Universidad de las Ciencias Informáticas

Facultad 2

Aplicación Android para interrelacionar clientes y proveedores

---

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

AUTOR

Julio César Vega Fuentes

TUTORES

MSc. Darling Darias Pérez

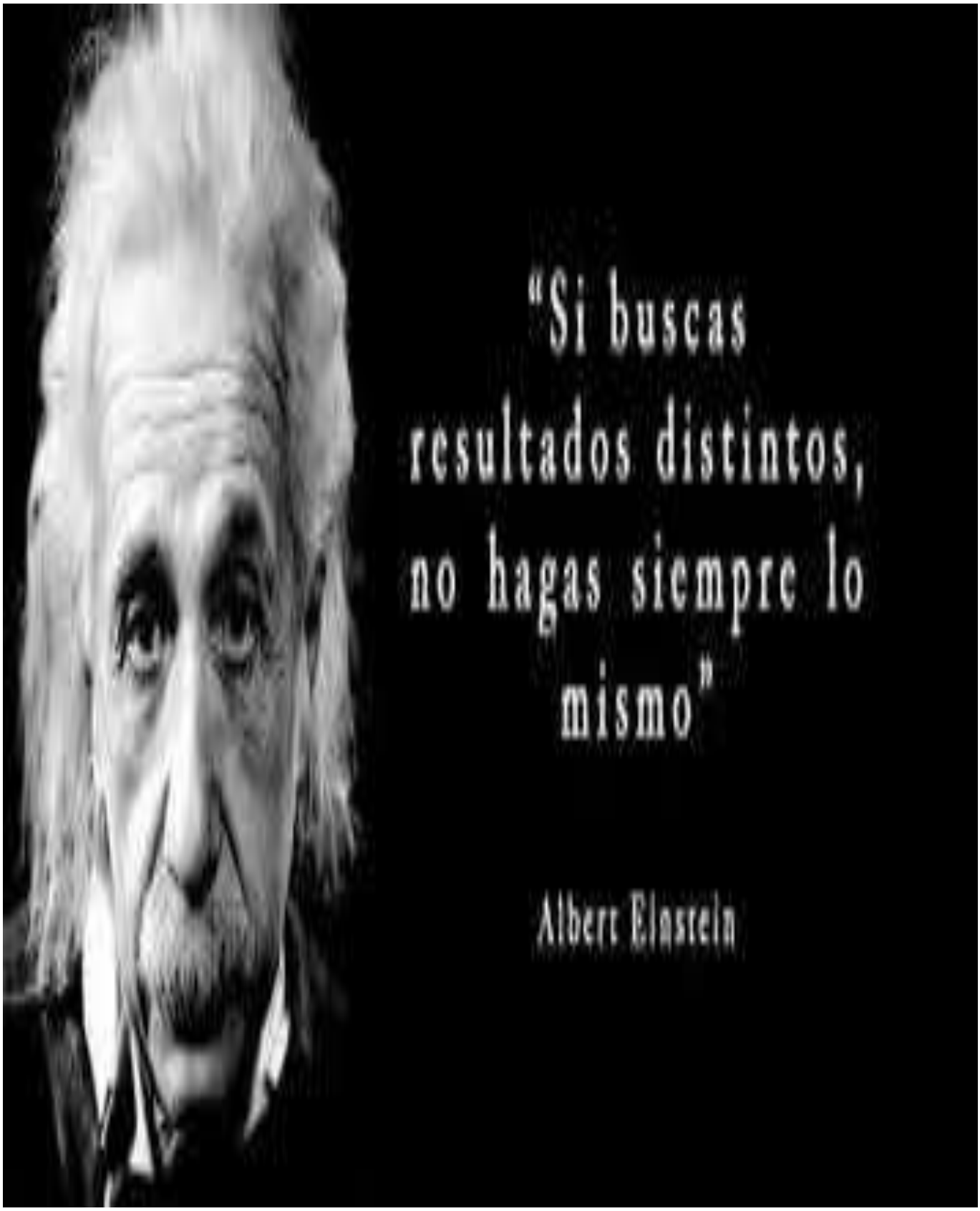
Ing. Miguel Alejandro Ulloa

CO-TUTOR

MSc. Andrés Sellés

La Habana, 2019

“Año 60 de la Revolución”


"Si buscas
resultados distintos,
no hagas siempre lo
mismo"
Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis que tiene por título: Aplicación Android para interrelacionar clientes y proveedores reconociendo a la Universidad de las Ciencias Informáticas y a la Empresa de Tecnologías de la Información para la Defensa los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Firma del autor

Julio César Vega Fuentes

Firma del tutor

MSc. Darling Darias Pérez

Firma del tutor

Ing. Miguel Alejandro Ulloa

Firma del co-tutor

MSc. Andrés Sellés

AGRADECIMIENTOS

Sin duda las primeras personas que vienen a mi mente son mis padres, y en especial mi madre, por haberme apoyado durante este largo trayecto en busca de lograr mi sueño. Por darme su amor y cariño en los buenos momentos y en las más difíciles circunstancias, por su educación, por sus valores y sus consejos.

A toda mi familia, especialmente a mis abuelos Adria y Luis, que, aunque no puedan estar conmigo siempre estuvieron pendiente de mí, apoyándome y aconsejándome en todo momento.

A mis tutores. Sin ellos este trabajo hubiese sido una meta inalcanzable. Gracias por todo el tiempo que invirtieron y sobre todo la paciencia que tuvieron para aguantarme.

A mi grupo 2501 que me ha acompañado en este viaje inolvidable, con sus vicisitudes y tropiezos.

A mis amigos del 4106, por compartir momentos malos y buenos, por ayudarme en todo lo que podían, sin duda, no los olvidaré nunca.

A la familia Armas por acogerme todo este tiempo y hacerme sentir como uno más, especialmente a Arlet por mostrarme el valor del sentimiento a otra persona y la diferencia entre compañía y soledad.

Agradecimiento especial a mi centro de estudios y a todos mis profesores, los que han contribuido a mi formación profesional, personal y educativa.

Al tribunal, por su ayuda, críticas constructivas y consejos que hicieron posible el desarrollo de esta investigación.

A todas esas personas que no mencione pero que saben que siempre formarán parte de mi vida.

Dedicatoria

A mi madre por ser mi guía, mi inspiración, mi modelo a seguir, por su amor, apoyo, esfuerzo, sacrificio y abnegación.

RESUMEN

Actualmente la contratación de servicios por parte de las empresas estatales es un proceso largo debido a que trae consigo muchos trámites y documentos, lo que provoca un exceso de pasos a seguir, la pérdida de tiempo, retrasos y, por tanto, la inconformidad y molestias de los clientes.

La presente investigación tiene como objetivo desarrollar una aplicación Android para mejorar la interrelación cliente-proveedor. Para la obtención de la misma se estudiaron las principales herramientas y tecnologías necesarias para su desarrollo, así como la metodología de software a emplear en ella. Se obtienen los artefactos para la implementación de la solución, la cual fue validada obteniéndose resultados satisfactorios.

Como resultado se obtuvo una aplicación Android para la interrelación cliente-proveedor que al desplegarse en el país simplificará enormemente el problema planteado al inicio.

PALABRAS CLAVE

Aplicación, cliente-proveedor, servicios.

ABSTRACT

Nowadays the contracting of services by state companies is a long process due to the fact that it brings with it many formalities and documents, which causes an excess of steps to follow, the loss of time, delays and, therefore, the discontent and inconvenience of the clients.

This research aims to develop an Android application to improve the customer-supplier relationship. In order to obtain it, the main tools and technologies necessary for its development were studied, as well as the software methodology to be used in it. The artifacts are obtained for the implementation of the solution, which was validated obtaining satisfactory results.

As a result, an Android application was obtained for the client-supplier interrelation that, when deployed in the country, will greatly simplify the problem raised at the beginning.

KEY WORDS

application, client-supplier, services.

ÍNDICE

Índice de tablas

INTRODUCCIÓN

En el ambiente empresarial es cada día más común la incertidumbre para tomar decisiones adecuadas y que sean proporcionales con la calidad requerida. En las propias entidades, existe una dispersión de datos e información que, en ocasiones, duplica tareas y hace imposible su recuperación por parte de quienes la necesitan en el momento oportuno. (Rojas, 2004)

La gestión de procesos es una forma de organización, diferente de la clásica organización funcional, en la que prima la visión del cliente sobre las actividades de la empresa. Los procesos así definidos son gestionados de modo estructurado y sobre su mejora se basa la de la propia empresa. La gestión de procesos aporta una visión y unas herramientas con las que se puede mejorar y rediseñar el flujo de trabajo para hacerlo lo mejor posible y adaptado a las necesidades de los clientes. Para los lograr que los procesos llevados a cabo por personas, sean mejores, se tiene en cuenta en todo momento las relaciones con proveedores y clientes. (ISOtools, 1998)

La gestión de procesos se soporta necesariamente sobre la integración de sus actividades sobre los recursos informáticos. A través de herramientas computacionales se facilita el tratamiento y acceso a la información, el flujo de los procesos y se pueden registrar, almacenar y difundir contenidos digitales sin costos adicionales empleando las redes disponibles.

La interrelación cliente-proveedor es la relación existente entre las personas que solicitan o contratan un servicio o producto (los clientes) y las que brindan el servicio o el producto (los proveedores). (Ospina, 2017).

Hay que tener en cuenta que una empresa puede hacer a la vez de proveedora y de cliente. Si los productos o servicios que comercializa van dirigidos a otras empresas que son las que luego lo venden a los consumidores finales, en esa relación haría de proveedora. (Ospina, 2017)

En Cuba la interrelación entre los proveedores y los clientes empresariales y los trámites que esta supone, generalmente, tiene como características el ser interinstitucional. Se realiza de forma manual y el cliente no tiene el control de la ejecución de los trámites sin depender de algún funcionario que lo gestione. Todo esto trae como consecuencia un exceso de pasos a seguir y, generándose la pérdida de tiempo debido a la generación de documentos en formato duro, lográndose la inconformidad y molestias de los clientes. Además, ocurren retrasos e ilegalidades amparadas por la forma en que se ejecuta el proceso actualmente.

Es importante valorar que aunque la mayor parte de la informática empresarial sigue estando localizada y fijada en ordenadores personales, en pocos años, esta tendencia se invierta. En los últimos años, las aplicaciones móviles se han posicionado como una de las herramientas más eficaces para las empresas. Estas aplicaciones promueven la interconectividad y mejoran la

experiencia en la adquisición de servicios y productos, pues permiten la comunicación desde cualquier lugar del mundo, siempre y cuando se cuente con una mínima conexión a internet.

Hoy en día se desea unir a todas las empresas estatales cubanas para que estas tengan más conocimiento de lo que poseen y brindan las demás. En la Empresa de Tecnologías de la Información para la Defensa(XETID) existe la necesidad de agilizar los trámites relacionados con el consumo de servicios y productos, mejorando así la relación cliente-proveedor.

Por lo antes expuesto se identifica el siguiente **problema a resolver:** ¿Cómo facilitar la interrelación entre clientes y proveedores para las empresas estatales en Cuba?

El **objeto de estudio** lo constituye**:** los proceso de interrelación entre clientes y proveedores.

Por lo que se define como **objetivo general:** Desarrollar una aplicación Android para la interrelación entre clientes y proveedores en las empresas estatales de Cuba.

Centrándose en el **campo de acción**: La interrelación entre clientes y proveedores en las empresas estatales de Cuba.

Para dar cumplimiento al objetivo general se definen los siguientes **objetivos específicos**:

1. Elaborar el marco teórico conceptual de la investigación.
2. Realizar el análisis y diseño de la propuesta de solución.
3. Implementar los elementos de servicios, productos y notificaciones para la interrelación cliente y proveedores.
4. Realizar pruebas de caja blanca y de aceptación a la aplicación Android.
5. Validar el resultado obtenido.

Con el fin de dar cumplimiento a los objetivos propuestos en la investigación se tomaron como **tareas científicas**:

1. Elaboración de los referentes teóricos y metodológicos de la investigación asociados a sistemas de compra-venta de productos y servicios.
2. Análisis de aplicaciones móviles de compra-venta de productos y servicios a nivel nacional e internacional para sentar las bases de la investigación.
3. Breve explicación de las herramientas, tecnologías y modelo de desarrollo de software establecidos, para lograr un entendimiento de estas.

4. Identificación de las Historias de usuario y de las características no funcionales establecidas por la metodología, para plasmar las funcionalidades que debe cumplir la propuesta de solución, así como dejar clara las condiciones bajo las cuales debe operar.
5. Análisis y diseño de la propuesta de solución.
6. Implementación de una aplicación Android para la interrelación cliente - proveedor.
7. Realización de pruebas de caja blanca para validar el código y las pruebas de aceptación para validar el correcto funcionamiento de la aplicación y satisfacción del cliente.

