del modo de operación lectura/escritura [10]

En este modo de operación los móviles NFC intercambian datos tales como: información de contacto, mensajes de texto o cualquier tipo de información. P2P tiene dos estándares de comunicación: NFCIP-1(*NFC Interface and Protocol-1*) y LLCP (*Logical Link Control Protocol*).

La información es enviada por ambos dispositivos en un canal *half-duplex*, de tal forma que, cuando un dispositivo transmite, el otro escucha. El segundo dispositivo comienza a transmitir después de que el otro dispositivo termina de transmitir los datos. Las posibles velocidades de transmisión son: 106, 212 y 424 Kbps.

El modelo de operación genérico en P2P sigue los pasos que se describen a continuación:

1.3.3.2.1 Intercambio de información

Los dos usuarios intercambian información con los móviles NFC.

1.1.3.2.1 Usos adicionales de la información

La información que obtiene un móvil NFC del otro puede ser usada para otros propósitos tales como: guardar la información de contacto en una base de datos de Internet o iniciar una solicitud de amistad en alguna de las redes sociales disponibles.

La **Figura 1.5** muestra el esquema de funcionamiento de este modo de operación.

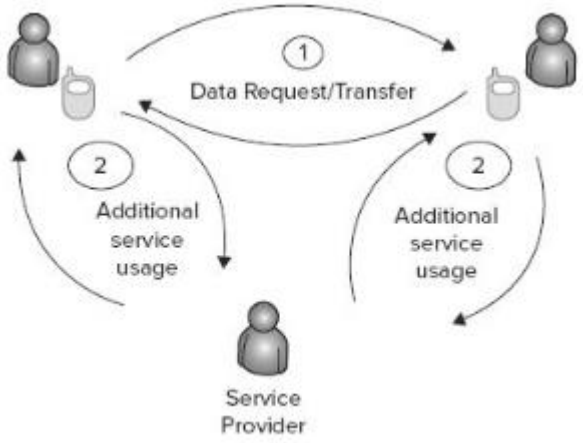

Figura 1.5 Esquema de operación del modo P2P [10]

1.3.3.3 Modo emulación de tarjeta

Este modo de operación permite al móvil NFC actuar como una tarjeta inteligente. El móvil NFC puede guardar múltiples aplicaciones de tarjetas inteligentes, como por ejemplo: tarjetas de débito, tarjetas de crédito o tarjetas de afiliación.

El modo de emulación de tarjeta resulta un modo de operación interesante, debido a que, el móvil NFC, siendo un dispositivo activo, actúa como un dispositivo pasivo. El lector NFC genera su propio campo de radiofrecuencia. Las interfaces de comunicación soportadas en este modo de operación son: ISO/IEC 14443 y FeliCa.

Este modo de operación es importante porque habilita al móvil NFC para realizar pagos y operaciones similares. También resulta práctica esta tecnología porque es compatible con la infraestructura instalada de tarjetas inteligentes. Las etapas de este modo de operación se describen a continuación.

1.3.3.3.1 Petición del servicio

El usuario realiza la petición del servicio acercando su móvil NFC al lector NFC, luego, se transmite la información requerida por el proveedor de servicios.

1.3.3.3.2 Servicios de Background[19]

El proveedor de servicios realiza las operaciones necesarias en segundo plano para aprobar la petición. Por ejemplo pide autorización al banco para aprobar el débito o validar algún ticket.

1.3.3.3.3 Respuesta del servicio

El proveedor luego de ejecutar las operaciones en segundo plano retorna el servicio al cliente, como por ejemplo la autorización de pago o la validación del ticket.

La **Figura 1.6** muestra el esquema de funcionamiento de este modo de operación.

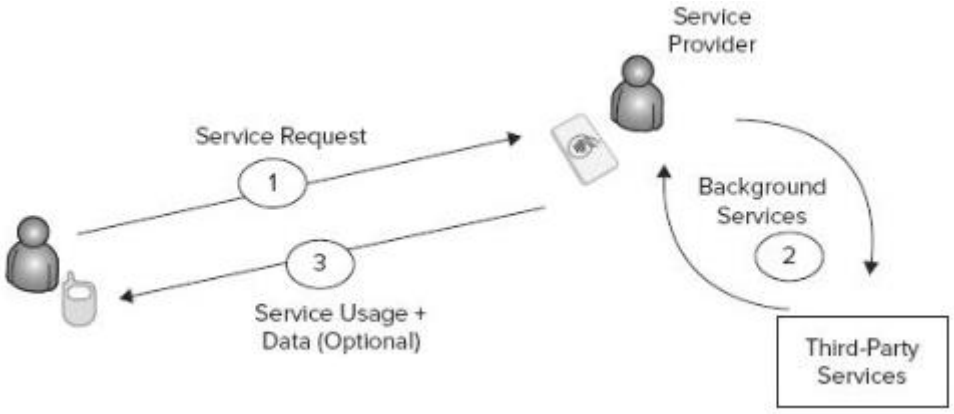

Figura 1.6 Esquema de operación del modo emulación de tarjeta [10]

En esta sección se dio una visión general de los modos de operación disponible en la tecnología NFC. En las próximas secciones se explicará a detalle el modo de operación lectura/escritura que concierne al presente proyecto. Se deja a consideración del lector investigar a detalle los otros dos modos de operación.

[19] **Servicios de *background*:** Son operaciones que se ejecutan en segundo plano y que son transparente para el cliente.

1.3.4 MÓVIL NFC [10]

Un dispositivo móvil con tecnología NFC, comúnmente un teléfono celular, tiene a más de los componentes comunes del dispositivo, la interfaz NFC y el SE[20] (Elemento Seguro).

La interfaz NFC está compuesta por el NFC CLF[21] (NFC Contactless Front End), la antena NFC, y un circuito integrado llamado controlador NFC que habilita las transacciones NFC. Un móvil que tenga integrada la tecnología NFC debe tener al menos un SE integrado, este componente provee un ambiente seguro para las transacciones y guarda información importante del usuario como la información de su tarjeta de crédito. El SE más común presente en estos dispositivos es la tarjeta SIM[22] (*Suscriber Identity Module*), aunque se puede integrar otros SE al móvil. En la **Figura 1.7** se puede apreciar la estructura de un teléfono celular con la tecnología NFC.

1.3.5 ESTANDAR ISO/IEC 14443 [10] [11]

Este estándar describe las transacciones basadas en la interacción entre las tarjetas de proximidad y lectores RFID (*Radio Frequency Identification*). El estándar ISO/IEC 14443 contiene cuatro partes (véase **Tabla 1.2**): la primera parte explica las características físicas; en la segunda parte, se explica los esquemas de funcionamiento del estándar (Potencia RF[23] e interferencia de la señal); en la tercera parte se explica los protocolos de inicialización y anticolisión; finalmente en la cuarta parte se explica los protocolos de transmisión.

[20] **SE:** Elemento de un dispositivo móvil que provee un ambiente seguro para las transacciones y guarda información del usuario.
[21] **NFC CLF:** Circuito en el móvil NFC que maneja la parte analógica de la comunicación NFC.
[22] **SIM:** Tarjeta inteligente desmontable usada en dispositivos móviles, almacena de forma segura la clave del suscriptor.
[23] **RF:** Radio Frecuencia.

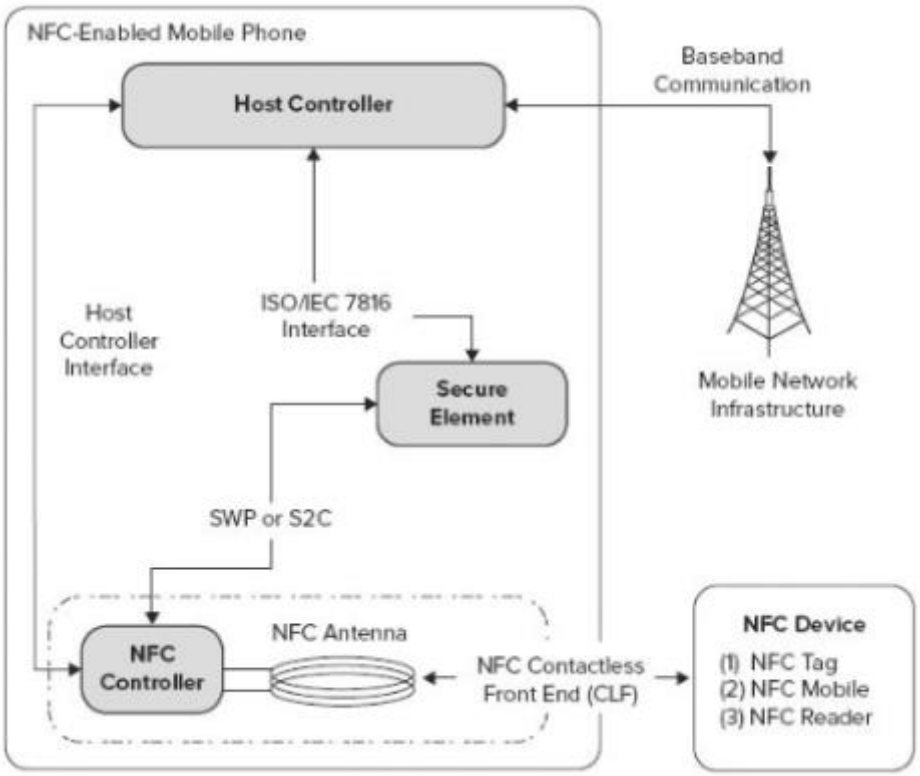

Figura 1.7 Componentes de un teléfono celular con tecnología NFC incorporada [10]

Parte	Contenido
Parte 1: Características físicas de las tarjetas de proximidad.	Define las características físicas de las tarjetas de proximidad, enlista los requerimientos y las pruebas para construir la tarjeta, el diseño de la antena, etc.
Parte 2: Potencia RF e interferencia de la señal.	Define la potencia RF y la interferencia de la señal, esquemas de la señal tipo A y tipo B y también determina como la tarjeta de proximidad es energizada por el campo RF.
Parte 3: Inicialización y Anticolisión.	Define los protocolos de inicialización y anticolisión para las interfaces A y B, así como comandos de anticolisión, respuestas, entramado y sincronización.
Parte 4: Protocolo de transmisión.	Determina los protocolos de transmisión de alto nivel para las interfaces tipo A y tipo B, que son opcionales ya que las tarjetas de proximidad pueden ser diseñadas siguiendo o no los protocolos que se mencionan en esta sección

Tabla 1.2 Partes del estándar ISO/IEC 14443 [10]

En el presente trabajo no se pretende explicar a detalle el funcionamiento de la tecnología NFC sino tener una visión general de su funcionamiento. La parte 2 del Estándar ISO/IEC 14443 permite comprender el modo de operación lectura/escritura de la tecnología NFC, por lo que se analizará a continuación.

1.3.5.1 Parte 2 Estándar ISO/IEC 14443

Las tarjetas de proximidad operan a la frecuencia de 13.56 MHz. El dispositivo activo de la comunicación genera un campo electromagnético que energiza al dispositivo pasivo, la comunicación entre los dispositivos se realiza vía el campo electromagnético que genera la antena del dispositivo activo.

De forma más clara, para efectuar la comunicación entre el dispositivo activo y pasivo los dos dispositivos deben acercarse entre sí; al momento que se acercan los campos electromagnéticos de las dos antenas de los dispositivos se acoplan produciendo el efecto de acoplamiento magnético que se produce en los transformadores (véase **Figura 1.8**).

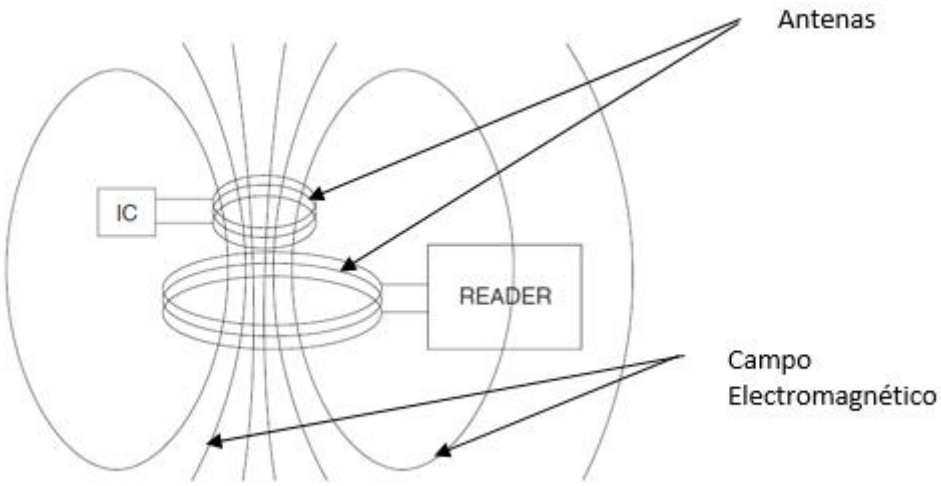

Figura 1.8 Acoplamiento magnético producido entre el dispositivo activo y pasivo [11]

En la antena del dispositivo activo se genera un campo electromagnético producido por la corriente eléctrica alterna que circula por el lazo de la antena. Cuando el dispositivo pasivo entra en el campo electromagnético generado por el dispositivo activo, una corriente alterna se genera en el lazo de la antena del dispositivo pasivo. El dispositivo pasivo posee un circuito integrado (IC *Integrated Circuit*) el cual posee un rectificador y un regulador para convertir la corriente alterna en corriente directa y así poder energizar el IC.

El dispositivo activo modula la amplitud del campo RF generado para enviar información al dispositivo pasivo, el IC del dispositivo pasivo posee un demodulador que convierte la señal modulada en señales digitales. El IC también genera una señal de reloj de 13.56 MHz utilizada para sincronizar la señal recibida. La señal recibida es decodificada y procesada por el IC del dispositivo pasivo.

El dispositivo pasivo se comunica con el dispositivo activo a través del mismo campo electromagnético, la frecuencia de portadora se carga para generar una subportadora de frecuencia aproximada de 847.5 kHz. Esta subportadora permite al dispositivo activo filtrar la información transmitida por el dispositivo activo y decodificar los datos.

1.3.5.1.1 Esquema señalización tipo A

Para la comunicación en el sentido del lector a la tarjeta, se utiliza modulación digital ASK (*Amplitud Shift Keying*) al 100%[24]. En este esquema de señalización, se utiliza el método de codificación de línea *Modified Miller*[25]. La **Figura 1.9** muestra un ejemplo de la señal modulada en sentido del lector a la tarjeta.

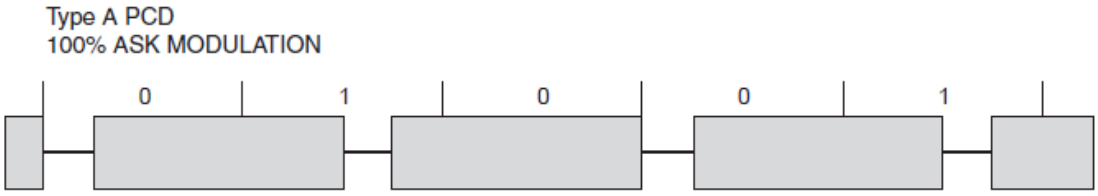

Figura 1.9 Codificación *Modified Miller*, tipo A [11]

Para la comunicación en el sentido de la tarjeta al lector se utiliza modulación OOK[26] (*ON/OFF KEY*) en una subportadora de frecuencia 847.5 KHz. En este esquema de señalización, se utiliza el método de codificación de línea

[24] **Modulación ASK al 100%:** Esquema de modulación digital con índice de modulación igual a 1.
[25] ***Modified Miller:*** Esquema de codificación de línea en donde el cambio de estado se produce a la mitad del tiempo de bit. Este esquema es modificado en el estándar ISO/IEC 14443 por este motivo se llama *Modified Miller.*
[26] **Modulación OOK:** Esquema de modulación con supresión de portadora e índice de modulación igual a 1.

Manchester[27]. La **Figura 1.10** muestra un ejemplo de la señal modulada en sentido de la tarjeta al lector.

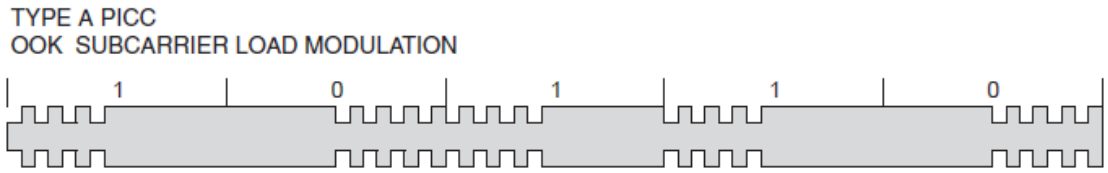

Figura 1.10 Codificación Manchester, tipo A [11]

En la señalización tipo A, el campo RF es apagado en periodos cortos de tiempo, por lo que el circuito integrado del dispositivo pasivo, debe almacenar la suficiente cantidad de energía en sus capacitores internos para continuar funcionando durante este periodo de tiempo.

1.3.5.1.2 Esquema de señalización tipo B

Para la comunicación en el sentido del lector a la tarjeta, se utiliza modulación ASK al 10%[28]. En este esquema de señalización, se utiliza el método de codificación NRZ[29] (*No Return Zero*). La **Figura 1.11** muestra un ejemplo de la señal modulada en el sentido del lector a la tarjeta.

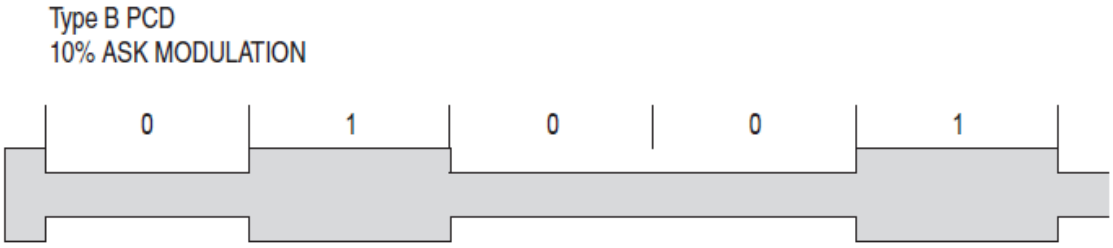

Figura 1.11 Codificación NRZ, tipo B [11]

Para la comunicación en el sentido de la tarjeta al lector, se utiliza modulación BPSK[30] (*Binary Phase Shift Key*) en una subportadora de frecuencia 847.5 KHz.

---

[27] **Manchester:** Esquema de codificación de línea en la que se produce cambio de polarización en la mitad del tiempo de bit.
[28] **Modulación ASK al 10%:** Esquema de modulación digital con índice de modulación igual a 0,1.
[29] **NRZ:** Esquema de codificación de línea en la que se asigna un pulso positivo a uno de los dígitos binarios y un pulso negativo al otro dígito binario de acuerdo a la lógica utilizada.
[30] **BPSK:** Esquema de modulación digital por desplazamiento de fase de dos símbolos.

En este esquema de señalización, se utiliza el método de codificación NRZ-L[31] (*No Return Zero Level*). La **Figura 1.12** muestra un ejemplo de la señal modulada en el sentido de la tarjeta al lector.

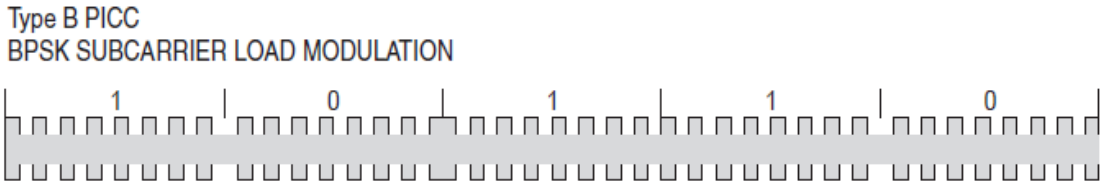

Figura 1.12 Codificación NRZ-L, tipo B [11]

1.3.5.2 Tecnologías de tarjetas de proximidad existentes

En la actualidad existen distintas tecnologías de tarjetas inteligentes sin embargo no todas son compatibles con el estándar ISO/IEC 14443. Las tecnologías más utilizadas son: MIFARE, Calypso y FeliCa.

1.3.5.2.1 MIFARE

Tecnología de tarjeta de proximidad que trabaja a 13.56 MHz desarrollada y de propiedad de NXP Semiconductors (empresa parte de Philips Semiconductors). MIFARE pertenece a las tarjetas tipo A del estándar ISO/IEC 14443. La familia de tarjetas MIFARE puede ser de diferentes tipos tales como: Ultralight, DESfire, Classic, Plus, SmartMX, entre otras. Esta familia de tarjetas se usa para distintos usos, actualmente se utiliza mayoritariamente en transporte público, control de acceso, pago en línea, tarjetas de afiliación, entre otros campos.

1.3.5.2.2 Calypso

Esta familia de tarjetas de proximidad es utilizada en varios sistemas de transporte público en el mundo; fue desarrollado por un grupo de operadores de transito de Bélgica, Alemania, Francia, Italia y Portugal. Este estándar habilita la interoperabilidad entre varios operadores de transporte público en la misma área.

[31] **NRZ-L:** Esquema de codificación de línea diferencial donde el 1L cambia el nivel de la señal codificada anterior y el 0L mantiene el nivel de la señal codificada anterior.

1.3.5.2.3 FeliCa

Tecnología de tarjeta de proximidad que trabaja a 13.56 MHz, pertenece a un sistema desarrollado por Sony y es principalmente utilizado para sistemas de pago.

1.3.6 ARQUITECTURA DEL MODO DE OPERACIÓN LECTURA /ESCRITURA [10]

La interfaz RF del modo de operación lectura/escritura sigue los estándar ISO/IEC 14443 Tipo A y tipo B, y la tecnología de tarjeta de proximidad FeliCa (véase **Figura 1.13**). El foro NFC[32] ha estandarizado varios tipos de tags, el modo de operación de esos tipos de tags y el protocolo de intercambio de datos entre los distintos componentes de la tecnología NFC.

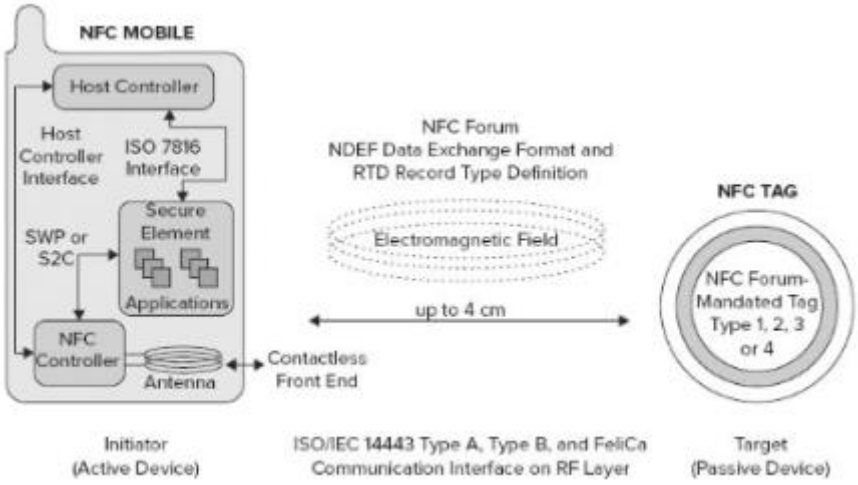

Figura 1.13 Diagrama de bloques del modo lectura/escritura [10]

En el modo de operación lectura/escritura generalmente no se necesita de un SE, ya que no se necesita obtener la identidad del usuario en este tipo de transacciones.

[32] **Foro NFC:** Grupo especial que conforman varias empresas y organizaciones para estandarizar la tecnología NFC.

1.3.6.1 Pila de Protocolos

La **Figura 1.14** muestra una visión general de la arquitectura de este modo de operación. A continuación se describe brevemente las capas presentes en el modo de operación lectura/escritura.

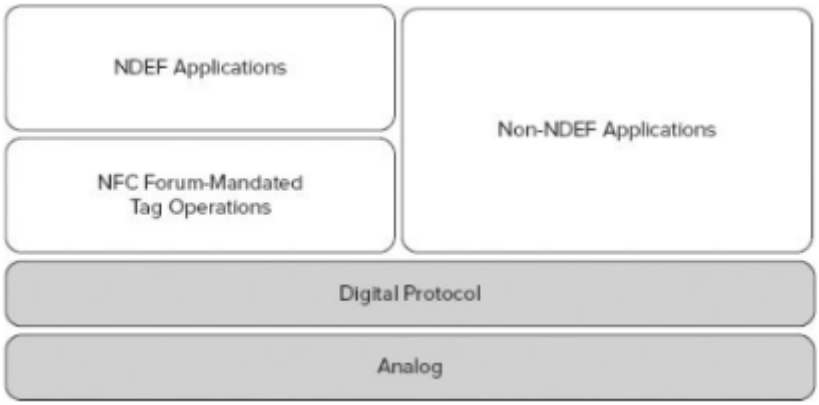

Figura 1.14 Arquitectura del modo de operación lectura/escritura [10]

- La capa *Analog* define las características RF de los dispositivos NFC y sus frecuencias de operación. En la capa *Digital Protocol* se define los aspectos de entramado de la información, estas dos capas se encuentran definidas en el estándar ISO/IEC 14443.
- *NFC Forum-Mandated Tag Operations* indica los comandos e instrucciones para operar con tags estandarizados por el foro NFC. De esta manera se habilita la comunicación usando el formato de datos NDEF[33] (NFC *Data Exchange Format*) y RTD[34] (*Record Type Definitions*).
- *NDEF Applications* son las aplicaciones basadas en el protocolo NDEF.
- *Non-NDEF Applications* son aplicaciones propietarias que no siguen el protocolo NDEF, un ejemplo de este tipo de aplicaciones son las aplicaciones de pago o tickets de acceso.

[33] **NDEF:** Formato de intercambio de mensajes entre dos dispositivos NFC estandarizado por el foro NFC.
[34] **RTD:** Define el tipo de mensaje que lleva un registro NDEF

1.3.6.2 Tipos de TAG de acuerdo al foro NFC

El foro NFC define cuatro tipos de tags designándolos de 1 al 4, cada uno tiene diferente formato y capacidades. Estos tags están basados en los estándares ISO/IEC 14443 Tipo A, ISO/IEC 14443 Tipo B o Sony FeliCa. La **Tabla 1.3** muestra un resumen de las características de los cuatro tipos de tags.

Parámetro	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Estándar	ISO/IEC 14443 Tipo A	ISO/IEC 14443 Tipo A	FeliCa	ISO/IEC 14443 Tipo A, Tipo B.
Nombre del Chip	Topaz	MIFARE	FeliCa	DESFire, SmartMX-JCOP
Tamaño de memoria	Hasta 1 KB	Hasta 2 KB	Hasta 1 MB	Hasta 64 KB
Tasa de transmisión (Kbps)	106	106	212	106-424
Costo	Bajo	Bajo	Alto	Medio/Alto
Seguridad	Firma digital de 16 o 32 bits	Insegura	Firma digital de 16 o 32 bits	Variable

Tabla 1.3 Comparación de los tags NFC estandarizados en el foro NFC [10]

1.3.6.3 NFC Data Exchange Format (NDEF) [12] [10]

NDEF es un formato de intercambio de mensajes entre dos dispositivos NFC estandarizado por el foro NFC. Se intercambia el mensaje NDEF cuando un dispositivo pasivo o activo está en la proximidad de un dispositivo activo, en cualquier caso el formato del mensaje es el mismo.

NDEF es un formato binario de mensaje que encapsula una o varias aplicaciones en un solo mensaje (véase **Figura 1.15**), es decir, un mensaje NDEF puede enviar uno o varios registros NDEF. Cada registro puede llevar una carga útil de hasta $2^{32} - 1$ octetos. En este formato de mensaje no existe un número límite de registros que se pueden agrupar.

En un mensaje NDEF, en el primer registro se activa la bandera MB (*Message Begin*) y el último registro se activa la bandera ME (*Message End*). El tamaño mínimo de un mensaje NDEF es un registro, en este caso en el mismo mensaje se marcarán los campos MB y ME.

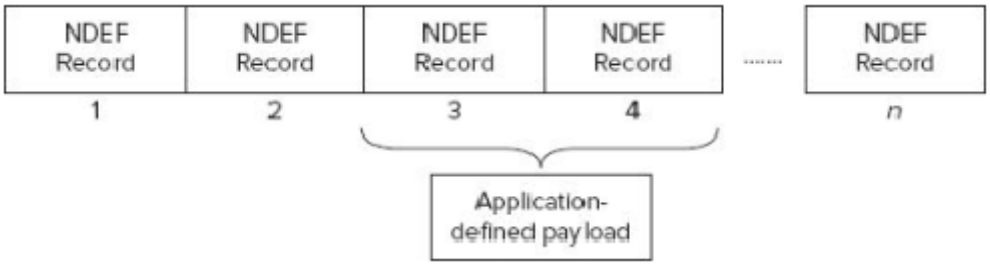

Figura 1.15 Estructura de un mensaje NDEF [10]

Cada registro NDEF lleva consigo tres parámetros para describir su carga útil, estos parámetros son:

- *Payload length:* indica el número de octetos presentes en la carga útil del registro NDEF.
- *Payload type:* indica el tipo de información de la carga útil. Indicando el tipo de formato se puede lanzar la aplicación apropiada en un dispositivo NFC.
- *Payload identifier:* este parámetro es opcional, permite a las aplicaciones del usuario identificar la carga útil que lleva un registro NDEF.

En la **Figura 1.16** se puede observar la estructura del registro NDEF, a continuación se explica a detalle cada uno de los campos que conforma el registro NDEF.

1.3.6.3.1 Message Begin (MB)

Campo de 1 bit que indica el comienzo de un mensaje NDEF.

1.3.6.3.2 Message End (MB)

Campo de 1 bit que indica el fin de un mensaje NDEF.

1.3.6.3.3 Chunk Flag (CF)

Campo de 1 bit que indica si es el primer registro o el registro intermedio de un mensaje NDEF seccionado.

1.3.6.3.4 Short Record (SR)

Campo de 1 bit que indica que la longitud de la carga útil es un solo octeto.

1.3.6.3.5 IL

Campo de 1 bit que indica que el campo ID LENGTH está presente en la cabecera como un único octeto.

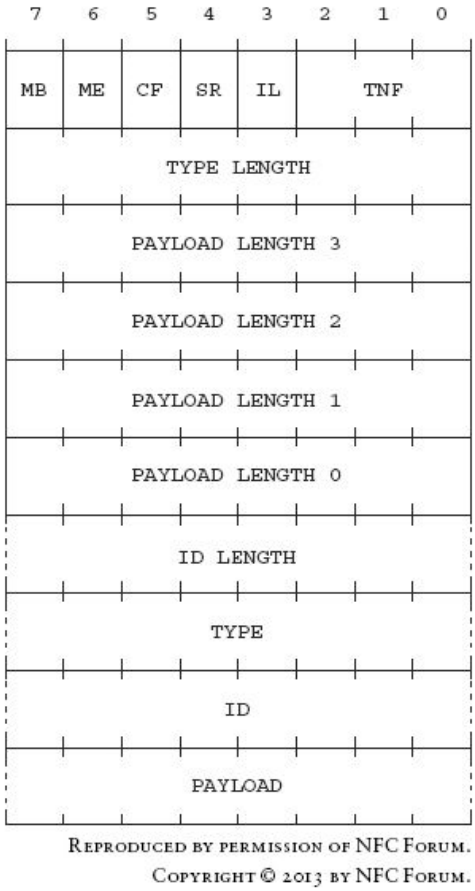

Figura 1.16 Estructura de un registro NDEF [10]

1.3.6.3.6 Type Name Format (TNF)

Representa la estructura del valor del campo tipo. Es un campo de 3 bits definido según la **Tabla 1.4**.

1.3.6.3.7 TYPE LENGTH

Campo de 8 bits que específica la longitud en octetos del campo TYPE.

1.3.6.3.8 ID LENGTH

Campo de 8 bits que especifica la longitud en octetos del campo ID.

Nombre del Tipo de formato	Valor
Empty	0x00
NFC Forum well-known type	0x01
Media-type	0x02
Absolute URI	0x03
NFC Forum external type	0x04
Unknown	0x05
Unchanged	0x06
Reserved	0x07

Tabla 1.4 Valores del campo TNF [10]

1.3.6.3.9 PAYLOAD LENGTH

Este campo especifica la longitud en octetos del campo PAYLOAD. El tamaño de este campo se define mediante la bandera SR.

1.3.6.3.10 TYPE

Este campo describe el tipo de payload (carga útil).

1.3.6.3.11 ID

El valor de este campo es un identificador en la forma de una referencia URI[35] (*Uniform Resource Identifier)*.

1.3.6.3.12 PAYLOAD

Este campo lleva la carga útil de la aplicación.

1.4 SISTEMA OPERATIVO ANDROID

1.4.1 DEFINICIÓN [10]

Android es el sistema operativo mayormente utilizado en dispositivos móviles, está basado en una versión modificada del kernel de Linux. Android es propiedad de la Open Handset Alliance*[36],* por lo que Android es completamente software

[35] **URI:** Identifica los recursos de una red en forma univoca.
[36] **Open Handset Alliance:** Alianza comercial de compañías dedicadas a desarrollar estándares abiertos para dispositivos móviles.

libre. Actualmente el proyecto es liderado por la compañía Google y la mayoría del código es producido bajo la licencia de Apache.

Las aplicaciones son desarrolladas en el lenguaje de programación Java, en este lenguaje de programación el código es convertido en *bytecodes*[37] y son ejecutados por la JVM[38] (*Java Virtual Machine*). Sin embargo, en Android las clases son compiladas en archivos ejecutables Dalvik y son ejecutados por la DVM[39] (*Dalvik Virtual Machine*). Después de codificar una aplicación en el SDK de Android, este SDK compila el código en un paquete de Android con la extensión .apk la cual es subida e instalada en el dispositivo móvil para su ejecución.

Todo dispositivo móvil de Android tiene instalada una versión del sistema operativo Android, existen diversas versiones del sistema operativo y con el desarrollo tecnológico van saliendo nuevas versiones en las que se incluyen mejoras. Es una buena práctica a la hora de desarrollar una aplicación para Android, considerar la versión del sistema operativo a la cual va dirigida la aplicación; considerando diferentes factores como hardware y software requerido en el dispositivo móvil donde se va a instalar la aplicación.

Una aplicación de Android que se ejecuta en un dispositivo móvil, puede utilizar servicios desarrollados dentro de la misma aplicación o puede usar servicios ya disponibles en el dispositivo. Por ejemplo, si la aplicación necesita grabar audio, resultaría más práctico utilizar servicios ya disponibles en el dispositivo que realicen esta funcionalidad.

1.4.2 ARQUITECTURA [10]

Los componentes más importantes del sistema operativo Android son: *Linux kernel*, *Android run time*, *libraries*, *application framework,* y *applications.* En la **Figura 1.17** se puede apreciar la estructura de esta arquitectura.

[37] **Bytecodes:** Código entendido por la JVM.
[38] **JVM:** Ambiente donde se ejecuta los byte codes.
[39] **DVM:** Máquina Virtual que permite ejecutar las aplicaciones programadas en Java para dispositivos móviles.

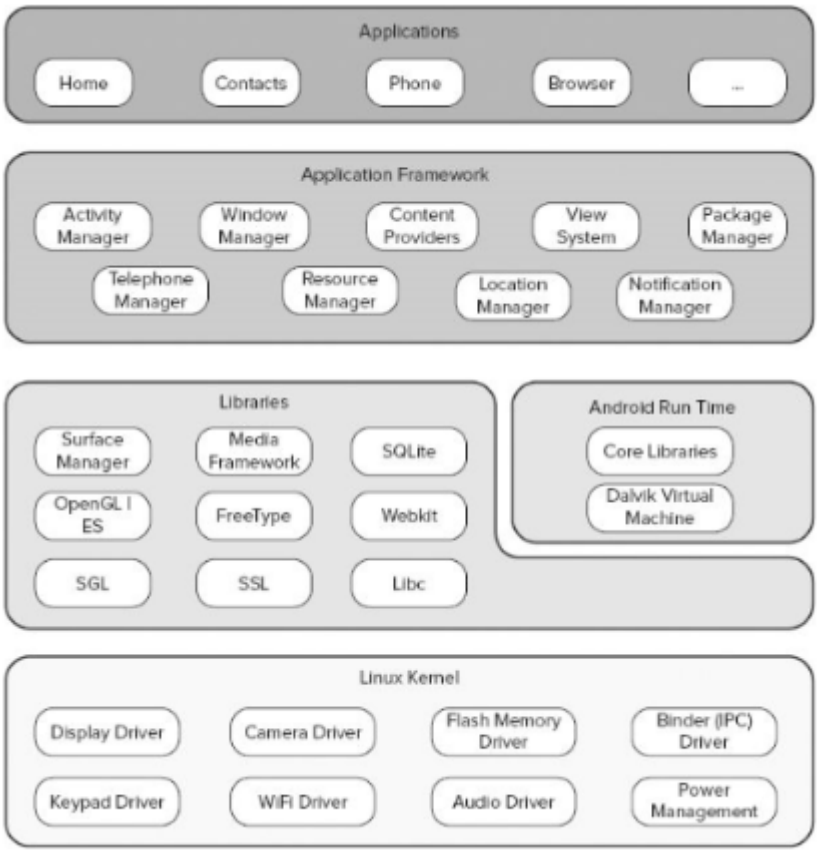

Figura 1.17 Arquitectura de Android [10]

1.4.2.1 Linux Kernel

Esta capa sirve para la comunicación entre los componentes de hardware y software. Android está basado en la versión 2.6 del kernel de Linux, la cual provee servicios del núcleo del sistema. Desde la versión 4.0 (*Ice Cream Sandwich),* la versión del kernel de Linux utilizada es la 3.x.

1.4.2.2 Android Runtime

Android incluye un conjunto de librerías de núcleo que tiene la mayoría de funcionalidades disponibles en las librerías de núcleo del lenguaje de programación Java. Cada aplicación de Android ejecuta su propio proceso con una instancia diferente de la DVM. Dalvik da la posibilidad a un dispositivo de ejecutar varias máquinas virtuales de manera eficiente. DVM ejecuta archivos Dalvik (.dex) que optimiza el uso de la memoria.

1.4.2.3 Libraries

Android incluye un conjunto de librerías escritas en C/C++ que provee las capacidades necesarias al sistema Android. Estas capacidades son dadas por los desarrolladores a través de la capa *application framework*.

1.4.2.4 Application Framework

Android provee a los desarrolladores una plataforma de desarrollo abierta, lo que les permite desarrollar aplicaciones innovadoras. Los desarrolladores pueden hacer uso de los distintos componentes y servicios de Android como: acceder al hardware del dispositivo, utilizar información de localización, ejecutar servicios en segundo plano, entre otras.

1.4.2.5 Applications

Android incluye un conjunto de aplicaciones esenciales como: cliente de e-mail, programa de SMS[40] (*Short Message Service*), calendario, mapas, navegador y libreta de contactos. Todas estas aplicaciones son escritas utilizando el lenguaje de programación Java.

1.4.3 ESTRUCTURA DE LAS APLICACIONES DE ANDROID [10]

1.4.3.1 Componentes de una aplicación de Android

Existen cuatro componentes en las aplicaciones Android que son: actividades, servicios, receptores de anuncios, y proveedores de contenido. Cada aplicación usa al menos una actividad y también puede usar cualquier número de los otros componentes.

[40] **SMS:** Servicio de mensajería instantánea utilizado en las redes móviles

1.4.3.1.1 Actividades

Una actividad representa una interfaz de usuario de la aplicación. Una aplicación debe contener al menos una actividad. Cada actividad presenta una vista de pantalla al usuario, es decir, el número de actividades presente en una aplicación es igual al número de vistas que el usuario puede visualizar en una aplicación.

1.4.3.1.2 Servicios

Un servicio es una tarea en segundo plano que no proporciona ninguna interfaz al usuario pero realiza operaciones en segundo plano. Los servicios persisten incluso si el usuario sale de la aplicación, por ejemplo, el servicio de e-mail puede continuar recibiendo correos aún cuando se estén ejecutando otras aplicaciones.

1.4.3.1.3 Receptores de Anuncios

Son los componentes que reciben y reaccionan a los anuncios. La mayoría de anuncios se origina del sistema operativo, por ejemplo: anuncio de baja batería, anuncio de apagado de pantalla, entre otros.

1.4.3.1.4 Proveedores de Contenido

Un proveedor de contenido pone a disponibilidad de otras plataformas los datos de una aplicación. A través de estos proveedores de contenidos una aplicación puede compartir datos con otros dispositivos o aplicaciones.

1.4.3.2 Intents

Las actividades, los servicios y los receptores de anuncios son desencadenados por los *intents,* en el caso de las actividades, los *intents* definen las acciones a realizar; por ejemplo al acercar un dispositivo móvil con NFC a un tag, se invoca un nuevo *intent* para abrir una nueva actividad informativa.

Los *intents* pueden ser implícitos o explícitos. Un *intent* explicito llama a un servicio o a una actividad explícitamente; en cambio un *intent* implícito solo da una definición del servicio requerido y el sistema operativo Android selecciona el servicio apropiado de los que tiene disponible.

1.4.3.3 Intent Filters

Actividades, servicios y receptores de anuncio pueden tener uno o más *intent filters.* Cada *intent filter* filtra el *intent* implícito deseado, solo el *intent* implícito que pasa este filtro es entregado al componente. Los *intents* explícitos no son filtrados por los *intent filters.*

1.4.3.4 Archivo Manifiesto

La primera actividad que realiza el sistema operativo Android al momento de abrir una nueva aplicación es leer el archivo manifiesto[41]. Este archivo incluye información acerca de todos los componentes correspondiente a la aplicación. Adicionalmente, el archivo manifiesto tiene otras funciones:

- Identifica los permisos de usuario que la aplicación requiere, por ejemplo acceso a Internet.
- Declara el nivel mínimo de API[42] (*Application Programming Interface)* requerido por la aplicación, en base a las API usadas por la aplicación.
- Declara las características de hardware y software requeridas por la aplicación, por ejemplo servicios de cámara, bluetooth, NFC entre otros.
- Declara las API que la aplicación necesita enlazar, por ejemplo la API de Google Maps.

La tarea principal del archivo manifiesto es informar al sistema sobre los componentes de una aplicación. Adicionalmente, para declarar un componente en el manifiesto de la aplicación, opcionalmente se puede declarar en el manifiesto de la aplicación un *intent filter* para definir las capacidades del componente.

---

[41] **Archivo Manifiesto:** Entre los archivos del .apk se encuentra el archivo AndroidManifest.xml el cual una vez instalada la aplicación en el dispositivo móvil se aloja en el directorio raíz de la aplicación

[42] **API:** Conjunto de herramientas que ofrece Android dependiendo de la versión del sistema operativo.

1.4.3.5 Recursos de la aplicación

Una aplicación de Android está compuesta por varios recursos que están separados del código fuente, por ejemplo: imágenes, texto o cualquier elemento relacionado a la presentación visual de la aplicación. En una aplicación de Android se puede definir estos elementos en archivos XML.

Separando los recursos del código fuente se puede actualizar las características de la aplicación sin necesidad de modificar el código fuente. Todo recurso incluido en la aplicación tiene un ID que identifica de manera única al recurso, este ID puede ser usado para referenciar el recurso dentro del código de la aplicación.

1.4.3.6 Hilos y Procesos

Cuando se llama a una nueva aplicación y si la aplicación no tiene ningún otro proceso activo, el sistema Android inicia un nuevo proceso para la aplicación con un único hilo de ejecución. Por defecto, todos los componentes de una misma aplicación se ejecutan sobre el mismo hilo llamado hilo principal. Si se ejecuta un componente de la aplicación y un proceso de la misma aplicación ya existe, el componente se inicia dentro de ese proceso y usa el mismo hilo de ejecución. Sin embargo se puede hacer que diferentes componentes de la aplicación se ejecuten en procesos separados, y se pueden crear hilos adicionales para cualquier proceso.

El sistema Android utiliza una jerarquía de importancia para determinar los procesos que debe mantener y los procesos que se deben eliminar, esta jerarquía se basa en los componentes que se están ejecutando en un proceso y el estado de esos componentes. Procesos con la importancia más baja son los primeros en ser descartados.

Los procesos son divididos en cinco niveles de importancia. Cada tipo de proceso se describe de acuerdo a su nivel de importancia.

1.4.3.6.1 Proceso en Primer Plano

Los procesos en primer plano se relacionan directamente con la actividad actual que el usuario esté realizando. Un proceso es considerado de primer plano si realiza cualquiera de las siguientes actividades o servicios:

- Una actividad con la que el usuario está interactuando, donde la actividad ha llamado el método `onResume()`[43].
- Un servicio que está enlazado a una actividad con la que el usuario está interactuando.
- Un servicio que fue llamado por el método `startForeground()`.
- Un servicio que está ejecutando uno de las llamadas de su ciclo de vida: `onCreate()`, `onStart()`, `onDestroy()`.
- Un receptor de anuncios que está ejecutando el método `onReceive()`.

1.4.3.6.2 Proceso Visible

Un proceso visible no tiene ningún componente en primer plano, pero tiene efecto en lo que el usuario ve en la pantalla. Un proceso es considerado visible si realiza cualquiera de las siguientes actividades o servicios:

- Una actividad que no está en primer plano, pero aún está visible para el usuario ya que se llamó al método `onPause()`.
- Un servicio que se encuentra enlazado a una actividad en primer plano.

1.4.3.6.3 Proceso de Servicio

Es un servicio que se ha iniciado mediante el método `startService()`. El sistema mantiene al proceso ejecutándose hasta no tener suficiente memoria para mantener todos los procesos de primer plano y visibles.

1.4.3.6.4 Proceso en Segundo Plano

Un proceso en segundo plano se ejecuta cuando una actividad no está visible para el usuario, en este caso, la actividad ha llamado al método `onStop()`.

[43] Los métodos que se hablan en esta sección se explicarán a detalle en las siguientes secciones.

Estos procesos no tienen influencia directa con la experiencia del usuario y el sistema los puede eliminar en cualquier momento para obtener memoria para los procesos de primer plano, visibles o procesos de servicio.

1.4.3.6.5 Proceso Vacío

Un proceso vacío no mantiene ningún componente de las aplicaciones activas. El único propósito de este tipo de procesos es para mejorar el inicio de ciertas aplicaciones de manera más rápida.

1.4.4 MÁQUINA VIRTUAL DALVIK (DVM)

Esta máquina virtual utiliza un lenguaje de maquina llamado *dex bytecode* desarrollado especialmente para los archivos ejecutables de Android.

Cuando una aplicación de Android se implementa, el compilador de Java convierte el código fuente en *bytecodes* (archivos de clase), luego un convertidor "dx" convierte los *bytecodes* en archivos ejecutables Dalvik (dex). Por último, el archivo dex se comprime en un paquete de Android (archivo apk) que es el que se instala en los dispositivos móviles. En la **Figura 1.18** se puede observar este proceso de forma más clara.

Cuando se utiliza el IDE[44] (*Integrated Development Enviroment*) de Android el proceso de convertir el código fuente al paquete apk es transparente.

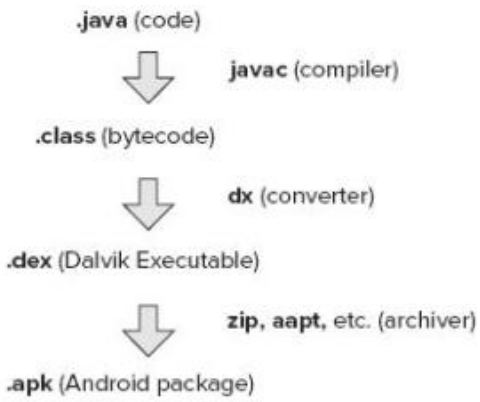

Figura 1.18 Proceso de generación de un paquete de Android [10]

[44] **IDE:** Software compuesto por un conjunto de herramientas de programación.

1.4.5 CICLO DE VIDA DE LAS ACTIVIDADES EN ANDROID [10]

En el sistema Android, las actividades son manejadas por pilas. Cuando se inicia una nueva actividad, esta es llevada al frente de la pila de actividades y se hace visible. Las otras actividades en ejecución permaneces activas si estas no son cerradas, pero estas son llevadas a niveles más bajos de la pila de actividades. Cuando se detiene una actividad que está en el frente de la pila, la actividad que le sigue regresa al frente. Una actividad tiene cuatro estados importantes que se ilustran en la **Figura 1.19**.

- *Activa:* Una actividad está activa cuando está al frente de la pila de actividades.
- *Pausada:* Una actividad está pausada cuando es visible pero otra actividad está al frente de la pila de actividades de forma minimizada o como una ventana transparente. Una actividad en este estado mantiene la información pero puede ser finalizada cuando el sistema Android requiere memoria para otros procesos.
- *Detenida:* Una actividad está detenida cuando se inicia otra actividad, esta nueva actividad pasa la frente de la pila de actividades. Esta actividad puede ser detenida en cualquier momento si se necesita espacio en memoria.
- *Destruida*: Una actividad está destruida cuando el sistema solicita terminar el proceso.

La **Tabla 1.6** y la **Tabla 1.6** describen los métodos usados en cada etapa del ciclo de vida. Esta tabla incluye información de cuando se llama a la actividad y cual método puede seguir al método actual.

Método	Cuando es llamado	Seguido por
`onCreate()`	Cuando la actividad se crea inicialmente.	Siempre por `onStart()`.
`onStart()`	Cuando la actividad se hace visible, lo que incluye después de crearse la actividad y cuando la actividad regresa del estado detenido.	`onResume()` cuando la actividad regresa al frente de la pila de actividades, `onStop()` cuando la actividad se vuelve oculta.

