Facultad 2

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Título

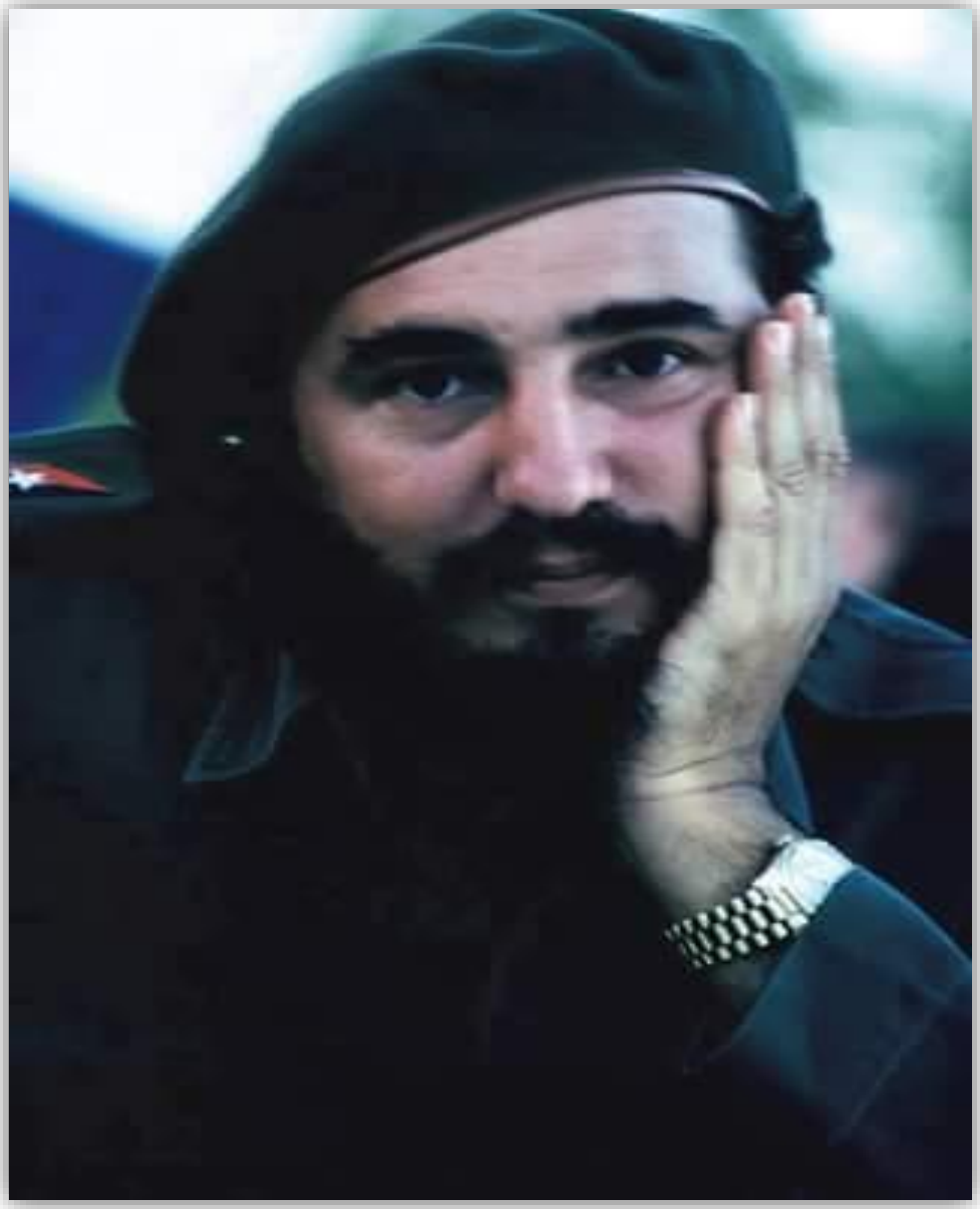
"Herramienta de gestión de oferta para el centro CIGED"

Autora: Suri Nancy Mustelier Villalón

Tutores
M. Sc. Aurelio Antelo Collado
Ing. Aray Villar Machado

La Habana, junio 2018

"Año 60 de la Revolución"



"Y yo creo, he creído siempre, y pienso que lógicamente ustedes también lo creen, en que la educación es el arma más poderosa que tiene el hombre para crear una ética, para crear una conciencia, para crear un sentido del deber, un sentido de la organización, de la disciplina, de la responsabilidad."

Fidel Castro Ruz

Dedicatoria

A mi abuela Loló, mi abuelo Rey y mi tío Rafael: porque a pesar de no recordarlos, siempre estuvieron presentes en mi vida. Los amo.

A mi mamá y mi papá: que son las personas más lindas e importantes que me ha obsequiado Dios, porque si pudiera elegir, los elegiría una y mil veces como mis padres, sin dudar; por ser unas personas decentes y educadas; por confiar siempre en mí, porque son mi vida, mi inspiración, mi motor impulsor; por educarme con tanto amor y sacrificio, porque deseo que la vida me permita regresarles un poquito de lo mucho que me han ofrecido, porque por ellos es que estoy haciendo realidad su sueño, mi sueño; con ellos todo, sin ellos nada; hacia ustedes, mi amor es innato.

A mi tía Nancy y mi tío Jorge: que son mis segundos padres, por cuidarme con tanto amor desde que nací, por su dedicación y comprensión, por darme una buena educación, porque estoy orgullosa de que ustedes formen parte de mi vida.

A mi hermano, mis primas y mis primos: que gracias a Dios los tengo y es un privilegio que estén siempre conmigo, por su apoyo incondicional y por todo el cariño que me profesan.

A Irma, Isel y Zayda: por ser tan preocupadas conmigo, por mirarme con los ojos del corazón.

A Marcos, Nora, mi tía Nego y Albertina: porque a pesar de que no estén físicamente, sé que hoy deben sentirse muy orgullosos de mí.

A la memoria de mis seres queridos, especialmente a mis bisabuelos Nieves y Valero.

A Fidel: porque es difícil no amarte y respetarte Comandante.

A mis futuros hijos: porque si estoy haciendo realidad este sueño, es por tener una mejor vida y ellos están incluidos en esta.

Los amo, admiro y respeto a todos.

Agradecimientos

Ante todo a Dios y todas las deidades: por concederme la familia que tengo, por cuidarnos y protegernos.

A mi abuela Loló: por cuidarme en mis primeros años con tanto amor y ternura, mi vida te la debo.

A mi mami: por ser mis ojos, mi ídolo, mi guía, mi mejor amiga, mi compañera, mi mejor consejera en cualquier tipo de circunstancia; por mostrarme siempre el camino correcto, por donde debo transitar; por compartir las penas y las alegrías, por reír y sufrir conmigo, por confiar en mí; por estar siempre pendiente de todos, olvidándose de ella; por ser una súper mujer; por ser la mejor madre que Dios me pudo obsequiar, porque es una bendición tenerla como mi mamá, es la mujer de mi vida; eres mi gloria; iluminas mi vida hoy, mañana y siempre.

A mi papi: por ser mi mejor amigo y apoyarme en todas las decisiones que he tomado, por no dudar de mí, por sus buenos consejos, por ser mi ejemplo a seguir, por ser tan bella persona, por ser el mejor padre que Dios me ha podido regalar, porque es una bendición tenerlo como mi papá, es el hombre de mi vida; eres mi dicha por siempre, mi amor hacia tí es infinito e incondicional.

A mi hermano: por ser parte de mi vida, el mejor hermano, una de las personas más importantes que poseo desde que tengo uso de razón.

A mi tía Nancy: por ser mi madrina, mi amiga y mi segunda madre, por tenerme como su hija demostrándolo con su cariño, la persona que siempre se ha preocupado por mí en todos los sentidos, por darme fuerza para conseguir mis metas.

A mi tío Jorge: que es mi segundo papá, mi tío favorito, por darme tan buenas lecciones de vida, por quererme gratuitamente, por ser más de lo que he pedido.

A mis primas Anne y Kachy: por ser mis hermanas, por no dudar de mí, por entenderme, por tender su brazo amigo en toda ocasión.

A mis primitos Annerys, Dannyer y Dennyer: porque a pesar de que no comprendan la magnitud de este evento, sin saberlo, llenan mi vida de alegría.

A mi padrino Marcos: por sacarme de apuros y ponerme a reflexionar en más de una ocasión, por ser mi luz en los momentos más difíciles, por sus certeros consejos, por su apoyo incondicional siempre; eres el hombre más perfecto que he conocido.

A Nora e Irma: por ser las abuelas más lindas que he tenido, por complacerme en las cositas que más me gustan, por quererme sin yo pedirlo.

A Olga, Albertina, Odalis, Virgen y Marcelina: por apoyar a mi familia en todo momento.

A mis tías Isel, Zayda, Papucha, Dayamí y Nena: porque de una forma u otra cada una ha sabido dar su granito de arena para lograr este sueño, gracias por todo el cariño brindado.

A Steven, Yolanda y sus hijos: porque a pesar de no tener lazos de sanguíneos, son parte de mi familia, gracias por su generosidad.

A mi gente linda de Palma Soriano: porque han cuidado de mis padres y he podido dormir un poquito más tranquila en todos estos años.

A Zamira: por ser como una hermana mayor o una madre en esta universidad, por ser una persona muy especial, una gran amiga.

A mis compañeros, profesores, tutores, tribunal y oponente: por la disposición brindada, por sus críticas, por contribuir en mi formación como profesional en este período estudiantil.

A Fidel: por ser un ejemplo fiel de una persona altruista.

A todas las personas que me quieren: porque siempre me dieron aliento para seguir adelante y cumplir mi objetivo.

Gracias a todos.

Declaración de autoría

Se declara que Suri Nancy Mustelier Villalón es la única autora del presente Trabajo de Diploma que tiene por título *"Herramienta de gestión de oferta para el centro CIGED",* se concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

Autora:

_____

Suri Nancy Mustelier Villalón

Tutores:

_____ _____

M. Sc. Aurelio Antelo Collado Ing. Aray Villar Machado

Síntesis de los tutores

- M. Sc. Aurelio Antelo Collado: graduado como Ingeniero Industrial y Master en Matemática Aplicada. Desempeña el cargo de Director del Centro de Informatización de la Gestión Documental. Correo electrónico: aantelo@uci.cu.

- Ing. Aray Villar Machado: graduada como Ingeniera en Ciencias Informáticas. Especialista General en la Dirección de Transferencia de Tecnología. Correo electrónico: aray@uci.cu.

Resumen

El contexto actual muestra la importancia que han alcanzado los productos intangibles, incluso por encima de lo que han experimentado los productos tangibles, pero se desconoce la forma de cómo medirlos o valorarlos. Este trabajo de diploma se centra en desarrollar una aplicación informática que permita fijar el precio de venta de los productos informáticos del Centro de Informatización de la Gestión Documental, para después definir una estrategia de venta para estos productos. El proceso de fijación del precio de venta de los productos intangibles que se comercializan en el centro, se realiza empleando un Excel que gestiona la ficha de cálculo de cada producto, por lo que el precio de venta depende en gran medida de los costos que se utilizan para realizar el *software* y no se puede generar reportes con la información relacionada con la oferta de un producto. Para llevar a cabo esta investigación y dar cumplimiento al objetivo trazado, se realizó un estudio del arte referente al modelo de negocio, la fijación de precio y la estrategia de venta de los productos intangibles a nivel mundial, con el fin de recopilar información relacionada con este tema. Con la ejecución de este trabajo de diploma se logra como resultado un *software* que resuelve la situación problemática anteriormente planteada y constituye a su vez, un aporte tecnológico al centro.

Palabras claves

Cliente, estrategia de venta, fijación de precio, producto informático, producto intangible.

Abstract

The current context shows the importance that intangible products have reached, even over what tangible products have experienced, but the way to measure or value them is unknown. This diploma thesis focuses on developing a computer application that allows you to set the selling price of computer products of the Documentation Management Computerization Center, and then define a sales strategy for these products. The process of setting the sale price of intangible products that are marketed in the center, is done using an Excel document that manages the calculation sheet of each product, so the sale price depends largely on the costs that are used to make the software and cannot generate reports with information related to the offer of a product. In order to carry out this research and fulfil the outlined objective, a study of the art regarding the business model, the pricing and the sales strategy of intangible products worldwide was made, in order to collect information related to this matter. With the execution of this work, a software that resolves the previously mentioned problematic situation is obtained and, at the same time, constitutes a technological contribution to the center.

Keywords

Client, computer product, intangible product, price fixing, sales strategy.

Índice de contenido

Índice de tablas

Índice de figuras

Introducción

Los activos intangibles han venido ganando importancia en los procesos de gestión, dado el impacto que generan en la creación de valor a largo plazo. La explicación que se puede dar a este comportamiento radica, entre otras cosas, al cambio de era, pues se pasa de la era industrial a la era de la información, con lo cual se cuestiona la importancia de los activos tangibles como elementos prioritarios en la generación de ventaja competitiva en los negocios y, consecuentemente, en la creación de valor en las organizaciones (González 2015).

En la actualidad, los activos intangibles son los responsables en un alto porcentaje de la creación de valor en las organizaciones. No obstante, el gran dilema que surge es cómo medirlos, pues se ha hecho famosa la frase que plantea: “lo que no se puede medir, no se puede gerenciar”, de tal manera que puedan ser identificados plenamente con la creación de valor, tal como se puede hacer con los activos tangibles, ya que su impacto en la generación de ventaja competitiva y en la creación de valor puede ser constatada de manera directa en los estados financieros de las compañías (González 2015).

Ninguna entidad está exenta de este fenómeno mundial, un ejemplo de ello es la Universidad de las Ciencias Informáticas (UCI), que posee 14 centros productivos que tienen que hacer el proceso de fijación de precios y definir la estrategia de venta de los productos y servicios que comercializan.

La UCI surge en el año 2002 con la visión de futuro del Comandante en Jefe Fidel Castro Ruz, que como estratega fundacional, recomendó que la universidad fuese concebida como un centro de nuevo tipo, de alcance nacional, atípico y con tareas concretas en el proyecto de informatización de la sociedad cubana, con énfasis en la producción de *software* (Informáticas 2017c).

La UCI tiene como misión formar profesionales comprometidos con su patria y altamente calificados en la rama de la informática. Producir aplicaciones y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Servir de soporte a la industria cubana de la informática (Informáticas 2017d).

Los centros desarrollan proyectos, productos y servicios, garantizando la integración y reutilización de aplicaciones y componentes, de acuerdo con las proyecciones y políticas del país; a la vez que propician la

integración de los procesos de formación, producción e investigación, con elevado nivel de compromiso, ética y profesionalidad (Informáticas 2017b).

El Centro de Informatización de la Gestión Documental (CIGED) adscrito a la Facultad 2, desarrolla sistemas y servicios informáticos integrales de alta calidad y competitividad en la informatización o mejora de los procesos de gestión documental. Su labor es integrar los procesos docentes, productivos e investigativos, garantizando la formación y superación de los estudiantes y especialistas de las diferentes áreas, altamente comprometidos con la Revolución, los clientes y la organización. Entre sus funciones se encuentran (Informáticas 2017a):

- Desarrollar soluciones informáticas a la medida para la informatización de la UCI, la sociedad cubana y para la exportación (Informáticas 2017a).
- Prestar servicios informáticos asociados a las soluciones y productos entregados a los clientes (Informáticas 2017a).
- Ofrecer soporte técnico y seguimiento a los clientes en cuanto a productos y soluciones desarrolladas (Informáticas 2017a).