Para la presente investigación se utilizó como **métodos científicos**:

- **Métodos teóricos**

Analítico-sintético: El empleo de este método se evidencia en el análisis de la documentación de los procesos de interrelación entre clientes y proveedores, este permite la extracción de los elementos fundamentales relacionados con la gestión de la interrelación entre los consumidores y proveedores empresariales.

Inductivo-deductivo: Al determinar las características de los sistemas de compra-venta de productos y servicios, en el contexto nacional (Cuba).

Histórico-lógico: Utilizado al tener en cuenta la caracterización de la evolución histórica de los sistemas de interrelación entre consumidores y proveedores empresariales como base para concebir el sistema actual.

- **Métodos empíricos:**

Entrevista: Mediante el intercambio con el cliente se obtuvo la mayor cantidad de información posible para determinar las funcionalidades a desarrollar, así como las deficiencias existentes que permitieron definir el problema a resolver y establecer el objeto de estudio.

El documento está estructurado de la siguiente forma:

Capítulo 1. Fundamentación teórica: En este capítulo se elabora el marco teórico conceptual de la investigación para un mejor entendimiento de la problemática. Se realiza un análisis de las soluciones existentes de compra-venta de productos y servicios y sobre la interrelación entre consumidores y proveedores empresariales. Se caracteriza la metodología de desarrollo XP a emplear. Además, se describe la arquitectura de software definida en el desarrollo de la solución propuesta. Se explican brevemente las herramientas y tecnologías definidas para el desarrollo de la propuesta de solución.

Capítulo 2. Análisis y diseño: En este capítulo se describen inicialmente la propuesta de solución de la aplicación a desarrollar y los procesos que serán informatizados, así como las historias de usuario especificadas por el cliente y las características no funcionales que debe cumplir la aplicación a desarrollar. Finalmente se generan los artefactos correspondientes según la metodología de desarrollo de software empleada.

Capítulo 3. Implementación y Validación: En este capítulo se abordan los aspectos relacionados con la construcción de la aplicación, dentro del cual se destacan los procesos de implementación y prueba.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

El presente capítulo se realiza un estudio de temas y relacionados con la conceptualización de la interrelación entre consumidores y proveedores empresariales. Así como los conceptos asociados al objeto de estudio y al campo de acción y otros sistemas a fin de realizar una investigación más detallada. Además, se analizan y explican las herramientas, tecnologías, lenguajes y metodología de software a utilizar para el desarrollo de la propuesta de solución.

1.1. Conceptos y aspectos fundamentales asociados al problema

Con el objetivo de lograr un entendimiento del problema que inicia a la presente investigación, se requiere el conocimiento de los conceptos y aspectos enunciados a continuación:

Aplicaciones móviles Android: son extensiones informáticas para dispositivos portátiles, como los teléfonos inteligentes y las tabletas electrónicas. En un primer momento, las aplicaciones clásicas tenían una función puramente recreativa. Recientemente, una serie de aplicaciones útiles han despertado gran interés en los usuarios; entre ellas figuran programas para el registro de gastos, manuales de modismos en idiomas extranjeros y convertidores de moneda. En otras aplicaciones pueden encontrarse reseñas de los restaurantes más cercanos, listas de eventos locales o visitas comentadas de sitios turísticos. (Alegsa, 2017)

Clientes: Un cliente es alguien que ha sido impactado por un producto, es el destinatario de un servicio o producto ofrecido por un suministrador, en una situación contractual, el cliente se denomina comprador. Pero a la vez puede ser consumidor final, usuario o beneficiario. (Jurán,1993).

Proveedores: son las personas naturales o jurídicas, de derecho público o privado, que de manera habitual fabrican, elaboran, manipulan, acondicionan, mezclan, envasan, almacenan, preparan, expenden, suministran productos o prestan servicios de cualquier naturaleza a los consumidores. (debitoor, 2019).

Interrelación entre clientes y proveedores: la interrelación se refiere a una correspondencia recíproca que existe entre individuos, objetos u otros elementos. Se trata, por lo tanto, de una relación mutua (Concepto.de, 2019). En correspondencia con lo anteriormente expuesto, en la presenta investigación se adopta la interrelación entre clientes y proveedores como el vínculo que se establece entre el proveedor que suministra un producto o presta un servicio y el cliente, que es quien lo adquiere o utiliza como destinatario final.

Gestión empresarial: es aquella actividad empresarial que, a través de diferentes individuos especializados, como ser: directores institucionales, consultores, productores, gerentes, entre otros, y de acciones, buscará mejorar la productividad y la competitividad de una empresa o de un

negocio. Es decir, la finalidad de la gestión empresarial es que la empresa o compañía en cuestión sea viable económicamente. (Ucha, 2011)

Actualmente existen numerosas aplicaciones informáticas que permiten la interrelación entre los consumidores y los proveedores. Se hace necesario realizar un análisis de soluciones similares para conocer las características de estos como guía a la solución al problema planteado en la presente investigación.

1.2. Análisis de soluciones similares

Para el desarrollo de la propuesta de solución fue necesario el análisis de aplicaciones a nivel nacional e internacional. A continuación, se describen aquellas que presentan objetivos similares a las necesidades del cliente.

Sistemas Internacionales:

Vibbo: Esta aplicación es la renovación de una de las marcas icónicas de la compraventa de productos usados. Permite enviar fotos a través del chat y ver si el comprador o vendedor está en línea. También ofrece la posibilidad de seguir a los usuarios que se interesen o sean más activos. Desde la perspectiva del comprador, se trata de un buen sistema de reputación y crecimiento de visibilidad: si se consigue que lo sigan más personas, más se va a vender. A su vez, no es de código abierto y no opera en Cuba. (Norguera, 2018)

Sellfun: Sellfun tiene poco más de dos años de vida y salió al mercado ya especializado en electrónica: móviles y tablets. Uno de los grandes atractivos de la aplicación es que se paga directamente en la misma, con pago seguro. Se recibe a domicilio lo que se ha comprado. En su última actualización, Sellfun amplió sus categorías de artículos para vender más allá de teléfonos y tablets, como televisores y servicios. A su vez, no es de código abierto, no ofrece todo tipo de productos ni opera en Cuba. (Norguera, 2018)

Sistemas Nacionales:

La Chopi: es una aplicación para dispositivos móviles con sistemas iOs y Android que permite consultar los anuncios de compra - venta en Cuba. Posee una interfaz de usuario muy cómoda, lo que facilita su uso. Esta aplicación tiene elementos importantes como la búsqueda de los artículos con sus imágenes y una breve explicación de los mismos. A su vez, entre sus limitantes, no es de código abierto y no permite la interacción entre los proveedores y los consumidores. (LaChopi, 2019)

PorLaLivre: es una aplicación móvil con sistemas Android que permite la compra - venta en Cuba. Posee una interfaz de usuario cómoda, lo que facilita su uso al crear los anuncios a partir de la categoría en la que se encuentre el servicio que se quiere publicar (Viviendas, Autos,

Computadoras, Celulares, Portátil, entre otras). A pesar de posees elementos importantes, la misma no permite la interrelación entre clientes y los consumidores. (PorLaLivre, 2019)

Análisis de las soluciones

Durante el análisis de las soluciones se definieron pautas como criterio de evaluación, estas son:

A: tiene en cuenta la interrelación cliente y proveedores.

B: permite la compra-venta de servicios y productos.

C: código abierto.

D: se puede usar para empresas estatales.

E: opera en Cuba.

Posterior al análisis realizado a los sistemas de planificación de las producciones existentes a nivel nacional e internacional se concluye que ninguno de estos cumple con las necesidades que se requieren. Los sistemas internacionales no son de código abierto y no se pueden emplear en empresas estatales. Los sistemas nacionales no implementan el proceso de interrelación cliente proveedores, no son de código abierto y no se usan en empresas estatales. Se puede afirmar que muchas de estas soluciones poseen elementos de valor que pudieran ser de interés en la propuesta de solución.

1.3. Metodología de desarrollo de software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Es por eso que se clasifican en dos grandes grupos:

- **Metodologías ligeras/ágiles**: son aquellas metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando.

- **Metodologías pesadas/tradicionales**: son aquellas metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. (Carrillo Pérez, y otros, 2008)

Como guía para el proceso ingenieril se escogió un enfoque ágil, definiéndose la metodología **XP**. La misma tiene como características emplearse para proyectos pequeños, a corto plazo y no se

generan artefactos y roles en exceso. Los requisitos suelen ser cambiados con frecuencia en la medida en que avanza el desarrollo del proyecto, así el cliente puede ir añadiendo Historias de Usuarios, dividirlas para agilizar el trabajo o eliminarlas simplemente. Permite la retroalimentación, corrección de errores y finalmente la realización de un producto capaz de satisfacer todas las necesidades del mismo. Por último, se tiene la ventaja de desarrollar pequeñas partes del producto y probarlo, permitiendo terminar todas las funcionalidades e integrar el producto final.

1.4. Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas o programas informáticos destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo de las mismas en términos de tiempo y de dinero.

Para el desarrollo de la propuesta de solución se decide utilizar la herramienta Visual Paradigm for UML en su versión 8.0, debido a la fácil integración de esta con el lenguaje de programación, lo que posibilita reducir el tiempo en el proceso de desarrollo del software pues no hay necesidad de capacitar a los desarrolladores. A continuación, se describe esta herramienta:

1.4.1 Visual Paradigm 8.0

Herramienta UML profesional que soporta el desarrollo de software desde el análisis y diseño orientado a objeto, construcción, pruebas y despliegue. Entre muchas de sus ventajas permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Además, permite la importación y exportación de archivos XML de diferentes versiones. Presenta licencia gratuita y comercial. Es multiplataforma, fácil de instalar y actualizar, compatible entre ediciones.

Características principales:

- Facilita el modelado de UML, ya que proporciona herramientas específicas para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.
- Controla que el modelado con UML sea correcto.
- Al disponer de un repositorio común, es posible visualizar el mismo elemento en varios diagramas, evitando duplicidades.
- Permite integrarse con otras aplicaciones, como herramientas ofimáticas, lo cual aumenta la productividad.
- Permite el trabajo en grupo, proporcionando herramientas de compartición de trabajo.
- Facilita la reutilización, ya que es una herramienta centralizada donde se encuentran los modelos utilizados para otros proyectos.

- Permite generar código de forma automática, reduciendo los tiempos de desarrollo y evitando errores en la codificación del software.
- Permite generar diversos informes a partir de la información introducida en la herramienta. (Visual Paradigm, 2017)

1.5. Lenguaje Unificado de Modelado

El modelado de sistemas de software es una técnica para tratar con la complejidad esencial de estos sistemas. El uso de modelos ayuda al ingeniero de software a visualizar el sistema a construir. Los modelos de un nivel de abstracción mayor pueden utilizarse para la comunicación con el cliente. En este sentido el Lenguaje Unificado de Modelado (*UML, por sus siglas en inglés de Unified Modeling Language)*, es un lenguaje capaz de abstraer cualquier tipo de sistema mediante la utilización de diagramas. Permite realizar presentaciones gráficas que contienen la información relevante. El modelado visual se usa para especificar, visualizar, construir y documentar artefactos del software. Se usa para entender, diseñar, hojear, configurar, mantener y controlar la información sobre tales sistemas. Este lenguaje permite que todo software de diseño orientado a objetos se visualice, especifique y documente con un lenguaje común. Cuenta con su propia metodología de desarrollo basada en componentes, notaciones estándar y semánticas esenciales para el modelado de un sistema orientado a objetos. (Sierra, 2013)

Para la presente investigación se decide seleccionar UML 5.0 al requerirse un lenguaje gráfico, con el objetivo de especificar y documentar el sistema de software de modo estándar.

1.6. Lenguaje de programación

Es un lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas. (Alonso Velazquez, 2014)

1.6.1. Java 1.8

Java es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana. También posee otras características importantes:

- Es un lenguaje compilado, que genera ficheros de clases, las que son en realidad interpretadas por la máquina virtual de Java, siendo esta la que mantiene el control sobre las clases que se estén ejecutando.
- Es un lenguaje multiplataforma: el mismo código escrito en Java que funciona en un SO, funcionará en cualquier otro que tenga instalada la máquina virtual de Java.