Tabla 1.5 Métodos presentes en el ciclo de vida de las actividades [10]

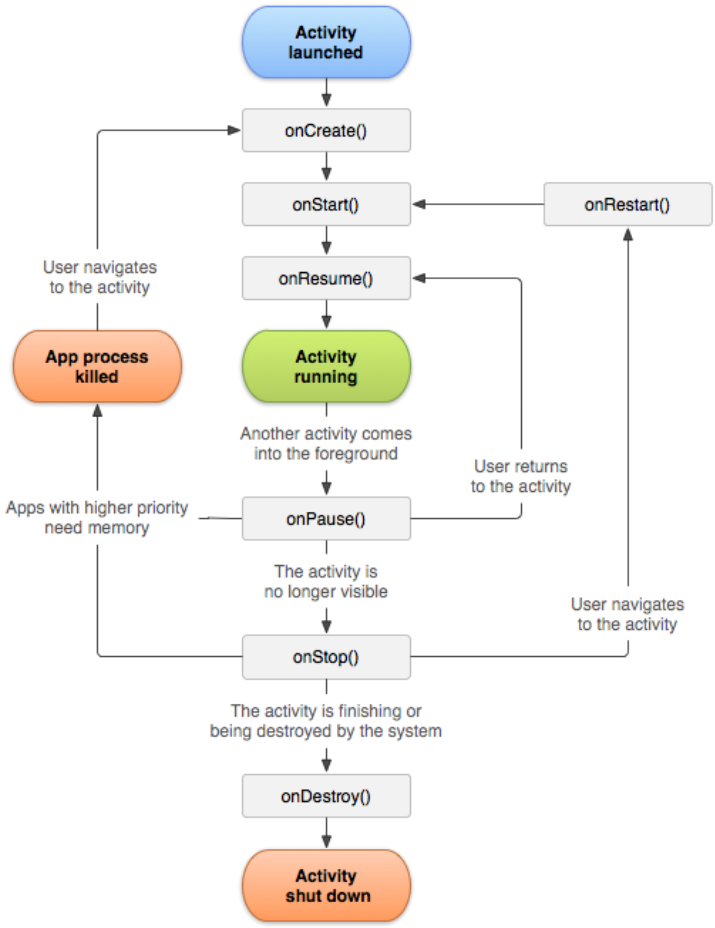

Figura 1.19 Diagrama de flujo del ciclo de vida de una actividad en Android [13]

Método	Cuando es llamado	Seguido por
`onRestart()`	Cuando la actividad que está detenida es iniciada de nuevo.	Siempre por `onStart()`.
`onResume()`	Cuando una actividad comienza a interactuar con un usuario de nuevo.	Siempre por `onStart()`.
`onPause()`	Cuando otra actividad diferente se reanuda.	`onResume()` cuando la actividad regresa al frente, `onStop()` cuando la actividad no es visible para el usuario.
`onStop()`	Cuando la actividad ya no es visible.	`onRestart()` cuando la actividad vuelve a interactuar con el usuario, `onDestroy()` si la actividad existe.
`onDestroy()`	Cuando la actividad es destruida.	Ninguna.

Tabla 1.6 Métodos presentes en el ciclo de vida de las actividades (continuación) [10]

1.4.6 SDK ANDROID [13]

El SDK de Android proporciona las librerías y herramientas necesarias para compilar y probar las Aplicaciones de Android.

1.4.6.1 HERRAMIENTAS DEL SDK [13]

Las herramientas del SDK son instaladas con el paquete inicial de Android y son periódicamente actualizadas. Las herramientas del SDK son necesarias para el desarrollo de las aplicaciones de Android. Las herramientas del SDK incluyen el Android SDK Manager, el AVD Manager, el Emulador y el Dalvik Debug Monitor Server.

A continuación se describen las principales herramientas que posee el SDK de Android para ayudar al programador en el desarrollo de aplicaciones.

- *Android:* Permite manejar los AVD[45] (Android Virtual Devices), los proyectos y los componentes instalados del SDK.
- *Dalvik Debug Monitor Server (ddms):* Permite la depuración de las Aplicaciones de Android.
- *Dmtracedump:* Genera diagramas gráficos del *call-stack*[46] de los archivos de logs, Esta herramienta utiliza el utilitario *Graphviz Dot*[47] para generar la salida del gráfico.
- *Draw 9-patch:* Permite fácilmente crear un gráfico *NinePatch*[48] por medio del uso de un editor WYSIWYG[49] (*What You See Is What You Get*). Este editor además nos proporciona vistas detalladas de la imagen, y resaltes del área con el contenido permitido.
- *Android Emulator:* Permite el desarrollo y prueba de aplicaciones Android sin la necesidad de utilizar un dispositivo físico.

---

[45] **AVD:** Herramienta del SDK de Android que permite probar aplicaciones desde un computador, sin tener el dispositivo Android físico.
[46] **Call-stack:** Es una estructura de datos dinámica "*Last Input, First Output*", utilizada para el almacenamiento de información [13].
[47] **Graphviz Dot:** Herramienta de código abierto para generar diagramas de *software*.
[48] **NinePatch:** Clase que permite dibujar un bitmap en nueve o más secciones.
[49] **WYSIWIG:** Editores que permite ver el resultado final al momento de elaborarse.

- *Hierarchy Viewer:* Proporciona depuración y optimización de las interfaces de la aplicación Android.
- *Hprof-conv:* Convierte el archivo HPROF que es generado por las herramientas estándar del SDK en un formato estándar escogido que se pueda ver.
- *Layoutopt:* Proporciona un rápido análisis de los *layouts*[50] de las aplicaciones con el fin de optimizarlas y hacerlas más eficientes.
- *Mksdcard:* Herramienta que permite rápidamente crear la imagen virtual de un disco FAT32[51] que se puede cargar en el emulador, con el fin de simular la presencia de una tarjeta SD[52] (*Secure Digital*) en el dispositivo virtual.
- *Monkey:* Se ejecuta en el dispositivo o emulador y genera cadenas pseudo-aleatorias de eventos de usuarios (*clicks*, *touches*, o gestos) como un número de niveles de sistema de eventos. Se puede usar Monkey para stress-test[53] de las aplicaciones que se estén desarrollando.
- *Monkeyrunner:* Provee una API para escribir programas que controlen el dispositivo Android o emular la salida del código Android.

1.5 FRAMEWORK .NET

1.5.1 INTRODUCCIÓN [14]

Framework .NET es una plataforma de desarrollo que permite la creación de aplicaciones windows. Esta plataforma incluye lenguajes de programación como C# y Visual Basic. Sus componentes principales son: la biblioteca de clases que proporciona un conjunto de código probado y el motor de ejecución que es el encargado de controlar las aplicaciones en ejecución.

[50] **Layouts:** Archivo que define la estructura visual de una interfaz de usuario, como por ejemplo una actividad.
[51] **Fat32:** Tabla de asignación de archivos que da el formato de un disco de almacenamiento.
[52] Dispositivo extraíble que permite el almacenamiento de información.
[53] **Stress-test:** Es un modo de pruebas de modo intenso y profundo que se realiza con el fin de probar la estabilidad del sistema

1.5.2 SERVICIOS DE .NET FRAMEWORK [14]

- *Administración de Memoria: .NET Framework* permite asignar y liberar memoria y administrar la vida útil de los objetos.
- *Sistemas de Tipo Comunes:* El sistema de tipos de *Framework .NET* establece los tipos básicos y estos tipos son comunes para todos los sistemas que tienen como fin *.NET Framework*.
- *Biblioteca de Clases Extensa:* Los programadores pueden utilizar la biblioteca de tipos y las clases de .NET en todo momento.
- *Framework y Tecnologías de Desarrollo:* En .NET se tienen bibliotecas para varias áreas de desarrollo como ASP.NET para aplicaciones web y WCF[54] (*Windows Comunication Foundation*) para aplicaciones orientadas a servicios.
- *Interoperabilidad de Lenguajes:* Los compiladores emiten un código intermedio denominado Lenguaje Intermedio Común que en tiempo de ejecución es ejecutado por el CLR[55] (*Common Language Runtime)*, gracias a esto las rutinas escritas en un determinado lenguaje están disponibles para otro lenguaje.
- *Compatibilidad de Versiones:* Las aplicaciones realizadas en *.NET Framework* se pueden ejecutar sin cambios en las versiones posteriores.
- *Ejecución en Paralelo:* .NET permite que varias versiones de CLR coexistan y funcionen paralelamente en el mismo equipo. Esto quiere decir que la aplicación se puede ejecutar con la versión de *.NET Framework* en que se compiló.
- *Compatibilidad con múltiples versiones:* Se pueden crear ensamblados que funcionen en varias plataformas de .NET Framework como Windows Server 2012, Windows 7, Windows 8, Windows Phone y Xbox 360, etc.

[54] **WCF:** Modelo de programación unificado de Microsoft para compilar aplicaciones orientadas a servicios.
[55] **CLR:** Motor de ejecución del *Framework .NET*

1.5.3 WINDOWS COMMUNICATION FOUNDATION (WCF) [14]

Es un modelo de programación unificado de Microsoft para compilar aplicaciones orientadas a servicios. WCF permite a los programadores compilar aplicaciones con transacciones seguras y de confianza para múltiples plataformas.

1.5.3.1 Conceptos básicos de WCF

1.5.3.1.1 Mensajería y extremos

WCF se basa en la comunicación mediante mensajes o cualquier cosa que se pueda modelar como un mensaje (por ejemplo, una petición HTTP).

WCF distingue entre clientes, los cuales pueden ser aplicaciones que inician la comunicación, y servicios, los cuales pueden ser aplicaciones que esperan a que los clientes se comuniquen con ellos y respondan a la comunicación.

Los mensajes son enviados entre extremos, los cuales son lugares donde los mensajes se envían o se reciben. Un servicio genera uno o más extremos de la aplicación, y el cliente genera un extremo compatible con uno de los extremos del servicio.

1.5.3.1.2 Protocolos de comunicaciones

En WCF se debe definir el protocolo de transporte usado. Los mensajes se pueden enviar a través de intranets o de Internet utilizando protocolos de transporte comunes como HTTP o TCP.

1.5.3.1.3 Patrones de mensajes

En WCF se puede definir varios patrones de respuesta, como por ejemplo la comunicación solicitud-respuesta unidireccional y dúplex.

1.6 BASES DE DATOS [15]

1.6.1 INTRODUCIÓN

Las bases de datos almacenan un conjunto de datos que pertenecen a un mismo contexto de forma organizada, para que luego la información pueda ser utilizada y recuperada fácilmente.

1.6.2 SISTEMAS DE GESTIÓN DE BASE DE DATOS

Un sistema de gestión de base de datos está compuesto por un conjunto de datos interconectados y de los programas para acceder a dichos datos. El objetivo principal de estos sistemas de gestión es proporcionar una forma de almacenar y recuperar los datos de una manera que sea eficiente y práctica. Si estos sistemas van a ser compartidos por varios usuarios, el sistema debe evitar resultados anómalos.

1.6.2.1 SQL Server

Es un Sistema de Gestión de base de datos desarrollado por Microsoft, que es utilizado desde ordenadores o portátiles hasta en servidores corporativos. SQL server proporciona servicios de réplica entre varias copias del mismo, así como también con otros sistemas de base de datos. Además, SQL server proporciona una gran cantidad de herramientas graficas que guían a los administradores de base de datos en tareas como: establecer copias de seguridad regulares, réplicas de datos entre servidores y ajuste en el rendimiento de una base de datos.

1.6.3 MODELO DE DATOS

Un modelo de datos representa la estructura de la base de datos. El modelo de datos permite describir la estructura de los datos: las relaciones, la semántica y las restricciones de consistencia. Antes de elaborar un sistema que implique el trabajar con bases de datos, es conveniente primero, elaborar un adecuado modelo de datos.

1.6.3.1 Modelo entidad-relación

El modelo entidad relación representa una percepción del mundo real. En este modelo se identifican las entidades, que son una representación del mundo real, y la relación que existen entre estos objetos. El modelo entidad-relación es utilizado en el proceso de diseño de bases de datos.

Las entidades son descritas en una base de datos por los atributos que describen al objeto mencionado. Las relaciones son las asociaciones que existen entre las entidades.

La estructura de una base de datos, se puede expresar gráficamente mediante un diagrama E-R (Entidad-Relación), el cual tiene los siguientes elementos:

- *Rectángulos:* Representan a las entidades.
- *Elipses:* Representan los atributos de las entidades.
- *Rombos:* Representan las relaciones entre las entidades.

La **Figura 1.20** muestra un ejemplo de un diagrama E-R.

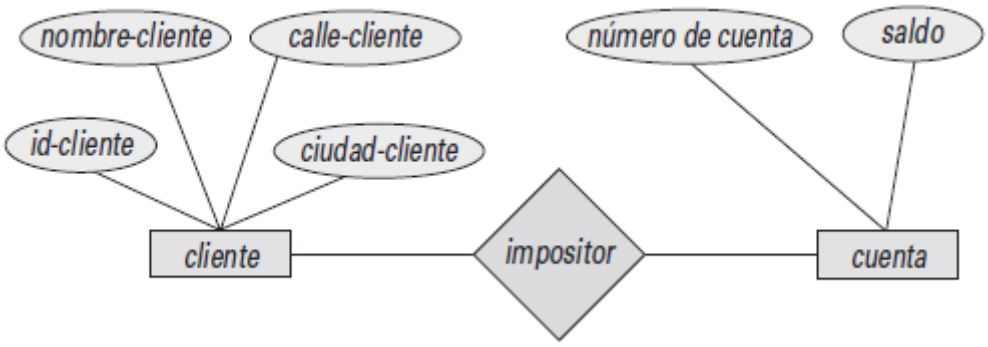

Figura 1.20 Ejemplo de diagrama E-R [15]

1.6.3.2 Modelo relacional

El modelo relacional representa los datos a través de tablas. Una tabla está compuesta por varias columnas que representan los atributos de la entidad que se está representando. La **Figura 1.21** muestra un ejemplo de base de datos relacional que consiste en tres tablas: la primera tabla muestra los clientes de un banco, la segunda las cuentas, y la tercera la relación entre las cuentas y los clientes.

id-cliente	*nombre-cliente*	*calle-cliente*	*ciudad-cliente*
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
18.273.609	Abril	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Ramblas	León
01.928.374	Gómez	Carretas	Cerceda

(a) La tabla *cliente*

número-cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

(b) La tabla *cuenta*

id-cliente	*número-cuenta*
19.283.746	C-101
19.283.746	C-201
01.928.374	C-215
67.789.901	C-102
18.273.609	C-305
32.112.312	C-217
33.666.999	C-222
01.928.374	C-201

(b) La tabla *impositor*

Figura 1.21 Ejemplo de base de datos relacional [15]

El modelo relacional representa un nivel de abstracción inferior al modelo de datos E-R. Usualmente, los modelos de datos primero se elaboran en E-R, y luego se traducen al modelo relacional.

1.6.4 TRANSACCIONES

Una transacción es un conjunto de operaciones que accede o modifica datos almacenados en una base de datos. Una transacción se inicia con la ejecución de una aplicación de usuario, está aplicación define instrucciones que están delimitadas por un inicio y un fin de la transacción. Para asegurar la integridad

de los datos, el sistema de base de datos debe mantener las siguientes propiedades de las transacciones:

- *Atomicidad:* En una transacción se deben realizar todas las operaciones definidas por la transacción o ninguna de ellas.
- *Consistencia:* Los datos que se obtienen después de una transacción, siempre son los esperados.
- *Aislamiento:* Consiste en garantizar que una sola transacción se ejecute al mismo tiempo en el sistema.
- *Durabilidad:* Una vez concluida una transacción con éxito, los cambios realizados en la base permanecen en el tiempo, incluso si llegaran a existir fallos en el sistema.

1.6.5 STRUCTURED QUERY LANGUAGE (SQL)

Es un lenguaje de consulta cómodo para el usuario, que surge de la combinación del algebra relacional y construcciones del cálculo relacional. Además de su lenguaje de consultas SQL incluye características para definir la estructura de los datos, para modificaciones de la base de datos y para la especificación de restricciones de seguridad.

1.6.5.1 Vistas

Una vista representa una tabla de datos virtual, formada de la información recuperada de una o varias tablas. Una vista tiene la misma estructura de una tabla, con la diferencia de que en una vista no se almacenan los datos sino la definición de la misma.

La elaboración de una vista representa la elaboración de una consulta, la cual se va a ejecutar cada vez que se llame a la vista.

1.6.5.2 Procedimientos almacenados

Un procedimiento almacenado representa un programa presente en una base de datos, el cual se ejecuta directamente en la base de datos cada vez que se

ejecuta dicho procedimiento almacenado. Por medio del lenguaje de módulos que posee SQL se pueden definir los procedimientos almacenados.

La elaboración de procedimientos almacenados permite el ahorro de recursos en el resto de componentes del sistema, ya que las operaciones se realizan directamente en el sistema de bases de datos.

1.6.5.3 Triggers

Los *triggers* representan procedimientos almacenados que se ejecutan al cumplirse una condición determinada dentro del sistema de la base de datos, tales como: inserción, actualización o eliminación de registros. Los *triggers* ayudan a mantener la consistencia de los datos dentro de la base datos.

1.7 METODOLOGIAS DE DESARROLLO DE SOFTWARE

1.7.1 DEFINICIÓN [16]

Una Metodología es un conjunto de reglas, técnicas, pasos y herramientas documentados que ayudan a los desarrolladores a realizar e implementar un nuevo software.

Cada una de estas técnicas indica cómo se debe realizar una actividad determinada. Por medio del uso de modelos, representaciones gráficas, y el empleo de procedimientos bien definidos, se debe tomar en cuenta que una de estas técnicas puede ser usada en varias metodologías de desarrollo.

Las metodologías de desarrollo ayudan a planificar, desarrollar y optimizar el software, y a definir qué hacer, como hacer y cuando hacer las actividades durante el desarrollo del proyecto.

1.7.2 CICLO DE VIDA [16]

Las metodologías siguen ciclos de vida, el ciclo de vida representa al conjunto de etapas o fases desde que nace la idea inicial hasta que el software es retirado

o reemplazado, en los ciclos de vida también se definen los criterios de transición para pasar de una fase a otra.

1.7.3 FASES DEL DESARROLLO DE SOFTWARE [17]

Existen varias metodologías para el desarrollo de software pero en general la mayoría de las metodologías siguen las siguientes fases.

- *Análisis*: En esta fase se especifica el comportamiento y el funcionamiento de los programas y la interfaz con el resto de elementos del sistema.
- *Diseño*: En esta fase se crea un algoritmo para resolver el problema o cumplir la meta que se quiere obtener con este software.
- *Codificación*: En esta fase se escribe el código en un lenguaje de alto nivel y se obtiene el código fuente para poder compilarlo.
- *Pruebas*: En esta fase se ejecuta la aplicación, se prueba su correcto funcionamiento rigurosamente, y se eliminan todos los errores que puedan aparecer.
- *Implementación*: En esta fase se instala la aplicación, se orienta a los usuarios sobre el uso de esta y se cargan los datos necesarios para su funcionamiento.
- *Mantenimiento*: En esta fase se optimiza y mejora el software después de haber sido entregado al usuario final y también se corrige y se previenen algunos defectos.

1.7.4 METODOLOGIAS ÁGILES [18]

Este tipo de metodología esta principalmente orientada hacia proyectos pequeños debido a que aporta una elevada simplificación sin perder un elevado estándar de calidad del producto. En febrero de 2001, en las montañas Wasatch de Utah se establecieron los cuatro principios sobre los cuales se basan las metodologías agiles y estos son:

- “Se valora a los individuos y las interacciones sobre los procesos y las herramientas.
- Se valora a las aplicaciones que funcionan sobre la documentación exhaustiva.
- Se valora la colaboración del cliente sobre las negociaciones contractuales.

- Se valora la respuesta al cambio sobre el seguimiento de un plan". [19]

En base a estos principios se establecen algunas de las principales características de las metodologías agiles como: satisfacer al cliente con entregas continuas y tempranas es una prioridad, el cambio de requerimientos resulta beneficioso para el proyecto, los clientes y desarrollares deben trabajar juntos diariamente en el desarrollo del proyecto, la manera más efectiva de transmitir información es la conversación frontal, se busca la simplicidad con el fin de evitar realizar trabajo innecesario, cada cierto tiempo se busca como ser más efectivos, ajustando los comportamientos para lograr este fin y tener un software que funciona es una medida de progreso.

1.7.5 DOCUMENTACIÓN [17]

La Documentación al momento de desarrollar software es de vital importancia ya que permite facilitar la utilización de la aplicación por el usuario final, conservar un historial del desarrollo del software y disminuir el costo de operación y de ejecución del proyecto.

La documentación de un proyecto de software principalmente incluye: análisis de requerimientos, algoritmos obtenidos en el diseño, códigos comentados, resultados de las pruebas, manuales de uso, entre otros.

1.7.6 PROGRAMACIÓN EXTREMA (XP)

1.7.6.1 Introducción [18]

Es una metodología ágil de desarrollo que está diseñada para entregar el software que los clientes necesitan en el momento en que lo necesitan y se basa esencialmente en la agilidad y simplicidad, con el fin de buscar un punto medio entre el abuso de procesos y la ausencia de los mismos, usando únicamente los procesos que realmente valen la pena [18].

Las metodologías ágiles como XP lo que buscan es adaptarse al cambio, este tipo de metodologías son enfocadas hacia la gente, es decir el fin de los procesos que se usan es el de ayudar al equipo de desarrollo.

XP está enfocado hacia el trabajo en equipo, tanto gerentes, propietarios, clientes, desarrolladores son parte de un equipo que tienen como fin desarrollar un software de calidad.

1.7.6.2 Ciclo de vida de Software en XP [18]

En XP se plantea un ciclo de vida dinámico que incluya lo que el cliente necesita, estimar el esfuerzo, crear la solución y entregar el producto final al cliente, a pesar de que en muchos de los casos, los clientes no son capaces de establecer todos los requerimientos al principio del proyecto.

Por su forma de trabajo en XP se establecen ciclos cortos denominados iteraciones que tienen entregables funcionales al finalizar cada ciclo. En cada una de estas iteraciones se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas propias de la metodología.

Típicamente un proyecto en XP está compuesto de varias iteraciones o ciclos. La **Figura 1.22** representa un ciclo de vida en XP.

A pesar de que el ciclo de vida de XP es muy dinámico, a este ciclo de vida se lo puede separar en fases y estas son:

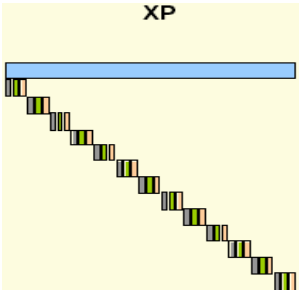

Figura 1.22 Ciclo de Vida XP [18]

- *Fase de Exploración:* En esta fase, se establece el alcance general del proyecto por medio de la redacción de las historias de usuarios. Las estimaciones realizadas en esta fase pueden variar cuando se analicen más a detalle cada una de las iteraciones.

- *Fase de Planificación:* En esta fase, el grupo de desarrolladores y el cliente acuerdan el orden en el que se deben implementar las historias de usuario y asocian una fecha de entrega a estas.
- *Fase de Iteraciones:* En esta fase, las funcionalidades son desarrolladas y al final de esta se tiene entregables de las iteraciones que implementan las historias de usuario, Cada vez que se llega a un entregable de las iteraciones sin errores se tiene una medida del avance del proyecto.
- *Fase de puesta en Producción:* Esta fase se da cuando ya se tiene la funcionalidad completa del proyecto y generalmente lo único que se realiza es un pequeño ajuste ya que en la fase anterior ya se entregaron los módulos funcionales y sin errores de las iteraciones.

1.7.6.3 Reglas y Practicas [18]

1.7.6.3.1 Planificación [18]

La planificación plantea un dialogo continuo entre las partes del proyecto. El proyecto comienza por medio de la recopilación de historias de usuario, una vez obtenidas estas, los desarrolladores evalúan el tiempo de desarrollo de cada historia y si alguna de estas presenta riesgos se realizan "spikes"[56], posteriormente se reúnen las partes del proyecto para establecer un cronograma de entregas en el que todos estén de acuerdo. Con el cronograma listo se comienza la fase de iteraciones en donde en cada iteración se desarrollan y prueban unas pocas historias de usuario.

Dentro de los conceptos básicos de la planificación se tienen los siguientes:

- *Historias de Usuarios:* Estas historias son escritas por el cliente como descripciones cortas de lo que el sistema debe realizar. Las historias de usuario deben tener un nivel de detalle mínimo para que los programadores puedan hacer una estimación poco riesgosa del tiempo en que se llevará a cabo cada historia.
- *Plan de Entregas:* Aquí se establece que historias de usuario van a ser agrupadas para conformar una entrega por medio de una reunión entre

[56] **Skypes:** Pequeños programas de prueba para reducir riesgos. [15]

cliente y desarrolladores. Generalmente los clientes agrupan las historias según sus prioridades y el cronograma de entrega se realizará en base a las estimaciones de tiempo realizadas por los desarrolladores. Luego de haber concluido algunas iteraciones se puede evaluar el plan de entregas con el fin de realizar algún ajuste.

- *Plan de Iteraciones:* Al comienzo de cada ciclo se realiza una reunión de planificación de iteración en esta reunión a cada historia se la traduce a una tarea específica y así mismo para cada una de las historias se establece una prueba de aceptación. Estas pruebas de aceptación se realizan al finalizar cada ciclo y también se las lleva a cabo en los ciclos siguientes con el fin de prevenir errores.
- *Reuniones Diarias de Seguimiento:* El objetivo de las reuniones diarias es compartir problemas y soluciones. Por lo general en estas reuniones algunos de los participantes se limitan a escuchar para no quitar tiempo innecesario al equipo.

1.7.6.3.2 Diseño [18]

La Metodología XP hace énfasis en que el diseño de los sistemas simples y claros, por lo que se utilizan los siguientes conceptos al momento del diseño.

- *Simplicidad:* En XP se trata de implementar el diseño más simple que funcione, también se establece que se debe evitar adelantar la implementación de funcionalidades de iteraciones futuras.
- *Recodificación:* Consiste en escribir una pequeña parte del código sin cambiar la funcionalidad del programa con el fin de obtener un código más simple, claro y entendible.
- *Metáforas:* En XP se sugiere usar Metáforas para explicar los propósitos del proyecto, ya que estas son claras y todos las pueden entender. También en esta metodología, en la nomenclatura de los métodos, se utilizan nombres claros que no requieran mayores explicaciones.

1.7.6.3.3 Pruebas [18]

- *Pruebas Unitarias:* Estas pruebas son de vital importancia en XP, ya que todos los módulos tienen que pasar las pruebas unitarias para ser liberados. El hecho de que los componentes pasen las pruebas unitarias hace que el código pueda funcionar de manera colectiva.
- *Detección y Corrección de Errores:* En XP, cuando se presenta un error, debe ser corregido de forma inmediata y se debe prevenir que errores de este tipo se vuelvan a presentar en el futuro. Se pueden generar nuevas pruebas con el fin de controlar que los errores se hayan resuelto.
- *Pruebas de Aceptación:* Estas pruebas son creadas en base a las historias de usuario de cada iteración. El cliente tiene que establecer bajo que escenarios comprobar que las historias han sido correctamente implementadas.

CAPÍTULO 2. DISEÑO DEL SISTEMA

2.1 INTRODUCCIÓN

En el presente capítulo se presenta el análisis de requerimientos del sistema prototipo, a partir de las aplicaciones NFC iShop y Mercado Libre que tienen funcionalidades similares a las del sistema prototipo propuesto.

Para obtener los requerimientos del sistema prototipo se describe el funcionamiento y características de ciertas aplicaciones móviles, similares al sistema prototipo planteado. Luego se realiza una comparación entre las aplicaciones propuestas en el análisis, y en base a los puntos anteriores se establecen los requerimientos mínimos del sistema.

Finalmente en base a los requerimientos mínimos y a otros requerimientos que se presenten, se realizan los diseños de cada uno de los componentes que conforman el sistema y se indica la función que cumple dentro del mismo.

2.2 ANÁLISIS DE APLICACIONES MÓVILES DISPONIBLES EN EL MERCADO

2.2.1 NFC ISHOP [20]

2.2.1.1 Introducción

NFC iShop es una aplicación desarrollada por la Universidad Politécnica de Singapur con fines académicos. Esta aplicación pretende demostrar el uso innovador de las comunicaciones NFC para crear una aplicación de compras.

Esta aplicación fue desarrollada para una tienda de abarrotes, los usuarios de esta aplicación pueden agregar productos al carro de compras, acercando el móvil NFC al tag NFC. Si la aplicación es personalizada, esta le permite al usuario recibir alertas cuando el usuario agrega productos que contengan

ingredientes de los cuales puede ser alérgico; además le permite al usuario llevar un historial de compras realizadas anteriormente.

Debido a que esta aplicación fue lanzada solamente con fines académicos no brinda soporte para el montaje y puesta en funcionamiento del sistema. Esta aplicación fue realizada para ser mostrada en una feria desarrollada en Singapur y no indica para que tipo de tags funciona ni cuál es la información que se debe ingresar en el tag para el funcionamiento de la aplicación. Es por eso que, para el análisis de esta aplicación únicamente se usa la información disponible en Google Store y la ayuda de la aplicación.

En las siguientes secciones se explica cada una de las actividades que conforman la aplicación, cual es la función que cumplen, entre otros.

2.2.1.2 Actividades de la aplicación

2.2.1.1.1 Actividad Introductoria

Es la primera actividad que aparece al abrir la aplicación, no tiene ninguna funcionalidad aparte de mostrar una imagen que muestra el logotipo de la aplicación. En la **Figura 2.1** se puede apreciar una captura de pantalla de esta actividad.

Figura 2.1 Actividad Introductoria [20]

2.2.1.1.2 Menú Principal

Esta actividad muestra las principales actividades que se pueden abrir en la aplicación. La forma de mostrar los ítems disponibles en el menú es a través de `ImageButtons`[57]*.* Los ítems de este menú son: *Shopping Cart, Profile, Payment History, Settings y About.* En la **Figura 2.2** se puede apreciar una captura de pantalla de esta actividad.

Figura 2.2 Menú Principal [20]

2.2.1.1.3 Shopping Cart

Esta actividad muestra la lista del carro de compras. En cada producto se detalla: su imagen, su nombre, una breve descripción del producto y el precio. Por ultimo muestra un `NumberPicker`[58] para seleccionar la cantidad de productos a adquirir. Al final de la lista se muestran dos `TextView`[59] que muestran la cantidad de ítems en el carro de compras y el valor subtotal.

En el `OptionsMenu`[60] de la actividad hay la opción *Clear Shopping Cart* que sirve para vaciar la lista del carro de compras. Finalmente en la parte inferior de

[57] **ImageButtons:** Control de Android para colocar en la actividad un botón con imagen.
[58] **NumberPicker:** Control de Android para seleccionar un número específico.
[59] **TextView:** Control de Android para mostrar texto.
[60] **OptionsMenu:** Control de Android que hace uso del botón Menú del móvil para mostrar otras opciones que no se pueden mostrar en la actividad.

la actividad existe un botón *Checkout*, que sirve para finalizar la selección de productos y pasar al proceso de pago. En la **Figura 2.3** se puede apreciar una captura de pantalla de esta actividad.

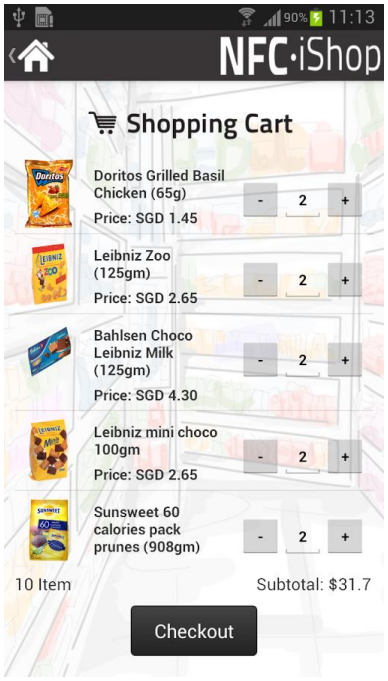

Figura 2.3 Actividad *Shopping Cart* [20]

2.2.1.1.4 Información de Producto

Esta actividad despliega información del producto seleccionado. Al momento de tocar el tag NFC con el móvil NFC, esta ventana se despliega. La información que se muestra del el producto es: su nombre, una foto del producto, información del producto, calificaciones y comentarios de los usuarios al producto, por ultimo una lista de productos similares. En la **Figura 2.4** se puede apreciar una captura de pantalla de la actividad.

2.2.1.1.5 Checkout

Una vez que se oprime el botón *checkout* en la actividad *Shopping Cart,* se despliega la actividad *Checkout.* En esta actividad se muestra un formulario en el cual el usuario debe llenar la siguiente información:

- Nombre
- Tipo de tarjeta

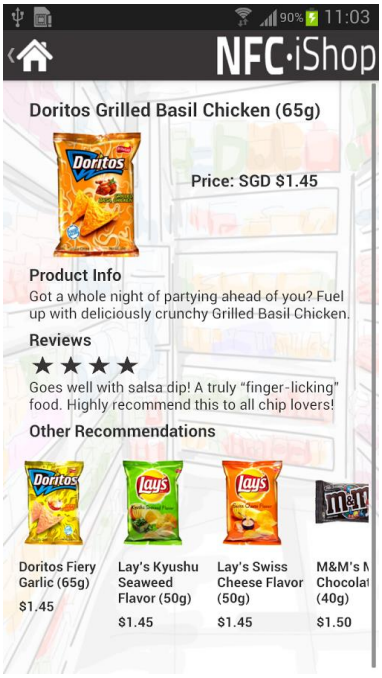

Figura 2.4 Actividad Producto [20]

- Número de tarjeta
- Código CSC (Código de seguridad de la tarjeta)
- Fecha de expiración de la tarjeta
- Fecha y hora de entrega del producto
- Dirección de envío

En la **Figura 2.5** se puede apreciar una captura de pantalla de esta actividad.

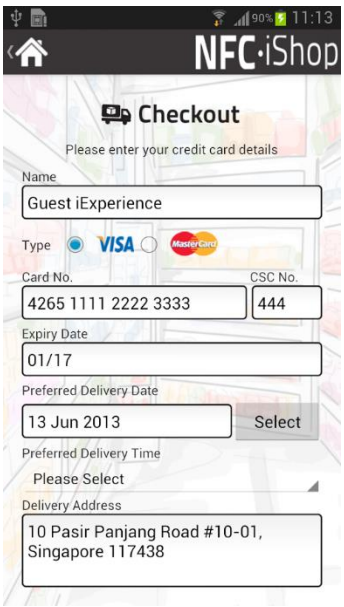

Figura 2.5 Actividad Checkout [20]

Una vez que el usuario termina de ingresar los datos, se despliega otra actividad con un resumen de las compras realizadas.

2.2.1.1.5 Profile

Esta actividad permite modificar los datos del perfil del usuario. Básicamente es un formulario que permite llenar los siguientes datos:

- Dirección de correo electrónico
- Nombre
- Apellido
- Número de celular
- Fecha de nacimiento
- Género
- Alergias
- Dirección de entrega de productos

La parte interesante de este formulario es que permite al usuario ingresar todos los ingredientes que le producen alergias. Con esta información, la aplicación emite alertas cuando el usuario agrega a la lista de compras productos que contengan esos ingredientes. En la **Figura 2.6** se puede apreciar una captura de pantalla de esta actividad.

Figura 2.6 Actividad *Profile* [20]

2.2.1.1.6 Payment History

Esta actividad muestra en una lista las transacciones realizadas en el pasado. De esta actividad no se encontró mayor información sobre su funcionamiento. En la **Figura 2.7** se puede apreciar una captura de pantalla de esta actividad.

Figura 2.7 Actividad *Payment History* [20]

2.2.1.1.7 Settings

Esta actividad permite configurar dos opciones las cuales son: añadir productos automáticamente al carro y configurar un perfil por defecto. La opción añadir productos automáticamente al carro habilita la posibilidad de añadir los productos directamente al carro, sin pasar por la actividad informativa que se despliega al momento de interactuar con el tag NFC. La opción configurar perfil por defecto permite cargar los datos del usuario automáticamente a la actividad de *Checkout.* En la **Figura 2.8** se puede apreciar una captura de pantalla de esta actividad.

2.2.1.1.8 About

Esta actividad muestra información sobre la organización que elaboró la aplicación e información de contacto. En la **Figura 2.9** se puede apreciar una captura de pantalla de esta actividad.

Figura 2.8 Actividad *Settings* [20]

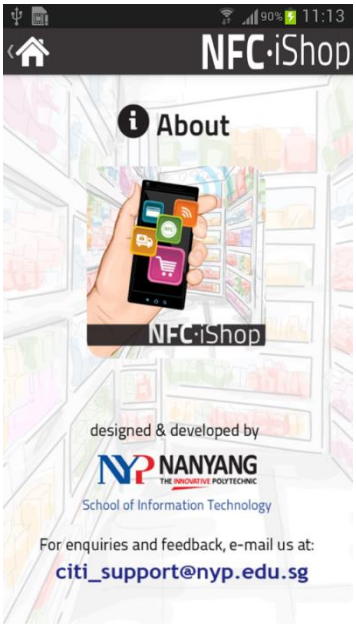

Figura 2.9 Actividad *About* [20]

2.2.2 MERCADO LIBRE [20] [21]

2.2.2.1 Introducción

Mercado Libre es una empresa que sirve de intermediario en la compra y venta entre usuarios inscritos en el sistema. Tiene operaciones en gran parte de Sudamérica y América Central. En este sistema de compra y venta, Mercado Libre no se hace responsable de las transacciones realizadas por sus clientes, sin embargo su sistema califica a compradores y vendedores para que el resto de usuarios conozca la reputación de cada comprador o vendedor.

En esta sección se pretende explicar el funcionamiento de la aplicación móvil de este sistema, para poder extraer requerimientos que sean útiles para el sistema prototipo planteado. Vale aclarar que este sistema de Mercado Libre tiene algunos módulos que no están asociados a la temática del presente proyecto por lo que solo se analizan las actividades asociadas a la temática del proyecto.

2.2.2.2 Principales Actividades de la Aplicación

2.2.2.2.1 Actividad Registrarte

Para poder realizar compras y ventas en el sistema de Mercado Libre, el usuario debe registrarse mediante un formulario. Este formulario pide la siguiente información: nombre, apellido, correo electrónico, clave del usuario, número de teléfono. En la actividad de registro, se puede observar que, los controles `EditText`[61] dan una pista al usuario de la forma en la que se debe llenar la información. El control de Android `EditText` ofrece esta posibilidad, cuando el usuario hace clic sobre el control, el texto pista desaparece para que el usuario llene el control. La **Figura 2.10** muestra una captura de pantalla de la actividad.

[61] **EditText:** Control de Android que permite al usuario ingresar cadenas de texto.

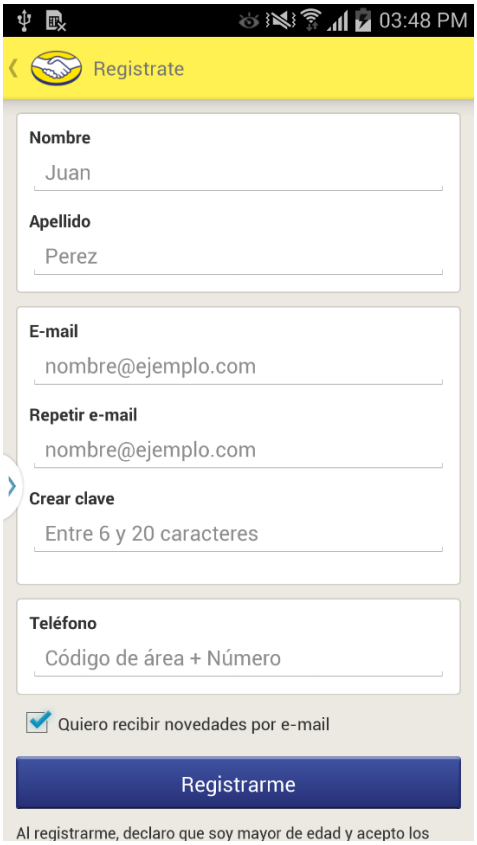

Figura 2.10 Actividad Registrarse [20]

2.2.2.2.2 Menú Principal

En esta aplicación el menú principal no se maneja como otra actividad sino como un control personalizado llamado `SideViewMenu`[62]. Entre las principales opciones del control están: Buscar, Categorías, Favoritos, Vender, Configuración. Cada una de estas opciones despliega una nueva actividad al momento de seleccionarlas. La **Figura 2.11** muestra una captura de pantalla de este control personalizado.

2.2.2.2.3 Ingresar

Para poder usar toda la funcionalidad de la aplicación, el usuario debe autenticarse. En caso de que el usuario de la aplicación no se autentique, solamente podrá realizar búsquedas, mas no compras y ventas.

[62] **SideViewMenu:** Control personalizado de Android que permite al usuario deslizar el menú desde la parte izquierda de la pantalla del móvil desde cualquier actividad.

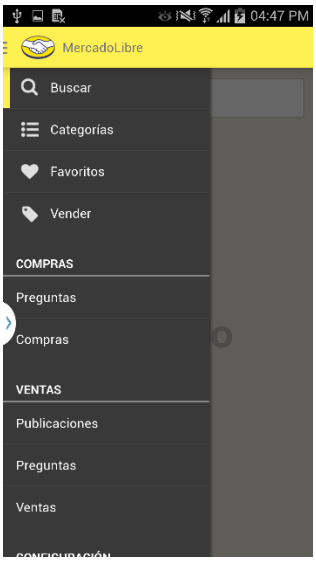

Figura 2.11 Menú Principal [20]

Para ingresar, la aplicación no hace uso de una actividad, sino de un `ContextMenu`[63]. La **Figura 2.12** muestra el control para la autenticación.

Figura 2.12 Control para la Autenticación [20]

[63] **ContextMenu:** Control de Android que se muestra sobre una actividad, usualmente usado como parte de una confirmación o cuando no entran ciertos controles en el espacio de la actividad.

2.2.2.2.4 Actividad Buscar y Categorías

Estas actividades permiten buscar productos que el usuario puede estar interesado en adquirir. La actividad de búsqueda consiste en un `EditText` en el cual el usuario puede ingresar palabras clave para buscar el producto que está interesado en adquirir. La actividad también muestra un historial de búsquedas anteriores. La **Figura 2.13** muestra una captura de pantalla de esta actividad.

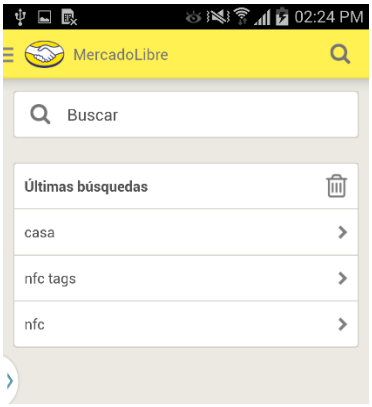

Figura 2.13 Búsqueda [20]

En la actividad categorías se agrupan a los productos, para que el usuario pueda realizar una búsqueda más detallada. Este tipo de búsqueda esta puesta en la aplicación para cuando el usuario no tiene claro el producto que desea adquirir. En la **Figura 2.14** se puede observar una captura de pantalla de esta actividad.

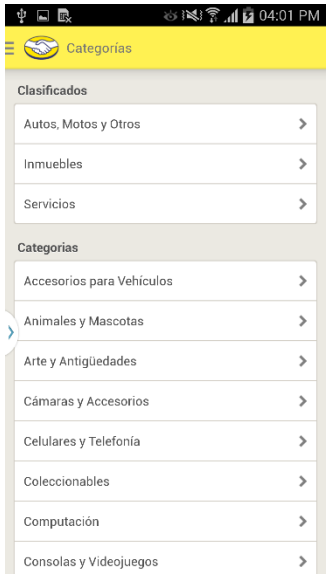

Figura 2.14 Categorías [20]

Una vez que se concluye la búsqueda, la actividad despliega una lista de productos que le pueden interesar al usuario. Para mostrar esta lista, la aplicación hace uso de un `ListView`[64]. Los elementos que se muestran en el `ListView` son: imagen del producto, nombre del producto, número de artículos vendidos y el precio. La **Figura 2.15** muestra la distribución del `ListView`.

Figura 2.15 ListView de Productos [20]

En este punto la actividad le permite al usuario realizar filtros personalizados de la búsqueda y ordenar los productos en base a parámetros como: precio, condición entre otros. La **Figura 2.16** muestra las opciones del filtro.

Si el usuario selecciona un producto del `ListView` se despliega otra actividad con información detallada del producto, descripción del vendedor del producto. Esta actividad muestra opciones para realizar preguntas, conocer la reputación del vendedor, ver productos parecidos al seleccionado y finalmente realizar la compra del producto. Como este sistema no recibe directamente los pagos sino

[64] **ListView:** Control de Android que permite visualizar elementos en forma de lista. Cada elemento de la lista brinda la posibilidad de: abrir nuevas actividades, ejecutar alguna acción, entre otras.

solo actúa como intermediario de la compra-venta, la aplicación, lo único que hace es poner en contacto al comprador y al vendedor. Finalmente, luego de realizado la transacción por distintos medios que el sistema no controla, solicita a las partes que califiquen a la otra parte para establecer la reputación de cada usuario. La **Figura 2.17** muestra una captura de pantalla de la actividad mencionada.

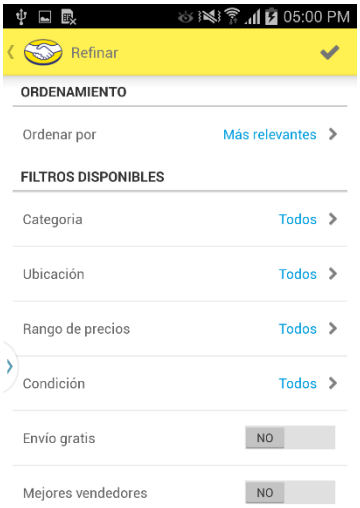

Figura 2.16 Filtro Personalizado [20]

Figura 2.17 Diseño de la Actividad Artículo [20]

2.2.3 COMPARACIÓN DE APLICACIONES

En esta sección se realiza una tabla comparativa con las aplicaciones analizadas, con el fin de determinar los requerimientos mínimos que el sistema prototipo planteado necesita, esta información se presenta en la **Tabla 2.1**.

Servicio de la Aplicación	NFC iShop	Mercado Libre
Autenticación	No autentica a los clientes. Al momento de pago se deben ingresar los datos para realizar el pago.	La aplicación brinda un servicio de autenticación para que la aplicación sea utilizada bajo un usuario específico. Si el usuario no se autentica no podrá disponer de todos los servicios de la aplicación, solo podrá realizar búsquedas.
Menú principal	Manejado como una actividad más de la aplicación y haciendo uso de `ImageButtons`	Manejado como un control personalizado denominado `Side ViewMenu`, este menú siempre podrá ser accesible desde cualquier actividad.
Carro de compras	Tiene una actividad específica para ir acumulando los productos y luego pagar en una sola cuenta todos los productos seleccionados.	No dispone de este servicio.
Selección de un producto	Acercando el móvil NFC a un tag NFC.	Utilizando las actividades de Buscar y Categorías.
Información del Artículo	Uso de una actividad dedicada para mostrar la información del artículo. La información que se muestra es: nombre, imagen, precio, información del artículo, calificaciones dadas por los usuarios sobre el artículo y recomendaciones al usuario para revisar artículos similares.	Uso de una actividad dedicada para mostrar la información del artículo. La información que muestra es: nombre, método de envió, medios de pago, reputación del vendedor, descripción del artículo, preguntas al vendedor, otras publicaciones del vendedor.
Pagos de las compras	Uso de la Actividad `Checkout` para pagar las compras mediante una tarjeta de crédito.	No brinda el servicio, una vez que el usuario realiza la compra, el sistema pone en contacto al comprador y al vendedor.
Modificación de los datos del usuario	Permitido en la aplicación mediante una actividad.	No permitido en la aplicación, solo hay la posibilidad de ver los datos del usuario.

Tabla 2.1 Cuadro comparativo de las aplicaciones NFC iShop y Mercado Libre

2.3 ANÁLISIS DE REQUERIMIENTOS DEL SISTEMA

Una vez realizado el análisis de aplicaciones similares al sistema prototipo propuesto, se puede tener una idea clara de cuáles son los requerimientos que debe tener la aplicación móvil y el sistema en general. Para el desarrollo del sistema prototipo se sigue la metodología de desarrollo XP, combinando con herramientas de otras metodologías que resulten útiles.

El sistema prototipo está compuesto por cuatro componentes principales: una aplicación de escritorio que permita manejar la gestión de inventario de productos y el sistema de facturación, una aplicación Android que a partir de este momento se denominará aplicación móvil para implementar el carro de compras virtual, el servidor que aloja el Servicio Web para implementar la lógica de negocio y en el mismo servidor, la base de datos para almacenar la información del sistema. En la **Figura 2.18** se puede observar el esquema general del sistema prototipo.

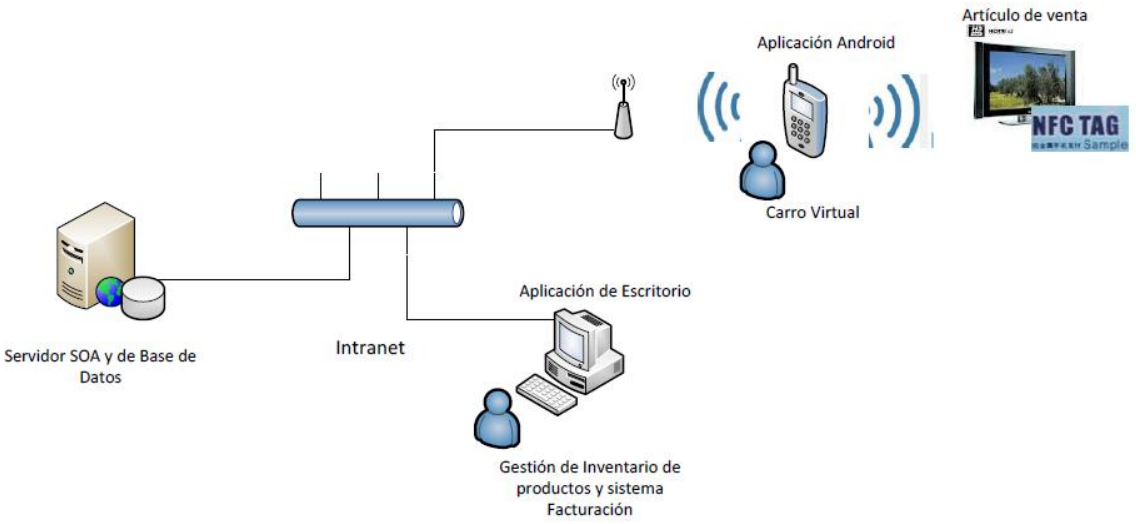

Figura 2.18 Esquema general del sistema prototipo

La principal función de la aplicación móvil es permitir al usuario seleccionar los productos que desea comprar y agregarlos a un carro de compras virtual. La aplicación móvil debe tener al menos dos perfiles para diferenciar usuarios comunes de usuarios VIP y así poder dar descuentos especiales a los usuarios

VIP. Adicionalmente la aplicación móvil debe permitir registrarse en el sistema para no tener que acudir a usuarios administrativos para esto.

La aplicación de escritorio tiene dos funcionalidades importantes: la primera permitir la facturación y la segunda manejar el inventario del sistema. La aplicación debe tener dos perfiles de usuarios. El primero, un usuario Administrador que debe tener control total de la aplicación de escritorio y no debe tener restricciones de acceso. El segundo, un usuario Cajero que debe tener únicamente acceso a las opciones relativas a la facturación, las opciones relativas a la administración del sistema deben estar restringidas.

Ambas aplicaciones deben permitir modificar los datos de la cuenta del usuario y cambiar la contraseña.

