UNIVERSIDAD DE HOLGUÍN
Facultad de Informática y Matemática

SISTEMA INFORMÁTICO PARA LA GESTIÓN COMERCIAL EN EL SECTOR ESTATAL EN LA EMPRESA DE ACUEDUCTO Y ALCANTARILLADO GUARDALAVACA

Trabajo de diploma en opción al título de Ingeniero Informático

Autor: Raúl Eliecer Botello Desdín

Tutores: Ing. Leandro Osorio Gámez
Lic. Yunieskis Ramayo Caldas

Holguín, 2014

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Informática de la Universidad de Holguín “Oscar Lucero Moya” para que hagan el uso que estimen pertinente con este trabajo.

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

_____

Nombre completo del autor

_____ _____

Nombre completo del tutor Nombre completo del tutor

A:

Mis padres, por su dedicación y enseñanza. Por su presencia a pesar de las dificultades. Por ser la mano que te pone en pie después de una caída.

Mi hermano y a mi sobrina que hacen que mi vida sea más feliz.

Mis abuelos por apoyarme siempre y por ser mi vida.

Mi familia que tanto me apoya en las decisiones que tomo a diario.

Mis amigos y compañeros, antiguos y nuevos.

A mami y papi, por su apoyo incondicional y por enseñarme, pese a las circunstancias que sean, a crecer como persona y como profesional.

A mi hermano por darme su apoyo incondicional, consejos y permitirme disfrutar momentos inolvidables juntos.

A mi abuela en especial por ser mi brazo de apoyo en los momentos difíciles y ser la persona que más quiero.

A mis tíos, primos en general a toda la familia.

A mis tutores por todo su tiempo y dedicación para conmigo, gracias por confiar en mí y tenerme en cuenta para esta investigación, mil gracias.

A todos mis amigos, pues son quienes me soportan, en especial a los cómplices de numerosas conversaciones, intercambios de filosofías de vidas, consejos, tertulias, fiestas, en fin, de llenar los espacios vacíos; a los cuales puedo llamar hermanos sin ningún tipo de problemas: Angel Mederos, Carlos, Luis Ernesto, Manolo, Oscarito, Camilo, Fernando, Héctor, Gustavo y Luisma.

A mis compañeros de clases, los cuales hacían de turnos tediosos momentos inolvidables, con nuestras virtudes y defectos funcionábamos bien, no todos tienen la suerte de tener un grupo de compañeros así. Creo que esa añoranza post graduación, mayormente está propiciada porque quizás nunca vuelva a ver a algunos de ustedes, no obstante, creo que inmortalizamos numerosos momentos: no los OLVIDARÉ NUNCA.

A mis amistades por su preocupación y su ayuda.

A los profesores de la carrera por su educación y maravillosa enseñanza durante estos cinco años.

A todas las personas que contribuyeron para que este maravilloso sueño se hiciera realidad.

A todos muchas gracias.

Resumen

En la actualidad, la necesidad de vincular la informática a los procesos laborales ha ido en aumento y con el desarrollo de esta ciencia, surgen muchas alternativas para lograrlo. La presente investigación se desarrolla en la Empresa de Acueducto y Alcantarillado Guardalavaca, encargada de brindar servicios de abastecimiento de agua, alcantarillado y depuración de los residuales líquidos al polo turístico y al sector residencial y estatal del norte de la provincia de Holguín. Dentro de los procesos que esta incorpora se encuentra la gestión comercial que juega un papel fundamental en el funcionamiento de dicha entidad, que tiene como objetivo establecer el modo de realizar y controlar los procesos de contratación, verificación de consumos, facturación y cobro a los clientes, de forma que garantice que se realicen con la máxima calidad y eficacia. Con el análisis de este proceso se detectó un conjunto de problemas referentes a la interacción con la información generada y su integridad. Para solucionar estos problemas se ha planteado la construcción de un sistema informático para la gestión comercial en el sector estatal que lleve a cabo este proceso, brindando un mayor flujo de información en tiempos más cortos y fiabilidad en los datos. En este documento se presenta un resumen del estudio bibliográfico realizado, así como la metodología de Ingeniería de *Software* que se siguió para el diseño y construcción de la solución que se propone. Además, se describe la implementación del sistema propuesto y la validación parcial de los resultados.

Abstract

At present, the increase of the necessity to link computer science to the labor processes is exponential and with the development of this science much alternatives have come about to achieve this integration. The present investigation which is being developed at the Guardalavaca Aqueduct and Sewage Company, this company is in charge of rendering services of water supply, drainage and depuration of liquid residues at this Touristic Pole and to the state and resident sectors to the north of the province of Holguin. Within the processes offered by this entity, commercial management plays an important fundamental role in entity's functioning, it has as its objective the establishment of a mode to realize and control the contracted processes, the verification of consumptions, invoicing and the receipt of payment by clients from a form that guarantees that they are realized with maximum quality and efficiency. With the analysis of this process a series of problems were detected with reference to the interaction of the information generated and its integrity. In order to solve these problems, a proposal was made to implement a computerized system for commercial management in the state sector which would carry out this process, rendering s a better flow of information at a shorter time and giving reliability to the data. The present document presents a summary of the bibliographic study, as well as the methodology of software engineering which was used to design and construct the proposed solution. Besides this, a detailed description of the systems implementation is given as well a partial validation of the results obtained.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

A través del curso de la historia el hombre se ha visto envuelto en la necesidad de crear herramientas que lo ayuden en la resolución de problemas originados a su alrededor o en el correcto desenvolvimiento de la actividad social que realiza; todo enfocado en la búsqueda de mejores soluciones que le permitan desarrollar mejor su trabajo y mantener un mejor control de las tareas vinculadas a su entorno.

Con el surgimiento de la Informática y las Comunicaciones, surgieron un conjunto de tecnologías que sirvieron de base a disímiles de personas en las cuales podían apoyarse para la organización y gestión de la información, automatización de procesos y manipulación de datos. Durante varias décadas se han descubierto miles de áreas donde estas pueden ser aplicadas, haciendo cada vez mayor la esfera en la cual esta joven ciencia se desarrolla.

Hoy día es un reto para las diferentes empresas ofrecer productos y/o servicios de calidad que satisfagan adecuadamente las necesidades que exigen los clientes, esto lleva a las diferentes organizaciones a implementar medidas que las ayuden a desarrollar niveles de eficiencia. Una estrategia es desarrollar un sistema informático para la gestión comercial que les permita coordinar, mejorar los procesos y procedimientos que se desarrollan dentro de la organización y optimizar así los recursos, productos y/o servicios. La importancia de contar con el *software* apropiado es crucial para el éxito de una empresa en términos de mantener el dinamismo de la cadena de oferta y demanda.

La Empresa de Acueducto y Alcantarillado Guardalavaca se encuentra en Perfeccionamiento Empresarial, se integra al Grupo Empresarial Acueducto y Alcantarillado (GEAAL) y se subordina al Instituto Nacional de Recursos Hidráulicos (INRH). Se crea el 19 de enero de 1999 como empresa, con personalidad jurídica independiente y patrimonio propio. Posee su domicilio legal en el barrio El Progreso s/n, playa Guardalavaca en el municipio de Banes de la provincia de Holguín. Implanta el Perfeccionamiento Empresarial a partir del año 2002 por acuerdo No 4753 del Comité Ejecutivo del Consejo de Ministros de fecha 27 de marzo de 2002, en la actualidad se encuentra en el noveno paso del perfeccionamiento, además cuenta con un Sistema de

Gestión de la Calidad implantado por la NC ISO 9001 Avalado desde el año 2010. Tiene como objetivo fundamental brindar servicios de abastecimiento de agua, alcantarillado y depuración y disposición final de los residuales líquidos al polo turístico y al sector residencial y estatal del norte de la provincia de Holguín.

En estos momentos la empresa se traza nuevas metas para mejorar la gestión de información, con la idea de centralizar el proceso de gestión comercial y agilizar el flujo de información, en vista a generar un aumento de la calidad y productividad del trabajo en dicha área.

En esta empresa parte de los procesos se efectúan manualmente, como es el caso del control, registro y gestión de los contratos en el sector estatal, la verificación de consumos, la facturación y el cobro a los clientes, conllevando a la pérdida de información, gastos de innumerables recursos; además de ser un trabajo tedioso, lo que provoca un retraso en el mismo, lo cual lleva a un mal funcionamiento del proceso de gestión comercial haciéndolo lento e ineficiente. También trae consigo que sea ineficiente a la hora de realizar búsquedas y al procesar la información, así como la inexistencia de una base de datos centralizada. Otras dificultades que se ponen de manifiesto es la complejidad en la revisión y consulta de información específica; la protección de los documentos no es segura y no brinda posibilidades de actualización. Ante estos problemas y en busca de una solución a ella, la empresa se ve en la necesidad de solicitar la confección de un sistema informático para el control del proceso de gestión comercial.

Considerando la importancia, actualidad del tema y la necesidad que tiene la Empresa de Acueducto y Alcantarillado Guardalavaca de implantar un sistema informático que les permita gestionar todo lo referido a los clientes del sector estatal. Tomando además las posibilidades que existen de generalizar la aplicación gracias a la inexistencia y pobre estandarización de este tipo de soluciones, que si bien existen son aplicaciones de escritorio confeccionadas en *Access* o libros de *Excel*, las que además de restringir la capacidad de análisis sobre toda la información desaprovecha las facilidades tecnológicas de conectividad de las que dispone la Empresa de Acueducto y Alcantarillado

Guardalavaca, encontrándose conectada en red mediante la Línea de Abonado Digital Asimétrica (ADSL).

Partiendo de las cuestiones antes mencionadas se identifica el siguiente **problema científico**: ¿Cómo favorecer el proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca?

A partir del problema se delimita el **objeto de investigación científica** que se corresponde con: el proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.

Para solucionar el problema se persigue el siguiente **objetivo** de investigación: Desarrollar un sistema informático para favorecer el proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.

El objetivo de la investigación delimita el **campo de acción**: Informatización del proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.

Para guiar la investigación se plantea la siguiente **hipótesis**:

Dotar a la Empresa de Acueducto y Alcantarillado Guardalavaca de un sistema informático, que agilice la búsqueda y obtención de datos; y a su vez centralice la información del proceso de gestión comercial; favorecerá la integridad de la información y el desempeño del proceso comercial en el área estatal de dicha entidad.

Para cumplir el objetivo, la investigación transcurrirá según las siguientes **tareas**:

1. Estudiar los fundamentos teóricos que sustentan el proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.
2. Diagnosticar el estado actual del proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.
3. Diseñar el sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.

4. Realizar el estudio de la valoración de sostenibilidad del producto informático según las dimensiones administrativa, socio-humanista, ambiental y tecnológica
5. Implementar el sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.
6. Evaluar la satisfacción de los usuarios con respecto al producto informático.

Para darles solución a las tareas planteadas se utilizaron los siguientes **métodos científicos**:

Métodos Teóricos

Análisis y síntesis: Se utilizó con el fin de analizar la información manejada durante los procesos de contratación, verificación de consumos, facturación y cobro a los clientes; además de elaborar los fundamentos teóricos; así como descomponer las necesidades en requerimientos del sistema. Para facilitar su comprensión, a través de este método, se realizó la valoración de sostenibilidad del sistema propuesto como solución.

Histórico - Lógico: se llevó a cabo un análisis del proceso de gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca, para conocer con mayor profundidad los antecedentes y las tendencias actuales. Así como comprender la lógica del negocio y las normas que rigen su funcionamiento.

Método de modelación: este método se utilizó durante el proceso de elaboración del sistema, pues a través de la metodología de ingeniería de *software* se logró con el conjunto de modelos que se describieran todas las perspectivas posibles del proceso de desarrollo en sentido general, y que el sistema se pudiese modelar de una forma menos abstracta y con mayor comprensión.

Hipotético - Deductivo: Fue utilizado en la elaboración de la hipótesis, la cual será analizada y demostrada por la investigación realizada, lo que permitirá poder implementar una solución al problema planteado.

Enfoque Sistémico: Fue utilizado para identificar y descomponer el sistema en subsistemas, así como las relaciones entre ellos, facilitando además organizar el trabajo y la lógica del negocio identificada.

Métodos Empíricos

Entrevista: se empleó para identificar los requerimientos del sistema, además se recogieron las deficiencias existentes y las expectativas del cliente con respecto a la aplicación, para entender los procesos que se llevan a cabo en la gestión comercial.

Revisión de documentos: Fue utilizado para la recopilación de la información necesaria a esbozar en el marco teórico, además de la que será almacenada y controlada en la aplicación.

Observación: Se llevó a cabo en diferentes períodos, puesto que permitió realizar un análisis detallado de las condiciones de los medios informáticos con que cuenta la Dirección Comercial de la Empresa de Acueducto y Alcantarillado Guardalavaca, para valorar el nivel de eficiencia y respuesta a las peticiones de los usuarios que deberá tener el sistema acorde con las condiciones reales y a partir de esto se estableció la problemática a investigar.

Encuestas: Fueron desarrolladas para elegir los expertos y obtener valoraciones conclusivas de estos con la implantación del sistema informático desarrollado como solución al problema planteado.

Criterio de expertos: se utiliza para llegar a un consenso entre los expertos valorando así el sistema informático resultante, para procesar estas encuestas se usará el método *Delphi.*

Métodos estadísticos

Para el tratamiento de las encuestas se hizo uso de métodos estadísticos como el trabajo con porcentajes y la frecuencia, y el Método *Delphi* para buscar el consenso de los encuestados.

Con este trabajo se dispondrá de una herramienta informática que garantizará que la información relacionada con el proceso de gestión comercial que se lleva a cabo por la Empresa de Acueducto y Alcantarillado Guardalavaca transite eficientemente por las vías correspondientes en sus diferentes etapas. Con la investigación se eliminará totalmente el trabajo manual que se hace

actualmente, almacenará un gran volumen de información, agilizará las búsquedas de los documentos que se encuentran inmersos en el proceso antes mencionado y se generarán los informes que debe entregarse cada cierto tiempo.

Con el objetivo de exponer los resultados de la investigación se ha dividido el presente documento en introducción, dos capítulos, conclusiones, recomendaciones, bibliografía y anexos.

El **Capítulo 1** (Fundamentos teóricos) está orientado a mostrar los principales fundamentos, conceptos, metodologías y herramientas utilizados para llevar a cabo el desarrollo de la investigación.

En el **Capítulo 2** (Descripción de la solución propuesta) se muestran los flujos de trabajo principales de la ingeniería de *software* realizada al sistema propuesto, a través de la metodología *ICONIX* y se incluye un estudio de factibilidad y una valoración de sostenibilidad del sistema como producto informático, según las dimensiones administrativa, socio-humanista, ambiental y tecnológica.

Capítulo 1. Fundamentos Teóricos

En el presente capítulo se muestra un análisis teórico de los principales conceptos asociados al problema, con la finalidad de comprender con precisión el objeto de estudio de la investigación. Además se refleja la metodología de desarrollo de *software* que es utilizada para dar solución a la propuesta, así como las tecnologías y herramientas escogidas para el desarrollo de la aplicación.

1.1 Objeto de estudio

1.1.1. Descripción general

La Empresa de Acueducto y Alcantarillado Guardalavaca tiene la misión de brindar un servicio de suministro de agua seguro y con calidad al polo turístico de la provincia de Holguín y al sector residencial y estatal de los municipios Banes y Antilla, así como evacuar las aguas residuales, mitigando el impacto de estas en el medio ambiente. Se logra así una gestión integral de las aguas, para satisfacer con calidad el servicio a los clientes, cuenta con un personal altamente especializado y motivado, que desarrolla soluciones integrales de abastecimiento de agua y evacuación de residuales. En dicha empresa se destaca el proceso de gestión comercial para el sector estatal, el cual tiene como objetivo establecer el modo de realizar y controlar los procesos de contratación, verificación de consumos, facturación y cobro a los clientes, de forma que garantice que los mismos se realicen con la máxima calidad, eficacia y de forma coherente con los objetivos de la organización.

1.2 Tendencias y tecnologías actuales para el desarrollo de sistemas informáticos

El desarrollo de aplicaciones informáticas requiere el empleo de modernas tecnologías y herramientas que permitan realizar productos informáticos que se ajusten a las necesidades de los usuarios. Se realizó un estudio preliminar para determinar las tecnologías a utilizar basándose en las ventajas y características que proporcionan para el desarrollo de este producto informático.

1.2.1 Gestión de información

La gestión de la información puede definirse como “el proceso de analizar y utilizar la información que se ha recabado y registrado para permitir a los administradores (de todos los niveles) tomar decisiones documentadas”. [1]

El valor de la gestión de la información en las organizaciones constituye un factor crítico de éxito, que se acepta unánimemente como recurso indispensable para ampliar la competitividad, aumentar la calidad y la satisfacción de los usuarios, así como para desenvolverse en el ámbito global. Esto ha conducido al desarrollo de sistemas de información para el manejo, tratamiento y uso de la información en las organizaciones que impulsados por los recursos humanos en su condición de agentes y líderes del cambio, posibilitan la agilidad y facilidad de acceso necesaria para ejecución de los procesos organizacionales y la toma de decisiones.

1.2.2 Sistemas de Información

Con el avance de las Tecnologías de la Información (TI), las organizaciones se apoyan en Sistemas de Información (SI) para informatizar los procesos productivos y de esta forma ganar ventajas competitivas e información para la toma de decisiones.

Un SI se define como un grupo de componentes interrelacionados que trabajan en conjunto hacia un objetivo común al aceptar entradas y producir salidas en un proceso de transformación organizado. Los equipos computacionales o *hardware*, el personal que interactúa con el sistema, los datos e información que son introducidos, almacenados, procesados y visualizados a través del sistema y el *software,* son los principales elementos que posibilitan su funcionamiento. Los SI pueden comunicarse con otros sistemas e incluso tener inmersos subsistemas [1].

Según (*Parra, 2010)*, el empleo adecuado de los SI en la organización mejora de manera gradual la:

- ✓ Calidad en productos y servicios
- ✓ Atención a los clientes
- ✓ Relación con los proveedores

- ✓ Condiciones en el ambiente de trabajo
- ✓ Comunicación interpersonal
- ✓ Estimulación de la participación de los trabajadores
- ✓ Reducción del número de procesos de gestión/producción
- ✓ Simplificación de los procesos de gestión/producción
- ✓ Eficiencia en el uso de los recursos
- ✓ Diseño de nuevas y mejores herramientas para la gestión de la dirección

La información que generan los SI sirve de apoyo a los mandos intermedios y a la alta administración en el proceso de toma de decisiones. Son interactivos y amigables, con altos estándares de diseño gráfico y visual, ya que están dirigidos al usuario final. Apoyan la toma de decisiones que, por su misma naturaleza, son repetitivas y de decisiones no estructuradas que no suelen repetirse.

Los elementos que interactúan entre sí de forma organizada dando lugar a una información más elaborada y orientada en función de los objetivos que se persiguen son: el equipo computacional, el recurso humano, los datos o información fuente, programas ejecutados por las computadoras, las telecomunicaciones y los procedimientos de políticas y reglas de operación [2], como se muestra en la Figura 1

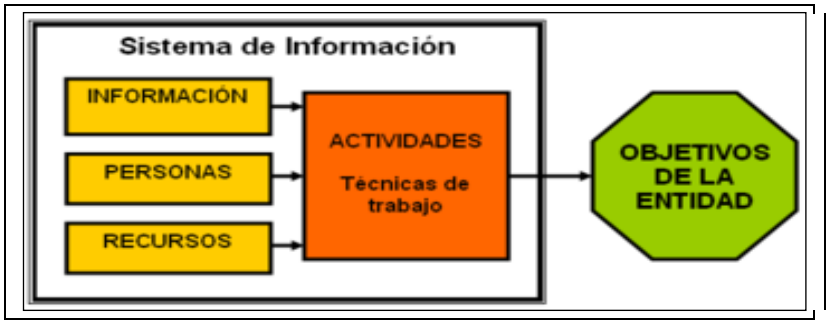

Figura 1 Elementos de un SI.

1.2.3 Clasificación de los Sistemas de Información

Según la función a la que vayan destinados o el tipo de usuario final del mismo, los SI pueden clasificarse en:

- ✓ Sistema de procesamiento de transacciones (**TPS** por sus siglas en inglés): Gestiona la información referente a las transacciones producidas en una empresa u organización.
- ✓ Sistemas de información gerencial (**MIS** por sus siglas en inglés): Orientados a solucionar problemas empresariales en general
- ✓ Sistemas de soporte a decisiones (**DSS** por sus siglas en inglés): Herramienta para realizar el análisis de las diferentes variables de negocio con la finalidad de apoyar el proceso de toma de decisiones
- ✓ Sistemas de información ejecutiva (**EIS** por sus siglas en inglés): Herramienta orientada a usuarios de nivel gerencial, que permite monitorizar el estado de las variables de un área o unidad de la empresa a partir de información interna y externa a la misma
- ✓ Sistemas de automatización de oficinas (**OAS** por sus siglas en inglés): Aplicaciones destinadas a ayudar al trabajo diario del administrativo de una empresa u organización
- ✓ Sistema experto (**SE** por sus siglas en inglés): Emulan el comportamiento de un experto en un dominio concreto
- ✓ Sistema de Planificación de Recursos (**ERP** por sus siglas en inglés): Integran la información y los procesos de una organización en un solo sistema

¿Por qué implementar un SI de tipo OAS para la Empresa de Acueducto y Alcantarillado Guardalavaca?

Los sistemas de automatización de oficinas (**OAS**) tienen como objetivo ayudar al trabajo diario del administrativo y a la colaboración entre las diferentes áreas de una organización, unido al incremento de la productividad y la eficiencia a través de múltiples tecnologías orientadas a mejorar el desempeño de las actividades realizadas dentro de una empresa. Las razones por las que debe crearse son las siguientes:

- ✓ Mejor control de las solicitudes de servicios prestadas por la empresa, así como la realización de los cobros
- ✓ Para tener un acceso a la información a través de la red.

1.3 Arquitectura Cliente – Servidor

La Arquitectura Cliente - Servidor es un modelo para el desarrollo de Sistemas de Información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina **cliente** a la entidad mediante la cual un usuario solicita un servicio, realiza una petición o demanda el uso de recursos. Se comunica con procesos auxiliares que facilitan la conexión con el servidor, enviar una solicitud, recibir una respuesta, controlar fallas y realizar actividades de sincronización y de seguridad. A cualquiera de las computadoras que proporcionan información se le conoce por el nombre de **servidor.** [3]

Es la entidad física que provee un servicio y devuelve resultados; ejecuta el procesamiento de datos, aplicaciones y manejo de la información o recursos. A través de procesos auxiliares en algunos casos reciben las solicitudes del cliente, comprueban la protección, activan un proceso servidor para devolver la petición, recibir la respuesta y enviarla al cliente, además controlan los bloqueos internos y la recuperación ante fallas.[4]

En la figura 2 se puede observar las entidades que interactúan con esta arquitectura para lograr su adecuado funcionamiento.

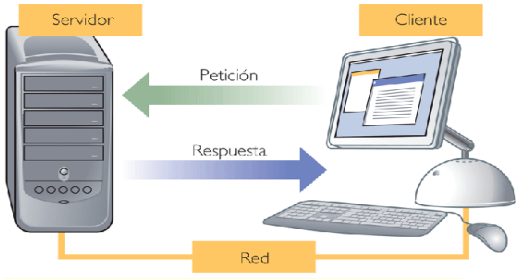

Figura 2 Arquitectura Cliente-Servidor

Dentro de los aspectos fundamentales que caracteriza esta arquitectura se encuentran los siguientes:

- ✓ El servidor presenta a todos sus clientes una interfaz única y bien definida

- ✓ El almacenamiento y procesamiento de los datos es centralizado, evitando tener información duplicada o no sincronizada en todos los clientes
- ✓ No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicios
- ✓ La relación establecida es de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a los recursos compartidos

Para la comunicación entre los componentes **cliente** y **servidor**, existe una **infraestructura lógica de comunicaciones** que proporciona los mecanismos básicos de direccionamiento y transporte.