- Es un lenguaje seguro: la máquina virtual, al ejecutar el código escrito en Java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.
- Gracias al API de Java, se puede ampliar el lenguaje para que sea capaz de comunicarse con equipos mediante red, acceder a bases de datos, crear páginas HTML dinámicas y crear aplicaciones visuales. (Gosling, y otros, 2019)

Se determinó utilizar Java para la implementación de la propuesta de solución, ya que es el lenguaje nativo en el desarrollo de aplicaciones Android, además es multiplataforma y el equipo de desarrollo posee experiencias, capacidades y habilidades con el desarrollo de aplicaciones en Java.

1.7. Entorno de desarrollo integrado

Un entorno de desarrollo integrado, llamado IDE (siglas en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios. Ha sido empaquetado como un programa de aplicación, es decir, que consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

Existen varios IDE de múltiples lenguajes tales como Eclipse, Oracle JDeveloper, Codenvy, Microsoft Visual Studio, NetBeans, etc. Para el desarrollo de la solución propuesta se define utilizar como IDE a Android Studio en su versión 1.2 y a continuación se describen sus principales características:

1.7.1. Android Studio 1.2

Es el entorno de desarrollo integrado oficial de Android, fue creado directamente por Google (buscador) para el desarrollo de aplicaciones móviles. Android Studio tiene herramientas especializadas que hacen el proceso más fácil, rápido y sencillo. Posee un editor gráfico que permite generar interfaces sin la necesidad de crear código. Se tiene acceso a herramientas personalizadas y optimizadas para trabajar en Android, de forma que las interfaces son específicamente pensadas para el trabajo en entornos Android. Posee completamiento automático de código y corrección de errores. Además, utiliza una licencia de software libre Apache 2.0, está programado en Java y es multiplataforma. (Digital Learning, 2015)

1.8. Marco de trabajo

En el desarrollo de software, un marco de trabajo o framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Normalmente, un marco de trabajo incluye soporte de programas y librerías, además de un lenguaje de programación que sirve de apoyo en el desarrollo y la integración de los diferentes componentes de una aplicación.

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. En general estos se diseñan para resolver la mayor parte de la complejidad de los desarrollos, aprovechando las mejores prácticas y facilitando la construcción de nuevas aplicaciones. (SEDICI, 2012)

Para el desarrollo de la solución fue necesario el empleo del marco de trabajo SDK(Software Development Kit) en su versión 28.0 para la integración del IDE con los plugins correspondientes para el desarrollo de aplicaciones en Android. A continuación, se describen sus principales características:

1.8.1. Software Development Kit 28.0

Un SDK (por sus siglas en inglés) es un conjunto de herramientas de desarrollo de software que permite a un desarrollador de software crear una aplicación informática para un sistema concreto, por ejemplo, ciertos paquetes de software, entornos de trabajo, plataformas de hardware, computadoras, videoconsolas, sistemas operativos, etcétera. Es tan sencillo como una interfaz de programación de aplicaciones o API (del inglés Application Programing Interface) creada para permitir el uso de cierto lenguaje de programación, o puede, también, incluir hardware sofisticado para comunicarse con un determinado sistema embebido. Las herramientas de desarrollo de software más comunes incluyen soporte para la detección de errores de programación como un IDE y otras utilidades. Los SDK frecuentemente también incluyen códigos de ejemplo y notas técnicas de soporte u otra documentación de soporte para ayudar a clarificar ciertos puntos del material de referencia primario. (AndroidDev, 2019)

1.9. Sistema gestor de bases de datos

Un Sistema Gestor de Base de Datos (SGBD) es un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información de un modo más factible. (Iruela, 2016) En el caso de la presente investigación, se decide utilizar como SGBD MySQL, que a continuación se describe:

1.9.1. MySQL

Es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation, este sistema es de bajo costo de propiedad, estable y seguro. Posee excelente documentación, ofrece soporte de alta calidad para sus productos, incluyendo un servicio que permite a los desarrolladores de MySQL iniciar sesión en el servidor para corregir problemas de forma proactiva y ayudar con la optimización. Está disponible para una gran variedad de arquitecturas de computadoras y muchos sistemas operativos diferentes. Entre sus principales características se encuentran:

- Es de código abierto
- Fácil de usar

- Multiplataforma
- Posee gran utilidad de carga rápida de datos. (MySQL.com, 2019)

1.10. Conclusiones del capítulo

Luego de finalizado el capítulo se concluye que:

- Durante la elaboración del marco teórico conceptual, se establecieron los principales referentes teóricos-metodológicos que sustentan la interrelación clientes y proveedores, permitiendo obtener conceptos e ideas para el apoyo de la propuesta de solución.
- Se realizó un estudio de soluciones existentes en el mundo que realizan el proceso compra y venta de productos y servicios, para sentar las bases en la elaboración de la propuesta de solución.
- Se explicó de forma breve la metodología, las herramientas y tecnologías a utilizar, lo que posibilitó un entendimiento de estas.

CAPÍTULO 2. ANÁLISIS Y DISEÑO.

En el presente capítulo se describe la propuesta de solución para dar cumplimiento al problema planteado. En el mismo se reflejan las dos primeras etapas de la metodología XP obteniendo artefactos necesarios para la elaboración del sistema. Además, se expone tanto la arquitectura utilizada para la creación de la aplicación como los patrones de diseños utilizados en la misma, incluyendo las tarjetas CRC, brindando una entrada fundamental al desarrollo del mismo.

2.1. Propuesta de solución

Con el objetivo de facilitar la interrelación de clientes y proveedores en las empresas estatales de Cuba, se realiza la aplicación Android de interrelación de clientes y proveedores. Esta será capaz de gestionar los servicios y productos que se definen en las empresas estatales mediante móviles. Permitiendo enviar y recibir notificaciones de los mismos. Para un mejor entendimiento se muestra a continuación las Ilustraciones 1 y 2 apoyadas de una breve descripción según el rol que la inicia.

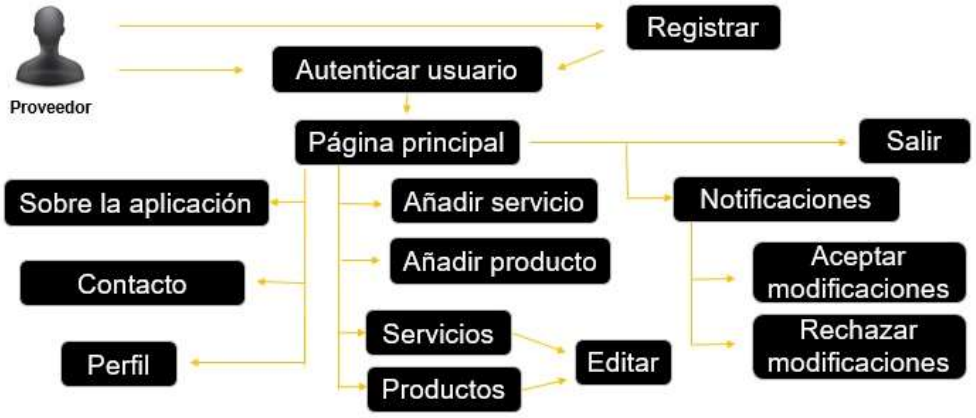


Ilustración 1.Mapa de navegación para el usuario Proveedor

El usuario Proveedor podrá autenticarse si ya está registrado en el sistema, o podrá registrarse. Una vez autenticado pasará a la página principal, donde podrá añadir servicios o productos y modificarlos. Tendrá las opciones: Sobre la aplicación, Contacto, Perfil, Salir y Notificaciones. En esta última podrá aceptar las modificaciones que le propone un cliente sobre un servicio o un producto o rechazarlas.

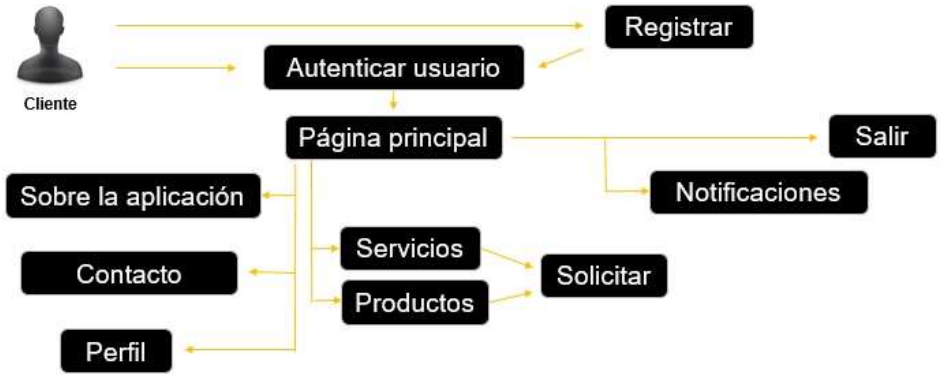


Ilustración 2.Mapa de navegación para el usuario Cliente

El usuario Cliente podrá autenticarse si ya está registrado en el sistema, o podrá registrarse. Una vez autenticado pasará a la página principal, donde verá todos los servicios y productos existentes en el sistema, los que podrá solicitar según sus necesidades. Tendrá las opciones: Sobre la aplicación, Contacto, Perfil, Salir y Notificaciones. En esta última podrá revisar si el proveedor de un producto o servicio que solicitó, acepta o no su solicitud.

2.2. Metodología XP. Fase I: Planificación

La metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente. El proyecto comienza recopilando “Historias de usuarios”, las que sustituyen a los tradicionales “casos de uso”. Una vez obtenidas las “historias de usuarios”, se evalúa rápidamente el tiempo de desarrollo de cada una. Una vez realizadas estas estimaciones, se establece un plan o cronograma de entregas (“Release Plan”), este establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Las historias de usuarios seleccionadas, para cada entrega, son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden establecido. (López, 2016)

Aunque la metodología no exige un modelo del dominio o negocio, para un mejor entendimiento del entorno se presenta el mapa conceptual siguiente. (Ver ilustración 3).

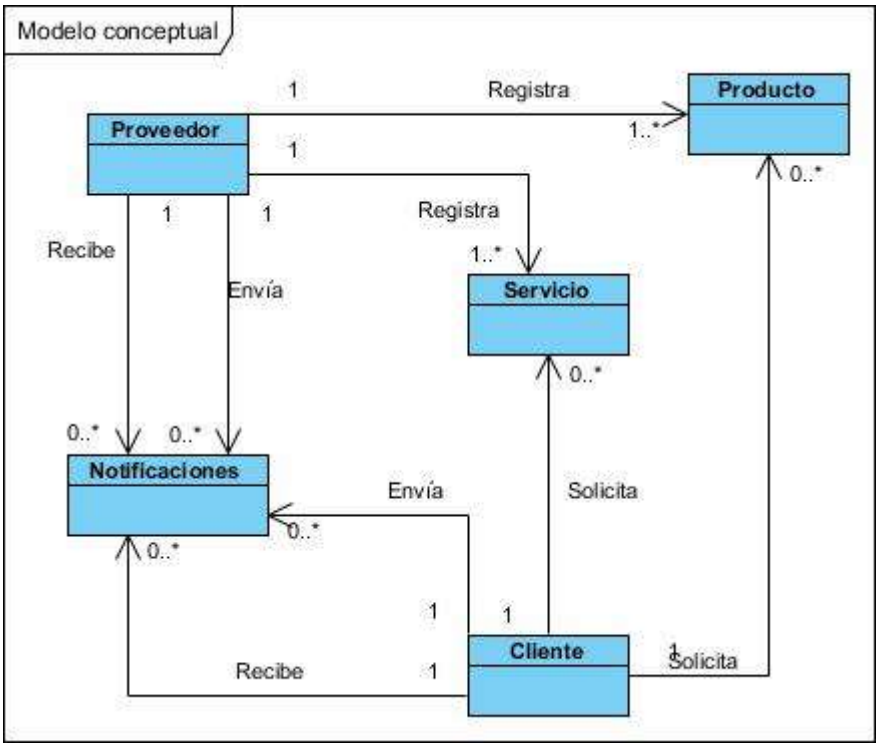


Ilustración 3.Mapa conceptual

A continuación, se describen cada una de las actividades que se llevaron a cabo en la fase como guía del desarrollo de la aplicación móvil para la propuesta de solución teniendo en cuenta el proceso de interrelación entre clientes y proveedores, objetivo general de la presente investigación.