2.3.1 REGISTRO DE NUEVOS USUARIOS

Para simplificar los procesos de registros de nuevos usuarios, la aplicación móvil debe contar con una actividad que permita ingresar la información que se necesite para crear una nueva cuenta.

2.3.2 AUTENTICACIÓN

El sistema prototipo propuesto al manejar perfiles y usuarios, debe tener algún método de autenticación, para que se puedan manejar sesiones en el servidor y atender los requerimientos del usuario de manera individualizada.

2.3.3 MODIFICACIÓN DE LOS DATOS DEL USUARIO

La aplicación debe permitir al usuario modificar los datos de su cuenta, con excepción de la cédula, con el motivo de evitar que se ingresen cédulas duplicadas, este campo lo podrá modificar un administrador del sistema directamente en la base de datos.

2.3.4 CAMBIO DE CONTRASEÑA

Por motivos de seguridad al usuario deberá permitírsele cambiar su contraseña. Así se evita que personas no autorizadas accedan a las cuentas de los usuarios,

en caso de que las contraseñas de los usuarios sean obtenidas por terceras personas.

2.3.5 CARRO DE COMPRAS VIRTUAL

Es la funcionalidad más importante de la aplicación móvil, pues el sistema prototipo propuesto plantea el diseño de un carro de compras virtual. La aplicación NFC iShop proveyó una idea de cómo estructurar esta funcionalidad. Esta funcionalidad debe mostrar la lista de compras del usuario y debe irse actualizando dinámicamente a medida que el usuario seleccione los artículos.

En cada artículo de la lista de compras se debe mostrar individualmente la siguiente información: una imagen del producto, el nombre del producto, la marca, la descripción, la cantidad de ítems agregados al carro de compras y el valor que se debe cancelar por el o los productos incluido impuestos. Adicionalmente debe permitir al usuario modificar la cantidad de productos de un mismo artículo o la posibilidad de eliminar por completo el artículo de la lista.

Al final de la lista del carro de compras el usuario tiene a su disposición, el subtotal de la compra para tener referencia de cuánto debe cancelar hasta el momento por los productos agregados.

El usuario debe tener la opción de limpiar completamente el carro de compras para iniciar una nueva compra si así lo desea.

Por último, esta funcionalidad debe tener una opción que le permita al usuario finalizar la compra, para poder acercarse a una caja y cancelar los valores correspondientes.

2.3.5.1 Método de selección de artículos

El método de selección de artículos debe ser mediante la interacción de un tag NFC con un móvil NFC. El tag NFC reemplaza la acción de tomar los productos de la percha y llevarlos a una caja para cancelar su valor.

2.3.5.2 Información del artículo

La aplicación móvil deberá mostrar posteriormente a la selección del artículo, la siguiente información del artículo: nombre, imagen descriptiva del artículo, marca, modelo, descripción, precio con impuestos y descuentos aplicados en el artículo. Adicionalmente, se deberá permitir elegir el número de productos de ese artículo y finalmente permitirle al usuario tomar la decisión de si agrega o no los productos al carro de compras.

2.3.5.3 Pago de compras

Luego de que el usuario finaliza la compra, la aplicación móvil debe indicarle que se acerque a pagar a una caja y posteriormente le muestra un desglose a modo de nota de venta de los productos adquiridos.

2.3.6 ADMINISTRACIÓN DEL SISTEMA

Este componente del sistema se encuentra en la aplicación de escritorio, y le permite al usuario administrar el inventario de todo el sistema. La aplicación de escritorio debe tener formularios para administrar: usuarios, artículos y productos. De cada artículo, la aplicación debe permitir administrar: marcas y modelos. Se considera también importante administrar las formas de pago y los impuestos.

Para facilitar la búsqueda de registros específicos, cada formulario debe tener filtros de búsqueda en base a ciertos parámetros que se consideren necesarios, por ejemplo filtros por: cedula, nombre, ID, entre otros.

2.3.7 FACTURACIÓN

La aplicación de escritorio debe permitir realizar la gestión de los pagos de los usuarios que han finalizado las compras en la aplicación móvil. Debe existir un formulario con los registros de todos los usuarios que hayan finalizado la selección de productos en el carro de compras. Al igual que en los formularios anteriores, debe también contar con filtros de búsqueda, para encontrar registros fácilmente.

Cuando el cliente se acerque a una caja, debe proporcionar los datos necesarios al cajero, para que el mismo seleccione el registro que contiene las compras del cliente. Una vez seleccionado el registro, el cajero debe seleccionar el tipo de pago e ingresar los parámetros necesarios para emitir la factura. Además debe disponer de un mecanismo para concluir el proceso de facturación y emitir la factura. Adicionalmente se debe permitir tener acceso a un historial de facturas para reimprimirlas o anularlas, en caso de ser necesario.

2.3.8 AYUDA

Tanto la aplicación móvil como la aplicación de escritorio deben contar con un pequeño manual de usuario, que le permita al usuario conocer cómo utilizar de manera correcta todas las actividades o formularios de la aplicación.

2.3.9 ACERCA DE

Las aplicaciones deben contener información de los desarrolladores de la aplicación.

2.3.10 CERRAR SESIÓN

Las aplicaciones deben tener una opción para que el usuario cierre su sesión y le permita usar la aplicación a otro usuario registrado en el sistema.

2.3.11 DIAGRAMA DE CASOS DE USO DEL SISTEMA

En la presente sección se muestra los diagramas de casos de uso que se generaron a partir de los requerimientos anteriormente expuestos. La **Figura 2.19** muestra la parte del diagrama de casos de uso relativa a la aplicación móvil, la **Figura 2.20** y la **Figura 2.21** muestran las partes del diagrama de casos de casos de uso relativas a la aplicación de escritorio.

Figura 2.19 Diagrama de casos de uso del sistema (primera parte)

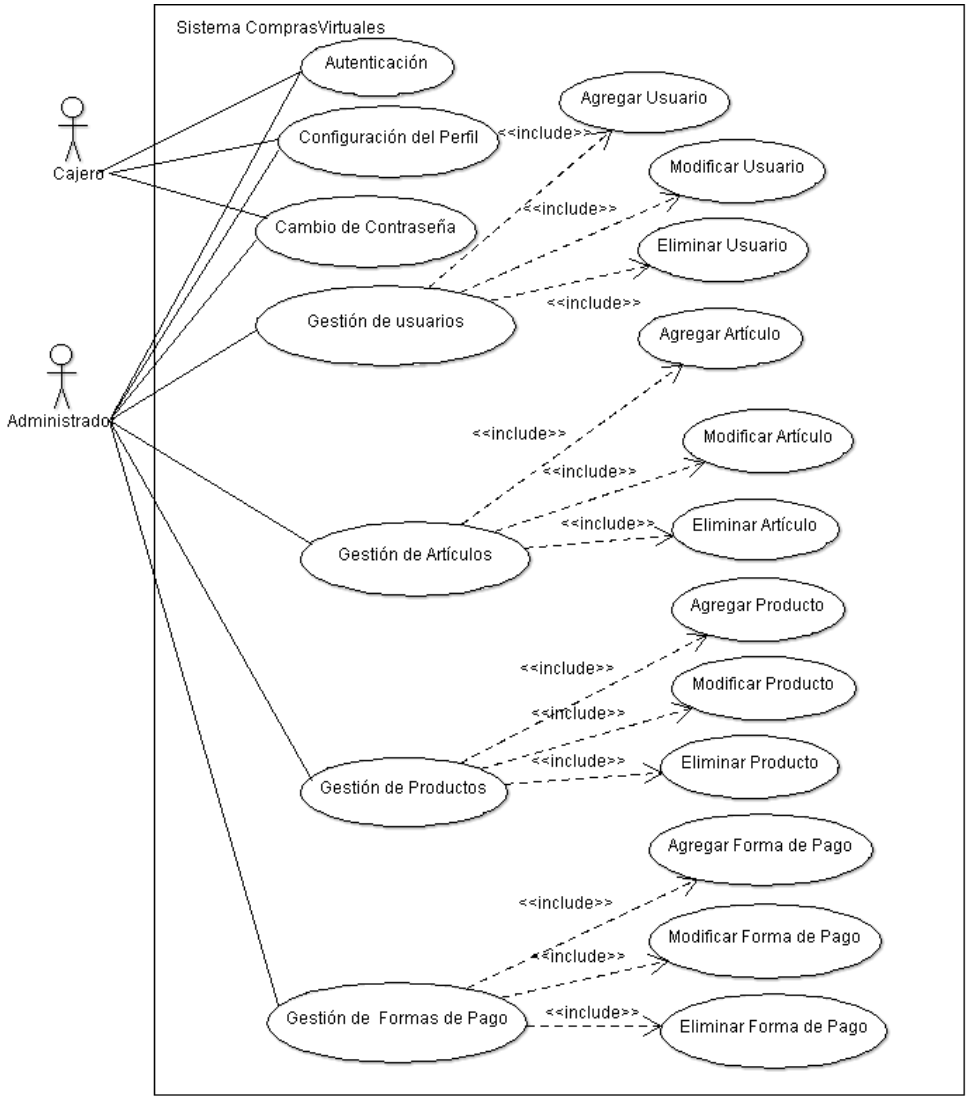

Figura 2.20 Diagrama de casos de uso del sistema (segunda parte)

Figura 2.21 Diagrama de casos de uso del sistema (tercera parte)

2.4 DISEÑO DEL SISTEMA

Una vez recopilada la información necesaria, y establecidos los requerimientos mínimos del sistema, en esta sección se realiza el diseño del sistema prototipo. Para ello se sigue la metodología de desarrollo XP, combinando con herramientas de otras metodologías que resulten útiles.

2.4.1 PLANIFICACIÓN DE ITERACIONES Y ENTREGAS

Lo primero que se debe realizar en la etapa de planificación es recopilar y organizar los requerimientos a través de la elaboración de Historias de Usuario.

2.4.1.1 Historias de Usuario

Las historias de usuario permiten recopilar y organizar los requerimientos del sistema, estas deben ser agrupadas por iteraciones y deben expresar de forma clara y sin lenguaje técnico las funcionalidades del sistema.

2.4.1.1.1 Elementos de una Historia de Usuario

El formato a llevar en cada historia de usuario se explica a continuación:

- *Número:* Código que identifica de forma única a cada historia de usuario. Está conformado por las siglas Historia de Usuario (HU), número de iteración, la letra M si la historia pertenece a la aplicación Móvil, E si la historia pertenece a la aplicación de Escritorio y ME si la historia aplica tanto a la aplicación móvil como a la aplicación de escritorio, adicionalmente las historias de usuario que implique la creación de métodos en el Servicio Web se asignan con la letra S y B si la historia corresponde a la base de datos, por último el número de historia en la iteración.
- *Usuario:* Indica el tipo de usuario al que le corresponde esta historia de usuario.
- *Nombre Historia:* Nombre descriptivo para identificar la necesidad.
- *Prioridad en Sistema:* Calificación que se le da a la historia en base a la importancia en la aplicación: Baja si la historia de usuario no es de mucha importancia para el funcionamiento del sistema, Media si la historia de

usuario representa alguna necesidad para el correcto funcionamiento del sistema y Alta si la historia de usuario es necesaria para el correcto funcionamiento del sistema.

- *Riesgo en desarrollo:* Es una estimación en cuanto a la complejidad de la historia de usuario y su solución.
- *Iteración asignada:* Indica el número de iteración asignado a la historia de usuario.
- *Estimación:* Número de horas estimadas para el desarrollo de la historia de usuario.
- *Programador Responsable:* Es la persona responsable de coordinar las tareas necesarias para el desarrollo y la solución de la historia de usuario.
- *Descripción:* Aquí se describe en breves palabras el requerimiento de la historia.
- *Observaciones:* Información adicional de la historia de usuario.

2.4.1.1.2 *Primera iteración*

En esta primera iteración se agrupan las historias de usuario que permiten un funcionamiento básico del sistema. La **Tabla 2.2** muestra la historia de usuario Desarrollo de la base de datos, la **Tabla 2.4** muestra la historia de usuario Autenticación, la **Tabla 2.5** muestra la historia de usuario Registro de Nuevos Usuarios, la **Tabla 2.6** muestra la historia de usuario Configuración del Perfil, la **Tabla 2.7** muestra la historia de usuario Cambio de Contraseña, y la **Tabla 2.7** muestra la historia de usuario Administración de Usuarios.

Historia de Usuario			
Número:	HU1B1	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Desarrollo de la base de datos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	1	**Estimación:**	16 Horas
Programador Responsable:	Víctor Reyes		
Descripción: Antes de comenzar el desarrollo del sistema se elaborará una base de datos que cumpla con los requerimientos del sistema prototipo.			
Observaciones: se elaborarán todos los componentes necesarios para el correcto funcionamiento de la base de datos y el sistema prototipo en general.			

Tabla 2.2 Historia de usuario Desarrollo de la base de datos

Historia de Usuario			
Número:	HU1MES2	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Autenticación		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	1	**Estimación:**	12 Horas
Programador Responsable:	Víctor Reyes		
Descripción: el usuario accede al sistema ingresando un nombre de usuario y una contraseña. Esta combinación permitirá comprobar en el servidor la identidad de la persona que quiere ingresar al sistema, y el servidor devolverá la información necesaria para el funcionamiento de las aplicaciones cliente.			
Observaciones: Usuarios y Usuarios VIP ingresarán al sistema por la aplicación móvil para utilizar el carro de compras virtual, Cajero y Administrador ingresarán a la aplicación de escritorio para realizar tareas de facturación e inventario.			

Tabla 2.3 Historia de usuario Autenticación

Historia de Usuario			
Número:	HU1MS3	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Registro de Nuevos Usuarios		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	1	**Estimación:**	8 horas
Programador Responsable:	Andrés Santos		
Descripción: el cliente dispone de una herramienta para registrarse en el sistema.			
Observaciones: los nuevos clientes del sistema podrán crear su cuenta con una opción que se incluirá en la autenticación. Los clientes únicamente podrán registrarse con perfil de Usuario. Solo los administradores podrán otorgar privilegios de usuario VIP.			

Tabla 2.4 Historia de usuario Registro de Nuevos Usuarios

Historia de Usuario			
Número:	HU1MES4	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Configuración del Perfil		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	1	**Estimación:**	8 horas
Programador Responsable:	Víctor Reyes		
Descripción: el usuario puede modificar sus datos personales.			
Observaciones: todos los usuarios dispondrán de una opción en la aplicación respectiva para actualizar sus datos personales.			

Tabla 2.5 Historia de usuario Configuración del Perfil

Historia de Usuario			
Número:	HU1MES5	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Cambio de Contraseña		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	1	**Estimación:**	4 horas
Programador Responsable:	Víctor Reyes		
Descripción: el usuario de la aplicación móvil o de escritorio tiene la posibilidad de cambiar la contraseña de su cuenta.			
Observaciones: deberá existir algún método de comprobación de ingreso de contraseña para asegurar que el usuario registra correctamente la contraseña.			

Tabla 2.6 Historia de usuario Cambio de Contraseña

Historia de Usuario			
Número:	HU1ES6	**Usuario:**	Administrador
Nombre Historia:	Gestión de Usuarios		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	1	**Estimación:**	8 horas
Programador Responsable:	Andrés Santos		
Descripción: el administrador puede: crear, modificar y eliminar usuarios. También en este componente se podrá asignar el perfil a cada usuario.			
Observaciones: un usuario administrador podrá modificar el perfil de un usuario para asignarlo a una categoría (Usuario, Usuario VIP, Cajero, Administrador).			

Tabla 2.7 Historia de usuario Gestión de Usuarios

2.4.1.1.3 Segunda iteración

En la segunda iteración se agruparán las historias de usuario que permiten el funcionamiento del carro de compras virtual. La **Tabla 2.8** muestra la historia de usuario Información del Artículo, la **Tabla 2.9** muestra la historia de usuario Selección de Artículos, la **Tabla 2.10** muestra la historia de usuario Lista del Carro de Compras, la **Tabla 2.11** muestra la historia de usuario Modificación de la Lista del Carro de Compras y la **Tabla 2.12** muestra la historia de usuario Finalizar Selección de Artículos.

Historia de Usuario			
Número:	HU2MS1	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Información del Artículo		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	2	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: Cuando el cliente pasa su dispositivo móvil por el tag la aplicación móvil consultará la información del artículo al servidor y desplegará una nueva actividad con la información del Artículo.			
Observaciones: Para la distribución de la información del artículo en la actividad se deberá considerar la limitación de espacio en una pantalla de dispositivos móviles y se deberá considerar un espacio libre para incluir herramientas que permita al usuario seleccionar el número de productos a adquirir y la inclusión de los productos al carro de compras.			

Tabla 2.8 Historia de usuario Información del Artículo

Historia de Usuario			
Número:	HU2MS2	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Selección de Artículos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	2	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: Después de que el usuario lee la información del artículo, selecciona el número de productos de un artículo especifico y los incluye en el carro de compras. Si el usuario no desea adquirir productos de ese artículo podrá regresar al carro de compras.			
Observaciones: Se deberá controlar que un producto pueda ser seleccionado una sola vez.			

Tabla 2.9 Historia de usuario Selección de Artículos

Historia de Usuario			
Número:	HU2MS3	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Lista del Carro de Compras		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	2	**Estimación:**	16 horas
Programador Responsable:	Víctor Reyes		
Descripción: El cliente dispone de una actividad donde puede visualizar los artículos agregados al carro de compras y el número de productos de cada uno de los mismos. También se dispondrá información resumida de los valores a cancelar hasta el momento.			
Observaciones: Los productos que se encuentren en la lista del carro de compras deberán estar en estado reservado y no se deberán encontrar en ningún otro carro de compras de clientes diferentes.			

Tabla 2.10 Historia de usuario Lista del Carro de Compras

Historia de Usuario			
Número:	HU2MS4	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Modificación de la Lista del Carro de Compras		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	2	**Estimación:**	8 horas
Programador Responsable:	Víctor Reyes		
Descripción: Si el usuario desea devolver productos ya reservados previamente deberá indicarlo mediante la aplicación.			
Observaciones: El cliente podrá devolver uno, varios o todos los productos del artículo seleccionado. Los productos devueltos pasaran al estado libre.			

Tabla 2.11 Historia de usuario Modificación de la Lista del Carro de Compras

Historia de Usuario			
Número:	HU2MS5	**Usuario:**	Usuario, Usuario VIP
Nombre Historia:	Finalizar la Selección de Artículos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	2	**Estimación:**	12 horas
Programador Responsable:	Víctor Reyes		
Descripción: Cuando el cliente finaliza la selección de artículos, se registrará la transacción y se le indicará el valor a cancelar en caja.			
Observaciones:			

Tabla 2.12 Historia de usuario Finalización de la Selección de Artículos

2.4.1.1.4 Tercera iteración

En esta iteración se agrupan todas las historias de usuario referentes a la administración de los registros de la base de datos. La **Tabla 2.13** muestra la historia de usuario Gestión de Artículos, la **Tabla 2.14** muestra la historia de usuario Gestión de Productos, la **Tabla 2.15** muestra la historia de usuario Gestión de Formas de Pago, la **Tabla 2.16** muestra la historia de usuario Gestión de Impuestos, la **Tabla 2.17** muestra la historia de usuario Gestión de Marcas y la **Tabla 2.18** muestra la historia de usuario Gestión de Modelos.

Historia de Usuario			
Número:	HU3ES1	**Usuario:**	Administrador
Nombre Historia:	Gestión de Artículos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	3	**Estimación:**	16 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tenar la posibilidad de agregar nuevos artículos, modificarlos y eliminarlos			
Observaciones: En esta sección el Administrador podrá controlar los descuentos que se realicen a los Usuarios y a los Usuarios VIP.			

Tabla 2.13 Historia de usuario Gestión de Artículos

Historia de Usuario			
Número:	HU3ES2	**Usuario:**	Administrador
Nombre Historia:	Gestión de Productos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	3	**Estimación:**	16 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tenar la posibilidad de: agregar nuevos productos, modificarlos y eliminarlos con el fin de tener un control del inventario existente de manera detallada.			
Observaciones: El control de que productos se pueden asignar al cliente, se lo realiza por medio del estado de los productos. Si el producto consta como libre podrá ser reservado para posteriormente comprarlo. Un producto nuevo ingresado por defecto se le asignara el estado de libre.			

Tabla 2.14 Historia de usuario Gestión de Productos

Historia de Usuario			
Número:	HU3ES3	**Usuario:**	Administrador
Nombre Historia:	Gestión de Formas de Pago		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Media
Iteración Asignada:	3	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tenar la posibilidad de: agregar nuevos tipos de pago, modificarlos y eliminarlos con el fin de permitir varios tipos de pagos.			
Observaciones:			

Tabla 2.15 Historia de usuario Gestión de Formas de Pago

Historia de Usuario			
Número:	HU3ES4	**Usuario:**	Administrador
Nombre Historia:	Gestión de Impuestos		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Media
Iteración Asignada:	3	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tenar la posibilidad de: agregar nuevos impuestos, modificarlos y eliminarlos con el fin de gestionar los impuestos a asignar a cada artículo.			
Observaciones:			

Tabla 2.16 Historia de usuario Gestión de Impuestos

Historia de Usuario			
Número:	HU3ES5	**Usuario:**	Administrador
Nombre Historia:	Gestión de Marcas		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	3	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tenar la posibilidad de: agregar nuevas marcas, modificarlas y eliminarlas con el fin de gestionar las marcas disponibles para cada artículo.			
Observaciones:			

Tabla 2.17 Historia de Usuario Gestión de Marcas

Historia de Usuario			
Número:	HU3ES6	**Usuario:**	Administrador
Nombre Historia:	Gestión de Modelos		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Media
Iteración Asignada:	3	**Estimación:**	12 horas
Programador Responsable:	Andrés Santos		
Descripción: el Administrador va a tener la posibilidad de: agregar nuevos modelos, modificarlos y eliminarlos.			
Observaciones:			

Tabla 2.18 Historia de Usuario Gestión de modelos

2.4.1.1.5 Cuarta iteración

En esta iteración se agrupan todas las historias de usuarios referentes a la facturación. La **Tabla 2.19** muestra la historia de usuario Nota de Venta, la **Tabla 2.20** muestra la historia de usuario Facturación, la **Tabla 2.21** muestra la historia

de usuario Impresión de la Factura y la **Tabla 2.22** muestra la historia de usuario Historial de Facturas.

Historia de Usuario			
Número:	HU4MS1	**Usuario:**	Cajero, Administrador
Nombre Historia:	Nota de Venta		
Prioridad en Sistema:	Media	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	4	**Estimación:**	16 horas
Programador Responsable:	Víctor Reyes		
Descripción: cuando el cliente finaliza el proceso de selección de productos en la aplicación móvil, el cliente visualizará una nota de venta con los productos que selecciono y un mensaje informativo para que se acerque a pagar a una caja			
Observaciones: El Usuario o Usuario VIP debió finalizar el proceso de selección para acercarse a cancelar los valores en caja.			

Tabla 2.19 Historia de usuario Nota de Venta

Historia de Usuario			
Número:	HU4ES2	**Usuario:**	Cajero, Administrador
Nombre Historia:	Facturación		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	4	**Estimación:**	16 horas
Programador Responsable:	Víctor Reyes		
Descripción: el cajero tendrá en la aplicación de escritorio una lista donde podrá ver todos los clientes que finalizaron la selección de producto. El cajero seleccionará uno de esos usuarios, y luego seleccionará el método de pago para emitir la factura.			
Observaciones: Una vez que el cajero recibe el dinero de acuerdo a la forma de pago genera la factura, en este momento los productos del carro de compras pasan al estado vendido.			

Tabla 2.20 Historia de usuario Facturación

Historia de Usuario			
Número:	HU4ES3	**Usuario:**	Cajero, Administrador
Nombre Historia:	Imprimir Factura		
Prioridad en Sistema:	Baja	**Riesgo en desarrollo:**	Media
Iteración Asignada:	4	**Estimación:**	12 horas
Programador Responsable:	Víctor Reyes		
Descripción: impresión de la factura para entrega al usuario.			
Observaciones:			

Tabla 2.21 Historia de usuario Imprimir de la Factura

Historia de Usuario			
Número:	HU4ES4	**Usuario:**	Cajero, Administrador
Nombre Historia:	Historial de Facturas		
Prioridad en Sistema:	Alta	**Riesgo en desarrollo:**	Alta
Iteración Asignada:	4	**Estimación:**	16 horas
Programador Responsable:	Víctor Reyes		
Descripción: el cajero tendrá un historial de facturas para poder realizar anulaciones o reimpresiones.			
Observaciones: Una factura ya impresa no puede ser modificada.			

Tabla 2.22 Historia de usuario Historial de facturas

2.4.1.1.6 Quinta iteración

En esta iteración se agrupan todas las historias de usuario referentes a la ayuda de la aplicación. La **Tabla 2.23** muestra la historia de usuario Acerca de y la **Tabla 2.24** muestra la historia de usuario Ayuda.

Historia de Usuario			
Número:	HU5ME1	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Acerca de		
Prioridad en Sistema:	Baja	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	5	**Estimación:**	8 horas
Programador Responsable:	Andrés Santos		
Descripción: se incluirá en cada aplicación información de la empresa propietaria del sistema así como información de los programadores que elaboraron el sistema.			
Observaciones:			

Tabla 2.23 Historia de usuario Acerca de

Historia de Usuario			
Número:	HU5ME2	**Usuario:**	Usuario, Usuario VIP, Cajero, Administrador
Nombre Historia:	Ayuda		
Prioridad en Sistema:	Baja	**Riesgo en desarrollo:**	Baja
Iteración Asignada:	5	**Estimación:**	8 horas
Programador Responsable:	Andrés Santos		
Descripción: se incluirá en cada aplicación un pequeño manual de usuario para conocer la forma de uso de la aplicación.			
Observaciones:			

Tabla 2.24 Historia de usuario Ayuda

2.4.1.2 Estimación de tiempo de desarrollo por iteraciones

El tiempo estimado que se empleará en desarrollar los componentes necesarios para cada historia de usuario, fue asignado en base al riesgo en desarrollo de cada historia de usuario. La **Tabla 2.25** muestra los tiempos que se asignan a cada categoría de riesgo en desarrollo.

Las estimaciones de tiempo se fijaron en base a experiencias de proyectos anteriores realizados por el equipo de trabajo. La prioridad no influye en la estimación de tiempo, sirve para dar una idea al equipo de trabajo la importancia de la historia de usuario en el sistema prototipo.

Categoría	Asignación de tiempo (Horas)
Alta	16
Media	12
Baja	8

Tabla 2.25 Asignación de tiempo de desarrollo por riesgo en desarrollo

2.4.1.2.1 Primera iteración

En esta primera iteración se agrupan las historias que permiten un primer funcionamiento básico del sistema. Específicamente las historias de usuario que permitan la autenticación del sistema, registros de nuevos usuarios y manejo de la cuenta del usuario. Al concluir esta iteración, se tendrá un entregable del sistema prototipo, que permitirá tener una visión general de cómo estará estructurado el sistema. La **Tabla 2.26** muestra el detalle del estimado de tiempo de desarrollo para la primera iteración.

2.4.1.2.2 Segunda iteración

En la segunda iteración se desarrollan todos los componentes del sistema que permitan el funcionamiento del carro de compras virtual. Al concluir esta iteración, se puede realizar un segundo entregable del sistema prototipo, ya que estará en la capacidad de realizar un proceso de selección de productos por parte del cliente. La **Tabla 2.27** muestra el detalle del estimado de tiempo de desarrollo para la segunda iteración.

Componente del sistema	Historia de Usuario	Prioridad	Riesgo en Desarrollo	Estimación (Horas)
Base de Datos	HU1B1 – Desarrollo de la Base de Datos	Alta	Alta	16
Escritorio	HU1ES6 – Administración de usuarios	Alta	Baja	8
Móvil	HU1MS3 – Registro de nuevos usuarios	Media	Baja	8
Escritorio y Móvil	HU1MES2 – Autenticación al sistema	Alta	Media	12
	HU1MES4 – Configuración del perfil	Media	Baja	8
	HU1MES5 – Cambio de contraseña	Media	Baja	4
			Total	56

Tabla 2.26 Tiempo de desarrollo de la primera iteración

Componente del sistema	Historia de Usuario	Prioridad	Riesgo en Desarrollo	Estimación (Horas)
Móvil	HU2MS1 – Información del artículo	Alta	Media	12
	HU2MS2 – Selección de productos	Alta	Media	12
	HU2MS3 – Lista del carro de compras	Alta	Alta	16
	HU2MES4 – Modificación de la lista del carro de compras	Media	Baja	8
	HU2MES5 – Finalización de la selección de artículos	Alta	Media	12
			Total	60

Tabla 2.27 Tiempo de desarrollo de la segunda iteración

2.4.1.2.3 Tercera iteración

En la tercera iteración se desarrollan aquellos componentes que sirven para realizar una adecuada gestión de inventario en el sistema. Al finalizar esta iteración se podrá realizar un tercer entregable ya que se contará con gran parte de las funciones del perfil de administrador en la aplicación de escritorio. La **Tabla 2.28** muestra el detalle del estimado de tiempo de desarrollo para la tercera iteración.

2.4.1.2.4 Cuarta iteración

En la cuarta iteración se desarrollan los componentes necesarios para el sistema de facturación. Al finalizar esta iteración se podrá realizar un cuarto entregable y la versión final del sistema prototipo ya que el mismo tendrá la capacidad de operar en todos los aspectos considerados: desde la selección de artículos por parte del cliente, hasta el sistema de facturación por parte de los cajeros, además del manejo de inventario. La **Tabla 2.29** muestra el detalle del estimado de tiempo de desarrollo para la cuarta iteración.

Componente del sistema	Historia de Usuario	Prioridad	Riesgo en Desarrollo	Estimación (Horas)
Escritorio	HU3ES1 – Gestión de artículos	Alta	Alta	16
	HU3ES2 – Gestión de productos	Alta	Alta	16
	HU3ES3 – Gestión de formas de pago	Media	Media	12
	HU3ES4 – Gestión de impuestos	Media	Media	12
	HU3ES5 – Gestión de marcas	Alta	Media	12
	HU3ES6 – Gestión de modelos	Alta	Media	12
			Total	80

Tabla 2.28 Tiempo de desarrollo de la tercera iteración

Componente del sistema	Historia de Usuario	Prioridad	Riesgo en Desarrollo	Estimación (Horas)
Móvil	HU4MS1 – Nota de venta	Media	Alta	16
Escritorio	HU4ES2 – Facturación	Alta	Alta	16
	HU1MES3 – Impresión de la factura	Baja	Media	12
	HU1MES4 – Historial de facturas	Alta	Alta	16
			Total	84

Tabla 2.29 Tiempo de desarrollo de la cuarta iteración

2.4.1.2.4 Quinta iteración

En la quinta iteración se desarrollan las ayudas de usuario y la información general de la aplicación. Al final de esta iteración se podrá entregar la versión final del sistema prototipo. La **Tabla 2.30** muestra el detalle del estimado de tiempo de desarrollo para la quinta iteración.

Componente del sistema	Historia de Usuario	Prioridad	Riesgo en Desarrollo	Estimación (Horas)
Móvil y Escritorio	HU5ME1 – Acerca de	Baja	Baja	8
	HU5ME2 – Ayuda	Baja	Baja	8
			Total	16

Tabla 2.30 Tiempo de desarrollo de la quinta iteración

2.4.1.3 Plan de Entregas

Además de la estimación de tiempo que se muestra en las secciones anteriores, se deben considerar también tiempos para: análisis de requerimientos, reuniones de trabajo, pruebas entre otras. El porcentaje de holgura se lo considerará de la siguiente forma:

20% para análisis de requerimientos, reuniones del equipo de trabajo e imprevistos.

30% para la realización de pruebas unitarias, de carga y de aceptación.

Es por ello que, al menos, se debe agregar la mitad de tiempo considerado al tiempo inicialmente considerado.

Para establecer los tiempos de desarrollo (considerados por el equipo de trabajo), se aclara que un mes tiene 20 días, una semana consta de 5 días y cada jornada de trabajo tiene 4 horas para la ejecución de tareas en el proyecto. En la **Tabla 2.31** se resume el tiempo estimado de entrega del proyecto por iteraciones.

Como se puede observar en la **Tabla 2.31** cada iteración tomará en promedio un mes y medio, con las excepciones de la primera y quinta iteración que no tienen mayor complejidad en el desarrollo. De acuerdo a la misma tabla el proyecto se estima concluir en aproximadamente 6 meses.

Iteración	Tiempo Estimado (Horas)	Holgura (Horas)	Total (Horas)	Total (Semanas)	Total (Meses)
Primera	56	20	76	3,8	0,95
Segunda	60	30	90	4,5	1,13
Tercera	80	40	120	6	1,5
Cuarta	84	42	126	6,3	1,6
Quinta	16	8	24	1,2	0,3
		TOTAL:	436	21,8	5,48

Tabla 2.31 Tiempo de entrega del proyecto por iteraciones

La **Tabla 2.32** y la **Tabla 2.33** muestran las historias de usuario que se tendrán concluidas en cada entregable. Vale la pena aclarar que el servidor SOA se realiza paralelamente con el desarrollo de las aplicaciones clientes, ya que las aplicaciones clientes necesitan consumir los métodos del Servicio Web alojado en el servidor SOA para su funcionamiento.

Iteración	Componente del sistema	Historia de Usuario
Primera iteración	Base de datos	Desarrollo de la base de datos
	Escritorio	Administración de usuarios
	Móvil	Registro de nuevos usuarios
	Móvil y Escritorio	Autenticación al sistema
		Configuración del perfil
		Cambio de contraseña
PRIMERA ENTREGA		
Segunda iteración	Móvil	Información del artículo
		Selección de productos
		Lista del carro de compras
		Modificación de la lista del carro de compras
		Finalización de la selección de artículos
SEGUNDA ENTREGA		

Tabla 2.32 Plan de entregas

Iteración	Componente del sistema	Historia de Usuario
Tercera iteración	Escritorio	Gestión de artículos
		Gestión de productos
		Gestión de formas de pago
		Gestión de impuestos
		Gestión de marcas
		Gestión de modelos
TERCERA ENTREGA		
Cuarta iteración	Móvil	Nota de venta
	Escritorio	Facturación
		Impresión de factura
		Historial de facturas
CUARTA ENTREGA		
Quinta iteración	Móvil y Escritorio	Acerca de
		Ayuda de la aplicación
QUINTA ENTREGA		

Tabla 2.33 Plan de entregas (continuación)

Finalmente la **Tabla 2.34** muestra los tiempos necesarios para realizar cada entrega.

Iteración	Tiempo estimado (Horas)	Holgura (Horas)	Total (Horas)	Total (Semanas)	Total (Meses)
Primera	40	20	60	3	0,75
PRIMERA ENTREGA			**60**	**3**	**0,75**
Segunda	60	30	90	4,5	1,13
SEGUNDA ENTREGA			**150**	**7,5**	**1,88**
Tercera	80	40	120	6	1,5
TERCERA ENTREGA			**270**	**13,5**	**3,38**
Cuarta	84	42	126	6,3	1,6
CUARTA ENTREGA			**396**	**19,8**	**4,98**
Quinta	16	8	24	1,2	0,3
QUINTA ENTREGA			**420**	**21**	**5,28**

Tabla 2.34 Tiempo estimado para cada entrega

2.4.2 DISEÑO DEL SISTEMA PROTOTIPO

En esta sección se presentan los diagramas de clases y de secuencia del sistema prototipo. En el desarrollo del sistema no se siguió un modelo rígido, sino que se siguió un modelo cambiante que se va consolidando a medida que se avanza en las iteraciones, es por eso que solo se presentan los diagramas obtenidos al final del desarrollo del proyecto.

El sistema prototipo emplea un modelo de tres niveles lógicos, lo que permite tener: mayor control sobre el código, disminuir su complejidad, dividir de forma estructural al sistema. Estos niveles son: nivel de presentación, nivel de negocios y nivel de datos. El nivel de datos es el encargado de almacenar y obtener la información que el cliente necesita. El nivel de negocios es el encargado de consumir los servicios de la capa de datos, este nivel es el encargado de manejar las operaciones internas del sistema. Por último, el nivel de presentación es el encargado de presentar los resultados e interactuar con el usuario.

2.4.2.1 Nivel de Datos

Lo primero que se debe realizar en el diseño del sistema, es la elaboración de un adecuado modelo de datos. En el transcurso del desarrollo del proyecto, el modelo inicial propuesto deberá cambiar necesariamente, de acuerdo a los nuevos requerimientos que se van presentando en el desarrollo del proyecto. En

consecuencia, el modelo expresado muestra el diagrama obtenido al final del proceso de desarrollo del proyecto. En la **Figura 2.22** se presenta el diagrama final de la base de datos.

En el modelo de datos de la **Figura 2.22** se definen las tablas: `Tipo_usuario`, `Articulo`, `Marca`, `Impuesto`, `Modelo`, `Producto`, `Estado_producto`, `tipo_pago`, `Usuario`, `Producto_seleccion`, `Seleccion` y `Factura`. La tabla `Producto_seleccion` es una tabla auxiliar que se utiliza por existir una relación de muchos a muchos entre las tablas `Producto` y `Seleccion`. La tabla `Modelo` contiene los campos que se mostrarán en la aplicación móvil al momento de leer el tag NFC. La tabla `Seleccion` es la que permite identificar la lista de compras del carro de compras virtual de cada cliente.

En esta sección se presenta también el conjunto de procedimientos almacenados que permitirán interactuar con la información de las tablas en la base de datos. La **Tabla 2.38** y **Tabla 2.39** presentan los procedimientos almacenados que se emplearán. De igual manera, se define en la **Tabla 2.40** los *triggers* o desencadenadores a utilizarse para manejar correctamente la base de datos.

Los datos almacenados en la base deberán conservarse a pesar de que estos sean eliminados en el sistema, es por esto que cada tabla define un campo llamado estado el cual es un valor entero que si se marca con el valor de 1, el registro existe en el sistema y si se marca con el valor de 0, el registro ha sido eliminado del sistema.

Para poder determinar en qué estado se encuentra cada selección, se utiliza el campo `sel_descripcion` para marcar cada selección con un estado determinado. De esta manera, el sistema puede identificar en que parte del proceso de compra se encuentra el usuario y realizar las acciones adecuadas. La **Tabla 2.35** muestra los posibles estados de una selección.

De la misma manera, para manejar el estado de un producto en la base de datos se establecen estados de los productos, los cuales se describen en la **Tabla 2.36**.

Estado de la selección	Descripción
En proceso	Existe una selección en proceso y esta no ha sido finalizada.
Finalizado	El cliente finalizo la selección.
Facturado	El cliente cancelo el valor de la selección realizada y se emitió la factura.

Tabla 2.35 Estados de una selección

Estado de un producto	Descripción
Libre	El Producto está disponible para su venta.
Reservado	El Producto se encuentra Reservado.
Vendido	El Producto se encuentra Vendido.
Eliminado	El Producto fue eliminado del sistema.

Tabla 2.36 Estados de un producto

Por último, para mostrar al usuario la información, se elaboran vistas, las cuales son tablas virtuales que muestran información que se obtiene de las tablas presentes en la base de datos. La **Tabla 2.37** muestran las vistas que se utilizarán.

Vista	Descripción
`ViewArticulo`	Muestra los campos: ID, nombre y descripción del artículo.
`ViewFactura`	Muestra los campos: ID, Selección, Subtotal, Impuestos, Descuentos, Total, Cedula, Tipo de Pago, Fecha y Nombre del cliente. Estos datos son los que se utilizan para elaborar la factura
`ViewImpuesto`	Muestra los campos: ID, Nombre, Porcentaje y Descripción del impuesto.
`ViewMarca`	Muestra los campos: ID, Nombre y Descripción de la marca.
`ViewModelo`	Muestra los campos: ID, Nombre, Marca, Modelo, Descripción, Url de la imagen, Disponibilidad, Valor, Descuento, Descuento VIP e Impuesto del modelo.
`ViewProducto`	Muestra los campos: ID, Artículo, Marca, Modelo, Serial, Estado, Fecha de Entrada y Fecha de Venta del producto.
`ViewSeleccion`	Muestra los campos: ID, Cedula del cliente, Descripción, Nombre y Costo de la selección.
`ViewTipoPago`	Muestra los campos: ID, Nombre y Descripción del tipo de pago.
`ViewUsuario`	Muestra los campos: Cedula, Nombres, Apellidos, Dirección, Teléfono, Correo, Tipo de usuario, Alias y *Password* del usuario.

Tabla 2.37 Vistas utilizadas en la base de datos

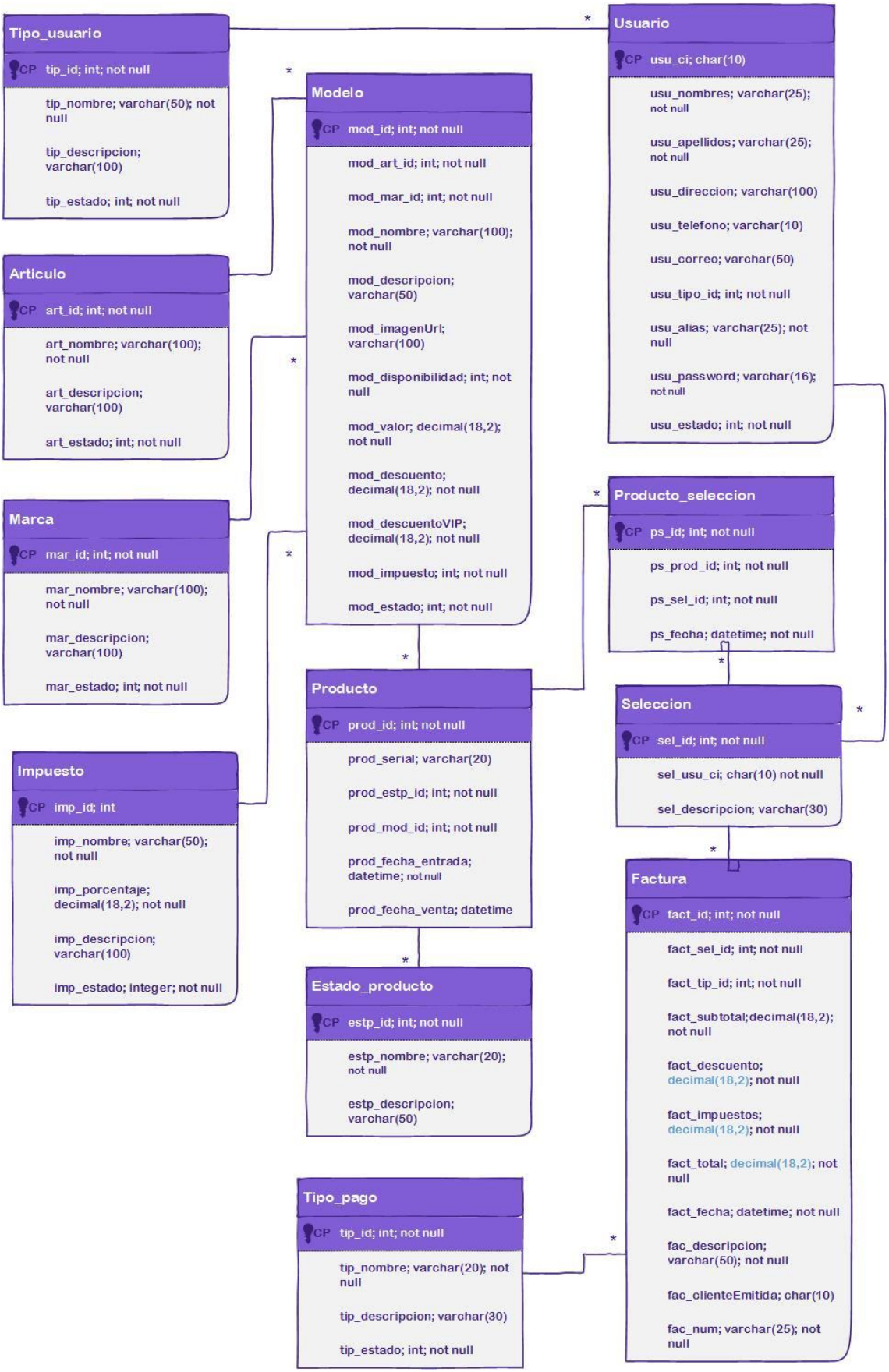

Figura 2.22 Diagrama relacional de la base de datos

PROCEDIMIENTO ALMACENADO	DESCRIPCIÓN
Sp_agregarProductoCarro	Agrega un producto a la lista de compras del cliente
Sp_art_del	Marca un artículo como eliminado
Sp_art_get	Obtiene un registro de la tabla Modelo
Sp_art_ins	Inserta un registro a la tabla Articulo
Sp_art_obtener_nuevoId	Obtiene el siguiente ID disponible en la tabla Articulo
Sp_art_upd	Actualiza un registro de la tabla Articulo
Sp_cargarSeleccion	Carga la lista de compras del cliente en base al estado de la selección
Sp_controlAccesoCarro	Identifica en qué estado se encuentra la selección de un cliente
Sp_datosSeleccion	Obtiene el resumen de la selección (subtotal, impuestos, etc.)
Sp_eliminarProductoCarro	Elimina todos los productos de un artículo existentes en una selección
Sp_eliminarUnProductoCarro	Disminuye un elemento de la cantidad de un artículo
Sp_fac_anul	Anula una factura
Sp_fac_del	Marca un registro de la tabla Factura como eliminado.
Sp_fac_ins	Inserta un registro en la tabla Factura
Sp_fac_obtener_nuevoId	Obtiene el siguiente ID disponible en la tabla Factura
Sp_fac_upd	Actualiza un registro de la tabla Factura
Sp_finalizarSeleccion	Establece una selección como finalizada
Sp_imp_del	Marca un registro de la tabla Impuesto como eliminado
Sp_imp_ins	Inserta un nuevo registro en la tabla impuesto
Sp_imp_obtener_idNuevo	Obtiene el siguiente id disponible en tabla Impuesto
Sp_imp_upd	Actualiza un registro de la tabla Impuesto
Sp_login	Procedimiento almacenado para verificar el par alias - contraseña y permitir la autenticación de un usuario
Sp_obtenerUsuarioPorCi	Obtiene un registro de la tabla Usuario buscando por el atributo cedula
Sp_prod_del	Marca un producto como eliminado
Sp_prod_id_upd	Actualiza el id de un registro de la tabla Producto
Sp_prod_ins	Inserta un registro en la tabla Producto
Sp_prod_sel_del	Marca un registro de la tabla Producto_seleccion como eliminado
Sp_prod_sel_ins	Inserta un registro en la tabla Producto_seleccion
Sp_prod_upd	Actualiza un registro de la tabla Producto
Sp_sel_del	Marca un registro de la tabla Seleccion como eliminado
Sp_sel_ins	Inserta un registro en la tabla Seleccion
Sp_sel_upd	Actualiza un registro en la tabla Seleccion
Sp_tipp_upd	Actualiza un registro de la tabla Tipo_pago

Tabla 2.38 Procedimientos almacenados utilizados en la base de datos

PROCEDIMIENTO ALMACENADO	DESCRIPCIÓN
Sp_tipp_ins	Inserta un registro en la tabla Tipo_pago
Sp_tipp_obtener_nuevoId	Obtiene el siguiente id disponible de la tabla Tipo_pago
Sp_tipp_upd	Actualiza un registro de la tabla Tipo_pago
Sp_tipu_del	Marca un registro de la tabla Tipo_usuario como eliminado
Sp_tipu_ins	Inserta un registro en la tabla Tipo_usuario
Sp_tipu_obtener_nuevoid	Obtiene el siguiente id disponible de la tabla Tipo_usuario
Sp_tipu_upd	Actualiza un registro de la tabla Tipo_usuario
Sp_usu_ci_upd	Actualiza el campo ci de un registro en la Tipo_usuario
Sp_usu_cont_upd	Actualiza el campo contraseña de un registro en la tabla Usuario
Sp_usu_del	Marca un registro de la tabla Usuario como eliminado
Sp_usu_ins	Inserta un registro en la tabla Usuario
Sp_usu_upd	Actualiza un registro de la tabla Usuario, este procedimiento se lo invoca desde la aplicación móvil
Sp_usu_upd_adm	Actualiza un registro de la tabla Usuario, se lo llama desde la aplicación de escritorio

Tabla 2.39 Procedimientos almacenados utilizados en la base de datos, (continuación)

TRIGGER	DESCRIPCIÓN
Tg_actualizar_art_disponibilidad_delete	Actualiza la disponibilidad de un artículo al momento de eliminar un registro de la tabla producto
Tg_actualizar_art_disponibilidad	Actualiza la disponibilidad de un artículo al momento de cambiar el estado de un producto
Tg_actualizar_disponibilidad_eliminar	Controla que un producto pase al estado libre después de haberlo eliminado en la tabla Producto_seleccion
Tg_actualizar_disponibilidad	Controla que un producto pase al estado reservado después de haberlo agregado a la tabla Producto_seleccion
Tg_actualizar_estado_vendido	Actualiza el estado del producto a vendido al momento de crearse una factura de su selección respectiva
Tg_actualizar_factura	Genera los valores de subtotal, impuesto, descuentos y total de la factura al momento de emitirse la misma.
Tg_actualizar_fecha_venta_producto	Actualiza la fecha de venta de un producto al momento que es vendido
Tg_eliminarSeleccion	Elimina una selección después de eliminar una factura relacionada a dicha selección
Tg_solo_una_seleccion_simultanea	Controla que un producto en estado reservado o comprado no se pueda agregar a la tabla Producto_seleccion

Tabla 2.40 *Triggers* utilizados en la base de datos

2.4.2.2 Nivel de Negocios

El Servicio Web utilizado implementa el nivel de negocios. En este nivel, los métodos no trabajan directamente con la base de datos sino que sirve de intermediario entre el nivel de presentación y el nivel de datos. La **Figura 2.23** muestra el diagrama de clases del Servicio Web `ServicioComprasVirtuales`. La **Tabla 2.41** y la **Tabla 2.42** proporcionan una breve descripción de cada método definido en el Servicio Web.

La arquitectura utilizada para implementar el Servicio Web de este sistema es la arquitectura REST. En esta arquitectura no se guarda estados entre el cliente y el servidor. Para manejar sesiones en el sistema prototipo se debe implementar algún método que guarde las sesiones individuales de los clientes que consumen el Servicio Web, es por eso que el sistema hace uso del objeto `context` de la clase `HttpContext`[65] para guardar los estados de las sesiones individuales de cada cliente que consume el Servicio Web.

Para realizar la comunicación con la base de datos se utiliza el objeto `data` de la clase `DatosComprasVirtualesDataContext`, clase que se genera mediante el uso de la herramienta LINQ to SQL[66].

2.4.2.3 Nivel de Presentación

En el nivel de presentación se implementan las aplicaciones clientes que consumen el Servicio Web, estas aplicaciones son: Aplicación Móvil y Aplicación de escritorio.

[65] **HttpContext:** Clase que encapsula la información necesaria relativa a una petición HTTP individual.
[66] **LINQ to SQL:** Herramienta que proporciona una infraestructura en tiempo de ejecución para administrar los datos relacionales.