Para un adecuado uso de esta arquitectura, se debe tener dominio de las ventajas que ofrece la misma, entre ellas se pueden mencionar:

- ✓ Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el sistema
- ✓ Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado
- ✓ Facilidad de mantenimiento: la centralización de la aplicación hace que los cambios se deban realizar en el servidor y se reflejen de manera automática en los clientes
- ✓ Los requerimientos de *hardware* son sustancialmente menores, pues la máquina cliente solo requiere soportar las rutinas del cliente, y los procesos costosos como la lógica del negocio queda circunscrita al servidor
- ✓ La disminución gradual del movimiento de grandes volúmenes de información por la red hacia las estaciones de trabajos, a partir del control de los datos por los servidores al procesar peticiones y enviar sólo las respuestas requeridas a la máquina cliente y de esta manera reducir el tráfico de la red
- ✓ Acceso a los datos desde cualquier sitio de la red y la total independencia de la plataforma en las máquinas clientes y servidores

Por las características del flujo de información identificado dentro de la organización que se describe, la arquitectura organizacional, la sensibilidad de la información gestionada, la infraestructura tecnológica que posee, la distribución espacial de los entes que actúan, la necesidad inminente de una eficiente gestión de información, el estado psicológico-emocional de los miembros y basado en los beneficio que trae implícito el empleo de esta arquitectura, se consideró que era el modelo que más se ajustaba a las necesidades del proyecto que se acomete.

1.3.1 Aplicaciones de Escritorio

Por aplicaciones de escritorio se entiende toda aplicación que ha sido desarrollada para ser ejecutada en una plataforma específica, ya sea *Windows*, *Linux* o *Mac*. El desarrollo sobre una plataforma, normalmente, implica que la aplicación "no" pueda ser ejecutada en otras. El desarrollo de la informática ha estado acompañado por el desarrollo de programas de gestión para *Windows*. Por ese motivo se usan muchas tecnologías como son: la plataforma *.Net, Visual C++, MS Access, Java,* entre otras.

Las principales ventajas de una aplicación de escritorio son:

- ✓ Navegación e interfaz de usuario más rápidas que en una aplicación *Web*
- ✓ Normalmente no es necesaria una conexión a *Internet* ya que funciona localmente o en la intranet de la empresa
- ✓ Es mejor opción cuando es necesario realizar numerosos cálculos o procesos de CPU (Unidad Central de Procesamiento) muy intensivos. En estos casos se suele usar el lenguaje *C++* para generar una aplicación compilada a código nativo y dar lugar a un programa que posee un extraordinario rendimiento
- ✓ Fácil acceso a recursos locales: disco duro, impresora, memoria de vídeo, sonido, *mouse*, teclado, entre otros
- ✓ Fácil acceso a aplicaciones locales en caso de que sea necesario compartir datos entre aplicaciones

1.3.2 Aplicaciones Web

"En inglés se denomina (*browser-based-aplication*), es decir, aplicación basada en navegadores. Son programas que se diseñan para funcionar a través de un

navegador de *internet*, es decir, son aplicaciones que se ejecutan de forma *online*."[5]

Las aplicaciones *Web* son las herramientas más efectivas para lograr un objetivo de interoperaciones o de mucho flujo de información. Durante el estudio de diferentes tecnologías estas son las más populares debido a lo práctico del navegador *Web* como cliente ligero, así como la facilidad para actualizar y mantener las aplicaciones sin distribuir e instalar *software* a miles de usuarios potenciales. Una aplicación *Web* permite que siempre se esté accediendo a una información actualizada en cualquier parte de dicha red, además de ser un conjunto de páginas enlazadas que visualizan diferentes partes de la información que se quiere mostrar a través de ella, siendo una de las mejores herramientas para divulgar, gestionar y compartir la información por lo que trae consigo un aumento de la eficiencia en cuanto a la manipulación del volumen de la misma. [6]

Estas aplicaciones poseen ventajas tales como:

Disponibilidad: Suele ser alta porque sus usuarios pueden acceder (siempre que se haya concedido este permiso) desde cualquier otra máquina que no sea la suya.

Actualizaciones inmediatas: Todos los usuarios al conectarse usan siempre la última versión que se haya mejorado hasta ese momento.

Ahorra tiempo: Se pueden realizar tareas sencillas sin necesidad de instalar *software* complementario.

Poca capacidad en disco duro: No ocupan espacio en el disco duro del cliente.

Consumo de recursos bajo: Muchas de las tareas que realiza el *software* no consumen recursos del cliente porque se realizan desde el propio servidor.

Colaboración: Es sencillo acceder a los datos y compartirlos por parte de varios usuarios ya que el acceso al servicio se realiza desde una única ubicación.

El *software* resultante en esta investigación está insertado en un sistema en ambiente *Web* ,se propone emplear este tipo de aplicación para la elaboración de la solución informática propuesta.[7]

1.4 Lenguajes de Desarrollo Web

Todo programa o sistema informático se sustenta en el código con el que se crea. Los lenguajes de programación expresan ideas o abstracciones de la realidad en forma de código. La eficacia de los programas es usualmente determinada por la calidad del código y la calidad del mismo puede también ser condicionada por la elección de un determinado lenguaje de programación.[8]

Cada lenguaje de programación tiene un nivel de expresividad, lo que influye en su uso en los distintos escenarios que pueda aplicarse. Esta expresividad está dada fundamentalmente por los paradigmas que implementa, los más utilizados hoy en día son los siguientes:

Imperativo: Describe la programación en términos del estado del programa y sentencias que cambian dicho estado. Los programas imperativos son un conjunto de instrucciones que le indican a la computadora como realizar una tarea.

Orientado a Objetos: La Programación Orientada a Objetos (*POO*) introduce un mayor nivel de abstracción que sus predecesores. Los conceptos de Clases y Objetos proporcionan una abstracción del mundo centrada en los seres y no en los verbos. Los datos aparecen encapsulados dentro del concepto de clase y mediante conceptos como la composición, la herencia y el polimorfismo se pueden implementar relaciones entre las Clases y Objetos.

Funcional: El paradigma funcional está basado en el concepto matemático de función. El paradigma funcional considera al programa como una función matemática donde el dominio representaría el dominio de todas las entradas posibles y el rango seria el conjunto de todas las salidas posibles. La forma en que funciona puede ser entendida como una caja negra.

Orientado a Aspectos: Este paradigma persigue permitir que un programa sea construido describiendo conceptos de forma separada. Los lenguajes orientados a aspectos brindan mecanismos y constructores para capturar a aquellos elementos que se diseminan por todo el sistema.

Lógica: En este paradigma la lógica representa conocimiento, el cual es manipulado mediante inferencias. A diferencia de los demás paradigmas,

trabajar en este significa que hacer y no como hacerlo, por ello son llamados lenguajes declarativos. [9]

Según el índice *TIOBE* (indicador acerca de la popularidad de los lenguajes de programación) los 5 primeros lenguajes de programación actuales en cuanto a mayor relevancia son:

Java: Lenguaje Orientado a Objetos puro e interpretado mediante *byte code*. Es el preferido para la construcción de aplicaciones empresariales. Su principal debilidad es el pobre rendimiento que posee para escenarios que requieran uso intensivo de los recursos computacionales.

C: Lenguaje procedural y compilado. De preferencia para escribir controladores de dispositivos y código de bajo nivel. Su principal debilidad es su pobre expresividad para escenarios más complejos, sacrificando productividad.

***C++*:** Lenguaje orientado a objetos y compilado. Es el indicado a la hora de implementar software de alta complejidad y que requiera altos índices de rendimiento. Su principal debilidad es debido a que su sintaxis es algo compleja y tiene menos expresividad que otros competidores como *Python* y *C#*.

Python: Lenguaje orientado a objetos, funcional y orientado a aspectos e interpretado mediante código fuente o *byte code*. El más utilizado para escribir aplicaciones utilitarias de sistemas operativos libres. Su principal debilidad al igual que *Java* yace en su rendimiento.

PHP: Lenguaje orientado a objetos e interpretado mediante código fuente. El preferido para escribir aplicaciones *Web* por su ligereza y simplicidad. Su principal debilidad está en que se encuentra orientado solo a la *Web* y no hacia otros usos. [9]

Lugar	Lenguaje	Índice (%)	Tendencia
1	Java	18.482	=
2	C	14.986	=
3	C++	8.187	↑
4	Python	7.038	↑↑↑
5	PHP	6.973	↓↓
6	C#	6.809	=
7	Visual Basic	4.924	↓↓
8	Objetive-C	2.571	↑↑↑↑
9	JavaScript	2.558	↑
10	Perl	1.907	↓↓

Leyenda
↑ Aumento del índice de popularidad
= Se mantiene el índice de popularidad
↓ Disminución del índice de popularidad

Figura 3 Lenguajes de Programación y su auge según índice TIOBE.

1.4.1 *Python*

Python es un lenguaje de programación de propósito general, cuya expansión y popularidad es relativamente reciente. Se puede decir que este lenguaje es una apuesta por la simplicidad, versatilidad y rapidez de desarrollo, este lenguaje de *scripting* es independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones *Windows* a servidores de red o incluso, páginas *Web*. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad de ejecución.

En los últimos años el lenguaje se ha hecho muy popular, gracias a varias razones como:

- ✓ La cantidad de librerías que contiene, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero
- ✓ La sencillez y velocidad con la que se crean los programas. Un programa en *Python* puede tener de 3 a 5 líneas de código menos que su equivalente en *Java* o *C*

- ✓ La cantidad de plataformas en las que podemos desarrollar, como *Unix, Windows, OS/2, Mac*, Amiga y otros
- ✓ Además, *Python* es gratuito, incluso para propósitos empresariales

El creador del lenguaje es un europeo llamado *Guido Van Rossum*. Hace ya más de una década que diseñó *Python*, ayudado y motivado por su experiencia en la creación de otro lenguaje llamado *ABC*. El objetivo de *Guido* era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en *Unix* usando *C*.

El desarrollo de *Python* duró varios años. En el 2000 ya disponía de un producto bastante completo y un equipo de desarrollo con el que se había asociado incluso en proyectos empresariales. Actualmente trabaja en *Zope*, una plataforma de gestión de contenidos y servidor de aplicaciones para la *Web*, programada por completo en *Python*.

Propósito general: Se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la *Web*, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

Multiplataforma: Hay versiones disponibles de *Python* en muchos sistemas informáticos distintos. Originalmente se desarrolló para *Unix*, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

Interpretado: Quiere decir que no se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos *byte codes* que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

Interactivo: *Python* dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

Orientado a Objetos: La programación orientada a objetos está soportada en *Python* y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de *strings*, números, archivos, etc. Además, existen muchas librerías que podemos importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en *red* o cosas tan interesantes como crear archivos comprimidos en *zip.*

Sintaxis clara: *Python* tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave *begin* y *end*. Para separar las porciones de código en *Python* se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un *bucle*. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar. [9]

Algunas empresas que utilizan *Python* son *Yahoo, Google, Walt Disney*, la *NASA, Red Hat*, entre otras.

***Python* y sus ventajas**: Este lenguaje presenta una serie de ventajas que lo hacen muy atractivo, tanto para su uso profesional como para el aprendizaje de la programación, entre las que se destacan:

- ✓ *Python* es un lenguaje muy “expresivo”, es decir, los programas son muy compactos, un programa en *Python* suele ser bastante más corto que su equivalente en lenguajes como *C*, por muchos, *Python* es considerado un lenguaje de programación de muy alto nivel
- ✓ *Python* es muy legible, la sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta fácil, en comparación con otros lenguajes
- ✓ *Python* puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objeto
- ✓ *Python* es un muy buen lenguaje para empezar a programar
- ✓ Una ventaja fundamental de *Python* es la gratuidad de su intérprete [10]

Otras Ventajas: Desarrollo más rápido ya que se puede escribir un programa, salvarlo y ejecutarlo. En un lenguaje compilado tienes que pasar por los pasos de compilar y ligar el *software*, lo cual puede ser un proceso lento. Además posee multiplataforma, el mismo código funciona en cualquier arquitectura, la única condición es que disponga del intérprete del lenguaje. No es necesario compilar el código una vez para cada arquitectura.

Inconvenientes: Lentitud ya que los programas interpretados son más lentos que los compilados. Sin embargo los programas interpretados suelen ser cortos, en los que la diferencia es inapreciable.

Se selecciona *Python* como lenguaje de programación ya que luego del estudio realizado acerca de los diferentes lenguajes de desarrollo *Web* se revelan la cantidad de ventajas que posee el mismo, sus facilidades, su creciente auge y popularidad.

1.5 *Framework*

Un *framework* o marco de trabajo, es un conjunto de técnicas y herramientas que permiten el desarrollo de algún producto abstrayendo y facilitando el desarrollo de ciertas tareas según el dominio para el cual está construido el mismo.

Estos son un conjunto de bloques y servicios pre elaborados que los desarrolladores pueden emplear para lograr alguna meta; esto puede incluir el desarrollo completo de una aplicación, partes de la aplicación, prueba.

Básicamente, los marcos de trabajo, proveen la infraestructura para la solución de un tipo de problema; no es la solución del mismo; son simplemente una herramienta con la cual podemos estructurar una solución.

En términos de ayuda en el diseño de la solución al problema, los *frameworks* varían en el nivel de intrusión. Algunos establecen vías para estructurar la solución, sin embargo otros proveen una infraestructura genérica y dejan muchas decisiones en manos del desarrollador.

Los marcos de trabajo se caracterizan principalmente por brindar una estructura para nuestra solución. Esto invariablemente nos ahorra tiempo, posibilitándonos concentrarnos en el problema del negocio en cuestión. El uso de uno de estos que sea popular usualmente significa aplicar una estructura o

herramienta que otras personas han analizado y mejorado a lo largo del tiempo.

Cuando un *framework* particular ha soportado el escrutinio de los desarrolladores a nivel global, seguramente que la solución al problema de esta investigación pueden ser resueltas con el mismo. Además promueven el uso de enfoques estándares para la solución de problemas particulares.

1.5.1 *Frameworks* para el desarrollo *Web*

Se denomina *framework Web* al conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en *XML*) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas *Web*.[11]

Estos son el esqueleto sobre el cual varios objetos son integrados para una solución dada. Son diseñados con el intento de facilitar el desarrollo de *software*, permitiendo de esa manera a los diseñadores y programadores pasar más tiempo identificando requerimientos de *software* tratando con los detalles de bajo nivel para proveer un sistema funcional.[6]

La mayoría de los *frameworks Web* emplean una Arquitectura Modelo Vista Controlador (*MVC*) en la que basan el desarrollo de la aplicación. Esto permite una clara separación entre la lógica del negocio y la capa de presentación de los datos.

La validación de las entradas de los usuarios respecto a las reglas del negocio es una de las tareas más engorrosas en el desarrollo de aplicaciones *Web*. Sin embargo la mayoría de los *frameworks* dedicados al desarrollo *Web* tienen, como mínimo, una estrategia para definir reglas de validación y mostrar mensajes de error cuando estas reglas no se cumplen correctamente.[8]

En conclusión, un f*ramework* de tipo *Web* es una base de programación que atiende a sus sucesores de una forma estructural y/o en cascada permitiendo cualquier respuesta ante sus miembros, o secciones de una aplicación *Web*.

1.5.2 Symfony

Symfony es un *framework* para crear aplicaciones *Web* en *PHP*. Contiene un conjunto de herramientas y utilidades que simplifican y aceleran el desarrollo de las aplicaciones. Este marco de trabajo emplea el modelo vista controlador *MVC* como patrón de diseño para el desarrollo.[12]

Con este marco de trabajo se logra que una aplicación *Web* tenga todo en su lugar, donde el mantenimiento y Ia corrección de errores se hace bastante sencilla. Cuenta con un gran número de librerías, herramientas y asistentes que ayudan a desarrollar una aplicación mucho más rápida que haciéndolo de Ia manera tradicional.

Symfony2 es la versión más reciente de *Symfony,* el popular *framework* para desarrollar aplicaciones *PHP*. Posee un contenedor de Inyección de Dependencias basado en *Spring.*

Ha sido ideado para exprimir al límite todas las nuevas características de *PHP 5.3* y por eso es uno de los f*rameworks PHP* con mejor rendimiento. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en tu proyecto.[13]

1.5.3 Django

Django es un *framework* de desarrollo *Web* de código abierto, escrito en *Python*, que cumple en cierta medida el paradigma del Modelo Vista Controlador. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la *World Company* de *Lawrence, Kansas*, y fue liberada al público bajo una licencia *BSD* en julio de 2005; el *framework* fue nombrado en alusión al guitarrista de *jazz* gitano *Django Reinhardt*. En Junio del 2008 fue anunciado que la recién formada *Django Software Foundation* se hará cargo de *Django* en el futuro. La meta fundamental de *Django* es facilitar la creación de sitios *Web*. *Django* pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, del desarrollo rápido y del principio de *DRY* (del inglés *Don't Repeat Yourself*). *Python* es usado en todas las partes del *framework*, incluso en configuraciones, archivos, y en los modelos de datos. [14]

Con *Django*, se construyen los sitios *Web* en cuestión de horas, no días, ni semanas, ni años. Cada una de las partes del *framework Django* ha sido diseñada con el concepto de productividad en mente.

Django sigue la arquitectura *MVC*, en términos simples, es una manera de desarrollo de *software* en donde el código para definir y acceder a los datos (el

modelo) se encuentra separado de la lógica de negocios (el controlador), a su vez está separado de la interfaz de usuario (la vista).

Django mantiene de manera estricta a lo largo de su propio código un diseño limpio, y le hace seguir las mejores prácticas cuando se refiere al desarrollo de su aplicación *Web.* En resumen, *Django* facilita el hacer las cosas de manera correcta.[15]

Visión general y característica: Los orígenes de *Django* en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, *Django* proporciona una aplicación incorporada para administrar los contenidos, que puede incluirse como parte de cualquier página hecha con *Django* y que puede administrar varias páginas hechas con el mismo a partir de una misma instalación; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno, y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).[16]

La distribución principal de *Django* también aglutina aplicaciones que proporcionan un sistema de comentarios, herramientas para sindicar contenido vía *RSS* y/o *Atom*, "páginas planas" que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas y un sistema de redirección de *URLs*.

Otras características de *Django* son:

- ✓ Un mapeador objeto-relacional
- ✓ Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con *Django*
- ✓ Una *API* de base de datos robusta
- ✓ Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes

- ✓ Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas
- ✓ Un despachador de *URLs* basado en expresiones regulares
- ✓ Documentación incorporada accesible a través de la aplicación administrativa (incluyendo documentación generada automáticamente de los modelos y las bibliotecas de plantillas añadidas por las aplicaciones)[6]

Arquitectura: Aunque *Django* está fuertemente inspirado en la filosofía de desarrollo Modelo Vista Controlador, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular y en cambio prefieren hacer "lo que les parece correcto".

Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" *framework MVC* se llama en *Django* "vista", y lo que se llamaría "vista" se llama "plantilla".[16]

Presentación: Aquí se maneja la interacción entre el usuario y el computador. En *Django*, ésta tarea la realizan el *template engine* y el *template loader* que toman la información y la presentan al usuario (vía *HTML*, por ejemplo). El sistema de configuración de *URLs* es también parte de la capa de presentación.

Soporte de bases de datos: Respecto a la base de datos, la recomendada es *PostgreSQL*, pero también son soportadas *MySQL y SQLite 3*. Una vez creados los *data models*, *Django* proporciona una abstracción de la base de datos a través de su *API* que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas *SQL* directamente. En el modelo de datos de *Django*, una clase representa una tabla en la base de datos y las instancias de esta serán las filas en la tabla.

Soporte de servidores *Web*: *Django* incluye un servidor *Web* liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda *Apache 2* con *mod_python*. Aunque *Django* soporta la especificación *WSGI*, por lo que puede correr sobre una gran

variedad de servidores como *FastCGI* o *SCGI* en *Apache* u otros servidores (particularmente *Lighttpd*).

Requerimientos: *Django* requiere *Python 2.3* o superior. No se necesitan otras bibliotecas de *Python* para poder obtener una funcionalidad básica. En un entorno de desarrollo (especialmente si queremos experimentar con *Django*) no necesitamos un *Web Server* instalado, ya que *Django* trae su propio servidor liviano para éste propósito, con la restricción de solo permitir un usuario a la vez. [17]

Ventajas:

- ✓ Es *Python*
- ✓ Es rápido de desarrollar
- ✓ Está pensado para la eficiencia
- ✓ Es modular
- ✓ Tiene muy bajo acoplamiento
- ✓ Genera automáticamente un panel de administración
- ✓ Sus bibliotecas hacen gran parte del trabajo
- ✓ Soporta varias bases de datos (*MySQL, SQLite, Postgres*, *MS-SQL*)
- ✓ Es *MVC*

Desventajas:

- ✓ Es *Python*
- ✓ No es tan simple de implantar
- ✓ Es más lento que un *framework* en un lenguaje compilado
- ✓ No incluye *AJAX* de serie (todavía)[18]

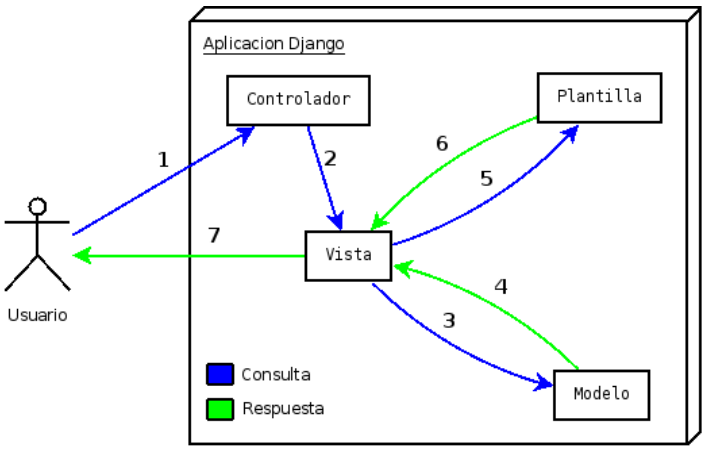

Figura 4 Esquema de consulta

1.6 Metodologías de desarrollo de software

Desarrollar un buen *software* depende en gran medida de la metodología que se elija para guiar las diferentes fases por las que debe transitar el mismo. Dado que estas propician el conjunto de procedimientos, técnicas, herramientas, y el soporte documental que necesitan sus desarrolladores. El paso inicial consiste en elegir cuál de las dos tendencias existentes (ágiles y tradicionales) se adapta mejor tanto a los intereses del cliente como a los del equipo de desarrollo, de forma tal que el equipo maximice sus capacidades y que los clientes queden satisfechos más allá de las necesidades definidas al inicio del proyecto. [19]

Las metodologías tradicionales centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido en la fase inicial del desarrollo del proyecto. Esto pone al cliente en una situación que puede ser muy incómoda para él pues deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio dada la falta de flexibilidad de estos ante un entorno volátil. Resultan eficaces para proyectos grandes, con un alto grado de complejidad que cuente con un numeroso equipo de desarrollo. Entre los principales exponentes de estas metodologías tradicionales se encuentran *Rational Unified Processes* (*RUP*, por sus siglas en inglés) y el Marco de trabajo de Soluciones de *Microsoft* (*MSF*, por sus siglas en inglés). [20]

En su lugar las metodologías ágiles se sustentan en principios tales como: los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados; es más importante crear un *software* que funcione que escribir documentación exhaustiva, la colaboración con el cliente debe prevalecer sobre la negociación de contratos; la capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.[21]

Estos principios aportan ventajas competitivas en entornos volátiles y propician una reducción de los costos. La retroalimentación con el cliente posibilita un

mayor entendimiento por parte de este de funcionalidades vitales para su negocio. Resultan efectivos en proyectos pequeños con un nivel medio de complejidad que cuenta con un equipo pequeño para su implementación. A continuación se describen algunas características de estas metodologías.