2.2.1. Historias de usuario

Las historias de usuario (HU) es la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (López, 2016)

Respecto de la información contenida en la historia de usuario, existen varias plantillas sugeridas, pero no existe un consenso al respecto. En muchos casos sólo se propone utilizar un nombre y una descripción o sólo una descripción, más quizás una estimación de esfuerzo en días. Beck en su libro presenta un ejemplo de ficha (customer story and task card) en la cual pueden reconocerse los siguientes contenidos: fecha, tipo de actividad (nueva, corrección, mejora), prueba funcional, número de historia, prioridad técnica y del cliente, referencia a otra historia previa, riesgo, estimación

técnica, descripción, notas y una lista de seguimiento con la fecha, estado, elementos por terminar y comentarios. (López, 2016)

Las historias de usuarios quedan estructuradas de la siguiente forma:

- **Nombre:** Nombre descriptivo de la historia de usuario.
- **Número:** Número consecutivo que identifica a cada historia de usuario.
- **Iteración:** Iteración en la cual será implementada la historia de usuario.
- **Prioridad:** Grado de prioridad que le asigna el cliente a la historia de usuario. Los valores que puede tomar son (Alta, Media o Baja).
- **Complejidad:** Grado de complejidad que le asigna el equipo de desarrollo a la historia de usuario luego de analizarla. (Alta, Media o Baja).
- **Descripción:** Descripción simple que brinda el cliente sobre lo que debe hacer la funcionalidad en cuestión.
- **Observaciones:** Condiciones que deben tenerse en cuenta para el desarrollo de la funcionalidad.

A continuación, se describen las HU #1 y #2 definidas para el desarrollo de la propuesta de solución, el resto pueden ser consultadas en los anexos (Ver Anexo #1) de la investigación.

Tabla 1. HU-1 Autenticar usuario

Número: **HU-1**	**Nombre:** Autenticar usuario
Iteración: **1**	
Prioridad: **Alta**	**Complejidad**: Alta
Descripción: **El usuario debe ingresar su usuario y su contraseña para poder acceder al sistema. Teniendo en cuenta los permisos y el rol que va a ocupar.**	
Observaciones: **Si esta errónea la contraseña o el usuario el sistema muestra un mensaje informativo.** **Si todos los datos son correctos el usuario accederá al sistema.**	
Prototipo:	

Fuente: Elaboración propia

Tabla 2. HU-2 Gestionar Usuario

Número: **HU-2**	**Nombre:** Gestionar usuario
Iteración: **1**	
Prioridad: **Baja**	**Complejidad**: Alta
Descripción: **Luego de que el usuario escoja la opción de registrarse el sistema mostrará la interfaz correspondiente, en la cual el usuario tendrá que introducir los datos necesarios. Como nombre de la empresa, provincia, su rol, usuario y contraseña.** **Una vez registrado, el usuario podrá modificar sus datos en el sistema (contraseña, información sobre él)** **El administrador de la aplicación puede eliminar un usuario.**	
Observaciones: **Si se dejan campos en blanco cuando se va a crear un usuario, el sistema muestra un mensaje de aviso.** **Si se rellenan campos con datos erróneos cuando se va a crear un usuario, el sistema muestra un mensaje de aviso.** **Si todos los datos son correctos el sistema muestra un mensaje informativo.**	

Fuente: Elaboración propia

Requisitos funcionales

Iteración 1. Prioridad Alta

RF1: Crear Servicio (nombre string, descripción string, correo string, teléfono int, precio int, moneda string)

RF2: Modificar Servicio (nombre string, descripción string, correo string, teléfono int, precio int, moneda string)

RF3: Eliminar Servicio (id int)

RF4: Listar Servicio (id int)

RF5: Crear Producto (nombre string, descripción string, correo string, teléfono int, precio int, moneda string)

RF6: Modificar Producto (nombre string, descripción string, correo string, teléfono int, precio int, moneda string)

RF7: Eliminar Producto (id int)

RF8: Listar Producto (id int)

Iteración 2. Prioridad Media

RF9: Buscar Servicio (nombre string)

RF10: Recibir notificación (idUsuario)

RF11: Enviar notificación (idUsuario int, nombre string descripción, string)

Iteración 3. Prioridad Baja

RF12: Autenticar usuario (usuario string, contraseña string)

RF13: Crear usuario (empresa string, provincia string, rolstring, usuario string, contraseña string)

RF14: Modificar usuario (contraseña string, información string)

RF15: Eliminar usuario (id int)

RF16: Listar Usuario (id int)

2.2.2. Estimación de esfuerzos

Las estimaciones de esfuerzo asociado a la implementación de las HU se establecen utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las HU generalmente valen de 1 a 3 puntos, según su complejidad. Por otra parte, se mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración (López, 2016).

La estimación de esfuerzo se ha realizado para cada una de las HU identificadas anteriormente, los resultados se muestran a continuación:

Tabla 3.Estimación de esfuerzos

Historia de Usuario	Puntos de estimación
Autenticar usuario	1
Gestionar Usuario	2
Gestionar Servicio	3
Gestionar Producto	3
Enviar Notificación	1
Recibir Notificación	1
Buscar Servicio	1

Fuente: Elaboración propia1

2.2.3. Iteraciones

En esta actividad se definen las HU que se implementarán en cada iteración, dando paso a la implementación de cada una de las funcionalidades de la solución propuesta.

Al final de cada iteración se genera un entregable funcional que implementa las historias de usuario asignadas a la iteración. Como las historias de usuario no tienen suficiente detalle como para permitir su análisis y desarrollo, al principio de cada iteración se realizan las tareas necesarias de análisis, recabando con el cliente todos los datos que sean necesarios. Al final de la última iteración el sistema estará listo para entrar en desarrollo. (López, 2016).

Para un mejor entendimiento y aunque la metodología XP no lo exige, se propone tener en cuenta en las iteraciones los RF relacionados por cada HU, teniéndose en cuenta la prioridad.

Iteración 1: En esta primera iteración se tiene como objetivo la implementación de la historia de usuario #3(Gestionar Servicio), que contiene los requisitos funcionales #1, #2, #3 y #4. También la historia de usuario #4(Gestionar Producto), que contiene los requisitos funcionales #5, #6, #7 y #8, pues tienen prioridad alta.

Iteración 2: En esta segunda iteración se tiene como objetivo la implementación de la historia de usuario #7(Buscar servicio), #6(Recibir notificación) y #5(Enviar notificación), que contiene los requisitos funcionales #9, #10, y #11 teniendo prioridad media.

Iteración 3: En esta iteración se tiene como objetivo la implementación de las historias de usuario #1(Autenticar Usuario) y #2(Gestionar Usuario), que contienen los requisitos funcionales #12, #13, #14, #15 y #16, pues tienen prioridad media.

Plan de duración de las iteraciones

Este artefacto tiene como objetivo especificar detalladamente el orden de desarrollo de las HU dentro de cada iteración y la duración total de cada una de ellas, brindando así una mayor organización.

Tabla 4.Plan de duración de las iteraciones

Iteración	Historia de Usuario	Duración total de Iteraciones
Iteración #1	Gestionar Servicio Gestionar Producto	6 semanas
Iteración #2	Buscar Servicio	3 semanas
	Enviar Notificación	
	Recibir Notificación	
Iteración #3	Autenticar Usuario	3 semanas
	Gestionar Usuario	

Fuente: Elaboración propia

2.2.4. Plan de entregas

En el plan de entrega se definen las Historias de Usuario que se entregarán al final de cada iteración. A continuación, se presenta el plan de entregas ideado, teniéndose en cuenta que la implementación comience en febrero del año 2019.

Tabla 5.Plan de entregas

Iteración	Duración	Fecha de Inicio	Fecha de Fin
Iteración #1	6 semanas	1-2-2019	14-3-2019
Iteración #2	3 semanas	15-3-2019	4-4-2019

Iteración #3	3 semanas	5-4-2019	26-4-2019

Fuente: Elaboración propia

2.3. Diseño

La metodología XP hace especial énfasis en los diseños simples y claros, presentando gran atención a conceptos tales como simplicidad, recodificación, disponibilidad del cliente, uso de estándares y otros. (Pressman, 2010)

Con el objetivo de sentar las bases para la fase de Desarrollo de la solución propuesta, en este epígrafe se describe la representación arquitectónica, los patrones de diseños a utilizar y por último las tarjetas CRC (Clases-Responsabilidad-Colaboración) propuesta por la metodología como artefacto en esta fase.

2.3.1. Arquitectura de la solución

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software para expresar una estructura de organización. Brindan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. **Fuente especificada no válida.**

Según la Real Academia Española (RAE), un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

Para el desarrollo de la aplicación se utiliza el patrón Modelo-Vista-Controlador (MVC). El cual es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento. Garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo.

Se especifica en la siguiente figura un ejemplo de MVC para dicho componente:

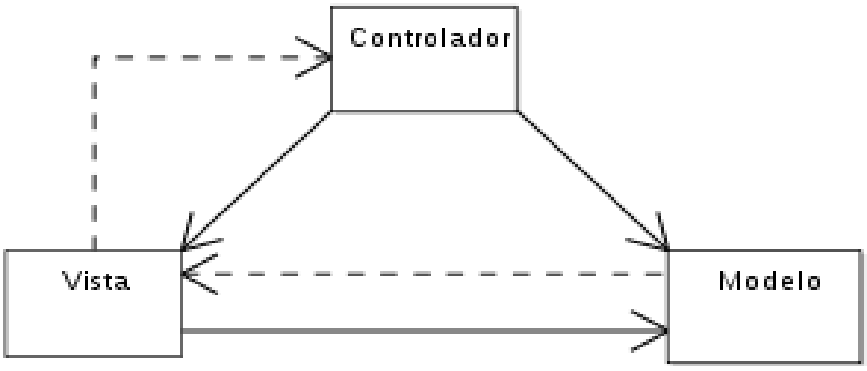


Ilustración 4.Modelo-Vista-Controlador (MVC)

Nota: las líneas sólidas indican una asociación directa, y las punteadas una indirecta.

En el **modelo** se manejan los datos del sistema y las operaciones asociadas a esos datos, además de las clases referentes a las notificaciones del componente. La **vista** define y gestiona cómo se presentan los datos al usuario, además de mostrar todas los *Activity* que son las clases que muestra la interfaz del sistema. El componente **controlador** dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo.
Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual, además de mostrar las clases referentes a los repositorios del componente.

Se usa cuando existen múltiples formas de ver e interactuar con los datos. También se utiliza al desconocerse los requerimientos futuros para la interacción y presentación.

Ventajas: Permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.
Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Desventajas: Puede implicar código adicional y complejidad de código cuando el modelo de datos y las interacciones son simples.

2.3.2. Patrones de diseño

Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Gamma, 1995).
En síntesis, son un esqueleto básico que cada diseñador adapta a las peculiaridades de su aplicación. Para el desarrollo de la aplicación se usó patrones del tipo GRASP y GoF los cuales serán descritos a continuación:

Patrones GRASP

Los Patrones GRASP (Acrónimo de General Responsibility Assignment Software Patterns) describen los principios fundamentales para asignar responsabilidades a los objetos, para la solución se utilizaron los siguientes:

- **Experto:** Asignar una responsabilidad al experto, clase que tiene la información para poder realizarla, y en información es el principio básico de asignación de responsabilidades.

El patrón experto posibilita que se mantenga el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener, además se distribuye el comportamiento entre las clases que contienen la información requerida, por tanto, se estimula las definiciones de clases más cohesivas y ligeras que son más fáciles de comprender y manejar. Se soporta normalmente una alta cohesión. Se pone de manifiesto en la clase Servicio, que es la encargada de crear todos los servicios en el sistema

- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Se facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza las actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

  Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Se manifiesta en el fichero Notificacion que se encarga de controlar todas las notificaciones que se realizan en el sistema.

- **Alta cohesión:** La información que almacena una clase debe ser coherente y debe estar relacionada con la clase. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

  Su función es asignar una responsabilidad de manera que la cohesión permanezca alta. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Los elementos pueden ser clases, subsistemas, etcétera.

  Posibilita que se incremente la claridad y facilita la comprensión del diseño, simplifica el mantenimiento y las mejoras. Soporta a menudo bajo acoplamiento y el grado bajo de funcionalidades altamente relacionadas. Incrementa la reutilización porque una clase cohesiva se puede utilizar para un propósito muy específico. El patrón se evidencia en la creación de las notificaciones.

- **Bajo acoplamiento:** Debe haber pocas dependencias entre las clases. En caso de producirse una modificación en alguna de ellas, se debe lograr la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

El patrón permite que en caso de modificarse algún elemento sus cambios no afecten a otro componente. Son fáciles de comprender de manera aislada y conveniente para la reutilización. La utilización de este patrón se evidencia cuando se envía una notificación a un usuario determinado, mediante la funcionalidad enviar notificación, donde se hace uso de diferentes servicios.