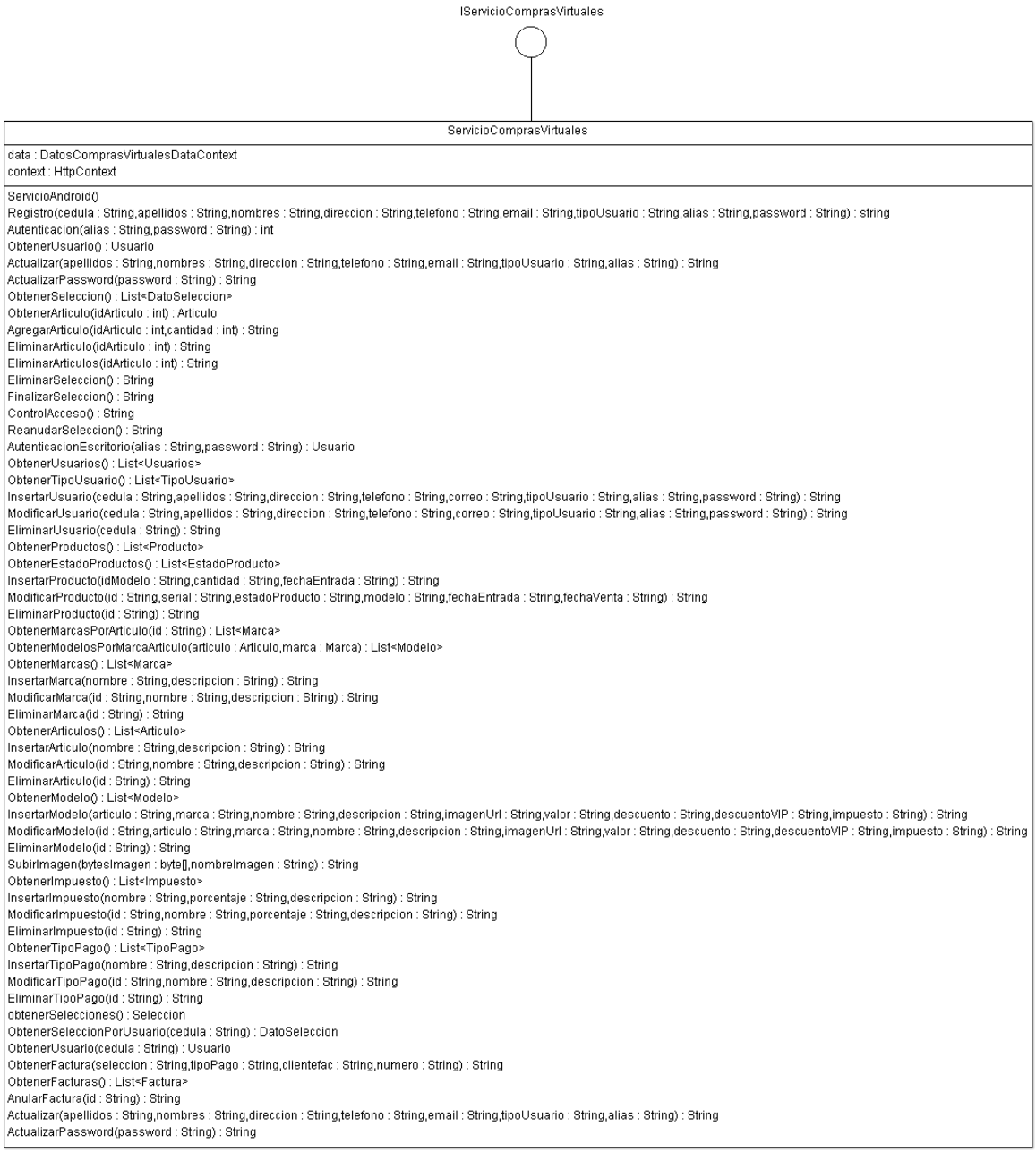

Figura 2.23 Diagrama de clases del Servicio Web `ServicioComprasVirtuales`

METODO	DESCRIPCIÓN
ServicioComprasVirtuales	Constructor de la clase que inicializa las variables en el Servicio Web
Registro	Registra un nuevo usuario en la base de datos
Autenticacion	Valida el alias y *password* del usuario para permitir el ingreso a la aplicación móvil
ObtenerUsuario	Obtiene un usuario
Actualizar	Actualiza los datos de un usuario
ActualizarPassword	Actualiza la contraseña de un usuario
ObtenerSeleccion	Obtiene la lista de productos seleccionados por un usuario.
ObtenerArticulo	Obtiene un artículo
AgregarArticulo	Agrega un artículo a la lista de compras actual
EliminarArticulo	Disminuye la cantidad de un artículo de la lista de compras actual
EliminarArticulos	Elimina todos los productos correspondientes a un artículo de la lista de compras
EliminarSeleccion	Elimina una selección en proceso de un cliente, este método se utilizará para limpiar el carro de compras
FinalizarSeleccion	Concluye el proceso de selección de artículos en el carro de compras virtual de un cliente
ControlAcceso	Controla el acceso del cliente al carro de compras virtual de acuerdo a el estado de la selección; si existe una selección en proceso se carga esta selección, si la selección está finalizada se carga la nota de venta, si la selección esta facturada se crea una nueva selección
ReanudarSeleccion	Método utilizado en la aplicación móvil para permitirle al cliente retornar al proceso de selección de productos en el caso de que requiera modificar una selección en estado finalizado.
AutenticacionEscitorio	Método para validar el alias y *password* del usuario y permitir el ingreso a la aplicación de escritorio
ObtenerUsuarios	Obtiene los registros de usuarios del sistema
ObtenerTipoUsuario	Obtiene los registros de tipos de usuario del sistema
InsertarUsuario	Ingresa un nuevo usuario al sistema
ModificarUsuario	Modifica un usuario registrado en el sistema
EliminarUsuario	Elimina un usuario del sistema
ObtenerProductos	Obtiene los registros de productos del sistema
ObtenerEstadoProductos	Obtiene los registros de estado producto del sistema
InsertarProducto	Inserta un producto
ModificarProducto	Modifica un producto
EliminarProducto	Elimina un producto

Tabla 2.41 Métodos del Servicio Web ServicioComprasVirtuales

METODO	DESCRIPCION
ObtenerMarcasPorArticulo	Obtiene el listado de marcas disponible para un artículo determinado
ObtenerModelosPorMarcaArticulo	Obtiene el listado de modelos disponibles para un artículo de una marca determinada
ObtenerMarcas	Obtiene el listado de marcas disponibles en el sistema
InsertarMarca	Inserta una marca
ModificarMarca	Modifica una marca
EliminarMarca	Elimina una marca
ObtenerArticulos	Obtiene los registros de artículos del sistema
InsertarArticulo	Inserta un artículo
ModificarArticulo	Modifica un artículo
EliminarArticulo	Elimina un artículo
ObtenerModelo	Obtiene el listado de modelos disponibles en el sistema
InsertarModelo	Inserta un modelo
ModificarModelo	Modifica un modelo
EliminarModelo	Elimina un modelo
SubirImagen	Este método sube la imagen de un artículo determinado al servidor
ObtenerImpuesto	Obtiene los registros de impuestos en el sistema
InsertarImpuesto	Inserta un impuesto
ModificarImpuesto	Modifica un impuesto
EliminarImpuesto	Elimina un impuesto
ObtenerTipoPago	Obtiene los registros de tipos de pago en el sistema
InsertarTipoPago	Inserta un nuevo tipo de pago en el sistema
ModificarTipoPago	Modifica un tipo de pago
EliminarTipoPago	Elimina un tipo de pago
ObtenerSelecciones	Obtiene las selecciones que están en estado de finalizado
ObtenerSeleccionPorUsuario	Obtiene la selección en estado finalizado de un usuario
ObtenerUsuario	Método para obtener los datos del usuario para la factura
ObtenerSeleccion	Método para obtener los artículos a facturar
ObtenerFactura	Método para obtener una factura específica
ObtenerFacturas	Método para obtener registros de facturas realizadas
AnularFactura	Método para anular una factura
Actualizar	Método para actualizar los datos de un usuario
ActualizarPassword	Método para actualizar la contraseña de un usuario

Tabla 2.42 Métodos del Servicio Web `ServicioComprasVirtuales` (continuación)

2.4.3.3.1 Aplicación móvil

La aplicación móvil diseñada para Android, cumple la función de una interfaz de usuario para los clientes del local. Esta aplicación principalmente debe implementar el carro de compras virtual mediante el consumo del Servicio Web. Las figuras: **Figura 2.24**, **Figura 2.25**, **Figura 2.26**, y **Figura 2.27** muestran el diagrama de clases de la aplicación móvil.

Las clases que comienzan con `WebServiceClient` son implementaciones de la clase abstracta `AsyncTask`. Para esta aplicación, las implementaciones de la clase `AsyncTask` son utilizadas para realizar operaciones de red, debido a que las últimas versiones de Android no permiten ejecutar tareas de red en el hilo principal, con el objetivo de mejorar el rendimiento de la aplicación. La clase `AsyncTask` permite el uso de forma simplificada de hilos secundarios en la aplicación.

Las clases que terminan en `Activity` son utilizadas para controlar cada actividad en la aplicación. Las clases que requieren consumir el Servicio Web se incluyen los métodos para realizar las peticiones necesarias.

Por ultimo clases que no tienen prefijos o sufijos en el nombre como la clase `Usuario` son utilizadas para recibir o enviar los datos entre la aplicación y el servidor.

La **Tabla 2.43** y la **Tabla 2.44** explican a detalle las variables y métodos de la clase `CarroActivity`, la clase más representativa de la aplicación móvil. El resto de clases tienen variables y métodos similares. Es importante indicar que la actividad antes mencionada, implementa las funcionalidades más importantes de la aplicación; entre las que están las funcionalidades de la selección de productos a través de la interacción del móvil con el tag NFC y la lista de productos agregados al carro de compras virtual.

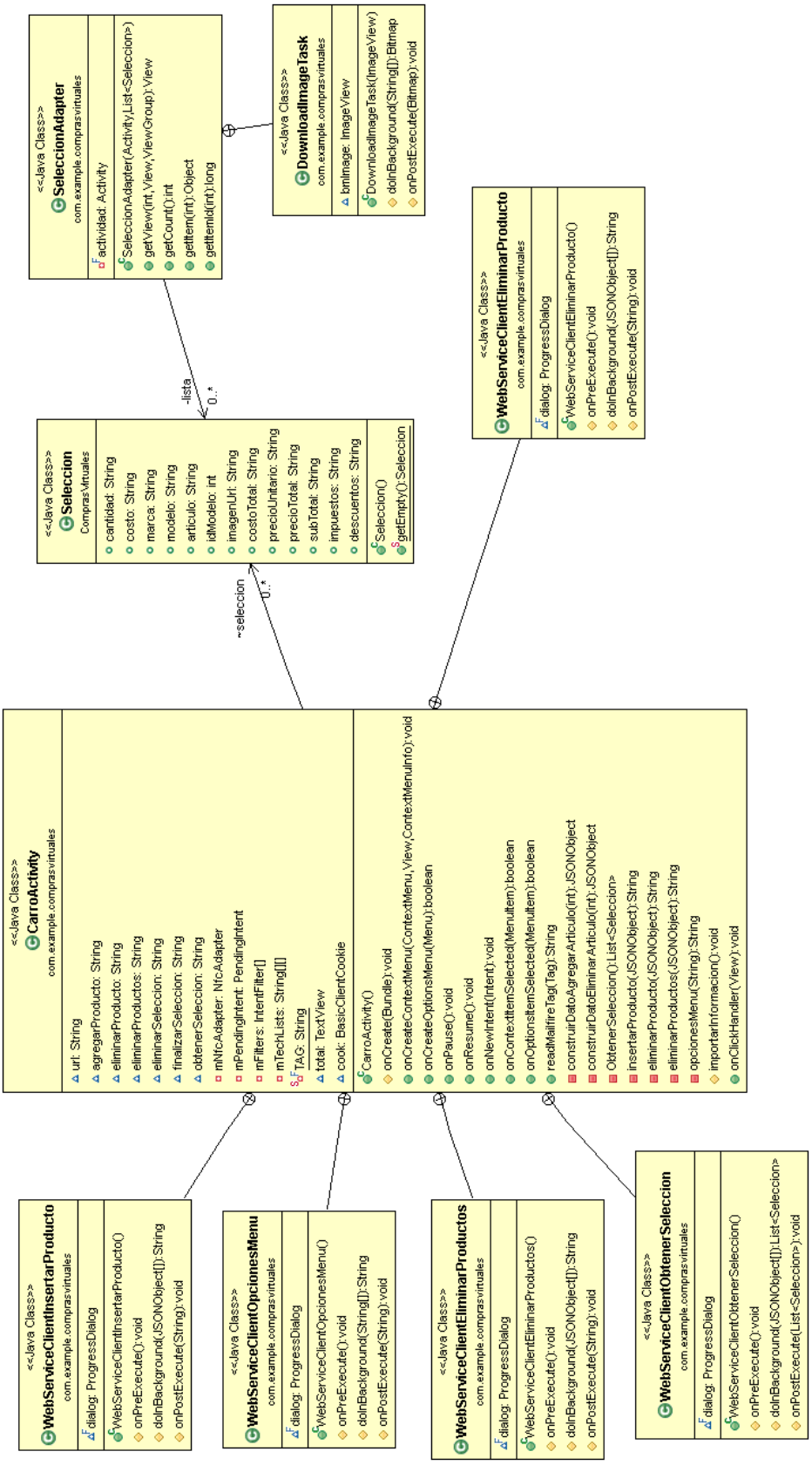

Figura 2.24 Diagrama de clases de la aplicación móvil (primera parte)

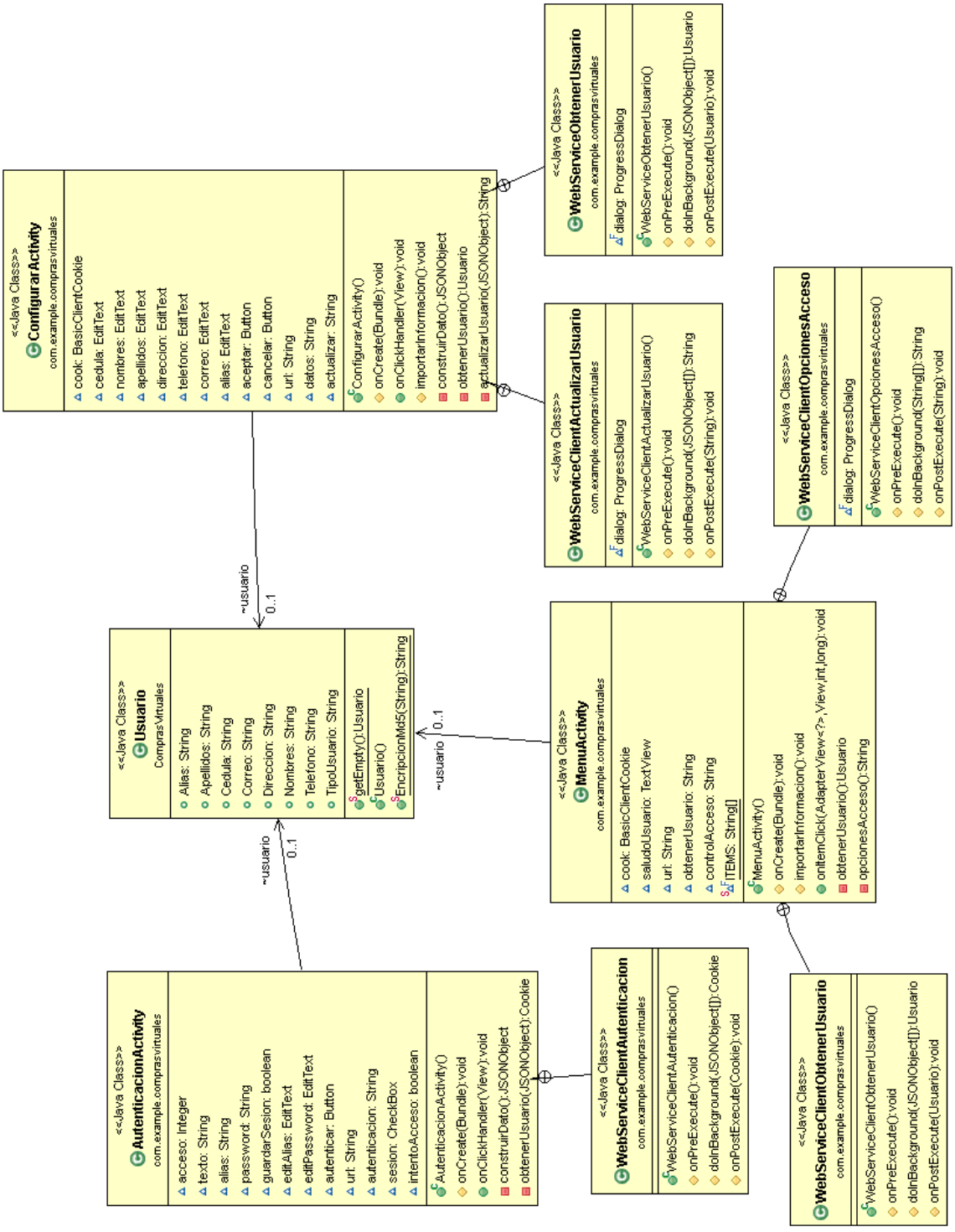

Figura 2.25 Diagrama de clases de la aplicación móvil (segunda parte)

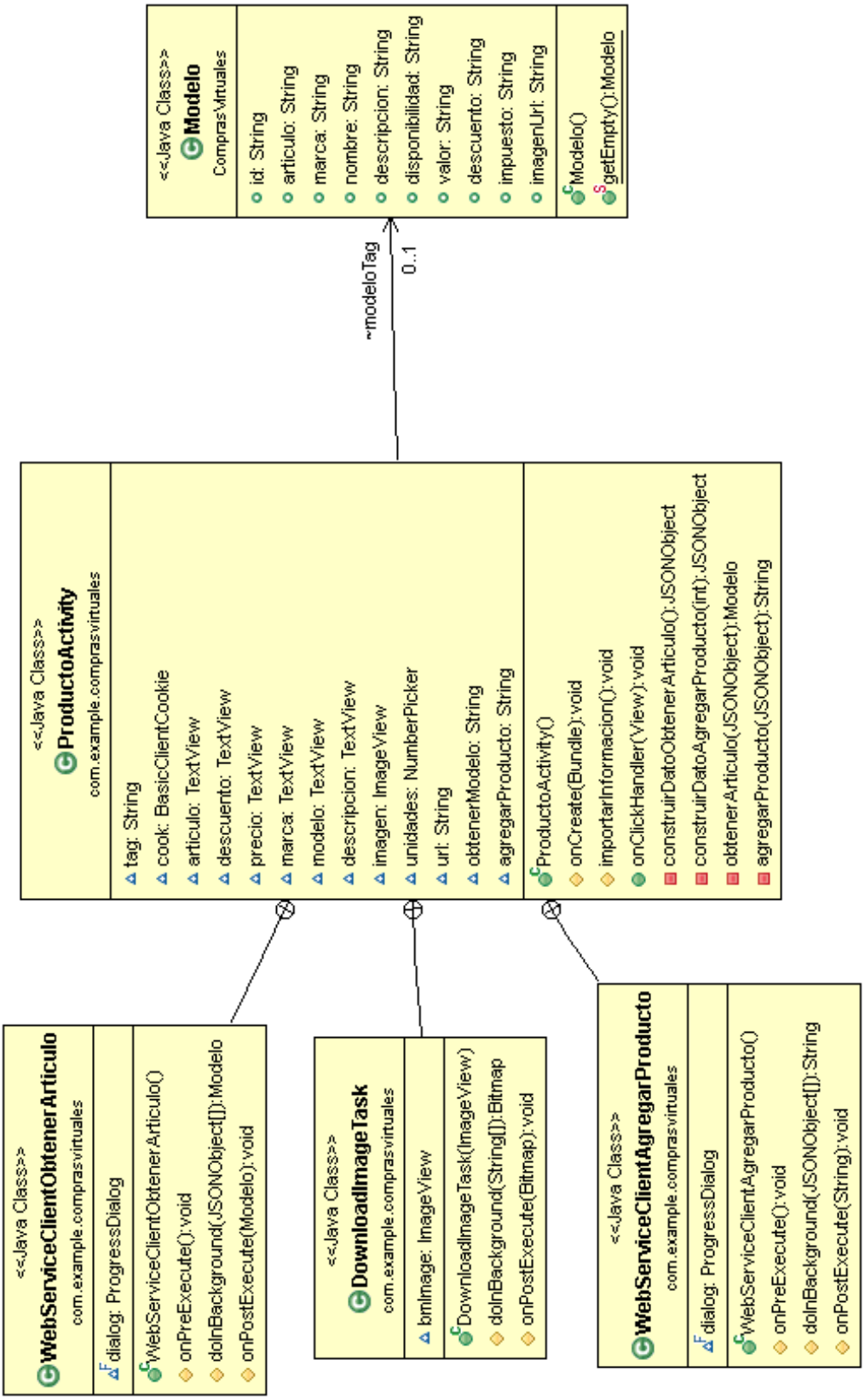

Figura 2.26 Diagrama de clases de la aplicación móvil (tercera parte)

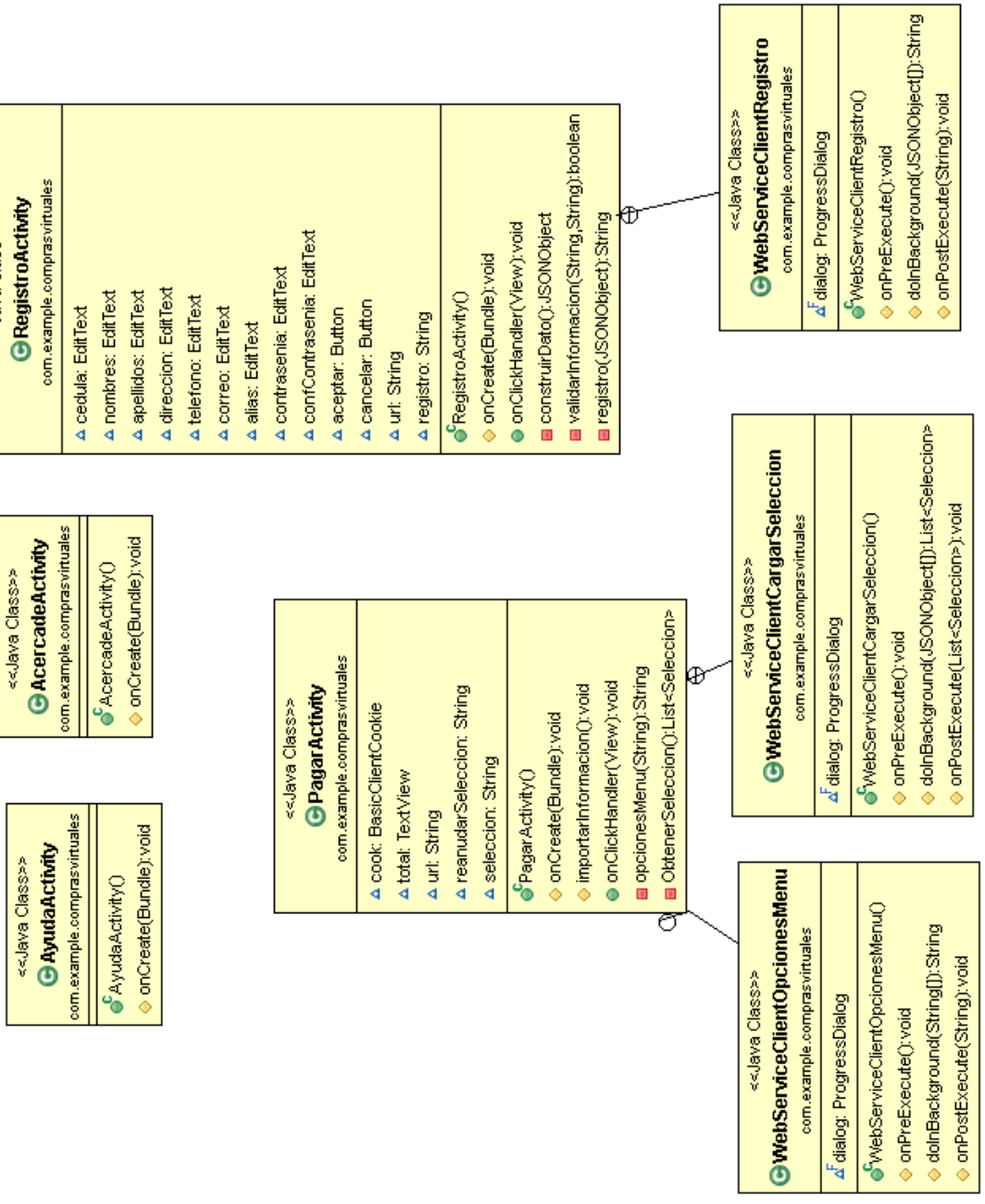

Figura 2.27 Diagrama de clases de la aplicación móvil (cuarta parte)

VARIABLES	
VARIABLE	**DESCRIPCIÓN**
`url`	Variable para recibir el URL del Servicio Web desde el archivo de cadenas de conexión de la aplicación.
`agregarArticulo`	Variables del tipo `String` que reciben partes específicas de los URL de los métodos del Servicio Web que se requiere utilizar.
`eliminarArticulos`	
`eliminarSeleccion`	
`finalizarSeleccion`	
`obtenerSeleccion`	
`finalizarCompra`	
`mNfcAdapter`	Variable que representa el adaptador local NFC.
`mFilters`	Variable para definir `IntentFilters` que sirven para definir la acción de descubrir un nuevo tag NFC.
`mTechList`	Arreglo para almacenar las tecnologías de tarjetas de proximidad compatibles con la aplicación.
`TAG`	Variable para definir el tag a leer.
`total`	Variable utilizada para definir el control `TextView` utilizado para mostrar en la actividad la suma de los precios de los artículos adquiridos.
`cook`	Variable para definir la *cookie* que se utiliza para mantener una sesión con el servidor.
MÉTODOS	
METODO	**DESCRIPCIÓN**
`onCreate`	Método que se ejecuta al momento de crear la actividad, asocia el `layout` en la actividad e inicializa todas las variables.
`onCreateContextMenu`	Método para crear el `ContextMenu`, para poder modificar cada ítem de la lista de compras.
`onCreateOptionsMenu`	Método para crear un `OptionsMenu`.
`onClickHandler`	Método para manejar el evento `click` de una actividad.
`onPause`	Método que se ejecuta al momento que la aplicación entra en pausa, en este método se deshabilita el adaptador NFC.
`onResume`	Método que se ejecuta al momento que la aplicación entra en uso después de haber estado en pausa, en este método se habilita el adaptador NFC y se obtiene la selección del usuario.
`onNewIntent`	Método que se ejecuta al provocarse un nuevo `Intent`, en esta aplicación se ejecuta al momento de leer un tag.
`onContextItemSelected`	Método para manejar la selección de ítems en el `ContextMenu`.
`onOptionsItemSelected`	Método para manejar la selección de ítems en el `OptionsMenu`.
`readMaifireMessageToTag`	Método para manejar la lectura del tag.

Tabla 2.43 Variables y métodos de la clase `CarroActivity`

METODOS	
METODO	**DESCRIPCIÓN**
`construirDatoAgregarArticulo`	Método para construir el cuerpo del mensaje en una petición al Servicio Web.
`construirDatoEliminarArticulo`	Método para construir el cuerpo del mensaje en una petición al Servicio Web.
`obtenerSeleccion`	Método que realiza la petición al Servicio Web para obtener la selección de un usuario.
`insertarArticulo`	Método que realiza la petición al Servicio Web para insertar un artículo al carro de compras.
`eliminarArticulo`	Método que realiza la petición al Servicio Web para eliminar un producto correspondiente a un artículo del carro de compras.
`eliminarArticulos`	Método que realiza la petición al Servicio Web para eliminar todos los productos de un mismo artículo del carro de compras
`opcionesMenu`	Método que realiza la petición al Servicio Web para finalizar la selección o limpiar todo el carro de compras y empezar una nueva selección.
`importarInformacion`	Método utilizado para pasar la información de la *cookie* entre distintas actividades.

Tabla 2.44 Variables y métodos de la clase `CarroActivity` (continuación)

La **Tabla 2.45** da una breve explicación de la clase `WebService ClientObtenerProducto`, la cual se utiliza para realizar la llamada de un método del Servicio Web.

VARIABLES	
VARIABLE	**DESCRIPCIÓN**
`dialog`	Variable utilizada para definir el control `ProgressDialog` el cual se utiliza para operaciones que impliquen tiempo de espera hasta terminar su ejecución.
METODOS	
METODO	**DESCRIPCIÓN**
`WebServiceClientObtenerProducto`	Constructor de la clase que inicializa las variables.
`onPreExecute`	Método que define las operaciones a realizarse antes de la ejecución del hilo secundario.
`doInBackground`	Método que define las operaciones a ejecutarse en el hilo secundario.
`onPostExecute`	Método que define las operaciones a ejecutarse después de concluir la ejecución del hilo secundario.

Tabla 2.45 Variables y métodos de la clase `WebServiceClientObtenerProducto`

2.4.3.3.2 Aplicación de escritorio

La aplicación de escritorio, diseñada para computadores que soportan el *framework* de .NET, cumple la función de interfaz de usuario para los cajeros y administradores del local. Esta aplicación principalmente debe implementar el manejo de inventario y la facturación mediante el consumo del Servicio Web.

En el caso de la aplicación de escritorio, se define una clase llamada `Proxy` en la cual se definen todos los métodos necesarios para realizar las peticiones al Servicio Web. Todos los formularios que requieren realizar peticiones al Servicio Web, hacen uso de la clase `Proxy`.

La clase `Proxy` utiliza `CookiedRequestFactory`[67] para realizar las peticiones HTTP al Servicio Web, esta clase realiza el manejo de la *cookie*[68] necesaria para mantener una sesión con el servidor.

La clase `CredencialesUsuario` se utiliza para serializar los datos de alias y *password* en la autenticación.

La **Figura 2.28** muestra el diagrama de clases de la aplicación de escritorio.

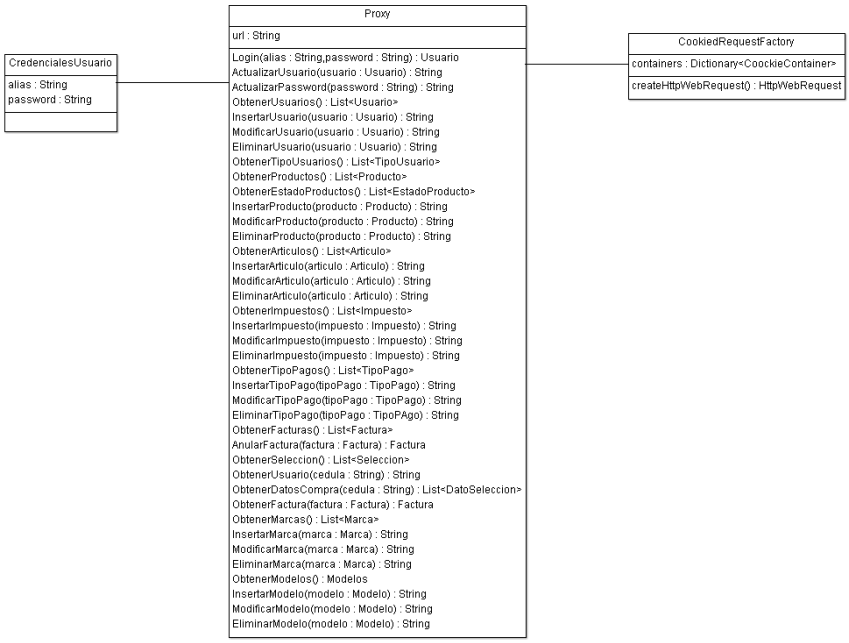

Figura 2.28 Diagrama de clases aplicación de escritorio

[67] **CookiedRequestFactory:** Esta clase se basa en un código obtenido de la referencia bibliográfica [23]
[68] **Cookie:** Información enviada por el servidor para recordar estado e identificar sesiones

2.4.3 DIAGRAMAS DE SECUENCIA

Mediante los diagramas de secuencias se puede entender, de manera más clara, la interacción de los distintos componentes del sistema. En esta sección se presentan los diagramas de secuencia de las principales interacciones, el resto de interacciones describen un comportamiento similar.

Las figuras: **Figura 2.29**, **Figura 2.30** y **Figura 2.31**, muestran los principales diagramas de secuencia del sistema prototipo.

2.5 DISEÑO DE LAS INTERFACES GRÁFICAS EN LAS APLICACIONES

2.5.1 CARACTERISTICAS GENERALES DE LAS INTERFACES GRÁFICAS

Las interfaces gráficas de cada aplicación deben guardar uniformidad en su diseño. Los elementos de cada interfaz deben tener una misma distribución para que sea agradable al usuario, y el mismo, pueda familiarizarse rápidamente con la interfaz. El diseño visual de la aplicación es un diseño sobrio y sencillo, se utilizan colores claros y bajos, siempre guardando uniformidad en toda la aplicación.

2.5.2 INTERFACES GRÁFICAS DE LA APLICACIÓN MÓVIL

Las interfaces gráficas en las aplicaciones Android toman el nombre de actividades, su diseño visual se lo realiza a través de archivos con extensión .XML denominados `layouts`. La codificación de los `layouts` está basado en el uso de etiquetas, similar a la codificación que utiliza HTML[69] (*HyperText Markup Language*). Las interfaces que se implementan en la aplicación móvil se detallan en la **Tabla 2.46**.

[69] **HTML:** Lenguaje de codificación de páginas web.

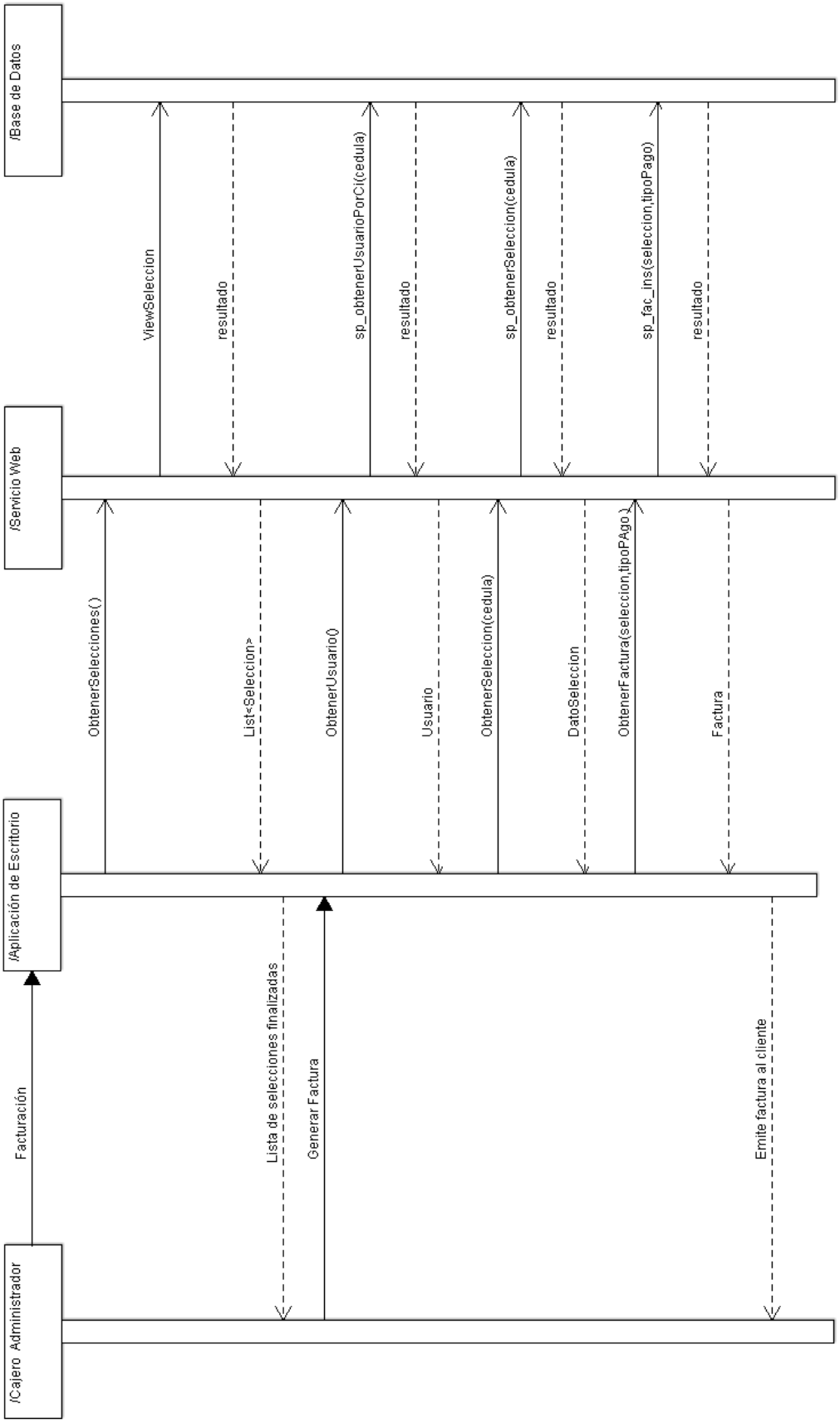

Figura 2.29 Diagrama de secuencia de la Facturación

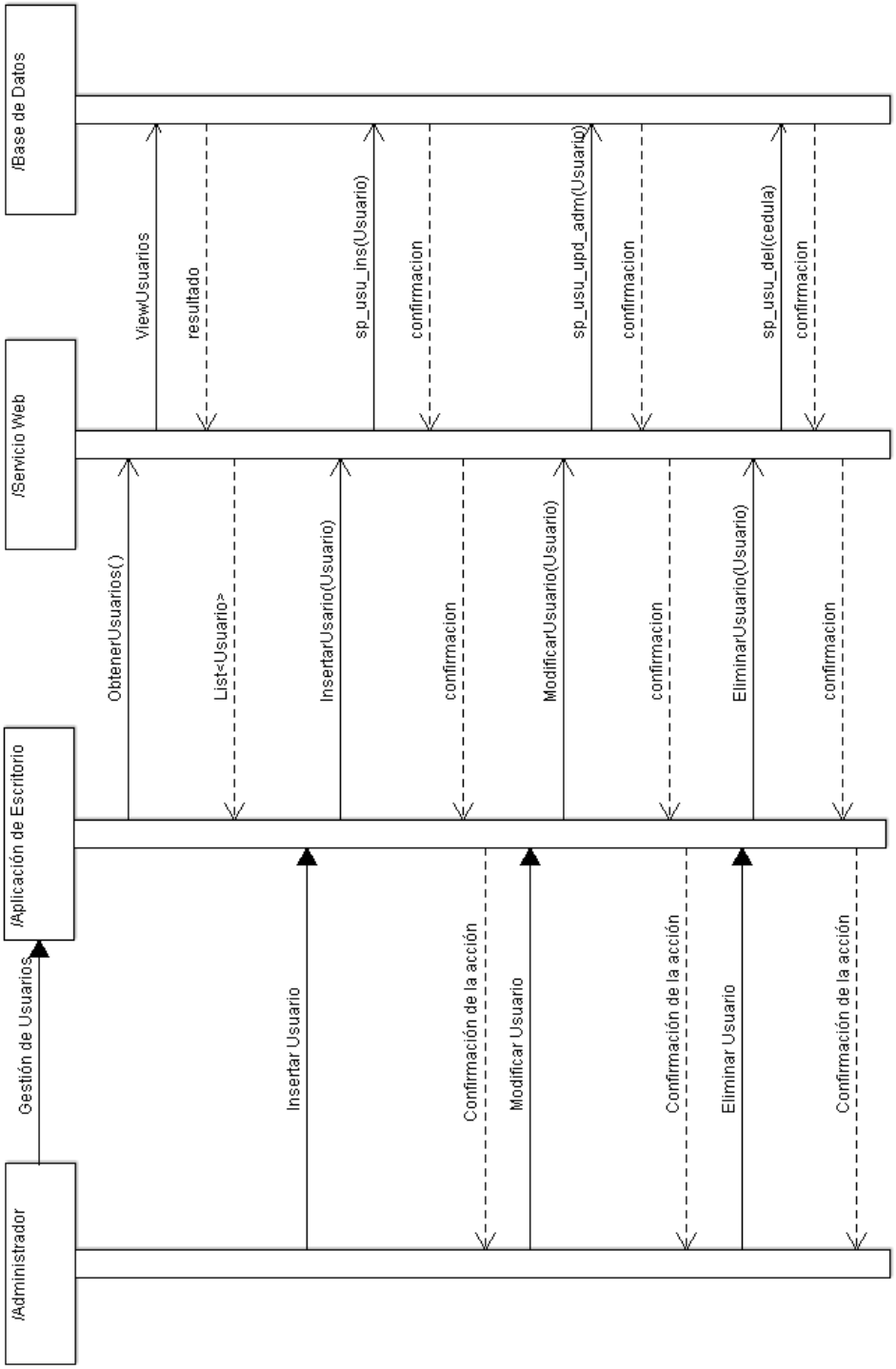

Figura 2.30 Diagrama de secuencia de la gestión de usuarios

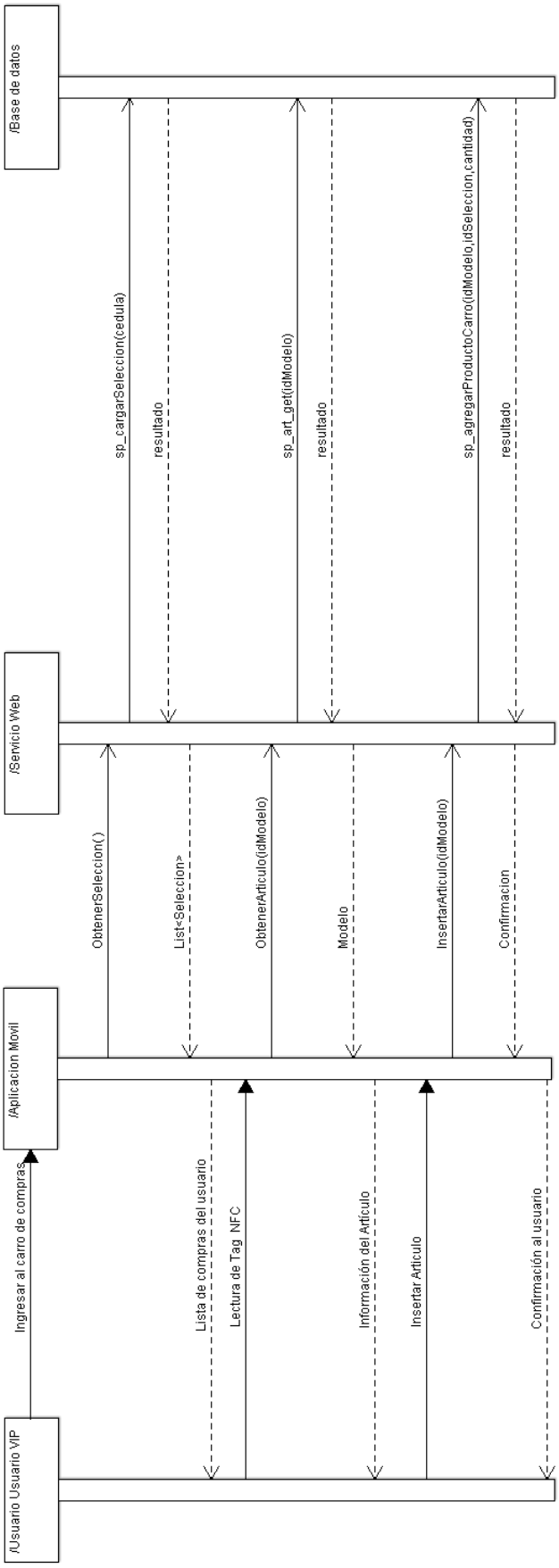

Figura 2.31 Diagrama de secuencia del carro de compras

TÍTULO DE LA ACTIVIDAD	DESCRIPCION
Acerca de	Actividad que muestra información de los desarrolladores
Autenticación	Actividad para permitir la autenticación del usuario en el sistema
Ayuda	Actividad que brinda ayuda al usuario para el uso de la aplicación
Carro	Actividad donde se muestra la lista de productos seleccionados en el carro de compras virtual
Configurar	Actividad donde se modifican los datos registrados en la cuenta del usuario
Contraseña	Actividad para modificar la contraseña del usuario
Menú	Actividad que se muestra después de la autenticación, esta actividad permite al usuario seleccionar la actividad a la cual desea ingresar
Producto	Actividad que se muestra después de la lectura del tag NFC para mostrar información detallada del producto
Registro	Actividad para permitir a usuarios no registrados registrarse en el sistema
Reporte	Actividad que se muestra al finalizar la selección de productos en el carro de compras, muestra un resumen de valores a cancelar y productos seleccionados

Tabla 2.46 Actividades de la aplicación móvil

En el diseño de la actividad inicial, se contempla la inclusión del logotipo de la aplicación. Este logotipo guarda relación con la temática de la aplicación para que el usuario pueda asociar la aplicación con la finalidad de uso. La **Figura 2.32** muestra una captura de pantalla de la actividad Autenticación.

Figura 2.32 Captura de pantalla de la actividad Autenticación

En los controles `EditText` se utiliza la propiedad `hint` para indicarle al usuario que debe ingresar en esos controles. El resto de actividades guardan las mismas características generales como: diseño, color, características de los controles, etc. La **Figura 2.33** muestra una captura de pantalla de la actividad Carro para poder comparar las características generales de la actividad.

Figura 2.33 Captura de pantalla de la actividad Carro

Un aspecto importante en el diseño de las actividades es la limitación de espacio. Se debe tratar de usar la menor cantidad de controles y distribuirlos correctamente en la pantalla.

2.5.3 INTERFACES GRÁFICAS DE LA APLICACIÓN DE ESCRITORIO

Las interfaces gráficas en las aplicaciones .NET toman el nombre de formularios, su diseño visual se lo puede realizar con la ayuda del IDE que proporciona Visual Studio. Las interfaces que se implementan en la aplicación de escritorio se detallan en la **Tabla 2.47** y la **Tabla 2.48**.

FORMULARIO	DESCRIPCIÓN
`frmArticulo`	Formulario principal del componente gestión de artículos
`frmAutenticación`	Formulario para permitir la autenticación del usuario al sistema
`frmCambiarContrasenia`	Formulario para cambiar la contraseña del usuario

Tabla 2.47 Formularios de la aplicación de escritorio

FORMULARIO	DESCRIPCIÓN
frmDatosUsuario	Formulario para modificar los datos de la cuenta del usuario
frmFactura	Formulario para visualizar la factura emitida
frmHistorialFactura	Formulario para visualizar facturas anteriores
frmImpuesto	Formulario principal del componente gestión de Impuestos
frmMarca	Formulario principal del componente de gestión de Marcas
frmModelo	Formulario principal del componente gestión de Modelos
frmNArticulo	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de artículos
frmNImpuesto	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de impuestos
frmNMarca	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de marcas
frmNModelo	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de modelos
frmNProducto	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de productos
frmNTipoPago	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de tipo de pago
frmNUsuario	Formulario utilizado para insertar, modificar y eliminar en el componente gestión de Usuarios
frmPreFactura	Formulario para edición de los datos de la factura
frmPrincipal	Formulario padre de la aplicación
frmProducto	Formulario principal del componente gestión de productos
frmTipoPago	Formulario principal del componente gestión de tipo de pago
frmUsuario	Formulario principal del componente gestión de Usuarios

Tabla 2.48 Formularios de la aplicación de escritorio (continuación)

En la aplicación de escritorio, al cumplir la función de gestión de inventario, se crean componentes para los principales elementos del sistema como son: artículos, productos, usuarios, etc. Cada componente tiene dos formularios: el primer formulario incluye una grilla para mostrar todos los registros de ese elemento; el segundo formulario se lo utiliza para las actividades de insertar, modificar y eliminar, se utiliza el mismo formulario para las tres actividades.

En el caso de la aplicación de escritorio los formularios brindan mayor espacio, en este caso los controles, como las grillas, deben ocupar la mayor cantidad de espacio. La **Figura 2.34** muestra una captura de pantalla de un formulario para

tener una mejor idea de cómo se ubican los controles en las interfaces graficas de la aplicación de escritorio.

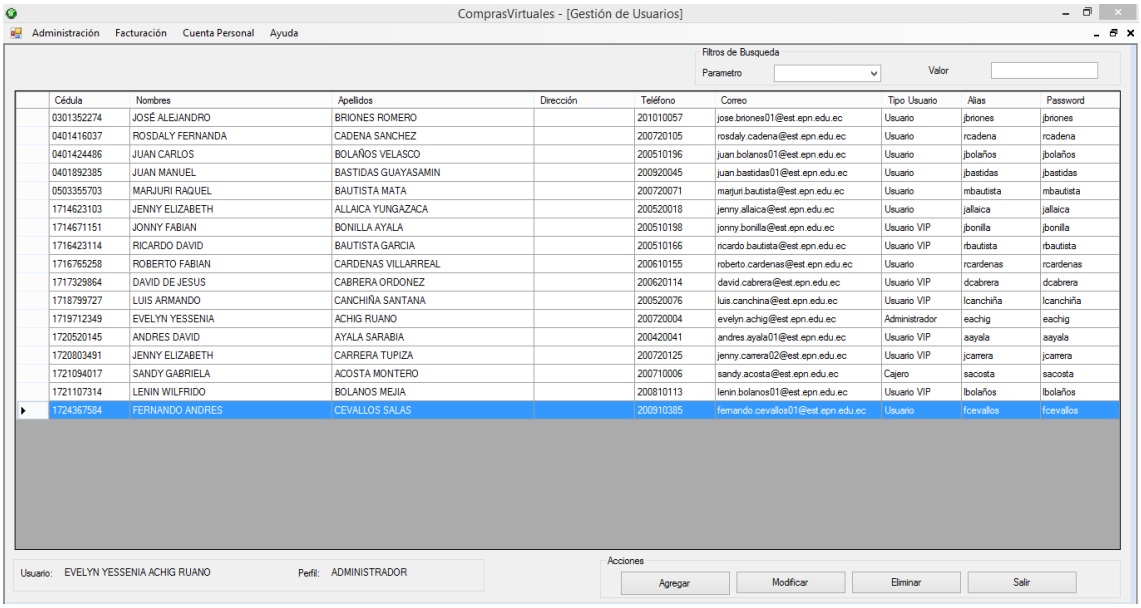

Figura 2.34 Captura de pantalla del formulario `frmUsuario`

CAPÍTULO 3. DESARROLLO

3.1 INTRODUCCIÓN

En el presente capítulo se explica el desarrollo de todos los componentes del sistema, tomando en cuenta los diseños presentados en el capítulo anterior.