CIGED cuenta con cuatro productos informáticos: ABCD 3.0 que permite la automatización de bibliotecas y centros de documentación, REPXOS 3.0 es un sistema de implantación de repositorios digitales, ARKHEIA 3.0 es un sistema de gestión de archivos históricos y EXCRIBA 3.1 es un sistema de gestión de documentos administrativos. Estos productos informáticos se encuentran desplegados en entidades de todo el país y algunos de ellos incluso, forman parte de comunidades reconocidas a nivel internacional, como ABCD 3.0 y REPXOS 3.0.

El modelo de negocio es el componente fundamental por lo cual un negocio trata de crear valor, ingresos y beneficios, ofrece a sus clientes sus mejores productos y/o servicios, satisface así las necesidades de los consumidores para que la empresa continúe siendo de la preferencia de estos. Un modelo de negocio no se obtiene, sino que se funda en base a las peculiaridades del producto y/o servicio que entrega al mercado. Cuando se decide generar un modelo de negocio hay que tener un conocimiento previo del negocio, por tales motivos fueron estudiados los productos del centro.

El modelo de negocio actual de los productos informáticos define en su alcance que puedan ser insertados en cualquier mercado donde la gestión documental sea fundamental. Las soluciones se comercializan en forma de paquete, cada uno va a contener el producto base, los módulos y los servicios asociados a cada uno de estos productos, los mismos son: levantamiento de información, personalización del sistema, despliegue y configuración, capacitación, acompañamiento y migración.

Hoy el cliente puede seleccionar lo que sea más idóneo para su entidad, por ejemplo, puede escoger el producto base, los módulos asociados al mismo y los servicios que considere necesarios. No es obligatorio que los emplee todos a menos que lo considere pertinente. Se tiene en cuenta la diversidad de las instituciones en cuanto a estructura, funcionamiento, métodos de trabajo, objetivos, entre otros aspectos, fue necesario definirlo así, realizando soluciones a la medida para cada entidad con el fin de lograr la satisfacción del cliente, y a la vez ganar prestigio como empresa productora de *software*.

Actualmente en el país no se ha estipulado una resolución que indique como debería ser o estar estructurada la gestión documental en las empresas, por esto en cada una de ellas se opera de manera diferente, esto incide directamente en el momento de definir el precio de venta que poseerá cada producto y/o servicio informático brindado. Como será una solución a la medida para cada cliente, no se puede establecer un precio fijo, debido a que las variables tiempo y recursos humanos a emplear, por ejemplo, van a ser desiguales para lograr el objetivo de cada entidad.

Por estas razones es muy difícil darle una rápida respuesta al cliente cuando pregunta cuánto le costará obtener el producto y/o servicio informático, hasta tanto esté confeccionada la oferta por parte de la Dirección de Transferencia de Tecnología (DTT), que es donde se tienen en cuenta las variables que inciden en el precio de venta, como pueden ser el producto, la cantidad de módulos y los servicios, así como la complejidad de ellos, para lograr una personalización lo más acertada posible.

Como resultado de la situación problemática se tiene el siguiente **problema de la investigación:** ¿cómo establecer el precio de venta para cada producto informático perteneciente al Centro de Informatización de la Gestión Documental?

Tomando como **objeto de estudio:** el proceso de gestión de ventas de productos intangibles.

El **objetivo general** de este trabajo de diploma es: desarrollar una aplicación informática que facilite la fijación de precio de los productos informáticos del Centro de Informatización de la Gestión Documental, para lograr establecer la estrategia de venta.

Su **campo de acción** lo constituye: la fijación de precio de los productos informáticos.

Con el propósito de dar cumplimiento al objetivo planteado se hace necesario realizar las siguientes **tareas de la investigación:**

1. Caracterización de las metodologías y estrategias que se utilizan a nivel mundial para fijar el precio de venta de los productos intangibles, para aplicarlas a los productos informáticos del Centro de Informatización de la Gestión Documental.
2. Caracterización de las estrategias de ventas de los productos intangibles empleadas a nivel mundial, para su aplicación a los productos informáticos del Centro de Informatización de la Gestión Documental.
3. Diseño de una solución que contribuya a la fijación del precio y definición de una estrategia de venta para lograr mayor comercialización de los productos informáticos del Centro de Informatización de la Gestión Documental.
4. Implementación de la solución propuesta.
5. Validación de la solución propuesta mediante el uso de métodos de pruebas para demostrar el correcto funcionamiento de la misma.

Se tienen los siguientes **métodos teóricos** para desarrollar las tareas de investigación:

- **Análisis histórico-lógico:** se utiliza en el estudio crítico de trabajos anteriores concerniente al proceso de gestión del modelo de negocio, la fijación de precio y la estrategia de venta de los productos intangibles, para utilizar estos como punto de referencia y compararlos con los resultados alcanzados.

- **Analítico-sintético:** se utiliza para descomponer el problema de investigación en elementos separados y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

- **Inductivo-deductivo:** se utiliza para la identificación de la situación problemática y de las soluciones en el proceso de gestión del modelo de negocio.

- **Modelación:** para modelar teóricamente el modelo de negocio propuesto.

Se tiene como **método empírico** para desarrollar las tareas de investigación el siguiente:

- **Entrevista:** para obtener la información necesaria, verificar la situación problemática y el problema de investigación sobre el proceso de fijación de precio y la estrategia de venta de los productos informáticos del centro. La misma se encuentra en el anexo 1 de este trabajo de diploma.

El presente trabajo de diploma está estructurado de la manera siguiente: introducción, tres capítulos, conclusión general, recomendaciones, referencias bibliográficas, bibliografía y anexos. La información comprendida en los capítulos se organiza del modo siguiente:

Capítulo 1 "Fundamentación teórica de la investigación": incluye la fundamentación teórica referente al modelo de negocio, la fijación de precio y la estrategia de venta de los productos intangibles a nivel mundial. Se fundamenta el uso de las herramientas, metodología de desarrollo de *software* y lenguaje de programación que se utilizan, lo que constituye la base teórica de la investigación.

Capítulo 2 "Descripción de la solución propuesta": se genera un modelo de dominio donde se analizan las entidades y conceptos existentes en el contexto, donde se realiza la solución propuesta, se especifican las relaciones presentes entre cada uno de estos. Se explica los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se muestra el diagrama de casos de uso del sistema, se representan los actores y los casos de uso del sistema. Se visualiza el modelo de clases, así como el diagrama de clases y el diagrama de entidad relación. Se selecciona el estilo arquitectónico y los patrones de diseño a emplear en la solución propuesta.

Capítulo 3 "Implementación y validación de la solución propuesta": se explica a partir de los resultados del diseño, la implementación de la aplicación. Se define los estándares de codificación. Como parte de la implementación se muestra el diagrama de despliegue, en el que se visualiza la situación física del sistema. Se especifican las pruebas de *software* y los resultados obtenidos de las mismas. Se realiza la validación de la solución propuesta.

Capítulo 1: Fundamentación teórica de la investigación

Este capítulo incluye la fundamentación teórica referente al modelo de negocio, la fijación de precio y la estrategia de venta de los productos intangibles a nivel mundial. Se fundamenta el uso de las herramientas, metodología de desarrollo de *software* y lenguaje de programación que se utilizan, lo que constituye la base teórica de la investigación.

1.1 Introducción al capítulo

Los productos intangibles son aquellos que consideramos servicios y su valor es difícil de apreciar, ya que no se recibe un producto físico (García 2017).

La Norma Internacional de Contabilidad nº 38 (NIC 38)[1] tiene como objetivo prescribir el tratamiento contable de los activos intangibles que no estén contemplados específicamente en otra norma. Esta norma requiere que las entidades reconozcan un activo intangible si, y solo si, se cumplen ciertos criterios. La NIC 38 también especifica cómo determinar el importe en libros de los activos intangibles, y exige la revelación de información específica sobre estos activos. Un activo intangible es un activo identificable, de carácter no monetario y sin apariencia física (IASC 2009).

Son activos intangibles tradicionales las patentes, marcas, la propiedad intelectual, la cartera de clientes, y otros más cuya enumeración se perdía en el concepto contable que tradicionalmente ha servido como saco sin fondo, el fondo de comercio o goodwill. Para este limitado grupo de activos se disponía hasta nuestros días de una métrica y de unas prácticas contables que parecían más que suficientes para la gestión del valor de una empresa (Ruz 2012).

Son los activos intangibles, hoy en la primera línea de la actualidad, el denominado capital humano, el diseño estructural de la firma, su reputación, los derechos de imagen y otros conceptos de objetos de intensos análisis y debates sobre su delimitación y correcto tratamiento contable. A los mismos se añade la

[1] La NIC 38 ofrece los siguientes ejemplos de activos intangibles: programas informáticos, patentes, derechos de autor, películas, listas de clientes, licencias de pesca, cuotas de importación, franquicias, relaciones comerciales con clientes o proveedores, lealtad de los clientes, cuotas de mercado y derechos de comercialización.

afirmación, hoy casi convertida en axioma, que la contabilidad actualmente no es capaz de exhibir una visión integral de la compleja realidad económica de una firma (Ruz 2012).

Las dificultades en la valoración y contabilización de un activo intangible derivan de sus propias características que son, según las definiciones que se han hecho en las normas contables vigentes, las siguientes (Ruz 2012):

- No tiene presencia física, aunque existe realmente (Ruz 2012).
- No es un activo ficticio (Ruz 2012).
- Presenta un alto grado de incertidumbre acerca de su capacidad de generar beneficios (Ruz 2012).

Un elemento significativo en esta investigación es el modelo de negocio, por lo que a continuación se expresan algunas de las definiciones realizadas por diferentes autores.

Las diferentes áreas que han estudiado la cuestión teórica de los modelos de negocio son muchas, de ahí la heterogeneidad de los estudios. Fundamentalmente, las definiciones conceptuales se encuentran ligadas a tres áreas de estudio: e-business y el uso de la información en las organizaciones, cuestiones relacionadas con estrategia como la creación de valor, ventajas competitivas y la estructura de las organizaciones; por último, el ámbito de la gestión de la innovación y la tecnología también se ha interesado en definir el concepto (Zott, Amit y Massa, 2011) (Díaz-Espina 2013).

Según Alexander Osterwalder e Yves Pigneur, del 2012: “un modelo de negocio describe las bases de cómo una organización crea, proporciona y captura valor.”

Un factor específico en esta investigación es el proceso de fijación de precio, por lo que a continuación se enuncia una definición del mismo.

La fijación de precio es el proceso que tienen las organizaciones con o sin fines de lucro para poner precios a sus productos o servicios, es considerada como actividad de rutina que depende en gran medida del desempeño de ventas y el éxito de la organización involucrada. Tanto la distribución, la promoción y la definición del producto son elementos primordiales para llegar hasta la fijación de precio (Autores 2014).

Una parte importante en esta investigación es la estrategia de venta, por lo que a continuación se emite algunas definiciones de la misma.

La estrategia de venta es la selección, definición y aceptación de un curso de acción futuro que permita, con miras al logro de los objetivos y metas de venta establecidos con anterioridad, guiar y controlar el uso óptimo de los recursos disponibles (Casado 2012).

La estrategia de venta se fundamenta en la determinación de metas y objetivos, adoptando líneas de acción, y establece los recursos precisos para compensar las escaseces y deseos concretos del cliente.

1.2 Elementos que componen un modelo de negocio

La mejor manera de describir un modelo de negocio es dividirlo en nueve módulos básicos que reflejen la lógica que sigue una empresa para conseguir ingresos. Estos nueve módulos cubren las cuatro áreas principales de un negocio: clientes, oferta, infraestructura y viabilidad económica. El modelo de negocio es una especie de anteproyecto de una estrategia que se aplicará en las estructuras, procesos y sistemas de una empresa (Osterwalder y Pigneur 2016).

El modelo de negocio siempre debe tener en cuenta el contexto en el cual se va a desenvolver como, por ejemplo: la economía y la competencia. Los nueve elementos de un modelo de negocio son (Autores 2016a):

1. **Segmentos de clientes**: definir el tipo de clientes que quiere tener. Normalmente los clientes comparten necesidades que difieren muy poco (Autores 2016a).

2. **Propuestas de valor:** pregúntese ¿Qué le diferencia a usted de sus competidores? (Autores 2016a).

3. **Canales:** seleccione los puntos de contacto con sus clientes, de acuerdo a sus expectativas. Experimente, venda y evalúe (Autores 2016a).

4. **Relaciones con los clientes:** defina las estrategias para atender a los diferentes segmentos de clientes (Autores 2016a).

5. **Flujos de ingreso:** cada tipo de flujo puede demandar un mecanismo diferente ya sea un precio fijo o dinámico o un precio negociado (Autores 2016a).

6. **Recursos clave:** los bienes pueden ser físicos, financieros, intelectuales o humanos, depende de la compañía, estos pueden ser propios o alquilados (Autores 2016a).

7. **Actividades fundamentales:** el personal ejecuta actividades cruciales para la compañía (Autores 2016a).

8. **Asociaciones clave:** considere la posibilidad de expandir su empresa por medio de varios mecanismos, como creando vínculos como proveedores, agrupación temporal, hacer alianzas estratégicas, entre otros. La asociación con otras empresas puede traerle ventajas (Autores 2016a).

9. **Estructura de costo:** que su negocio esté dirigido al costo o al valor, ofrece un servicio de bajo costo, determina la estructura de costo de su modelo (Autores 2016a).

De los nueve módulos del modelo de negocio, este trabajo de diploma se centra en el **flujo de ingreso,** que se emplea para fijar el precio del producto informático y la **relación con los clientes**, que se utiliza para crear la estrategia de venta de dicho producto.

1.3 Factores que influyen en la determinación de precios

Existen varios factores que influyen en la determinación del precio base del producto o precio de lista, como son (Mejía 2012):

- La demanda estimada: el tamaño de la demanda y la frecuencia de compra afectará las decisiones de precios hasta determinar el precio esperado, de acuerdo con lo que se cree que será el valor para los consumidores (Mejía 2012).

- Las reacciones de la competencia en productos semejantes, en productos sustitutos o productos no relacionados destinados a los mismos consumidores (“¿guerra de precios?”) (Mejía 2012).

- Otros elementos de la mezcla de marketing: si es un producto nuevo o ya establecido, el ciclo de vida del producto, su uso final, los canales y los tipos de intermediarios, la promoción que dan al producto los fabricantes o los intermediarios y el costo del producto compuesto por varios tipos de costos que influyen según los cambios en la cantidad producida: los costos fijos, los variables y los

marginales (Mejía 2012).

1.4 Metodología para establecer precios

La empresa debe considerar diversos factores al establecer su política de precios. El procedimiento recomendado comprende seis pasos (Mejía 2012):

1. **Seleccionar el objetivo de la fijación de precios:** supervivencia, maximización de utilidades, crecimiento en la participación, liderazgo en diferenciación, entre otros (Mejía 2012).

2. **Determinar la demanda:** cada precio genera un nivel de demanda distinto y por tanto tiene un impacto diferente sobre los objetivos de marketing de la empresa. La relación entre las diferentes alternativas de precio y la demanda resultante se captura en una curva de demanda con el fin de conocer la elasticidad de la demanda con respecto al precio y los puntos de equilibrio. La curva de demanda muestra la cantidad de compra probable del mercado a diferentes precios; toma en cuenta las reacciones de muchos individuos que tienen sensibilidad a los precios (Mejía 2012).

3. **Estimaciones de los costos:** la demanda establece un límite superior para el precio que la empresa puede cobrar por su producto, los costos establecen el límite inferior. Las compañías esperan cobrar un precio que cubra su costo de producir, distribuir y vender el producto, tal que incluya un rendimiento justo por su esfuerzo y riesgo (Mejía 2012).

4. **Analizar los costos, precios y ofertas de los competidores:** se deben comparar las ofertas de los competidores con la de la empresa para estimar un precio cercano al del competidor, superior o inferior, depende del liderazgo y posicionamiento que se posea en el mercado, además la empresa debe tener presente la reacción de los competidores a través de un cambio de precios (Mejía 2012).