1.6.1 Programación Extrema

Programación Extrema (*XP*, por sus siglas en inglés) es una metodología ágil centrada en potenciar las relaciones interpersonales, propiciando una retroalimentación continua con el cliente como clave para el éxito en el desarrollo de *software*. Simplicidad en las soluciones implementadas mediante un desarrollo iterativo e incremental.[22]

XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Apuesta por una planificación más transparente para los clientes, pues le permite conocer las fechas de entrega de funcionalidades vitales para su negocio en diferentes momentos del desarrollo del *software*. Esto propicia que la presión se distribuya a lo largo de todo el proyecto y no en una entrega final.[23]

El objetivo de *XP* son grupos pequeños y medianos de construcción de s*oftware* en donde los requisitos aún son muy ambiguos, cambian rápidamente o son de alto riesgo. *XP* busca la satisfacción del cliente tratando de mantener durante todo el tiempo su confianza en el producto. Además, sugiere que el lugar de trabajo sea una sala amplia, si es posible sin divisiones (en el centro los programadores, en la periferia los equipos individuales). Una ventaja del espacio abierto es el incremento en la comunicación y el proporcionar una agenda dinámica en el entorno de cada proyecto. *XP* es un proceso muy orientado a la implementación, en el que se genera poca documentación y en que la funcionalidad exacta del sistema final no se define nunca formal y contractualmente. Es por eso que este método es más aplicable para desarrollos internos.[24]

1.6.2 *Iconix*

Es una metodología elaborada por *Doug Rosemberg y Kerdall Scott*. Considerado por este autor como un “proceso” de desarrollo de *software* práctico, emplea el modelo de desarrollo de *software* basado en el enfoque o

paradigma de desarrollo de *software* iterativo e incremental y utiliza el método orientado a objetos con el enfoque basado en el Lenguaje Unificado de Modelado (*UML* por sus siglas en inglés); es una metodología ágil que toma varios aspectos de *Rational Unified Processes* (*RUP* por sus siglas en inglés) y de Programación Extrema (*XP*), y constituye el punto medio entre estas metodologías, la primera de gran tamaño y complejidad y la segunda muy ágil.

Al igual que *RUP*, *Iconix* es una metodología conducida por casos de uso, pero no incorpora tantos artefactos *UML*. Es una metodología relativamente pequeña al igual que *XP*, pero no descarta las etapas de análisis y diseño. Utiliza *UML* en sus etapas, de modo que se pueden seguir los requerimientos y adaptarse a nuevos cambios.[23]

La Figura 3 muestra el esquema general de la metodología *Iconix*, en ella se reflejan los principales modelos y los diagramas.

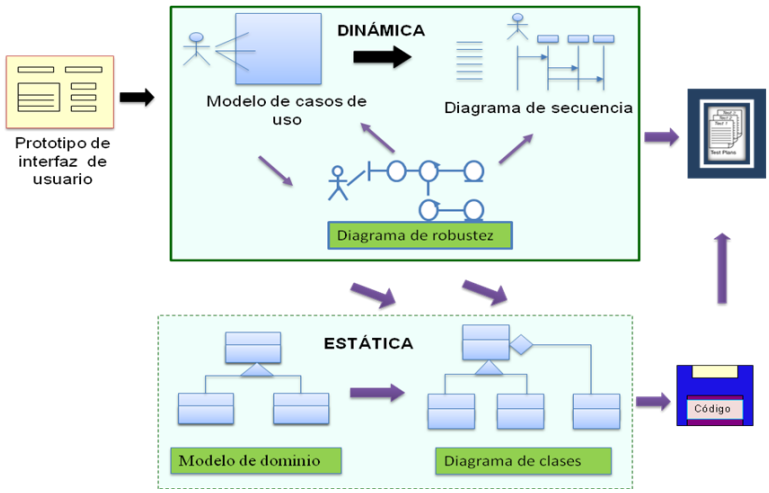

Figura 5 Esquema de trabajo ICONIX.

Según [25] modelos ágiles de metodologías como *Iconix* ofrecen una mayor ventaja en su curva de aprendizaje. La metodología *Iconix* incluye el análisis de robustez, el cual reduce la ambigüedad en las descripciones de caso de uso y asegura que sean escritos en el contexto de un modelo de dominio.

"Este proceso es iterativo e incremental, pues permite varias iteraciones entre el desarrollo del modelo del dominio e identificar y analizar los casos de usos. El modelo estático es incremental refinado a través del modelo dinámico"[26]. Cuenta con amplias bondades en los servicios de negocios. Las soluciones de negocios se enfocan en los servicios en áreas primarias (la experiencia del

usuario, funcionalidad comercial, e infraestructura), con la planeación y la estrategia se fortalece cada área. Esta especialización en las áreas primarias tributa al éxito final de los productos que se entregan a los clientes

Iconix se adapta a los requerimientos ya que está menos orientada al documento y más centrada en el código fuente, exigiéndose menor documentación y ahorro de tiempo. No existe una enorme plantilla de documentación para los casos de uso, sino que estos hablan de qué están haciendo los usuarios en las interfaces finales, es conducida por casos de uso pero no incorpora tantos artefactos *UML*, sino que utiliza un subconjunto mínimo pero suficiente de estos para realizar un buen trabajo de Ingeniería de *Software* en poco tiempo.

Sus principales características son:

Iterativo e incremental: varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. El modelo estático es incrementalmente refinado por los modelos dinámicos.

Trazabilidad: cada paso está referenciado por algún requisito. Se define trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos producidos.

Dinámica del *UML*: la metodología ofrece un uso dinámico del *UML* evidenciándose en los diagramas de caso de uso, diagrama de secuencia y de colaboración.

Modelo de Dominio: con los requisitos se construye el diagrama de clases, que representa el modelo estático del sistema.

Prototipación Rápida: se usa para simular el diseño del sistema. Se espera que los usuarios lo evalúen como si fuera el sistema final. Los cambios al prototipo son planificados con los usuarios antes de llevarlos a cabo. El proceso se repite y finaliza cuando los usuarios y analistas están de acuerdo en que el sistema ha evolucionado lo suficiente como para incluir todas las características necesarias o cuando es evidente que no se obtendrá mayor beneficio con una iteración adicional.

Las fases que establece *Iconix* para el desarrollo de un *software* son:

- ✓ Análisis de requisitos

- ✓ Análisis y diseño preliminar
- ✓ Diseño
- ✓ Implementación[27]

1.6.3 Metodología seleccionada.

Entre las metodologías que se estudiaron se escogió *Iconix* pues permite obtener una aplicación en menor tiempo, garantizando la comprensión de sus procesos mediante el uso dinámico del *UML* como lenguaje de modelado. Favorece la participación de los usuarios finales en la prototipación temprana, en la descripción de los casos de uso, y en las pruebas del sistema. Resulta bastante flexible ante los cambios, pues aunque se prevé que los requerimientos se mantengan bastante estables, siempre es una garantía conocer que en el momento que se produzca alguno el sistema será capaz de adaptarse a ellos sin necesidad de volver al inicio. Es una alternativa factible para proyectos medianos y pequeños como es el caso y para equipos de desarrollo pequeños, en este caso se cuenta solamente con un desarrollador. Además permite la documentación de todo el proceso sin llegar a ser tan exhaustiva como *RUP* ni tan simple como *XP*. En la actualidad se emplea en el desarrollo de aplicaciones *Web* por las facilidades que proporciona, puesto que se realizan *software* con un menor número de errores y en un tiempo mínimo, permite una estrecha relación con el cliente.

1.7 El Lenguaje Unificado de Modelado *(UML)*

El Lenguaje Unificado de Modelado es la sucesión de una serie de métodos de análisis y diseño orientados a objetos que aparecen a finales de los años 80 y principios de los 90. *UML* es llamado un lenguaje de modelado, no un método. Los métodos consisten de ambos, de un lenguaje de modelado y de un proceso. El *UML* fusiona los conceptos de la orientación a objetos aportados por *Booch, OMT* y *OOSE*. Este lenguaje de modelado incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. Los autores del mismo apuntaron también al modelado de sistemas distribuidos y concurrentes para asegurar que el lenguaje maneje adecuadamente estos dominios.

Los elementos de *UML* se muestran mediante diagramas que presentan

múltiples vistas del sistema, ese conjunto de vistas son conocidos como modelos. *UML* presenta varios diagramas donde cada uno representa un aspecto del sistema.[28]

Para el manejo de este leguaje se escoge en la presente investigación la herramienta informática *Enterprise Architect* (*EA*), de la cual se abordan a continuación algunos aspectos de importancia.

1.8 Herramientas de Apoyo a la Ingeniería de Software

La Ingeniería de *Software* Asistida por Computación (*CASE*, por sus siglas en inglés) se considera como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación.

Las herramientas *CASE* representan una forma que permite modelar los procesos de negocios de las empresas y desarrollar los Sistemas de Información Gerenciales. También permiten aumentar la productividad en el desarrollo de un proyecto y como herramientas que son, deben ser aplicadas a una metodología determinada.[29]

Algunos de los componentes de las herramientas *CASE* permiten:

- ✓ Confeccionar la definición de requerimientos de los usuarios
- ✓ Mejorar el diseño de los sistemas
- ✓ Mejorar la eficiencia en la programación (por su generación automática de códigos)
- ✓ Otorgar a la administración un mejor soporte en la documentación
- ✓ Mejorar la planificación de un proyecto
- ✓ Aumentar la calidad del *software*
- ✓ Ayudar a la reutilización del *software*, portabilidad y estandarización de la documentación
- ✓ Ellas en sí mismas son una metodología; su uso está restringido a la metodología elegida para llevar adelante el análisis y diseño del proyecto

1.8.1 *Enterprise Architect (EA)*

Arquitectura Empresarial (*EA*, por sus siglas en inglés) es una herramienta comprensible de diseño y análisis *UML*, cubriendo el desarrollo de *software* desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. *EA* es una herramienta multi-usuario, diseñada para ayudar a construir *software* robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.[30]

Dentro de las principales características que posee esta herramienta *CASE* se mencionan:

El *UML* provee beneficios significativos para ayudar a construir modelos de sistemas de *software* rigurosos y donde es posible mantener la trazabilidad de manera consistente, *EA* soporta este proceso en un entorno fácil de usar, rápido y flexible.

Provee trazabilidad completa desde el análisis de requerimientos y los artefactos de diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados, los equipos de Desarrolladores de Proyectos y Calidad están equipados con la información que ellos necesitan para ayudarles a entregar los proyectos en tiempo.[31]

Utiliza perfiles *UML* para extender el dominio de modelado, mientras que la validación del modelo asegura integridad.

Brinda soporte para los diagramas de *UML* 2 como los Estructurales, de Comportamiento y los Extendidos.

Soporta la generación e ingeniería inversa de código fuente para muchos lenguajes populares, incluyendo *C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript y PHP.*

Las plantillas de generación de código le permiten personalizar el código fuente generado de acuerdo a los requerimientos del cliente.

Soporta transformaciones de Arquitectura avanzada dirigida por modelos (*MDA*) usando plantillas de transformaciones fáciles de editar y desarrollar. Con las transformaciones incorporadas para *DDL, C#, Java, EJB y XSD*, puede desarrollar rápidamente soluciones complejas desde los "modelos

independientes de plataforma" (*MIP*) simples que son el objetivo en los "modelos específicos de plataforma" (*MEP*).

1.9 Sistemas de Gestión de Base de Datos

Un Sistema Gestor de Bases de Datos *(SGBD*) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un *SGBD* permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos.[32]

Las características de un *SGBD* son:

Seguridad: Los *SGBD* deben garantizar que la información que poseen cuente con total seguridad, para que ningún usuario pueda poner en riesgo la confiabilidad de dicha información. Normalmente disponen de un complejo sistema de permisos a usuarios y grupos de usuarios, que permiten otorgar diversas categorías de permisos.

Abstracción de la información: Los *SGBD* ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

Independencia: La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una Base de Datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

Redundancia mínima: Un buen diseño de una base de datos logrará evitar la aparición de información repetida o redundante. De entrada, lo ideal es lograr una redundancia nula; no obstante, en algunos casos la complejidad de los cálculos hace necesaria la aparición de redundancias.

Consistencia: En aquellos casos en los que no se ha logrado esta redundancia nula, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.

Integridad: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, proteger los datos ante fallos de

hardware, datos incorrectos introducidos por usuarios o cualquier otra circunstancia capaz de corromper la información almacenada.

Respaldo y recuperación: Los *SGBD* deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos, y de restaurar a partir de estas copias los datos que se hayan podido perder.

Control de la concurrencia: Un *SGBD* debe controlar el acceso concurrente a la información, que podría derivar en inconsistencias, ya que lo más habitual es que sean muchas las personas que acceden a una base de datos, bien para recuperar información o para almacenarla. Y es también frecuente que dichos accesos se realicen de forma simultánea. [33]

1.9.1 *PostgreSQL*

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) libre. Fue el pionero en muchos de los conceptos existentes en el sistema objeto-relacional actual, incluido, más tarde en otros sistemas de gestión comerciales. Se dice que es objeto-relacional pues incluye características de la orientación a objetos, como puede ser la herencia, tipos de datos, funciones, restricciones, disparadores, reglas e integridad transaccional.

PostgreSQL lleva más de una década de desarrollo, siendo hoy en día, el sistema libre más avanzado, soportando la gran mayoría de las transacciones SQL, control concurrente, teniendo a su disposición varios lenguajes como por ejemplo *C, C++, Java, Python, PHP* etc.[34]

Además es una herramienta muy potente para los desarrolladores, tiene todo de lo que carece *MySQL*. A continuación se muestran las características más notables de esta herramienta:

- Posee una gran escalabilidad. Es capaz de ajustarse al número de clientes y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta (en algunos *benchmarks* [1]se dice que ha llegado a soportar el triple de carga de lo que soporta *MySQL*).

[1] Proceso continuo de medir productos, servicios y prácticas contra los competidores más duros o aquellas compañías reconocidas como líderes en la industria.

- Implementa el uso de *rollback*[2], subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz, y ofreciendo soluciones en campos en las que *MySQL* no podría.
- Tiene la capacidad de comprobar la integridad referencial, así como también la de almacenar procedimientos en la propia base de datos, equiparándolo con los gestores de bases de datos de alto nivel, como puede ser *Oracle.*

Es un motor de bases de datos: avanzado y de código abierto. Este es manejado por comunidades de desarrolladores que hacen posible su evolución. Esta comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Postgres está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, además posee disímiles características que lo hacen ser un sistema de gestor de base de datos muy popular.

Las principales características de este gestor de bases de datos son:

- ✓ Implementación del estándar *SQL92/SQL99*
- ✓ Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (*MAC, IP*), cadenas de bits, entre otras
- ✓ Incorpora una estructura de datos (arreglos)
- ✓ Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes
- ✓ Permite la declaración de funciones propias, así como la definición de disparadores
- ✓ Soporta el uso de índices, reglas y vistas
- ✓ Incluye herencia entre tablas (aunque no entre objetos, ya que no existen), por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales
- ✓ Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos

[2] Es una operación que devuelve a la base de datos a algún estado previo.

- ✓ Permite crear una amplia funcionalidad a través de su sistema de activación de disparadores (*triggers*)[34]

1.9.2 *MySQL*

MySQL es un sistema de Gestión de Bases de Datos relacional, multihilo y multiusuario. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido. [35]

Este gestor de bases de datos es, probablemente, el gestor más usado en el mundo del *software* libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración.

Las principales características de este gestor de Bases de Datos son:

- ✓ Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo
- ✓ Soporta gran cantidad de tipos de datos para las columnas
- ✓ Dispone de bibliotecas o *API's* en gran cantidad de lenguajes *(C, C++, Java, PHP)*
- ✓ Gran portabilidad entre sistemas
- ✓ Soporta hasta 32 índices por tabla
- ✓ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos
- ✓ Es multiplataforma
- ✓ Agrupa transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de estas por segundo

MySQL es *software* de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de *MySQL* y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. *MySQL* usa Licencia Pública General (*GPL*, por sus siglas en inglés) para definir qué puede hacer y qué no puede hacer con el *software* en diferentes situaciones.

1.10 Tecnologías de Servidores *Web*

Los servidores *Web* son aquellos cuya tarea es alojar sitios y/o aplicaciones, las cuales son accedidas por los clientes utilizando un navegador que se comunica con el servidor utilizando el protocolo Lenguaje de Marcado de Hipertexto (*HTTP*, por sus siglas en inglés).

Básicamente un servidor *Web* consta de un intérprete *HTTP* el cual se mantiene a la espera de peticiones de clientes y le responde con el contenido según sea solicitado. El cliente, una vez recibido el código, lo interpreta y lo exhibe en pantalla.

El servidor *Web* sirve las páginas al navegador (el cliente). La mayoría de los servicios de *Internet* son tipos de servidores. Por ejemplo, si se accede a un artículo en *Wikipedia*, la computadora-usuario y el navegador *Web* se consideran el cliente, y las computadoras-servidores, las bases de datos, y los usos que componen *Wikipedia*, el servidor.[23]

Entre los servidores web se encuentran:

1.10.1 *Microsoft Internet Information Server* (*IIS*, por sus siglas en inglés): Sólo funciona sobre sistemas *Windows*. Si se desea usar sobre otro sistema, se tendrá que utilizar una máquina virtual.

Los servicios de *IIS* proporcionan las herramientas y funciones necesarias para administrar de forma sencilla un servidor *Web* seguro. El servidor *Web* se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas, por ejemplo *Microsoft* incluye los de *ASP* y *ASP.NET*. También pueden ser incluidos los de otros fabricantes, como *PHP* o *Perl.*

Su última versión en estos momentos es *IIS 7.5;* lanzado con *Windows Server* 2008 *R2* y *Windows* 7.

1.10.2 *Apache*

Es un servidor *Web HTTP* para una multitud de plataformas *Unix* (*BSD*, *GNU/Linux*), *Microsoft Windows*, *Macintosh* y otras, que implementa el protocolo *HTTP/1.1* y la noción de sitio virtual, ofrece tecnología libre y de código abierto. El servidor A*pache* se desarrolla dentro del proyecto *HTTP Server (httpd)* de la Fundación de *Software* de *Apache*.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache es el componente de servidor *Web* en la popular plataforma de aplicaciones *LAMP*, junto a *MySQL* y trabaja en conjunto con gran cantidad de lenguajes de programación interpretados como *PHP/Perl/Java/JSP/ Python/Ruby*; y otros lenguajes de *script*, que son el complemento ideal para los sitios *web* dinámicos.

La mayor parte de la configuración se realiza en el fichero *apache2.conf* o *httpd.conf,* según el sistema donde esté ejecutándose. Cualquier cambio en este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente. [36]

Tener un servidor bajo *Apache* es una solución sencilla, eficaz y rápida para que los sitios *Web* funcionen al 100% sobre todo sin pagar un solo centavo.

1.10.3 *Apache Tomcat*

También conocido como simplemente *Tomcat* o *Jakarta Tomcat*, es un servidor *Web* multiplataforma que funciona como contenedor de *servlets* y que se desarrolla bajo el proyecto denominado *Jackarta* perteneciente a la *Apache Software Foundation* bajo la licencia *Apache 2.0* y que implementa las especificaciones de los *servlets* y de *Java Server Pages* o *JSP* de *Sun Microsystem*.[37]

Dentro de las principales características de este servidor *Web* encontramos:

- ✓ Puede funcionar como servidor *Web* por sí mismo en entornos de alto nivel de tráfico y alta disponibilidad
- ✓ Es multiplataforma
- ✓ Limpieza interna de código
- ✓ Soporte para la inclusión de contenidos externos directamente en una aplicación *Web*
- ✓ Autenticación de acceso básico
- ✓ Protocolo de Seguridad de Lenguaje de Marcado de Hipertexto (*HTTPS*, por sus siglas en inglés)
- ✓ Consola de administración

- ✓ Negociación de credenciales

Siendo *Apache* uno de los servidores más antiguos, es el mayormente usado hoy en día, por lo que es fácil hallar documentación sobre el mismo. Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona. Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos. Por las razones antes expuestas, es el servidor *Web* escogido para el desarrollo de este PI.

1.11 Conclusiones del Capítulo

Se estudiaron los fundamentos teóricos relacionados a la gestión comercial y de manera general las tecnologías para el desarrollo de aplicaciones *Web.* Dada las características que debe cumplir el *software* y las bondades que ofrecen las tecnologías antes descritas se escogieron: *Python* como lenguaje de programación, el *framework* o marco de trabajo *Django*, como *SGBD PostgreSQL*, *Apache 2* como servidor *Web*, el *EA* como herramienta *CASE* para el modelado de diagramas, y empleando las fases propuestas por la metodología de desarrollo *Iconix* en la elaboración del sistema propuesto.

Capítulo 2 Descripción de la solución propuesta.

En este capítulo se lleva a cabo el análisis, modelación y diseño de la propuesta de solución que será como resultado el Sistema Informático para la gestión comercial en la Empresa de Acueducto y Alcantarillado Guardalavaca. Siguiendo las etapas que componen la metodología *Iconix* permite la construcción del Modelo del Dominio a partir de la captura de los Requerimientos Funcionales (RF), se especifican los Requerimientos No Funcionales (RNF) para modelar las propiedades que el sistema requiere, se modelan los casos de uso, su descripción y sus correspondientes diagramas de robustez como guía para describir los pasos hacia la codificación del sistema.

El capítulo incluye un análisis de sostenibilidad en las dimensiones administrativa, socio–humanista, ambiental y tecnológica, que tiene la aplicación como producto terminado en el marco de la institución. Además del criterio de expertos para evaluar la satisfacción con respecto al PI.

2.1 Definición y análisis de los requerimientos

En esta fase del proceso se tiene como pauta la revisión y análisis de requerimientos que debe cumplir el sistema, así como un primer acercamiento al diagrama del modelo del dominio del sistema.

2.1.1 Análisis de requisitos

El análisis de los requisitos es una etapa vital para todo el proceso de desarrollo de *software*, esta surge mediante un análisis de los requerimientos. Para realizar una buena captura de requerimientos es muy importante la comunicación que se establezca entre el cliente y los desarrolladores. Además este análisis se lleva a cabo para determinar si estos son confusos, incompletos, ambiguos, o contradictorios. Estos son determinados a partir de la información brindada por el cliente acerca del sistema que necesita para la solución del problema y algunos propuestos por los desarrolladores.

El objetivo del modelado del dominio es comprender y describir las clases más importantes dentro del contexto del sistema para contribuir a una comprensión del problema que se supone que el sistema resuelve en relación a su contexto. Los

requerimientos pueden dividirse en Requerimientos Funcionales y Requerimientos No Funcionales.[38]

Los requerimientos funcionales que se enumeran a continuación han sido agrupados en los paquetes definidos, que representan las funcionalidades de interés para el usuario final.