2.3.3. Tarjeta Clase Responsabilidad Colaborador (CRC)

XP estimula el uso de las tarjetas CRC como un mecanismo mejor para pensar el software en un contexto orientado a objetos, estas tarjetas son el único producto del trabajo de diseño que se genera como parte del proceso XP (Pressman, 2010).

La utilización de tarjetas CRC (Class-Responsibility-Collaboration) es una técnica de diseño orientado a objetos propuesta por Kent Beck. El objetivo de la misma es hacer, mediante tarjetas, un inventario de las clases que vamos a necesitar para implementar el sistema y la forma en que van a interactuar, de esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objeto de que el diseño sea lo más simple posible verificando las especificaciones del sistema (López, 2016). Las tarjetas CRC están esquematizadas de la siguiente forma:

- **Clase:** Nombre de la clase a la cual pertenece la tarjeta CRC
- **Responsabilidad:** Describe cuales son las funcionalidades que deben ser implementadas por la clase.
- **Colaboración**: Enumera las diferentes clases con las cuales tiene relación la clase a la cual pertenece la tarjeta CRC. Se representa la dirección de la clase dentro del código del módulo de configuración.

Tabla 6.Tarjeta CRC #1

Clase: **Usuario**	
Responsabilidad: • **crearUsuario():** funcionalidad que permite crear un nuevo usuario. • **usuarioRegistrado():** funcionalidad que permite que el usuario entre en el sistema.	**Colaboración**:

• **mostrarUsuario():** funcionalidad que permite mostrar todos los usuarios existentes en el sistema. • **eliminarUsuario(id):** funcionalidad que permite eliminar a un usuario.	

Fuente: Elaboración propia

Tabla 7.Tarjeta CRC #2

Clase:**Servicio**	
Responsabilidad: • **adicionarServicio():** funcionalidad que permite al usuario crear un nuevo servicio. • **eliminarServicio(idServicio):** funcionalidad que permite al usuario eliminar un servicio. • **mostrarServicio():** funcionalidad que permite mostrar al usuario los servicios existentes en el sistema. • **actualizarServicio(idServicio,values):** funcionalidad que permite al usuario modificar un servicio en el sistema.	**Colaboración**: • Usuario

Fuente: Elaboración propia

Tabla 8.Tarjeta CRC #3

Clase: **Notificacion**	
Responsabilidad:	**Colaboración**:

• **crearNotificacion():** funcionalidad que permite enviar una notificación al usuario. • **mostrarNotificacion(idUsuario):**funcionalidad que permite mostrar las notificaciones del usuario.	• Usuario • Servicio • Producto

Fuente: Elaboración propia

Tabla 9.Tarjeta CRC #4

Clase: **Producto**	
Responsabilidad: • **adicionarProducto():** funcionalidad que permite al usuario crear un nuevo producto. • **eliminarProducto (idProducto):** funcionalidad que permite al usuario eliminar un producto. • **mostrarProducto ():** funcionalidad que permite mostrar al usuario los productos existentes en el sistema. • **actualizarProducto (idProducto,values):** funcionalidad que permite al usuario modificar un producto en el sistema.	**Colaboración**: • Usuario

Fuente: Elaboración propia

2.4 Características no funcionales

Las características no funcionales (CNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. A menudo

son aplicados en su totalidad al sistema y normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2005).

La siguiente tabla expone las características no funcionales definidas para el módulo, atendiendo a sus clasificaciones.

Tabla 10.Características no funcionales

Identificador	Descripción
Usabilidad	
CNF# 1	La aplicación debe presentar un acceso fácil y rápido.
CNF # 2	Ofrecer interfaz amigable, interactiva e intuitiva.
CNF # 3	La aplicación será distribuida en idioma español.
Fiabilidad	
CNF # 5	El acceso a la aplicación se realizará mediante roles.
Eficiencia	
CNF # 6	Proporcionar una aplicación que sea capaz de ejecutar todas las peticiones y operaciones en un máximo de 3 segundos.
Interfaz	
CNF # 7	La interfaz de la aplicación contendrá los datos de forma estructurada, permitiendo la interpretación correcta de la información.
CNF # 8	Todos los textos y mensajes en pantalla serán mostrados en idioma español.
Software	
CNF # 9	Sistema Gestor de Base de Datos MySQL
CNF # 10	Cliente: Sistema Operativo Android 4.0
Seguridad	
CNF # 11	El módulo debe garantizar la seguridad a través de la autenticación de los usuarios
CNF # 12	Las diferentes áreas del módulo se encontrarán protegidas contra acceso no autorizado utilizando roles y grupos de usuarios.
CNF # 13	La aplicación debe permitir que la contraseña se almacene de manera encriptada en la base de datos.
Hardware	
CNF # 14	Servidor: Intel Core i3, 2GB de memoria RAM

2.5 Conclusiones del capítulo

Una vez terminado este capítulo se puede concluir que:

- El modelo conceptual de la propuesta de solución permitió representar los conceptos significativos del negocio y la relación que existe entre estos.

- La fase de Planificación propuesta por la metodología XP permitió describir un total de 7 Historias de Usuario (HU) del sistema, obtener una estimación de cada una de ellas, así como las iteraciones y el plan de entrega del producto.

- La descripción de las HU identificadas permitió un mejor entendimiento para su futura implementación.

- La identificación de las características no funcionales permitió conocer las condiciones bajo las cuales debe operar la aplicación Android, identificándose un total de 14 requisitos no funcionales.

- La fase de Diseño favoreció obtener una representación de la arquitectura del sistema (MVC), así como los patrones de diseño a utilizar y las tarjetas CRC, esto último, como artefacto propuesto en la fase por la metodología que guía el desarrollo de la presente investigación.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

En este capítulo se desarrollan las dos últimas fases que plantea la metodología XP: Desarrollo y Pruebas. En el caso de la primera se da inicio con las tareas de ingeniería, donde cada HU se convierte en tareas a cumplir por los programadores, y por último se definen los estándares de codificación a tener en cuenta durante el desarrollo. Ya en la última fase se realizan pruebas de aceptación para verificar el correcto funcionamiento de la aplicación Android que se obtuvo como resultado de la presente investigación.

3.1. Fase III: Desarrollo

En esta fase se encuentra el desarrollo del código, el cual se sustenta en buenas prácticas planteadas por el ciclo de vida de XP, entre estas se encuentran: disponibilidad del cliente, uso de estándares, programación dirigida por las pruebas (“test-driven programming”), programación en pares, integraciones permanentes y ritmo sostenido. (López, 2016)

3.1.1 Tareas de ingeniería

Asociado a cada iteración se encuentra la planificación de las tareas de ingeniería o programación, cada HU se transforma en estas tareas que son desarrolladas por programadores, dentro del equipo de desarrollo, aplicando la práctica de la programación en parejas. Para cada iteración se realizó la distribución de tareas en correspondencia con las HU que se desarrollaron.

Tabla 11.Tareas de ingeniería

Historia de usuario:	**No. Tarea de Ingeniería**	**Tareas de ingeniería**:
Gestionar usuario	1	Crear Usuario
	2	Modificar Usuario
	3	Eliminar Usuario
	4	Listar Usuario
Gestionar Servicio	5	Crear Servicio
	6	Modificar Servicio
	7	Eliminar Servicio
	8	Listar Servicio
Buscar Servicio	9	Buscar Servicio

Enviar Notificación	10	Enviar Notificación
Recibir Notificaciones	11	Recibir Notificaciones
Autenticar usuario	12	Autenticar Usuario
Gestionar Producto	13 14 15 16	Crear Producto Modificar Producto Listar Producto Eliminar Producto

Fuente: Elaboración propia

A continuación, se evidencian las tareas ingeniería o programación en las que fueron desglosadas la HU Crear proveedor, para un mejor funcionamiento de la aplicación. El resto se pueden consultar en el Anexo X. Para la confección de cada una de las tareas se utilizó la tabla que cuenta con los siguientes campos:

- **Número de la tarea:** Numeración continua que identifica a la tarea.
- **No. de HU:** Número de la HU a la cual pertenece.
- **Nombre de la tarea:** Identificación literal de la tarea.
- **Tipo de tarea:** Tipo de tarea, dígase diseño, desarrollo, prueba.
- **Puntos estimados:** Representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización.
- **Fecha inicio:** Fecha estimada de inicio de realización.
- **Fecha fin:** Fecha estimada de fin de realización.
- **Descripción:** Se describe en qué consiste la tarea y qué elementos deben cumplirse para declarar la tarea terminada.

A continuación, se describen las Tareas de Ingeniería #1 y #2, el resto pueden ser consultadas en los anexos (Ver Anexo #2) de la investigación.

Tabla 12.Tarea de ingeniería #1 desglosada

Número de la tarea: 1	**No. De HU:** 4
Nombre de la tarea: Crear Usuario	
Puntos de estimación: 1	
Tipo de tarea: desarrollo	**Responsable:** Julio César Vega Fuentes
Fecha inicio:1-2-2019	**Fecha fin:**14-2-2019
Descripción: Cuando el usuario selecciona la opción de registrarse se muestra una interfaz donde deberá introducir todos los datos necesarios para registrarse en el sistema (empresa, provincia, rol en el sistema, usuario y contraseña).	

Fuente: Elaboración propia

Tabla 13.Tarea de ingeniería #2 desglosada

Número de la tarea: **2**	**No. De HU:** 4
Nombre de la tarea: **Modificar Usuario**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**15-2-2019**	**Fecha fin:**21-2-2019
Descripción: **El usuario selecciona la opción Ver Perfil y se mostrará una interfaz donde podrá modificar sus datos (información sobre él y contraseña) en el sistema.**	

Fuente: Elaboración propia

3.1.2. Estándares de codificación

Los estándares de codificación establecen un conjunto de reglas que los desarrolladores deben seguir para escribir el código fuente de un software. Esto tiene como objetivo que dicho código sea entendible por cualquier desarrollador del grupo de trabajo, no solo por el que lo creó; garantizando un mantenimiento del sistema más rápido y eficiente, ya sea creando nuevas funcionalidades o modificando las ya existentes. (Arias Calleja, 2014)

Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible (Arias Calleja, 2014). Para el desarrollo de la propuesta de solución se utilizan varios estándares de codificación, tales como:

Nomenclatura general

- El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras se utilizará el estilo de escritura lowerCamelCase, que dicta que para un nombre compuesto por varias palabras comenzará con minúscula, pero todas las palabras internas que lo componen comienzan con mayúscula.

```
public void solicitarServicio(){
    final Button solicitarServicio = (Button) findViewById(R.id.btn_SolicitarServicio);
    solicitarServicio.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Bundle parametros = getIntent().getExtras();
            int idServicio = parametros.getInt( key: "ID");
            final String nombreServ = parametros.getString( key: "Nombre_Servicio");
```

Ilustración 5.Fragmento del método para solicitar un servicio.

Paquetes

- Por defecto todos los paquetes se escribirán en minúsculas y sin utilizar caracteres especiales.
- Se tendrá, así mismo, otro nivel extra dentro del paquete definido como el nombre del proyecto o de la capa.

Identación

- En el contenido siempre se identificará con tabulaciones, nunca utilizando espacios en blanco.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()){
        case R.id.mo_verPerfil:
            Intent intent4 = new Intent( packageContext: ActivityRegistradoCliente.this,ActivityVerPerfil.class);
            startActivity(intent4);
            break;
        case R.id.mo_Acerca_de:
            Intent intent = new Intent( packageContext: this,ActivitySobreLaAplicacionCliente.class);
            startActivity(intent);
            break;
        case R.id.mo_Contacto:
            Intent intent1 = new Intent( packageContext: this,ActivityContactoCliente.class);
            startActivity(intent1);
            break;
        case R.id.mo_campana_llena:
            Intent intent2 = new Intent( packageContext: this,ActivityNotificacionesCliente.class);
            startActivity(intent2);
            break;
        case R.id.mo_cerrarSesion:
            Intent intent3 = new Intent( packageContext: this,ActivityLogin.class);
            finish();
            startActivity(intent3);
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

Ilustración 6.Método de opciones.

Clases

- El nombre de las clases comenzará con mayúsculas y cada salto de palabras debe iniciar con mayúsculas.

Ilustración 7.Ejemplo de clases

Métodos

- Los métodos deberán ser verbos (en infinitivo), en mayúsculas y minúsculas con la primera letra del nombre en minúsculas, y con la primera letra de cada palabra interna en mayúsculas.