3.2 CODIFICACIÓN DEL SISTEMA PROTOTIPO

El desarrollo del sistema prototipo se realizó con un equipo de dos personas, por lo que fue necesario establecer ciertas convenciones en la codificación, con el fin de que cualquier miembro del equipo pueda entender el código.

De manera general, los nombres de clases, métodos, archivos, entre otros, tratan de ser lo más descriptivo posible.

3.2.1 NORMAS DE CODIFICACIÓN DE LA BASE DE DATOS

3.2.1.1 Tablas

Las tablas y sus campos siguen las siguientes convenciones:

- El nombre de la tabla debe indicar de forma clara el tipo de información que almacena; por ejemplo, si la tabla almacena información de usuarios, la tabla correspondiente se llamará Usuario.
- Los atributos comenzarán con las tres primeras letras del nombre de la tabla, seguido del nombre del atributo.
- La clave principal de cada tabla tendrá el nombre de ID.
- Las claves foráneas en una tabla se nombrarán haciendo referencia a la tabla de donde vienen, seguido de las letras ID.

3.2.1.2 Procedimientos Almacenados, Triggers, Vistas

Estos elementos se almacenan en archivos con extensión .SQL y su nombre de archivo se realiza considerando que comenzará con las siglas sp, seguido de un nombre descriptivo.

Para la codificación de estos elementos se seguirán las convenciones y recursos del leguaje declarativo para acceso de datos SQL.

3.2.2 NORMAS DE CODIFICACIÓN DE LAS CLASES

Las clases y sus elementos tienen las siguientes convenciones:

- El nombre de la clase debe ser lo más descriptivo posible, puede estar conformado por uno o varios sustantivos singulares, usando la notación *PascalCase*[70].
- Los atributos de la clase se nombran de la misma manera que las clases, usando la notación *camelCase*[71].

3.3 DESARROLLO DE LA BASE DE DATOS

La base de datos desarrollada, trabaja con el motor de bases de datos del sistema SQL Server. Los componentes desarrollados para la base de datos son: tablas, procedimientos almacenados, *triggers* y vistas.

3.3.1 PROCEDIMIENTOS ALMACENADOS

Realizan el procesamiento de la información en la base de datos, la ventaja principal de utilizar procedimientos almacenados es: reducir la complejidad en la codificación de los otros componentes del sistema y a la vez reducir el procesamiento en los otros componentes del sistema. En la gran mayoría de métodos en el Servicio Web definido, se invocan a estos procedimientos almacenados para realizar las tareas requeridas en la base de datos.

3.3.1.1 Ejemplo de un procedimiento almacenado

El procedimiento almacenado que se presenta a manera de ejemplo es el `sp_datosSeleccion`, este procedimiento almacenado obtiene los datos necesarios para presentar la factura, estos son: subtotal, impuestos, descuentos. Lo primero que realiza el procedimiento almacenado es identificar el tipo de

[70] **PascalCase:** En esta notación se escribe la primera letra de cada palabra en mayúscula y sin espacios.
[71] **camelCase:** En esta notación se escribe la primera letra de cada palabra en mayúscula a excepción de la primera palabra.

usuario mediante una sentencia `if`, luego mediante consultas se obtienen los datos y se realizan las operaciones necesarias para obtener los valores de la factura. El **Código 3.1** muestra el código de este procedimiento.

```
-- Comando para Acceder a la Base Compras Virtuales
USE [ComprasVirtuales]
GO

/* Procedimiento almacenado que consulta el Subtotal, Impuestos, Descuentos, de la
 selección de un usuario determinado */
create Procedure [dbo].[sp_datosSeleccion]
(@seleccion int, @cedula char(10))
As
Begin
-- Condición que identifica que el usuario es un usuario Normal
IF EXISTS ( Select * from Usuario where usu_ci=@cedula and usu_tipo_id=3 )
                    BEGIN
                    select SUM(mod_valor) + SUM(mod_valor)*imp_porcentaje/100 -
                    SUM(mod_valor)*mod_descuento as 'Costo', SUM(mod_valor) as 'Precio',
                    SUM(mod_valor)*imp_porcentaje/100 as 'Impuesto',
SUM(mod_valor)*mod_descuento as 'Descuento'
                    from Articulo, Marca, Modelo, Producto, Producto_seleccion,
Seleccion, Impuesto
                    where mod_id=prod_mod_id and prod_id=ps_prod_id and ps_sel_id=sel_id
                    and sel_id=@seleccion and mod_impuesto= imp_id and mar_id= mod_mar_id
                    and art_id = mod_art_id
                    group by art_nombre, mar_nombre, mod_nombre, mod_imagenUrl,
                    mod_descuento, imp_porcentaje, art_id
                    END
-- Condición que identifica que el usuario es un usuario VIP
ELSE
                    BEGIN
                    select SUM(mod_valor) + SUM(mod_valor)*imp_porcentaje/100 -
                    SUM(mod_valor)*mod_descuentoVIP as 'Costo', SUM(mod_valor) as
'Precio',
                    SUM(mod_valor)*imp_porcentaje/100 as 'Impuesto',
SUM(mod_valor)*mod_descuentoVIP as 'Descuento'
                    from Articulo, Modelo, Marca, Producto, Producto_seleccion,
Seleccion, Impuesto
                    where mod_id=prod_mod_id and prod_id=ps_prod_id and ps_sel_id=sel_id
                    and sel_id=@seleccion and mod_impuesto= imp_id
                    group by art_nombre, mar_nombre, mod_nombre, mod_imagenUrl,
                     mod_descuentoVIP, imp_porcentaje, art_id
                    END
END
```

Código 3.1 Procedimiento almacenado `sp_datosSeleccion`

El procedimiento almacenado explicado en esta sección permite tener una idea general de cómo se implementan los mismos, el resto de procedimientos almacenados se implementan de manera similar al ejemplo expuesto, manteniendo las particularidades que se presentan en cada caso. En el **Anexo A.1** se incluyen los procedimientos almacenados elaborados, debidamente comentados para una fácil comprensión.

3.3.2 TRIGGERS

Los *triggers* ejecutan operaciones ante eventos que ocurren en la base de datos, tales como: inserción, modificación, eliminación de registros, entre otros. Estas operaciones se ejecutan para mantener consistencia en los registros que almacena la base de datos.

3.3.2.1 Ejemplo de un trigger

El *trigger* que se presenta a manera de ejemplo es el `tg_actualizar_disponibilidad` el cual se ejecuta al momento en que se agrega un producto al carro de compras. El *trigger* actualiza el campo `prod_estp_id` para indicar que el producto está reservado. El **Código 3.2** muestra el código de este procedimiento.

```
-- Comando para Acceder a la Base Compras Virtuales
USE [ComprasVirtuales]
GO

-- Trigger que controla que un producto pase a reservado(1) despues de haberlo agregado a
Producto_seleccion
CREATE TRIGGER [dbo].[TG_actualizar_disponibilidad]
 ON [dbo].[Producto_seleccion]
 AFTER INSERT
 AS
       BEGIN
              -- Se actualiza el estado del producto que fue afectado por la inserción del
nuevo registro en Producto_seleccion
              UPDATE Producto
              set prod_estp_id=1 where prod_id in (select ps_prod_id from inserted)
       END

-- Comando para habilitar los triggers en la tabla Producto_seleccion
  ALTER TABLE Producto_seleccion ENABLE TRIGGER ALL
```

Código 3.2 *Trigger* `tg_actualizar_disponibilidad`

El *trigger* explicado en esta sección permite tener una idea general de cómo se implementan los mismos, el resto de *triggers* se implementan de manera similar al ejemplo expuesto, manteniendo las particularidades que se presentan en cada caso. En el **Anexo A.2** se incluyen los *triggers* elaborados, debidamente comentados para una fácil comprensión.

3.3.3 VISTAS

Las vistas son consultas realizadas a las tablas de las bases de datos. Se puede decir que las vistas son tablas virtuales. En el presente proyecto, las vistas se utilizan para obtener los registros que se van a mostrar a los usuarios.

3.3.3.1 Ejemplo de una vista

La vista que se presenta a manera de ejemplo es la `ViewFactura`, esta vista se realiza a partir de una consulta que obtiene los campos de la tabla que se van a mostrar al cliente.

```
/* Creación de la Vista ViewFactura con el fin de mostrar los datos necesarios
para generar una factura*/
create view ViewFactura
as

SELECT                      Factura.fac_id as 'ID',
                            Factura.fac_sel_id as 'Seleccion',
                            Factura.fac_subtotal as 'Subtotal',
                            Factura.fac_impuestos as 'Impuestos',
                            Factura.fac_descuento as 'Descuentos',
                            Factura.fac_total as 'Total',
                            Factura.fac_clienteEmitida as 'Cedula',
                            Tipo_pago.tip_nombre as 'TipoPago',
                            Factura.fac_fecha as 'Fecha',
                            Usuario.usu_nombres+' '+ Usuario.usu_apellidos as 'Nombre'
from  Usuario, Factura, Tipo_pago
where Factura.fac_clienteEmitida= Usuario.usu_ci and Factura.fac_tip_id= Tipo_pago.tip_id
and Factura.fac_sel_id!=0 and (Factura.fac_descripcion!='Eliminado' or
Factura.fac_descripcion IS NULL)

GO
```

Código 3.3 Vista `ViewFactura`

La vista explicada en esta sección permite tener una idea general de cómo se implementan las mismas, el resto de vistas se implementan de manera similar al ejemplo expuesto. En el **Anexo A.3** se incluyen las vistas elaboradas, debidamente comentadas para una fácil comprensión.

3.4 DESARROLLO DEL SERVICIO WEB

La arquitectura utilizada para implementar el Servicio Web en este sistema es la arquitectura REST. En esta arquitectura no se guardan estados entre el cliente y el servidor. La forma de invocar el Servicio Web es mediante peticiones POST

de HTTP, con el fin de ocultar en el cuerpo del mensaje los datos que se envían al servidor.

Para representar los datos en las peticiones se emplea el formato JSON. Este formato es bastante ligero ya que serializa los datos mediante pares nombre/valor.

Para la implementación del Servicio Web, que en el presente proyecto se denomina `ServicioComprasVirtuales` se definen los siguientes elementos:

- `IServicioComprasVirtuales`: es una interfaz que corresponde al contrato del servicio RESTful.
- `ServicioComprasVirtuales`: es una clase que implementa los métodos de la interfaz `IServicioComprasVirtuales`.

El servidor está desarrollado en Visual Studio 2013 empleando el lenguaje de programación C#. La principal función del Servicio Web es proveer los servicios necesarios para que las aplicaciones cliente funcionen correctamente.

3.4.1 FORMATO DE COMUNICACIÓN CON LOS CLIENTES

Para establecer JSON como formato de comunicación en el Servicio Web, cada método declarado en el contrato del Servicio Web debe tener el atributo `WebInvoke` y definir las siguientes propiedades:

- `Method`: Define el tipo de operación HTTP a utilizar.
- `BodyStyle`: Define el estilo del cuerpo del mensaje, que se usa en la operación del servicio.
- `RequestFormat`: Formato de datos del mensaje de petición.
- `RensponseFormat`: Formato de datos del mensaje de respuesta.
- `UriTemplate`: Define el formato del URI para invocar el método.

En el **Código 3.4** se presentan las líneas de código que se incluyen antes de la definición del método en la interfaz para establecer las propiedades mencionadas anteriormente.

```
[OperationContract(Name = "autenticacion")]
[WebInvoke(Method = "POST",
           BodyStyle = WebMessageBodyStyle.WrappedRequest,
           RequestFormat = WebMessageFormat.Json,
           ResponseFormat = WebMessageFormat.Json,
           UriTemplate = "postmethod/autenticacion")]
 //metodo que realiza la autenticacion del usuario en la aplicacion de escritorio
 ComprasVirtuales.Usuario Autenticacion(string alias, string password);
```

Código 3.4 Declaración de un método del servicio REST

Para que el Servicio Web emplee la arquitectura REST y el formato JSON se debe modificar el archivo de configuración `Web.config`[72]. En el **Código 3.5**, las líneas subrayadas muestran la parte del archivo de configuración que se debe modificar para poder emplear la arquitectura REST y el formato JSON. Para permitir que múltiples clientes consuman el Servicios Web se debe habilitar la opción `multipleSiteBindingsEnabled` en el archivo de configuración del Servicio Web.

```
<system.serviceModel>
    <behaviors>
      <serviceBehaviors>
        <behavior>
          <!-- Para evitar revelar información de los metadatos, establezca los
valores siguientes en false antes de la implementación -->
          <serviceMetadata httpGetEnabled="true"/>
          <!-- Para recibir detalles de las excepciones en los fallos, con el fin
de poder realizar la depuración, establezca el valor siguiente en true. Para no
revelar información sobre las excepciones, establézcalo en false antes de la
implementación -->
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
      <endpointBehaviors>
        <behavior>
          <webHttp helpEnabled="true" defaultOutgoingResponseFormat="Json"/>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <protocolMapping>
      <add scheme="http" binding="webHttpBinding"/>
    </protocolMapping>
    <serviceHostingEnvironment aspNetCompatibilityEnabled="true"
                               multipleSiteBindingsEnabled="true" />
  </system.serviceModel>
```

Código 3.5 Parte del archivo `Web.config`

[72] **Web.config:** Archivo principal para establecer opciones de configuración en una aplicación web.

3.4.2 COMUNICACIÓN DEL SERVICIO WEB CON LA BASE DE DATOS [14]

Para la comunicación del Servicio Web con la base datos, se utiliza un componente del *framework* de .NET llamado LINQ to SQL, el cual es una herramienta que proporciona una infraestructura en tiempo de ejecución para administrar los datos relacionales.

La herramienta LINQ to SQL crea un modelo de objetos a partir de una base de datos existente. La **Figura 3.1** muestra la relación entre el programador y los datos, haciendo uso de la herramienta LINQ to SQL.

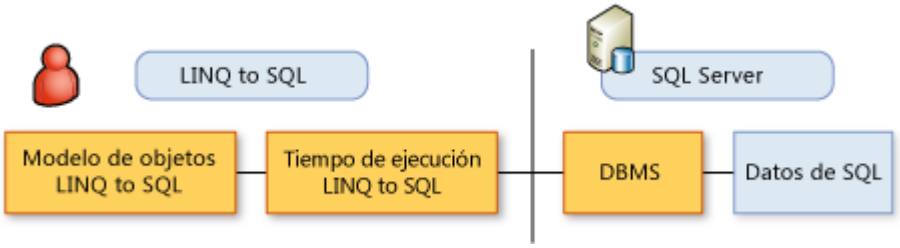

Figura 3.1 Relación entre el programador y los datos cuando se usa el componente LINQ to SQL [14]

3.4.3 MANEJO DE SESIONES [14] [23]

REST al basarse en un cliente/servidor sin estado, requiere que se implemente algún mecanismo para mantener sesiones entre los clientes y el servidor. Para resolver este inconveniente se hace uso de *cookies*.

3.4.3.1 Clase HttpContext

Para implementar el uso de *cookies*, en el Servicio Web se utiliza la clase `HttpContext`. La clase `HttpContext` encapsula la información necesaria relativa a una petición HTTP individual.

En el constructor del Servicio Web se asigna el objeto `HttpContext` de la petición HTTP actual, utilizando la propiedad `current` de la clase. De esta manera se obtiene la sesión del usuario de la petición HTTP actual. El **Código 3.6** muestra las líneas de código para la asignación del objeto `HttpContext`.

```
public ServicioComprasVirtuales()
{
    //Se inicializa un contexto para poder almacenar las
    //sesiones de los usuarios
    context = HttpContext.Current;
}
```

Código 3.6 Constructor del Servicio Web

3.4.3.2 Almacenamiento del estado en el objeto HttpContext

Luego de que se obtiene la sesión actual, se deberá en el primer método utilizado por el cliente, guardar la información requerida para recordarle al servidor que esa sesión esta activa y la información necesaria para obtener los datos del usuario de esa sesión.

En el caso de este proyecto el método que guardará esta información de la sesión será el método `Autenticacion`. Si el método recibe la información correcta de alias y *password* del usuario, se guarda el estado para permitir usar los demás métodos del servicio. Si el usuario no se autentica correctamente, la sesión le negará utilizar los demás métodos del Servicio Web. El **Código 3.7** muestra el código desarrollado para el método de autenticación.

```
public int Autenticacion(string alias, string password)
        {
            //variable auxiliar que identifica el permiso de acceso de un usuario
            int acceso = 5;
            //variable auxiliar que permite identificar si el usuario puede o no
            //ingresar
            //a la aplicación.
            bool comprobar = false;
            //variable para almacenar la cedula del usuario
            string cedula = "";

            //ejecuta el metodo de acceso en la base de datos
            var resultado = data.sp_login(alias, password);

            //transforma el valor recibido de tipo string a entero
            acceso = int.Parse(resultado.ReturnValue.ToString());

            //si el numero devuelto por la base de datos es 3 o 4 se permite el
            //acceso a la aplicacion
            if (acceso.Equals(3) || acceso.Equals(4))
            {

                comprobar = true;
                // se envuelve la peticion en un lazo foreach porque siempre la
                //base de datos
                //va a devolver un arreglo
                foreach (var ci in resultado)
                {
                    cedula = ci.usu_ci;
                }
            }
            //mantiene el estado de autenticado en la sesion del usuario
            context.Session["USERLOGGEDIN"] = comprobar ? "YES" : null;
            context.Session["USUARIO"] = comprobar ? cedula.ToString() : null;
            context.Session["TIPOUSUARIO"] = comprobar ? acceso.ToString() :
            null;
            return acceso;
        }
```

Código 3.7 Método `Autenticacion` definido en el Servicio Web

Como se puede observar en el **Código 3.7**, las líneas resaltadas son las que guardan el estado de autenticación en la sesión. En caso de que el usuario este autorizado a ingresar al sistema la variable `comprobar` que es del tipo `boolean` se asigna en `true`. Al final del método con el operador `?:`[73] se comprueba que el usuario este autorizado para ingresar y se guarda la información en la sesión correspondiente.

3.4.3.3 Comprobación de estado en los métodos del Servicio Web

Los métodos del Servicio Web deberán comenzar con líneas de código que comprueben que el usuario está autenticado. El **Código 3.8** muestra el método `obtenerUsuario` como ejemplo de esta comprobación; los datos del usuario son entregados al cliente solo si el usuario que los pide esta autenticado, si no está autenticado, el método interrumpe su ejecución. Las líneas subrayadas corresponden al bloque de código que realiza esta comprobación.

```
public ComprasVirtuales.Usuario ObtenerUsuario()
        {
            //arreglo para almacenar los datos devueltos por la base de datos
            List<ComprasVirtuales.Usuario> lista = new
            List<ComprasVirtuales.Usuario>();

            //este condicional interrumpe la ejecucion del metodo si el usuario
            //no esta autenticado
            if ((string)context.Session["USERLOGGEDIN"] != "YES")
            {
                return new ComprasVirtuales.Usuario();
            }

            //si el usuario esta autenticado obtiene la cedula del usuario
            //guardado en la sesion
            string cedula = (string)context.Session["USUARIO"];

            //ejecuta el procedimiento almacenado de la base que devuelve un
            //usuario por su cedula
            var resultado = data.sp_obtenerUsuarioPorCi(cedula);

            //como se menciona anteriormente la base de datos siempre va a
            //devolver un arreglo de
            //elementos por lo que se debe envolver a la instruccion en un lazo
            //foreach
            foreach (var usuario in resultado)
            {
                lista.Add(new ComprasVirtuales.Usuario(usuario.Cedula,
                    usuario.Nombres,
                    usuario.Apellidos,
                    usuario.Direccion,
                    usuario.Telefono,
                    usuario.Correo,
                    usuario.TipoUsuario,
                    usuario.Alias,
                    ""));
            }

            //obtiene el primer elemento de la lista en este caso el unico que
            //contiene la lista
            ComprasVirtuales.Usuario usu = lista[0];
            return usu;
        }
```

Código 3.8 Método `ObtenerUsuario` definido en el Servicio Web

[73] `?:` Operador condicional que devuelve uno de dos valores según el valor de la expresión booleana.

3.4.4 EJEMPLO DE UN MÉTODO DEL SERVICIO WEB

Se presenta a manera de ejemplo el método `Registro`. Para implementar un método en el Servicio Web se debe definir el método en el contrato del Servicio Web. Antes de la firma[74] del método se debe especificar el atributo `[OperationContract]` para indicar que el método será publicado por el servicio REST. El **Código 3.9** muestra las líneas de código que deben estar presentes para definir un método del Servicio Web en la interfaz.

```
//indica que el metodo define una operacion que es parte del servicio
        [OperationContract(Name = "registro")]
        //indica el metodo utilizado en la comunicacion
        [WebInvoke(Method = "POST",
                   //especifica si se ajusta los parametros y valores devueltos
                   BodyStyle = WebMessageBodyStyle.WrappedRequest,
                   //indica el formato del mensaje peticion
                   RequestFormat = WebMessageFormat.Json,
                   //indica el formato del mensaje respuesta
                   ResponseFormat = WebMessageFormat.Json,
                   //define el URI del servicio
                   UriTemplate = "postmethod/registro")]
        //metodo que registra a nuevos usuarios mediante la aplicacion de android
        string Registro(string cedula,
                        string apellidos,
                        string nombres,
                        string direccion,
                        string telefono,
                        string email,
                        string tipoUsuario,
                        string alias,
                        string password);
```

Código 3.9 Definición de un método del Servicio Web en la interfaz

Posteriormente a definir el método en la interfaz se debe implementar el método en la clase del Servicio Web. Para el caso del método `Registro` que se presenta como ejemplo, el método utiliza el modelo de objetos que se construyó con la herramienta LINQ to SQL para realizar la llamada al procedimiento almacenado de la base de datos. Los datos que retornan los procedimientos almacenados, vistas, tablas, entre otras, deben ser almacenados en una variable del tipo `var` ya que en tiempo de ejecución no se conoce que tipo de datos se

[74] **Firma:** Es la sentencia que se emplea para la declaración de un método, incluye valor de retorno, nombre del método y parámetros de entrada.

retornan de la base de datos. El **Código 3.10** muestra el método `Registro` del Servicio Web.

```
public string Registro(string cedula,
                       string apellidos,
                       string nombres,
                       string direccion,
                       string telefono,
                       string email,
                       string tipoUsuario,
                       string alias,
                       string password)
        {

            try
            {
                //el tipo de dato var es un tipo de dato implicito
                //es decir el tipo de dato es asignado en tiempo de ejecucion
                //mediante el objeto data se puede hacer llamada a los componentes de una
                //base de datos
                //tal como procedimientos almacenados
                var resultado = data.SP_usu_ins(cedula,
                                                nombres,
                                                apellidos,
                                                direccion,
                                                telefono,
                                                email,
                                                int.Parse(tipoUsuario),
                                                alias,
                                                password);
                //si el metodo se ejecuta correctamente devuelve un mensaje positivo
                return "RegistroCorrecto";
            }
            //en caso de que ocurra una excepcion se captura y se devuelve el mensaje de
            //error
            catch (Exception e)
            {
                string mensaje = e.Message.ToString();
                return mensaje;
            }
        }
```

Código 3.10 Método `Registro` definido en el Servicio Web

El método explicado en esta sección permite tener una idea general de cómo se implementan los mismos, el resto de métodos se implementan de manera similar al ejemplo expuesto ya que los métodos ejecutan uno o varios procedimientos almacenados definidos en la base de datos. El Servicio Web elaborado se presenta de forma completa y comentada en el **Anexo B.1**.

3.5 DESARROLLO DE LA APLICACIÓN DE ESCRITORIO [23]

La aplicación de escritorio es una de las aplicaciones clientes que consumen el Servicio Web, esta aplicación está desarrollada en Visual Studio 2013 empleando el lenguaje de programación C#. La función de esta aplicación es

brindar una interfaz gráfica amigable para las tareas que ejecutan los usuarios Cajero y Administrador.

En las siguientes secciones se detalla el desarrollo y codificación de ciertas partes de la aplicación que se consideran importantes, el código fuente completo y comentado se encuentra en el **Anexo C.1**.

3.5.1 MANEJO DE SESIONES

Para manejar las sesiones en la aplicación de escritorio de forma transparente, se utiliza la clase `CookiedRequestFactory`, clase obtenida en [23]. Esta clase permite realizar el manejo de las *cookies* para mantener sesiones con el servidor. El **Código 3.4** muestra el código de la clase `CookiedRequestFactory`.

```
public class CookiedRequestFactory
    {

        //este diccionario mantiene todos los cookie containers
        //para cada dominio
        private static Dictionary<string, CookieContainer> containers
            = new Dictionary<string, CookieContainer>();

        public static HttpWebRequest CreateHttpWebRequest(string url)
        {
            // Crea un objeto HttpWebRequest
            var request = (HttpWebRequest)WebRequest.Create(url);

            // toma la parte del dominio del URL
            string domain = (new Uri(url)).GetLeftPart(UriPartial.Authority);

            //trata de tomar un contenedor del diccionario, si esta en el
            //diccionario, lo usa. Caso contrario, crea uno nuevo y lo pone
            //en el diccionario y luego lo usa
            CookieContainer container;
            if (!containers.TryGetValue(domain, out container))
            {
                container = new CookieContainer();
                containers[domain] = container;
            }

            //asigna el cookie container al objeto HttpWebRequest
            request.CookieContainer = container;

            return request;
        }
    }
```

Código 3.11 Clase `CookiedRequestFactory` [23]

El método `CreateHttpWebRequest` del **Código 3.11**, se utiliza para crear un objeto `HttpWebRequest` que realiza las peticiones HTTP al servidor y además realiza el manejo de las *cookies* para mantener sesión con el servidor.

La variable `containers` es un diccionario que almacena un `string` y un `cookieContainer`[75] para cada dominio[76].

Con el uso de esta clase se facilita la programación, ya que todas las peticiones al Servicio Web, se realizan a través del método `CreateHttpWebRequest`. Los objetos de esta clase van a ser los encargados de manejar la sesión del cliente con el servidor.

3.5.2 CONSUMO DEL SERVICIO WEB A TRAVÉS DE LA CLASE PROXY

La clase `Proxy` es la encargada de serializar los mensajes, realizar las peticiones HTTP haciendo uso de la clase `CookiedRequestFactory` y crear el flujo de bytes para que la información viaje por la red. Esta clase también se encarga de recibir las respuestas enviadas por el Servicio Web y reconstruir la información en el formato indicado. El **Código 3.12** muestra un extracto de la clase `Proxy` con un método de ejemplo, que permite entender el funcionamiento de esta clase, el resto de métodos tienen la misma lógica, adaptada para cada petición que el cliente realiza al Servicio Web.

Como se observa en el **Código 3.12**, para serializar y reconstruir los datos de las peticiones en formato JSON se utiliza un objeto de la clase `JavaScriptSerializer`[77].

3.5.3 BIBLIOTECA DE CLASES PARA EL MANEJO DE LOS DATOS

Par manejar los datos, se elaboró una biblioteca de clases que permite almacenar los datos en objetos, estas clases tienen una estructura similar al modelo de datos de la base de datos. Las variables definidas en las clases son encapsuladas para asegurar una correcta manipulación de los datos. Se definen

---

[75] **CookieContainer:** Contenedor para almacenar *cookies*.
[76] **Dominio:** Parte de la URL que identifica el sitio web incluido el número de puerto.
[77] **JavaScriptSerializer:** Clase que permite la serialización y reconstrucción de objetos en formato JSON.

constructores de inicialización[78], constructores de copia[79] y constructores con valores iniciales[80] para facilitar la creación de objetos de acuerdo a las necesidades del programador. Finalmente, se sobrescribe el método `ToString`, para que devuelva el valor de la variable más significativa de la clase.

Con esta biblioteca de clases se facilita la manipulación de datos que se envían y reciben del servidor. La biblioteca de clases se utiliza en el servidor y en la aplicación de escritorio. En el **Anexo D** se incluye el código completo de la biblioteca de clases.

```
class Proxy
    {
        private static string Url;

        // autentica al usuario al servicio
        public static Usuario Login(string alias, string password)
        {
            //obtiene la URL del servicio de autenticacion
            Url = ConfigurationManager.AppSettings.Get("Ip")
                + ConfigurationManager.AppSettings.Get("Autenticacion");
            CredencialesUsuario credential = new CredencialesUsuario();
            credential.alias = alias;
            credential.password = password;

            // Construye la peticion json
            var serializer = new JavaScriptSerializer();
            var jsonRequestString = serializer.Serialize(credential);
            var bytes = Encoding.UTF8.GetBytes(jsonRequestString);


            //Inicia la peticion HTTP con la sesion manejada por la clase CookiedFactory
            var request = CookiedRequestFactory.CreateHttpWebRequest(Url);
            request.Method = "POST";
            request.ContentType = "application/json";
            request.Accept = "application/json";

            //envia los datos JSON al servicio REST
            var postStream = request.GetRequestStream();
            postStream.Write(bytes, 0, bytes.Length);
            postStream.Close();

            // Toma los resultados de la peticion
            var response = (HttpWebResponse)request.GetResponse();
            var reader = new StreamReader(response.GetResponseStream());
            var jsonResponseString = reader.ReadToEnd();
            reader.Close();
            response.Close();

            //Reconstruye los datos json y retorna el resultado
            serializer = new JavaScriptSerializer();
            return serializer.Deserialize<ComprasVirtuales.Usuario>(jsonResponseString);
        }

    }
```

Código 3.12 Extracto de la clase `Proxy`

[78] **Constructor de inicialización**: Crea objetos sin datos.
[79] **Constructor de copia:** Permite crear un objeto a partir de otro objeto del mismo tipo.
[80] **Constructor con valores iniciales:** Recibe como argumentos los campos que se necesitan para crear el objeto.

3.5.4 FILTROS PARA BÚSQUEDA DE REGISTROS

Cuando existe gran cantidad de registros en los controles `DataGridView`[81] o `ComboBox`[82], es una buena idea implementar algún método que filtre la información, a fin de que el usuario pueda encontrar el registro requerido de manera más adecuada.

En la aplicación de escritorio se implementan filtros de búsqueda en todo los formularios que utilizan controles `DataGridView` o `ComboBox`.

3.5.4.1 Filtro de Búsqueda en controles DataGridView [24]

En este tipo de controles se establecen filtros de búsqueda de acuerdo a los parámetros más representativos de la tabla mostrada. Para ello, se emplea un control `ComboBox` para seleccionar el parámetro a establecer para la búsqueda, junto al `ComboBox` se coloca un `TextBox`[83] para ingresar el texto que se desea buscar. El **Código 3.13** muestra un extracto del código utilizado para implementar el filtro de búsqueda.

El evento asociado al método `txtValor_TextChanged` es `TextChanged`, cada vez que se cambia el contenido del `TextBox` se llama al método mencionado.

Como se puede observar en el **Código 3.13**, lo primero que realiza el método es identificar el parámetro que se va a utilizar para la búsqueda. Luego, con una sentencia `switch` identifica el bloque que se debe ejecutar para realizar la búsqueda deseada. Luego, con base a los caracteres ingresados en el `TextBox`, se filtran los registros que no tengan coincidencias y solo se dejan los registros con coincidencia. Finalmente, el resultado se pasa al `DataGridView`. Este método se vuelve a ejecutar cada vez que se ingresa un nuevo carácter en el `TextBox`.

---

[81] **DataGridView:** Control utilizado para desplegar la información en forma de tablas
[82] **ComboBox:** Control utilizado para la selección de un registro de una lista.
[83] **TextBox:** Control utilizado para ingresar texto.

```
private void txtValor_TextChanged(object sender, EventArgs e)
        {
            string parametro = "";

            try
            {
                //identifica el parametro que se utiliza para la busqueda
                parametro = cmbParametro.SelectedItem.ToString();
            }
            catch (Exception a)
            {
                MessageBox.Show(a.ToString());
            }

            //condicional para identificar el parametro de busqueda
            switch (parametro)
            {
                case "Cédula":

                    //identifica el nombre de la columna que se utiliza para realizar la
                    //busqueda
                    string campoCedula = string.Concat("[",
                    tablaUsuarios.Columns[0].ColumnName, "]");
                    //establece el parametro de busqueda en la tabla
                    tablaUsuarios.DefaultView.Sort = campoCedula;
                    //toma la vista actual del DataTable para realizar la busqueda
                    DataView viewCedula = tablaUsuarios.DefaultView;
                    //inicializa la busqueda
                    viewCedula.RowFilter = string.Empty;
                    //identifica que el textbox utilizado para la busqueda no este vacio
                    if (txtValor.Text != string.Empty)
                        //metodo que filtra los registros en base a los caracteres
                        //ingresados
                        viewCedula.RowFilter = campoCedula + " LIKE '%" + txtValor.Text +
                        "%'";
                    //muestra en el datagridview los resultados de la busqueda
                    dgvUsuarios.DataSource = viewCedula;

                    break;

            }

        }
```

Código 3.13 Extracto del método `txtValor_TextChanged` [24]

3.5.4.2 Filtro de búsqueda en controles ComboBox [25]

La implementación de un filtro de búsqueda en el control `ComboBox` es más sencilla. Simplemente hay que almacenar los elementos de la lista del `ComboBox` en un arreglo llamado `AutoCompleteStringCollection`[84]. Esta colección se la pasa a la propiedad `AutoCompleteSource` del `ComboBox` y se selecciona el método de autocompletar `suggest`. En el **Código 3.14**, las líneas subrayadas muestran las líneas de código que habilitan la característica de autocompletar del `ComboBox`.

[84] **AutoCompleteStringCollection:** Colección de cadenas para utilizar las características de autocompletar de ciertos controles de Windows Forms.

3.5.5 IMPRESIÓN DE FACTURAS

La aplicación de escritorio permite realizar la facturación. Por lo tanto, en esta aplicación se debe implementar una forma de imprimir las facturas que se emiten, para que puedan ser entregadas al cliente.

En la aplicación de escritorio se resuelve esta necesidad tomando una captura del área de la pantalla que contiene la factura y mandando a imprimir esa área. El **Código 3.15** muestra el método para realizar la captura de pantalla y la variable que se debe crear para almacenar las dimensiones de la captura.

```
/// <summary>
/// metodo utilizado para cargar los datos en el combobox
/// </summary>
private void cargarArticulos()
{
    //llamada al servicio web
    Articulos = Proxy.obtenerArticulos();



    int i = 0;

    //carga de los datos en el combobox
    foreach (Articulo articulo in Articulos)
    {

        cmbArticulo.Items.Add(Articulos[i]);
        autoCompletarArticulos.Add(Articulos[i].ToString());
        i++;
    }

    cmbArticulo.AutoCompleteCustomSource = autoCompletarArticulos;
    cmbArticulo.AutoCompleteMode = AutoCompleteMode.Suggest;
    cmbArticulo.AutoCompleteSource = AutoCompleteSource.CustomSource;

}
```

Código 3.14 Método `cargarArticulo`

El método que se muestra en el **Código 3.15** guarda una imagen temporalmente de la captura realizada. Esta imagen luego se imprime mediante el control `printDocument`. El **Código 3.16** muestra el método que realiza la impresión de la factura.

```
/// <summary>
/// Captura el formulario frmFactura como un bitmap y guarda
/// una copia de la imagen dentro del proyecto con una extensión .jpg
/// </summary>
private void CapturarPantalla()
{

    Graphics mygraphics = this.CreateGraphics();
    Size sz = this.ClientRectangle.Size;
    bmp = new Bitmap(sz.Width, sz.Height, mygraphics);
    Graphics memoryGraphics = Graphics.FromImage(bmp);
    IntPtr dc1 = mygraphics.GetHdc();
    IntPtr dc2 = memoryGraphics.GetHdc();
    // Se asignan las dimenciones para la captura de pantalla
    BitBlt(dc2, 0, 0, this.ClientRectangle.Width,
           this.ClientRectangle.Height, dc1, 0, 0, 13369376);
    mygraphics.ReleaseHdc(dc1);
    memoryGraphics.ReleaseHdc(dc2);
    //Se guarda la imagen de la factura dentro del proyecto
    bmp.Save("factura" + factura.Id + ".jpg", ImageFormat.Png);
    //SubirFactura(bmp);

}

[System.Runtime.InteropServices.DllImport("gdi32.dll")]
//Se crea la variable BitBlt del tipo long para almacenar las dimensiones  de la
//captura
public static extern long BitBlt(IntPtr hdcDest,
    int nXDest,
    int nYDest,
    int nWidth,
    int nHeight,
    IntPtr hdcSrc,
    int nXSrc,
    int nYSrc,
    int dwRop);
```

Código 3.15 Método `CapturarPantalla`

```
private void btnImprimir_Click(object sender, EventArgs e)
{
    btnImprimir.Visible = false;
    //Captura el frmFactura en un dataGrid y lo guarda
    CapturarPantalla();
    btnImprimir.Visible = true;
    //Se asigna el docomuento de impresión a un dialogo de vista previa
    VistaPrevia.Document = DocumentoParaImprimir;
    //Se muestra el cuadro de dialogo de vista previa
    VistaPrevia.ShowDialog();
    //Se asigna el docomuento de impresión al dialogo de impresión
    Impresora.Document = DocumentoParaImprimir;
    //Muestra el cuadro de dialogo previo a la impresión
    DialogResult Result = Impresora.ShowDialog();

    if (Result == DialogResult.OK)
    {
        DocumentoParaImprimir.DefaultPageSettings.Landscape = false;
        //Se Manda a Imprimir el Documento
        DocumentoParaImprimir.Print();
    }

}

void DocumentoParaImprimir_PrintPage(object sender, PrintPageEventArgs e)
{
    //Se asigna el bitmap obtenido al capturar la pantallo a un control
    //de documento de impresión
    e.Graphics.DrawImage(bmp, 0, 0, bmp.Width, bmp.Height);
}
```

Código 3.16 Métodos `btnImprimir_Click` y `DocumentoParaImrpimir_PrintPage`

3.5.6 CARGA DE IMÁGENES AL SERVIDOR

La aplicación de escritorio, al realizar la administración de los artículos que se ofertan a los clientes, debe permitir la opción de cargar la imagen del artículo que se va a mostrar en la aplicación móvil. Las imágenes se cargan invocando al método del Servicio Web `SubirImagen`. Este método recibe como argumento la matriz de bytes de la imagen a cargar, por lo que en la clase `Proxy` de la aplicación de escritorio se desarrolló el método `TransformarImagenAMatrizdeBytes`, este método transforma una imagen en una matriz de bytes. El **Código 3.17** muestra el método `TransformarImagenAMatrizdeBytes`.

```
public static byte[] TransformarImagenAMatrizdeBytes(System.Drawing.Image
imageIn)
{
    //Se Declara y se inicializa un Stream de Memoria
    MemoryStream ms = new MemoryStream();
    //Se carga la imagen en el stream de memoria en forma de bytes
    //con formato Jpeg
    imageIn.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
    //Se Retorna el Stream de Memoria en forma de Matriz de bytes
    return ms.ToArray();
}
```

Código 3.17 Método `TransformarImagenAMatrizdeBytes`

3.5.7 DESCARGA DE IMÁGENES DEL SERVIDOR

En ciertos formularios es necesario visualizar imágenes alojadas en el servidor. Para descargar las imagenes y mostrarlas en los controles correspondientes se incluyó en la clase `Proxy` el método `ObtenerImagen` el cual descarga el flujo de bytes de la imagen alojada en el servidor y devuelve la imagen para cargarla en el control requerido. El **Código 3.18** muestra el método `ObtenerImagen`

```
public static Image ObtenerImagen(string url)
{

    // Se crea un Web Client para acceder a la imagen remota
    WebClient clienteWeb = new WebClient();
    // Se guarda en un arreglo de bytes los datos de la imagen
    byte[] imagenBytes = clienteWeb.DownloadData(url);
    // Se crea un Stream de datos de la imagen remota
    MemoryStream stream = new MemoryStream(imagenBytes);
    // Se devuelve una imagen construida con el stream de datos
    return Image.FromStream(stream);
}
```

Código 3.18 Método `ObtenerImagen`

3.5.8 PASO DE DATOS ENTRE FORMULARIOS

En la aplicación de escritorio se utiliza delegados[85] para poder pasar datos de un formulario a otro. Para ello, el formulario que envía el dato debe definir un objeto delegado y un evento que tenga como firma el delegado. El **Código 3.19** muestra las líneas que definen el evento y el delegado en el formulario.

```
//delegado utilizado para pasar el usuario autenticado
//al formulario principal
public delegate void pasar(Usuario dato);

//evento que se ejecuta para pasas datos de un formulario a otro
public event pasar pasado;
```

Código 3.19 Líneas de código para definir el evento y el delegado

El evento `pasado` se llama en el botón aceptar del formulario `frmAutenticacion`, este evento es el que permite pasar la información del usuario al formulario `frmPrincipal`. El **Código 3.20** muestra el método `btn Aceptar_Click` asociado al evento `click` del control `btnAceptar`, en el cual se llama al evento `pasado`.

```
private void btnAceptar_Click(object sender, EventArgs e)
   {
       //Mediante el evento pasado se envia el usuario que retorna el metodo login
      //Se envia el password del usuario como un hash calculado de 16 bytes con  MD5
       pasado(Proxy.Login(txtAlias.Text,
           ComprasVirtuales.Usuario.EncriptarMd5(txtContrasenia.Text)));
   }
```

Código 3.20 Método `btnAceptar_Click`

Por último en el formulario destino de los datos, en este caso el formulario `frmPrincipal`, se debe asignar el delegado `pasar` al evento `pasado` en el método asociado al evento `Load` del formulario. El delegado deberá recibir como argumento un método que tenga la misma firma del delegado y este método será el que recibe la información del usuario que se generó en el formulario `frmAutenticacion`. El **Código 3.21** muestra el método `frmPrincipal_`

[85] **Delegado:** es un objeto que permite pasar métodos como argumentos a otros métodos

`Load` asociado al evento `Load` del formulario, la línea subrayada es la línea de código que asigna el delegado al evento.

```
    private void frmPrincipal_Load(object sender, EventArgs e)
        {
            //deshabilita todos los controles del sistema al momento que se abre la
aplicacion
            tsmInicio.Enabled = false;
            tsmAdministracion.Enabled = false;
            tsmFacturacion.Enabled = false;

            //despliega la ventana de autenticacion al iniciar la aplicación
            autenticacion = new frmAutenticacion();
            autenticacion.MdiParent = this;
            autenticacion.Show();

            //al metodo pasado del formulario autenticacion se asigna el delegado pasar
            //del formulario autenticacion
            autenticacion.pasado += new frmAutenticacion.pasar(ejecutar);


        }
```

Código 3.21 Método `frmPrincipal_Load`

3.5.9 ARCHIVO DE CONFIGURACIÓN

El archivo de configuración sirve para almacenar los URL[86] (*Uniform Resource Locator*) del Servicio Web y cierta información que se muestra en la aplicación de escritorio.

El **Código 3.22** muestra la parte del archivo de configuración que se utiliza para almacenar los URL necesarios para consumir los métodos del Servicio Web. De esta manera, si se necesitan cambiar estos URL, los cambios se realizan únicamente en este archivo.

El **Código 3.23** muestra la parte del archivo de configuración que se utiliza para almacenar información del establecimiento que utiliza la aplicación. Específicamente, almacena la información del establecimiento que se muestra en la factura. De esta manera, si se necesitan cambiar estos datos, los cambios deberán realizarse únicamente en el archivo de configuración.

[86] **URL:** Establece la localización de un recurso de la red.

```
<configuration>
    <startup>
        <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
    </startup>

  <appSettings>
    <add key="Ip" value="http://localhost:70"/>
    <add key="Ftp" value="ftp://localhost/Imagenes/"/>
    <add key="FtpReportes" value="ftp://localhost/reportes"/>
    <add key="Autenticacion"
value="/ServicioEscritorio.svc/postmethod/autenticacion"/>
    <add key="ActualizarUsuario"
value="/ServicioEscritorio.svc/postmethod/actualizarUsuario"/>
    <add key="ActualizarPassword"
value="/ServicioEscritorio.svc/postmethod/actualizarPassword"/>
    <add key="ObtenerUsuarios"
value="/ServicioEscritorio.svc/postmethod/obtenerUsuarios"/>
    <add key="ObtenerProductos"
value="/ServicioEscritorio.svc/postmethod/obtenerProductos"/>
    <add key="ObtenerArticulos"
value="/ServicioEscritorio.svc/postmethod/obtenerArticulos"/>
    <add key="ObtenerMarcas"
value="/ServicioEscritorio.svc/postmethod/obtenerMarcas"/>
    <add key="ObtenerModelos"
value="/ServicioEscritorio.svc/postmethod/obtenerModelos"/>
    <add key="ObtenerImpuestos"
value="/ServicioEscritorio.svc/postmethod/obtenerImpuestos"/>
    <add key="ObtenerTipoPagos"
value="/ServicioEscritorio.svc/postmethod/obtenerTipoPAgo"/>
    <add key="ObtenerTipoUsuario"
```

Código 3.22 Parte del archivo de configuración `app.config`

```
    <add key="NumLote" value="100178"/>
    <add key="NumFactura" value="1"/>
    <add key="NomLocal" value="Compras Virtuales"/>
    <add key="Direccion" value="Ladrón de Guevara E11-253 y Andalucia"/>
    <add key="Contacto" value="compras.virtuales@epn.edu.ec"/>
    <add key="Telefono" value="02999999"/>
    <add key="Ruc" value="1768149930001"/>

  </appSettings>

</configuration>
```

Código 3.23 Parte del archivo de configuración `app.config` (continuación)

3.6 DESARROLLO DE LA APLICACIÓN MÓVIL

La aplicación móvil es una de las aplicaciones clientes que consumen el Servicio Web, esta aplicación está desarrolla usando Eclipse empleando el lenguaje de programación Java. La función de esta aplicación es brindar una interfaz gráfica amigable para las tareas que ejecutan los usuarios Usuario y Usuario VIP.

En las siguientes secciones se detalla el desarrollo y codificación de ciertas partes de la aplicación que se consideran importantes, el código fuente completo y comentado se encuentra en el **Anexo E.1**.

3.6.1 MANEJO DE SESIONES

En el caso de la aplicación móvil, para mantener sesión con el servidor, se debe almacenar la *cookie* que se recibe en la actividad Autenticación e importarla a través de todas las actividades posteriores que realizan peticiones al servidor.

La *cookie* se la extrae de la petición HTTP y se la almacena en un arreglo del tipo *cookie*, el **Código 3.24** muestra las líneas de código que extraen la *cookie* de la petición HTTP.

```
//extrae y almacena la cookie de la respuesta
List<Cookie> cook=httpclient.getCookieStore().getCookies();
Cookie cookie=cook.get(0);
```

Código 3.24 Líneas de código para extraer la *cookie* de la petición HTTP

3.6.2 CONSUMO DEL SERVICIO WEB

Para el caso de la aplicación móvil, el Servicio Web se consume a través de métodos incluidos en cada actividad que requiere consumir el Servicio Web, esto debido a que cada vez que se requiere consumir métodos del Servicio Web se lo deben realizar a través de una implementación de la clase abstracta `AsyncTask` que permite ejecutar operaciones de red, estas clases son definidas dentro de las clases de las actividades que requieren consumir el Servicio Web.

Los métodos desarrollados para el consumo del Servicio Web devuelven la *cookie* para obtener la sesión mantenida con el servidor en cada petición, los mismos métodos reciben como argumentos el dato serializado en un objeto JSON. Las peticiones son realizadas a través de los objetos de la clase `DefaultHttpClient` y `HttpPost` y las respuestas son recibidas en un objeto de la clase `HttpResponse`. El **Código 3.25** muestra un método como ejemplo de los métodos desarrollados para el consumo del Servicio Web en cada actividad.

```
      private Cookie obtenerUsuario(JSONObject dato) throws
ClientProtocolException, IOException
      {
            //objeto utilizado para realizar peticiones HTTP
            DefaultHttpClient httpclient = new DefaultHttpClient();
            //objeto utilizado para realizar peticiones post
            HttpPost post = new HttpPost(url + autenticacion);
            //añade las cabeceras necesarias para la peticion
            post.setHeader("content-type", "application/json");
            //añade a la peticion el cuerpo de mensaje (parametros requeridos
            por el servicio)
            StringEntity stringEntity = new StringEntity(dato.toString());
            post.setEntity(stringEntity);

            //objeto para recibir la respuesta del servicio web
            HttpResponse response;
            //llamada al servicio web
            response = httpclient.execute(post);

            //extrae y almacena la cookie de la respuesta
            List<Cookie> cook=httpclient.getCookieStore().getCookies();
            Cookie cookie=cook.get(0);
            //extrae los datos devueltos en la respuesta (cuerpo del mensaje)
            HttpEntity entity = response.getEntity();
            //condicional que comprueba que la respuesta no este vacia
            if (entity != null) {
                  //transforma a String el resultado
                  String result= EntityUtils.toString(entity);
                  //transforma a int el String anterior
                  acceso = Integer.parseInt(result);

                  return cookie;
            }

            else {
                  Log.e("ComprasVirtuales", "entity == null");
                  return new BasicClientCookie("","");
            }
      }
```

Código 3.25 Método `ObtenerUsuario`

3.6.3 SERIALIZACIÓN Y DESERIALIZACIÓN DE OBJETOS JSON [26]

3.6.3.1 Serialización de objetos JSON

En la aplicación móvil desarrollada, para construir objetos JSON utilizados para realizar las peticiones al servidor, se utiliza objetos de la clase `JSONObject`. Esta clase permite la construcción de objetos JSON mediante el método de esta `put`, el cual agrega la llave nombre/valor al objeto requerido. El **Código 3.26** muestra un ejemplo de los métodos desarrollados para construir los objetos JSON, necesarios para realizar las peticiones al servidor.

```
//Metodo para construir el objeto JSON utilizado para realizar la
peticion al servicio Web
    private JSONObject construirDatoAgregarArticulo(String idArticulo)
throws JSONException{

            JSONObject dato = new JSONObject();

            dato.put("idArticulo", idArticulo);
            dato.put("cantidad", 1);

            return dato;
      }
```

Código 3.26 Método `construirDatoAgregarArticulo`

3.6.3.2 Deserialización de objetos JSON

En la aplicación móvil desarrollada, con la finalidad de simplificar el tratamiento de los datos que se reciben del servidor en formato JSON, se utiliza la biblioteca de clases `GSON`[87].

Los datos que se reciben en formato JSON, son convertidos a objetos a través del método `fromJson` de la librería `GSON`. Este método recibe como parámetros el mensaje JSON y el tipo de resultado que se quiere recibir. El **Código 3.27** muestra las líneas de código utilizadas para realizar la deserialización.

```
//objeto utilizado para manipular los objetos JSON
Gson gson = new Gson();

//arreglo para almacenar la informacion
List<Seleccion> resultado;

//obtiene el tipo de resultado
Type tipoResult = new TypeToken<List<Seleccion>>(){}.getType();

//deserializa los objetos JSON
resultado = gson.fromJson(result, tipoResult);
```

Código 3.27 Parte del código utilizada para deserializar objetos JSON

[87] **GSON:** Librería de Java desarrollada por Google que permite convertir objetos Java en su representación JSON.