Paquete Gestión de Entidades

1. Gestionar Empresa
    - 1.1 Agregar empresa
    - 1.2 Eliminar empresa
    - 1.3 Modificar empresa
2. Gestionar UEB
    - 2.1 Agregar UEB
    - 2.2 Eliminar UEB
    - 2.3 Modificar UEB
3. Gestionar Oficina Comercial
    - 3.1 Agregar Oficina Comercial
    - 3.2 Eliminar Oficina Comercial
    - 3.3 Modificar Oficina Comercial

Paquete Gestión Comercial

4. Gestionar Cliente
    - 4.1 Agregar Cliente
    - 4.2 Eliminar Cliente
    - 4.3 Modificar Cliente
5. Imprimir datos de clientes.
6. Gestionar Contrato
    - 6.1 Agregar Contrato
    - 6.2 Eliminar Contrato
    - 6.3 Modificar Contrato
7. Imprimir contrato de clientes.
8. Facturar clientes.

9. Imprimir facturas de clientes
10. Exportar comprobantes de ventas del mes para la contabilidad.
11. Registrar pagos de facturas de los clientes.
12. Imprimir comprobantes de cobro.
13. Exportar órdenes de cobro para el BANDEC
14. Imprimir órdenes de cobro presentadas al BANDEC.
15. Imprimir órdenes de cobro no aceptadas por el BANDEC.
16. Imprimir estado de cuentas.
17. Imprimir informe de ventas del mes.
18. Imprimir informe de cuentas por cobrar.
19. Imprimir volúmenes de agua consumidos por sistemas de abasto.
20. Imprimir volúmenes de agua consumidos por organismos.

Paquete Seguridad

21. Gestionar Usuario
    21.1 Agregar Usuario
    21.2 Eliminar Usuario
    21.3 Modificar Usuario
22. Autenticarse
23. Cerrar Sesión
24. Realizar salva de seguridad
25. Validar Usuario

2.1.2 Definición de los conceptos principales del modelo del dominio.

Concepto	Definición
Cliente	Es el protagonista de la acción comercial. Es el destinatario de un servicio o producto ofrecido por un suministrador.
Contrato	Es la realización de la acción necesaria para que el cliente quede con un vínculo legal.
Servicios	Es una actividad o conjunto de actividades de naturaleza casi siempre intangible que se realiza a través de la interacción

	entre el cliente y el empleado y/o instalaciones físicas de servicio, con el objetivo de satisfacerle un deseo o necesidad
Oficina Comercial	Lugar donde se mantiene el control y registro de los servicios de la empresa.

Tabla 1 Definición de los principales conceptos del modelo del dominio

2.1.3 Diagrama del modelo del dominio

La elaboración del modelo del dominio constituye una parte esencial dentro de las etapas que componen la metodología de desarrollo *Iconix,* ya que se modifica de forma constante durante todo el ciclo de vida del proyecto. De esta manera siempre refleja la comprensión actual del ámbito del problema. Según [39] el modelo del dominio es un glosario de los términos (sustantivos y frases sustantivas) fundamentales usados en el proyecto, pero más que eso, es la representación gráfica de los mismos a través de las relaciones de agregación (tiene-un) y generalización (es-un). Además, define el alcance del proyecto y crea las bases para el desarrollo de los casos de uso que se abordarán posteriormente.

Luego de haberse obtenido, organizado y analizado los principales conceptos de la lista de requerimientos para la construcción del modelo del dominio, a continuación en la figura 6 se muestra el diagrama donde se ilustran las relaciones entre estos objetos.

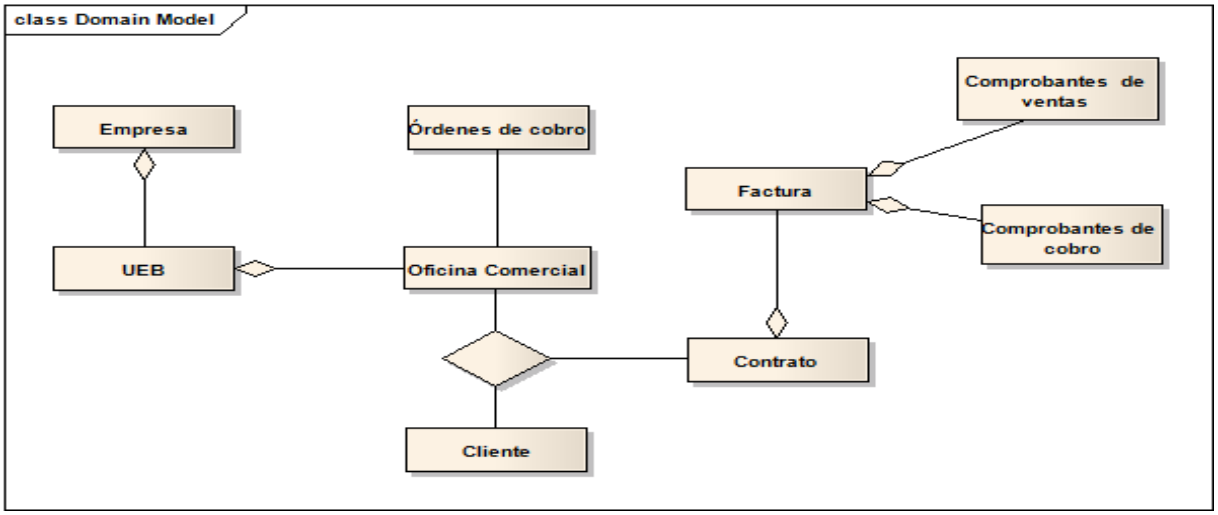

Figura 6 Modelo del Dominio

2.2 Análisis y Diseño Preliminar

El objetivo de la fase de análisis es la construcción de un sistema correcto, mientras que la del diseño tiene como fin la construcción correcta de un sistema. El diseño preliminar constituye el paso intermedio entre el análisis y el diseño, y es precisamente el que posibilita que se puedan entender por completo los requerimientos, mejorando y excluyendo las imprecisiones de los mismos, a través del vínculo existente entre los casos de uso y los objetos del modelo del dominio [38].

2.2.1 Modelo de casos de uso

Los casos de uso brindan una manera estructurada de capturar los requerimientos de comportamiento de un sistema, de esta forma se puede crear un diseño razonable de estos. El modelo de casos de uso es un catálogo de las funcionalidades del sistema. Cada caso de uso representa una interacción que el usuario o actor experimenta cuando usa el sistema[39].

Un sistema puede llegar a tener un número elevado de casos de uso, de esta forma es necesario organizarlos para tener el control sobre ellos. El *UML* brinda la solución con un mecanismo de paquetes que es una forma de agrupar elementos como clases, diagramas o casos de uso. Además permite dividir el trabajo en diferentes áreas. A continuación en la figura 7 se muestra el diagrama de paquetes del sistema.

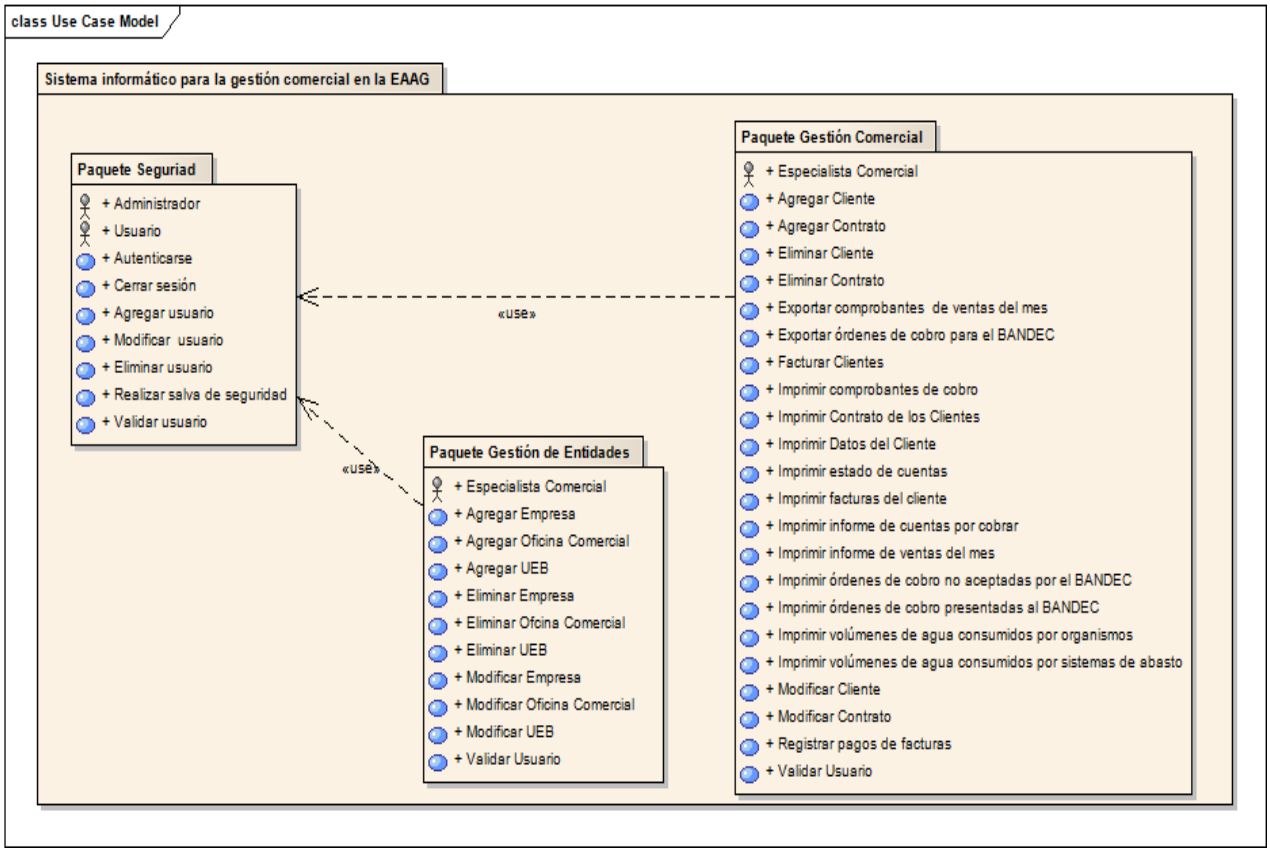

Figura 7 Diagrama de paquetes

A continuación la figura 8 representa el diagrama de Caso de Uso del paquete Gestión de Entidades con sus requerimientos. En este paquete existe el actor que es el Especialista Comercial que realiza todo lo relacionado con la gestión de las Empresas, UEB y Oficinas Comerciales existentes, ya sea insertar, modificar y eliminar. Todo esto va precedido por el caso de uso Validar Usuario ya que es necesario estar autenticado para poder hacer cualquier acción. (Para consultar los restantes diagramas de casos de uso del sistema ver el **Anexo I**).

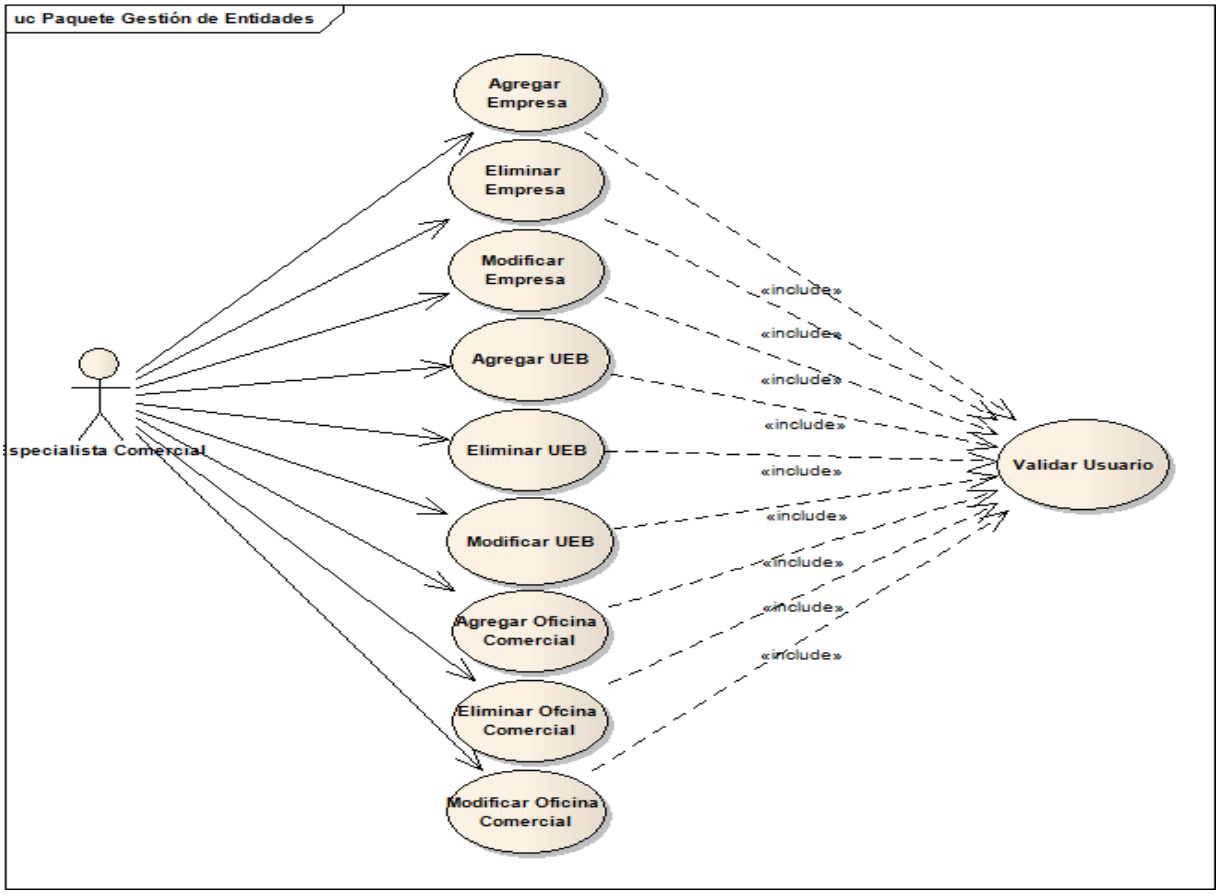

Figura Diagrama de Caso de Uso del paquete Gestión de Entidades

2.2.2 Actores del sistema

Un actor es representado en el diagrama como una figura incrustada y es análogo a un rol que el usuario puede jugar, pero algunas veces el actor solo deberá ser llamado "Usuario", pero frecuentemente le es dado un nombre de rol específico[39]. A continuación se muestran los actores que se identificaron:

Cada diagrama de caso de uso, es iniciado por un actor, los actores reflejan todas las entidades que deben intercambiar información con el sistema, el actor principal del sistema es el Especialista Comercial.

Actor	Definición
Usuario	Persona que interactúa con el software para realizar búsqueda o visualizar alguna información, además de registrar informaciones.
Administrador	Es aquel usuario con privilegios en el sistema, que se encarga de supervisar el mismo, y tiene acceso a toda información. Además de realizar salva de seguridad.
Especialista Comercial	Es el responsable de gestionar a los clientes, así como los contratos, las facturaciones y de llevar todo lo relacionado al proceso que realiza.

Tabla 2 Actores del Sistema.

Entre cada uno de estos actores se establece una jerarquía que está determinada por el rol que asume cada uno de ellos dentro del negocio. Este a su vez determinará las páginas a las que tendrán acceso dentro del sistema y los permisos en cada una de ellas.

2.2.3 Descripción de los casos de uso

Los casos de uso permiten describir la utilización del sistema en el contexto del modelo de objeto. Los casos de uso deben ser descritos siguiendo la regla de los dos párrafos, en voz activa, usando un flujo respuesta/evento entre el usuario y el sistema, y utilizando una estructura de oración sustantivo-verbo-sustantivo. En la descripción de los casos de uso se reflejan dos cursos, un curso básico que muestra cómo se debe realizar el proceso y un curso alterno para describir los posibles errores o sucesos que pueden ocurrir durante la operación [39].En la tabla 3 y 4, se muestra la descripción textual de los casos de uso “Agregar Cliente” y “Autenticarse” respectivamente. En el **Anexo II** se pueden ver otras descripciones.

El resto de las descripciones por cada caso de uso de los diagramas correspondientes, se encuentran en el fichero *EAP*, el cual recoge todas las descripciones de los casos de uso y demás diagramas del proceso de desarrollo *de Iconix*.

Caso de Uso	Agregar Cliente
Curso básico	El Especialista Comercial se autentica y da clic en la opción clientes de la página principal, luego da clic en la opción adicionar dentro de clientes, el sistema muestra la interfaz adicionar clientes, el Especialista Comercial llena los datos correspondientes y da clic en el botón adicionar, el sistema inserta el cliente en la base de datos y muestra el mismo junto a los clientes guardados en el sistema.
Curso alterno	El Especialista Comercial no se ha autenticado, el sistema muestra la ventana Autenticarse y cuando se especifiquen los datos mostrará la ventana correspondiente. Algunos de los campos están en blanco o con algunos formatos incorrectos. El sistema rechaza el cliente y muestra un mensaje con el correspondiente error cometido.

Tabla 3 Descripción textual del caso de uso "Agregar Cliente"

Caso de Uso	Autenticarse
Curso básico	El sistema le muestra al usuario la página de login. El usuario llena los campos del nombre y contraseña de la página de login y hace clic en el botón acceder. El sistema comprueba los datos y autentica al usuario, registrándolo en su base de datos y le muestra la página principal.
Curso alterno	El usuario ha dejado campos en blanco, ha introducido mal la contraseña o algunos de los campos de entrada tienen un formato incorrecto. El sistema rechaza la propuesta y muestra un mensaje de error.

Tabla 4 Descripción textual del caso de uso "Autenticarse"

2.2.4 Análisis de robustez

El análisis de robustez ayuda a identificar los objetos que participan en cada caso de uso mediante un diagrama de robustez. En el diagrama de robustez se debe ilustrar gráficamente las interacciones entre los objetos participantes de un caso de uso. Este diagrama permite analizar el texto narrativo de cada caso de uso e identificar un conjunto inicial de objetos participantes de cada caso de uso[39].

Los objetos que participan en el diagrama de robustez se clasifican dentro de los tres tipos siguientes:

- ✓ **Objetos de interfaz:** usados por los actores para comunicarse con el sistema, generalmente como ventanas, pantallas, diálogos y menús
- ✓ **Objetos entidad:** son objetos del modelo del dominio
- ✓ **Objetos de control:** es la unión entre la interfaz y los objetos entidad. Sirve como conexión entre los usuarios y los datos

Según[40], [41]plantea que la descripción de los casos de uso y los diagramas de robustez se deben acoplar correctamente. Por ello, el texto de la descripción de los casos de uso y los objetos del diagrama de robustez deben tener una relación 1:1, es decir, las frases sustantivas se convierten en entidades e interfaces, mientras que los verbos se convierten en controladores de estos objetos. A continuación se ejemplifica los diagramas de robustez perteneciente a los Casos de uso Agregar Cliente (Ver figura 8) y Autenticarse (Ver figura 9) descrito en la tabla 3 y 4 respectivamente. En el **Anexo III** se pueden ver otros diagramas de robustez y el resto se encuentra en el fichero *EAP*.

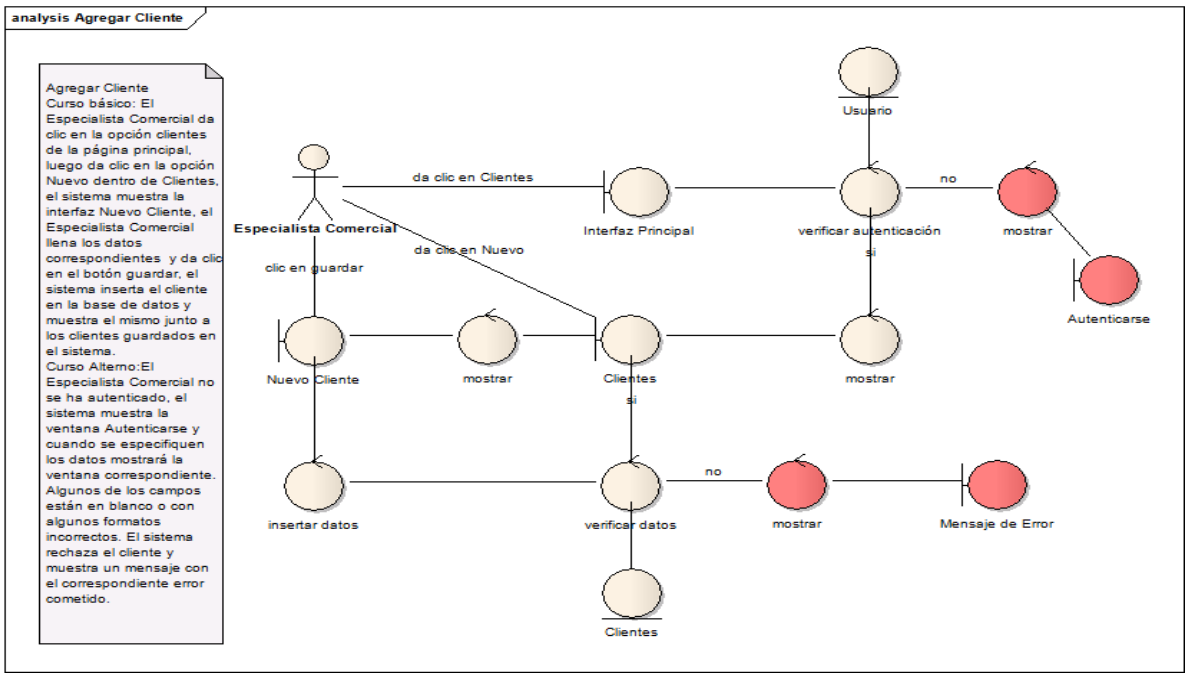

Figura 8 Diagrama de robustez de Agregar Cliente.

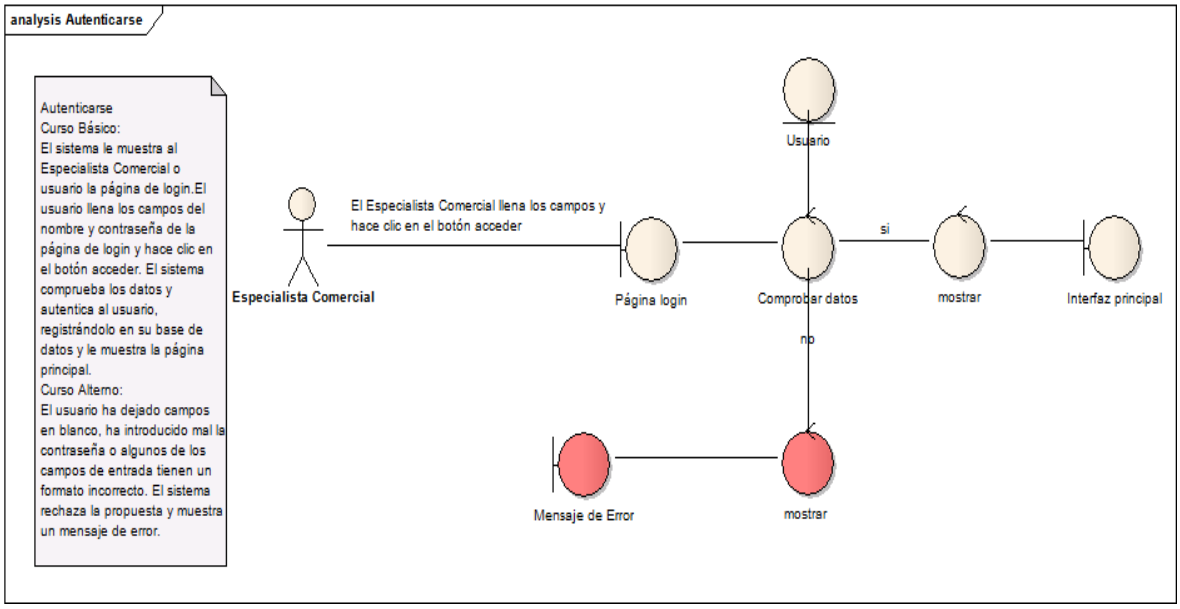

Figura 9 Diagrama de robustez Autenticarse.