- No se permiten caracteres especiales.

```
public void editarPerfil(){
    final TextView edit = (TextView)findViewById(R.id.tv_VerPerfilEditar);
    edit.setOnClickListener(new View.OnClickListener() {
```

Ilustración 8.Ejemplo del método editarPerfil()

Nombre de variables

- El nombre de las variables debe empezar con letra minúsculas y de existir un salto de palabra comenzaría con mayúscula.

```
Bundle parametros = getIntent().getExtras();
String nombreServicio = parametros.getString( key: "Nombre_Servicio");
String telefono = parametros.getString( key: "Telefono");
String correo = parametros.getString( key: "Correo");
String descripcion = parametros.getString( key: "Descripcion");
String precio = parametros.getString( key: "Precio");
```

Ilustración 9.Ejemplo de variables

3.2. Fase IV: Pruebas

La metodología XP propone la aplicación de las pruebas de aceptación utilizadas para evaluar si se obtuvo las funcionalidades deseadas por parte del cliente. (López, 2016)

3.2.1 Pruebas de rendimiento

Ilustración 10.Pruebas de rendimiento

Al realizar las pruebas de rendimiento a la aplicación, en diferentes dispositivos móviles, se comprobó que el uso del CPU no excede el 12%, la memoria RAM no excede los 65 MB y el uso de la red no excede los 0.8 kb/s, demostrándose que el rendimiento de la aplicación es adecuado.

3.2.2 Pruebas de aceptación

Las pruebas de aceptación, también llamadas pruebas funcionales son supervisadas por el cliente basándose en los requerimientos tomados de las historias de usuario. En todas las iteraciones, cada una de las historias de usuario seleccionadas por el cliente deberá tener una o más pruebas de

aceptación, de las cuales deberán determinar los casos de prueba e identificar los errores que serán corregidos.

Las pruebas de aceptación son pruebas de caja negra, que representan un resultado esperado de determinada transacción con el sistema. Para que una historia de usuario se considere aprobada, deberá pasar todas las pruebas de aceptación elaboradas para dicha historia. Es importante resaltar la diferencia entre las pruebas de aceptación y las unitarias en lo que al papel del usuario se refiere. Mientras que en las pruebas de aceptación juega un papel muy importante seleccionando los casos de prueba para cada historia de usuario e identificando los resultados esperados, en las segundas no tiene ninguna intervención por ser de competencia del equipo de programadores (Luis Miguel Echeverry Tabón, y otros, 2007).

Las pruebas de aceptación tienen más peso que las unitarias ya que constituyen un indicador de la satisfacción del cliente con la solución además de marcar el final de una iteración y el comienzo de la siguiente. Se recomienda que el cliente sea quien diseñe estas pruebas o que al menos participe de manera activa en el proceso.

Como resultado de las pruebas de aceptación se obtendrán artefactos descritos en tablas, estas contarán con los siguientes campos:

- **Código:** identificador de la prueba realizada.
- **HU:** número de la historia de usuario a la que hace referencia la prueba a realizar.
- **Nombre:** nombre de la prueba a realizar.
- **Descripción:** se describe la funcionalidad que se desea probar.
- **Condiciones de Ejecución:** condiciones que deben cumplirse para poder llevar a cabo el caso de prueba, estas condiciones deben ser satisfechas antes de la ejecución del caso de prueba para que se puedan obtener los resultados esperados.
- **Pasos de Ejecución:** pasos seguidos durante el desarrollo de la prueba, se tendrá en cuenta cada una de las entradas que hace el usuario con el objetivo de ver si se obtiene el resultado esperado.
- **Resultado esperado:** breve descripción del resultado que se espera obtener con la prueba realizada.

- **Evaluación de la prueba:** acorde al resultado de la prueba realizada se emitirá una evaluación sobre la misma. Esta evaluación tendrá uno de los tres resultados que a continuación se describen:
    - **Bien:** cuando el resultado de la prueba es exactamente el esperado por el cliente.
    - **Regular:** cuando el resultado no es completamente el esperado por el cliente de la aplicación y muestra resultados erróneos o fuera de contexto.
    - **Mal:** cuando el resultado de la prueba realizada genera un error de codificación en la aplicación o muestra como resultado elementos no deseados o fuera de contexto.

Pruebas de Caja negra

Las pruebas de Caja negra son pruebas funcionales que se aplican sobre un sistema, empleando un determinado conjunto de datos de entrada y observando las salidas que se producen, para determinar si la función se está desempeñando correctamente por el sistema que se encuentra bajo prueba. Las herramientas básicas son observar la funcionalidad y contrastar con la especificación. (Pressman, 2010)

Existen diferentes técnicas de prueba de Caja Negra, descritas por Pressman, para validar la funcionalidad del sistema, sin entrar a analizar su ejecución interna.

1. Métodos de prueba basados en grafos
2. Análisis de valores límites
3. Tabla ortogonal
4. Partición equivalente (Pressman, 2010)

De las técnicas planteadas por Pressman, se decide utilizar la técnica de Partición equivalente, ya que es una de las más efectivas, esta permite examinar los valores válidos e inválidos de las entradas existentes en el software. (Pressman, 2010)

El diseño de casos de prueba para la partición equivalente, se basa en una evaluación de las clases de equivalencia para una condición de entrada:

1. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.
2. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

3. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. (Pressman, 2010)

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases. Los resultados de la prueba de caja negra se ven reflejados una vez llenados los diseños de casos de pruebas. (Pressman, 2010)

Diseño de pruebas

Un caso de prueba se diseña según las funcionalidades descritas en los casos de usos. Este diseño se elabora previamente al realizar las pruebas exploratorias o de interfaz como también pueden llamarse y se parte de la descripción de los casos de usos del sistema, como apoyo para las revisiones. Cada planilla de caso de prueba, recoge la especificación de un caso de uso, dividido en secciones y escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso en cuestión. Además, quedan plasmadas las revisiones realizadas al caso de prueba, así como un registro de todo aquello que no satisface y corresponde a la calidad del software. (Pressman, 2010)

A continuación, se muestran las pruebas de aceptación realizadas para las historias de usuario Autenticar usuario y Gestionar Usuario, el resto pueden ser consultadas en los anexos (Ver Anexo 3) de la investigación.

Tabla 14.Prueba de aceptación # 1

Caso de prueba de aceptación	
Código: HU1_P1	**Historia de usuario:** 1
Nombre: Autenticar Usuario.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Autenticar Usuario.	
Condiciones de ejecución: El usuario debe existir en la base de datos y tiene que haber sido aceptado.	
Pasos de ejecución: Se llenan los campos según su usuario y su correspondiente contraseña, luego se opta por la opción Acceder. El sistema valida los datos y en caso de ser correctos re-direcciona al usuario a la interfaz principal. Si uno de los datos no son correctos el sistema muestra un mensaje de error especificando el dato incorrecto.	
Resultados esperados: Se autentica de forma correcta al usuario.	
Evaluación de la Prueba: Bien.	

Fuente: Elaboración propia

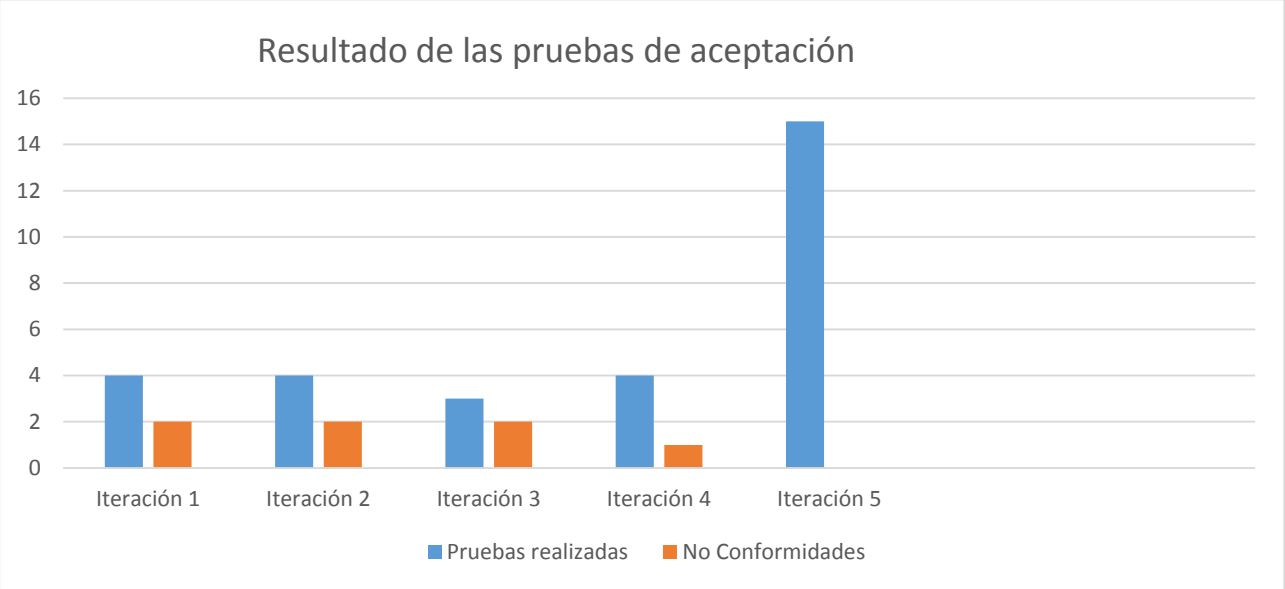
Tabla 15.Prueba de aceptación # 2

Caso de prueba de aceptación	
Código: HU2_P2	**Historia de usuario:**
Nombre: Gestionar Usuario.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Gestionar Usuario.	

Condiciones de ejecución: No puede existir en la base de datos un usuario igual.
Pasos de ejecución: Se llenan los campos requeridos y se opta por la opción Registrarse. El sistema valida los datos y en caso de ser correctos muestra un mensaje. Si uno de los datos no son correctos el sistema muestra un mensaje de error especificando el dato incorrecto. El usuario selecciona la opción Ver Perfil y se muestran sus datos en una interfaz, donde los puede modificar. Al usuario administrador se le muestran en una interfaz se muestran todos los usuarios existentes en el sistema y cuando los selecciona se muestra la opción Eliminar.
Resultados esperados: Se crea, modifica, muestra y elimina de forma correcta al usuario.
Evaluación de la Prueba: Bien.

Fuente: Elaboración propia

El gráfico que se muestra a continuación muestra los resultados obtenidos en el proceso de pruebas de aceptación.



Se realizaron un total de 15 pruebas a todas las funcionalidades en 5 iteraciones, corrigiendo todas las no conformidades existentes de forma satisfactoria.

3.2.3. Pruebas unitarias

Las pruebas unitarias son una de las piedras angulares de XP. Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Las pruebas deben ser definidas antes de realizar el código (López, 2016).

Que todo código liberado pase correctamente las pruebas unitarias es lo que habilita que funcione la propiedad colectiva del código. En este sentido, el sistema y el conjunto de pruebas debe ser guardado junto con el código, para que pueda ser utilizado por otros desarrolladores, en caso de tener que corregir, cambiar o recodificar parte del mismo (López, 2016).

Cuando se encuentra un error (“bug”), éste debe ser corregido inmediatamente, y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se generan nuevas pruebas para verificar que el error haya sido resuelto (López, 2016).

Método de prueba aplicado

El **método de caja blanca** se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que examinen que están correctas todas las condiciones y/o bucles para determinar si el estado real coincide con el esperado.

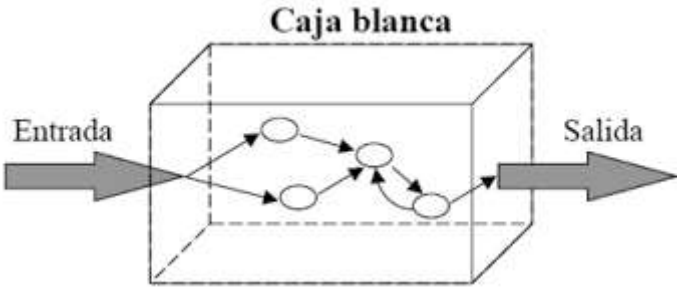


Ilustración 11.Ejemplo de representación del método de caja blanca

Una de las técnicas empleadas por este método para comprobar el funcionamiento correcto de las estructuras de datos es la técnica del camino básico.

Camino Básico

Esta técnica permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez. Para realizar la prueba es necesario realizar primeramente el análisis de la complejidad del algoritmo sobre el que se va a realizar la prueba, con el fin de calcular los valores de la complejidad ciclomática, como se muestra en la figura siguiente:

Ilustración 12.Código de la prueba unitaria al método BuscarServicio()

El grafo del flujo asociado al algoritmo Buscar Servicio representa los caminos de ejecución que se pueden seguir en el procedimiento. Es un grafo orientado en el que los vértices representan instrucciones o conjuntos de instrucciones que se ejecutan como una unidad. Cada arista representa la posibilidad de que una vez terminada la ejecución de un vértice se pase a ejecutar otro.