3.6.4 BIBLIOTECA DE CLASES PARA EL MANEJO DE DATOS

Para permitir la deserialización de objetos JSON, se elaboran clases con estructura similar a la del objeto JSON, estas clases reciben directamente los datos contenidos en el objeto JSON. El **Código 3.28** muestra la estructura de la clase que se debe elaborar para poder deserializar objetos JSON.

```
public class Modelo {
    //mediante el uso de este tipo de clases permite la serializacion
    //utilizando JSON
    @SerializedName("Id")
    public String id;
    @SerializedName("Articulo")
    public String articulo;
    @SerializedName("Marca")
    public String marca;
    @SerializedName("Nombre")
    public String nombre;
    @SerializedName("Descripcion")
    public String descripcion;
    @SerializedName("Disponibilidad")
    public String disponibilidad;
    @SerializedName("Valor")
    public String valor;
    @SerializedName("Descuento")
    public String descuento;
    @SerializedName("Impuesto")
    public String impuesto;
    @SerializedName("ImagenUrl")
    public String imagenUrl;
    //bloque de codigo para retornar un objeto vacio
    public Modelo (){}
  public static Modelo getEmpty(){
    return new Modelo();
  }
}
```

Código 3.28 Clase `Articulo`

3.6.5 CLASE ABSTRACTA ASYNCTASK [13]

Las operaciones de red pueden implicar comportamientos impredecibles. Para prevenir que este tipo de comportamiento pueda influir en el comportamiento de la aplicación y la misma no responda, todas las operaciones de red deben ejecutarse en un hilo diferente al hilo principal. La clase `AsyncTask` proporciona una forma simplificada de ejecutar operaciones de red fuera del hilo principal.

Una tarea asíncrona se define como un proceso que se ejecuta en un hilo secundario y en segundo plano, y cuyo resultado se publica en el hilo principal. Una tarea asíncrona implementada con esta clase define el tipo de datos para tres argumentos genéricos: Parámetros, Progreso y Resultado. También en esta tarea asíncrona se definen cuatro métodos principales, llamados `onPreExecute`, `doInBackground`, `onProgressUpdate` *y* `onPostExecute`.

3.6.5.1 Tipos de datos genéricos utilizados en la clase AsyncTask

En una tarea asíncrona se debe definir los tipos de datos `<Parametros,Progreso,Resultado>` que se explican a continuación:

- *Parámetros:* Define el tipo de datos que van a ser enviados como argumentos para la ejecución de la tarea asíncrona.
- *Progreso:* Define el tipo de datos que van a ser publicados durante la ejecución del hilo secundario.
- *Resultado:* Define el tipo de datos que se van a devolver al final del procesamiento en el hilo secundario.

La definición de estos tipos de datos se lo realiza en la declaración de la clase. El **Código 3.29** muestra los tipos de datos que se definen para la implementación de la clase abstracta `AsyncTask` en operaciones de red.

```
//clase utilizada para ejecutar una tarea asincronica
//(tareas de red no se pueden ejecutar en el hilo principal)
public class WebServiceClientObtenerSeleccion extends AsyncTask
<JSONObject,Void,List<Seleccion>>{ . . . }
```

Código 3.29 Implementación de la clase abstracta `AsynkTask`

3.6.5.2 Principales métodos de la clase AsyncTask

Cuando se ejecuta una tarea asíncrona, esta tarea pasa por cuatro métodos que se explican a continuación.

3.6.5.2.1 onPreExecute

Este método se invoca antes de que la tarea se ejecute en el hilo secundario. En la aplicación móvil del proyecto, este método se sobre escribe para mostrar en

pantalla un dialogo de progreso, el cual se muestra mientras la tarea se está ejecutando. El **Código 3.30** se utiliza para realizar la sobre escritura del método `onPreExecute`.

```
@Override
protected void onPreExecute() {
        super.onPreExecute();
    dialog.setMessage("Cargando");
    dialog.show();
}
```

Código 3.30 Sobre escritura del método `onPreExecute`

3.6.5.2.2 doInBackGround

Este método se invoca inmediatamente después de la ejecución del método `onPreExecute`. Para el caso de la aplicación móvil que se implementa, el método sirve para realizar las peticiones al Servicio Web. El **Código 3.31** se utiliza para la sobre escritura del método `doInBackGround`.

```
  @Override
protected List<Seleccion> doInBackground(JSONObject... dato)
{
        List<Seleccion> result;
        try    {
              result=ObtenerSeleccion();
        }
        catch(Exception e){
              result= new ArrayList<Seleccion>();
              e.printStackTrace();
        }
    return result;
}
```

Código 3.31 Sobre escritura del método `doInBackGround`

Para el caso del **Código 3.31**, el método `obtenerSeleccion` realiza las peticiones al Servicio Web y devuelve los datos resultantes mediante la variable `result`.

3.6.5.2.3 onPostExecute

Este método se invoca al finalizar las tareas del método `doInBackGround`, el resultado del método `doInBackGround` se pasa como parámetro al método `onPostExecute`. Para el caso de la aplicación móvil que se implementa, el método controla que la petición al Servicio Web se haya ejecutado con éxito; en caso de que la sesión con el servidor este caducada, se redirecciona al usuario a la actividad de autenticación, si la información se recibe correctamente, se cargan los datos en los controles correspondiente para que el usuario pueda visualizarlos. El **Código 3.32** se utiliza para la sobre escritura del método `doInBackGround`.

```
protected void onPostExecute(List<Seleccion> result) {
            dialog.dismiss();
            try{
                  //Bloque de codigo a ejecutarse en caso de haber error en la
                  //ejecucion del servicio se redirecciona a la actividad de
                  //autenticacion
            if (result.get(0)==null){
                   Intent actAutenticacion = new
                   Intent(CarroActivity.this, AutenticacionActivity.class);
                   startActivity(actAutenticacion);
                   CarroActivity.this.finish();
            }
            else{
                  //alamacena la seleccion que recibe del servicio
                  seleccion=result;

                  if (seleccion.size()!=0)
                  {
                        //carga los datos del costo total en la actividad
                        total.setText(seleccion.get(0).costoTotal);
                  }
                       //carga la lista en el Adapter
                       setListAdapter(new
                       SeleccionAdapter(CarroActivity.this,seleccion));
              }
            }
            catch (Exception e){

            seleccion=result;

            if (seleccion.size()!=0)
            {
                  total.setText(seleccion.get(0).costoTotal);
            }

                  setListAdapter(new
                  SeleccionAdapter(CarroActivity.this,seleccion));

            }

               super.onPostExecute(result);
        }
```

Código 3.32 Sobre escritura del método `onPostExecute`

3.6.6 USO DE MENÚS EN LAS ACTIVIDADES

Por la limitación de espacio que existe en los dispositivos móviles, ciertos controles se deben ubicar en menús. En la actividad `CarroActivity` se hace uso de los menús `OptionsMenu` y `ContextMenu`.

3.6.6.1 OptionsMenu

Este control usa el botón menú presente en todos los dispositivos móviles que utilizan el sistema operativo Android. En el caso de la aplicación desarrollada, al momento de presionar el botón menú en la actividad `CarroActivity` se muestran las opciones “limpiar carro de compras” que permite eliminar todos los artículos que están en el carro de compras y “finalizar compra” que le permite al usuario finalizar la compra.

Para incluir este menú en la actividad se sobre escribe el método `onCreateOptionsMenu`, este método asigna el `layout` a utilizar. Además se debe definir el método `onOptionsItemSelected` asociado al evento `ItemSelected` del menú. El método antes mencionado sirve para determinar las acciones que realiza cada elemento del `OptionsMenu`. El **Código 3.33** muestra el método `onOptionsItemSelected` y el **Código 3.34** muestra el método `onCreateOptionsMenu`.

```
//metodo utilizado para hacer uso del menu del movil
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
    case R.id.item1:
        //llamada al servicio web para eliminar la seleccion
        //al eliminar la seleccion se borran todos los articulos seleccionados
        // y empieza una nueva seleccion
        new WebServiceClientOpcionesMenu().execute(eliminarSeleccion);
      return true;
    case R.id.item2:
        new WebServiceClientOpcionesMenu().execute(finalizarSeleccion);
      return true;
    default:
      return super.onOptionsItemSelected(item);
            }
        }
```

Código 3.33 Método `onOptionsItemSelected`

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

Código 3.34 Método `onCreateOptionsMenu`

3.6.6.2 ContextMenu

Este menú se despliega al momento que se mantiene seleccionado un elemento de un `ListView`. En el caso de la actividad `CarroActivity` al momento de desplegar el `ContextMenu` de un ítem del carro de compras, se muestran las opciones: “aumentar cantidad”, “disminuir cantidad” y “eliminar producto”.

De la misma manera que el control `OptionsMenu`, para incluir este menú en la actividad se debe sobre escribir el método `onCreateContextMenu`, este método define los elementos del menú. Además se debe definir el método `onContextItemSelected` asociado al evento `ItemSelected` del menú. Para determinar las acciones que realiza cada elemento del `ContextMenu`.

3.6.7 DESCARGA DE IMÁGENES DEL SERVIDOR

Para poder descargar el flujo de bytes de la imagen del servidor se implementa una clase abstracta `AsyncTask`. Este flujo de bytes se almacena en formato `bitmap` y se lo asigna al control. El **Código 3.35** muestra la clase desarrollada para descargar imágenes del servidor.

3.6.8 LECTURA DEL TAG NFC

La lectura del tag NFC se lo realiza en la actividad `CarroActivity`. En el método `onCreate` de la actividad se debe agregar líneas de código para: reconocer el adaptador NFC del equipo móvil, crear un `PendingIntent`[88], leer el tag NFC, crear un `IntentFilter`[89] para la acción de descubrir un nuevo tag,

[88] **PendingIntent:** Da permisos a una aplicación externa para ejecutar líneas de código predefinidas.
[89] **IntentFilter:** Filtra acciones específicas como la lectura del tag.

y definir las tecnologías NFC que soportará la aplicación. El **Código 3.36** muestra las líneas de código que se deben incluir en el método `onCreate` de la actividad.

```
//clase para descargar una imagen desde una URL
      class DownloadImageTask extends AsyncTask<String, Void, Bitmap> {
            ImageView bmImage;

            public DownloadImageTask(ImageView bmImage) {
                  this.bmImage = bmImage;
            }

            @Override
            protected Bitmap doInBackground(String... urls) {
                  String urldisplay = urls[0];
                  Bitmap result = null;
                  try {
                        //recibe el flujo de bytes de la imagen desde el
                        servidor
                        InputStream in = new
                        java.net.URL(urldisplay).openStream();
                        //guarda la imagen en formato bitmap
                        result = BitmapFactory.decodeStream(in);
                  } catch (Exception e) {
                        Log.e("Error", e.getMessage());
                        e.printStackTrace();
                  }
                  return result;
            }

            @Override
            protected void onPostExecute(Bitmap result) {
                  //asigna la imagen al control
                  bmImage.setImageBitmap(result);
            }
      }
```

Código 3.35 Clase `DownloadImageTask`

```
//obtiene el adaptador NFC para su uso
mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
//Un PendingIntent especifica una accion a tomar en el futuro
//en este caso especifica la accion de lectura del tag NFC
mPendingIntent = PendingIntent.getActivity(this, 0,
                  new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
//filtro para definir la accion de descubrir un nuevo TAG
            IntentFilter mifare = new
IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
            mFilters = new IntentFilter[] {
              mifare,
            };
           mTechLists = new String[][] { new String[] {
MifareUltralight.class.getName()}};
```

Código 3.36 Líneas de código que se incluyen en el método `onCreate` para la lectura del tag NFC

En el método `onResume` de la actividad, se habilita el adaptador NFC para escuchar nuevas lecturas de tags NFC. El adaptador se habilita mediante el método `enableForegroundDispatch` el cual recibe como parámetros: la actividad, el `pendingIntent`, el `intentFilter` y las tecnologías que reconocerá el adaptador. El **Código 3.37** muestra el método sobrescrito `onResume`, que habilita el adaptador NFC.

En el método `onPause` de la actividad, se deshabilita el adaptador NFC para optimizar el uso de recursos en el móvil. El **Código 3.38** muestra el método `onPause` que sirve para deshabilitar el adaptador NFC.

```
@Override
public void onResume() {

      //cuando la actividad vuelve a estar activa
      //se habilita la escucha de nuevos TAGS NFC
      super.onResume();
      new WebServiceClientObtenerSeleccion().execute();
      if (mNfcAdapter != null)
      mNfcAdapter.enableForegroundDispatch(this, mPendingIntent,
      mFilters,mTechLists);

}
```

Código 3.37 Sobre escritura del método `onResume`

```
@Override
public void onPause() {
    //cuando la actividad entra en pausa se deshabilita la escucha de
    //nuevos TAGS NFC
    super.onPause();
    mNfcAdapter.disableForegroundDispatch(this);
}
```

Código 3.38 Sobre escritura del método `onPause`

Para definir las operaciones a realizar al momento de la lectura del tag NFC, se sobre escribe el método `onNewIntent`, el cual va a ser llamado al momento que se produce la interacción entre el tag y el móvil NFC. Este método llama al método `readMifareTag`, el cual obtiene el contenido del tag y despliega una nueva actividad para mostrar la información del producto. El **Código 3.39** muestra la sobre escritura del método `onNewIntent`, para definir las

operaciones a realizar cuando se produce la interacción entre el tag y el móvil NFC.

El método `readMifareTag` es el que realiza la lectura de los datos contenidos en el tag. Este método, establece una conexión con el tag, toma los bytes contenidos y transforma dichos bytes a una cadena de texto. El **Código 3.40** muestra el método `readMifareTag`.

Los datos contenidos en el tag se escriben en formato NDEF, por lo que contienen algunas cabeceras que no se usan en la aplicación. Estas cabeceras son separadas con el método `split` que permite separar cadenas de texto.

```
@Override
public void onNewIntent(Intent intent) {
    //desencadenador para el caso de que se lea un
    //nuevo tag en la aplicacion

    Log.i("Foreground dispatch", "Discovered tag with intent: " + intent);
    Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);

    //despliega una nueva actividad con la información del producto
    //del TAG leido
    Intent actProducto= new Intent(CarroActivity.this,ProductoActivity.class);
    actProducto.putExtra("Tag",(readMifireTag(tag)).toString());
    //paso de la información de la cookie a la nueva actividad
    actProducto.putExtra("CookieName",cook.getName());
    actProducto.putExtra("CookieValue",cook.getValue());
    actProducto.putExtra("CookieDomain",cook.getDomain());
    startActivity(actProducto);
}
```

Código 3.39 Sobre escritura del método `onNewIntent`

3.6.9 PASO DE DATOS ENTRE ACTIVIDADES

En la aplicación móvil la información que se pasa entre actividades son los datos de la *cookie*. Para pasar los datos de la *cookie* a través de las actividades se optó por utilizar el método `putExtra` de la clase `Intent`, el cual permite añadir información adicional en forma de cadena de texto a la nueva actividad desplegada. El **Código 3.41** muestra las líneas de código que se utilizan para importar la información de la *cookie* a la siguiente actividad.

Para recibir la información de la *cookie* de la actividad anterior, se desarrolló el método `importarInformacion`, el cual recibe la información para reconstruir

la *cookie.* El **Código 3.42** muestra el método para recibir información de la *cookie* y reconstruirla.

```
//metodo para leer el contenido del TAG
public String readMaifireTag(Tag detectedTag) {

  //obtencion del TAG leido
  MifareUltralight mifare = MifareUltralight.get(detectedTag);
    try {
        //establece conexion con el TAG
        mifare.connect();
        //lectura de las 6 primeras paginas del TAG
        byte[] payload = mifare.readPages(6);

        //separa informacion extra escrita en el tag
        String datos = (new String(payload, Charset.forName("US-
        ASCII"))).toString();
        String[] codigo = datos.split("@");
        //retorno de los datos utiles para la aplicacion
        return codigo[1];

    } catch (IOException e) {
        Log.e(TAG, "No se pudo leer el Mensaje...", e);
    } finally {
        if (mifare != null) {
           try {
               //al final de la lectura se cierra la conexion con el TAG
               mifare.close();
           }
           catch (IOException e) {
               Log.e(TAG, "Error al cerrar tag...", e);
           }
        }
    }
    return null;
}
```

Código 3.40 Método `readMifareTag`

```
//despliega una nueva actividad con la información del producto
//del TAG leido
Intent actProducto= new Intent(CarroActivity.this,ProductoActivity.class);
actProducto.putExtra("Tag",(readMailfireMessageToTag(tag)).toString());
//paso de la información de la cookie a la nueva actividad
actProducto.putExtra("CookieName",cook.getName());
actProducto.putExtra("CookieValue",cook.getValue());
actProducto.putExtra("CookieDomain",cook.getDomain());
startActivity(actProducto);
```

Código 3.41 Líneas de código utilizadas para importar información de las *cookies* entre diferentes actividades

```
//metodo para recibir la informacion de la actividad anterior
//y crear la cookie a utilizar para la peticion al servicio web
protected void importarInformacion(){

    //obtencion de valores de la actividad anterior
    String name = getIntent().getExtras().getString("CookieName");
    String value = getIntent().getExtras().getString("CookieValue");
    String dominio = getIntent().getExtras().getString("CookieDomain");
    //cracion de la cookie a utilizar
    cook= new BasicClientCookie(name ,value);
    cook.setDomain(dominio);


}
```

Código 3.42 Método `importarInformacion`

3.6.10 ARCHIVO CONEXIONSTRINGS.XML

El **Código 3.43** muestra el contenido del archivo `conexionStrings.xml` que se utiliza para almacenar los URL necesarios para la conexión con el servidor. De esta manera, si se necesitan cambiar los URL para conectarse con el servidor, los cambios se realizan únicamente en este archivo.

```
<resources>
    <string name="ip">http://192.168.1.137/</string>
    <string name="autenticacion">/ServicioComprasVirtuales.svc
/postmethod/autenticacionAndroid</string>
    <string name="registro">/ServicioComprasVirtuales.svc/
postmethod/registro</string>
    <string name="obtenerUsuario">/ServicioComprasVirtuales.svc
/postmethod/datosUsuario</string>
    <string name="actualizar">/ServicioComprasVirtuales.svc/
postmethod/actualizarUsuario</string>
    <string name="actualizarPassword">/ServicioComprasVirtuales.svc/
postmethod/actualizarPassword</string>
    <string name="obtenerSeleccion">/ServicioComprasVirtuales.svc/
postmethod/datosSeleccion</string>
    <string name="obtenerModelo">/ServicioComprasVirtuales.svc/
postmethod/obtenerModelo</string>
    <string name="agregarProducto">/ServicioComprasVirtuales.svc/
postmethod/agregarProductoCarro</string>
    <string name="eliminarProducto">/ServicioComprasVirtuales.svc/
postmethod/eliminarProductoCarro</string>
    <string name="eliminarProductos">/ServicioComprasVirtuales.svc/
postmethod/eliminarProductosCarro</string>
    <string name="eliminarSeleccion">/ServicioComprasVirtuales.svc/
postmethod/eliminarSeleccion</string>
    <string name="finalizarSeleccion">/ServicioComprasVirtuales.svc/
postmethod/finalizarSeleccion</string>
    <string name="controlAcceso">/ServicioComprasVirtuales.svc/
postmethod/controlAcceso</string>
    <string name="reanudarSeleccion">/ServicioComprasVirtuales.svc/
postmethod/reanudarSeleccion</string>
</resources>
```

Código 3.43 Archivo `conexionStrings.xml`

3.6.11 ARCHIVO MANIFIESTO

Para que la aplicación móvil desarrollada funcione correctamente se debe definir el archivo manifiesto, en este archivo se define: el nombre del paquete, la versión de la aplicación, el nivel de API requerido, los permisos, las actividades utilizadas, entre otros. Este archivo es el que da los permisos de uso del adaptador NFC. El **Código 3.44** muestra el contenido del archivo manifiesto de la aplicación desarrollada.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.comprasvirtuales"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="19"
        />
            <uses-permission
            android:name="android.permission.INTERNET"/>
            <uses-permission android:name="android.permission.NFC" />
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic launcher"
        android:label="@string/app name"
        android:theme="@style/AppTheme" >
        <activity android:name="MenuActivity" ></activity>
        <activity android:name="RegistroActivity"></activity>
        <activity android:name="CarroActivity"></activity>
        <activity android:name="ConfigurarActivity"></activity>
        <activity android:name="ContraseniaActivity"></activity>
        <activity android:name="AyudaActivity"></activity>
        <activity android:name="AcercadeActivity"></activity>
        <activity android:name="ProductoActivity"></activity>
        <activity android:name="PagarActivity"></activity>
        <activity android:name="ReporteActivity"
            android:screenOrientation="landscape"></activity>
        <activity
       android:name="com.example.comprasvirtuales.
       AutenticacionActivity"
            android:label="@string/app name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Código 3.44 Contenido del archivo manifiesto

3.7 ESCRITURA DE TAGS NFC

Los tags NFC utilizados en el sistema prototipo, contienen el ID del modelo de un artículo registrado en la base de datos. De esta manera, cuando el móvil NFC lee los datos contenidos en el tag, el mismo realiza la petición de la información del producto al servidor con el ID de dicho modelo.

La escritura de los datos en el tag se lo realiza mediante aplicaciones ya desarrolladas que permiten realizar la escritura de tags. Estas aplicaciones permiten escribir datos en el tag con el formato NDEF. En el caso de la aplicación móvil desarrollada para el sistema prototipo, se lee toda la información contenida en el tag sin reconocer el formato NDEF, las cabeceras de este formato son reconocidas por la aplicación como caracteres. Para que la aplicación móvil pueda filtrar las cabeceras del formato NDEF, el ID del artículo es colocado entre los caracteres "@".

La aplicación seleccionada para realizar la escritura de los tags se llama "TagWriter" y fue desarrollada por la empresa NXP Semiconductors.

3.8 SEGURIDADES EN EL SISTEMA

En el sistema prototipo desarrollado, se implementan seguridades básicas con el fin de evitar accesos o modificaciones no autorizadas en los componentes del sistema. A continuación se detallan las seguridades implementadas en cada componente del sistema.

3.8.1 SEGURIDAD EN EL SERVIDOR

El servidor aloja todos los datos en una base de datos, el acceso a registros de la base de datos se realiza únicamente a través del Servicio Web. El Servicio Web establece sesiones con los clientes para aceptar solicitudes únicamente a usuarios que mantengan sesiones activas con el servidor.

La información de las contraseñas de los usuarios, se almacenan cifradas en la base de datos con el algoritmo MD5[90].

3.8.2 SEGURIDAD EN LA APLICACIÓN DE ESCRITORIO

La aplicación de escritorio establece sesiones con el servidor y recibe *cookies* en cada sesión. Con la *cookie* las peticiones al servidor no se realizan con la información del usuario sino con el ID de la *cookie* que identifica la sesión del usuario en el servidor. La información de autenticación se la envía al servidor

[90] **MD5:** Algoritmo de cifrado que transforma una cadena de texto en un número hexadecimal de 32 dígitos.

cifrado con el algoritmo MD5. El **Código 3.45** muestra el método `EncriptarMd5` desarrollado para cifrar los datos de autenticación con MD5.

```
/// <summary>
/// Metodo que encripta una cadena de Texto en md5
/// Este metodo retorna un hash de 16bytes en hexadecimal como un String
/// </summary>
/// <param name="cEntrada"></param>
/// <returns></returns>
public static string EncriptarMd5(string cEntrada) {
    //Se declaran dos matrices de bytes  para almacenar los bytes de la cadena de de origen
    //y el valor de hash resultante.
    byte[] matrizOrigen;
    byte[] matrizHashMd5;
    //Se Crea una matriz de bytes a partir de la cadena de texto que se ingreso como password
    matrizOrigen = ASCIIEncoding.ASCII.GetBytes(cEntrada);
    //Se calcula el hash a partir de la matriz de bytes obtenida
    matrizHashMd5 = new MD5CryptoServiceProvider().ComputeHash(matrizOrigen);

    return MatrizBytesACadenaTexto(matrizHashMd5);
}
```

Código 3.45 Método `EncriptarMd5`

3.8.3 SEGURIDAD EN LA APLICACIÓN MÓVIL

Al igual que en la aplicación de escritorio, se establecen sesiones con el servidor para no enviar los datos del usuario. La información de autenticación se la envía al servidor cifrado con el algoritmo MD5. El **Código 3.46** muestra el método `EncripcionMd5` desarrollado para cifrar los datos de autenticación con MD5.

```
//Función que encripta una cadena de datos en md5
    public static String EncripcionMd5(String cEntrada) throws
Exception {
          MessageDigest md = MessageDigest.getInstance("MD5");
          //Matriz b que contiene el Hash
          byte[] b = md.digest(cEntrada.getBytes());
          int size = b.length;
          StringBuffer h = new StringBuffer(size);
       //Lazo que transforma la Matriz del Hash a formato Hexadecimal
              for (int i = 0; i < size; i++) {
                  int u = b[i] & 255;
                      if (u < 16) {
                          h.append("0" + Integer.toHexString(u));
                      }
                     else {
                          h.append(Integer.toHexString(u));
                      }
                }
            //Se retorna la cadena encriptada con Md5
            return h.toString();
    }
```

Código 3.46 Método `EncripcionMd5`

CAPÍTULO 4. PRUEBAS Y ANÁLISIS DE RESULTADOS

4.1 INTRODUCCIÓN

En el presente capítulo se presentan los resultados de las pruebas unitarias, de carga y funcionalidad realizadas en ambientes controlados de los componentes que conformarán el sistema, también se realiza una comparación entre la aplicación móvil concluida con las aplicaciones que se utilizaron para elaborar el análisis de requerimientos.

La realización de pruebas tiene como finalidad verificar el cumplimiento de los requisitos obtenidos del análisis de requerimientos, verificar errores y a partir de los resultados obtenidos brindar una realimentación para el proyecto.

4.2 PRUEBAS UNITARIAS

Las pruebas unitarias comprueban el comportamiento del código en respuesta a casos estándar, límites y datos de entrada incorrectos. Estas pruebas se realizan para comprobar el funcionamiento de un módulo del código. El objetivo de estas pruebas es comprobar que cada módulo funcione correctamente por separado y al integrar los módulos el programa logre la funcionalidad deseada.

4.2.1 PRUEBAS AL SERVICIO WEB WCF

Para probar el correcto funcionamiento de los métodos del Servicio Web se emplea el programa Fiddler 2[91], tomando en cuenta que las peticiones que se realizan son del tipo POST, que el formato de los datos será JSON y además que se va a manejar una *cookie* para que el servidor identifique que usuario es el que le está realizando la petición.

Para realizar las peticiones se tiene que definir el formato de la petición, el tipo de sesión y si es necesario la *cookie* que contiene la sesión del usuario. La **Figura 4.1** muestra una captura de pantalla con la petición realizada.

[91] **Fiddler 2:** Es una aplicación proxy para programas que utilicen el protocolo HTTP

Figura 4.1 Sintaxis de petición POST JSON en Fiddler 2

Fiddler muestra los resultados para el análisis en pantalla dividida; en la parte superior se encuentran la información de la petición realizada y en la parte inferior se encuentra la respuesta enviada desde el Servicio Web.

El primer método del Servicio Web que se debe probar es el de autenticación dado que este método proporciona la *cookie* que es necesaria para poder realizar las pruebas de los demás métodos.

Para comprobar el método `AutenticacionAndroid` del Servicio Web, se toma en cuenta que se necesita un alias y un *hash* MD5 que representa al *password* que se debe enviar en la petición POST y además que el servidor responderá a la solicitud con el tipo de usuario que se autenticó y establecerá una *cookie* que asocia la sesión del usuario que se autentico. En la **Figura 4.2** se muestra la captura de las cabeceras de la petición POST y la respuesta a esta.

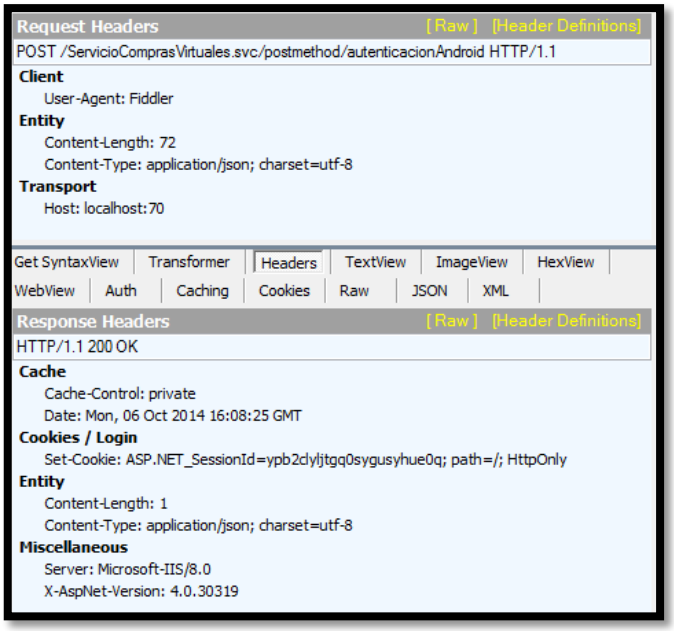

Figura 4.2 Cabeceras de petición mediante POST y respuesta del método `AutenticacionAndroid`

En caso de que se ingrese el alias y el *password* de manera incorrecta el servidor responde con el número 5 que indica que la autenticación falló. La **Figura 4.3** muestra la respuesta que recibe el programa ante una autenticación errónea.

Figura 4.3 Petición POST y respuesta del método `AutenticacionAndroid` con datos de acceso incorrectos

En caso de que se introduzca el alias y el *password* de manera correcta el servidor responde con el número 3 si es un usuario normal (Usuario) y con el número 4 si es un usuario con descuentos especiales (Usuario VIP). La **Figura 4.4** y la **Figura 4.5** muestran la respuesta que recibe el programa ante cada autenticación.

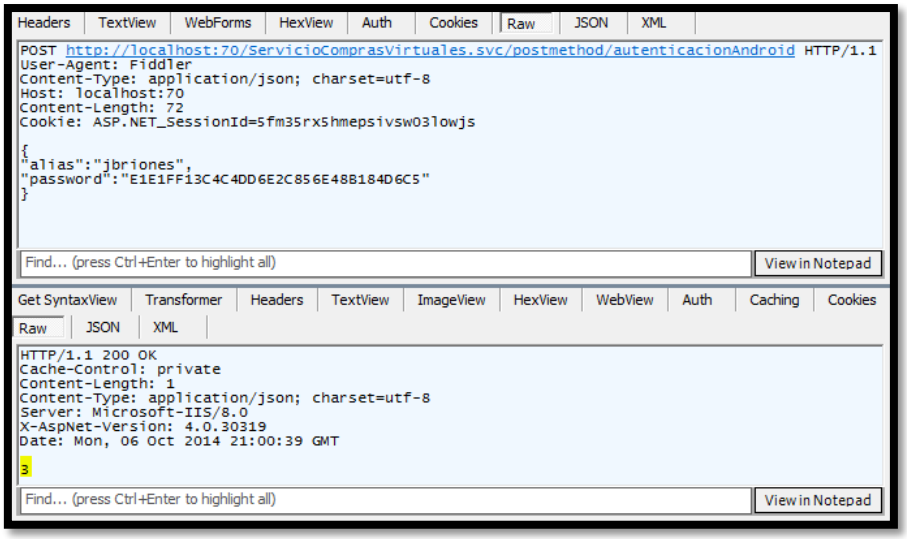

Figura 4.4 Petición POST y respuesta del método `AutenticacionAndroid` con datos de acceso de Usuario

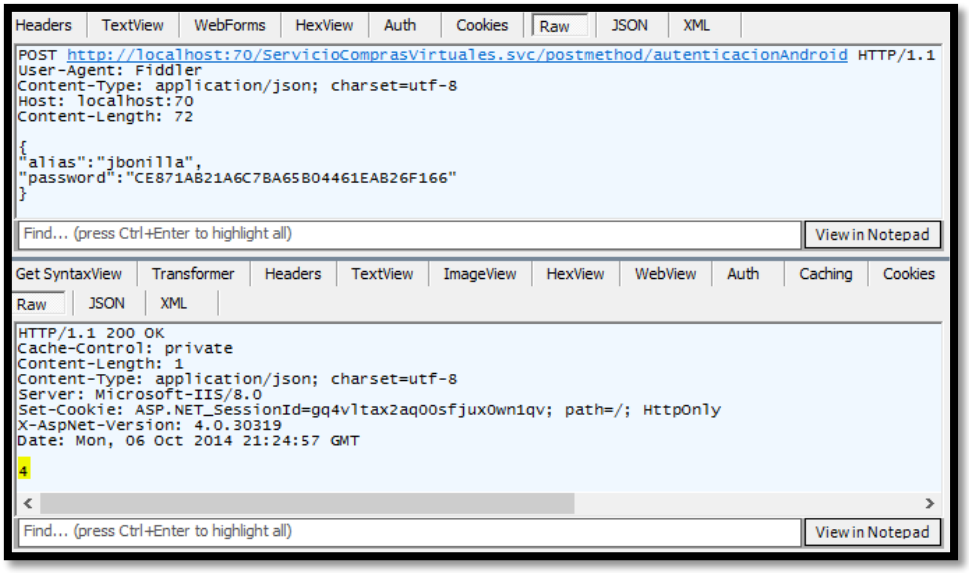

Figura 4.5 Petición POST y respuesta del método `AutenticacionAndroid` con datos de acceso de Usuario VIP

Para realizar las pruebas del resto de métodos del Servicio Web se utiliza la *cookie* que se obtuvo del método de autenticación, si la *cookie* contiene una sesión de un usuario valido va a responder a las peticiones con los datos requeridos como se puede ver en la **Figura 4.6**, en cambio sí se usa una *cookie* de una sesión con datos erróneos el servidor va a responder con objetos vacíos como se puede ver en la **Figura 4.7**.

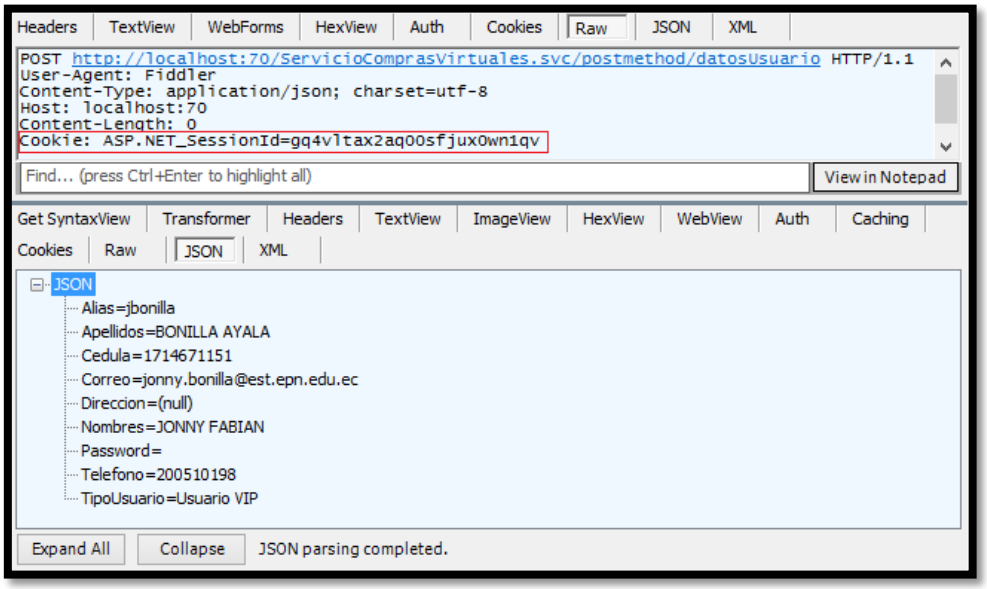

Figura 4.6 Petición POST y respuesta del método `DatosUsuario` con una *cookie* con datos válidos

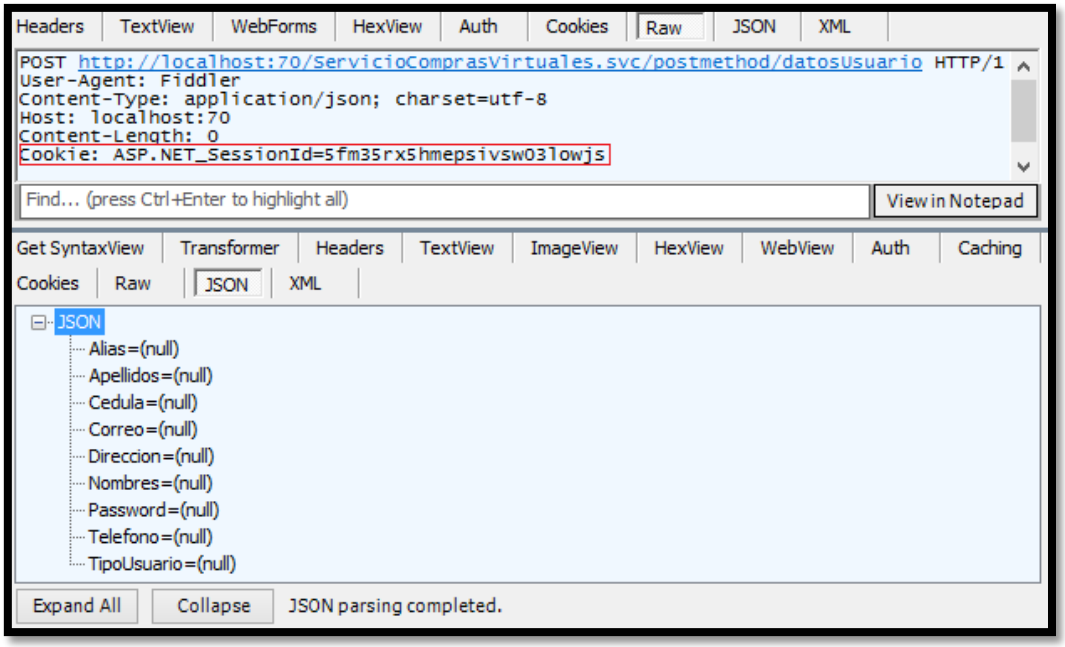

Figura 4.7 Petición POST y respuesta del método `DatosUsuario` con una *cookie* con datos no válidos

Para realizar las pruebas de los métodos que se utilizan en el carro de compras se toma en cuenta que el primer método al que se realiza la petición es `DatosSeleccion`, ya que al realizar una petición a este método, el servidor proporciona los productos de la lista de compras del usuario y se carga la información del identificador de la lista de compras en la sesión del usuario; con esta información contenida en la sesión se realizan las peticiones a los métodos que llevan a cabo tareas como: agregar, quitar, eliminar productos de la lista o finalizar una compra. La **Figura 4.8** muestra la respuesta del método `DatosSeleccion`.

Las peticiones que se realizan a los métodos para agregar, quitar o eliminar productos de la lista de compras requieren de un identificador del artículo deseado y de la cantidad de productos deseada para que con estos datos el servidor pueda realizar los cambios en la base de datos y si la operación se realiza correctamente, el servidor responderá con el mensaje “IngresoCorrecto”. Un ejemplo de este tipo de métodos es `AgregarProductoCarro`. La **Figura 4.9** muestra la petición realizada al servidor con los datos requeridos y la respuesta obtenida.

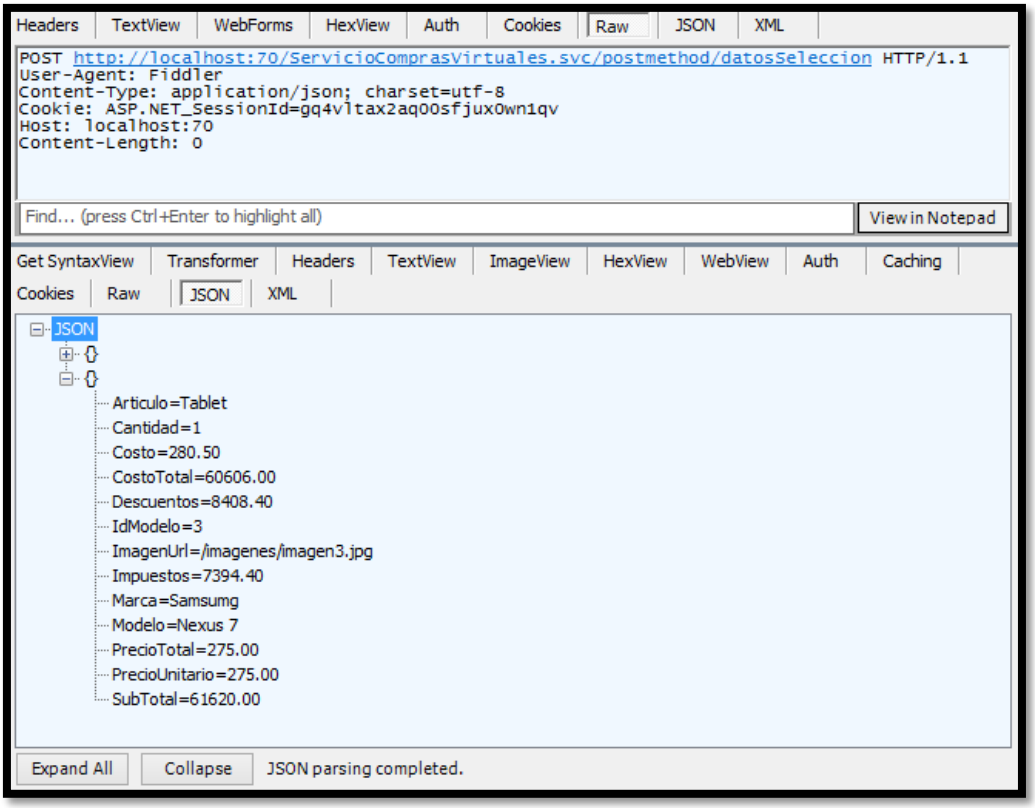

Figura 4.8 Petición POST y respuesta del método `DatosSeleccion`

Figura 4.9 Petición POST y respuesta del método `AgregarProductoCarro`

Además el método `AgregarProductoCarro` alerta si se intenta adquirir una cantidad mayor a la de productos disponibles como se muestra en la **Figura 4.10** y si se intenta adquirir un producto que no cuente con unidades disponibles como se muestra en la **Figura 4.11**.

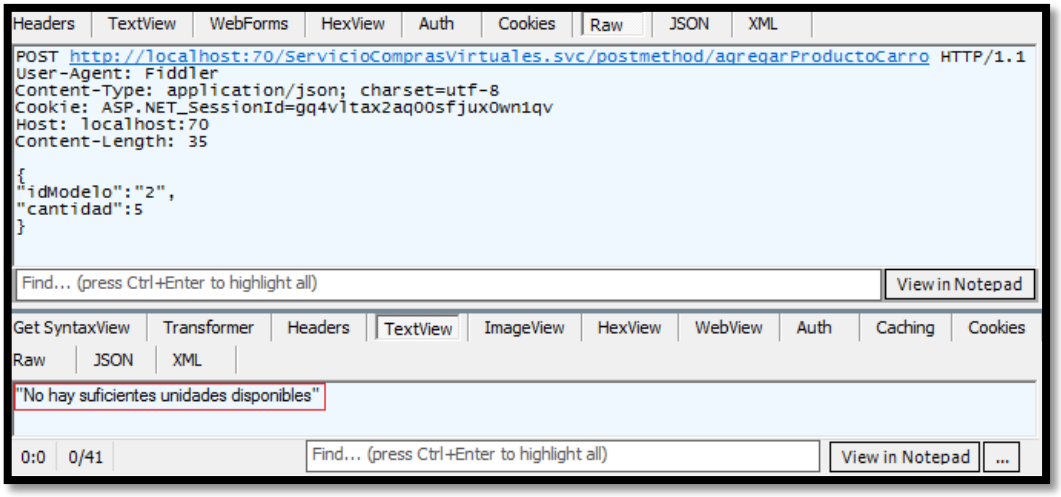

Figura 4.10 Petición POST y respuesta del método `AgregarProductoCarro` (Cantidad mayor a productos disponibles)

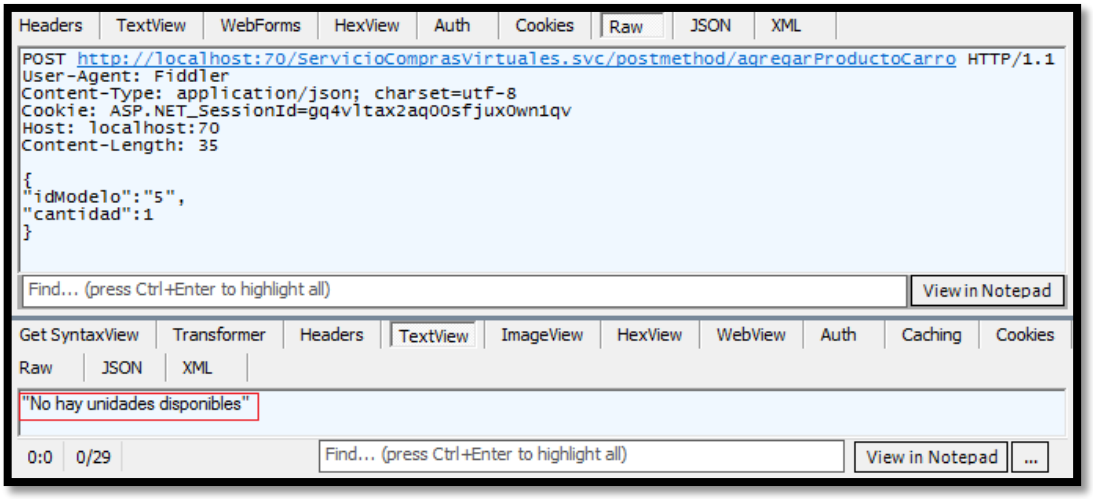

Figura 4.11 Petición POST y respuesta del método `AgregarProductoCarro` (No hay unidades disponibles)

Los métodos del Servicio Web utilizados en la aplicación de escritorio para la administración y facturación, se emplean para: obtener, insertar, actualizar y eliminar información de la base de datos. A continuación se analizan estos métodos tomando como ejemplo los métodos empleados para la gestión de Usuarios.

Al hacer una petición al método `ObtenerUsuarios` el servidor responde con la lista de usuarios activos de la base de datos como se muestra en la **Figura 4.12**

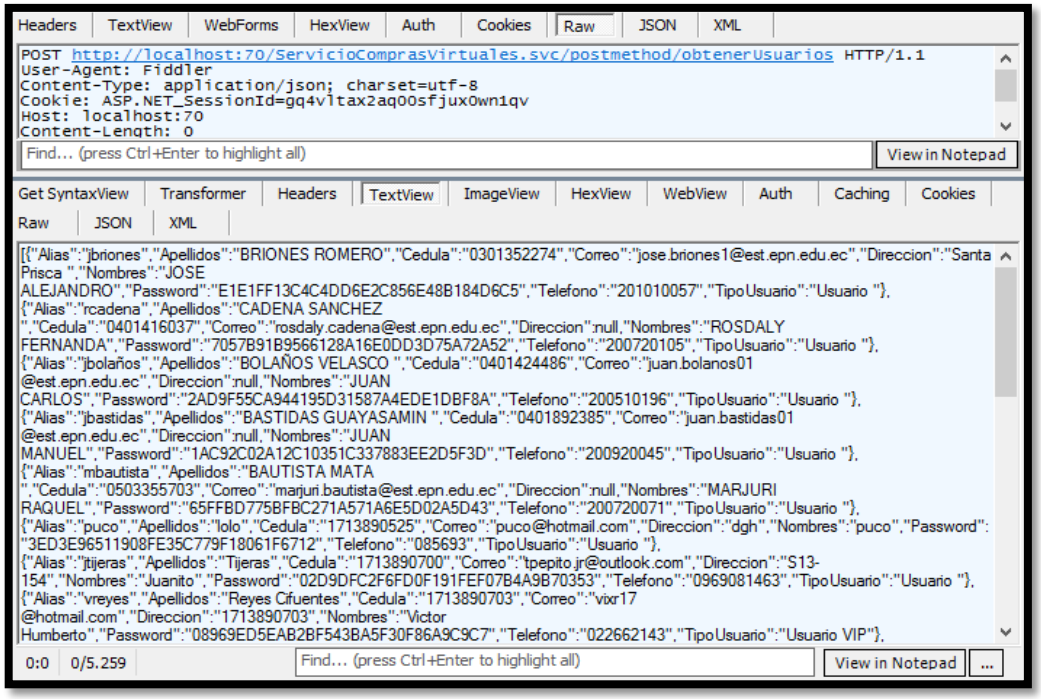

Figura 4.12 Petición POST y respuesta del método `ObtenerUsuarios`

Al hacer una petición al método `EliminarUsuario` se debe enviar la cedula del usuario en la petición y el servidor responde con el mensaje “IngresoCorrecto” indicando que la operación se realizó correctamente como se muestra en la **Figura 4.13**.

Figura 4.13 Petición POST y respuesta del método `EliminarUsuario`

Al hacer una petición al método `InsertarUsuario` se debe enviar los datos del usuario en la petición y el servidor responde con el mensaje “IngresoCorrecto” indicando que la operación se realizó correctamente como se muestra en la **Figura 4.14** .

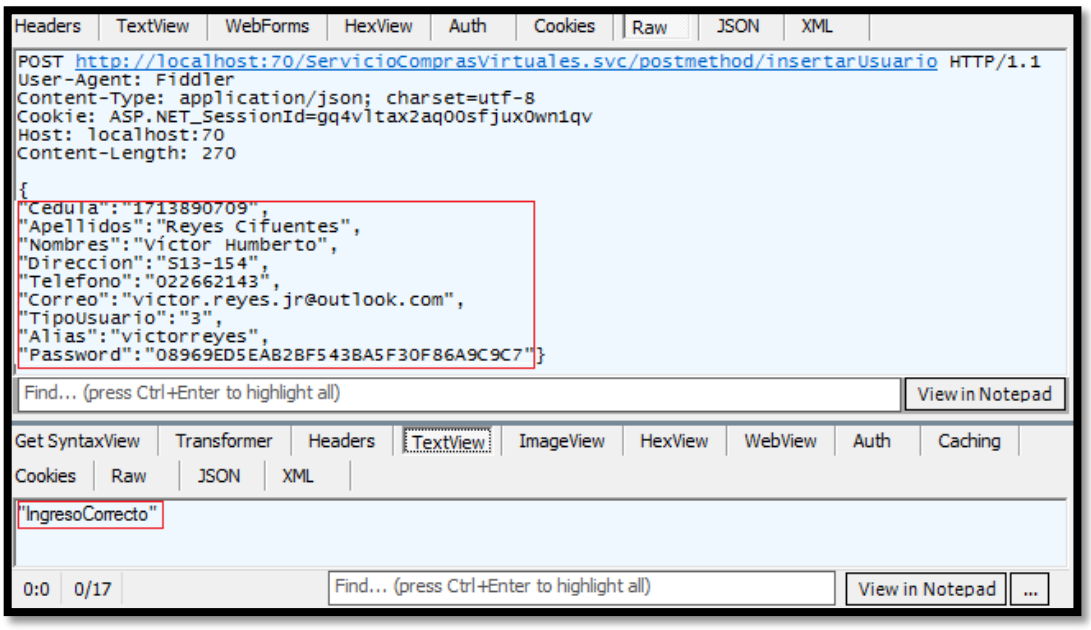

Figura 4.14 Petición POST y respuesta del método `InsertarUsuario`

Se realizaron pruebas de cada uno de los métodos del Servicio Web y las capturas de pantalla de los resultados se presentan en el **Anexo F.1**.