2.2.5 Arquitectura técnica

El propósito de la Arquitectura Técnica (AT) es tener una noción general del sistema que se va a desarrollar en términos de estructura. Se construye para satisfacer los requerimientos del negocio y del nivel de servicios del sistema que será desarrollado.

La AT incluye (pero no se limita) a la topología del sistema, es decir, la localización física en la red y la elección del servidor de aplicación.

La arquitectura abarca:

1. Requerimientos no funcionales
2. El modelo de despliegue (red y servidores de aplicación, la topología del sistema y navegadores soportados)

2.2.5.1 Requerimientos No Funcionales

Los requerimientos no funcionales tienen que ver con características que de una u otra forma puedan limitar el sistema, como por ejemplo, el rendimiento (en tiempo y espacio), interfaces de usuario, fiabilidad (robustez del sistema, disponibilidad de equipo), mantenimiento, seguridad, portabilidad, estándares, etc. Los requerimientos no funcionales del sistema han ayudado a determinar las propiedades que debe tener el sistema y se describen a continuación.

Los requerimientos no funcionales para este sistema son los siguientes:

Apariencia o Interfaz externa

- ✓ El sistema debe ser fácil de manipular, pues los usuarios finales no son expertos en computación, a pesar de usar la computadora para las labores rutinarias
- ✓ El flujo de trabajo en el sistema debe asemejarse al proceso rutinario de gestión comercial que se lleva a cabo en la empresa.
- ✓ El diseño debe ser agradable y atrayente a los usuarios para lograr una mejor concentración, sin desviar demasiado su atención del contenido de trabajo
- ✓ Los colores deben encontrarse en la gama del azul, blanco y negro, pero con tonalidades suaves y relajantes para evitar mucho esfuerzo visual, por la gran constancia del uso y dependencia del sistema en el trabajo de los usuarios
- ✓ La interfaz no debe recargarse con imágenes para proporcionar una navegación cómoda
- ✓ El vocabulario utilizado debe ser el español

Usabilidad

- ✓ El sistema debe ser accesible desde cualquier lugar de la empresa y desde las áreas de trabajo de las distintas UEB
- ✓ El sistema debe estar funcionando durante las 24 horas del día
- ✓ El diseño del sistema debe ser sencillo para agilizar su tiempo de conexión

Rendimiento

- ✓ El tiempo de respuesta en la búsqueda de la información almacenada en el archivo digital debe ser corto (5 - 10 segundos aproximadamente), por lo que el acceso a la base de datos se debe efectuar de forma rápida

Portabilidad

- ✓ El sistema puede ser usado desde la máquina computadora-cliente y bajo la plataforma *Windows* o *Linux*
- ✓ La máquina computadora Servidor puede funcionar bajo la plataforma *Windows* o *Linux*

Seguridad

- ✓ Sólo los usuarios autorizados podrán acceder al sistema
- ✓ Garantizar que las funcionalidades del sistema se realicen de acuerdo con el nivel del usuario que esté activo
- ✓ No existe información que se pueda obtener sin ser usuario del sistema
- ✓ Sólo el administrador del sistema tendrá acceso a la base de datos, a los ficheros fuentes del sistema y es responsable de la autorización en general del mismo
- ✓ El sistema debe tener protección contra acciones no autorizadas evitando la corrupción y estados inconscientes que puedan afectar el proceso de gestión comercial o integridad de la información almacenada
- ✓ La información almacenada en el sistema debe ser estrictamente confidencial por ser una entidad que se rige bajo las resoluciones estipuladas por el Instituto Nacional Recursos Hidráulicos (INRH) y Grupo Empresarial de Acueducto y Alcantarillado (GEAAL)

- ✓ Sólo el administrador del sistema podrá hacer salvas de seguridad de manera periódica, según se estipule en la propia entidad

Políticos-culturales

- ✓ Se rige bajo las resoluciones estipuladas por el Instituto Nacional Recursos Hidráulicos (INRH) y Grupo Empresarial de Acueducto y Alcantarillado (GEAAL)

Software

- ✓ El Sistema Operativo (SO) de la máquina computadora cliente debe ser *Windows 7* o superior
- ✓ La máquina computadora servidor debe tener *Windows* 7 o superior, Servidor *Web Apache, PostgreSQL*

Hardware

- ✓ Para ejecutar el *software* como requerimiento mínimo es necesario una computadora con procesador *Pentium* 4 o superior tanto en la máquina cliente como en el servidor
- ✓ La máquina computadora servidor y las computadoras clientes deben estar conectadas a la red

2.2.6 Modelo de despliegue

Dentro de la arquitectura técnica, un paso muy importante es el modelo de despliegue, el cual representa la correspondencia entre la arquitectura *software* y la arquitectura *hardware*. Este modelo se utilizó como entrada fundamental en las actividades de diseño e implementación, debido a que la distribución del sistema tiene una influencia principal en su diseño. Como se muestra en la figura 8 se cuenta con un servidor en el cual están contenidas la aplicación y la base de datos. Este servidor será accedido desde las máquinas clientes a través de un navegador *Web* para procesar las peticiones realizadas por los usuarios.

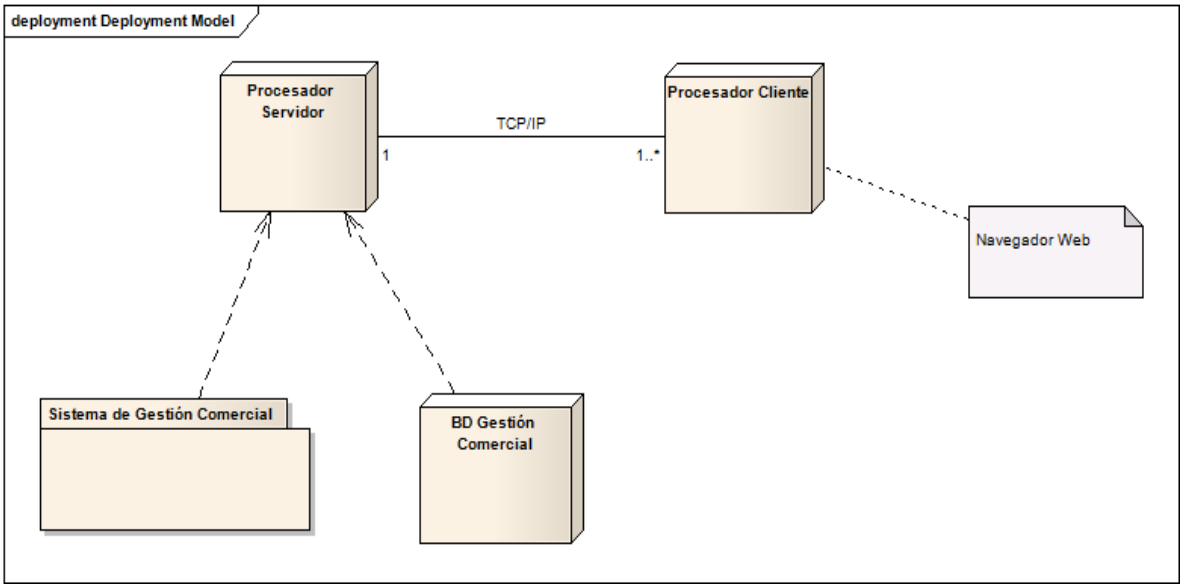

Figura 10 Diagrama de despliegue.

2.3 Diseño detallado

El análisis y diseño preliminar solo era una teoría de cómo funcionarían las clases de la aplicación, al contrario que con el diseño detallado se logra asignar funciones a cada una de las clases que fueron detectadas. Además es donde se empiezan a ver qué métodos llevarán las clases del sistema. Esto se debe a que, hasta ahora, solo se ha interactuado con los objetos de las clases, con los actores y con otros objetos de manera dinámica, por lo que se tiene suficiente información como para poder empezar a especificar los métodos de las respectivas clases[39].

2.3.1 Diagramas de secuencia

A partir de los diagramas de casos de uso y de los diagramas de robustez se define gran parte de los atributos de las clases, pero no es hasta los diagramas de secuencias donde se empiezan a ver qué métodos llevarán las clases del sistema que se desarrolla, así como las interacciones entre objetos durante el tiempo de vida del caso de uso[39]. *Iconix* propone realizar un diagrama para cada caso de uso, facilitando entender su diseño.

Los objetivos elementales del diagrama de secuencia son:

- ✓ Asignar comportamiento a las clases

- ✓ Mostrar en detalle, cómo las clases interactúan entre sí durante el tiempo de vida del caso de uso
- ✓ Terminar la distribución de las operaciones entre clases

A continuación se representan algunos diagramas de secuencia de los casos de uso antes mencionados. En el **Anexo IV** se pueden ver otros diagramas de secuencias y el resto se encuentran en el fichero *EAP*.

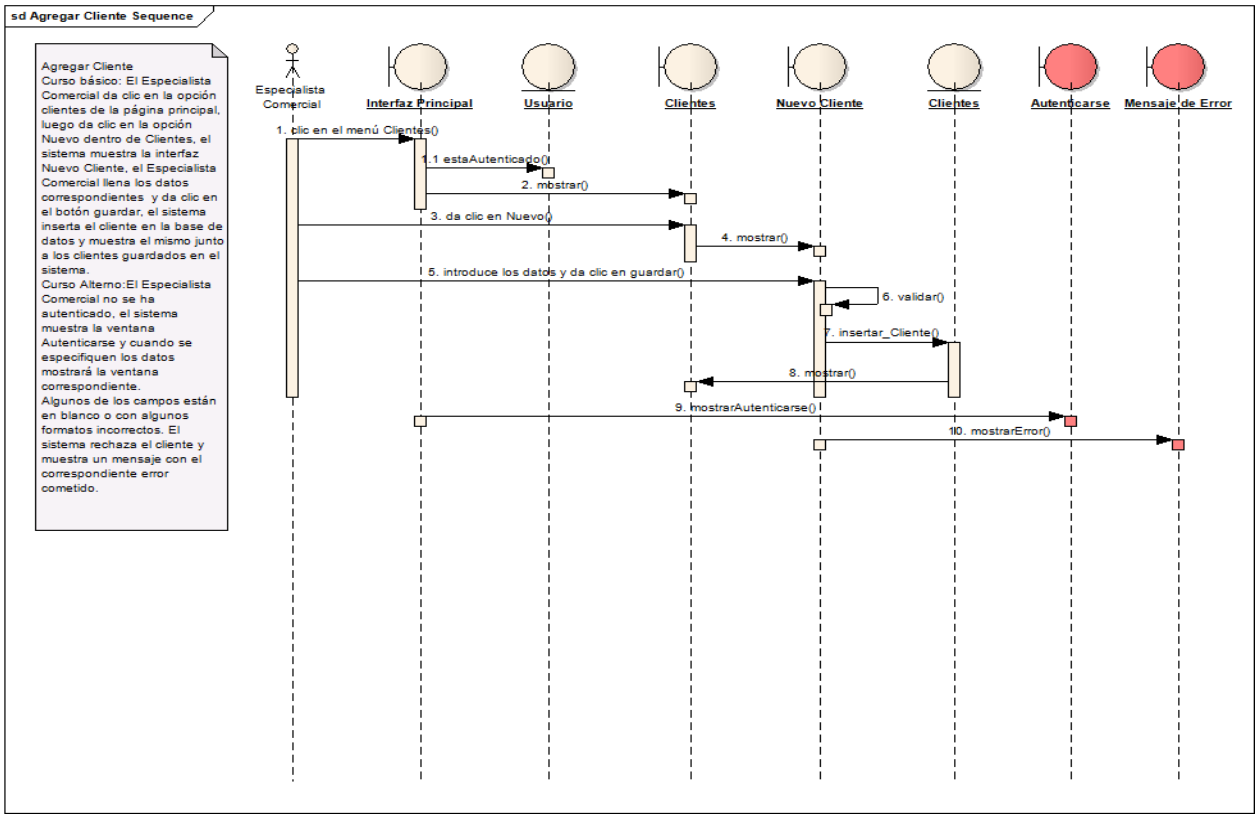

Figura 11 Diagrama de secuencia Agregar Cliente

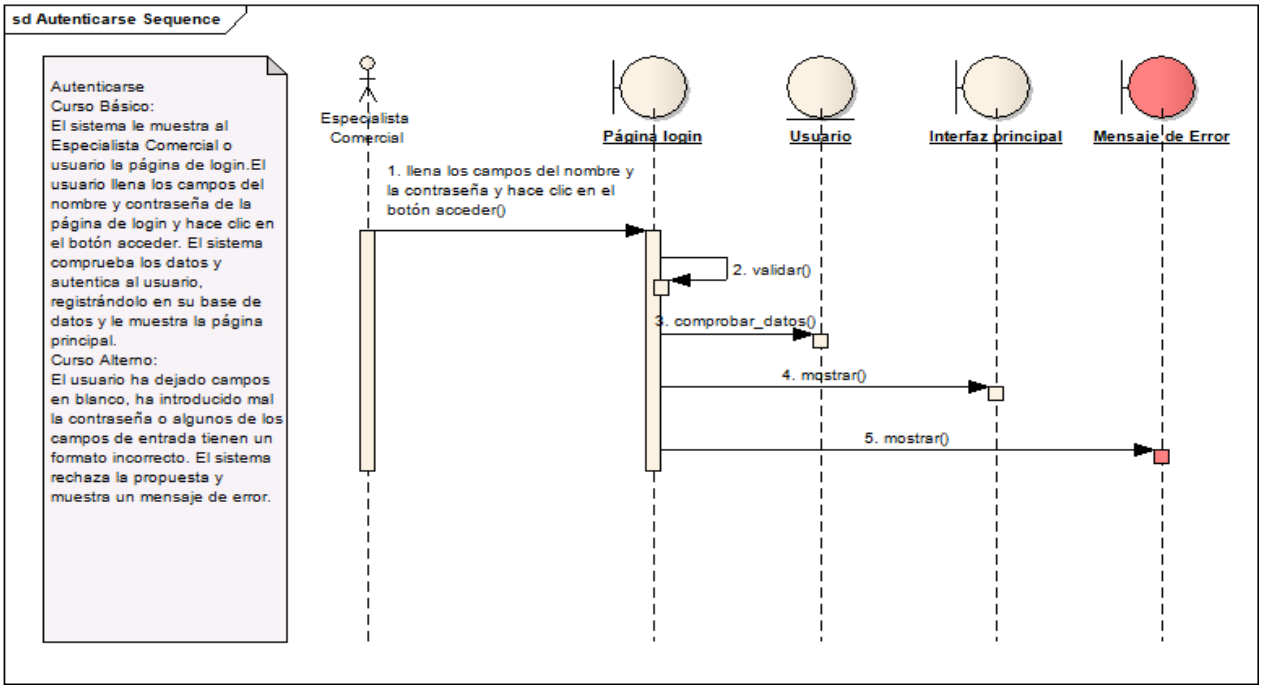

Figura 12 Diagrama de secuencia Autenticarse

2.3.2 Diagrama de clases persistentes

Las clases persistentes guardan su estado en un medio permanente y representan además un modelo lógico de la base de datos, formado por las tablas que permanecen en la misma. El diagrama de clases persistentes del sistema modela la información que trasciende en el tiempo, incluso después de cerrada la aplicación[42].

Además describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. (Ver diagrama en el **Anexo V**).

2.3.3 Modelo de Datos

El modelo de datos describe de manera abstracta la representación de los datos persistentes que se utilizan en el sistema. Además poseen una interacción con las clases servidoras que definen la aplicación y todos los accesos físicos que se realizan en la Base de Datos. (Ver diagrama en **Anexo VI**)

2.3.4 Arquitectura del sistema

La arquitectura del *software* identifica los elementos más importantes de una aplicación, así como sus relaciones. Este es necesario para entender el sistema, organizar su desarrollo. Esta representa la interacción entre las diferentes capas de la aplicación. Durante el diseño de la arquitectura del *software* se toman las decisiones importantes en cuanto a qué *framework* de persistencia utilizar, qué Sistema Gestor de Base de Datos, entre otros, y cómo se relacionarán unos con otros a nivel conceptual.

Patrón de arquitectura de *software*

"Un patrón de arquitectura de *software* describe un problema particular y recurrente del diseño, que surge en un contexto específico y presenta un esquema genérico y probado de su solución." [43]

Modelo Vista Controlador (*MVC*) como patrón de arquitectura de *software*

Modelo Vista Controlador es un patrón de arquitectura de *software* que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Frecuentemente se ve en aplicaciones *Web*, donde la vista es la página *HTML* y el código es el que provee de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y el controlador representa la lógica de negocio.

- ✓ **Modelo**: es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. Encapsula los datos y las funcionalidades
- ✓ **Vista**: es usualmente la interfaz de usuario, es decir muestra la información al mismo. Presenta el modelo en un formato adecuado para interactuar
- ✓ **Controlador**: Responde a eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc., usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases.

Esta separación permite construir y probar el modelo, independientemente de la representación visual. El patrón de arquitectura de *software MVC* tiene flexibilidad para

cambiar las vistas y los controladores. Además, tiene vista sincronizada y se adapta al cambio. El sistema se organizó por este patrón de arquitectura, ya que el *framework* de desarrollo *Web Django* sobre el cual está desarrollada la aplicación, lo implementa por defecto.

2.4 Implementación del Sistema

La implementación se realizó para materializar el diseño y análisis de los requerimientos planteados en el problema y los que surgieron durante las fases anteriores. Durante esta fase, se generó el código fuente necesario para el funcionamiento correcto del sistema. Además, se realizaron las integraciones necesarias siguiendo la arquitectura diseñada explicada en apartado anterior. Una cuestión importante que influyó en la reducción del tiempo de implementación fue la utilización del *Framework Django*.

2.4.1 Estándares de Código

Para la implementación del sistema se tiene en cuenta la calidad del *software*, para la facilidad de mantenimiento del mismo, el cual puede resultar realmente complejo y costoso. La comprensión del código constituye un aspecto esencial para el correcto mantenimiento del sistema. Es por esto que se crean los estándares de codificación que aseguren la legibilidad del código, en *Python* se usa el *PEP (Python Enhancement Proposal)* una guía de estilo que además de garantizar la legibilidad lo hace consistente.

Indentación: Usa 4 espacios por cada nivel de indentación. Para código realmente antiguo que no se quiera estropear, se puede continuar usando tabuladores de 8 espacios.

Tabuladores o espacios: Nunca se mezclan tabuladores y espacios. La forma más popular de indentar en *Python* es utilizar sólo espacios. La segunda forma más popular es usar sólo tabuladores.

Tamaño máximo de línea: Se limitan todas las líneas a un máximo de 79 caracteres. Para cadenas de texto largas (cadenas de documentación o comentarios), es aconsejable se limitan a 72 caracteres.

Líneas en blanco: Las funciones no anidadas y las definiciones de clases se separan con dos líneas en blanco. Las definiciones de métodos dentro de una misma clase se separan con una línea en blanco. Se pueden usar líneas en blanco extra (de forma

reservada) para separar grupos de funciones relacionadas. Las líneas en blanco se pueden omitir entre un grupo de funciones con una sola línea (por ejemplo, con un conjunto de funciones sin implementación). Se usan líneas en blanco en las funciones de forma limitada para indicar secciones lógicas. *Python* acepta el carácter control-L (o lo que es lo mismo ^L) como un espacio en blanco.

***Imports*:** Los *imports* se colocan siempre en la parte superior del archivo, justo después de cualquier comentario o cadena de documentación del sistema, y antes de las variables globales y las constantes del sistema.

Los *imports* se agrupan siguiendo el siguiente orden:

1. *imports* de la librería estándar.
2. *imports* de proyectos de terceras partes relacionados.
3. *imports* de aplicaciones locales*/imports* específicos de la librería.

Espacios en blanco en expresiones y sentencias: Se debe evitar espacios en blanco extra inmediatamente después de entrar en un paréntesis o antes de salir de un paréntesis, corchete o llave. Inmediatamente antes de una coma, punto y coma, o dos puntos. Inmediatamente antes de abrir un paréntesis para una lista de argumentos de una llamada. Inmediatamente antes de abrir un paréntesis usado como índice o para particionar (*slicing*). Más de un espacio alrededor de un operador de asignación (u otro operador) para alinearlo con otro.

Comentarios: Los comentarios deberán ser frases completas. Si un comentario es una frase o sentencia, la primera palabra debería estar en mayúsculas, a menos que sea un identificador que comience con una letra en minúsculas. Si un comentario es corto, se puede omitir el punto al final. Los comentarios de bloque generalmente consisten en uno o más párrafos construidos con frases completas, y cada frase debería terminar con un punto.

Nombres a Evitar: Nunca se utilizan los caracteres 'l' (letra ele minúscula), 'O' (letra o mayúscula), o 'I' (letra i mayúscula) como nombres de variables de un solo carácter. En algunas fuentes, estos caracteres son indistinguibles de los numerales uno y cero. Cuando se está tentado de usar 'l', se debe utilizar una 'L' en su lugar.

Nombres de Paquetes y Módulos: Los módulos deben tener nombres cortos formados en su totalidad por letras minúsculas. Se puede utilizar guiones bajos en el nombre del módulo si mejora la legibilidad. Los paquetes *Python* también deben tener nombres cortos formados de letras minúsculas, aunque se desaconseja el uso de guiones bajos. Dado que los nombres de los módulos se mapean a nombres de archivos, y algunos sistemas de ficheros no diferencian entre mayúsculas y minúsculas y truncan los nombres largos.

Nombres de Excepciones: Dado que las excepciones deberían ser clases, se aplica la convención relativa al nombrado de clases. De todas formas, se debería usar el sufijo "Error" en los nombres de las excepciones (si la excepción es realmente un error).

Nombres de Variables Globales: Las convenciones son prácticamente las mismas que para las funciones. Los módulos diseñados para ser utilizados usando "*from M import* *" se debe usar el mecanismo *__all__* para prevenir que se exporten variables globales, o bien usar la convención antigua de añadir un guión bajo como prefijo a dichas variables globales.

Nombres de Funciones: Los nombres de funciones deben estar en letras minúsculas, con palabras separadas mediante guiones bajos según sea necesario para mejorar la legibilidad. Sólo se acepta capitalización mezclada en contextos en los que ya se trate del estilo principal (por ejemplo *threading.py*), para mantener la compatibilidad hacia atrás.

Argumentos de funciones y métodos: Se utiliza siempre *'self'* como primer argumento de los métodos de instancia. Se utiliza siempre *'cls'* como primer argumento de métodos de clase.

Nombres de métodos y variables de instancia: Se deben utilizar las reglas de los nombres de funciones: minúsculas con palabras separadas por guiones bajos cuando sea necesario para mejorar la legibilidad. Utilizar guiones bajos al inicio sólo para métodos no públicos y variables de instancia. Para evitar colisiones de nombres con subclases, utilizar dos guiones bajos al principio del nombre para invocar las reglas de planchado de nombres de *Python*.

Diseñar para la herencia: Se debe decidir siempre si los métodos y variables de instancia de una clase (llamados de forma colectiva "atributos") deberían ser públicos o no públicos.

Por último mencionar que debido a la utilización del *framework Django* el cual posibilita que el código para definir y acceder a los datos (el modelo) se encuentre separado de la lógica de negocios (el controlador) y a su vez está separado de la interfaz de usuario (la vista) permite la legibilidad y facilidad de la comprensión del mismo. Además al contar con una gran parte del código implementado proporciona gran facilidad ya que este se reutilizará haciendo el trabajo más sencillo.