5. **Escoger un método de fijación de precios:** una vez que se conocen las tres "C" – la estructura de demanda de los **clientes**, la función de **costos** y los precios de los **competidores** – se puede elegir el precio. Los precios de los competidores y de los sustitutos sirven de orientación, los costos establecen el límite inferior para el precio y la evaluación que hacen los clientes de las características diferenciales o exclusivas del producto establecen el precio máximo. La empresa debe seleccionar un método de fijación de precios que incluye una o más de estas tres consideraciones. A

continuación, se definen algunos de estos métodos (Mejía 2012):

a) Fijación de precios mediante márgenes:

El método más elemental para fijar precio es sumar un margen de utilidad o sobreprecio al costo total unitario del producto. Ello supone que se ha estimado un volumen de ventas y producción determinado para fijar el costo unitario a partir del cual se adiciona el margen. El método es simple pero si no se acierta en el volumen estimado, los precios resultantes pueden no ser realistas (Mejía 2012).

b) Fijación de precios por rendimiento objetivo:

La empresa determina el precio que produciría la tasa de rendimiento sobre la inversión (ROI por sus siglas en inglés) que requiere obtener: dado un monto de inversiones necesario para fabricar el producto y el costo de capital o la expectativa de ganancia, ¿cuánto y a cómo debo vender para obtener el ROI? Este método no toma muy en cuenta las consideraciones del mercado (Mejía 2012).

c) Fijación de precios por el precio vigente en el mercado:

En la fijación de precios por el precio vigente, la empresa basa su precio primordialmente en los de sus competidores, por encima o debajo según la estrategia comercial de penetración del mercado o el nivel de competitividad y posicionamiento que se disponga. En los casos que los costos son difíciles de medir o la respuesta competitiva es incierta, las empresas sienten que el precio vigente representa una buena solución (Mejía 2012).

d) Fijación de precios por contribución de costos variables o marginales:

El precio se fija con relación a los costos variables de producción calculados para cada ítem que se produce, adicionándole una cantidad extra de contribución; por tanto cada producto contribuye particularmente a los costos fijos totales y a las utilidades netas (Mejía 2012).

e) Fijación de precios basada en las condiciones del mercado:

Existen enfoques adicionales de fijación de precios basados en las condiciones de los mercados,

que son aquellos que se realizan a partir de factores externos a la organización. Para empresas que lanzan nuevos productos al mercado, por ejemplo, existen dos estrategias: el descremado o la penetración (Mejía 2012).

- Las estrategias de descremar el mercado involucran la fijación de precios altos y una intensa promoción del nuevo producto. Los objetivos de ganancia se logran a través de un alto margen por unidad vendida en lugar de maximizar el volumen de ventas. Las estrategias de descremado solo pueden emplearse donde la demanda es relativamente inelástica, para productos que tienen beneficios y/o rasgos únicos que el consumidor valora (Mejía 2012).

- Las estrategias de penetración apuntan a lograr una entrada amplia en los mercados; el énfasis está en el volumen de ventas por lo que los precios por unidad tienden a ser bajos. Esto facilita la rápida adopción y difusión del nuevo producto. Los objetivos de ganancia se alcanzan logrando un gran volumen de las ventas en lugar de un margen grande por unidad (Mejía 2012).

f) Fijación de precios sobre bases psicológicas:

La fijación de precios tiene dimensiones psicológicas, así como económicas, por tanto, se deben tener en cuenta al tomar decisiones de determinación de precios. Apelando a las emociones de los compradores existen diversas formas de fijar los precios. Estas emociones pueden ser tales como reacciones a: percepciones de calidad, precios de promoción, estímulos a la fidelidad, precios por paquete, sentimientos filiales o de amistad (Mejía 2012).

g) Precios negociados:

En los últimos años se ha introducido como modalidad de precio el compartir la ganancia entre el comprador y el vendedor. Esto sucede en productos especializados, con alto valor agregado, generalmente soluciones a la medida, donde puede determinarse el beneficio a favor del comprador, el cual es tan significativo que se entra a negociar no por precio, sino por valor agregado y a compartir las ganancias derivadas de la solución. Este modelo de precio forma parte de los llamados esquemas estratégicos de colaboración y requiere un gran conocimiento y confianza entre comprador vendedor

(Mejía 2012).

6. **Seleccionar el precio final:** para escoger su precio final la empresa debe considerar en lo posible varios de los métodos anteriores y otros factores tales como las reacciones psicológicas del cliente, la influencia de otros elementos de la mezcla de marketing, las políticas de precios tradicionales y el impacto del precio sobre otros competidores (Mejía 2012).

Por lo anteriormente expuesto se selecciona como metodología para establecer precios, la **fijación de precios mediante márgenes,** debido a que actúa como una medida de rentabilidad de una empresa, depende en gran magnitud de los costos que se emplea para desarrollar un producto, por lo que, disminuyendo los costos fijos y variables, se aumenta el precio de venta de un producto y se obtiene mayor ganancia.

1.5 Margen de utilidad

En este epígrafe se aborda el tema del margen de utilidad que se emplea para fijar el precio de venta de un determinado producto.

1.5.1 ¿Qué es el margen de utilidad?

La utilidad es el retorno positivo de la inversión originada por la empresa. En otras palabras, es la diferencia entre el precio de venta y todos los costos fijos y variables involucrados en la comercialización y en el mantenimiento de la empresa (Autores 2015).

El margen de utilidad consiste en los cargos adicionales a la tasa de interés aprobada para el consumidor y que se comparte entre el concesionario y la compañía financiera (GMAC [sin fecha]).

1.5.2 Tipos de margen de utilidad

Margen de utilidad bruta: es el precio del producto una vez deducidos los costos directos e indirectos de fabricación. En el caso de los servicios**,** es lo que resta de la cantidad abonada por la tarea después de deducir de todos los costos necesarios para su realización (Autores 2015).

Margen de utilidad neta: es la utilidad obtenida por la empresa después de pagar todos los gastos e impuestos**.** Además de los costos de producción que inciden directamente sobre el valor del producto (Autores 2015).

1.5.3 ¿Qué efecto tiene el margen de utilidad?

El margen de utilidad aumenta el costo del crédito para el consumidor. El margen de utilidad se agrega únicamente después de que la compañía financiera determina una tasa aprobada basada en el historial de crédito del consumidor. Esta tasa aprobada a menudo es conocida como "tasa de compra". El margen de utilidad se suma entonces a la tasa de compra y el resultado es la tasa más cara que paga el consumidor (GMAC [sin fecha]).

1.5.4 ¿Cómo calcular el margen de utilidad?

- La fórmula para el cálculo del margen de utilidad bruta es simple (Autores 2015):

$$\text{margen bruto} = \text{ingresos totales} - \text{costo de los productos o servicios comercializados}$$

- Para calcular el margen porcentual, se debe hacer el siguiente cálculo (Autores 2015):

$$\text{margen bruto porcentual} = \text{utilidad bruta} / \text{ingresos totales x } 100$$

- La fórmula para el cálculo del margen de utilidad neta es (Autores 2015):

$$\text{margen neta} = \text{ingresos totales} - (\text{gastos fijos} + \text{gastos variables})$$

- Para calcular el porcentaje de margen de utilidad neta, la fórmula es la siguiente (Autores 2015):

$$\text{margen neta porcentual} = \text{utilidad bruta} - \text{los gastos e impuestos} / \text{ingresos totales x } 100$$

1.6 Métodos de venta

Hoy día, las empresas se enfrentan a clientes más exigentes y con demandas más específicas. La globalización ha permeado en las ventas a tal grado que, se requiere plantear estrategias bien definidas para un mercado en donde cada día existe una mayor competencia y consumidores que esperan obtener

altos beneficios por el mismo precio, por ello, la organización de las ventas cobra relevancia y se debe definir con claridad desde un inicio (Navarro Mejía 2012).

Los vendedores son la carta de presentación de las empresas y una de las actividades que más cuidado merecen es que estos sean capaces de administrar correctamente sus relaciones con los clientes; si se satisfacen las necesidades del consumidor y se le otorga un seguimiento adecuado, será el inicio de una larga y fructífera relación comercial (Navarro Mejía 2012).

Hasta hace unas décadas, el principal método de venta era la venta personal y frecuentemente se podía ver a los vendedores de casa en casa ofreciendo sus productos. Posteriormente las ventas multinivel comenzaron a cobrar importancia y con ellas la generación de vendedores acompañados, por lo regular, de un catálogo. Las ventas a distancia por lo regular se manejaban por correspondencia y han tenido una evolución tan palpable, que hoy uno de los medios más eficaces de comercialización es internet (Navarro Mejía 2012).

A continuación, se enuncian los métodos de venta:

1. **Venta a distancia:**

Es un método en el que “no existe un contacto directo entre el comprador y el vendedor”. Las ventas a distancia se clasifican a groso modo en ventas por correspondencia, ventas por teléfono, ventas electrónicas y ventas por televisión (Navarro Mejía 2012).

Tabla 1: Clasificación de la venta a distancia

Venta a distancia	Características
Venta por correspondencia	Se realizan por medio de un catálogo que se envía a través del correo postal o se deposita directamente en los buzones. También se puede insertar a manera de anuncios, en revistas o periódicos. La ventaja de utilizar este método de venta es que describe la información del producto o servicio con un gran detalle, sin embargo las ventas por correspondencia han bajado en medida de que ha aumentado el correo electrónico.

Ventas por teléfono	Este tipo de venta se denomina también tele marketing y es especialmente usado en “mercados de gran consumo o para el lanzamiento de nuevos productos o el desarrollo y anuncio de promociones especiales”
Ventas electrónicas	En términos generales, este tipo de venta es la que se sustenta en la transmisión de datos en redes de comunicación electrónica como Internet.
Ventas por televisión	También conocidas como tele venta, consiste en la demostración de productos o servicios de manera detallada y siempre destacando sus beneficios como: precio o ventajas competitivas. Por lo regular se transmiten a diario y en los mismos horarios.

2. **Venta personal:**

Se realiza mediante el contacto directo entre el vendedor y el comprador, puede tener dos variables: realizarse dentro o fuera de las instalaciones de la empresa o comercio (Navarro Mejía 2012).

Las ventas internas, a su vez se clasifican en:

- **Venta directa:** el posible comprador entra a un establecimiento y es atendido por un vendedor que se encarga de atenderlo de forma personal (Navarro Mejía 2012).
- **Venta en libre servicio:** son las que se efectúan en tiendas de autoservicio y en donde los consumidores entran para comprar un producto elegido y evaluado por ellos mismos (Navarro Mejía 2012).
- **Venta en ferias, salones y exhibiciones:** en este tipo de venta las empresas tienen manifestaciones comerciales, donde exponen sus productos o servicios a compradores potenciales. Por lo regular se efectúan en espacios especiales o en salones de hoteles (Navarro Mejía 2012).

En el caso de las ventas externas, se subdividen en:

- **Ventas a domicilio:** se realiza en el domicilio del comprador y por lo regular en ese momento se levanta la orden de compra y posteriormente se entrega el producto o servicio. Estas ventas pueden realizarse bajo las siguientes modalidades (Navarro Mejía 2012):

- Venta de puerta en puerta (Navarro Mejía 2012).
- Venta en lugar de trabajo (Navarro Mejía 2012).
- Venta por cita en el domicilio del comprador (Navarro Mejía 2012).

- **Venta ambulante:** esta venta se caracteriza porque no tiene un espacio fijo (Navarro Mejía 2012).
- **Autoventa:** en este caso las empresas establecen rutas para llevar a los consumidores productos que por lo regular se consumen diariamente y son perecederos. El vendedor sigue la ruta que le ha sido encomendada y entrega la mercancía a clientes que casi siempre son establecimientos minoristas (Navarro Mejía 2012).

3. **Venta multinivel:**

- Bajo esta modalidad de venta, "se construye una red de vendedores independientes, a diferentes niveles, que a cambio de una comisión comercializan diferentes productos". Este tipo de ventas suelen ser muy efectivas para los consumidores y para los vendedores tienen implícitos estímulos que les resultan muy atractivos (Navarro Mejía 2012).

Por lo anteriormente expuesto se llega a la conclusión que se selecciona como método de venta, la **venta personal** debido a que se logra la comunicación directa entre el vendedor y el comprador, al vender un sistema informático el cliente debe verificar si cumple con los requisitos que este solicitó y el vendedor debe explicar cuáles son las funcionalidades con que va a contar el *software*, se realiza una reunión para debatir dichos aspectos con el objetivo de evitar malentendidos entre estos, ya sea cara a cara, o por medio de una videoconferencia. No se escoge otro tipo de venta porque, por ejemplo, en la venta a distancia las personas involucradas en la venta no siempre se ven, es decir, el vendedor no tiene como confirmar en el momento de la venta que el producto sea del agrado del comprador y que satisface sus necesidades; y en la venta multinivel no sería satisfactorio para el cliente que terceras personas le vendieran un producto informático, cuando pudiera darse el caso, de que estos no podrían estar calificados para realizar dicha tarea.

1.7 Proceso para crear una estrategia de venta

Las ventas son un proceso que implica un orden secuencial que incluye diferentes fases (Navarro Mejía 2012):

1. **Localización y calificación de los clientes:**

- Para que exista una operación de venta es necesario que existan dos elementos que son el producto o servicio, y el cliente, aquí es un prospecto ya que no ha comprado nada aún. Los prospectos potenciales son aquellos individuos que tienen la necesidad de adquirir un determinado producto o servicio y con los que por lo tanto, debe existir un acercamiento (Navarro Mejía 2012).

- Una vez que se ha localizado a los prospectos, es conveniente enlistarlo con el propósito de hacer una valoración de cada uno de ellos basados en aspectos como: insistencia en la necesidad de adquirir el producto, poder adquisitivo, posibilidad de generar clientes frecuentes (que compren constantemente el producto o servicio) (Navarro Mejía 2012).

- Por último dentro de esta fase, se procede a priorizar a los clientes prestando especial atención a aquellos que se puedan atraer a corto plazo (Navarro Mejía 2012).

2. **Acercamiento con el cliente:**

- Una vez evaluados los prospectos, inicia el acercamiento con el futuro cliente y para ello se debe (Navarro Mejía 2012):

    - Conocer al cliente, buscando la mayor información posible de él incluyendo su situación comercial. En el caso de los servicios financieros (tarjetas de crédito, cuentas bancarias, inversiones, entre otros) es particularmente importante este punto y de hecho está señalado en los procesos internos de las empresas que los ofrecen (Navarro Mejía 2012).

    - Ubicación del cliente en el mercado, que se refiere a determinar la frecuencia con la que el cliente compra productos o servicios similares al que se le va a ofrecer (Navarro Mejía 2012).

    - Preparación de la visita, ya con toda la información necesaria y planteando estrategias que

faciliten la negociación y el cierre de la venta (Navarro Mejía 2012).

3. **Argumento de ventas:**

- Dentro de esta parte del proceso de las ventas el vendedor hace un sondeo de las necesidades del cliente. Para hacer más efectiva esta fase se recurre al argumentario de ventas, que es un documento creado por el vendedor o bien, por el departamento de ventas y cuyo fin es determinar los puntos en los que se dará más énfasis. Este argumentario "se basa en la fórmula AIDA (atención, interés, deseo, acción) expresada por Kotler, es decir, debe captar la atención, animar el interés, provocar el deseo y promover la acción, y se concreta en un discurso que tiene tres elementos básicos" (Navarro Mejía 2012):

    1. Descripción objetiva de las características del producto o servicio (Navarro Mejía 2012).

    2. Énfasis en las ventajas que ofrece el producto o servicio, principalmente en comparación con similares que ofrezca la competencia (Navarro Mejía 2012).