4.2.2 PRUEBAS UNITARIAS APLICACIÓN DE ESCRITORIO [14]

Visual Studio cuenta con un explorador de pruebas que permite incorporar pruebas unitarias en el desarrollo del software. El explorador de pruebas permite ejecutar las pruebas en cualquier momento.

Habitualmente en una prueba unitaria, se presentan tres secciones de código, las cuales son:

- *Arrange*: En esta sección se inicializa objetos y establece los valores que van a ser pasados al método que se desea probar.
- *Act*: En esta sección se invoca al método que se desea probar
- *Assert*: En esta sección se comprueba si el método probado se comporta de la manera prevista.

El **Código 4.1** muestra las secciones de código que contiene una prueba unitaria.

```
 [TestMethod]
//Metodo que prueba el funcionamiento de la autenticación de
//un usuario como Administrador en la aplicación de Escritorio
 | 0 referencias
public void LoginPruebaAdministrador()
{
    //Valores de Prueba
    string alias = "eachig";
    string password = "4D8B47CF6B5640DEB2D51FDFFEE1991A";

    //Resultado esperado
    string esperado = "1";
    Usuario usuario = new Usuario();

    //Metodo que se prueba su funcionamiento
    usuario = Proxy.Login(alias,password);

    //Comprobación de que el resultado sea el esperado
    Assert.AreEqual(esperado, usuario.TipoUsuario);
}
```

Arrange

Act

Assert

Código 4.1 Método de prueba `LoginPruebaAdministrador`

En las pruebas realizadas, para comprobar el correcto funcionamiento de los métodos, se utilizan tres formas de comprobación: la primera en la que se compara que el valor esperado sea el mismo que el obtenido como respuesta del método probado, como se muestra en el **Código 4.1**, la segunda en la que se comprueba que la respuesta del método no sea un valor nulo, como se muestra en el **Código 4.2**, y la tercera que comprueba que la longitud de la respuesta sea la esperada, como se muestra en el **Código 4.3**.

```
//Metodo que prueba el funcionamiento de la obtención
//de los usuarios
[TestMethod]
 | 0 referencias
public void ObtenerUsuariosPrueba()
{
    //Metodo que se prueba su funcionamiento
    List<Usuario> resultado = Proxy.obtenerUsuarios();

    //Comprobación de que el resultado sea el esperado
    Assert.IsNotNull(resultado);
}
```

Código 4.2 Método de prueba `ObtenerUsuariosPrueba`

```
//Metodo que prueba que el metodo de encripción MD5
//devuelva una cadena de 16bytes
[TestMethod]
0 referencias
public void EncriptarMd5Prueba()
{
    //Valores de Prueba
    string valor = "clave";
    //Resultado esperado
    int longitudEsperada = 32;

    //Metodo que se prueba su funcionamiento
    int resultado =  Usuario.EncriptarMd5(valor).Length;

    //Comprueba que la encripción devuelva una cadena
    //de longitud valida
    Assert.AreEqual(longitudEsperada, resultado);
}
```

Código 4.3 Método de prueba `EncriptarMd5Prueba`

Se realizaron métodos de prueba para las principales partes del código de la aplicación de escritorio. La clase de prueba con los métodos realizados se incluye en el **Anexo F.2**.

Los resultados de las pruebas se pueden observar en el explorador de pruebas en el que se muestra: el listado con los nombres de los métodos de prueba, los métodos que pasan la prueba exitosamente se marca de color verde y seleccionando cada uno los métodos de prueba se muestra mayor información de los resultados del método de prueba en la parte derecha del explorador, como se muestra en la **Figura 4.15**.

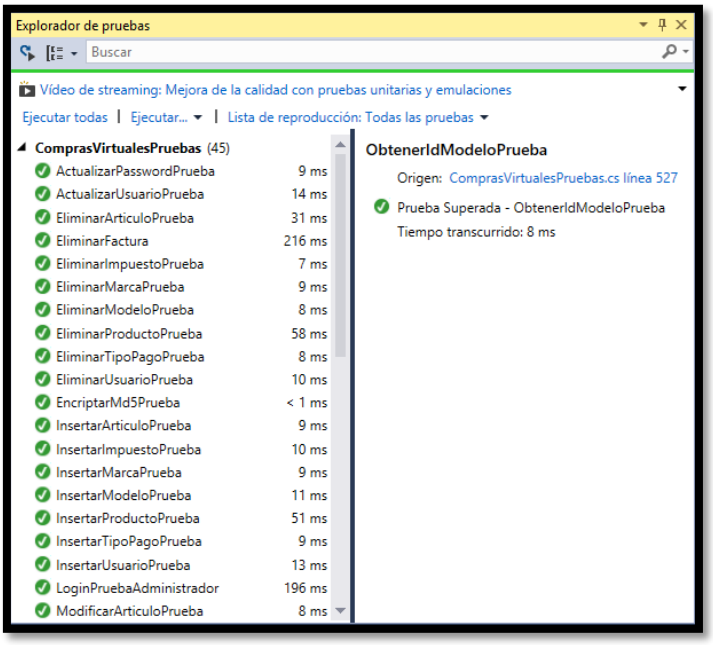

Figura 4.15 Explorador de pruebas

4.2.3 PRUEBAS UNITARIAS DE LA APLICACIÓN MÓVIL [27]

Para realizar las pruebas unitarias de la aplicación móvil se emplea JUnit[92], los métodos se comprueban de manera similar a la aplicación de escritorio, se llama al método a ser comprobado y se verifica que el resultado obtenido sea igual al resultado esperado por medio del uso de *Asserts* como se muestra en el **Código 4.4**, también JUnit cuenta con un navegador en el que se muestran con color verde las pruebas superadas y en la parte inferior información de las causas por las que se produjeron errores en las pruebas realizadas si estos existen como se muestra en la **Figura 4. 16**

```
@Test
//Metodo que prueba el correcto funcionamiento
//de la obtención del hash MD5
 public void pruebaEncriptarMd5() {
    //Variable que almacena el hash
    String hash;
  try {
      //Obtencion del hash MD5
      hash = Usuario.EncripcionMd5("cadena");
  } catch (Exception e) {
      //Se devuelve un hash vacio en caso de error
      hash="";
  }
    //Se obtiene la longitud del hash
    int longitudHash= hash.length();

    //Mediante un Asserto se comprueba que
    //el hash tenga la longitud deseada
    assertTrue( longitudHash==32);
}
```

Assert

Código 4.4 Método de prueba `pruebaEncriptarMd5`

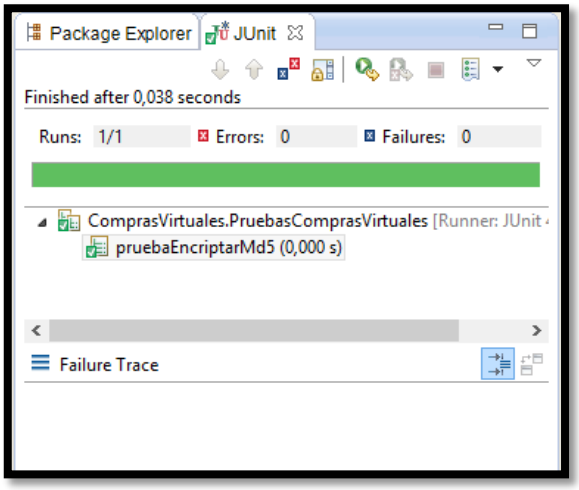

Figura 4. 16 Explorador de pruebas JUnit

[92] **JUnit:** Conjunto de Bibliotecas utilizadas para realizar pruebas de las clases Java [27]

Al igual que la aplicación de escritorio, se realizaron métodos de prueba de las partes principales del código. La clase de prueba con sus métodos realizados se encuentra en el **Anexo F.3**.

4.3 PRUEBAS DE CARGA [14]

Las pruebas de carga se emplean para evaluar cómo responde un programa a diferentes niveles de uso, estas pruebas simulan varios usuarios que pueden acceder al programa simultáneamente.

4.3.1 ANÁLISIS DE CARGA

Para el análisis de la prueba de carga se crea varios escenarios de prueba en los cuales se considera que el servidor recibe peticiones POST, por lo que la pruebas de carga de los distintos escenarios se realizaron en base a una prueba de rendimiento web[93] del Servicio Web.

Los escenarios que se usaron en la prueba de carga son: carga ligera 25 usuarios, carga moderada 100 usuarios, carga fuerte 200 usuarios, carga masiva 1000 usuarios y carga escalonada de 10 a 1000 usuarios. Al realizar las pruebas de carga durante un periodo de prueba de 10 minutos se obtuvieron los resultados que se muestran en la **Tabla 4.1**.

Escenario	Peticiones Realizadas	Peticiones Exitosas	Peticiones/Segundo
Carga Ligera	2.030	2.030	3,38
Carga Moderada	10.150	10.150	16,9
Carga Fuerte	20.300	20.300	33,8
Carga Incremental	103.601	103.601	173
Carga Masiva	202.443	202.443	337

Tabla 4.1 Pruebas de carga Servicio Web

En base a los resultados de las pruebas de carga se tiene que en promedio se realizaron alrededor de 60 pruebas por segundo de los diversos usuarios y todas concluyeron satisfactoriamente como se observa en la **Figura 4.17**, por lo que

[93]**Prueba de rendimiento web:** En este tipo de pruebas, se mide el rendimiento de la aplicación web con una carga de varios usuarios [14]

se concluye que el Servicio Web está en condiciones de trabajar bajo cualquiera de los ambientes de carga establecidos.

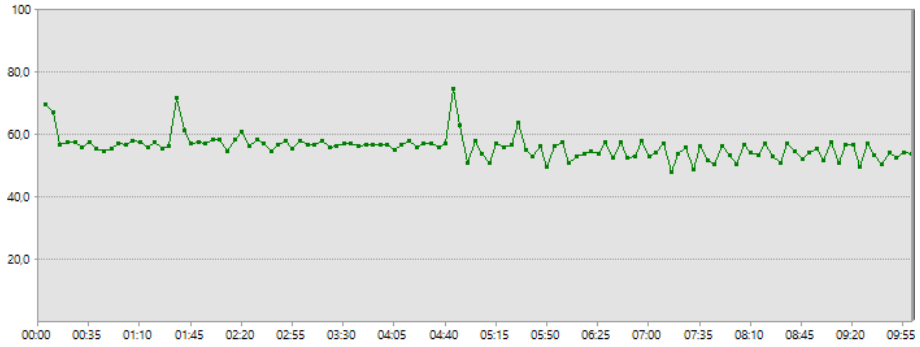

Figura 4.17 Promedio de pruebas por segundo

4.3.2 ANÁLISIS DE REQUERIMIENTOS SERVIDOR

Todas las pruebas fueron realizadas sobre un sistema de 64bits con las características que se muestran en la **Tabla 4.2**.

Componente	Características
RAM	8GB y 1600Mhz
Disco Duro	5400rpm
Procesador	2,4 GHz
Sistema Operativo	Windows 8
Gestor de Base de Datos	SQL Server 2012
Servidor Web	Internet Información Services

Tabla 4.2 Características del equipo utilizado para realizar las pruebas de carga del servidor

4.3.2.1 Análisis de memoria [14]

Para determinar la memoria requerida para el servidor se tomaron en cuenta los siguientes indicadores:

- *Porcentaje de bytes de uso confirmado*: Proporciona la relación entre la memoria y los bytes comprometidos de memoria. La memoria comprometida es la parte de la memoria física que se ha reservado para

poder escribir en el disco, mientras este indicador tenga un menor valor el sistema se va a comportar de una mejor manera.

- *MBytes disponibles*: Indica la cantidad de memoria física en MBytes disponible para los procesos del ordenador, este valor se obtiene sumando el conjunto de memoria vacía, de ceros y de las listas de memoria en espera. Las secciones de memoria que se llenan por ceros son las utilizadas para prevenir que un proceso pueda ver los datos de un proceso posterior. En el caso de que este indicador tenga un mayor valor, se reducen las posibilidades de una falla de memoria.
- *Errores de página por segundo*: Indica el promedio de páginas con errores por segundo, se pueden producir errores leves en los que las fallas se alojan en algún lugar de la memoria física y errores severos en los que el fallo se produce a nivel del disco lo que puede causar retrasos en la respuesta del servidor, por lo que entre menor sea valor de este indicador se tendrá un mejor trabajo del servidor.

Para el análisis de memoria se utilizó una RAM de 12GB y 1600Mhz y se realizaron cinco niveles de carga: carga ligera 25 usuarios, carga moderada 100 usuarios, carga fuerte 200 usuarios, carga masiva 1000 usuarios y carga escalonada de 10 a 1000 usuarios, el resultado del análisis de la memoria física se puede observar en la **Tabla 4.3**

Valores Promedio	Carga Ligera	Carga Moderada	Carga Fuerte	Carga Masiva	Carga Escalonada
% Bytes Confirmados	26,4	26,9	29,3	35,3	28,9
MBytes Disponibles	7924	7896	7350	5862	7380
Errores Pagina/s	3,660	4,68	4,42	4,41	6,31

Tabla 4.3 Resultados de pruebas de carga para memoria

Como se puede observar en la **Tabla 4.3** comparativa cuando se tiene una mayor carga en el servidor los indicadores van a variar hacia valores menos óptimos. Los errores de página de memoria por segundo tienen un valor bajo, dado que se presentan errores leves que no tienen consecuencias significativas, por lo que

se considera que una memoria física con una velocidad de 1600Mhz es adecuada.

Analizando la cantidad de MBytes disponibles se puede observar que en las peores circunstancias, con una carga constante de 1000 usuarios quedará disponible 5,862GB de los 12GB de la memoria de prueba, por lo que se considera que la cantidad de memoria disponible es más que suficiente para un optimó funcionamiento del servidor.

Por medio del porcentaje de bytes confirmados podemos obtener un aproximado de la memoria física que sería requerida para que el servidor funcione correctamente. En el caso extremo de que en el sistema estén 1000 usuarios simultáneos, el Servicio Web consume el 35,3% de la memoria física para recibir y responder las peticiones es decir 4,24GB de los 12GB de la memoria física son utilizadas por el Servicio Web. Tomando en cuenta que para los procesos propios de sistemas operativos como Windows 7, Windows 8, Windows Server 2008 y Windows 2012 es necesario un espacio de memoria entre 512MB y 2GB [14], y considerando un rango de tolerancia, una memoria de 8GB sería suficiente para que el servidor pueda dar soporte hasta 1000 usuarios.

4.3.2.2 Análisis de disco físico [14]

Para determinar las características del disco físico para el servidor, se tomaron en cuenta los siguientes indicadores.

- *Porcentaje de tiempo de disco*: Proporciona el porcentaje de tiempo que el disco está ocupado atendiendo solicitudes de lectura y escritura.
- *Promedio de bytes transferidos del disco*: proporciona el promedio de bytes transferidos hacia o desde el disco durante las operaciones de lectura o escritura.
- *Porcentaje de inactividad*: Proporciona el porcentaje de tiempo que el disco está sin trabajar durante un intervalo de muestreo, este porcentaje se recomienda que sea mayor al 40% y sí su valor es menor al 20% [94] puede causar problemas críticos en el desempeño del servidor.

[94] Estos valores fueron obtenidos en base a las reglas de umbral que se proporcionan en el módulo de pruebas de carga de Visual Estudio 2013

- *Promedio de la longitud de cola del disco*: Proporciona el promedio de las peticiones de lectura y escritura puestas en cola para el disco durante el intervalo de muestreo.
- *Promedio del Tiempo de Transferencia*: Proporciona el promedio en segundos de una transferencia de datos del disco.

Para el análisis del disco se utilizó un disco duro de 1 TB y 5400 RPM y se realizaron cinco niveles de carga: carga ligera 25 usuarios, carga moderada 100 usuarios, carga fuerte 200 usuarios, carga masiva 1000 usuarios y carga escalonada de 10 a 1000 usuarios, el resultado del análisis de la memoria física se puede observar en la **Tabla 4.4**

En base a los resultados de los indicadores, se observa que el disco duro funcionó de manera adecuada durante las pruebas lo que indica que un disco con una velocidad de rotación de 5400rpm funciona correctamente.

Valores Promedio	Carga Ligera	Carga Moderada	Carga Fuerte	Carga Masiva	Carga Escalonada
% Tiempo de Disco	10,9	11,1	11,3	12	11,3
Promedio Bytes Transferidos Disco (bytes/transferencia)	14,424	17,453	21,345	27,643	22,456
%Inactividad Disco	94,4	90,1	87,5	89,4	85,5
Promedio de Longitud de Cola	0,11	0,11	0,12	0,12	0,12
Promedio de Tiempo de Transferencia (s/transferencia)	0,0052	0,0042	0,0040	0,0035	0,0041

Tabla 4.4 Resultados del disco en prueba de carga

4.3.2.3 Análisis del procesador

Para determinar las características del procesador del servidor se tomaron en cuenta los siguientes indicadores.

- *Porcentaje de tiempo de procesador*: Proporciona el porcentaje de tiempo que un procesador está ejecutando una aplicación o un proceso del sistema operativo. Este indicador muestra la actividad del procesador.
- *Porcentaje de tiempo privilegiado*: Proporciona el porcentaje de tiempo que el procesador pasa en modo privilegiado. El modo privilegiado es un

modo de procesamiento diseñado para los componentes del sistema operativo y la manipulación de los controladores de hardware.

- *Porcentaje de tiempo de usuario*: Proporciona el porcentaje de tiempo que el procesador pasa en modo de usuario. El modo de usuario es un modo de procesamiento restringido diseñado para aplicaciones, subsistemas de entorno y subsistemas integrales.

Para el análisis del procesador se utilizó un procesador Intel(R) Core(TM) i7 de 2.4Ghz y se realizaron cinco niveles de carga: carga ligera 25 usuarios, carga moderada 100 usuarios, carga fuerte 200 usuarios, carga masiva 1000 usuarios y carga escalonada de 10 a 1000 usuarios, el resultado del análisis de la memoria física se puede observar en la **Tabla 4.5**.

Como muestran los indicadores el tiempo del procesador se divide en 2: el tiempo que el procesador trabajo en modo privilegiado y el tiempo que el procesador trabaja en modo de usuario.

Valores Promedio	Carga Ligera	Carga Moderada	Carga Fuerte	Carga Masiva	Carga Escalonada
% Tiempo de Procesador	11,01	15,09	17,98	24,6	17,97
% Tiempo Privilegiado	5,99	8,11	9,09	13,4	9,11
% Tiempo de Usuario	5,03	6,98	8,87	11,2	8,82

Tabla 4.5 Resultados del procesador en prueba de carga

Con los resultados arrojados por la prueba de carga del procesador se pudo comprobar que un procesador de 2.4GHz funciona de manera adecuada para cumplir los requerimientos del sistema, dado que el procesador no se encuentra sobrecargado en ningún instante de la prueba.

4.3.2.4 Características del servidor

Con los resultados obtenidos de los análisis de memoria, disco y procesador se obtuvieron las características del servidor que se muestran en la **Tabla 4.6**

Componente	Características
RAM	8GB y 1600Mhz
Disco Duro	5400rpm
Procesador	2,4 GHz

Tabla 4.6 Características del servidor

4.3.3 ANÁLISIS DE REQUERIMIENTOS DISPOSITIVO MÓVIL

La aplicación móvil al usar comunicaciones NFC, tiene como requerimiento mínimo trabajar con el API 14 es decir con la versión 4.0 de Android comercialmente conocida como *Ice Cream Sandwich*.

Adicionalmente es necesario que el dispositivo móvil disponga de 1,64 MB para la instalación del archivo .apk y el almacenamiento de los datos de la aplicación como se muestra en la **Figura 4.18**

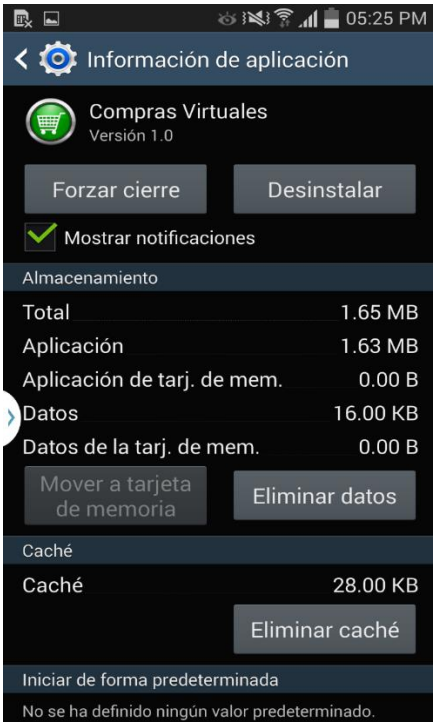

Figura 4.18 Información de la aplicación Compras Virtuales

4.3.4 ANÁLISIS DE REQUERIMIENTOS DEL EQUIPO DE ESCRITORIO

El equipo de escritorio para poder consumir el Servicio Web basado en REST, es necesario tener instalado un sistema operativo Windows 7 o superior y tener el framework 4.5 de .Net. Adicionalmente, es necesario que el equipo de escritorio disponga de 2 MB para la instalación y el almacenamiento de los datos de la aplicación

4.4 PRUEBAS DE FUNCIONALIDAD

4.4.1 PRUEBAS DE ACEPTACIÓN

Estas pruebas se realizan en base a las historias de usuario de cada iteración y evalúan el grado de calidad del software en base al cumplimento de requerimientos establecidos en fases anteriores.

4.4.1.1 Elementos de una prueba de aceptación

El formato a llevar en cada prueba de aceptación se explica a continuación:

- *Número*: Código que identifica de forma única a cada prueba de aceptación. Está conformado por las siglas Prueba de Aceptación (PA), número de iteración, la letra M si la historia pertenece a la aplicación Móvil, E si la historia pertenece a la aplicación de escritorio y ME si la historia aplica tanto a la aplicación móvil como a la aplicación de escritorio, adicionalmente las historias de usuario que implique la creación de métodos en el Servicio Web se asignan con la letra S, por último el número de prueba en la iteración.
- *Historia de usuario*: Indica la historia de usuario al que corresponde esta prueba de aceptación.
- *Nombre de la prueba*: Nombre descriptivo para identificar el ámbito de la prueba.
- *Descripción*: Aquí se describe en breves palabras lo que debe cumplir la prueba.

- *Condiciones de ejecución*: Indica las condiciones que se deben cumplir para realizar la prueba.
- *Pasos de ejecución*: Indica los pasos a seguir para realizar la prueba.
- *Resultado esperado*: Indica el resultado esperado de la prueba
- *Evaluación de la prueba*: Aquí se indica si se cumplió o no el objetivo de la prueba.

4.4.1.2 Primera iteración

En la primera iteración se agrupan las pruebas de aceptación que permiten comprobar el funcionamiento básico del sistema. La **Tabla 4.7** muestra la prueba de aceptación Desarrollo de la base de datos, la **Tabla 4.8** y la **Tabla 4.9** muestra la prueba de aceptación Autenticación, la **Tabla 4.10** muestra la prueba de aceptación Registro de Nuevos Usuarios, la **Tabla 4.11** muestra la prueba de aceptación Configuración del Perfil, la **Tabla 4.12** muestra la prueba de aceptación Cambio de Contraseña y la **Tabla 4.13** muestra la prueba de aceptación Gestión de Usuarios.

Prueba de Aceptación	
Número:	PA1B1
Historia de Usuario:	HU1B1-Desarrollo de la base de datos
Nombre:	Desarrollo de la base de datos
Descripción:	Para almacenar la información del sistema se desarrolló una base de datos de acuerdo al modelo de datos elaborado.
Condiciones de Ejecución:	Tener acceso a la base de datos mediante un usuario con credenciales válidas y establecer conexión con el Servicio Web.
Pasos de Ejecución:	Acceder a la base de datos mediante el Servicio Web.
Resultado Esperado:	El Servicio Web podrá utilizar los servicios de la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido.

Tabla 4.7 Prueba de aceptación Desarrollo de la base de datos

Prueba de Aceptación	
Número:	PA1MES2
Historia de Usuario:	HU1MES2-Autenticación
Nombre:	Autenticación
Descripción:	El usuario puede acceder al sistema ingresando un nombre de usuario y una contraseña. Al comprobar que los datos son válidos se establece una sesión para el usuario

Tabla 4.8 Prueba de aceptación Autenticación

Prueba de Aceptación	
Condiciones de Ejecución:	Conexión con el Servidor Web. Debe existir un usuario con credenciales validas en la base de datos.
Pasos de Ejecución:	Acceder a la Aplicación Móvil o de escritorio En la pantalla que se despliega se ingresa el usuario y el *password* Click en el botón “Aceptar”.
Resultado Esperado:	Se accede a la aplicación y se crea una sesión del usuario autenticado.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.9 Prueba de aceptación Autenticación (continuación)

Prueba de Aceptación	
Número:	PA1MS3
Historia de Usuario:	HU1MS3-Registro de Nuevos Usuarios
Nombre:	Registro de Nuevos Usuarios
Descripción:	El Cliente puede registrarse en la Aplicación Móvil
Condiciones de Ejecución:	Conexión con el Servidor Web.
Pasos de Ejecución:	Se ingresa a la Aplicación. Presionar el botón Registrarse. Ingresar todos los datos Correctamente Click en el botón “Aceptar”.
Resultado Esperado:	Se crea un nuevo registro en la tabla `Usuario` en la base de datos con los datos ingresados desde la aplicación móvil.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.10 Prueba de aceptación Registro de Nuevos Usuarios

Prueba de Aceptación	
Número:	PA1MES4
Historia de Usuario:	HU1MES4-Configuración del Perfil
Nombre:	Configuración del Perfil
Descripción:	El usuario puede modificar sus datos personales
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Aplicación de escritorio: Seleccionar la opción de Cuenta Personal del Menú Principal. Hacer Click sobre la opción de Datos Personales. Modificar los datos deseados en los distintos campos del formulario. Click en Aceptar. Aplicación Móvil: Seleccionar la opción Configurar perfil del menú principal. Modificar los datos deseados en los distintos campos de la actividad. Click en Aceptar.
Resultado Esperado:	Se actualizan los datos del usuario en la base de datos con la información ingresada en las aplicaciones móvil o de escritorio.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.11 Prueba de aceptación Configuración del Perfil

Prueba de Aceptación	
Número:	PA1MES5
Historia de Usuario:	HU1MES5-Cambio de Contraseña
Nombre:	Cambio de Contraseña
Descripción:	El usuario puede modificar la contraseña de su cuenta
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Aplicación de escritorio: Seleccionar la opción Cuenta Personal del Menú Principal. Hacer click sobre la opción Cambiar Contraseña. Ingresar la contraseña en el formulario. Confirmar la contraseña Click en Aceptar. Aplicación Móvil: Seleccionar la opción Cambiar Contraseña del Menú Principal. Ingresar la contraseña. Confirmar la contraseña. Click en Aceptar.
Resultado Esperado:	Se actualiza el *password* del usuario en la base de datos con el *hash* MD5 calculado de los datos ingresados.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.12 Prueba de aceptación Cambio de Contraseña

Prueba de Aceptación	
Número:	PA1ES5
Historia de Usuario:	HU1ES5-Gestión de Usuarios
Nombre:	Gestión de Usuarios
Descripción:	El usuario con perfil de administrador puede crear, modificar y eliminar usuarios. Además, este usuario podrá modificar el perfil de los usuarios.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción de Administración del Menú Principal. Hacer click sobre la opción Usuarios. Seleccionar de la lista, el usuario de interés. Escoge añadir, modificar o eliminar un usuario. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla Usuario en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.13 Prueba de aceptación Gestión de Usuarios

4.4.1.3 Segunda iteración

En la segunda iteración se agrupan las pruebas de aceptación que permiten comprobar el funcionamiento del carro de compras virtual. La **Tabla 4.14** muestra la prueba de aceptación Información del Artículo, la **Tabla 4.15** muestra la prueba

de aceptación Selección de Artículos, la **Tabla 4.16** y la **Tabla 4.17** muestra la prueba de aceptación Lista del carro de compras, la **Tabla 4.18** muestra la prueba de aceptación Modificación de la Lista del Carro de Compras y la **Tabla 4.19** muestra la prueba de aceptación Finalizar la Selección de Artículos.

Prueba de Aceptación	
Número:	PA2MS1
Historia de Usuario:	HU2MS1-Información del Artículo
Nombre:	Información del Artículo
Descripción:	Al acercar el dispositivo móvil al tag, se desplegará una nueva actividad con la información artículo.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción de carro de compras del menú principal. Pasar el dispositivo móvil por el tag.
Resultado Esperado:	Al pasar el tag por el dispositivo móvil se abre una nueva actividad con los datos del artículo identificado por el id almacenado en el tag.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.14 Prueba de aceptación Información del Artículo

Prueba de Aceptación	
Número:	PA2MS2
Historia de Usuario:	HU2MS1-Selección de Artículos
Nombre:	Selección de Artículos
Descripción:	El usuario puede leer la información del artículo, seleccionar el número de productos a adquirir y agregarlos al carro de compras si lo desea.
Condiciones de Ejecución:	Haber obtenido la información de un artículo mediante un tag NFC. Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la cantidad de Productos deseados Presionar el botón Agregar al Carro
Resultado Esperado:	Se agregan los productos al carro de compras (se agregan registros a la tabla `Producto_seleccion`, cada uno de estos registros está asociado a una Selección)
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.15 Prueba de aceptación Selección de Artículos

Prueba de Aceptación	
Número:	PA2MS3
Historia de Usuario:	HU2MS3-Lista del Carro de Compras
Nombre:	Lista de Carro del Compras.
Descripción:	El cliente dispone de una actividad donde puede visualizar los artículos agregados al carro y la cantidad de los mismos. También se presenta información resumida de los valores a cancelar hasta el momento.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar carro de compras en el Menú Principal

Tabla 4.16 Prueba de aceptación Lista del Carro de Compras

Prueba de Aceptación	
Resultado Esperado:	Se mostrará en la actividad carro de compras de la aplicación móvil con una lista de artículos (Obtenida de la base de datos) que están en el carro de compras del usuario autenticado con sus valores mediante un `listView`.
Evaluación de la prueba:	Resultado esperado conseguido
Pasos de Ejecución:	Seleccionar carro de compras en el Menú Principal
Resultado Esperado:	Se mostrará en la actividad carro de compras de la aplicación móvil con una lista de artículos (Obtenida de la base de datos) que están en el carro de compras del usuario autenticado con sus valores mediante un `listView`.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.17 Prueba de aceptación Lista del Carro de Compras (continuación)

Prueba de Aceptación	
Número:	PA2MS4
Historia de Usuario:	HU2MS4- Modificación de la Lista del Carro de Compras.
Nombre:	Modificación de la Lista del Carro de Compras
Descripción:	El usuario puede: disminuir, aumentar o eliminar los productos de un artículo determinado mediante la aplicación.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Haber accedido a la actividad carro de compras Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar Carro de compras en el Menú Principal Seleccionar un artículo del carro de compras, presionando el elemento deseado por 3 segundos. En el menú que se despliega seleccionar agregar, disminuir o eliminar según la operación deseada.
Resultado Esperado:	Se aumentará o disminuirá la cantidad de productos de un artículo o se eliminará el artículo seleccionado (Se agregarán y eliminarán registros en tabla `Producto_Seleccion`)
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.18 Prueba de aceptación Modificación de la Lista del Carro de Compras

Prueba de Aceptación	
Número:	PA2M5
Historia de Usuario:	HU2M5- Finalizar la Selección de Artículos
Nombre:	Finalizar la Selección de Artículos
Descripción:	El cliente puede finalizar la selección de artículos, se registrará la transacción y se le indicará el valor a cancelar en caja en forma de reporte.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Haber accedido a la actividad del carro de compras Conexión con el Servidor Web.
Pasos de Ejecución:	Click sobre el botón Finalizar Selección.
Resultado Esperado:	El estado de la selección pasará a Finalizada en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.19 Prueba de aceptación Finalizar la Selección de Artículos

4.4.1.4 Tercera iteración

En la tercera iteración se agrupan todas las pruebas de aceptación que permiten comprobar la correcta administración de los registros de la base de datos. La **Tabla 4.20** muestra la prueba de aceptación Gestión de Artículos, la **Tabla 4.21** y la **Tabla 4.22** muestra la prueba de aceptación Gestión de Productos, la **Tabla 4.23** muestra la prueba de aceptación Gestión de Formas de Pago, la **Tabla 4.24** muestra la prueba de aceptación Gestión de Impuestos, la **Tabla 4.25** muestra la prueba de aceptación Gestión de Marcas y la **Tabla 4.26** muestra la prueba de aceptación Gestión de Modelos.

Prueba de Aceptación	
Número:	PA3ES1
Historia de Usuario:	HU3ES1- Gestión de Artículos
Nombre:	Gestión de Artículos
Descripción:	El administrador puede agregar nuevos artículos, modificarlos y eliminarlos.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción de Administración del Menú Principal. Seleccionar la opción Otros. Hacer click sobre la opción Artículos. Seleccionar de la lista el artículo de interés. Escoger añadir, Modificar o eliminar un artículo. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla `Articulo` en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.20 Prueba de aceptación Gestión de Artículos

Prueba de Aceptación	
Número:	PA3ES2
Historia de Usuario:	HU3ES2- Gestión de Productos
Nombre:	Gestión de Productos
Descripción:	El Administrador puede agregar nuevos productos, modificarlos y eliminarlos.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción administración del Menú Principal. Hacer click sobre la opción Productos. Seleccionar de la lista el producto de interés. Escoge añadir, Modificar o eliminar un producto. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.

Tabla 4.21 Prueba de aceptación Gestión de Productos

Prueba de Aceptación	
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla `Producto` en la base de datos, se pueden agregar productos por bloques al especificar una cantidad mayor a 1.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.22 Prueba de aceptación Gestión de Productos (continuación)

Prueba de Aceptación	
Número:	PA3ES3
Historia de Usuario:	HU3ES3- Gestión de Formas de Pago
Nombre:	Gestión de Formas de Pago
Descripción:	El Administrador puede agregar nuevas formas de pago, modificarlas y eliminarlas.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción Administración del Menú Principal. Seleccionar la opción Otros. Hacer click sobre la opción Formas de Pago. Se selecciona de la lista la forma de pago de interés. Escoge añadir, modificar o eliminar una forma de pago. Ingresa o modifica los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla `Tipo_pago` de en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.23 Prueba de aceptación Gestión de Formas de Pago

Prueba de Aceptación	
Número:	PA3ES4
Historia de Usuario:	HU3ES4-Gestión de Impuestos
Nombre:	Gestión de Impuestos
Descripción:	El Administrador puede agregar nuevos impuestos, modificarlos y eliminarlos.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción de Administración del Menú Principal. Seleccionar la opción Otros. Hacer click sobre la opción de Impuestos. Seleccionar de la lista el impuesto de interés. Escoger añadir, Modificar o eliminar un impuesto. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla `Impuesto` de en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.24 Prueba de aceptación Gestión de Impuestos

Prueba de Aceptación	
Número:	PA3ES5
Historia de Usuario:	HA3ES5- Gestión de Marcas
Nombre:	Gestión de Marcas
Descripción:	El Administrador puede agregar nuevas marcas, modificarlas y eliminarlas.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción Administración del Menú Principal. Selecciona la opción Otros. Hacer click sobre la opción Marcas. Seleccionar de la lista la marca de interés. Escoger añadir, Modificar o eliminar una marca. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla `Marca` en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.25 Prueba de Aceptación Gestión de Marcas

Prueba de Aceptación	
Número:	PA3ES6
Historia de Usuario:	HU3ES6- Gestión de Modelos
Nombre:	Gestión de Modelos
Descripción:	El Administrador puede agregar nuevos modelos, modificarlos y eliminarlos.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de administrador. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción Administración del Menú Principal. Hacer click sobre la opción Modelos. Seleccionar de la lista el modelo de interés. Escoge añadir, Modificar o eliminar un modelo. Ingresar o modificar los datos en los distintos campos del formulario si es necesario. Click en el botón Aceptar.
Resultado Esperado:	Se ingresan, actualizan o eliminan los registros de la tabla Modelo en la base de datos.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.26 Prueba de Aceptación Gestión Modelos

4.4.1.5 Cuarta iteración

En esta iteración se agrupan todas las pruebas de aceptación que permiten comprobar la facturación. La **Tabla 4.27** muestra la prueba de aceptación Nota de venta, la **Tabla 4.28** muestra la prueba de aceptación Facturación, la **Tabla 4.29** muestra la prueba de aceptación Imprimir Factura y la **Tabla 4.30** muestra la prueba de aceptación Historial de facturas.

Prueba de Aceptación	
Número:	PA4MS1
Historia de Usuario:	HU4MS1- Nota de venta
Nombre:	Nota de venta
Descripción:	Cuando el cliente finaliza el proceso de selección de productos en la aplicación móvil, el cliente visualizará una nota de venta con los productos que selecciono y un mensaje informativo para que se acerque a pagar a una caja.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales de válidas. Haber finalizado la compra en la actividad Carro de Compras de la aplicación móvil Conexión con el Servidor Web.
Pasos de Ejecución:	Acceder a la actividad Carro de Compras
Resultado Esperado:	Se observará en la actividad del Carro de compras un reporte en forma de nota de venta en el que se mostrará los modelos de artículos seleccionados y la información del subtotal, impuestos, descuentos y valor total a pagar.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.27 Prueba de aceptación Nota de venta

Prueba de Aceptación	
Número:	PA4ES2
Historia de Usuario:	HU4ES2- Facturación
Nombre:	Facturación
Descripción:	El usuario tendrá en la aplicación de escritorio una lista donde podrá ver todos los clientes que finalizaron la selección del producto. El usuario escogerá de la lista, la selección deseada y luego seleccionará el método de pago para emitir la factura.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción Facturación del Menú Principal. Seleccionar la opción Crear Factura. Seleccionar de la lista, la selección del usuario de interés. Seleccionar el tipo de Pago. Si se desea facturar a nombre de otro usuario seleccionar esa opción Click sobre el botón Facturar
Resultado Esperado:	Se añade un registro a la tabla Factura de la base de datos que estará relacionado con la selección escogida, al solicitar datos de la selección y del usuario, estos se muestran en el formulario de Factura.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.28 Prueba de aceptación Facturación

Prueba de Aceptación	
Número:	PA4E3
Historia de Usuario:	HU4E3- Imprimir Factura
Nombre:	Impresión de la factura
Descripción:	Impresión de la factura para entrega al usuario.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Acceder al formulario de Factura. Conexión con el Servidor Web.
Pasos de Ejecución:	Hacer click sobre el botón con el grafico de una impresora.
Resultado Esperado:	Se transforma el formulario en un bitmap, se guarda este como imagen para respaldo y se manda el bitmap a un dialogo de impresión.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.29 Prueba de aceptación Imprimir Factura

Prueba de Aceptación	
Número:	PA4ES4
Historia de Usuario:	HU4ES4- Historial de Facturas
Nombre:	Historial de facturas
Descripción:	El usuario puede acceder a un historial de facturas para poder realizar anulaciones o reimpresiones.
Condiciones de Ejecución:	Haberse autenticado en la aplicación con credenciales válidas. Conexión con el Servidor Web.
Pasos de Ejecución:	Seleccionar la opción de Facturación del Menú Principal. Seleccionar la opción Crear Factura. Seleccionar de la lista, la factura de interes. Click sobre el botón Anular o Reimprimir.
Resultado Esperado:	Si se selecciona la opción de Anular se anula la factura y la selección regresa al estado de finalizada en la base de datos para poderla volver a facturar en caso de que se desea realizar algún cambio de datos. En caso de seleccionar reimprimir se imprime una copia de la factura emitida.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.30 Prueba de aceptación Historial de Facturas

4.4.1.6 Quinta iteración

En esta iteración se agrupan las pruebas de aceptación de las opciones de ayuda de las aplicaciones. La **Tabla 4.31** muestra la prueba de aceptación Acerca de y la **Tabla 4.32** muestra la prueba de aceptación Ayuda.

Prueba de Aceptación	
Número:	PA5ME1
Historia de Usuario:	HU5ME1- Acerca de
Nombre:	Acerca de
Descripción:	Se incluirá en cada aplicación información de la empresa propietaria del sistema así como información de los programadores que elaboraron el sistema.
Condiciones de Ejecución:	Ninguna
Pasos de Ejecución:	Aplicación Móvil Seleccionar la opción Acerca de en el menú principal Aplicación de escritorio Seleccionar la opción de Ayuda del Menú Principal. Selecciona la opción de Acerca de.
Resultado Esperado:	Se mostrará una pantalla con información acerca la aplicación y de los programadores de la misma.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.31 Prueba de aceptación Acerca de

Prueba de Aceptación	
Número:	PA5ME2
Historia de Usuario:	HU5ME2- Ayuda
Nombre:	Ayuda
Descripción:	Se incluirá en cada aplicación un pequeño manual de usuario para conocer la forma de uso de la aplicación.
Condiciones de Ejecución:	Ninguna
Pasos de Ejecución:	Aplicación Móvil Seleccionar Ayuda en el menú principal. Aplicación de escritorio Seleccionar la opción de Ayuda del Menú Principal. Seleccionar la opción de Manual de Usuario.
Resultado Esperado:	Se mostrara una pantalla con un manual de usuario.
Evaluación de la prueba:	Resultado esperado conseguido

Tabla 4.32 Prueba de aceptación Ayuda

4.4.2 PRUEBAS FINALES

4.4.2.1 Ambiente de pruebas

Para el ambiente de pruebas se utilizan los equipos que se detallan a continuación:

- Para el servidor se hará uso de un computador el cual alojará la base de datos y el servidor WCF.
- Para los clientes Android se utiliza una Tablet y un celular, estos dispositivos cuentan con el lector de NFC.
- Se utilizan 6 tags NFC.
- Para la aplicación de escritorio se utilizan dos computadores los cuales van a contener: el uno la aplicación para la gestión de inventario de productos (perfil Administrador) y el otro la facturación (perfil Cajero).
- Para la red de comunicaciones se utiliza un *Access Point* y un conmutador.
- En la base de datos se cargan 10 usuarios, 6 artículos y 30 productos para probar los diferentes perfiles y realizar las pruebas del sistema.

Un esquema del ambiente de pruebas planteado se muestra en la **Figura 4.19**. Los equipos utilizados en el ambiente de pruebas se muestran en la **Tabla 4.33**. El software empleado en cada equipo se muestra en la

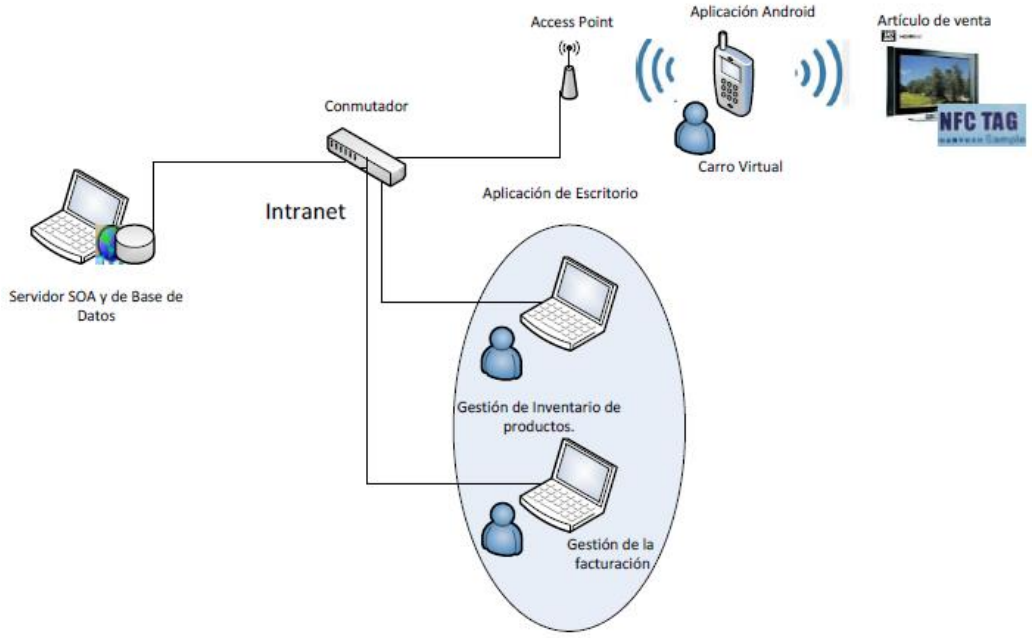

Figura 4.19 Ambiente de pruebas

EQUIPO	CARACTERÍSTICAS
Servidor SOA y de Base de Datos	- Laptop HP 1000 - Procesador Intel Core i5 2,2 GHz - Memoria RAM 8,00 GB - Disco Duro SATA 500 GB 5400 RPM - Sistema Operativo Windows 8.1 de 64 bit
Computador para la Gestión de inventario de productos	- Laptop HP Envy 15 - Procesador Intel Core i5 2,4 GHz - Memoria RAM 12 GB - Disco Duro SATA 1000 GB - Sistema Operativo Windows 8.1 de 64 bit
Computador para la Gestión de Facturación	- Laptop HP Envy 15 - Procesador Intel Core i5 2,4 GHz - Memoria RAM 12 GB - Disco Duro SATA 1000 GB - Sistema Operativo Windows 8.1 de 64 bit
Dispositivo Móvil	- Smartphone Samsung Galaxy Note 3 - Tecnología NFC incluida - Sistema operativo Android versión 4.4.2
Tags NFC	- Familia de Tags MifireUltraligth - Mínimo 28 bytes de almacenamiento

Tabla 4.33 Equipos utilizados en el ambiente de pruebas

EQUIPO	SOFTWARE REQUERIDO
Servidor SOA y de Base de Datos	- SQL server 2012 - Servidor IIS versión 8.0 - .NET framework versión 4.0 - Servicio Web "ComprasVirtuales"[95]
Computador para la Gestión de inventario de productos	- .NET framework versión 4.0 - Aplicación "ComprasVirtuales"
Computador para la Gestión de Facturación	- .NET framework versión 4.0 - Aplicación "ComprasVirtuales"
Dispositivo Móvil	- Aplicación "ComprasVirtuales"

Tabla 4.34 Software empleado

4.4.2.2 Pruebas aplicación de escritorio

4.4.2.2.1 Autenticación

Al probar la autenticación de la aplicación de escritorio se presentan tres posibilidades: autenticarse con perfil de Administrador, autenticarse con perfil de Cajero o ingresar datos no válidos y no poder acceder a los componentes de la aplicación.

Al momento de acceder a la aplicación con datos validos se puede observar cómo se habilitan los componentes: Administración, Cuenta Personal y Facturación y además en la parte inferior del formulario principal se muestran los datos del usuario que ingreso a la aplicación así como también su perfil como se muestra en la **Figura 4.20**

En el caso de ingresar datos incorrectos se presenta un mensaje que indica que el usuario o la contraseña no son correctos y al único componente de la aplicación al que es posible acceder es al de ayuda como se muestra en la **Figura 4.21**

[95] **ComprasVirtuales:** Software desarrollado en el presente proyecto

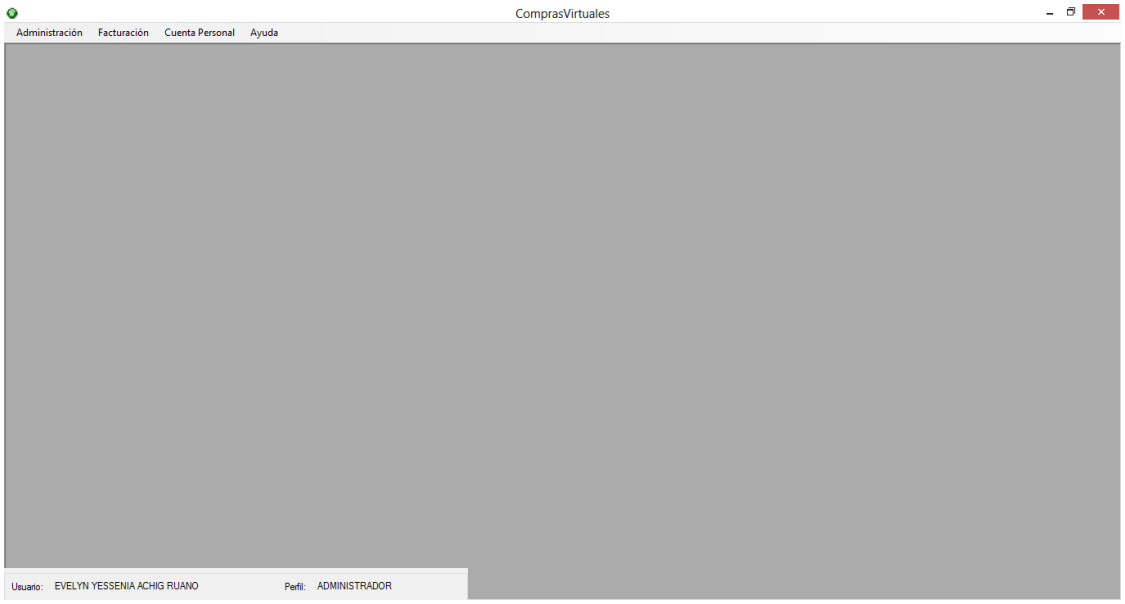

Figura 4.20 Formulario Principal cuando la autenticación es correcta

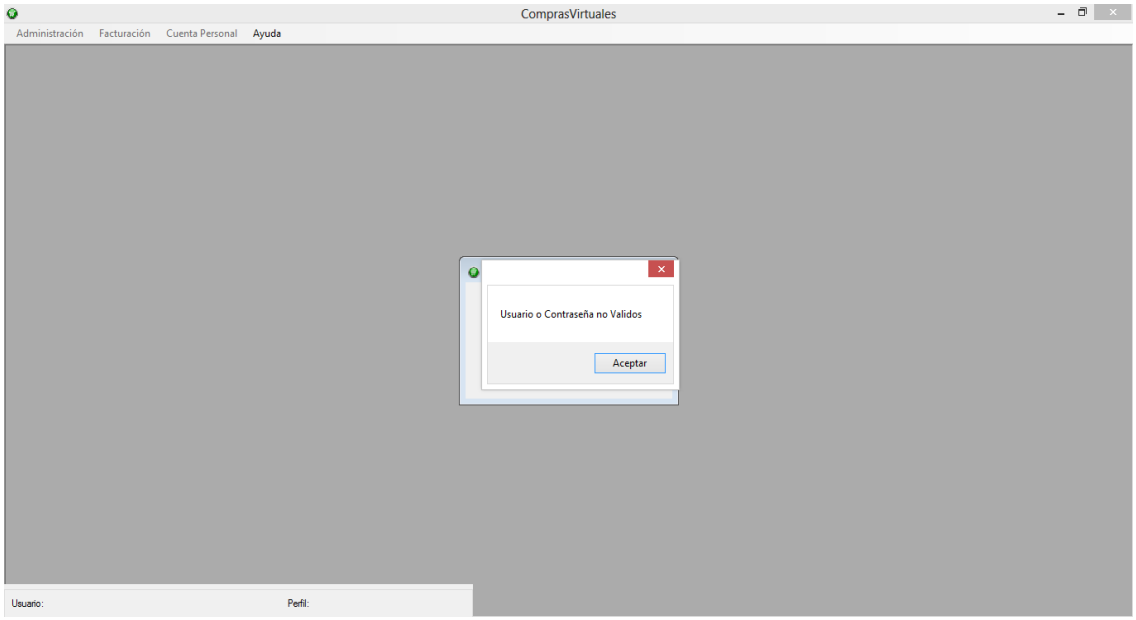

Figura 4.21 Formulario Principal cuando la autenticación es incorrecta

4.4.2.2.2 Administración

Al probar la aplicación se comprueba que este componente permite gestionar: usuarios, productos, modelos, marcas, artículos. Este componente del sistema permite: crear nuevos registros, modificarlos y borrarlos.