2.5 Pruebas al sistema

La prueba es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.

La prueba de *software* es un elemento crítico para la garantía de la calidad del *software* y representa una revisión final de las especificaciones del diseño y de la codificación. Demuestran que el sistema se corresponde adecuadamente con las especificaciones realizadas.

Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final.

Los casos de pruebas deben verificar si el producto satisface los requerimientos del usuario, tal y como se describe en las especificación de los requerimientos. Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

La etapa de prueba es tan o más importante que todas las realizadas hasta el momento puesto que en ella se refleja la calidad con que ha sido llevada a cabo la proyección del sistema.

Caso de Uso	Autenticarse
Caso de Prueba	Autenticar usuario que no se encuentra registrado.
Entrada	El usuario introduce los datos en los campos del formulario: Usuario: Ramiro Contraseña: **********
Salida	El sistema muestra un mensaje de error en el cual informa que el usuario y la contraseña insertada no han sido registrados aún.

Tabla 5 Caso de prueba: Autenticarse con un usuario que no existe en el sistema.

Caso de Uso	Autenticarse
Caso de Prueba	Autenticar usuario con errores en el nombre o en la contraseña.
Entrada	El usuario introduce los datos en los campos del formulario: Usuario: Ramiro Contraseña: **********
Salida	El sistema muestra un mensaje al usuario informándole que su nombre o su contraseña son incorrectos.

Tabla 6 Caso de prueba: Autenticar usuario con errores en el nombre o en la contraseña.

Caso de Uso	Agregar Cliente
Caso de prueba	Agregar un cliente con campos en blanco.
Entrada	El Especialista Comercial en la interfaz Agregar Cliente inserta la dirección: Calle H % General Marrero y Ave. Cárdenas #4231, el código REUP: 104, deja en blanco el campo nombre y da clic en agregar.

Salida	El sistema muestra un mensaje de error al Especialista Comercial en el cual informa que el cliente debe tener un nombre.

Tabla 7 Caso de Prueba: Agregar un cliente con campos en blanco.

Caso de Uso	Agregar Cliente
Caso de Prueba	Agregar un cliente que ya existe.
Entrada	El Especialista Comercial en la interfaz Agregar Cliente inserta el nombre: Copextel, la dirección: Calle H % General Marrero y Ave. Cárdenas #4231, el código REUP: 104 y da clic en agregar.
Salida	El sistema muestra un mensaje de error al Especialista Comercial en el cual informa que el cliente ya ha sido registrado.

Tabla 8 Caso de Prueba: Agregar un cliente que ya existe.

2.5.1 Pruebas en Django

Las pruebas automáticas son una herramienta extremadamente útil para la eliminación de errores para el desarrollador *Web* moderno. Puedes utilizar una colección de pruebas – una *test suite* - para solucionar, o evitar un número de problemas:

- ✓ Cuando estás escribiendo un nuevo código, puedes utilizar pruebas para validar que tu código trabaja como esperas
- ✓ Cuando estás cambiando o modificando un viejo código, puedes utilizar pruebas para asegurarte que tus cambios no han afectado la conducta de tu aplicación inesperadamente

Probar una aplicación *Web* es una tarea compleja, porque una aplicación *Web* está hecha de varias capas de lógica -desde el manejo de peticiones al nivel *HTTP*, hasta la validación y procesamiento de formularios, al renderizado de plantillas. Con el *framework* de ejecución de pruebas *Django* y las herramientas asociadas, puedes simular peticiones, insertar datos de prueba, inspeccionar la salida de tu aplicación y

generalmente verificar que tu código este haciendo lo que debería. La mejor parte es que es realmente fácil.

La parte preferida de escribir pruebas en Django es utilizar el módulo *unittest* que se encuentra dentro de la librería estándar de *Python. Django* provee un *API* y herramientas para ese tipo de integración.

Los *test* en *Django* cubren los modelos creados de la base de datos y las *apps* instaladas de terceros. Para escribir un *test* debe heredar de la clase *TestCase*. Para ejecutar los *test* se usará el comando "*python manage.py test*".

El *test* se construye con una clase que herede de *testCase*, el *test* debe escribirse en un archivo *test.py* dentro de la *app*, en este caso principal. Dentro de la clase estarán el método *setUp* en el que se crearán los objetos que van a probar los modelos creados. Una vez creado el método *setUp* y los objetos a estudiar se crean tantos métodos como se necesiten para evaluar los modelos, sabiendo que si algún método falla lo que devolverá será el nombre del método.

2.6 Valoración de sostenibilidad

La valoración de sostenibilidad de un producto informático no es más que el proceso de evaluación de impactos ambientales, socio-humanistas, administrativos y tecnológicos de un producto informático, previsibles desde el diseño del proyecto, que favorece su autorregulación, para la satisfacción de la necesidad que resuelve, con un uso racional de recursos y la toma de decisiones adecuadas a las condiciones del contexto y el cliente [44].

Además permite prever posibles daños o problemas que puede ocasionar el uso de la aplicación informática con impacto ambiental en la entidad teniendo en cuenta las necesidades que el mismo puede resolver. Luego de este estudió se determinará si el producto informático que se desarrolla es sostenible o no.

2.6.1 Dimensión Administrativa

Esta dimensión se sustenta en el ahorro, gasto, calidad de la producción y los servicios, administración de recursos y en la toma de decisiones administrativas que brinda como resultado el PI.

La estimación por Puntos de Caso de Uso es el método empleado para estipular el esfuerzo que requiere el desarrollo del *software*, mediante una aproximación a los primeros casos de usos desarrollados. Este resulta efectivo para este proyecto pues la metodología que sustenta su desarrollo está basada en iteraciones continuas donde los primeros casos de uso son los que permiten diseñar la mayor parte de la arquitectura del *software* y los que a su vez ayudan a mitigar los riesgos más significativos. Este método ayuda a estimar el tiempo de desarrollo de un PI mediante la asignación de "pesos" a cierto número de factores que este plantea tener en cuenta, para posteriormente, contabilizar el tiempo total estimado para el PI a partir de dichos factores.

El paso inicial para la estimación consiste en el cálculo de los Puntos de Casos de Uso sin ajustar (UUCP). Este valor se calcula a partir del Factor de Peso de los Actores sin ajustar (UAW) y el Factor de Peso de los Casos de Uso sin ajustar (UUCW).

Estos valores se calculan mediante un análisis de la cantidad de actores y casos de uso presentes en el sistema y la complejidad de cada uno de ellos respectivamente.

Una vez que se tienen los Puntos de Casos de Uso sin ajustar, se debe calcular los Puntos de Casos de Uso ajustados (UCP) teniendo en cuenta los Puntos de Casos de Uso sin ajustar (UUCP), el Factor de complejidad técnica (TCF) y el Factor de complejidad ambiental (ECF).

El TCF se calcula mediante la cuantificación de un conjunto de factores que determinan la complejidad técnica del sistema, como se muestra en la tabla.

Factor	**Descripción**	**Peso**	**Valor**	**TCF**
TCF01	Sistema Distribuido	2.00	0.00	0.00
TCF02	Objetivos de performance o tiempo de respuesta	1.00	1.00	1.00
TCF03	Eficiencia del usuario final	0.50	2.00	1.00
TCF04	Procesamiento interno complejo	1.00	2.00	2.00

TCF05	El código debe ser reutilizable	1.00	4.00	4.00
TCF06	Facilidad de instalación	0.50	2.00	1.00
TCF07	Facilidad de uso	0.50	5.00	2.50
TCF08	Portabilidad	1.00	0.00	0.00
TCF09	Facilidad de cambio	1.00	1.00	1.00
TCF10	Concurrencia	0.50	2.00	1.00
TCF11	Incluye objetivos especiales de seguridad	1.00	5.00	5.00
TCF12	Provee acceso directo a terceras partes	1.00	3.00	3.00
TCF13	Se requieren facilidades especiales de entrenamiento a usuarios	1.00	1.00	1.00
			Total:	**22.50**

Tabla 9 Factores de Complejidad técnica.

El Factor de complejidad técnica se calcula de la siguiente manera:

Factor	**Valor**
Valor desajustado de TCF (UTV)	22.50
Pesos TCF (TWF)	0.01
Constante TCF (TC)	0.60
TCF = TC + (UTV * TWF)	**0.82**

Tabla 10 Cálculo del TCF.

Las habilidades y el entrenamiento del grupo involucrado en el desarrollo tienen un gran impacto en las estimaciones de tiempo. Estos factores son los que se contemplan en el cálculo del ECF. En la siguiente tabla se muestra los factores que se tienen en cuenta en el cálculo del mismo.

Factor	Descripción	Peso	Valor	TCF
ECF01	Familiaridad con el modelo de proyecto utilizado	1.50	4.00	6.00
ECF02	Experiencia en la aplicación	0.50	3.00	1.50
ECF03	Experiencia en orientación a objetos	1.00	5.00	5.00
ECF04	Capacidad del analista líder	0.50	3.00	1.50
ECF05	Motivación	1.00	5.00	5.00
ECF06	Estabilidad de los requerimientos	2.00	4.00	8.00
ECF07	Personal parte de tiempo	-1.00	0.00	0.00
ECF08	Dificultad del lenguaje de programación	-1.00	0.00	0.00
			Total:	**27.00**

Tabla 11 Factor ambiental.

El Factor de complejidad ambiental resulta de la siguiente manera:

Factor	Valor
Valor desajustado ECF (UEV)	27.00
Peso ECF (EWF)	-0.03

Constante ECF (EC)	1.40
ECF = EC + (UEV * EWF)	**0.59**

Tabla 12 Cálculo del ECF.

Para determinar la cantidad de hombres por caso de uso se contabilizan cuántos factores de los que afectan al ECF están por debajo del valor medio (3), para los factores ECF1 a ECF6. Luego se contabilizan cuántos factores de los que afectan al ECF están por encima del valor medio (3), para los factores ECF7 y ECF8. Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.

Una vez calculado todos los factores que se tienen en cuenta para el Análisis de Puntos de Casos de Uso se obtuvieron los resultados siguientes:

Pasos	**Valor**
Total de Casos de Uso	22
UUCP	110.00
TCF	0.82
ECF	0.59
UCP = (UUCP * TCF * ECF)	53.00
CF	10.00
Esfuerzo en hora-hombres (E = CF * UCP)	530.00
Costo Total	7950.00

Tabla 13 Estimación del esfuerzo y costo.

Analizando los resultados mostrados en la tabla 13 se puede concluir que se emplearan 20 horas-hombres para realizar un caso de uso, obteniéndose un esfuerzo de 530 horas-hombres, para un costo total de 7950 pesos o 318.00 cuc. Este valor está

dado fundamentalmente por el tiempo dedicado al mismo el cual disminuyó considerablemente como resultado del empleo del *framework Django* el cual aporta un ciclo de desarrollo muy ágil.

El gasto en recursos informáticos es mínimo pues la infraestructura que sirve de soporte al proyecto ya estaba creada. El sistema informático más que minimizar la mano de obra la optimiza. Aporta mejoras significativas para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca, disminuyendo el trabajo manual de los tramitadores y favoreciendo el flujo de trabajo de los usuarios que interactúan con el *software*. Proporciona además una disminución considerable en los tiempos de búsqueda y garantiza la perdurabilidad de información valiosa. No se incurren en gastos de electricidad asociado al uso de las computadoras y el servidor porque esta situación ya se presentaba.

Las herramientas utilizadas para el desarrollo del *software* en su totalidad son libres, con lo que se absuelve a la Empresa de Acueducto y Alcantarillado Guardalavaca de incurrir en gastos para llevar a cabo el desarrollo del proyecto. A partir de lo analizado y los beneficios que proporciona el *software*, se arribó a la conclusión que éste es sostenible desde la dimensión administrativa.

2.6.2 Dimensión Socio-Humanista.

La satisfacción de los trabajadores es un factor esencial para lograr la calidad de sus servicios, por lo que actualmente las entidades hacen énfasis en lograr su bienestar. La automatización de los procesos proporciona el bienestar de estos en el desempeño laboral a partir de las comodidades brindadas.

El sistema informático propuesto resolverá la necesidad planteada en la Empresa de Acueducto y Alcantarillado Guardalavaca, pues facilitará que el proceso de gestión comercial en el sector estatal de la entidad se realice de una forma cómoda propiciando así que disminuya el tiempo que se emplea en las tareas que se ejecutan. Con el uso del sistema, la carga de trabajo se reducirá, al realizar las labores automáticamente. Estas mejoras en las condiciones de trabajo darán lugar a una gran satisfacción del personal implicado, lo que favorecerá una mejor calidad en el proceso productivo. Se garantizará que los reportes tanto diarios como mensuales se realicen de forma

correcta, segura y en un tiempo breve. Se tuvo en cuenta el rechazo al cambio que podía surgir al implantar el sistema, lo cual se considera normal, debido a la tendencia del ser humano a hacer costumbre de lo habitual y rutinario y a la resistencia inconsciente ante los cambios de su entorno. Para favorecer la aceptación del sistema se llevaron a cabo entrevistas con los usuarios potenciales, para explicarles en profundidad las ventajas que el sistema proporcionaría y cómo el sistema podía serles fácil y cómodo de usar. El sistema se desarrolló en una interfaz con ambiente web. El flujo de trabajo inmerso en el sistema será similar al que solían realizar anteriormente, es decir, lo menos complejo posible, de forma tal que los usuarios no se pierdan mientras trabajaban sobre él y no lo rechacen ante el cambio. El sistema hizo aportes a la ciencia y la tecnología, ya que no existían en la entidad otras herramientas que facilitaran este trabajo, logrando así mejoras en las tareas. A partir de lo analizado anteriormente, se arribó a la conclusión de que el sistema es sostenible desde la dimensión socio-humanista.

2.6.3 Dimensión Ambiental.

La creación de una cultura y una conciencia ambiental se hace cada día una tarea más ardua, aumentan considerablemente los productos que no tienen en cuenta su impacto negativo en la naturaleza, por lo que la presente investigación plantea una solución razonable ambientalmente.

En la Empresa de Acueducto y Alcantarillado Guardalavaca existen algunas condiciones de trabajo para la correcta explotación del sistema, como son: protectores de pantalla, ubicación correcta de la PC, asientos cómodos y con medidas adecuadas para evitar daños a la salud que serían mayoritariamente daños en la vista y en la postura de los implicados, efectos provocados por el tiempo en que deben estar intercambiando información.

En este aspecto también se tiene en cuenta el uso de colores e imágenes que resulten agradables y eviten el estrés; las correctas alineaciones de textos y otros elementos en las interfaces como: iluminación, tamaño de letra, espaciamiento entre caracteres, tipografía. Todo esto ayuda a evitar el cansancio visual, los daños en la columna y el estrés de los usuarios al minimizar el tiempo frente al monitor, por lo que se favorece la

calidad en las labores productivas. Debido a las características que presenta, no genera contaminación por ruido, ni tiene un impacto directo desfavorable en el medio ambiente.

Se podrán reutilizar los componentes y recursos del sistema, debido a las características de las tecnologías empleadas y a que se tuvo en cuenta la generalidad en el diseño del mismo. A partir de lo analizado anteriormente se arribó a la conclusión de que el sistema es sostenible desde la dimensión ambiental.

2.6.4 Dimensión Tecnológica.

Para el empleo del sistema los usuarios se encuentran capacitados y no necesitan de preparación informática, debido a que hacen uso de computadoras para su labor diaria; además el sistema será intuitivo y fácil de usar y poseerá una ayuda y un manual de usuario que facilitará su manipulación.

En cuanto a la infraestructura electrónica, la Empresa de Acueducto y Alcantarillado Guardalavaca cuenta con los recursos precisos para la implantación y aplicación del sistema. Las características que poseen las computadoras utilizadas por los trabajadores cumplen con los requerimientos necesarios para hacer uso del sistema y además se encuentran conectadas a la red. Con el fin de facilitar el mantenimiento del sistema, se definió un estándar de código, se describieron con comentarios las funciones fundamentales, además, se le pusieron nombres intuitivos para proporcionar una mejor comprensión y entendimiento. Por otra parte, el sistema permitirá su evolución en el tiempo. Además, permitirá cambios, ya sea de mejoras de hardware, red e incluso de plataforma. A partir de lo analizado anteriormente se arribó a la conclusión de que el sistema es sostenible desde la dimensión tecnológica.

2.6.5 Conclusiones de la VSPI

Se valoró la sostenibilidad en la dimensiones administrativa, socio-humanista, ambiental y tecnológica del sistema informático, llegando a la conclusión que la solución empleada es factible y el sistema es sostenible. El producto informático se mantendrá en el tiempo por la perdurabilidad de la necesidad que existe de una herramienta informática que les permita gestionar de una forma rápida y fiable la

información referente a los procesos de contratación, verificación de consumos, facturación y cobro a los clientes.

2.7 Valoración de los resultados obtenidos en la encuesta a los posibles usuarios del sistema.

Una vez concluida la implementación del sistema propuesto así como las pruebas correspondientes, se aplicaron encuestas con el objetivo de evaluar la satisfacción de los usuarios respecto al sistema, mediante el método *Delphi*.

Para la aplicación de este método se realizó una primera encuesta (**Anexo VII**), que tuvo como finalidad seleccionar los expertos, y obtener a partir de su procesamiento, el coeficiente de competencia.

Teniendo en cuenta una serie de aspectos propuestos por el método para la selección de los expertos se escogieron 15 expertos de una población de 18 encuestados. Una vez seleccionados los expertos, se sometió el sistema a evaluación a través de una encuesta (**Anexo VIII**), la cual se procesó por el método *Delphi* (**Anexo IX**) con el objetivo de buscar el consenso de los encuestados en los aspectos:

1. ¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?
2. ¿Cómo evalúa el tiempo de respuesta del sistema?
3. ¿Cómo aprecia el diseño de las interfaces del sistema?
4. ¿Cree correcto el uso de los colores en las interfaces?
5. ¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?
6. ¿Cuánto contribuirá el sistema a la toma de decisiones?
7. ¿Cómo evalúa las facilidades que brinda para la gestión de los servicios de ofertas y ventas?
8. ¿Cómo evalúa la estructura organizativa del sistema?
9. ¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?

A partir del procesamiento estadístico de los aspectos anteriormente mencionados y tratados en la encuesta, se determinó, como resultado final de la evaluación, que los encuestados coinciden en que todos los aspectos son —Muy Adecuados, lo cual influye positivamente en la calidad del sistema.

2.8 Conclusiones del capítulo

La utilización de la metodología *Iconix* para modelar y construir el sistema propuesto, permitió resolver una gran parte de su desarrollo, sin descartar ninguna de las etapas que lo componen, logrando desarrollar eficientemente el diseño del problema planteado en la investigación. Se concluyó que la solución propuesta es factible y el sistema es sostenible. El análisis de sostenibilidad arrojó que el proyecto es sostenible y perdurable en el tiempo debido a la necesidad que existe de una herramienta informática que les permita gestionar de una forma rápida y fiable el flujo de información referente a la gestión comercial en el sector estatal. Mediante la encuesta aplicada, se pudo observar la satisfacción de los usuarios en cuanto a las funcionalidades de la aplicación, las cuales permitieron resolver el problema previamente planteado en la investigación. El criterio de experto permitió validar la hipótesis, por lo que se puede afirmar que esta ha sido cumplida siendo este un elemento más a favor del impacto que tendrá la implantación del mismo.

Conclusiones generales

Como conclusiones generales de la presente investigación, se obtienen las siguientes ideas:

1. El estudio realizado en la Empresa de Acueducto y Alcantarillado Guardalavaca permitió detectar las deficiencias en la gestión de la información en el proceso de gestión comercial, lo que constituyó el punto de partida de la investigación.
2. Se cumplió con el objetivo de la investigación desarrollándose un sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca, brindando un mayor flujo de información en tiempos más cortos, fiabilidad en los datos y la existencia de una interconexión con el portal corporativo de la empresa.
3. Las herramientas y tecnologías seleccionadas constituyeron una buena opción para el desarrollo del sistema.
4. La metodología de desarrollo de *software* empleada resultó fundamental para maximizar los índices de calidad en los procesos de ingeniería de *software* implicados.
5. Con la valoración del sistema en las cuatro dimensiones de gestión de sostenibilidad, quedó demostrado que el producto informático es sostenible y perdurable en el tiempo.
6. El sistema desarrollado presenta un nivel apropiado de aceptación por los usuarios, y una opinión favorable por parte de los expertos, lo cual favorecerá la implantación del mismo.
7. A través de los resultados obtenidos en el criterio de expertos se verificó el cumplimiento de la hipótesis planteada.

Recomendaciones

Una vez concluida esta investigación se proponen las siguientes recomendaciones:

- ✓ Implantar el sistema
- ✓ Monitorear de manera continua el funcionamiento del Sistema Informático y tomar las medidas y acciones correspondientes para su perfeccionamiento
- ✓ Generalizar el uso de la aplicación al resto de las empresas pertenecientes al mismo sector de la entidad

Glosario de términos

Cliente/Servidor: Arquitectura de sistemas de información en la que los procesos de una aplicación se dividen en componentes que se pueden ejecutar en máquinas diferentes. Modo de funcionamiento de una aplicación en la que se diferencian dos tipos de procesos y su soporte se asigna a plataformas diferentes.

Base de Datos: Una Base de Datos consta de una colección de tablas que contienen datos y otros objetos, como vistas, índices, procedimientos almacenados y desencadenadores, que se definen para poder llevar a cabo distintas operaciones con datos. Los datos almacenados en una Base de Datos suelen estar relacionados con un tema o un proceso determinados, como por ejemplo, la información de inventario para el almacén de una fábrica.

***Framework*:** Un *framework,* es una estructura de soporte definida mediante la cual otro proyecto de *software* puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros *software* para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

***Hardware*:** Componentes electrónicos, tarjetas, periféricos y equipo que conforman un sistema de computación. Se distinguen de los programas (*software*) porque éstos le indican a los componentes mencionados lo que deben hacer.

***HTML* (*Hypertext Markup Language*):** Lenguaje usado para escribir documentos para servidores *World Wide Web*. Es una aplicación de la *ISO Standard* 8879:1986.

***HTTP* (*Hypertext Transport Protocol*):** Conjunto de especificaciones para el intercambio de ficheros (texto, gráfico, imagen, sonido, vídeo) en la *Web*.

Interfaz: Parte de un programa informático que permite a éste comunicarse con el usuario o con otras aplicaciones permitiendo el flujo de información.

Sistema: Conjunto de elementos interrelacionados que trabajan juntos para obtener un resultado deseado.

***WEB* (WWW):** Red de documentos *HTML* intercomunicados y distribuidos entre servidores del mundo entero.

Referencias Bibliográficas

1. Purón, C.Y.Z., *Sistema para la Gestión de la Capacitación en la Empresa 'Comandante Ernesto Che Guevara'. Subsistemas de implementación y evaluacion*, 2012, Universidad de Holguín "Oscar Lucero Moya".

2. Parra, Y.A., *Módulo para la Gestión de Trámites de la Dirección de Relaciones Internacionales de la Universidad de Holguín 'Oscar Lucero Moya*, 2013, Universidad de Holguín "Oscar Lucero Moya".

3. Cohen Karen, D., *Sistemas de Información Gerencial*2000, Mc Graw Hill

4. Rodríguez, M.d.C., *Sistema de gestión de información Editorial. Semanario ¡AHORA!* , 2008, Universidad de Holguín¨Oscar Lucero Moya¨.

5. *plataforma ROR*. 2012 [cited 2014 3-3]; Available from: http:// www.plataformaROR.com.

6. Mir, R., *Sistema de Gestión de la Información de los Servicios de Ventas a Turistas en Infotur, Holguín*, 2010, Universidad de Holguín ¨Oscar Lucero Moya¨.