    3. Beneficios adicionales que el cliente obtendrá del producto (Navarro Mejía 2012).

4. **Consolidación o cierre de la venta:**

- El cierre de venta es el momento clave del proceso, si un vendedor no tiene la suficiente pericia para manejar esta fase, es muy probable que no se logre el objetivo. Enseguida se detallan las variables a considerar durante esta etapa (Navarro Mejía 2012):

    - **Estrategia de cierre:** el vendedor debe prepararse y generar estrategias para: clientes que presentan muchas objeciones, clientes que dan largas o clientes a los que les dejó de interesar la venta (Navarro Mejía 2012).

    - **Técnicas de cierre:** aquí el vendedor puede optar por: preparar un breve discurso final que resume las ventajas del producto o servicio, mencionarle al cliente las opciones de pedido, para que él pueda decidir cerrar la venta (Navarro Mejía 2012).

    - **Tipos de cierre:** estos pueden ser: directos, que son en los que se logra el acuerdo y se

levanta el pedido; condicionado, se cierra la venta pero queda sujeta a alguna condición como un descuento, por ejemplo; cierre a prueba, el cliente acepta probar el producto o servicio durante determinado período para después decidir si lo compra o no (Navarro Mejía 2012).

5. **Servicio de atención al cliente:**

- El verdadero objetivo de las empresas actuales es conservar a sus clientes y atraer más. Los costos de un mal servicio son tan perjudiciales, que se deben evitar a toda costa. El servicio de atención a clientes tiene dos funciones fundamentales (Navarro Mejía 2012):

  a) Seguimiento posventa: aquí se debe corroborar si el producto o servicio fue de la entera satisfacción del cliente y si se entregó según lo estipulado en el cierre. También es importante hacer una llamada de cortesía donde se le agradece al cliente el haber adquirido el producto o contratado el servicio. También se puede aprovechar para ofrecerle al cliente otros productos o solicitarle que de una lista de tres prospectos que considere podrían interesarse en lo que se le ha vendido (Navarro Mejía 2012).

  b) Seguimiento de quejas: las quejas son una insatisfacción que el cliente expresa por el producto o servicio y casi siempre se generan por un defecto en el producto o por el incumplimiento de una de las condiciones de compra. Para dar seguimiento a una queja se debe escuchar atentamente al cliente garantizándole que su problema quedará resuelto pero sin falsas promesas y por supuesto, resolver el problema (Navarro Mejía 2012).

1.8 Metodología, herramientas y lenguajes utilizados

En este epígrafe se explica la selección de las metodologías, herramientas y lenguaje de programación empleados en la solución de este trabajo de diploma.

1.8.1 Proceso Unificado Ágil (AUP)

De las cuatro fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se

unifican las restantes tres fases de AUP en una sola, la que se llama Ejecución y se agrega la fase de Cierre (Rodríguez 2014).

Fases Variación AUP-UCI:

1. **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, decidir si se ejecuta o no el proyecto (Rodríguez 2014).

2. **Ejecución:** en esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software* (Rodríguez 2014).

3. **Cierre:** en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto (Rodríguez 2014).

Descripción de las disciplinas:

AUP propone siete disciplinas (Modelo, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyecto y Entorno), se decide para el ciclo de vida de los proyectos de la UCI tener ocho disciplinas, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos y Análisis, y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en tres disciplinas: Pruebas Internas, de Liberación y Aceptación y la disciplina Despliegue se considera opcional. Las restantes tres disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMI-DEV v1.3 para el nivel 2, serían Gestión de la configuración (CM), Planeación de proyecto (PP) y Monitoreo y control de proyecto (Rodríguez 2014).

El escenario de la disciplina Requisitos que se emplea en esta aplicación es el Escenario No 2. Este escenario se aplica para proyectos que modelen el negocio con Modelo Conceptual (MC) solo pueden modelar el sistema con Caso de Uso del Sistema (CUS), el mismo plantea lo siguiente:

- **Escenario No 2:** aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información (Rodríguez 2014).

Se selecciona la metodología de desarrollo de *software* AUP para la UCI, porque es una variedad creada por la universidad a la metodología ágil AUP y está definida por la casa de altos estudios como el escrito oficial que rige la actividad productora.

1.8.2 Microsoft SQL Server 2008

SQL significa lenguaje estructurado de consulta (*Structured Query Language*). Es un lenguaje estándar de cuarta generación que se utiliza para definir, gestionar y manipular la información contenida en una Base de Datos Relacional (Santamaría y Hernández 2015).

En los lenguajes procedimentales de tercera generación se deben especificar todos los pasos que hay que dar para conseguir el resultado. Sin embargo en SQL tan solo se debe indicar al Sistema Gestor de Base de Datos (SGBD) qué es lo que se quiere obtener, y el sistema decidirá cómo obtenerlo (Santamaría y Hernández 2015).

Es un lenguaje sencillo y potente que se emplea para la gestión de la base de datos a distintos niveles de utilización: usuarios, programadores y administradores de la base de datos (Santamaría y Hernández 2015).

Características de *Microsoft SQL Server*:

- Soporte de transacciones (Santamaría y Hernández 2015).
- Escalabilidad, estabilidad y seguridad (Santamaría y Hernández 2015).

- Soporta procedimientos almacenados (Santamaría y Hernández 2015).
- Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL[2] y DML[3] gráficamente (Santamaría y Hernández 2015).
- Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y las terminales o clientes de la red solo acceden a la información (Santamaría y Hernández 2015).
- Además permite administrar información de otros servidores de datos (Santamaría y Hernández 2015).

Microsoft SQL Server constituye la alternativa de *Microsoft* a otros potentes SGBD como son: *Oracle, Sybase ASE, PostgreSQL o MySQL* (Santamaría y Hernández 2015).

En este trabajo de diploma se emplea *Microsoft SQL Server* 2008 para crear la base de datos de la solución propuesta, donde se encuentra guardada la información relacionada con la oferta de un producto informático. *Microsoft SQL Server* es un sistema que permite maniobrar las bases de datos del modelo relacional, además puede configurarse para usar múltiples instancias en el mismo servidor físico.

1.8.3 Microsoft Visual Studio 2010 Ultimate

Visual Studio es una suite de aplicaciones creada por *Microsoft* para dar a los desarrolladores un ambiente completo de desarrollo de *software* para plataformas *Windows* y .NET. *Visual Studio* puede ser usado para escribir aplicaciones de consola, aplicaciones de escritorio, aplicaciones móviles, aplicaciones ASP.NET y servicios web ASP.NET, en lenguajes de programación como C++, C#, VB.NET, *Java* y más; permite que estas se puedan intercomunicar entre estaciones de trabajo, páginas web y dispositivos móviles. *Visual Studio* también incluye varias herramientas de desarrollo adicionales, estas, dependen de la versión de

[2] Un lenguaje de definición de datos (*Data Definition Language*) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

[3] Lenguaje de Manipulación de Datos (*Data Manipulation Language*) es un lenguaje proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de la misma llevar a cabo las tareas de consulta o modificación de los datos contenidos en las Bases de Datos del Sistema Gestor de Bases de Datos.

Visual Studio que se esté usando (Morales 2012).

La aplicación se implementa en *Microsoft Visual Studio* 2015 debido a que es un entorno de desarrollo integrado y soporta varios lenguajes de programación, seleccionando C#.

1.8.4 Lenguaje de programación C#

C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como *Visual Basic*, *Java* o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues *Microsoft* ha escrito la mayor parte del Lenguaje de Programación Básico Combinado (por sus siglas en inglés BCPL) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET *Framework* SDK[4] (Seco 2000).

Características de C#:

- **Sencillez:** elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET (Seco 2000).
- **Modernidad:** incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como *Java* o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción *foreach* que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico *string* para representar cadenas o la distinción de un tipo *bool* específico para representar valores lógicos (Seco 2000).
- **Orientación a objetos:** como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo

[4] Un *kit* de desarrollo de *software* o SDK (*software development kit*) es generalmente un conjunto de herramientas de desarrollo de *software* que le permite al programador o desarrollador de *software* crear aplicaciones para un sistema concreto.

que reduce problemas por conflictos de nombres y facilita la legibilidad del código (Seco 2000).

- **Orientación a componentes:** la propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros) (Seco 2000).

- **Gestión automática de memoria:** todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR[5]. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y solo se realiza cuando este se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en la fuente-, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción *using* (Seco 2000).

- **Seguridad de tipos:** incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura (Seco 2000).

- **Instrucciones seguras:** para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (==) con el de asignación (=); y todo caso de un *switch* ha de terminar en un *break* o goto que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación (Seco 2000).

- **Sistema de tipos unificado:** a diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada

[5] *Common Language Runtime* o CLR es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma *Microsoft*.NET. El CLR es el encargado de compilar una forma de código intermedio llamada *Common Intermediate Language* (por sus siglas en inglés CIL), al código de máquina nativo, mediante un compilador en tiempo de ejecución.

System.Object, por lo que dispondrán de todos los miembros definidos en esta clase (es decir, serán "objetos") (Seco 2000).

- **Extensibilidad de tipos básicos:** permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos), se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref. (Seco 2000).

- **Extensibilidad de operadores:** para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que C++ y a diferencia de *Java*, C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se apliquen a diferentes tipos de objetos (Seco 2000).

- **Extensibilidad de modificadores**: ofrece a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente de información adicional a la generada por el compilador que luego podrá ser consultada en tiempo de ejecución a través de la librería de reflexión de .NET. Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores (Seco 2000).

- **Versionable:** incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos (Seco 2000).

- **Eficiente:** en principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de *Java*, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador *unsafe*) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una

eficiencia y velocidad procesamiento muy grandes (Seco 2000).

- **Compatible:** para facilitar la migración de programadores, C# no solo mantiene una sintaxis muy similar a C, C++ o *Java* que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados *Platform Invocation Services (PInvoke),* la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs[6] de la API[7] Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que estas muchas veces esperan recibir o devuelven punteros (Seco 2000).

La aplicación se implementa en el lenguaje de programación C# debido a que existe un rango más amplio y definido de tipos de datos, permite la declaración de propiedades dentro de cualquier clase y permite mantener múltiples versiones de clases en forma binaria, colocándolas en diferentes espacios de nombres.

1.8.5 Lenguaje Unificado de Modelado (UML 2.1)

UML es un lenguaje estándar para especificar, visualizar, construir y documentar los artefactos de sistemas informáticos, así como para el modelado de negocios y otros sistemas. UML brinda variados elementos de esquematización para representar las diferentes partes de un sistema, de ellos se utilizan en el desarrollo de la aplicación los diagramas de casos de uso, de clases del diseño, de despliegue, de componentes, de colaboración y el resto de los diagramas (Autores 2016b).

La aplicación emplea UML debido a que es fácil de entender, se puede implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

[6] Una biblioteca de enlace dinámico o DLL (*Dynamic-Link Library*) es el término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo.

[7] La interfaz de programación de aplicaciones o API *(Application Programming Interface*) es un conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

1.8.6 Visual Paradigm 8.0

Visual Paradigm es una herramienta profesional que se utiliza para el modelado, soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una rápida construcción de aplicaciones. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación (Autores 2016c).

Para la realización de la aplicación se emplea la herramienta *Visual Paradigm* porque facilita el modelado de UML, ya que proporciona instrumentos específicos para ello. Esto también permite la estandarización de la documentación, ya que la misma se ajusta al estándar soportado por la herramienta.

1.9 Sistemas informáticos para la fijación de precio

Al culminar un estudio minucioso sobre múltiples bibliografías consultadas, tanto a nivel nacional como internacional sobre aplicaciones informáticas existentes para fijar el precio de venta de un producto intangible, se verifica que no existe actualmente ningún sistema informático que permita la fijación del precio de venta de productos intangibles, por lo que, no se puede realizar comparaciones con otros sistemas similares, esto imposibilita llegar a conclusiones sobre las semejanzas y diferencias que se pudieran encontrar entre estos. Por lo que se ratifica que es necesario implementar una herramienta que permita fijar el precio de venta de los productos informáticos y consigo generar la oferta de estos productos en CIGED.

1.10 Conclusiones del capítulo

Este capítulo describe los principales conceptos asociados con el modelo de negocio, la fijación de precios y la estrategia de venta de productos, guiado sobre la base de diferentes autores. Se selecciona como metodología de fijación de precio, la fijación de precios mediante márgenes, empleando un margen de utilidad. Se selecciona la estrategia de venta para los productos del centro, empleando la venta personal debido a que se logra la comunicación directa con el cliente. La metodología a emplear para darle solución al objetivo planteado es AUP-UCI, se tiene como escenario de la disciplina Requisitos que aplica a esta sistema, el Escenario No 2. Se evidencia que en la actualidad no existe un sistema informático que posibilite fijar el precio de venta de un producto intangible. La aplicación se desarrolla utilizando la plataforma

Microsoft .NET, específicamente el *Framework* 4.0. Para la implementación del sistema se utiliza el lenguaje C#, se usa como entorno de desarrollo integrado *Visual Studio* 2010 *Ultimate*. Toda la información manejada por el sistema se almacena bajo el servidor SQL 2008. La solución que se propone está basada en la guía de aplicaciones compuestas para *WPF\Silverlight*.

Capítulo 2: Descripción de la solución propuesta

En este capítulo se genera un modelo de dominio donde se analizan las entidades y conceptos existentes en el contexto, donde se realiza la solución propuesta, se especifican las relaciones presentes entre cada uno de estos. Se explica los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se muestra el diagrama de casos de uso del sistema, se representan los actores y los casos de uso del sistema. Se visualiza el modelo de clases, así como el diagrama de clases y el diagrama de entidad relación. Se selecciona el estilo arquitectónico y los patrones de diseño a emplear en la solución propuesta.

2.1 Solución propuesta

Se propone como solución el desarrollo de una aplicación informática con nombre “Sistema de Gestión de Ofertas Comerciales (SIGOC)”, la cual pretende satisfacer la necesidad de fijar el precio de venta de los productos informáticos que se comercializan en CIGED y consigo gestionar las ofertas de dichos productos, permite al usuario que interactúa con la misma trabajar de forma sencilla sobre las funcionalidades que brindará.

Para acceder a la aplicación los usuarios deben introducir sus credenciales y cada uno tendrá las siguientes funciones específicas:

- El comercial se encarga de generar y exportar la oferta de los productos informáticos de CIGED, así como seleccionar el método de fijación de precio para posteriormente calcular el precio de venta de los productos.
- El especialista se encarga de gestionar un reporte de la oferta.
- El administrador se encarga de gestionar las personas, los usuarios, los productos, los módulos y los servicios.

2.2 Modelo de dominio o conceptual

En el modelo de dominio se caracterizan los conceptos reales que se vinculan con el proyecto y las relaciones presentes entre estos. Se emplea el vocablo “dominio” para distinguirlo del modelo de negocio,

porque este último tiene un significado más abarcador. Se basa en una porción del negocio particularmente, la vinculada con el contexto del proyecto, por lo que el vocablo “dominio” simboliza una fracción del “negocio”. El modelo de dominio permite ampliar la visión del problema y favorece el esclarecimiento de la terminología o nomenclatura del dominio.