Por ejemplo al ingresar a la gestión de usuarios se puede observar la lista de los usuarios del sistema, al seleccionar el usuario deseado es posible modificarlo o

eliminarlo y si se desea también se puede crear un nuevo usuario. La **Figura 4.22** muestra el componente gestión de usuarios.

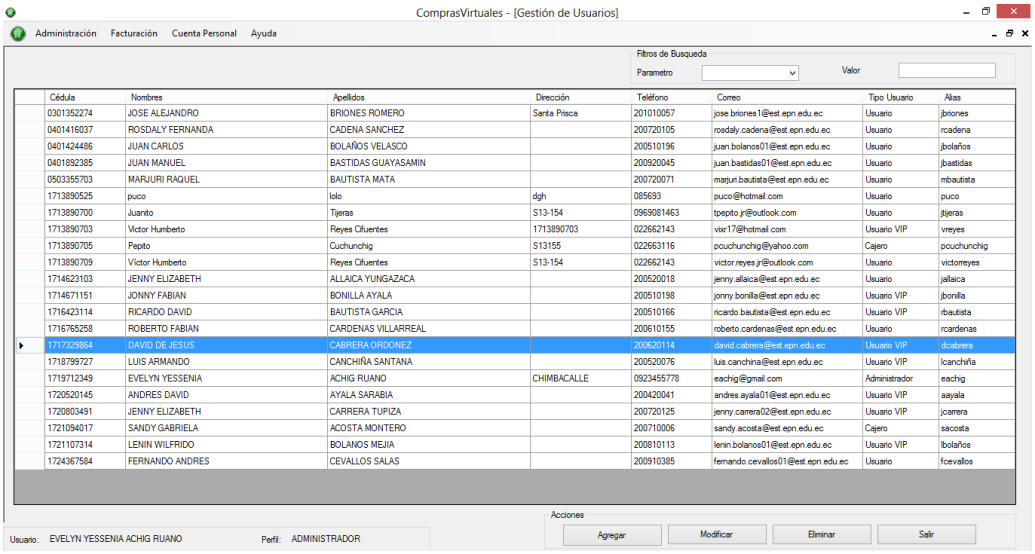

Figura 4.22 Gestión de Usuarios

Al presionar el botón Agregar se despliega un nuevo formulario llamado “Agregar Usuario” con los campos en blanco para ingresar los datos del nuevo usuario como se muestra en la **Figura 4.23**, después de ingresar los datos se presiona aceptar y si los datos se ingresan correctamente, aparece un mensaje con el texto “Actualización Correcta”.

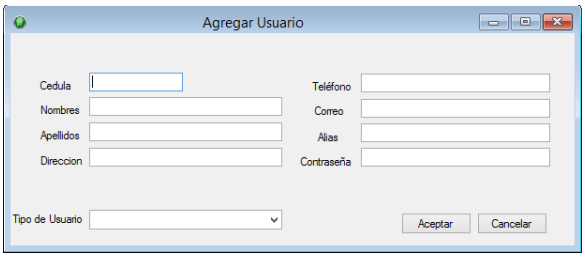

Figura 4.23 Agregar Usuario

Al presionar el botón Modificar se despliega un nuevo formulario llamado “Modificar Usuario” con los datos del usuario seleccionado para modificarlos como se muestra en la **Figura 4.24**, cuando se realizan los cambios se presiona aceptar y aparece un mensaje con el texto “Actualización Correcta”.

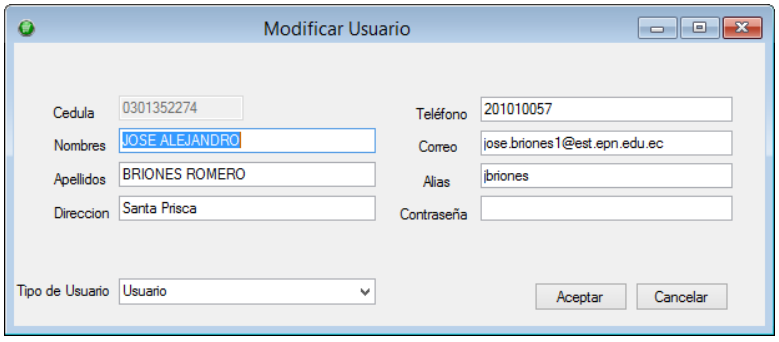

Figura 4.24 Modificar Usuario

Al presionar el botón Eliminar se despliega un nuevo formulario llamado “Eliminar Usuario” que contiene los datos del usuario seleccionado para confirmar que el usuario es el mismo que se quiere eliminar, como se muestra en la **Figura 4. 25**, se presiona Aceptar y se muestra un mensaje con el texto “Eliminación Correcta”.

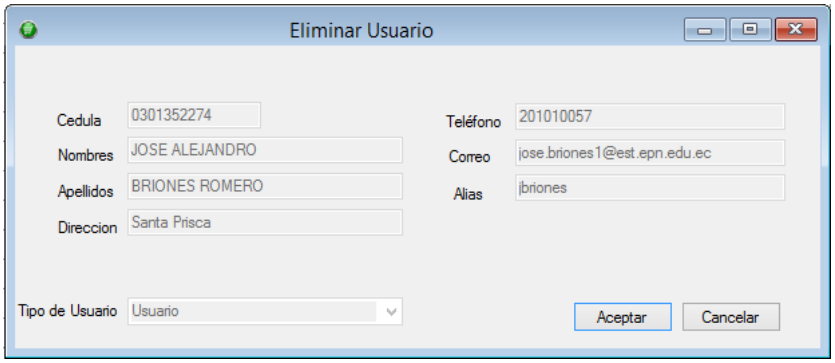

Figura 4. 25 Eliminar Usuario

4.4.2.2.3 Facturación

Al realizar las pruebas del componente Facturación, se muestra que es posible emitir la factura con los datos de un cliente diferente al que realizo la selección, seleccionando la casilla “Cambiar datos del Cliente”. También se puede seleccionar el tipo de pago deseado como se muestra en la **Figura 4.26**. Para emitir la factura se selecciona el usuario deseado, se presiona el botón Facturar y se obtiene una factura como la de la **Figura 4.27**.

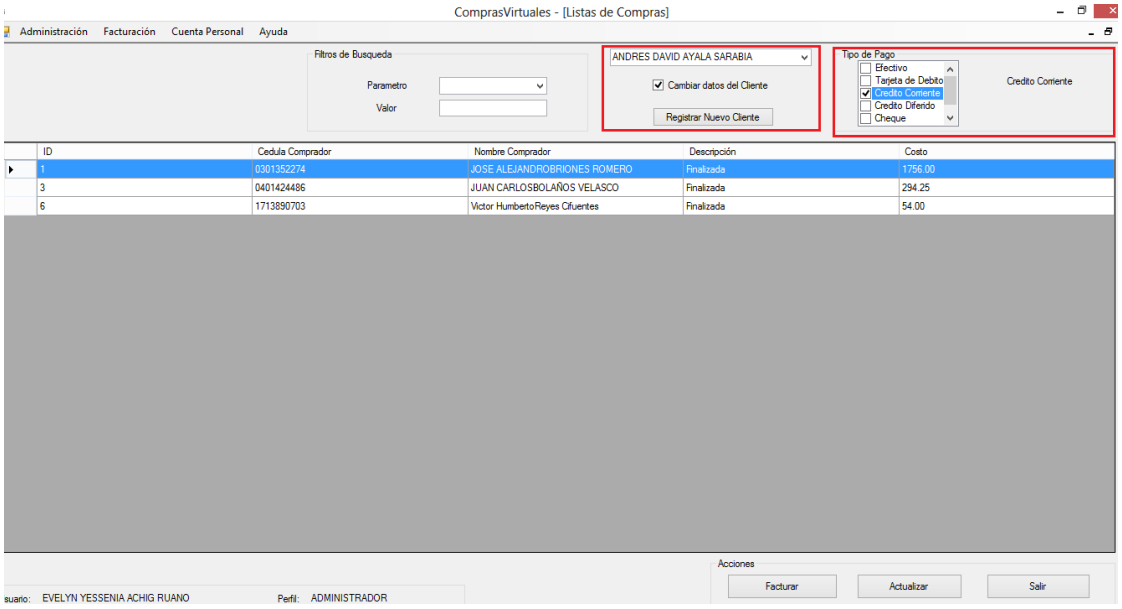

Figura 4.26 Listas de Compras

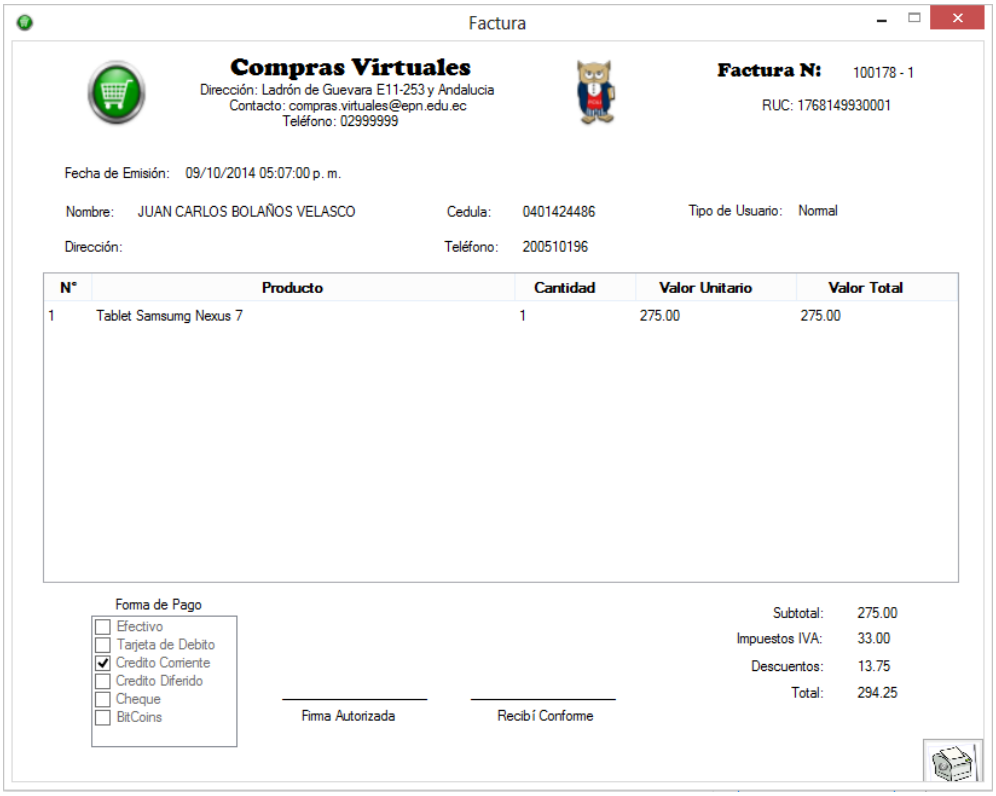

Figura 4.27 Ejemplo factura generada

4.4.2.2.4 Cuenta Personal

En las pruebas de este componente se comprobó que el usuario autenticado, sea este Administrador o Cajero, tiene la posibilidad de cambiar su contraseña de acceso o sus datos personales.

4.4.2.3 Pruebas aplicación móvil

4.4.2.3.1 Autenticación

Al realizar las pruebas de la autenticación de la aplicación móvil, se presentan tres posibilidades: autenticarse con perfil de Usuario, autenticarse con perfil de Usuario VIP o ingresar datos no válidos y no poder acceder a los componentes de la aplicación.

Al momento de acceder a la aplicación con datos validos se puede observar que se muestra el menú principal y en la parte superior derecha de la actividad se encuentra el nombre completo de usuario que ingreso sus datos como se muestra en la **Figura 4.28**, en el caso de que se hayan ingresado datos erróneos se mostrará un mensaje indicando “Error de Autenticación ¡Verifique los datos ingresados!”.

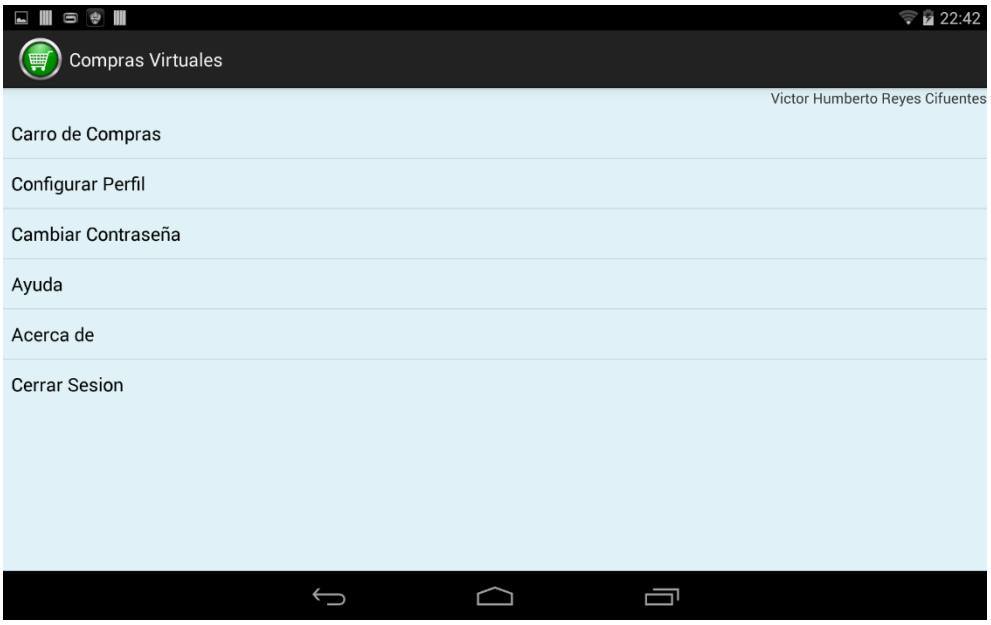

Figura 4.28 Menú Principal de la Aplicación Móvil

4.4.2.3.2 Carro de Compras

Al realizar las pruebas de esta actividad, se muestra que al ingresar a la misma, se muestra la lista de compras que el usuario tiene hasta el momento como se muestra en la **Figura 4.29**

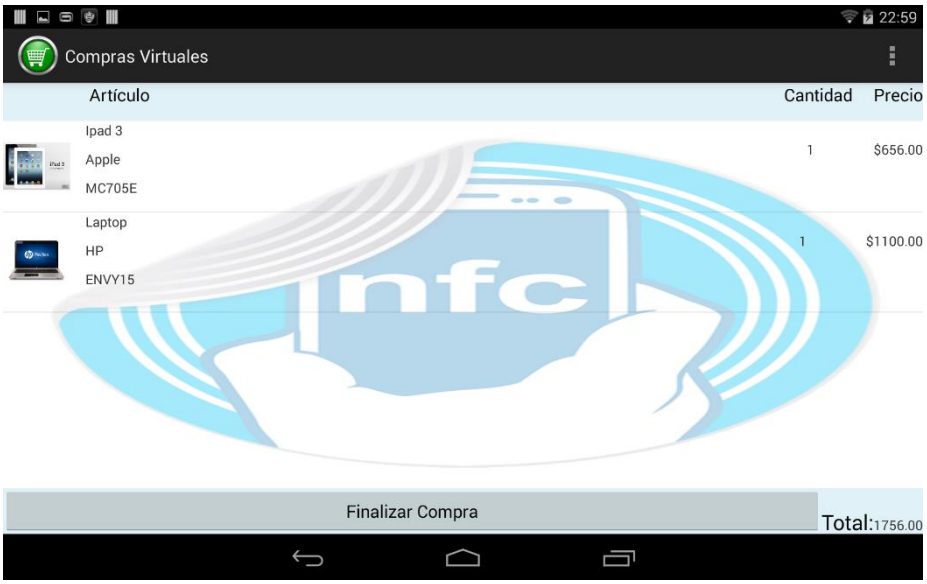

Figura 4.29 Actividad Carro de Compras de la Aplicación Móvil

Además se comprueba que al pasar el dispositivo móvil por el tag cuando la aplicación está en la actividad Carro de Compras, se muestra una actividad con la información del artículo que está identificado por el tag, en esta actividad es posible seleccionar la cantidad de productos deseada de un artículo como se muestra en la **Figura 4.30**.

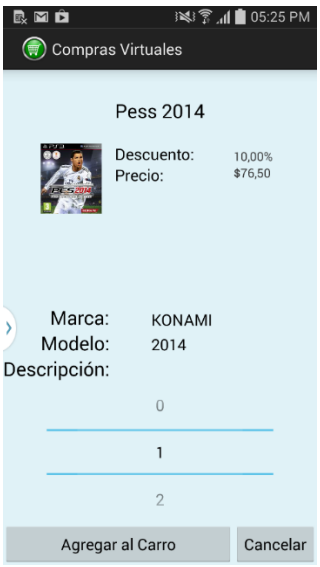

Figura 4.30 Actividad Información del Artículo de la Aplicación Móvil

Finalmente se muestra que al mantener presionado un artículo de la lista del carro de compras, se presenta un menú que da la posibilidad de aumentar o disminuir la cantidad de productos de dicho artículo, como se muestra en la **Figura 4.31**.

Figura 4.31 Menú Opciones de Compra de la Aplicación Móvil

Las pruebas que se mostraron en esta sección son las pruebas que se realizaron al final del proceso de desarrollo del sistema prototipo. Lógicamente, durante cada iteración se realizaron las pruebas correspondientes y a partir de ellas se corrigieron los errores que presentaba el sistema prototipo, como por ejemplo los errores que se presentan a continuación:

- Con respecto a la base de datos se presentaron errores al momento de realizar la conexión con el Servicio Web, para solucionar esto se creó un nuevo usuario en la base de datos con los permisos necesarios para acceder a la base de datos del proyecto. Las credenciales de dicho usuario se reemplazaron por las que pone por defecto el IDE de desarrollo.
- Inicialmente solo se manejaba una tabla que contenía los datos de artículo, marca y modelo lo que daba cierta confusión al momento de manejar el inventario, por lo que se procedió a separar estos datos en varias tablas con la finalidad de proporcionar organizar de mejor forma los productos registrados.
- La aplicación móvil dejaba de responder al momento de realizar una llamada al Servicio Web, para solucionar este problema se usó Tareas

Asíncronas, lo que permitió que las llamadas al servidor se realicen en segundo plano para evitar que la aplicación móvil no responda.

- Inicialmente se enviaban los datos de alias y de *password* del usuario en texto plano, lo que producía riesgos en la seguridad del sistema, ya que con un simple analizador de tráfico se podía capturar la petición al método Autenticación del Servicio Web y obtener los datos del usuario. Por tal razón se optimizo el método de autenticación enviando el *password* cifrado utilizando MD5.
- En la aplicación de escritorio que se observó que al momento de mostrar gran cantidad de registros en las grillas, la búsqueda de información se dificultaba. Por tal razón, se implementaron filtros de búsqueda para acceder a la información deseada.
- Originalmente, las imágenes de los artículos se cargaban al servidor por medio del protocolo FTP, pero se comprobó que no era la mejor forma de hacerlo ya que FTP no es un protocolo seguro. Por tal razón, se optó por cargar las imágenes de los artículos a través del Servicio Web.
- Originalmente, el Servicio Web estaba basado en la tecnología SOAP, pero se comprobó que no era una tecnología óptima para implementar el Servicio Web, ya que solo permitía la transferencia de información a través del formato XML. Por tal razón, se optó por utilizar la tecnología REST que permite la transferencia de información a través del formato JSON.

4.5 COMPARACIÓN DE LA APLICACIÓN MÓVIL CON LAS APLICACIONES UTILIZADAS EN EL ANÁLISIS DE REQUERIMIENTOS

En esta sección se presenta una tabla comparativa entre la aplicación realizada y las aplicaciones que se utilizaron para elaborar el análisis de requerimientos, con el fin de evaluar y comparar la aplicación realizada. Esta información se presenta en la **Tabla 4.35** y la **Tabla 4.36**.

Servicio de la Aplicación	NFC iShop	Mercado Libre	Compras Virtuales
Autenticación	No autentica a los clientes, al momento de realizar el pago, se deben ingresar los datos de una tarjeta de crédito para poder realizar el pago.	La aplicación brinda un servicio de autenticación para que la aplicación sea utilizada por un usuario específico. Si el usuario no se autentica no podrá disponer de todos los servicios de la aplicación, solo podrá realizar búsquedas.	Para poder hacer uso de la aplicación, se debe ingresar en la activadad de autenticación un usuario y contraseña válidos, en base a esto se va a establecer una sesión para que el usuario pueda realizar sus compras. Además, si el usuario lo desea puede mantener su sesión iniciada en su dispositivo móvil.
Menú principal	Manejado como una Actividad más de la aplicación y haciendo uso de `ImageButtons`	Manejado como un control personalizado denominado `Side ViewMenu`, este menú siempre podrá ser accesible desde cualquier Actividad.	Manejado como una Actividad que contiene un `ListView` en el que se muestran las opciones a realizar.
Carro de compras	Tiene una Actividad específica para ir acumulando los productos y luego pagar en una sola cuenta todos los productos seleccionados.	No dispone de este servicio.	Tiene una Actividad específica para mostrar la información de los productos que se seleccionó con los respectivos precios. Además en esta Actividad se presenta el costo total de la compra, y se cuenta con un menú de opciones que permite limpiar el carro de compras (vaciar la lista de productos seleccionados) y finalizar la compra.

Tabla 4.35 Cuadro comparativo de la aplicación realizada con las aplicaciones NFC iShop y Mercado Libre

Servicio de la Aplicación	NFC iShop	Mercado Libre	Compras Virtuales
Selección de un producto	Acercando el móvil NFC a un tag NFC.	Utilizando las actividades de Buscar y Categorías.	Acercando el móvil NFC a un tag NFC.
Pagos de las compras	Uso de la Actividad `Checkout` para pagar las compras mediante una tarjeta de crédito.	No brinda el servicio, una vez que el usuario realiza la compra, el sistema pone en contacto al comprador y al vendedor.	Si bien no brinda un servicio de pago directo, al momento de finalizar la compra el cajero del local podrá acceder a la facturación de la aplicación de escritorio, para realizar el pago. Al momento de finalizar la compra se muestra una Actividad con un reporte de los valores a pagar.
Modificación de los datos del usuario	Permitido en la aplicación mediante una Actividad.	No permitido en la aplicación, solo hay la posibilidad de ver los datos del usuario.	Permitido en la aplicación mediante una Actividad.

Tabla 4.36 Cuadro comparativo de la aplicación realizada con las aplicaciones NFC iShop y Mercado Libre (continuación)

Realizando una evaluación final de la aplicación móvil realizada, cumple con los requisitos establecidos. Si bien las aplicaciones utilizadas para el análisis de requerimientos se utilizan en ambientes diferentes al que se plantean en el proyecto, las mismas fueron útiles para obtener una idea general de las funciones que debe tener la aplicación móvil.

CAPÍTULO 5. ANÁLISIS DE COSTOS

5.1 INTRODUCCIÓN

En el presente capítulo se presenta el análisis de costos del sistema prototipo propuesto. Se estiman los costos de desarrollo del software, y el costo del hardware mínimo que se requiere para el correcto funcionamiento del sistema. Finalmente, se realiza una comparación de costos de la aplicación móvil desarrollada, con las aplicaciones móviles Mercado Libre y NFC iShop utilizadas en el análisis de requerimientos.

5.2 COSTOS DE DESARROLLO DEL SOFTWARE DEL SISTEMA PROTOTIPO

Para el costo de desarrollo del software, se considera el costo que se invierte en el sueldo de los programadores durante el tiempo estimado de desarrollo del sistema que se presentó en el plan de entregas elaborado en la sección 2.4.1.3.

Se considera que el sueldo de un programador junior en el sector público equivale a 817 dólares americanos más los beneficios de ley los cuales son: décimo tercer sueldo, décimo cuarto sueldo y el aporte patronal; los valores equivalentes a décimo tercer sueldo y décimo cuarto sueldo serán valores prorrateados ya que no se trabaja un año completo. El sueldo considerado equivale al sueldo que percibe un servidor público 1[96].

De acuerdo al plan de entregas elaborado, el tiempo estimado de desarrollo del sistema prototipo es de 6 meses. La **Tabla 5.1** muestra el total de dinero invertido en la remuneración económica de los programadores, que equivale al costo de desarrollo del software utilizado en el sistema prototipo.

[96] Remuneración fijada de acuerdo a la escala de remuneraciones unificadas para el sector público de la resolución del ministerio de relaciones laborales No. MRL-2012-0021.

MIEMBRO DEL EQUIPO	REMUNERACIÓN ECONOMICA (USD)
PROGRAMADOR 1	6.019,72
PROGRAMADOR 2	6.019,72
TOTAL	12.039,44

Tabla 5.1 Remuneración económica considerada a cada programador por 6 meses de trabajo

5.3 COSTOS DE HARDWARE MÍNIMO REQUERIDO EN EL SISTEMA PROTOTIPO

Para determinar los costos del hardware mínimo requerido en el sistema para su correcto funcionamiento, se utiliza los resultados obtenidos en el análisis de requerimientos de la sección 4.3.2. Los requisitos de hardware que se requieren para el funcionamiento del sistema prototipo considerando una carga de hasta 1000 usuarios se detallan en la **Tabla 5.2**.

DISPOSITIVO	ROL EN EL SISTEMA	CARACTERISTICAS MÍNIMAS
Computador de escritorio	Servidor	- Memoria RAM: 8GB, 1600 MHz - Disco Duro: 500GB, 5400rpm - Procesador: Core i5, 2.4 GHz
Computador de escritorio	Cliente (aplicación de escritorio – cajero)	- Memoria RAM: 8GB, 1600 MHz - Disco Duro: 500GB, 5400rpm - Procesador: Core i5, 2.4 GHz
Computador de escritorio	Cliente (aplicación de escritorio – administrador)	- Memoria RAM: 8GB, 1600 MHz - Disco Duro: 500GB, 5400rpm - Procesador: Core i5, 2.4 GHz
TAGS NFC	Almacenar el id de los artículos del sistema	- Familia de tags MifireUltraligth - Mínimo 28 bytes de almacenamiento

Tabla 5.2 Dispositivos utilizados en el sistema prototipo

Los dispositivos descritos en la **Tabla 5.2** permiten el funcionamiento del sistema prototipo, vale la pena indicar que para que el sistema se utilice en otras condiciones, se deben volver a dimensionar los equipos de acuerdo a la demanda de usuarios que va a tener el sistema.

5.3.1 COMPARACIÓN DE PRECIOS DE COMPUTADORES DE ESCRITORIO [27] [28] [29]

Para elegir los computadores a utilizar en el sistema prototipo, se realiza una comparación de precios entre 3 marcas diferentes que cumplen los requerimientos mínimos establecidos, para posteriormente escoger la opción más conveniente. La **Tabla 5.3** muestra la comparación de características y precios de 3 modelos consultados en el mercado.

MARCA	MODELO	CARACTERÍSTICAS	PRECIO (USD)
DELL	New Inspiron 13 7000 Series 2-in-1	- Procesador Intel® Core™ i5 4ta generación - Windows 8.1 Edición de un Solo Idioma (64-bit), Español - 8GB de Memoria DDR3 1600 MHz - Disco Duro SATA de 500 GB 5400 RPM	749,99 [27]
LENOVO	THINKPad L440	- Procesador Intel® Core™ i5 4ta generación - Windows 8.1 Edición de un Solo Idioma (64-bit), Español - 8GB de Memoria DDR3 1600 MHz - Disco Duro SATA de 500 GB 5400 RPM	602,10 [29]
HP	PC Portátil HP Pavilion 14-n026la	- Procesador Intel® Core™ i5 4ta generación - Windows 8 Edición de un Solo Idioma (64-bit), Español - 8GB de Memoria DDR3L a 1600MHz - Disco Duro SATA de 750 GB 5400 RPM	740,20 [28]

Tabla 5.3 Comparación de precios de computadores

Como se puede observar en la **Tabla 5.3** los precios de los computadores con similares características no tienen mayores diferencias. Sin embargo, la marca de computador de escritorio que mayormente se ajusta a los requerimientos establecidos y por calidad es la HP Pavilion 14-n026la . El costo de adquisición de los 3 computadores de escritorio necesarios para el sistema prototipo es de 2.220,6 dólares americanos.

5.3.2 COMPARACIÓN DE PRECIOS DE TAGS NFC

Los tags NFC son productos que no se encuentran en el mercado nacional por lo que en esta sección se muestran precios obtenidos en Estados Unidos a través del sitio web Amazon.com. Para poder realizar una comparación de diferentes modelos de tags NFC y para comprobar la compatibilidad de cada modelo con el sistema prototipo, se optó usar en el sistema prototipo kits de desarrollo. La **Tabla 5.4** muestra la comparación de precios de kits de desarrollo de tags NFC de 3 vendedores diferentes.

VENDEDOR	CONTENIDO DEL KIT	PRECIO (USD)
TagsForDroid	- 5 tags NFC NTAG203 - 1 Llavero NFC NTAG203	8,99
GoToTags	- 2 tags NXP NTAG203 - 2 tags NTAG210 - 2 tags NTAG212 - 2 tags NTAG213 - 2 tags NTAG216 - 2 tags Kovio 2K - 2 tags Mifare Ultralight - 2 tags Topaz 512 - 2 tags Mifare 1K - 2 tags ICode SLIX	14,98
WhizTags	- 10 tags NFC NTAG203 - 1 Llavero NFC NTAG203	12,99

Tabla 5.4 Comparación de precios de kits de desarrollo de tags NFC

Como se puede observar en la **Tabla 5.4**, la mejor opción para elegir de los kit de desarrollo es la ofrecida por el vendedor GoToTags ya que ofrece mayor variedad para realizar pruebas con los distintos tipos de tags disponibles en el mercado. El precio por tag es alrededor de 75 centavos por unidad, sin embargo el precio de cada modelo varia si se compra por separado.

Adicionalmente en esta sección se incluyen cotizaciones de rollos de tags NFC para montar un sistema que permita tener una mayor cantidad de artículos. La **Tabla 5.5** muestra una comparación de precios de rollos de tags.

VENDEDOR	MODELO DE TAG	CANTIDAD DE TAGS EN EL ROLLO	PRECIO UNITARIO DEL TAG (USD)	PRECIO POR EL ROLLO DE TAGS (USD)
atlasRFIDstore	NXP NTAG203	2000	0,40	792
GoToTags	NXP NTAG210	1000	0,29	285
GoToTags	NXP NTAG203	1000	0,36	352

Tabla 5.5 Comparación de precios de rollos de tags NFC

5.3.3 COSTO TOTAL DE HARDWARE

De acuerdo a los costos establecidos en las secciones anteriores para cada componente de hardware, la **Tabla 5.6** muestra el costo que debe invertirse en equipos para poner en funcionamiento el sistema prototipo.

EQUIPO	CANTIDAD	PRECIO UNITARIO (USD)	PRECIO TOTAL (USD)
COMPUTADORES	3	740,20	2.220,60
KIT DE DESARROLLO DE TAGS NFC	1	14,98	14,98
TOTAL			2.235,58

Tabla 5.6 Detalle de costos de hardware en el sistema prototipo

5.4 COSTO TOTAL DEL SISTEMA PROTOTIPO

De acuerdo a los costos establecidos en las secciones anteriores por desarrollo y hardware mínimo requerido, se puede sacar un costo total del sistema prototipo. La **Tabla 5.7** muestra el estimado del costo total del sistema.

COMPONENTE DEL SISTEMA	COSTO(USD)
Software	12.039,44
Hardware	2.235,58
TOTAL	14.275,02

Tabla 5.7 Detalle de costo total del sistema

5.5 ALTERNATIVAS PARA RECUPERAR LA INVERSIÓN POR LA ADQUISICIÓN DEL SISTEMA PROTOTIPO

5.5.1 FIJAR PRECIO DE VENTA A LA APLICACIÓN MÓVIL

Una alternativa para recuperar la inversión es fijar un precio de venta a la aplicación móvil. De acuerdo a precios establecidos en Google Store de aplicaciones similares, estos varían entre 1 a 2 dólares americanos, por lo que se considera que la aplicación móvil debería tener un precio similar. Considerando la complejidad de la aplicación y la funcionalidad que presta dentro del sistema prototipo, la aplicación móvil debería tener un costo de 2 dólares americanos. De acuerdo a esta estimación, la inversión efectuada se recuperaría al momento que aproximadamente 6.000 usuarios compren la aplicación.

Vale la pena aclarar que la estimación realizada es tan solo una aproximación, se tendría que considerar también el nivel de aceptación que tenga esta aplicación por parte del cliente y el número de potenciales usuarios de la aplicación que tengan integrados en sus dispositivos móviles el sistema NFC.

5.5.2 RECUPERAR LA INVERSIÓN A TRAVES DEL VOLUMEN DE VENTAS

Otra alternativa viable para recuperar la inversión es aumentar el volumen de ventas. Al implementar un modelo novedoso de ventas para el cliente, se atraerá nuevos clientes e incrementará el volumen de ventas del local que utilice el sistema prototipo. Para aumentar el volumen de clientes del local, la aplicación móvil del sistema deberá ser gratuita, ya que esto permitirá que los clientes se animen a probar la aplicación y a hacer sus compras a través del sistema.

Igual que en la sección anterior esta es tan solo una estimación, ya que habría que medir los niveles de aceptación de la aplicación así como también el número de potenciales clientes que dispongan de equipos móviles con NFC integrados.

5.6 COMPARACIÓN DE COSTOS DE LA APLICACIÓN MÓVIL CON LAS APLICACIONES UTILIZADAS PARA EL ANÁLISIS DE REQUERIMIENTOS

En esta sección se incluye una comparación de los costos de venta de la aplicación móvil del sistema prototipo con las aplicaciones utilizadas para el análisis de requerimientos. La comparación de precios se realiza de acuerdo a las 2 alternativas presentadas para recuperar la inversión del sistema prototipo. Los costos de las aplicaciones móviles fueron obtenidos del sitio de descargas Google Store.

5.6.1 ALTERNATIVA DE APLICACIÓN MÓVIL CON COSTO DE VENTA

La **Tabla 5.8** muestra una comparación de precios de las aplicaciones Mercado Libre, NFC iShop y la aplicación Compras Virtuales del sistema prototipo, fijando un precio a la aplicación. Vale la pena indicar que la aplicación móvil del sistema prototipo no fue publicada en el sitio de descargas Google.

APLICACIÓN	PRECIO DE VENTA (USD)
NFC iShop	0
Mercado Libre	0
Compras Virtuales	2

Tabla 5.8 Comparación de precios de aplicaciones móviles, primera opción

Como se puede observar en la **Tabla 5.8**, las aplicaciones utilizadas para el análisis de requerimiento son gratuitas, lo que representaría una desventaja competitiva a la aplicación móvil del sistema prototipo. Esta desventaja se podría compensar mostrando a la aplicación como una aplicación novedosa que reemplaza la forma tradicional de realizar las compras en un local comercial.

5.6.2 ALTERNATIVA DE APLICACIÓN MÓVIL GRATUITA

La **Tabla 5.9** muestra una comparación de precios de las aplicaciones Mercado Libre, NFC iShop y la aplicación Compras Virtuales del sistema prototipo, en este caso la aplicación móvil es gratuita.

APLICACIÓN	PRECIO DE VENTA (USD)
NFC iShop	0
Mercado Libre	0
Compras Virtuales	0

Tabla 5.9 Comparación de precios de aplicaciones móviles, segunda opción

Como se puede observar en la **Tabla 5.9** las aplicaciones utilizadas para el análisis de requerimientos y la aplicación del sistema prototipo son gratuitas, en este caso no existe desventaja competitiva.

CAPITULO 6. CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- La aplicación de escritorio del sistema prototipo desarrollado permite automatizar el manejo de inventario de los productos de un establecimiento; el sistema registra cuando un producto ingresa al sistema y de igual manera cuando el producto es reservado o vendido a los clientes. De igual manera, la aplicación de escritorio del sistema prototipo, permite automatizar el proceso de facturación de un establecimiento ya que el cajero al ingresar el número de identificación del cliente en la aplicación de escritorio, tendrá acceso a la selección del cliente para poder recibir el pago y posteriormente emitir la factura, sin la necesidad de que el cajero registre uno a uno los productos que adquirió el cliente.

- La aplicación móvil del sistema prototipo desarrollado, permite automatizar el proceso de selección de artículos por parte de los usuarios; al momento de realizar la acción de acercar el móvil NFC al tag NFC, el cliente visualiza información detallada del artículo que consulto y si es de su interés, agrega el producto al carro de compras virtual. De esta manera, se reemplaza la acción de tomar físicamente el producto de las perchas y transportarlo hasta la caja para cancelar el valor de los mismos.

- Las aplicaciones que se utilizaron para el análisis de requerimientos y otras aplicaciones encontradas en el mercado con funcionalidad similar al sistema prototipo desarrollado, permiten determinar que no existe en el mercado aplicaciones que tengan las características del sistema prototipo desarrollado. El sistema prototipo desarrollado, demuestra el potencial que tiene el campo de la tecnología NFC para el desarrollo de numerosas aplicaciones innovadoras y atractivas para el usuario.

- El sistema prototipo desarrollado trabaja con una arquitectura de tres niveles para separar los componentes del sistema. De esta forma, cada nivel cumple una función específica, distribuyendo entre los distintos componentes el consumo de recursos necesario para el funcionamiento del sistema prototipo.

- El uso de Servicios Web independiza la plataforma en la que se deben implementar las aplicaciones cliente que consumen el Servicio Web. El sistema prototipo desarrollado demuestra esta afirmación, ya que las dos aplicaciones se desarrollaron para plataformas diferentes; la aplicación de escritorio que maneja el inventario y la facturación del local comercial se desarrolló empleando el lenguaje de programación C# y se implementó en computadores de escritorio con sistema operativo Windows; la aplicación móvil que maneja el carro de compras virtual se desarrolló empleando el lenguaje de programación Java y se implementó en dispositivos móviles con sistema operativo Android.

- El uso de la arquitectura REST en el sistema prototipo desarrollado, permitió utilizar el formato JSON para la comunicación entre el servidor y los clientes, lo que permitió construir mensajes ligeros. En las aplicaciones clientes, con el uso de las librerías que proporcionan cada entorno de desarrollo, la deserialización de estos mensajes se realiza de forma transparente.

- En las aplicación móvil y la aplicación de escritorio desarrolladas para el sistema prototipo, las *cookies* permiten al servidor identificar el usuario que realiza la petición y obtener los datos necesarios para atender las peticiones de los clientes ya que el servidor identifica que usuario esta autenticado en la aplicación cliente por medio de la sesión establecida con el servidor.

- El uso de hilos y tareas asíncronas para las peticiones al Servicio Web en la aplicación móvil evitan que la aplicación se detenga o falle en caso de que el servidor tarde en dar una respuesta a una petición.

- Para el desarrollo de la aplicación móvil para el sistema operativo Android se separaron los recursos de la aplicación del código, como es el caso de: actividades, cadenas de texto, controles, entre otros. Esto facilita el poder cambiar fácilmente el diseño y contenidos de dichos recursos sin la necesidad de tener que realizar cambios en el código de la aplicación.

- Los *Intents* permiten solicitar la acción de un componente del dispositivo móvil, como en el caso de la aplicación móvil desarrollada para el sistema prototipo, donde la Actividad Carro de Compras ejecuta un nuevo *intent* cada vez que el dispositivo móvil reconoce la presencia de un tag NFC, lo que permite observar la información del artículo identificado en el tag.

- La aplicación móvil del sistema prototipo desarrollado funciona con las versiones del sistema operativo Android equivalente al API nivel 14 que equivale a la versión 4.0 de Android "*Ice Cream Sandwich*", ya que a partir del API nivel 14 se incluyen las librerías necesarias para el uso del adaptador NFC del dispositivo móvil.

- El manifiesto define los recursos del dispositivo móvil que se desean utilizar como: el adaptador NFC, la cámara, entre otros. También define la versión del sistema operativo mínima y máxima en las que funciona la aplicación desarrollada, la Actividad en la que va a comenzar la aplicación, las actividades que puede mostrar la aplicación, entre otros.

- En el sistema prototipo desarrollado los *triggers* permiten: actualizar la disponibilidad de los artículos cada vez que un producto es reservado o vendido, registrar la fecha de venta de los productos, etc. El uso de *triggers* permite realizar cambios en las tablas automáticamente sin la

necesidad de ejecutar procedimientos almacenados cada vez que se produce un requerimiento de actualización en la base de datos.

- El uso de procedimientos almacenados permite que las acciones que requieran hacer modificaciones a la base de datos se realicen en el motor de base de datos, liberando el uso de recursos de los demás componentes del sistema ya que los mismos solo realizaran la petición al motor de base de datos.

- El uso de pruebas unitarias permiten comprobar que cada uno de los componentes de la aplicación trabajen de la manera esperada, además permite mejorar el código para que tenga un mejor funcionamiento. Las pruebas unitarias se pueden realizar independientemente del lenguaje de programación o de la plataforma de desarrollo utilizada.

- El uso de metodologías de desarrollo para el diseño y desarrollo del sistema prototipo permitió: establecer los requerimientos del sistema prototipo de forma organizada, definir y organizar todas las tareas que se realizaron, documentar de manera adecuada el sistema prototipo, establecer tiempos de desarrollo de cada componente del sistema prototipo.

- La metodología de desarrollo XP ayudo a organizar el proyecto en iteraciones, estas iteraciones permitieron controlar el avance del proyecto de manera adecuada y realizar las pruebas necesarias en cada iteración, al finalizar cada iteración se realizaron las pruebas de aceptación de las historias de usuario que permitieron comprobar el correcto funcionamiento del sistema.

6.2 RECOMENDACIONES

- Para mejorar la seguridad del sistema, se recomienda utilizar un protocolo seguro como HTTPS que crea un canal cifrado para la transferencia de los datos entre el cliente y el servidor, con lo que los datos que envía el usuario estarían protegidos frente a posibles problemas de seguridad.

- Para evitar fallos en el sistema, o pérdida de datos, se debe considerar respaldar periódicamente la base de datos, con el fin de tener la posibilidad de recuperar los datos en caso de pérdida de información.

- En el sistema prototipo desarrollado los tags no se encuentran protegidos contra escritura para poder hacer el reúso de los mismos y disminuir los costos invertidos. Si el sistema prototipo es puesto en producción, se recomienda proteger contra escritura los tags utilizados para prevenir que usuarios que los usuarios del sistema alteren el contenido de los mismos y provoquen fallos en el sistema.

- Una mejora al sistema prototipo que se podría incluir, es la implementación de un módulo de pagos a través de la tecnología NFC. En el sistema prototipo desarrollado se utilizó únicamente el modo de lectura del esta tecnología, se podría utilizar también el modo de emulación de tarjeta para realizar los pagos a través de tarjetas de créditos almacenadas en el dispositivo móvil.

- Antes de poner el sistema prototipo desarrollado en producción, es recomendable realizar las pruebas de carga al servidor que alojará el Servicio Web para comprobar que el mismo funcione correctamente de acuerdo a la demanda requerida.

REFERENCIAS BIBLIOGRÁFICAS

[1] A. Lafuente, «Departamento de Arquitectura y Tecnología de Computadores, UPV/EHU,» http://www.sc.ehu.es/acwlaroa/SDI/Apuntes/Cap1.pdf. [Último acceso: 13 Mayo 2014].

[2] D. Márquez, «Componentes Software para Sistemas Distribuidos de Tiempo Real,» de *Componentes Software para Sistemas Distribuidos de Tiempo Real*, Málaga, Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga España, 2005, pp. 4,5,6.

[3] L. Navarro Moldes y J. Marqués, «bernal.pro,» Junio 2006. http://bernal.pro/informatica/apuntesdeinformatica/finish/6-apuntes-de-informatica/63-arquitectura-de-sistemas-distribuidos. [Último acceso: 01 Julio 2014].

[4] Microsoft Corporation, «La Arquitectura Orientada a Servicios (SOA) de Microsoft,» Microsoft Corporation, 2006.

[5] Área de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Vigo, «ccia.ei.uvigo.es,» Universidad de Vigo, Octubre 2008. http://ccia.ei.uvigo.es/docencia/SCS. [Último acceso: 7 Julio 2014].

[6] H. Ramos Morillo, J. V. Berna Martinez y F. Maciá Perez, «Computación Ubicua Mediante Dispositivos RFID,» de *Computación Ubicua Mediante Dispositivos RFID*, Alicante, Servicio de Publicaciones de la Universidad de Alicante, 2006, pp. 59-71.

[7] S. Pomares Hernández, «Computación Ubicua; un gran desafío,» Instituto nacional de Astrofísica, Óptica y Electrónica (INAOE), Mexico, 2010.

[8] A. Aransay, «Computación Ubicua: Diseño de iteracción centrada en el usario,» Universidad de Vigo, 2009.

[9] S. CLARK, «NFC world,» 5 Junio 2013. http://www.nfcworld.com/2013/06/05/324448/one-in-three-mobile-phones-to-come-with-nfc-by-2017/. [Último acceso: 7 Mayo 2014].

[10] C. VEDAT, O. KEREM y O. BUSRA, Professional NFC Application Development for Android, Wiley, 2013.

[11] *Understanding the Requirements of ISO/IEC 14443,* ATMEL, 2005.

[12] E. Alberca, Estudio de la tecnología inalámbrica NFC y sus aplicaciones en el ámbito de las telecomunicaciones, Escuela Politécnica Nacional, 2013.

[13] Android Developers, «developer.android.com,» http://developer.android.com/. [Último acceso: 07 Julio 2014].

[14] Microsoft, «msdn.microsoft.com,» Microsoft, http://msdn.microsoft.com/. [Último acceso: 07 Julio 2014].

[15] H. Korth, Fundamentos de Base de Datos, Madrid-España: McGRAW-HILL, 2002.

[16] A. Méndez, «METODOLOGÍAS DE DESARROLLO DE SOFTWARE,» ITSA, 2010.

[17] Universidad Nacional Autónoma de Mexico, «UNAM,» http://dicyg.fi-c.unam.mx:8080/lalo/aed-teo/FasesDS.pdf. [Último acceso: 07 Julio 2014].

[18] J. Joskowicz, Reglas y Prácticas en eXtreme Programming, Instituto de Ingeniería Eléctrica (IIE), 2008.

[19] agilemanifesto.org, «Agile Manifiesto,» Febrero 2001. http://agilemanifesto.org/. [Último acceso: 13 Junio 2014].

[20] Google, «Google Store,» https://play.google.com/store/apps/details?id=sg.edu.nyp.nfcishop. [Último acceso: 27 Julio 2014].

[21] MercadoLibre, «MercadoLibre Institucional,» http://institucional.mercadolibre.com/. [Último acceso: 23 Julio 2014].

[22] L. Song, «CODE PROJECT,» http://www.codeproject.com/Articles/322436/Maintain-HTTP-Session-State-in-WCF-REST-Services-w. [Último acceso: 20 Agosto 2014].

[23] Á. Ulises y Pedro, «Wordpress,» http://pedroavilanu.wordpress.com/2013/07/23/filtrar-datos-en-datagridview/. [Último acceso: 22 Agosto 2014].

[24] Tuttini y Leandro, «msdn foro social,» http://social.msdn.microsoft.com/Forums/es-ES/bf95294f-8591-4e82-bc67-81b4f027f608/autocompletar-combobox-c?forum=vcses. [Último acceso: 22 Agosto 2014].

[25] Google, «Google Project Hosting,» https://code.google.com/p/google-gson/. [Último acceso: 26 Agosto 2014].

[26] JUnit, «JUnit,» http://junit.org. [Último acceso: 23 Junio 2014].

[27] DELL, «DELL,» http://www.dell.com/ec/p/inspiron-3647-small-desktop/pd?ref=PD_OC#overrides=la_gensff1505_315_id3647_i581tbw8s_5:5~E2014RL. [Último acceso: 6 Octubre 2014].

[28] HP, «HP,» http://www8.hp.com/ec/es/products/desktops/product-detail.html?oid=7253523#!tab=specs. [Último acceso: 6 Octubre 2014].

[29] LENOVO, «LENOVO,» http://www.lenovo.com/ec/es/. [Último acceso: 6 Octubre 2014].

ANEXOS

Los Anexos se incluyen en el CD adjunto al presente documento

ANEXO A: Base de datos del sistema

- **ANEXO A.1:** Procedimientos Almacenados
- **ANEXO A.2:** *Triggers*
- **ANEXO A.3:** Vistas
- **ANEXO A.4:** Script de creación de la base de datos
- **ANEXO A.5:** Respaldo de la base de datos

ANEXO B: Servicio Web WCF

- **ANEXO B.1:** Código fuente
- **ANEXO B.2:** Instalador
- **ANEXO B.3:** Manual de instalación

ANEXO C: Aplicación de escritorio

- **ANEXO C.1:** Código fuente
- **ANEXO C.2:** Instalador
- **ANEXO C.3:** Manual de instalación
- **ANEXO C.4:** Manual de usuario

ANEXO D: Biblioteca de clases

- **ANEXO D.1:** Código fuente

ANEXO E: Aplicación móvil

- **ANEXO E.1:** Código fuente
- **ANEXO E.2:** Instalador
- **ANEXO E.3:** Manual de instalación