Los componentes de dicho grafo son:

1. Los nodos que son círculos representados en el grafo de flujo que representan una o más secuencias del procedimiento, donde cada nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al principio y al final del grafo.

2. Los nodos predicados que son los que contienen una condición y se caracterizan porque de ellos salen una o más aristas.

3. Las aristas son las flechas del grafo, iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando este no represente la sentencia de un procedimiento.

4. Las regiones son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo como una región más. Las regiones se enumeran: siendo la cantidad de

regiones equivalente a la cantidad de caminos independientes del conjunto básico del procedimiento.

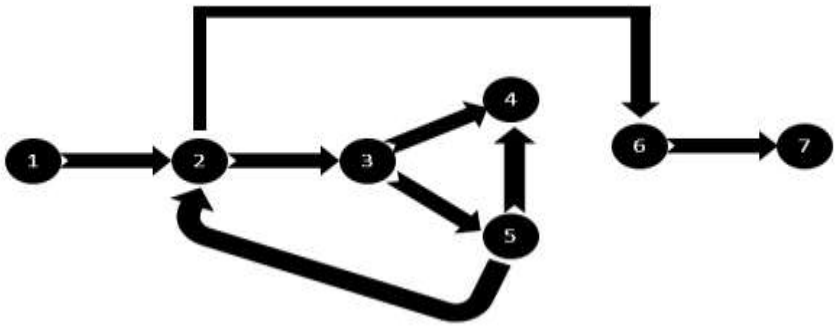


Ilustración 13.Grafo del flujo asociado al método BuscarServicio() de los nodos

La complejidad ciclomática es un parámetro de los grafos de flujo y representa la complejidad lógica de ejecución del programa. En el contexto de las pruebas, la cuantía de la complejidad ciclomática representa el número de caminos independientes que forman un conjunto de caminos básicos y por ello nos da el número mínimo de pruebas que garantiza la ejecución de cada instrucción al menos una vez.

La complejidad ciclomática se representa por V(G) y se puede calcular de varias formas:

- V(G) = Aristas – Nodos + 2 = 8 – 7 + 2 = 3
- V(G) = Nodos predicados + 1 = 2 + 1 = 3
- Como el grafo tiene 2 regiones V(G) = 3

Se realizaron pruebas al final de cada iteración contando al concluir con las iteraciones con un total de 18 pruebas.

El gráfico que se muestra a continuación muestra los resultados obtenidos en el proceso de pruebas unitarias.

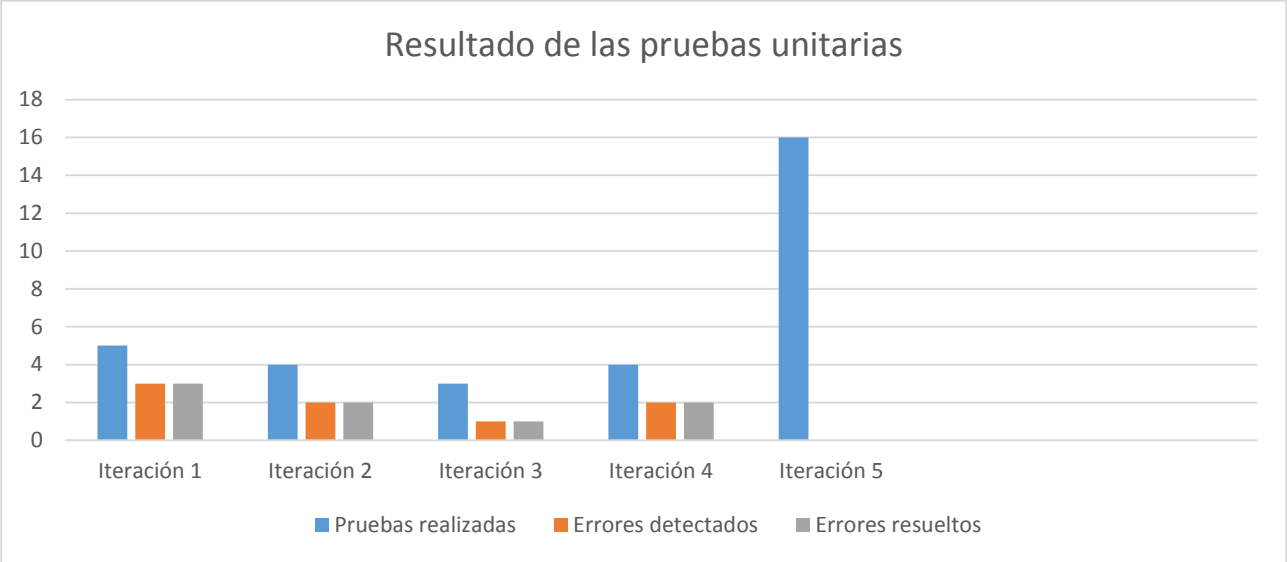


Ilustración 14.Resultado de las pruebas unitarias

Se realizaron un total de 16 pruebas a todo el código en 5 iteraciones definidas, corrigiendo todos los errores existentes.

3.3. Conclusiones parciales

Tras haber desarrollado una de las fases más importantes de la metodología XP se arribó a las siguientes conclusiones parciales:

- El uso de estándares de codificación permitió generar un cogido legible.
- El plan de entrega fue cumplido acorde al tiempo planificado para el desarrollo de las tareas por cada historia de usuario.
- A partir de la realización de las pruebas de Caja blanca, en específico la técnica de Camino básico se logró identificar los posibles caminos, garantizando que cada sentencia del mismo sea ejecutada al menos una vez.
- Al realizar las pruebas de Caja negra se logra describir los diferentes escenarios, donde queda clara la salida del sistema ante estos.
- Las pruebas de rendimiento permitieron conocer la capacidad de respuesta que presenta la aplicación ante las peticiones que realiza el usuario.
- El resultado de las pruebas dio a conocer que cada una de las no conformidades fue atendida correctamente, logrando un correcto funcionamiento de la aplicación ante cualquier situación.

Conclusiones generales

Una vez culminado el presente trabajo de diploma donde se comprende la interrelación clientes y proveedores para empresas estatales en Cuba, se puede concluir que se le dio cumplimiento al objetivo general de la investigación, resaltando que:

1. La elaboración del marco teórico conceptual de la investigación permitió sentar las bases de la misma. Se realizó un estudio de soluciones en el mundo que contienen el proceso de interrelación cliente y proveedores, aportando ideas a la aplicación que posteriormente fue desarrollada.
2. El análisis y diseño empleando la metodología XP, permitió identificar los requisitos de software y la descripción de las Historias de Usuario de la propuesta de solución para un mejor detalle de los procesos y funcionalidades la aplicación desarrollada.
3. Las pruebas permitieron detectar y corregir los errores no detectados durante la implementación, posibilitando cumplir con las especificaciones requeridas y la validación la aplicación implementada.

De esta manera se concluye, que, al desarrollar una aplicación Android para la interrelación entre clientes y proveedores, se inicia y facilita el trabajo a cada una de las empresas estatales de Cuba que participan en este proceso de clientes y proveedores, además de cumplir los principios de soberanía e independencia tecnológica.

Recomendaciones

A lo largo de la presente investigación fueron identificados elementos que no se contemplaron en el desarrollo de la solución al no ser parte del alcance inicial, pero se considera que son relevantes e incrementarían la utilidad del sistema y la calidad de uso del mismo por lo que se recomienda:

A los futuros desarrolladores que trabajen en la herramienta:

- Implementar la funcionalidad de geolocalización.
- Implementar la funcionalidad de mensajería instantánea.

A la XETID:

- Realizar una investigación para determinar nuevas funcionalidades que se puedan agregar a la aplicación.

Glosario de términos

Tarjetas C.R.C: Clases Responsabilidades Colaboradores.

UML: Lenguaje Unificado de Modelado o UML, por sus siglas en inglés, Unified Modeling Language.

XP: La programación extrema o eXtreme Programming (XP)

MVC: Modelo Vista Controlador.

XETID: Empresa de Tecnologías de la Información para la Defensa.

Referencias

Alegsa, Leandro. 2017. Alegsa.com. [En línea] 20 de Junio de 2017. http://www.alegsa.com.ar/Dic/aplicacion_movil.php.

Alonso Velazquez, José Luis. 2014. *Lenguaje de programación: Introducción C/C++ (IDE).* s.l. : Universidad de Guanajuato, 2014.

AndroidDev. 2019. ADT Plugins Release Notes. *Android Developers.* [En línea] 2019. http://developer.android.com/intl/es/tools/sdk/eclipse-adt.html.

Arias Calleja, Manuel. 2014. Carmen. Estándares de codificación. 2014.

Baltazar, Rodrigo. 2012. Prezi.com. [En línea] 2012. https://prezi.com/jkqcshkmkqxs/metodologia-de-desarrollo-del-software/.

Blanco, Carlos. 2008. *Patrones de Diseño. Ingeniería del Software I.* s.l. : Universidad de Cantabria, 2008.

Carrillo Pérez, Isaías, Pérez González, Rodrigo y Rodríguez Martín, Aureliano. 2008. *Metodologías de desarrollo de software.* 2008.

Concepto.de. 2019. [En línea] 2019. https://definicion.de/interrelacion/.

—. 2018. Concepto.de. [En línea] Noviembre de 2018. https://concepto.de/consumidor/.

debitoor. 2019. [En línea] 2019. https://debitoor.es/glosario/definicion-proveedor.

Digital Learning, SL. 2015. Academia Android. [En línea] 2015. http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/.

Gamma, Erich. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software. s.l.* s.l. : Pearson Education, 1995.

Gosling, James, Joy, Bill y Smith, Daniel. 2019. *The Java Language Specification.* s.l. : Oracle America, Inc., 2019.

Iruela, Juan. 2016. *Los gestores de bases de datos más usados.* 2016.

LaChopi. 2019. [En línea] 2019. http://www.lachopi.com.

Larman, Craig. 2004. *UML y Patrones.* 2004.

López, Yolanda Borja. 2016. *Metodología Ágil de Desarrollo de Software – XP.* s.l. : ESPE, MEVAST, 2016.

Luis Miguel Echeverry Tabón y Delgado Carmona, Luz Elena . 2007. *Caso Prático de la metodología ágil xp al desarrollo de software.* s.l. : Universiada Tecnológica de Pereira, 2007.

MySQL.com. 2019. [En línea] 2019. https://www.mysql.com/.

Norguera, Toni. 2018. www.xatakamovil.com. *www.xatakamovil.com.* [En línea] 20 de abril de 2018.

Ospina, Jaime. 2017. *Cooperación entre empresas. Innovación y gestión de proveedores.* 2017.

—. 2017. *Cooperación entre empresas. Innovación y gestión de proveedores.* 2017.

Patricio, Letelier y Penadés, Carmen. 2013. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* 2013.

PorLaLivre. 2019. [En línea] 2019. http://www.porlalivre.com.

Pressman, Roger S. 2010. *INGENIERÍA DEL SOFTWARE. UN ENFOQUE PRÁCTICO.* Nueva York : McGraw-Hill, 2010. 978-607-15-0314-5.

Rodríguez Sanchez, Tamara. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : Universidad de las Ciencias Informáticas, 2015.

SEDICI. 2012. *Aprendizaje combinado, aprendizaje electrónico centrado en el alumno y nuevas tecnologías.* Repositorio Institucional de la UNLP, Universidad Nacional de la Plata. Buenos Aires : s.n., 2012.

Sierra, Antonio. 2013. *UML (Unified Medling Language) Lenguaje Unificado de Modelado.* 2013.

Sommerville, Ian. 2005. *Ingeniería del Software. Septima edición.* Madrid : Pearson Educación, S.A., 2005. 84-7829-074-5.

Ucha, Florencia. 2011. Definición ABC. [En línea] 2011. https://www.definicionabc.com/economia/gestion-empresarial.php.

UIT. 2019. Actualidades de la Uit. *Actualidades de la Uit.* [En línea] 2019. https://www.itu.int/net/itunews/issues/2009/06/04-es.aspx.

Visual Paradigm, International Ltd. 2017. *Visual Paradigm.* 2017.