A continuación, se muestra la definición de los conceptos del modelo del dominio:

- **Oferta:** representa un objeto que contiene los entregables, los hitos, las condiciones de los contratos, el centro donde pertenece y el área correspondiente a este, la naturaleza y la clasificación de una oferta, el producto informático, los módulos, los servicios, la ficha de cálculo, el precio de venta y el método de fijación de precio para calcularlo, los reportes de la oferta; las personas y los usuarios con su respectivo cargo, rol, permisos y subsistemas.
- **Entregable:** representa un objeto destinado al cliente, ya sea interno o externo al centro.
- **Hito:** representa un evento importante que se usa para supervisar el progreso de la planificación del proyecto.
- **Condición de contrato:** representa las cláusulas que redacta el vendedor del producto informático para utilizarlas en los contratos que realice con sus clientes, impide que estos las negocien o transformen, anunciando todos los aspectos de la relación entre los mismos.
- **Centro:** representa el lugar al que pertenecen los productos informáticos.
- **Área:** representa el ámbito donde se desarrollan los productos informáticos.
- **Naturaleza:** representa la característica propia de la oferta.
- **Clasificación:** representa el grupo al que pertenece la oferta.
- **Producto:** representa el producto informático al que se le desea calcular su precio de venta.
- **Módulo:** representa una parte de un producto informático.

- **Servicio:** representa las actividades que intentan satisfacer las necesidades de los clientes.
- **Ficha de cálculo:** representa la herramienta donde se calcula el precio de venta del producto informático.
- **Precio de venta:** representa el precio que tendrá el producto informático, para su posterior venta a los clientes.
- **Método de fijación de precio:** representa la técnica que se utiliza para calcular el precio de venta de un producto informático.
- **Reporte:** representa el informe de una determinada oferta.
- **Persona:** representa un individuo con sus características.
- **Usuario:** representa una persona que interactúa con el sistema y sus funcionalidades.
- **Rol:** representa la función que una persona desempeña en el centro.
- **Permiso:** representa el derecho de acceso que se les dan a los usuarios para realizar sus tareas.
- **Cargo:** representa una plaza, se demuestra que una persona tiene una responsabilidad en el centro.
- **Subsistema:** representa un conjunto de elementos interrelacionados.

2.2.1 Diagrama del modelo de dominio

A continuación, se muestra el modelo de dominio donde se representan los principales conceptos relacionados con el sistema, así como las relaciones entre ellos:

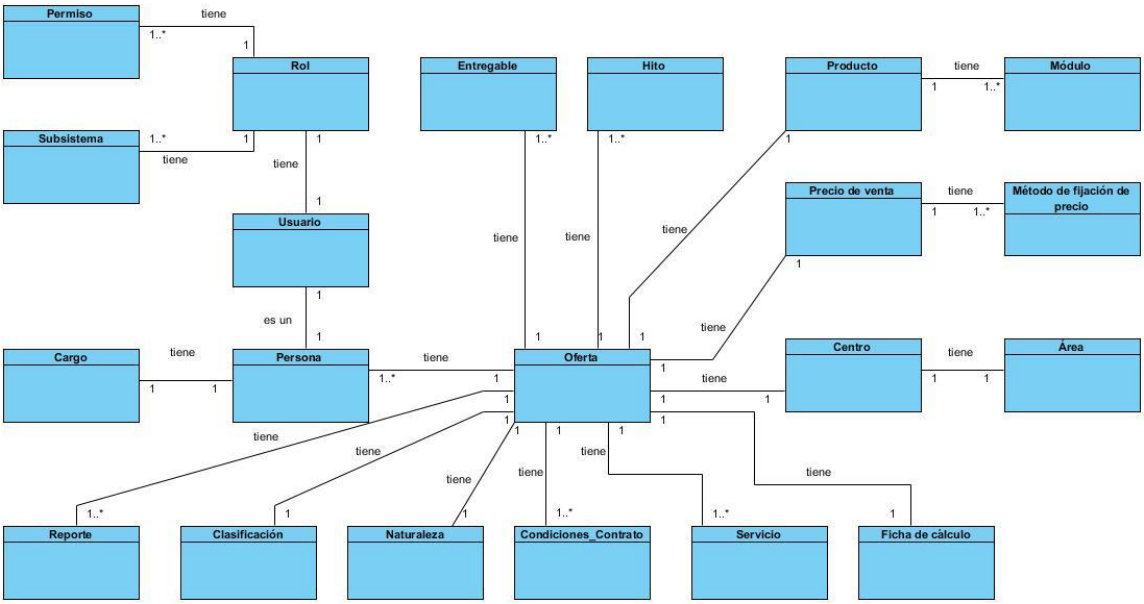


Figura 1: Modelo de dominio

2.3 Especificación de los requisitos de *software*

Existen diversas actividades esenciales en el momento que se decide desarrollar un *software*, una de estas es la definición de los requisitos del *software*. El sistema de cumplir con los requerimientos especificados para que logre su objetivo, es preciso verificar y validar estos, de esta manera impedir errores en las etapas siguientes del desarrollo del *software*.

2.3.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Sommerville 2005).

La tabla siguiente muestra los requisitos funcionales (RF) identificados para el desarrollo de la aplicación y la descripción de los mismos:

Tabla 2: Requisitos funcionales

Requisito	Descripción
RF1. Autenticar usuario.	El usuario introduce sus credenciales para poder acceder al sistema.
RF2. Crear oferta.	Se crea una oferta en el sistema.
RF3. Listar oferta.	Se listan los datos de la oferta existente en el sistema.
RF4. Modificar oferta.	Se modifican los datos de la oferta existente en el sistema.
RF5. Guardar oferta.	Se guardan los datos de la oferta existente en el sistema.
RF6. Exportar oferta.	Se exporta la oferta existente en el sistema.
RF7. Seleccionar el método de fijación de precio.	Se selecciona el método de fijación de precio para establecer el precio de venta del producto informático.
RF8. Calcular el precio del producto.	Se calcula el precio de venta del producto informático en la ficha de cálculo.
RF9. Crear reporte de la oferta.	Se crea un reporte de la oferta en el sistema.
RF10. Eliminar reporte de la oferta.	Se elimina un reporte de la oferta existente en el sistema.
RF11. Crear persona.	Se crea una persona en el sistema.
RF12. Listar persona.	Se listan los datos de la persona existente en el sistema.
RF13. Modificar persona.	Se modifican los datos de la persona existente en el sistema.
RF14. Guardar persona.	Se guardan los datos de la persona existente en el sistema.
RF15. Crear usuario.	Se crea un usuario en el sistema.
RF16. Listar usuario.	Se listan los datos del usuario existente en el sistema.
RF17. Modificar usuario.	Se modifican los datos del usuario existente en el sistema.
RF18. Guardar usuario.	Se guardan los datos del usuario existente en el sistema.
RF19. Crear producto.	Se crea un producto informático en el sistema.
RF20. Listar producto.	Se listan los datos del producto informático existente en el sistema.
RF21. Modificar producto.	Se modifican los datos del producto informático existente en el sistema.
RF22. Guardar producto.	Se guardan los datos del producto informático existente en el sistema.
RF23. Crear módulo.	Se crea un módulo en el sistema.

RF24. Listar módulo.	Se listan los datos del módulo existente en el sistema.
RF25. Modificar módulo.	Se modifican los datos del módulo existente en el sistema.
RF26. Guardar módulo.	Se guardan los datos del módulo existente en el sistema.
RF27. Crear servicio.	Se crea un servicio en el sistema.
RF28. Listar servicio.	Se listan los datos del servicio existente en el sistema.
RF29. Modificar servicio.	Se modifican los datos del servicio existente en el sistema.
RF30. Guardar servicio.	Se guardan los datos del servicio existente en el sistema.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville 2005).

A continuación, se listan los requisitos no funcionales (RNF):

- **RNF1.** Usabilidad: la aplicación debe ser de fácil entendimiento para todos los usuarios que la utilicen.

- **RNF2.** Confiabilidad: la información almacenada en la base de datos debe ser confiable en cuanto a su veracidad e integridad desde su recopilación.

- **RNF3.** Disponibilidad: el sistema debe responder en toda ocasión al acceso a la información, siempre y cuando el servidor de base de datos esté trabajando correctamente.

- **RNF4.** Recursos: para el correcto funcionamiento del sistema se debe tener:

    - **Servidor de Base de Datos:**
      Procesador: Intel (R) Celeron (R) 1.86 GHz.
      Memoria RAM: 4 GB mínimo.
      Disco Duro: 160 GB mínimo.

- **RNF5.** Interfaz de *software*: el sistema debe integrarse con el sistema gestor de base de datos

Microsoft SQL Server 2008.

- **RNF6.** Diseño e implementación: se empleará el lenguaje de programación C#.

Por lo tanto: para que el usuario se sienta satisfecho con la aplicación, se recomienda usar los recursos mínimos que deben estar en correspondencia con los definidos anteriormente, de lo contrario no se garantiza el correcto funcionamiento de la misma.

2.4 Definición de los actores

Un actor presenta comportamientos, incluyendo el propio sistema que se está estudiando cuando solicita los servicios de otros sistemas, (…). Los actores no son solamente roles que juegan personas, sino también organizaciones, *software* y máquinas (Larman 1999).

En la siguiente tabla se puntualizan los objetivos de los actores presentes en el sistema:

Tabla 3: Objetivos del actor

Actor	Objetivos
Usuario	• Autenticar usuario.
Comercial	• Gestionar oferta. • Exportar oferta. • Seleccionar el método de fijación de precio. • Calcular el precio del producto.
Especialista	• Gestionar reporte de la oferta.
Administrador	• Gestionar persona. • Gestionar usuario. • Gestionar producto. • Gestionar módulo. • Gestionar servicio.

2.5 Definición de los casos de uso

Un caso de uso especifica una cadena de acciones realizadas por un sistema, para proporcionar un efecto visible del valor a una persona o a otro sistema empleando el producto en desarrollo.

Luego del análisis de los requisitos funcionales del sistema se definieron los casos de uso (CU) siguientes:

- **CU1.** Autenticar usuario.
- **CU2.** Gestionar oferta.
- **CU3.** Exportar oferta.
- **CU4.** Seleccionar el método de fijación de precio.
- **CU5.** Calcular el precio del producto.
- **CU6.** Gestionar reporte de la oferta.
- **CU7.** Gestionar persona.
- **CU8.** Gestionar usuario.
- **CU9.**Gestionar producto.
- **CU10.** Gestionar módulo.
- **CU11.** Gestionar servicio.

2.5.1 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema evidencia la correlación entre los actores y los casos de uso del sistema. Simboliza la funcionalidad que brinda dicho sistema en cuanto a su interacción externa.

El diagrama de casos de uso del sistema se muestra en la figura siguiente:

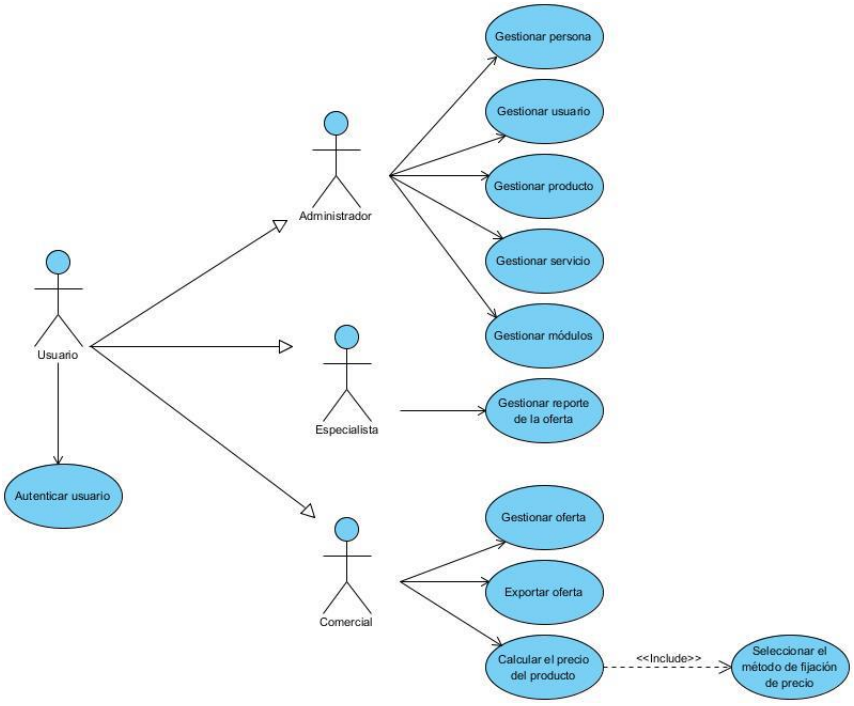


Figura 2: Diagrama de casos de uso del sistema

2.5.2 Descripción de los casos de uso del sistema

A continuación, se muestran la descripción del CU2.Gestionar oferta. El resto de las descripciones se encuentran el anexo 2 de este trabajo de diploma.

Tabla 4: Descripción del CU2.Gestionar oferta

Objetivo	Gestionar una oferta
Actores	Comercial
Resumen	El caso de uso inicia cuando el comercial desea crear, listar, modificar o guardar una oferta en la aplicación
Complejidad	Media
Prioridad	Alta
Precondiciones	El comercial accede a la aplicación
Postcondiciones	Se crea, se lista, se modifica o se guarda una oferta

Flujo de eventos		
Flujo básico: Gestionar oferta		
	Actor	**Sistema**
1.	Indica que desea crear, listar, modificar o guardar una oferta	**a)** Si el usuario decide crear una oferta ver Sección 1: Crear Oferta **b)** Si el usuario decide modificar una oferta ver Sección 2: Modificar Oferta **c)** Si el usuario decide listar una oferta ver Sección 3: Listar Oferta **d)** Si el usuario decide guardar una oferta ver Sección 4: Guardar Oferta
Sección 1: Crear Oferta		
Flujos alternos <Crear Oferta>		
	Actor	**Sistema**
1.	Selecciona la opción Crear	
2.		Muestra la interfaz Crear Oferta que permite crear una oferta
3.	Selecciona el botón Guardar	
4.		Se muestra un mensaje de información “La oferta ha sido creada correctamente”
Sección 2: Modificar Oferta		
Flujo alterno <Modificar Oferta>		
	Actor	**Sistema**
1.	Selecciona la opción Modificar	
2.		Muestra la interfaz Modificar Oferta que permite modificar una oferta
3.	Selecciona el botón Guardar	
4.		Se muestra un mensaje de información “La oferta se ha modificado correctamente”
Sección 3: Listar Oferta		
Flujos alternos <Listar Oferta>		
	Actor	**Sistema**

1.	Selecciona la opción Listar	
2.		Muestra la interfaz Listar Oferta que permite listar una oferta
3.	Selecciona el botón Guardar	
4.		Se muestra un mensaje de información “La oferta se ha listado correctamente”
Sección 4: Guardar Oferta		
Flujos alternos <Guardar Oferta>		
	Actor	Sistema
1.	Selecciona la opción Guardar	
2.		Se muestra un mensaje de información “La oferta ha sido guardada correctamente”

2.6 Modelo basado en clases

El modelado basado en clases representa los objetos que manipulará el sistema, las operaciones (también llamadas métodos o servicios) que se aplicarán a los objetos para efectuar la manipulación, las relaciones (algunas de ellas jerárquicas) entre los objetos y las colaboraciones que tienen lugar entre las clases definidas. Los elementos de un modelo basado en clases incluyen las clases y los objetos, atributos, operaciones, modelos clase-responsabilidad-colaborador (CRC), diagramas de colaboración y paquetes (Pressman 2010).

2.6.1 Diagrama de clases persistentes

Un tipo de diagrama de vital importancia para la definición del sistema es el diagrama de clases. En este diagrama se representan las entidades de forma estática en forma de clases. Una clase puede contener atributos, propiedades y métodos (León 2000).