7. *aplicaciones-web-vs-aplicaciones-de-escritorio*. 2011 [cited 2014 14-3]; Available from: http://www.webprogramacion.com.

8. Alvarez, M.A. *Qué es Python*. 2003 [cited 2014 15-2]; Available from: http://www.desarrolloweb.com/articulos/1325.php.

9. Día, C.R., *"Python y sus ventajas"*2008.

10. Duvallier. *Aptana uno de los mejores IDE multiplataforma*. 2010 [cited 2014 23-2]; Available from: http://www.aclibre.org/archives/Aptana/Aptana.html.

11. Gutiérrez, J.J., *¿ Qué es un framework web ?* pp. 1–4.

12. Eguiluz, J., *Desarrollo Web ágil con Symfony2*. pp. 1-6.

13. E, C.B.D., *Symfony.* 2012.

14. Zarete, S., *Django el Framework Web Definitivo*2009.

15. Ilmazz, M.C. *Desarrollo web en Python usando el framework Django*. 2006 2-3-2014; Available from: http://the-geek.org/django-book/chapter1/.

16. *Django web framework _ Django en Español, django-es*. 2010 [cited 2014 28-3]; Available from: http://es.django.net.

17. Org, D.W. *Django*. Available from: http://es.wikipedia.org/wiki/Django.

18. García, J.E. *Django: Framework MVC en python*. 2010 [cited 2014 4-4]; Available from: http://www.whyfloss.com/en/conference/madrid08/getpdf/.

19. *Metodologías_de_desarrollo_del_software*. 2009 [cited 2014 12-1]; Available from: http://www.metodologias.com.

20. Leyva, Y.C., *Sistema de apoyo a la gestión de la información en el centro de Diagnóstico y Orientación del Municipio Holguín*, 2010, Universidad de Holguín ¨Oscar Lucero Moya¨.

21. *Metodologias tradicionales y metodologias ágiles*. 2012 [cited 2014 10-4]; Available from: http://www.metodologia.com.

22. Riehle, D., *A Comparison of the Value Systems of Adaptive Software Development and Extreme Programming : How Methodologies May Learn from Each Other*2000. pp. 1-13.

23. Díaz, A.S., *Sistema de Gestión de Información de apoyo a la toma de decisiones en la Dirección de Relaciones Internacionales de la Universidad de Holguín "Oscar Lucero Moya"*, 2011, Universidad de Holguín "Oscar Lucero Moya".

24. *Programación_Extrema_o_XP*. 2010 [cited 2014 12-3]; Available from: http://www.metodologias-buenaspracticas.com.

25. Rivera, D.E., *Módulo de Configuración del Sistema de Gestión Empresarial Medioambiental (GEMA), para el Sistema Integrado de Gestión (SIG) de la Oficina Territorial de Normalización de Holguín*, 2011, Universidad de Holguín "Oscar Lucero Moya".

26. Cornejo, J.E.G. *Lenguaje de Modelado Unificado.* [cited 2011 17 de marzo]; Available from: http://www.docirs.cl/uml.htm.

27. Stephens, D.R.a.M., *Use Case Driven Object Modeling with UML Theory and Practice*.

28. Toledo, Y.C., *Sistema para el control de la producción científica de la Universidad de Holguín*, 2010, UHO"Oscar Lucero Moya": Holguin.

29. *Herramienta_CASE*. 2010 [cited 2014 12 -1]; Available from: http://www.herramienta_case.com.

30. *Enterprise Architect Information*. 2008 [cited 2014 21-1]; Available from: http://www.enterprise.com.

31. *Enterprise Architect*. 2010 [cited 2014 10-2]; Available from: http://es.architect.net.

32. *Sistema_de_gestión_de_bases_de_datos*. 2012 [cited 2014 12-2]; Available from: http://es.wikipedia.org.

33. Asenjo, J.S., *Sistemas gestores de Base de Datos*.

34. *Sistema Gestor de Base de Datos PostgreSQL*. 2008 [cited 2014 2-4]; Available from: http://www.postgreSQL.com.

35. *postgres vs mysql*. 2011 [cited 2014 17-2]; Available from: http://www.comparative.org.

36. *que-es-un-servidor-web*. 2011 [cited 2014 24-3]; Available from: http://www.web.com.

37. *apache-tomcat, "tomcat_-_introduccion_134*. 2012 [cited 2014 21-3]; Available from: http://es.prmob.net.

38. Menéndez, Y.A.d.l.C., *Definición espacial de redes hidráulicas de abasto en la actividad de proyecto de la EIPH Holguín "Raudal": Módulo de un Sistema CAD distribuido*, 2010, UHO "Oscar Lucero Moya": Holguin.

39. Stephens, D.R.a.M., *Use Case Driven Object Modeling with UML Theory and Practice*2007.

40. Pupo, A.L.S., *Módulo informático para la gestión de la información del SGIE ¡ahora!*, 2010, UHO "Oscar Lucero Moya": Holguin.

41. *Visual Paradigm for UML (ME) - (Paradigma Visual para UML (ME)) (Visual Paradigm for UML (ME)) por Visual Paradigm International Ltd.* 2011; Available from: http://www.visualparadigm.com.

42. Díaz, Z.M., *SISGEBLIO: Sistema para la gestión del plan bibliográficos en los departamentos docentes de la Universidad 'Oscar Lucero Moya' de Holguin*, 2011, UHO "Oscar Lucero Moya": Holguin.

43. Lutz, M.A., *D. Learning Python Language* 1999.

44. Concepción, R., *Procedimiento para la valoración de sostenibilidad de un producto informático* 2006.

Anexo I Diagramas de Casos de Uso

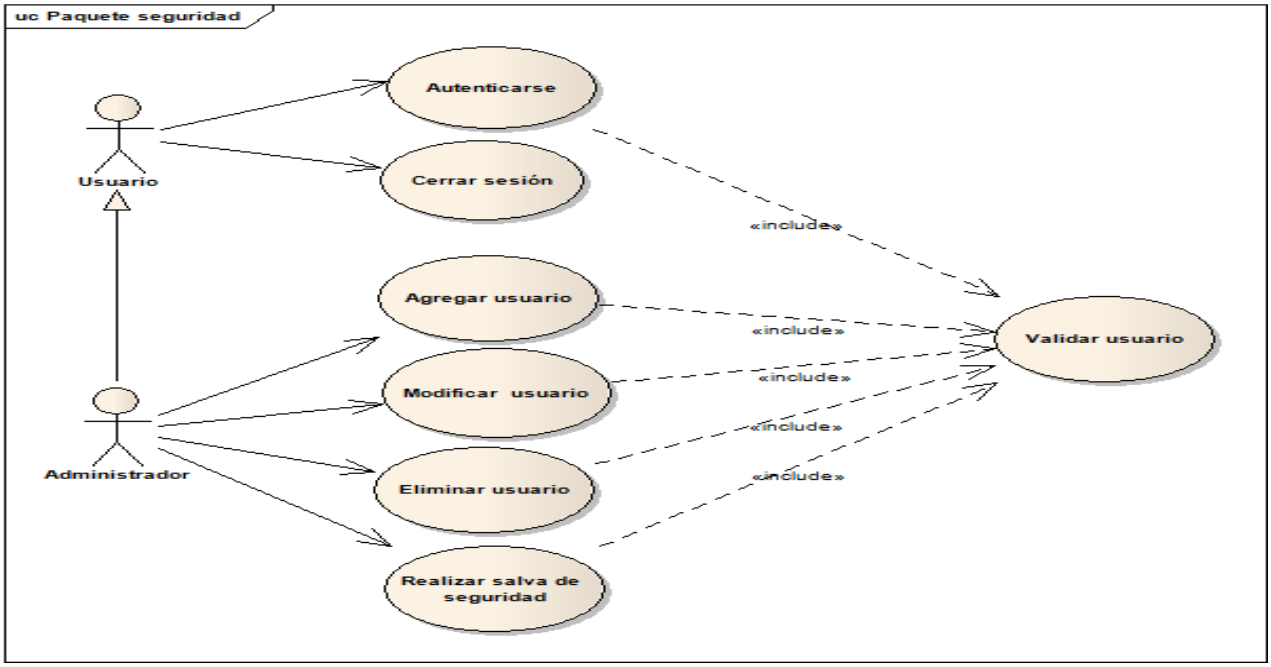

Diagrama de caso de uso del paquete Seguridad.

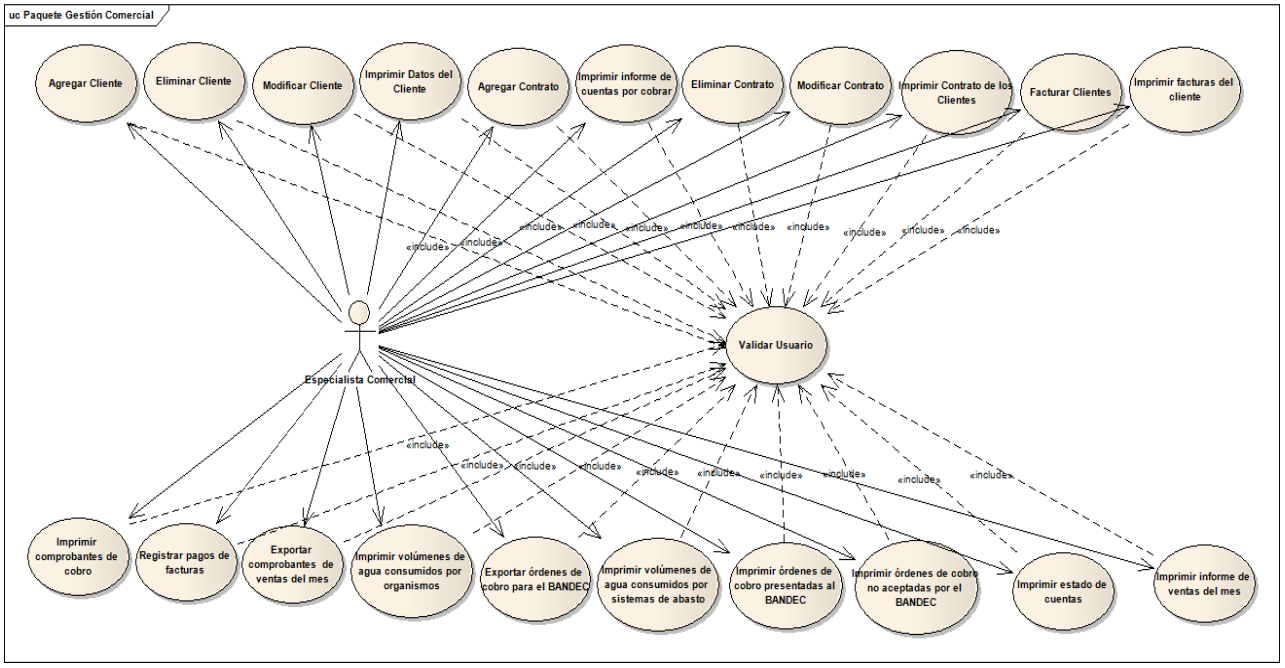

Diagrama de caso de uso del paquete Gestión Comercial.

Anexo II Descripción de los casos de uso

Caso de Uso	Eliminar Cliente
Curso básico	El Especialista Comercial da clic en la opción clientes de la página principal, el sistema muestra la tabla de los clientes, luego se selecciona el cliente y se da clic en la opción eliminar, el sistema muestra la ventana de confirmación de eliminar, el Especialista Comercial da clic en eliminar y se elimina el cliente deseado, el sistema elimina el cliente de la base de datos y muestra los demás clientes guardados en el sistema.
Curso alterno	El Especialista Comercial no se ha autenticado, el sistema muestra la ventana Autenticarse y cuando se especifiquen los datos mostrará la ventana correspondiente. Mensaje de confirmación cancelado: El sistema no elimina el cliente.

Descripción del Caso de Uso Eliminar Cliente

Caso de Uso	Modificar Cliente
Curso básico	El Especialista Comercial da clic en la opción Clientes de la página principal, luego da clic sobre el cliente que desea modificar y hace clic en la opción de modificar, el sistema muestra la interfaz modificar clientes, el Especialista Comercial modifica los datos correspondientes y da clic en el botón modificar, el sistema modifica el cliente en la base de datos y muestra el mismo junto a los clientes guardadas en el sistema.

Curso alterno	El Especialista Comercial no se ha autenticado, el sistema muestra la ventana Autenticarse y cuando se especifiquen los datos mostrará la ventana correspondiente. Algunos de los campos están en blanco o con algunos formatos incorrectos. El sistema rechaza al cliente y muestra un mensaje con el correspondiente error cometido.

Descripción del Caso de Uso Modificar Cliente

Caso de Uso	Facturar Cliente
Curso básico	El Especialista Comercial da clic en la opción Facturas de la página principal, luego da clic en la opción facturar dentro de Facturas, el sistema muestra la interfaz facturar, el Especialista Comercial llena los datos correspondientes y da clic en el botón facturar, el sistema inserta la factura en la base de datos y muestra la misma junto a las facturas guardadas en el sistema.
Curso alterno	El Especialista Comercial no se ha autenticado, el sistema muestra la ventana Autenticarse y cuando se especifiquen los datos mostrará la ventana correspondiente. Algunos de los campos están en blanco o con algunos formatos incorrectos. El sistema rechaza la factura y muestra un mensaje con el correspondiente error cometido.