Anexos

Anexo #1 Historias de usuario:

Tabla 16.HU-3 Gestionar Servicio

Número: **HU-3**	**Nombre:** Gestionar Servicio
Iteración: **1**	
Prioridad: **Alta**	**Complejidad**: Alta
Descripción: **Luego de que el usuario escoja la opción de Añadir servicio, el sistema mostrará la interfaz correspondiente, en la cual el usuario tendrá que introducir los datos correspondientes (nombre, descripción, teléfono, correo, precio, moneda).** **Una vez añadido en el sistema, el servicio se mostrará en una interfaz. Para modificarlo, el usuario selecciona el servicio que desea modificar y se mostrará en una interfaz todos los datos de ese servicio (nombre, descripción, teléfono, correo, precio, moneda), donde los podrá cambiar.** **Para eliminarlo, el usuario selecciona el servicio que desea eliminar y se mostrará la opción de Eliminar.**	
Observaciones: **Si se dejan campos en blanco el sistema muestra un mensaje de aviso.** **Si se rellenan campos con datos erróneos el sistema muestra un mensaje de aviso.** **Si todos los datos son correctos el sistema muestra un mensaje informativo.**	

Fuente: Elaboración propia

Tabla 17. HU-4 Gestionar Producto

Número: **HU-4**	**Nombre:** Gestionar Producto
Iteración: **2**	
Prioridad: **Alta**	**Complejidad**: Alta
Descripción: **Luego de que el usuario escoja la opción de Añadir producto, el sistema mostrará la interfaz correspondiente, en la cual el usuario tendrá que introducir los datos correspondientes (nombre, descripción, teléfono, correo, precio, moneda).** **Una vez añadido en el sistema, el producto se mostrará en una interfaz. Para modificarlo, el usuario selecciona el producto que desea modificar y se mostrará en una interfaz todos los datos de ese producto (nombre, descripción, teléfono, correo, precio, moneda), donde los podrá cambiar.** **Para eliminarlo, el usuario selecciona el producto que desea eliminar y se mostrará la opción de Eliminar.**	
Observaciones: Si se dejan campos en blanco el sistema muestra un mensaje de aviso. Si se rellenan campos con datos erróneos el sistema muestra un mensaje de aviso. Si todos los datos son correctos el sistema muestra un mensaje informativo.	

Fuente: Elaboración propia

Tabla 18.HU-5 Enviar Notificación

Número: **HU-5**	**Nombre:** Enviar Notificación
Iteración: **3**	
Prioridad: **media**	**Complejidad**: media
Descripción: Luego de que el usuario escoja la opción de solicitar servicio el sistema enviará una notificación al usuario que brinda ese servicio.	
Observaciones:	

Fuente: Elaboración propia

Tabla 19.HU-6 Recibir Notificación

Número: **HU-6**	**Nombre:** Recibir Notificaciones
Iteración: **3**	
Prioridad: **media**	**Complejidad**: media
Descripción: **El sistema mostrará todas las notificaciones del usuario.**	
Observaciones:	

Fuente: Elaboración propia

Tabla 20.HU-7 Buscar Servicio

Número: **HU-7**	**Nombre:** Buscar Servicio
Iteración: **3**	
Prioridad: **Media**	**Complejidad**: Media
Descripción: **El usuario podrá buscar un servicio por su nombre, mostrándose el resultado en una interfaz del sistema.**	
Observaciones:	

Fuente: Elaboración propia

Anexo #2 Tareas de Ingeniería:

Tabla 21.Tarea de ingeniería #3 desglosada

Número de la tarea: **3**	**No. De HU:** 2
Nombre de la tarea: **Eliminar Usuario**	

Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**22-2-2019**	**Fecha fin:**28-2-2019
Descripción: El usuario administrador selecciona el usuario que desea eliminar y el sistema muestra la opción de Eliminar	

Fuente: Elaboración propia

Tabla 22.Tarea de ingeniería #4 desglosada

Número de la tarea: **4**	**No. De HU:** 2
Nombre de la tarea: **Listar Usuario**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**1-3-2019**	**Fecha fin:**6-3-2019
Descripción: **El sistema mostrará una interfaz en la cual el usuario podrá ver todos los usuarios del sistema.**	

Fuente: Elaboración propia

Tabla 23.Tarea de ingeniería #5 desglosada

Número de la tarea: **5**	**No. De HU:** 3
Nombre de la tarea: **Crear Servicio**	

Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**7-3-2019**	**Fecha fin:**21-3-2019
Descripción: **El sistema mostrará una interfaz en la cual el usuario podrá ingresar los datos para crear el servicio (nombre, descripción, teléfono, correo, precio, moneda).**	

Fuente: Elaboración propia

Tabla 24.Tarea de ingeniería #6 desglosada

Número de la tarea: **6**	**No. De HU:** 3
Nombre de la tarea: **Modificar Servicio**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**22-3-2019**	**Fecha fin:**28-3-2019
Descripción: **El usuario selecciona el servicio que desea modificar y el sistema mostrará una interfaz en la cual el usuario podrá modificar los datos de su servicio (nombre, descripción, teléfono, correo, precio, moneda).**	

Fuente: Elaboración propia

Tabla 25.Tarea de ingeniería #7 desglosada

Número de la tarea: **7**	**No. De HU:** 3
Nombre de la tarea: **Eliminar Servicio**	
Puntos de estimación: **1**	

Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**29-3-2019**	**Fecha fin:**4-4-2019
Descripción: **El usuario proveedor selecciona el servicio que desea eliminar y el sistema mostrará la opción Eliminar.**	

Fuente: Elaboración propia

Tabla 26.Tarea de ingeniería #8 desglosada

Número de la tarea: **8**	**No. De HU:** 3
Nombre de la tarea: **Listar Servicio**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**5-4-2019**	**Fecha fin:**11-4-2019
Descripción: **El sistema mostrará una interfaz en la cual el usuario proveedor podrá ver todos sus servicios en el sistema. Para el usuario cliente se mostrarán todos los servicios existentes en el sistema.**	

Fuente: Elaboración propia

Tabla 27.Tarea de ingeniería #9 desglosada

Número de la tarea: **9**	**No. De HU:** 7
Nombre de la tarea: **Buscar Servicio**	
Puntos de estimación: **1**	

Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**12-4-2019**	**Fecha fin:**16-4-2019
Descripción: **El usuario podrá buscar el servicio que desee según su nombre, mostrándose los resultados en una interfaz del sistema.**	

Fuente: Elaboración propia

Tabla 28.Tarea de ingeniería #10 desglosada

Número de la tarea: **10**	**No. De HU:** 6
Nombre de la tarea: **Enviar Notificación**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**17-4-2019**	**Fecha fin:**21-4-2019
Descripción: **El sistema enviará una notificación una vez que el usuario cliente solicite un servicio o producto.**	

Fuente: Elaboración propia

Tabla 29.Tarea de ingeniería #11 desglosada

Número de la tarea: **11**	**No. De HU:** 5
Nombre de la tarea: **Recibir Notificaciones**	
Puntos de estimación: **1**	

Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**22-2-2019**	**Fecha fin:**26-4-2019
Descripción: **El usuario recibirá sus notificaciones y el sistema las mostrará en una interfaz.**	

Fuente: Elaboración propia

Tabla 30.Tarea de ingeniería #12 desglosada

Número de la tarea: **12**	**No. De HU:** 1
Nombre de la tarea: **Autenticar Usuario**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**27-2-2019**	**Fecha fin:**2-5-2019
Descripción: **Se mostrará una interfaz en la que el usuario podrá ingresar su usuario y su contraseña para acceder al sistema.**	

Fuente: Elaboración propia

Tabla 31.Tarea de ingeniería #13 desglosada

Número de la tarea: **13**	**No. De HU:** 4
Nombre de la tarea: **Crear Producto**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**3-5-2019**	**Fecha fin:**6-5-2019

Descripción: **En la página principal del sistema, aparecerá un botón circular azul, el usuario proveedor lo selecciona y se mostrarán dos opciones, selecciona la opción de producto y aparecerá una interfaz donde podrá ingresar todos los datos de su servicio.**

Fuente: Elaboración propia

Tabla 32.Tarea de ingeniería #14 desglosada

Número de la tarea: **14**	**No. De HU:** 4
Nombre de la tarea: **Modificar Producto**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**7-5-2019**	**Fecha fin:**11-5-2019
Descripción: **El usuario proveedor seleccionará el producto que desee modificar y el sistema mostrará una interfaz con todos los datos de ese producto para que los pueda modificar.**	

Fuente: Elaboración propia

Tabla 33.Tarea de ingeniería #15 desglosada

Número de la tarea: **15**	**No. De HU:** 4
Nombre de la tarea: **Listar Producto**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**12-5-2019**	**Fecha fin:**14-5-2019
Descripción: **El sistema mostrará una interfaz en la cual el usuario cliente podrá ver todos los productos en el sistema. Para el usuario proveedor solo se mostrarán sus productos.**	

Fuente: Elaboración propia

Tabla 34.Tarea de ingeniería #16 desglosada

Número de la tarea: **16**	**No. De HU:** 4
Nombre de la tarea: **Eliminar Producto**	
Puntos de estimación: **1**	
Tipo de tarea: **desarrollo**	**Responsable**: Julio César Vega Fuentes
Fecha inicio:**15-5-2019**	**Fecha fin:**17-5-2019
Descripción: **El usuario proveedor selecciona el producto que desee eliminar y aparecerá la opción de eliminar.**	

Fuente: Elaboración propia

Anexo #3 Pruebas de aceptación:

Tabla 35.Prueba de aceptación # 3

Caso de prueba de aceptación	
Código: HU3_P3	**Historia de usuario:** 3
Nombre: Gestionar Servicio.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Gestionar Servicio.	
Condiciones de ejecución: El usuario tiene que estar autenticado en el sistema.	
Pasos de ejecución: Se llenan los campos requeridos y se opta por la opción Añadir Servicio. El sistema valida los datos y en caso de ser correctos re-direcciona a la interfaz principal, mostrando el servicio adicionado y un mensaje. Para modificarlo el usuario proveedor selecciona el servicio que desea y se muestra una interfaz con todos los datos de ese servicio, donde los puede modificar. Para eliminarlo el usuario proveedor selecciona el servicio que desea y aparece la opción de Eliminar.	
Resultados esperados: Se añade, modifica, muestra y elimina de forma correcta el servicio.	
Evaluación de la Prueba: Bien.	

Fuente: Elaboración propia

Tabla 36.Prueba de aceptación # 4

Caso de prueba de aceptación

Código: HU5_P4	**Historia de usuario:** 5
Nombre: Enviar notificación.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Enviar Notificación.	
Condiciones de ejecución: Tiene que existir el servicio en la base de datos.	
Pasos de ejecución: El usuario selecciona la opción solicitar servicio.	
Resultados esperados: Se envía de forma correcta la notificación.	
Evaluación de la Prueba: Bien.	

Fuente: Elaboración propia

Tabla 37.Prueba de aceptación # 5

Caso de prueba de aceptación	
Código: HU6_P5	**Historia de usuario:** 6
Nombre: Recibir Notificaciones.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Recibir Notificaciones.	
Condiciones de ejecución: La notificación tiene que existir en la base de datos.	
Pasos de ejecución: El usuario escoge la opción de Notificaciones y se muestran todas las notificaciones existentes en la base de datos para el usuario correspondiente.	
Resultados esperados: Se reciben de forma correcta las notificaciones.	
Evaluación de la Prueba: Bien.	

Fuente: Elaboración propia

Tabla 38.Prueba de aceptación # 11

Caso de prueba de aceptación	
Código: HU4_P6	**Historia de usuario:** 4
Nombre: Gestionar Producto.	
Descripción: Se comprueba el funcionamiento de la historia de usuario Gestionar Producto.	
Condiciones de ejecución:	
Pasos de ejecución: Se llenan los campos requeridos y se opta por la opción Añadir Producto. El sistema valida los datos y en caso de ser correctos re-direcciona a la interfaz principal, mostrando el producto adicionado y un mensaje. Para modificarlo el usuario proveedor selecciona el producto que desea y se muestra una interfaz con todos los datos de ese producto, donde los puede modificar. Para eliminarlo el usuario proveedor selecciona el producto que desea y aparece la opción de Eliminar.	
Resultados esperados: Se añade, muestra, modifica y elimina de forma correcta el producto.	
Evaluación de la Prueba: Bien.	

Fuente: Elaboración propia

Tabla 39.Prueba de aceptación # 12

Caso de prueba de aceptación	
Código: HU12_P12	**Historia de usuario:** 12

Nombre: Buscar Servicio.
Descripción: Se comprueba el funcionamiento de la historia de usuario Buscar Servicio.
Condiciones de ejecución: El servicio tiene que existir en la base de datos.
Pasos de ejecución: El usuario escribe el nombre del servicio que desea buscar y luego selecciona el botón de Buscar. Se muestran en una interfaz los resultados de la búsqueda.
Resultados esperados: Se busca de forma correcta al servicio.
Evaluación de la Prueba: Bien.

Fuente: Elaboración propia