A continuación, se muestra el diagrama de clases persistentes de la solución propuesta:

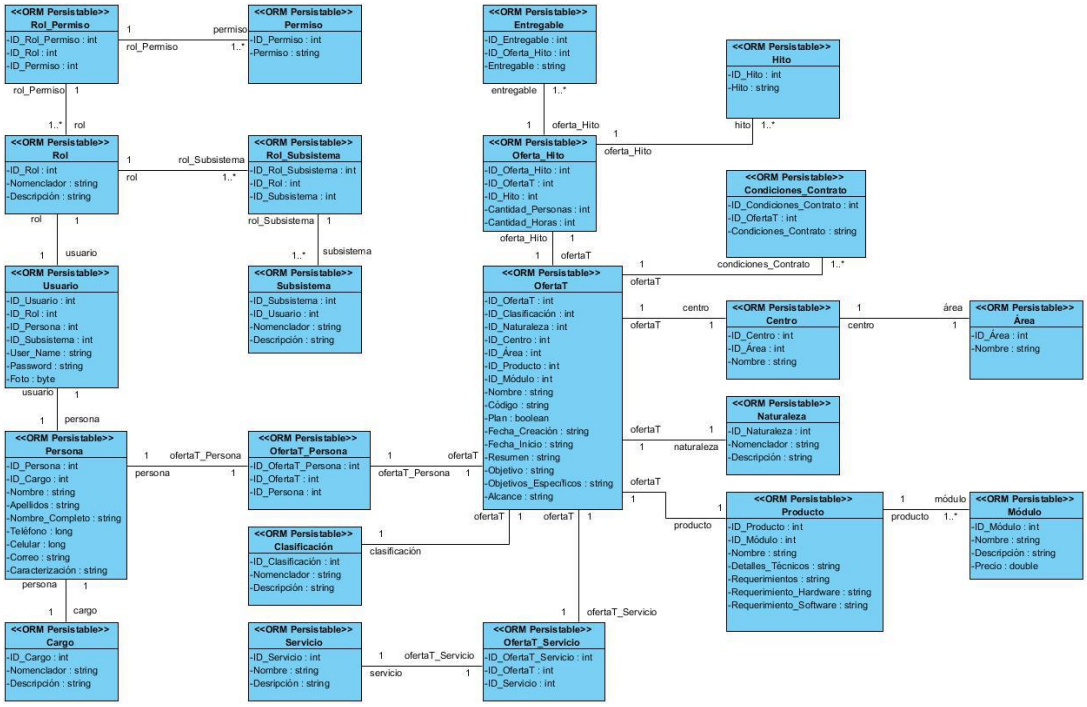


Figura 3: Diagrama de clases persistentes

2.7 Modelo de datos

Si los requerimientos del *software* incluyen la necesidad de crear, ampliar o hacer interfaz con una base de datos, el equipo del *software* tal vez elija crear un modelo de datos como parte del modelado general de los requerimientos. Un ingeniero o analista de *software* define todos los objetos de datos que se procesan dentro del sistema, la relación entre ellos y otro tipo de información que sea pertinente para las relaciones. El diagrama entidad-relación (DER) aborda dichos aspectos y representa todos los datos que se introducen, almacenan, transforman y generan dentro de una aplicación (Pressman 2010).

Un objeto de datos es una representación de información compuesta que debe ser entendida por el *software*. Información compuesta quiere decir, algo que tiene varias propiedades o atributos diferentes (Pressman 2010).

Los atributos de los datos definen las propiedades de un objeto de datos y tienen una de tres diferentes características. Se usan para (Pressman 2010):

1. Nombrar una instancia del objeto de datos (Pressman 2010).

2. Describir la instancia (Pressman 2010).

3. Hacer referencia a otra instancia en otra tabla (Pressman 2010).

2.7.1 Diagrama entidad relación

En el diagrama entidad-interrelación se representa la estructura que debe tener la base de datos para almacenar los objetos persistentes del sistema. En este diagrama se conectan las entidades, que hacen uso de claves ajenas para mantener la semántica descrita en el diagrama de clases (León 2000).

A continuación, se muestra el diagrama entidad relación de la solución propuesta:

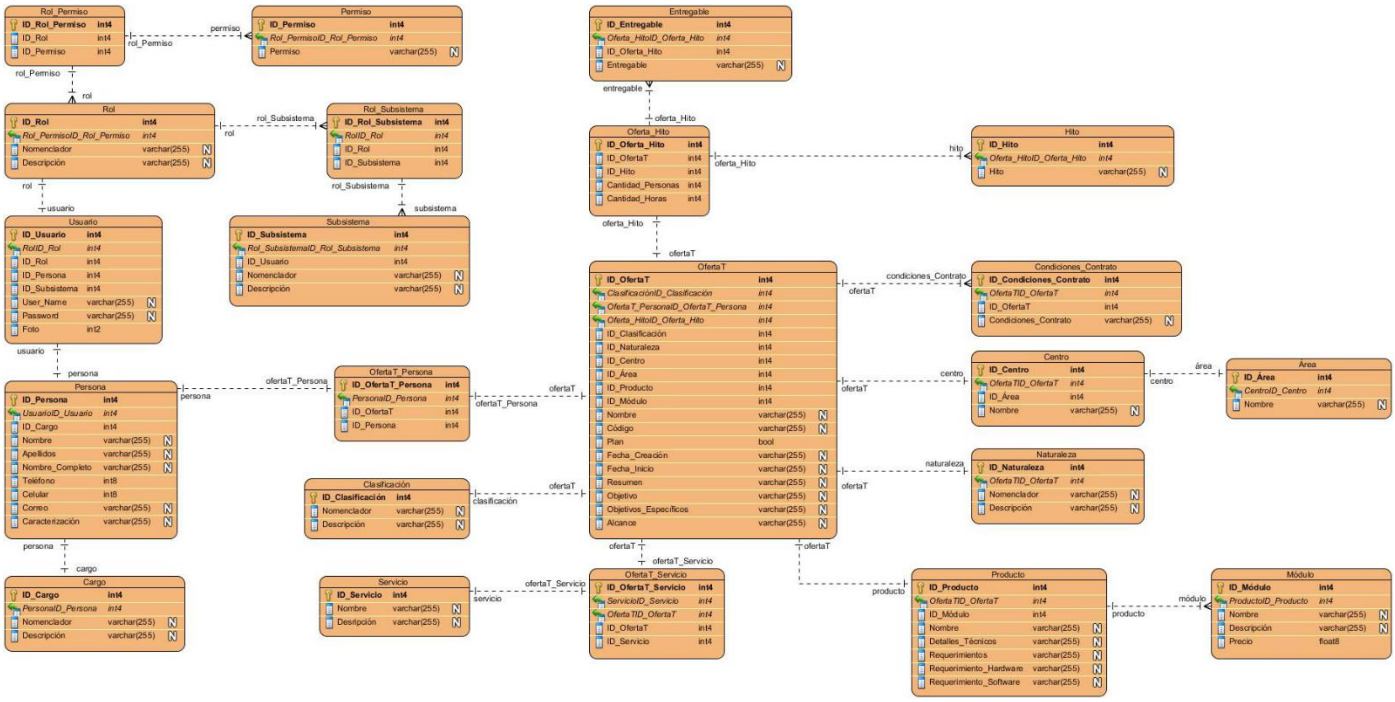


Figura 4: Diagrama entidad relación

2.8 Estilo arquitectónico

Los patrones arquitectónicos son los relacionados con el diseño a gran escala, que se aplican típicamente durante las primeras iteraciones (la fase de elaboración) cuando se establecen las estructuras y conexiones más importantes (Larman 1999).

Los estilos de Llamada y Retorno son: Modelo-Vista-Controlador (MVC), Arquitecturas en Capas, Arquitecturas Orientadas a Objetos y Arquitecturas Basadas en Componentes.

Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Son los estilos más generalizados en sistemas en gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas (Reynoso y Kiccillof 2004).

Una arquitectura de múltiples capas (n-*tiers*), permite que un número ilimitado de programas corran simultáneamente, enviando información de uno a otro, utilizando diferentes protocolos para comunicarse e interactuar simultáneamente; esto permite crear aplicaciones más poderosas que proporcionen diversos servicios a varios clientes (Trellini 2015).

SIGOC presenta un diseño modular, que facilita las futuras modificaciones o agregaciones que se hagan. El estilo arquitectónico seleccionado para la implementación es un estilo en capas donde cada una de estas solo accede a la inmediata inferior y mediante interfaces (nunca clases concretas), de forma que el nivel de acoplamiento entre una y otra se mantenga al mínimo posible.

Vista arquitectónica de la solución propuesta:

La aplicación está compuesta por tres subsistemas que a su vez se componen de diferentes módulos. Todos estos, dispuestos sobre el *Framework* desarrollado por el equipo de arquitectura del proyecto. La organización de subsistemas y módulos es la siguiente:

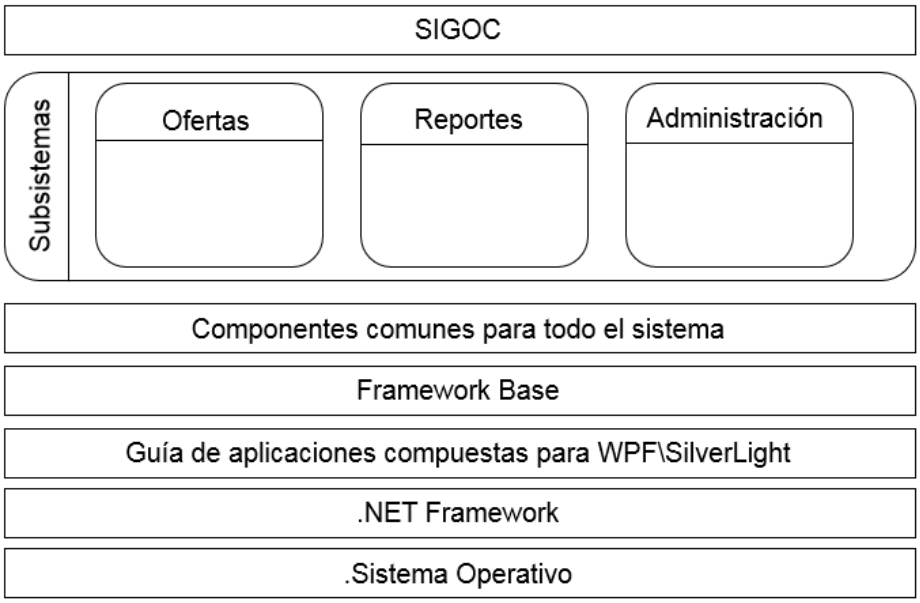


Figura 5: Vista arquitectónica de la solución propuesta

Vista arquitectónica de un subsistema:

La figura muestra cómo sería el diseño de cada uno de los subsistemas que se definen dentro de la aplicación. Resulta válido aclarar en este punto, que los aspectos transversales que se destacan, son a nivel de aplicación. Es decir, que el subsistema en específico no se encarga de manipular la seguridad, la auditoría ni ninguno de estos aspectos, sino que los mismos son tratados a nivel de aplicación. Cada uno de los subsistemas planteados presenta un diseño en capas, donde cada una de estas tiene una función específica.

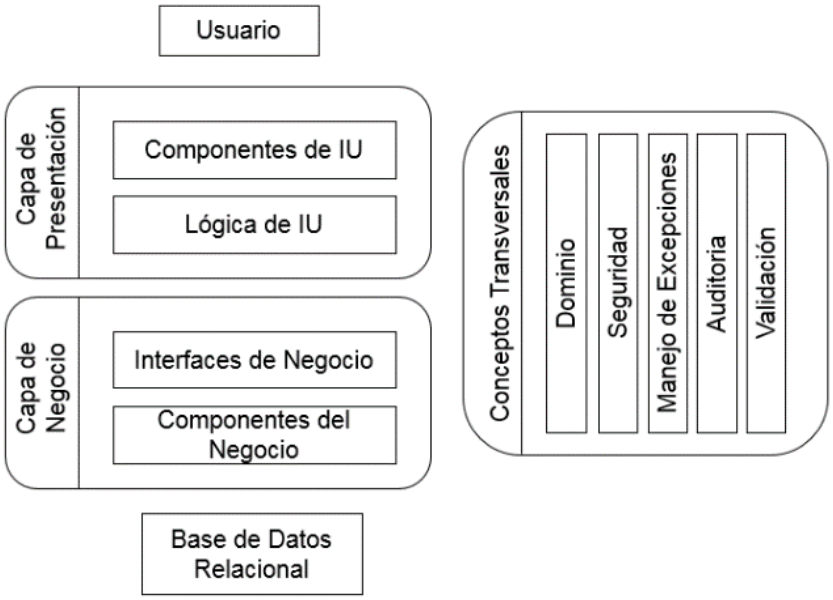


Figura 6: Vista arquitectónica de un subsistema

La aplicación está compuesta por cuatro capas, estas son:

1. **Presentación (Interfaz de Usuario)**

    - La capa presentación hace referencia a todas las interfaces gráficas (UI) que se asocian a cada una de las acciones del sistema. Estas carecen totalmente de funcionamiento, limitándose únicamente a la presentación de la información y la interacción con el usuario final.

    - Toda la lógica asociada a las mismas (buscar datos en la base de datos (BD), guardar una nueva entidad, entre otros) se hacen en el controlador de presentación (acción). Para ello se declaran eventos que son disparados cuando se desea hacer alguna acción concreta, a los cuales se subscribe la acción para realizar toda la lógica necesaria.

    - Estas interfaces no son ventanas propiamente, sino que son controles de usuario los cuales son incrustados en ventanas en el momento de su visualización.

2. **Comandos**

    - El control de la presentación o acciones realiza toda la lógica de la interfaz, dividiéndose estas acciones en cuatro tipos (visuales, no visuales, región y modal) para garantizar un ordenamiento del trabajo con los componentes. De igual manera se pueden hacer uso de *ShortCuts* de forma declarativa que permiten asociar acciones a teclas específicas. Las vistas asociadas a las acciones no tienen lógica ninguna y mediante el uso de *bindings* se puede interactuar entre los controles y propiedades de los objetos.

3. **Acceso a Datos**

    - La capa de acceso a datos es la encargada de recuperar y almacenar información en el servidor de bases de datos. Además de definir la interacción con servicios provistos por sistemas externos a la aplicación. En esta capa se utiliza el patrón DAO con implementación genérica de los métodos comunes (*GetById, Query, Merge, Delete, and Count*).

4. **Dominio**

    - La capa de dominio contiene las clases generadas al realizar el mapeo de las entidades de la base de datos. Estas clases son visibles a todos los subsistemas, módulos y capas de la aplicación. Además, las clases del dominio hacen uso del *Selft-Tracking* para gestionar los cambios en sus propiedades. Cada clase del modelo implementa la interfaz *IObjectWithChangeTracker*, que tiene los métodos necesarios para gestionar los cambios (*Unchanged, Modified, Added o Deleted*), y no solo gestiona los cambios, sino que también mantiene los valores originales.

2.9 Patrones de diseño

Un patrón de diseño es una buena práctica documentada de la solución de un problema que ha sido aplicado satisfactoriamente en múltiples entornos. Es una solución recurrente a un problema común observado o descubierto durante el estudio o construcción de numerosas aplicaciones. Su principal objetivo es incrementar la calidad del *software* en términos de reusabilidad, mantenimiento y extensibilidad (Larman 1999).