Descripción del Caso de Uso Facturar Cliente

Anexo III Diagramas de Robustez

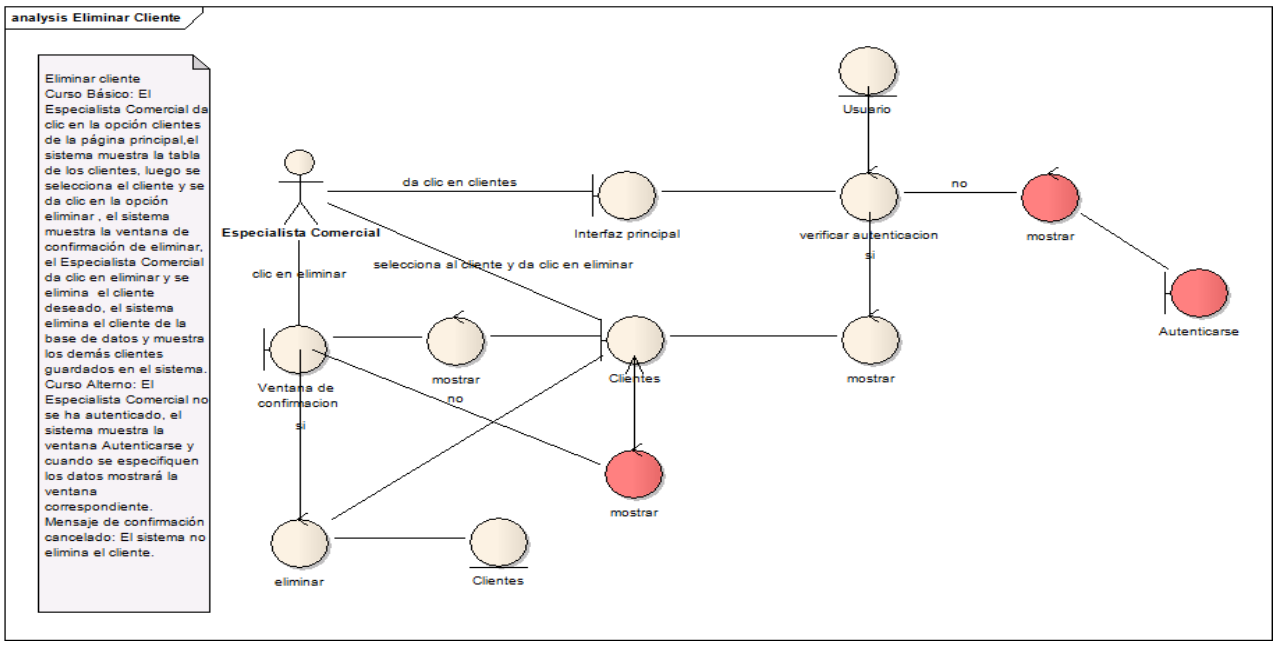

Diagrama de robustez Eliminar Cliente

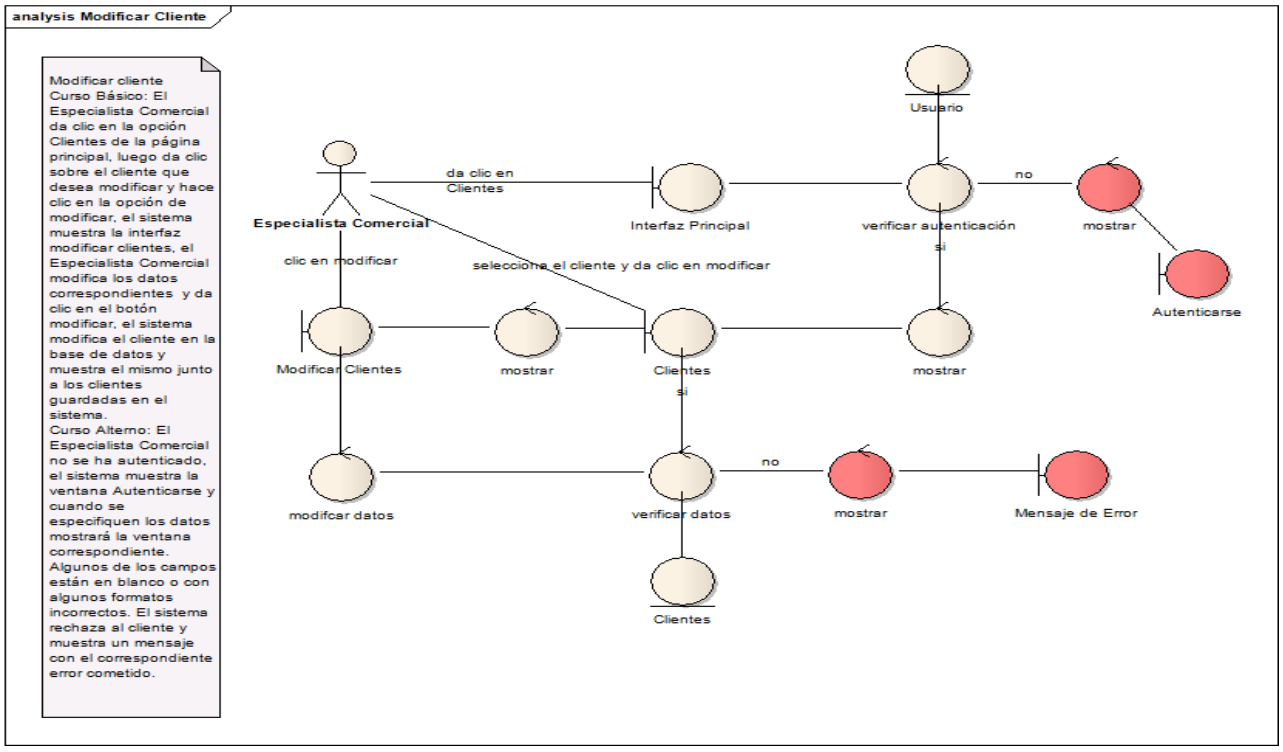

Diagrama de robustez Modificar Cliente

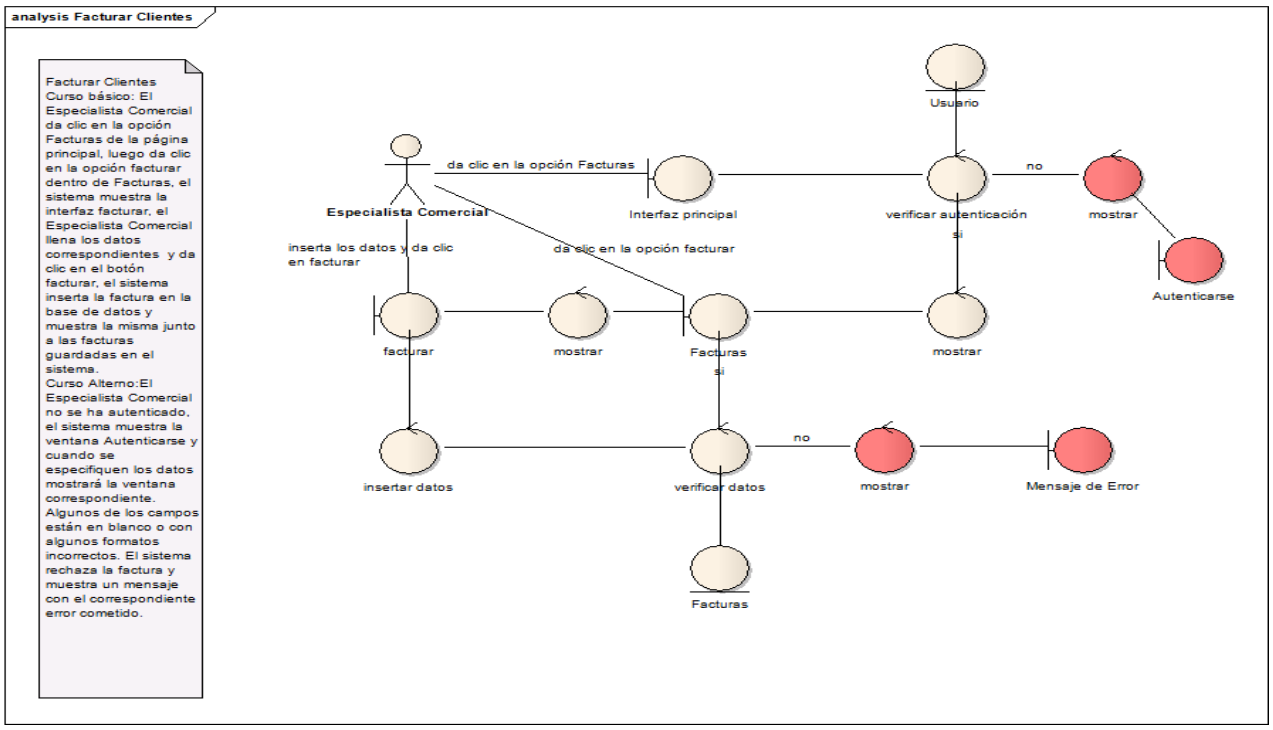

Diagrama de robustez Facturar Cliente

Anexo IV Diagramas de Secuencias

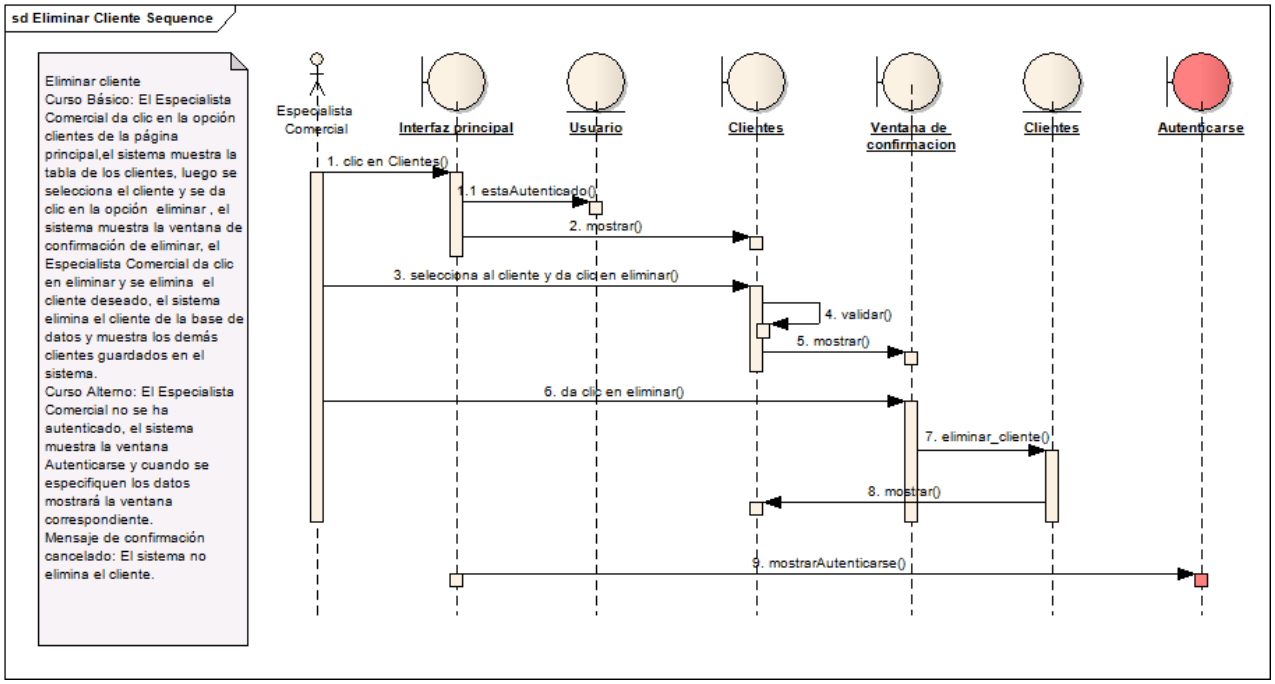

Diagrama de Secuencia Eliminar Cliente

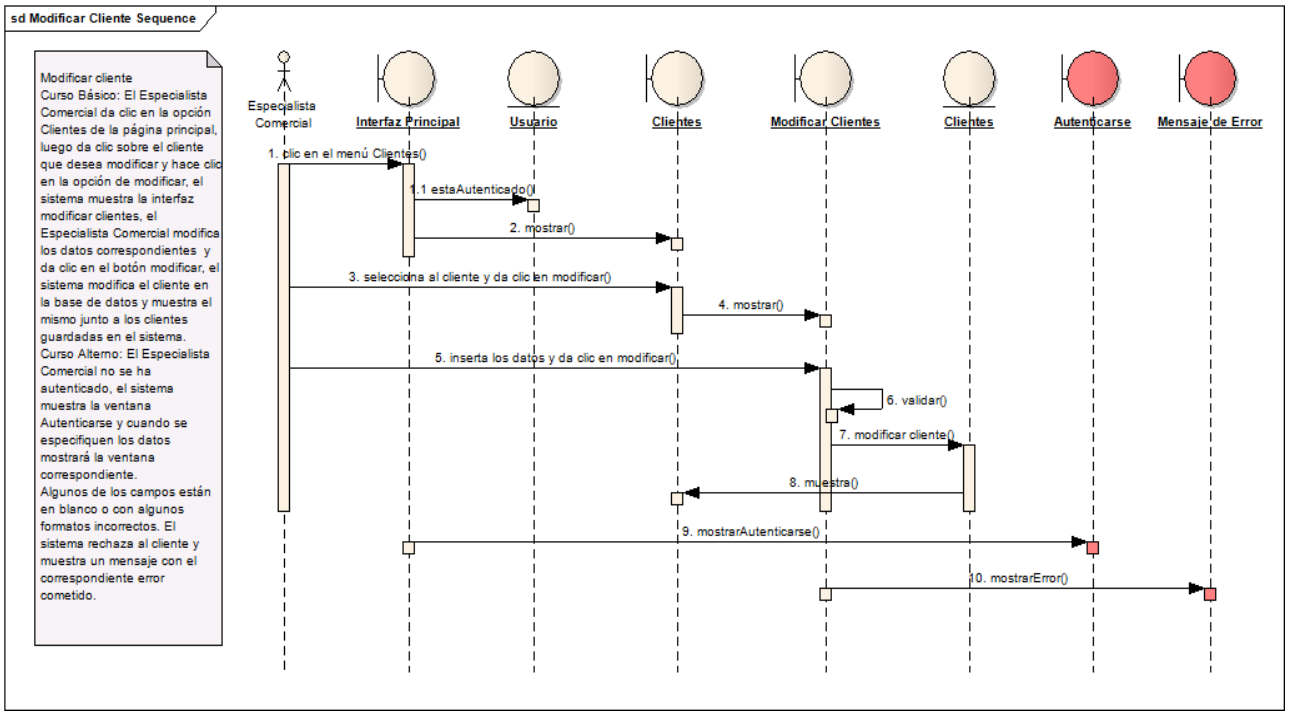

Diagrama de Secuencia Modificar Cliente

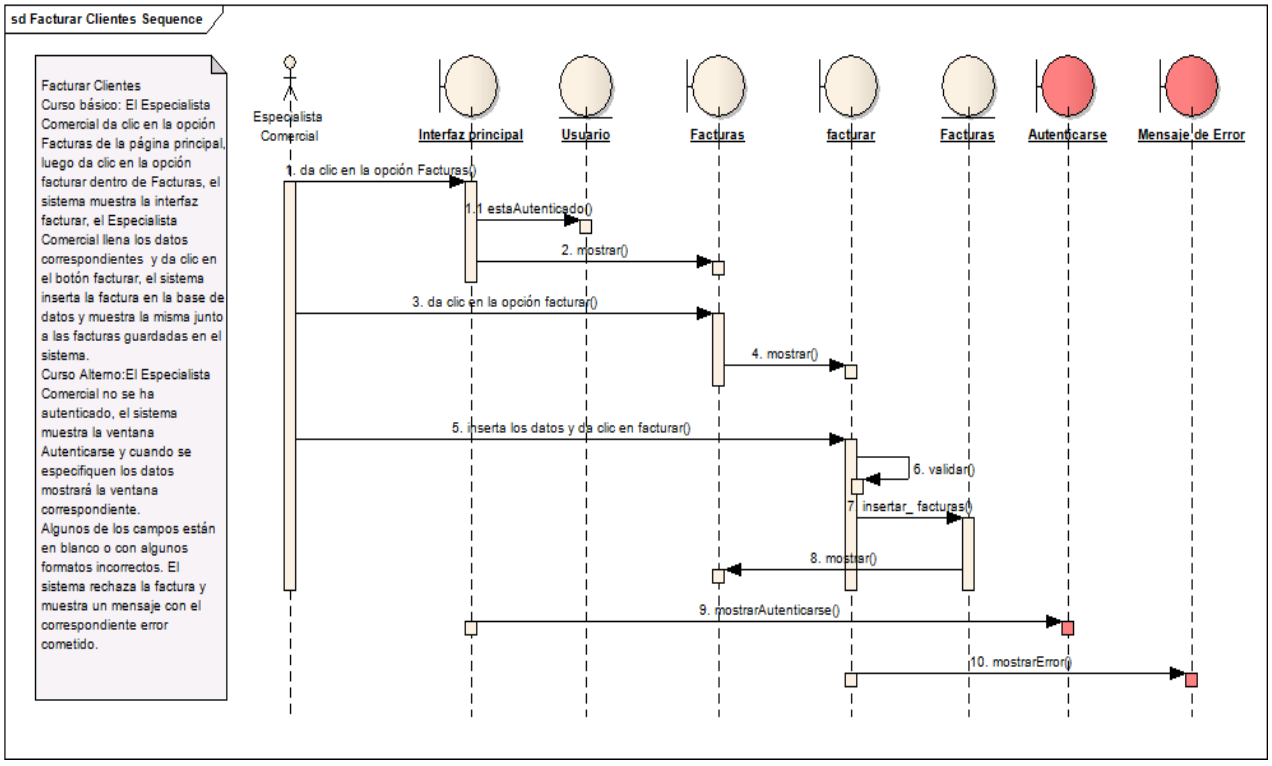

Diagrama de Secuencia Facturar Cliente

Anexo V Diagrama de Clases Persistentes

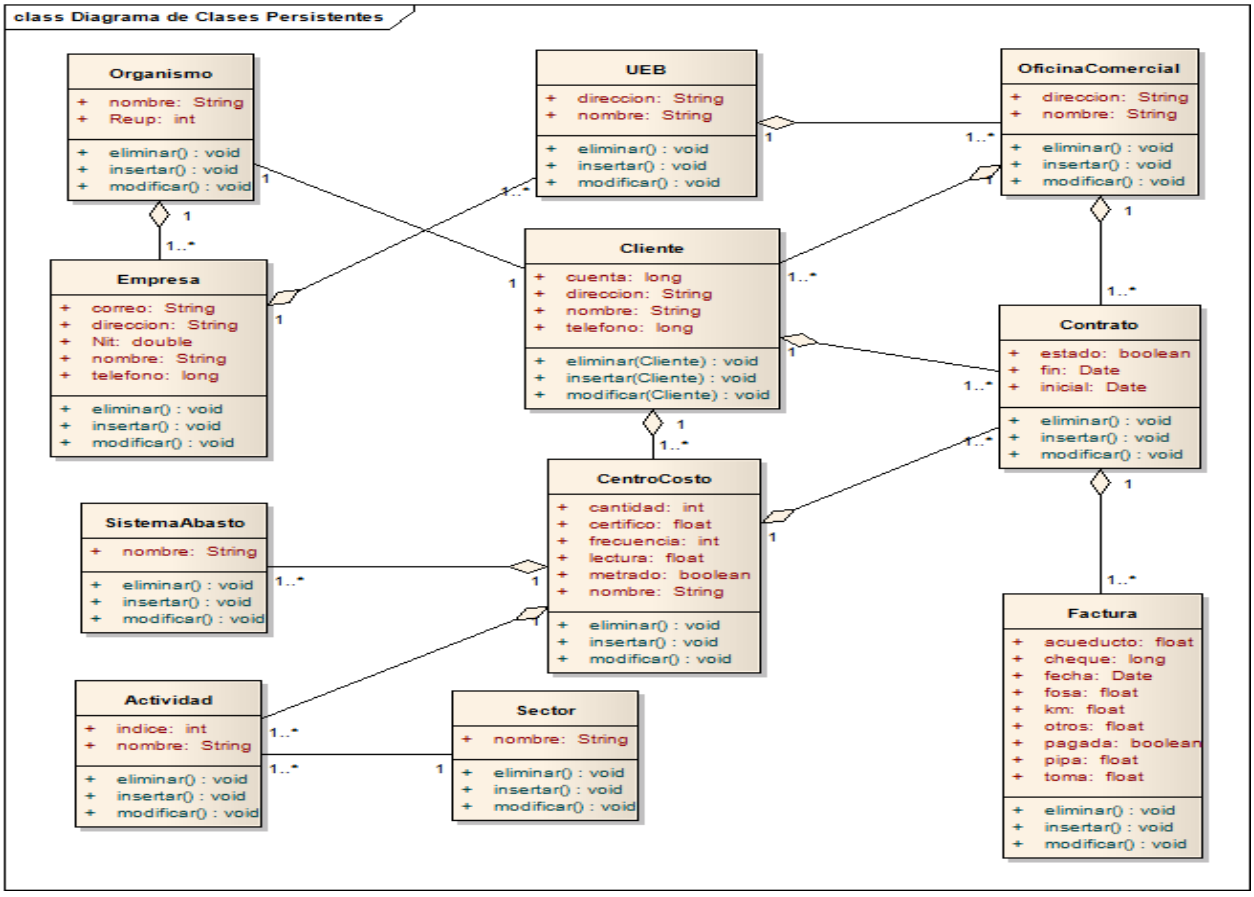

Diagrama de clases persistentes

Anexo VI Modelo de datos

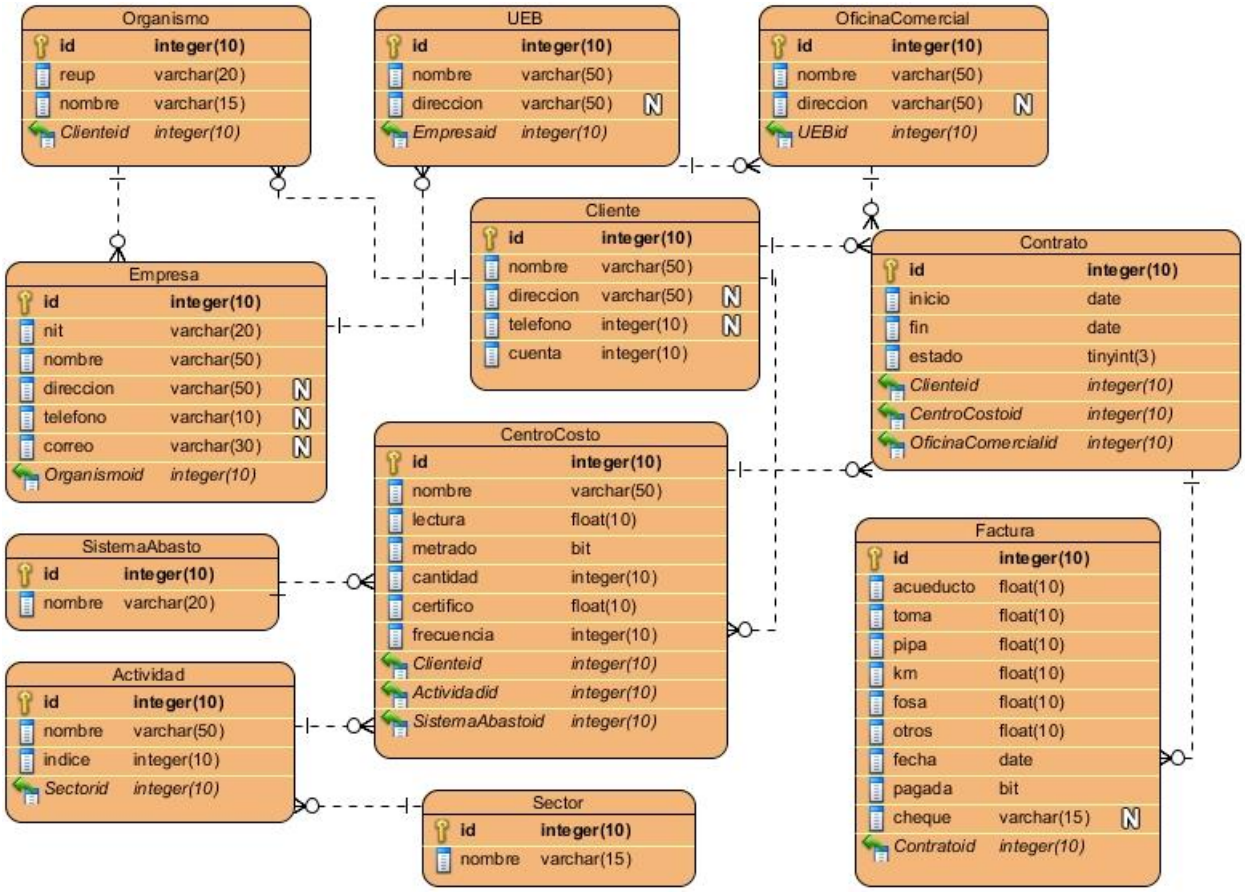

Diagrama del Modelo de Datos

Anexo VII

ENCUESTA PARA DETERMINAR EL COEFICIENTE DE COMPETENCIA DEL EXPERTO

Nombre y apellidos: _____

Usted ha sido seleccionado como posible experto para ser consultado respecto al grado de relevancia del **Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca**.

Necesitamos antes de realizarle la consulta correspondiente como parte del método empírico de investigación "consulta a expertos", determinar su coeficiente de competencia en este tema, a los efectos de reforzar la validez del resultado de la consulta que realizaremos. Por esta razón le rogamos que responda las siguientes preguntas de la forma más objetiva que le sea posible.

1.- Marque con una cruz (X), en la tabla siguiente, el valor que se corresponde con el grado de conocimientos que usted posee sobre el tema "**Tecnologías informáticas para el desarrollo de Sistemas de Gestión de Información**". Considere que la escala que le presentamos es ascendente, es decir, el conocimiento sobre el tema referido va creciendo desde 1 hasta 10.

1	2	3	4	5	6	7	8	9	10

2.- Realice una autovaloración del grado de influencia que cada una de las fuentes que le presentamos a continuación, ha tenido en su conocimiento y criterio sobre el tema "**Tecnologías informáticas para el desarrollo de Sistemas de Gestión de Información**".

Para ello marque con una cruz (X), según corresponda, en **A** (alto), **M** (medio) o **B** (bajo).

Fuentes de argumentación.	Grado de influencia de cada una de las fuentes.		
	A (alto)	M (medio)	B (bajo)
Análisis teórico realizado por usted.			
Su experiencia obtenida.			
Trabajo de autores nacionales.			
Trabajo de autores extranjeros.			
Su propio conocimiento del estado del problema en el extranjero.			
Su intuición.			

Muchas gracias.

Anexo VIII

ENCUESTA A EXPERTOS

Nombre y apellidos: _____

Institución a la que pertenece: _____Cargo actual: _____

Calificación profesional, grado científico o académico:

Profesor: _____ Licenciado: _____ Especialista: _____ Máster: _____ Doctor: __

Años de experiencia en el cargo: _____

Años de experiencia docente y/o en la investigación: _____

Estimado compañero(a), con motivo de evaluar el "**Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca**", desarrollado recientemente, se aplica esta encuesta. Por lo que la información que brinde será crucial para estos objetivos; rogamos que al responder estas preguntas lo haga de la manera más explícita posible. De antemano gracias.

Se requiere su opinión con relación a:

- ✓ Grado de relevancia de los aspectos del Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.
- ✓ ¿Qué aspectos cree Ud. que deben ser incluidos o eliminados?
- ✓ Sugerencias de cambios de denominación de los aspectos propuestos, cuyo grado de relevancia, sometemos a su consideración.

Indicaciones:

A continuación le presentamos una tabla que contiene los aspectos del Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca. A la derecha aparece la escala:

MA: Muy adecuado. **BA:** Bastante adecuado. **A:** Adecuado.

PA: Poco adecuado **NA:** No adecuado.

- ✓ Marque con una cruz (X) en la celda que se corresponda con el grado de relevancia que usted otorga a cada aspecto del Sistema de Gestión de Información.

Le agradecemos anticipadamente el esfuerzo que sabemos hará para responder, con la mayor fidelidad posible a su manera de pensar la presente encuesta.

Sobre el Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca.					
	MA	**BA**	**A**	**PA**	**NA**
Criterios					
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?					
¿Cómo evalúa el tiempo de respuesta del sistema?					
¿Cómo aprecia el diseño de las interfaces del sistema?					
¿Cree correcto el uso de los colores en las interfaces?					
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?					
¿Cuánto contribuirá el sistema a la toma de decisiones?					
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?					

¿Cómo evalúa la estructura organizativa del sistema?					
¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?					

MA: Muy adecuado **BA:** Bastante adecuado **A:** adecuado.

PA: Poco adecuado **NA:** No adecuado

- ✓ Escriba a continuación qué aspectos considera que deben ser incluidos o eliminados en esta propuesta:

Aspectos que se proponen ser incluidos	**Aspectos que se proponen ser eliminados**

- ✓ Otra sugerencia que usted desee hacer sobre el Sistema informático para la gestión comercial en el sector estatal en la Empresa de Acueducto y Alcantarillado Guardalavaca que estamos sometiendo a su consideración.

Anexo IX

Procesamiento de la encuesta de opinión de los expertos aplicando el método Delphi.

Tabla de frecuencia absoluta						
Criterios	**MA**	**BA**	**A**	**PA**	**NA**	**TOTAL**
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?	11	3	1	0	0	15
¿Cómo evalúa el tiempo de respuesta del sistema?	14	1	0	0	0	15
¿Cómo aprecia el diseño de las interfaces del sistema?	13	1	1	0	0	15
¿Cree correcto el uso de los colores en las interfaces?	13	1	1	0	0	15
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?	14	1	0	0	0	15
¿Cuánto contribuirá el sistema a la toma de decisiones?	13	1	1	0	0	15
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?	13	2	0	0	0	15
¿Cómo evalúa la estructura organizativa del sistema?	13	2	0	0	0	15

¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?	11	2	2	0	0	15

Tabla de frecuencia absoluta acumulada					
	MA	**BA**	**A**	**PA**	**NA**
Criterios					
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?	11	13	15	15	15
¿Cómo evalúa el tiempo de respuesta del sistema?	14	15	15	15	15
¿Cómo aprecia el diseño de las interfaces del sistema?	13	14	15	15	15
¿Cree correcto el uso de los colores en las interfaces?	13	14	15	15	15
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?	14	15	15	15	15
¿Cuánto contribuirá el sistema a la toma de decisiones?	13	14	15	15	15
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?	13	15	15	15	15
¿Cómo evalúa la estructura organizativa del sistema?	13	15	15	15	15
¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?	11	13	15	15	15
Tabla del inverso de la frecuencia absoluta acumulada					

	MA	**BA**	**A**	**PA**
Criterios				
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?	0,7143	0,9286	1	1
¿Cómo evalúa el tiempo de respuesta del sistema?	0,9286	1	1	1
¿Cómo aprecia el diseño de las interfaces del sistema?	0,8571	0,9286	1	1
¿Cree correcto el uso de los colores en las interfaces?	0,8571	0,9286	1	1
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?	0.9286	1	1	1
¿Cuánto contribuirá el sistema a la toma de decisiones?	0,8571	0.9286	1	1
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?	0,8571	1	1	1
¿Cómo evalúa la estructura organizativa del sistema?	0,8571	1	1	1
¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?	0,7143	0,8571	1	1

Tabla de determinación de los puntos de cortes							
	MA	**BA**	**A**	**PA**	**Suma**	**Promedio**	**N-Prom.**
Criterios							
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?	0,57	1,47	3,49	3,49	9,02	2,26	0,33
¿Cómo evalúa el tiempo de respuesta del sistema?	1,47	3,49	3,49	3,49	11,94	2,99	-0,4
¿Cómo aprecia el diseño de las interfaces del sistema?	1,07	1,47	3,49	3,49	9,52	2,38	0,21
¿Cree correcto el uso de los colores en las interfaces?	1,07	1,47	3,49	3,49	9,52	2,38	0,21
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?	1,47	3,49	3,49	3,49	11,94	2,99	-0,4
¿Cuánto contribuirá el sistema a la toma de decisiones?							

	1,07	1,47	3,49	3,49	9,52	2,38	0,21
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?	1,07	3,49	3,49	3,49	11,54	2,89	-0,3
¿Cómo evalúa la estructura organizativa del sistema?	1,07	3,49	3,49	3,49	11,54	2,89	-0,3
¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?	0,57	1,07	3,49	3,49	8,62	2,16	0,43
Suma	9,43	20,91	31,41	31,41	93,16		
Punto de Corte	1,05	2,32	3,49	3,49	10,35	2,59	=N(Pro .Ge)

Conclusiones Generales					
	MA	**BA**	**A**	**PA**	**NA**
Criterios					
¿Cómo evalúa los requerimientos seleccionados para la confección del sistema?	Si	-	-	-	-
¿Cómo evalúa el tiempo de respuesta del sistema?	Si	-	-	-	-
¿Cómo aprecia el diseño de las interfaces del sistema?	Si	-	-	-	-
¿Cree correcto el uso de los colores en las interfaces?	Si	-	-	-	-
¿Cómo evalúa la combinación de las tecnologías y herramientas de desarrollo?	Si	-	-	-	-
¿Cuánto contribuirá el sistema a la toma de decisiones?	Si	-	-	-	-
¿Cómo evalúa las facilidades que brinda para la gestión de los contratos y las facturas?	Si	-	-	-	-
¿Cómo evalúa la estructura organizativa del sistema?	Si	-	-	-	-
¿Cómo evalúa el sistema en cuanto a facilidad y comodidad?	Si	-	-	-	-