En la implementación se decide utilizar como patrones de diseño los siguientes:

1. ***Dependency Injection***

- Es un patrón de diseño orientado a objetos, el cual permite suministrar objetos a una clase en lugar de ser la propia clase quien cree el objeto.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico *Dependency Injection*:

```
namespace Prosoft.Seguridad
{
    public class ProveedorSeguridad : ISecurityProvider
    {
        [Dependency]
        public GenericDao GenericDao { get; set; }

        [Dependency]
        public ProsoftUnitOfWork UnityOfWork { get; set; }
```

Figura 7: Patrón arquitectónico *Dependency Injection*

2. ***Data Access Object***

- Este patrón permite abstraer y encapsular los accesos, gestionar las conexiones a la fuente de datos y obtener los datos almacenados.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico *Data Access Object*:

```
namespace Administracion.AccesoDatos
{
    public class AdministracionDao
    {
        [Dependency]
        public ProsoftUnitOfWork Ctx { get; set; }

        [Dependency]
        public GenericDao GenericDao { get; set; }

        #region Clientes
        public ObservableCollection<Persona> ObtenerPersonas()
        {
            var includes = new List<Expression<Func<Persona, object>>>
            {
                t => t.Cargos,
            };
            return GenericDao.Obtener(includes);
        }
```

Figura 8: Patrón arquitectónico *Data Access Object*

3. *Unit of Work*

- Este patrón mantiene un registro de todos los objetos leídos desde la base de datos, junto con todos los modificados. También maneja cómo las actualizaciones se escriben en la base de datos.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico *Unit of Work*:

```
namespace Ironhide.DataAccess
{
    public class UnitOfWork<T> : IUnitOfWork<T>, IDisposable where T : DbContext, new()
    {
        public UnitOfWork();

        ~UnitOfWork();

        public bool AutoCommit { get; set; }
        public T GetContext { get; }
        protected T Context { get; set; }
        protected IDictionary<Type, ICommitable> CreatedRepositories { get; set; }

        public virtual int CommitChanges(bool autoRollbackOnError = true);
        public virtual Task<int> CommitChangesAsync(bool autoRollbackOnError = true);
        public IGenericRepositoryAsync<TEntity, TKey> DbReporitory<TEntity, TKey>() where TEntity : class;
        public void Dispose();
        public void RollBack();
        public void SetContext(T cont);
        protected virtual Type GetRepositoryType();
    }
}
```

Figura 9: Patrón arquitectónico *Unit of Work*

4. Modelo-Vista-Presentador (MVP)

- Es un patrón arquitectónico que se deriva del Modelo-Vista-Controlador (MVC) por lo que su principal función se refleja en la capa de presentación, logrando una estricta regulación de la interacción entre la vista y el presentador.

- En el MVP la vista y el modelo de datos son separados claramente por medio de una interfaz que muestra la vista, y el presentador tiene acceso de manera polimórfica.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico Modelo-Vista-Presentador (Vista):

```
<a:IronRegionCommandView x:Class="Administracion.Presentacion.NuevoModuloView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:cal="http://www.codeplex.com/CompositeWPF"
             xmlns:a="clr-namespace:Ironhide.Commands;assembly=Ironhide.Commands"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:combobox="clr-namespace:Ironhide.UI.Combobox;assembly=Ironhide.UI"
             Name="NuevoModuloVista"
             mc:Ignorable="d"
             d:DesignHeight="630" Title="Nuevo Módulo" d:DesignWidth="1123">
    <StackPanel Name="HeaderPanel" Margin="0,0,0,0">
        <Grid Name="DatosModulo" Margin="15,10,15,10">
            <Grid.ColumnDefinitions...>
            <Grid.RowDefinitions...>
            <StackPanel Grid.Column="0" Grid.ColumnSpan="3">
                <Label  Content="Nombre"/>
                <TextBox Name="txtNombre" Text="{Binding Path=Nombre, UpdateSourceTrigger=PropertyChanged}" />
            </StackPanel>
            <StackPanel Grid.Column="4" >
                <Label  Content="Precio"/>
                <TextBox Name="txtPrecio" Text="{Binding Path=Precio, UpdateSourceTrigger=PropertyChanged, Str
            </StackPanel>
            <StackPanel Grid.Column="6" >
                <Label  Content="Producto"/>
                <combobox:ComboBox Name="cbProducto" IsRequired="True" SelectedItem="{Binding ElementName=Nuev
                  ItemsSource="{Binding ElementName=NuevoModuloVista, Path=Producto}" />
            </StackPanel>
```

Figura 10: Patrón arquitectónico Modelo-Vista-Presentador (Vista)

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico Modelo-Vista-Presentador (Presentación):

```
namespace Administracion.Presentacion
{
    /// <summary>
    /// Interaction logic for NuevoModuloView.xaml
    /// </summary>
    public partial class NuevoModuloView
    {
        public Producto SelectProducto { get; set; }

        public static readonly DependencyProperty ProductoProperty;
        public ObservableCollection<Producto> Producto
        {
            get { return (ObservableCollection<Producto>)GetValue(ProductoProperty); }
            set { SetValue(ProductoProperty, value); }
        }

        private static readonly DependencyProperty PrecioProperty;
        public double Precio
        {
            get { return (double)GetValue(PrecioProperty); }
            set { SetValue(PrecioProperty, value); }
        }

        static NuevoModuloView()
        {
            ProductoProperty = DependencyProperty.RegisterAttached("Producto", typeof(Obse
            PrecioProperty = DependencyProperty.RegisterAttached("Precio", typeof(double),
        }
```

Figura 11: Patrón arquitectónico Modelo-Vista-Presentador (Presentación)

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico Modelo-Vista-Presentador (Controlador):

```
namespace Administracion.Comandos
{
    [Command("Administracion.NuevoModulo", "Nuevo Modulo")]
    public class NuevoModuloCommand : IronVisualRegionCommand
    {
        [Dependency]
        public NuevoModuloView Vista { get; set; }

        [Dependency]
        public AdministracionDao DaoOferta { get; set; }

        public NuevoModuloCommand(IUnityContainer container)
            : base(container, false, WellKnowNames.RegionDockingShell)
        {
        }

        protected override bool CanExecuteCommand(object parameter)
        {
            return true;
        }

        protected override IronRegionCommandView CreateView()
        {
            if (!Vista.IsOpen)
            {
                Vista.IsOpen = true;
                Vista.Producto = DaoOferta.ObtenerProductos();
            }
            return Vista;
        }
```

Figura 12: Patrón arquitectónico Modelo-Vista-Presentador (Controlador)

5. Command

- Pertenece al grupo de patrones de comportamiento, el mismo define una clase por cada acción que implementa una interfaz común. Brinda la posibilidad a los programadores de encapsular la funcionalidad deseada en un objeto reutilizable.

- Este patrón tiene gran utilidad en las interfaces de usuario, pues es utilizado en el subsistema, a través de los comportamientos de WPF que se agregan a los diferentes componentes, como pueden ser los botones.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico *Command*:

```
namespace Administracion.Comandos
{
    [Command("Administracion.GuardarUsuario", "Guardar un Usuario")]
    public class GuardarUsuarioCommand : IronNonVisualCommand
    {
        [Dependency]
        public NuevoUsuarioView Vista { get; set; }

        [Dependency]
        public GenericDao Dao { get; set; }

        public GuardarUsuarioCommand(IUnityContainer container)
            : base(container)
        {
        }

        private bool _canCommit = true;
        private string MB_Titulo = string.Empty;
        private string MB_Cuerpo = string.Empty;

        protected override bool CanExecuteCommand(object parameter)
        {
            return (Vista.IsActived);
        }

        protected override void Initialize()
        {
            _canCommit = true;
            var dtcontext = SelectedView.Selected.DataContext;
            Save(dtcontext);
        }
```

Figura 13: Patrón arquitectónico *Command*

6. Singleton

- Es clasificado como un patrón de creación. Define un método estático en las clases con el fin de lograr que los objetos tengan un único punto de acceso global al admitirse una sola instancia de una clase.

- Este patrón se utiliza para tener solo una instancia del contenedor de inyección de dependencias en el subsistema. Puede ser empleado, por ejemplo, cuando se crea una acción, que permite especificar que esta sea un *Singleton*, para que cada vez que se invoque a la misma, no se cree una instancia diferente, sino que se almacene en el contenedor de inyección de dependencias, para que pueda ser tomada cada vez que sea necesario utilizarla.

- La siguiente figura muestra un fragmento del código donde se pone de manifiesto el patrón arquitectónico *Singleton*:

```
namespace Administracion.Comandos
{
    [Command("Administracion.NuevoProducto", "Nuevo Producto")]
    public class NuevoProductoCommand : IronVisualRegionCommand
    {
        [Dependency]
        public NuevoProductoView Vista { get; set; }

        [Dependency]
        public GenericDao Dao { get; set; }

        public NuevoProductoCommand(IUnityContainer container)
            : base(container, false, WellKnowNames.RegionDockingShell)
        {
        }

        protected override void Initialize()
        {
            Resource = new Producto();
        }

        protected override bool CanExecuteCommand(object parameter)
        {
            return true;
        }
```

Figura 14: Patrón arquitectónico *Singleton*

2.10 Conclusiones del capítulo

Este capítulo muestra el modelo de dominio correspondiente a la solución propuesta, especificando los conceptos esenciales del negocio, se logra un enfoque detallado del medio donde se desenvuelve el problema a resolver. Se define los requisitos funcionales y no funcionales que debe tener en cuenta la aplicación para su buen funcionamiento. Se crea el diagrama de casos de uso del sistema que posibilita la comprensión del proceso en cuestión. Se realiza los flujos básicos para los casos de uso explicando minuciosamente las características de estos. Se realiza el diagrama de clases de la solución propuesta para representar los atributos que tienen las clases y generar el diagrama entidad relación para representar la estructura que debe tener la base de datos. Se identifica la arquitectura en capa debido a la posibilidad de que varios programas corran paralelamente. La selección de los patrones de diseño concernientes a la descripción e implementación de la aplicación, sirve para mantener un orden lógico entre las clases; que trajo consigo la organización del desarrollo del sistema.

Capítulo 3: Implementación y validación de la solución propuesta

En este capítulo se explica a partir de los resultados del diseño, la implementación de la aplicación. Se define los estándares de codificación. Como parte de la implementación se muestra el diagrama de despliegue, en el que se visualiza la situación física del sistema. Se especifican las pruebas de *software* y los resultados obtenidos de las mismas. Se realiza la validación de la solución propuesta.

3.1 Estándares de codificación

Los estándares de código se refieren a un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación. Los estándares de programación son frecuentemente dependientes del lenguaje de programación que se haya elegido para escribir. Usar técnicas de codificación sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para la calidad del *software* y para obtener un buen rendimiento.

Los estándares de codificación se especifican con el fin de explicar el código referente a la aplicación por otros desarrolladores y alcanzar una uniformidad en el mismo.

A continuación, se listan los elementos que pertenecen al estándar de codificación definido para el desarrollo de la aplicación:

- Dejar espacios dentro del código, para que se pueda comprender fácilmente.
- Los nombres de las variables deben ser significativos.
- Los nombres de las clases y atributos están escritos en letra inicial mayúscula y sin caracteres especiales.

3.2 Diagrama de despliegue

Los diagramas de despliegue UML muestran cómo los componentes de *software* se despliegan físicamente en los procesadores; es decir, el diagrama de despliegue muestra el *hardware* y el *software* en el sistema, así como el *middleware* usado para conectar los diferentes componentes en el sistema. En esencia, los

diagramas de despliegue se pueden considerar como una forma de definir y documentar el entorno objetivo (Sommerville 2005).

La siguiente figura muestra el diagrama de despliegue de la solución propuesta:

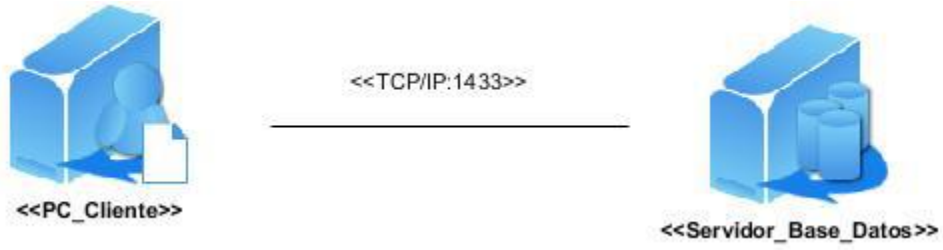


Figura 15: Diagrama de despliegue

A continuación, se puntualizan las características de los nodos:

- **Nodo PC_Cliente:** simboliza el ordenador que emplea el usuario para interactuar con el sistema.
- **Nodo Servidor_Base_Datos:** simboliza el servidor SQL donde se almacenan los datos del sistema.

En el diagrama de despliegue presentado anteriormente se utiliza la familia de protocolos de comunicación TCP/IP, por el puerto 1433, que se emplea para conectar computadoras que utilizan sistemas operativos similares o diferentes, minicomputadoras y computadoras centrales sobre redes de área local (LAN). En el caso de la aplicación se utiliza para interconectarla con el servidor de base de datos.

3.3 Pruebas de *software*

Las pruebas intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el *software*, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa (Sommerville 2005).

Un instrumento adecuado para determinar el status de la calidad de un producto *software* es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del *software* o al sistema de

software en su totalidad. En las pruebas se usan casos de prueba, especificados de forma estructurada mediante técnicas de prueba («Pruebas de Software» 2018).

La prueba de *software* es imprescindible para garantizar la calidad de la aplicación representando una revisión final de las especificaciones del diseño y del código («Pruebas de Software» 2018).

Elementos a tener en cuenta para que una prueba tenga éxito («Pruebas de Software» 2018):

- Estrategia de prueba.
- Niveles de prueba.
- Tipo de prueba.
- Método de prueba.
- Casos de prueba.

Estrategia de prueba:

- Describe el enfoque y los objetivos generales de las actividades de prueba («Pruebas de Software» 2018).
- Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos («Pruebas de Software» 2018).
- Define:
    - Técnicas de pruebas (manual o automática) y herramientas a ser usadas («Pruebas de Software» 2018).
    - Criterios de éxitos y culminación de las pruebas («Pruebas de Software» 2018).
    - Consideraciones especiales relacionadas con los recursos necesarios para realizar las pruebas («Pruebas de Software» 2018).

Una estrategia de prueba del *software* integra los métodos de diseño de casos de pruebas del *software* en una serie bien planeada de pasos que desembocará en la eficaz construcción del *software*. La estrategia proporciona un mapa que describe los pasos que se darán como parte de la prueba, indica cuando se planean y cuando se dan estos pasos, además de cuanto esfuerzo, tiempo y recursos consumirán. Por tanto, cualquier estrategia de prueba debe incorporar la planeación de pruebas, el diseño de casos de pruebas, la ejecución de pruebas y la recolección y evaluación de los datos resultantes (Sommerville 2005).

Una estrategia de prueba del *software* debe ser lo suficientemente flexible como para promover un enfoque personalizado. Al mismo tiempo, debe ser lo adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del proyecto (Sommerville 2005).

Niveles de prueba:

La prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se distinguen los siguientes niveles de pruebas (Jorrin 2007):

- Pruebas unitarias.
- Prueba de integración.
- Prueba de sistema.
- Prueba de implantación.
- Prueba de aceptación.
- Prueba de regresión.

Tipos de prueba:

Los tipos de pruebas de *software* son los siguientes (Jorrin 2007):

- Funcionalidad (función, seguridad, volumen, usabilidad).
- Fiabilidad (integridad, estructura, estrés).

- Rendimiento (*benchmark*, contención, carga, *performance profile*).
- Soportabilidad (configuración, instalación).

Las pruebas funcionales se centran en validar la correcta implementación de las necesidades del cliente. La funcionalidad puede ser relacionada a los datos de entrada y de salida. Los datos de entrada serán realizados y revelarán un resultado y este será comprobado con el resultado deseado (comportamiento).

Las pruebas de función afirman la labor adecuada de los requisitos funcionales, contienen la navegación, entrada de datos, procesamiento y obtención de resultados. Estas pruebas utilizan el método de prueba Caja Negra.

Las pruebas de aceptación son realizadas con el cliente y define su aceptación del sistema. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, sino una vez pasadas todas las pruebas por parte del desarrollador. Pueden realizarse durante un período de semanas o meses, y mediante ellas se descubren errores acumulados que, con el tiempo, puedan degradar el sistema. La mayoría de los constructores de productos de *software* usan un proceso llamado prueba alfa y prueba beta para descubrir errores que al parecer solo el usuario final es capaz de encontrar (Pressman 2010).

La prueba alfa se lleva a cabo en el sitio del desarrollador por un grupo representativo de usuarios finales. El *software* se usa en un escenario natural con el desarrollador, se realizan en un ambiente controlado (Pressman 2010).

La prueba beta se realiza en uno o más sitios del usuario final. A diferencia de la prueba alfa, por lo general el desarrollador no está presente. El cliente registra todos los problemas (reales o imaginarios) que se encuentran durante la prueba beta y los reporta al desarrollador periódicamente. En ocasiones se realiza una variación de la prueba beta, llamada prueba de aceptación del cliente, cuando el *software* se entrega a un cliente bajo contrato. El cliente realiza una serie de pruebas específicas con la intención de descubrir errores antes de aceptar el *software* del desarrollador (Pressman 2010).

Método de prueba (Caja Negra):

Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requisitos funcionales del *software*; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requisitos funcionales para un programa (Sommerville 2005).

Se basa en técnicas como:

- **Técnica de partición de equivalencia:** es un método de prueba de caja negra que divide el dominio de entrada de un programa en clases de datos de los que pueden derivarse casos de prueba. Un caso de prueba ideal descubre de primera mano una clase de errores que de otro modo podrían requerir la ejecución de muchos casos de prueba antes de observar el error general (Jorrin 2007).

- **Técnica de análisis de valores límites:** esta técnica, prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables (Jorrin 2007).

- **Técnica de grafos de causa-efecto:** es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones (Jorrin 2007).

Para validar la solución propuesta se emplea el método de prueba Caja Negra, utilizando la técnica de partición de equivalencia para la elaboración de los casos de pruebas, esta técnica permite inspeccionar los datos válidos e inválidos de las variables de entradas, demostrando que los datos entrados funcionan correctamente. Se realiza una prueba del sistema, específicamente prueba de funcionalidad y dentro de esta, la prueba de función. También se realiza la prueba de aceptación del sistema, particularmente la prueba alfa, por el usuario final.

Se realiza la prueba de funcionalidad porque se emplea una arquitectura que no pertenece a ningún sistema de los que se están desarrollando en el centro actualmente, probada y validada, está realizada con patrones bien específicos y bien implementados, donde cada vista se puede interpretar fácilmente y no existe una acción interna en ninguna de las interfaces de la solución.

Casos de prueba:

Los casos de prueba deben diseñarse para descubrir errores debidos a cálculos erróneos, comparaciones incorrectas o flujo de control inadecuado. Se diseñan para garantizar que: se satisfagan todos los requisitos de funcionamiento, se logran todas las características de comportamiento, todo el contenido es preciso y se presenta de manera adecuada, se logran todos los requisitos de rendimiento, la documentación es correcta y se satisfacen la facilidad de uso y otros requisitos (Sommerville 2005).

A continuación, se muestra el caso de prueba al CU1.Autenticar usuario para el método de prueba Caja Negra. El resto de los casos de prueba y la descripción de las variables se muestran en el anexo 3.

Tabla 5: Método de prueba Caja Negra al CU1.Autenticar usuario

Escenario	Descripción	Variables		Respuesta del sistema	Flujo central
		Usuario	Contraseña		
EC 1.1 Autenticar usuario	El usuario introduce su user_name y contraseña para acceder al sistema	V	V	Valida los datos y se accede al sistema satisfactoriamente	1. El usuario introduce su user_name y contraseña para poder acceder a la aplicación, selecciona el botón “Aceptar” 2. Valida los datos
EC 1.2 Campos vacíos	Se valida que no existan campos obligatorios vacíos	V	I	Muestra el mensaje de información del error	1. Introduce las credenciales y deja un campo vacío 2. Muestra el mensaje de información “El campo no puede ser nulo”
		I	V		
EC 1.3 Datos incorrectos	Se valida que no existan datos incorrectos	V	I	Muestra el mensaje de información del error	1. Introduce las credenciales y un dato es incorrecto 2. Muestra el mensaje de información “Usted no tiene los permisos para acceder al sistema”
		I	V		

					3. Muestra el mensaje de información “Contraseña incorrecta”

3.4 Resultado de la prueba aplicada a SIGOC

El siguiente gráfico de barras evidencia el resultado de la prueba de funcionalidad realizada al sistema, donde se muestra el total de no conformidades encontradas y las que fueron resueltas por cada iteración.

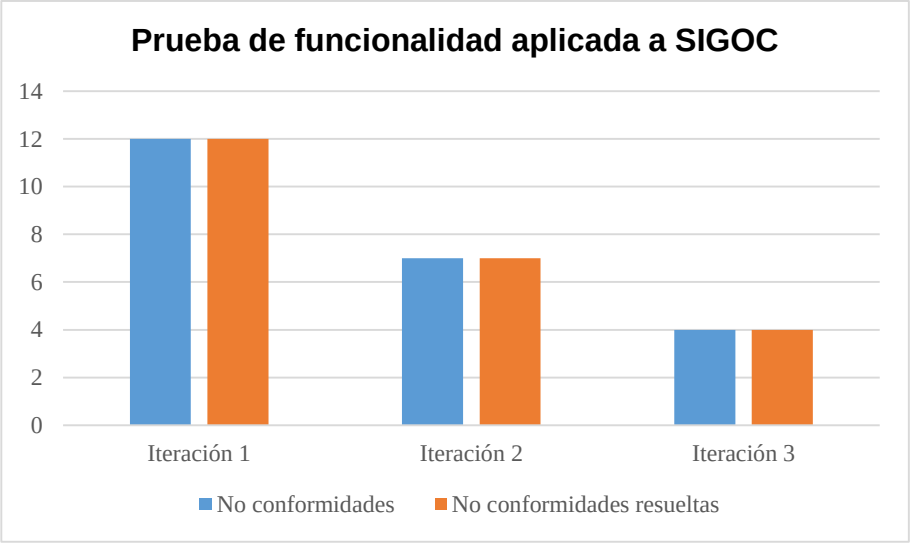


Figura 16: Prueba de funcionalidad aplicada a SIGOC

En la primera iteración se detectó un total de 12 no conformidades y todas fueron resueltas. En la segunda iteración se detectó un total de 7 no conformidades y fueron resueltas todas. En la tercera iteración se detectó un total de 4 no conformidades y fueron resueltas en su totalidad. Se demuestra al culminar esta prueba, que los errores percibidos fueron resueltos totalmente.

En la siguiente tabla se especifica la descripción de las no conformidades por las distintas iteraciones, realizadas al sistema en la prueba de funcionalidad.

Tabla 6: Descripción de las no conformidades por iteración

		Iteraciones		
No.	**Descripción de las no conformidades**	**I1**	**I2**	**I3**
1.	Error ortográfico.	6	3	2
2.	Error de validación en los datos de entrada.	4	2	1
3.	Error de concordancia entre la aplicación y el diseño de los casos de prueba.	2	2	1

La siguiente figura muestra el acta de la prueba de aceptación realizada a SIGOC por el usuario final.

UCI Universidad de las Ciencias Informáticas

Acta de aceptación

En cumplimiento del trabajo de diploma **"Herramienta de gestión de oferta para el centro CIGED"** se hace entrega de los productos que se relacionan a continuación:

- Código de la herramienta.
- Artefactos generados durante las fases como parte de la metodología AUP.

El director del centro CIGED, luego de haber revisado los productos del trabajo de diploma determina que se **aceptan** y que la herramienta cumple con los requisitos definidos y con la calidad necesaria para ser integrada al centro.

Entrega	**Recibe**
Nombre y apellidos: Suri Nancy Mustelier Villalón	**Nombre y apellidos:** Aurelio Antelo Collado
Cargo: Autora del trabajo de diploma	**Cargo:** Director del centro CIGED
Firma:	**Firma:**

CENTRO DE INFORMATIZACIÓN DE LA GESTIÓN DOCUMENTAL

Fecha: 14 de junio de 2018

Figura 17: Acta de la prueba de aceptación realizada a SIGOC

3.5 Conclusiones del capítulo

Este capítulo explica a partir de los resultados del diseño, la implementación de la aplicación SIGOC que favorece la fijación de precio de los productos informáticos del centro. Se define además, los estándares de codificación y las pruebas de *software* que posibilita verificar la implementación del sistema, se obtienen resultados satisfactorios, debido a que los errores detectados fueron corregidos en su totalidad. Se realiza el diagrama de despliegue, que permite modelar el *hardware* empleado en la implementación de la solución propuesta y sirve para comprender la integración física de los componentes de *hardware* y *software*, se demuestra la integración de la PC_Cliente y el servidor de base de datos, mediante la familia de protocolos de comunicación TCP/IP usando el puerto 1433.

Conclusión general

Este trabajo de diploma culmina con el cumplimiento del objetivo general propuesto desarrollándose una aplicación que permite fijar el precio de los productos informáticos de CIGED. A modo general se alcanzó los resultados siguientes:

1. Se seleccionó como método de fijación de precio, la fijación de precio mediante márgenes, empleando un margen de utilidad.

2. Se seleccionó la estrategia de venta para los productos del centro, empleando la venta personal debido a que se logra la comunicación directa con el cliente.

3. El diseño de la solución propuesta facilitó tener una idea precisa de lo que se debía implementar y se definió las funcionalidades que tenía que cumplir la herramienta.

4. La implementación de la solución propuesta favoreció al proceso de fijación de precio de los productos informáticos del centro, dando solución a los problemas descritos en el marco teórico de la investigación.

5. Se validó la solución propuesta usando el método de prueba Caja Negra, lo que permitió identificar las no conformidades existentes en la aplicación y su posterior corrección. Además se efectuó la prueba de aceptación del sistema, particularmente la prueba alfa, realizada por el cliente.

Recomendaciones

Este trabajo de diploma culmina de forma satisfactoria, cumpliendo con el objetivo planteado, con el fin de favorecer aún más la investigación se realizan las recomendaciones siguientes:

1. Desplegar la herramienta propuesta en CIGED, con el objetivo de mejorar el proceso de fijación de precio de los productos informáticos que se comercializan.

2. Integrar esta herramienta en otros centros productivos de la universidad, para contribuir al proceso de fijación de precio de los productos o servicios que comercialicen.

Referencias bibliográficas

GONZÁLEZ, Patricia. Propuesta de un modelo para medir activos intangibles en empresas de software a partir de una herramienta multicriterio. 2015. https://www.sciencedirect.com/science/article/pii/S0123592315000029

INFORMÁTICAS, Universidad de las Ciencias. *Historia | Universidad de las Ciencias Informáticas*. 2017. http://www.uci.cu/universidad/historia/

INFORMÁTICAS, Universidad de las Ciencias. *Misión | Universidad de las Ciencias Informáticas*. 2017. http://www.uci.cu/universidad/mision

INFORMÁTICAS, Universidad de las Ciencias. *Centros de Desarrollo | Universidad de las Ciencias Informáticas*. 2017. http://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo

INFORMÁTICAS, Universidad de las Ciencias. *Centro de Informatización de la Gestión Documental (CIGED) | Universidad de las Ciencias Informáticas*. 2017. http://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo/centro-de-informatizacion-de-la-gestion-documental

GARCÍA, Patricia. *Venta de productos tangibles e intangibles*. 2017. http://www.escuelanacionaldeventas.com/url2017/index.php?option=com_content&view=article&id=169:=87&Itemid=437&lang=en

IASC, Fundación. NIC 38 activos intangibles. 2009. http://www.nicniif.org/files/u1/IAS38.pdf

RUZ, Agustín Moreno. El valor contable de los activos intangibles. 2012. https://eciencia.urjc.es/bitstream/handle/10115/12293/Tesis Agust%EDn Moreno Ruz.pdf;jsessionid=8EC318D12491B178ACC2B8926675BF74?sequence=1

DÍAZ-ESPINA, Carolina. Modelos de negocio y medios online. Aproximación teórica a la cuestión. 2013. http://www.redalyc.org/pdf/1995/199527531048.pdf

AUTORES, Colectivo de. *Definición de Fijación de precios _ Que es, Conceptos y Significados*. 2014. http://definicionyque.es/fijacion-de-precios/

CASADO, Juan Carlos Alcalde. *La estrategia de ventas y el valor de vida de los clientes.* 2012. https://gestion.com.do/ediciones/enero-2012/item/259-la-estrategia-de-ventas-y-el-valor-de-vida-de-los-clientes

OSTERWALDER, Alexander y PIGNEUR, Yves. Generación de modelos de negocio. 2016. http://www.convergenciamultimedial.com/landau/documentos/bibliografia-2016/osterwalder.pdf

AUTORES, Colectivo de. *9 elementos de un modelo de negocio*. 2016. http://mprende.co/mercado/9-elementos-de-un-modelo-de-negocio

MEJÍA, Carlos Alberto. Métodos para la determinación del precio. 2012. http://www.planning.com.co/bd/mercadeo_eficaz/Agosto2005.pdf

AUTORES, Colectivo de. *¿Cómo calcular el margen de utilidad?* 2015. http://destinonegocio.com/pe/economia-

pe/aprende-a-calcular-el-margen-de-ganancia-de-tu-negocio/

GMAC, Coleman V. Margen de utilidad. 2015. https://www.nclc.org/images/pdf/litigation/closed/gmac-faq-spanish.pdf

NAVARRO MEJÍA, Mariana Elizabeth. *Técnicas de ventas*. 2012. https://www.aliat.org.mx/BibliotecasDigitales/economico/Tecnicas_de_venta.pdf

RODRÍGUEZ, Tamara. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2014.

SANTAMARÍA, Jose y HERNÁNDEZ, Javier. Microsoft SQL Server. 2015. https://iessanvicente.com/colaboraciones/sqlserver.pdf

MORALES, Carlos Eduardo Ospina. Análisis, diseño, desarrollo, pruebas y despliegue de software con los estándares de calidad, proceso y tecnologías usadas en Pragma S.A. 2012. https://repository.lasallista.edu.co/dspace/.../Metodologia_desarrollo_software_Pragma.pdf

SECO, José Antonio González. *El lenguaje de programación C#*. 2000. https://users.dsic.upv.es/~jlinares/csharp/lenguajeCsharp.pdf

AUTORES, Colectivo de. Lenguaje Unificado de Modelado (UML). [online]. 2016. Available from: https://www.lucidchart.com/pages/es/qué-es-el-lenguaje-unificado-de-modelado-uml

AUTORES, Colectivo de. Visual Paradigm. [online]. 2016. Available from: http://www.visual-paradigm.com

SOMMERVILLE, Ian. *Ingeniería del software Séptima edición*. 2005. http://zeus.inf.ucv.cl/~bcrawford/EnfoquesDeDesarrolloDeSwYLenguajesDeModelado/Ingenieria del Software 7ma. Ed. - Ian Sommerville.pdf

LARMAN, Craig. *UML y patrones. Introducción al análisis y diseño orientado a objetos*. 1999.

PRESSMAN, Roger S. *Ingeniería del software. Un enfoque práctico*. 2010. http://cotana.informatica.edu.bo/downloads/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

LEÓN, Eduardo. Tutorial Visual Paradigm for UML. 2000. http://www.slion2000.blogspot.com

REYNOSO, Carlos y KICCILLOF, Nicolás. Estilos y patrones en la estrategia de arquitectura de Microsoft. 2004. http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:estiloypatron.pdf%0A%0A

TRELLINI, Ariel. Estilos y patrones arquitectónicos. 2015. http://cs.uns.edu.ar/~atrellini/ayds/downloads/Clases/05.%20AyDS%20-%20Estilos%20y%20Patrones%20Arquitectonicos%20-1.pdf

Pruebas de Software. [online]. 2018. Available from: http://www.pruebasdesoftware.com/laspruebasdesoftware.htm

JORRIN, Ing. Michaell González. Proceso de pruebas para la liberación de productos software. 2007.