UNIVERSIDAD MAYOR DE SAN ÁNDRES
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

Sistema De Gestión Jurídico
(Caso: Estudio Jurídico Belzu & Ortuño)

PARA OPTAR AL TITULO DE LICENCIATURA EN INFORMÁTICA
MENCIÓN INGENIERÍA DE SISTEMAS INFORMÁTICOS

Postulante:	**Jacqueline Wendy Carvajal Romero**
Docente Tutor:	**M.Sc. Lic. Rosa Flores Morales**
Docente Revisor:	**Lic. Rubén Alcón López**

La Paz – Bolivia
2011

DEDICATORIA

A nuestro creador Dios que con toda su bondad me ayudó concluir uno de mis objetivos, por el regalo más grande que me dio en la vida mis hijos Felipe y Aldo, por la alegría y amor que generan en mi vida, a mi mamá Felicia Romero que en paz descanse.

jacarowen@hotmail.com

AGRADECIMIENTOS

Agradezco a toda mi familia, a mi mamá Felicia Romero Llanos que en paz descanse, por haberme otorgado la vida, por estar siempre conmigo apoyando, impulsando para la conclusión de mis estudios, para seguir adelante y constante lucha en la vida, gracias mamita querida siempre vivirás en mi corazón, a mi papá Mario Carvajal, por todo su apoyo y enseñanza, a mis hermanos que de alguna manera siempre me apoyaron, a mi abuelita Ana por su constante apoyo y alegría, en especial agradezco la presencia de mis hijos su alegría y amor, a mi esposo por sus constantes reflexiones y hacerme ver en la vida que no todo es como uno piensa.

Agradezco a la Lic. Rosa Flores Morales, por su completa dedicación y voluntad de ayudarme para la culminación del proyecto, por haberme otorgado su apoyo en todo momento, por sus aportes en el planteamiento del problema, objetivos, metodología y sugerencias de material adecuado, por el tiempo dedicado en la corrección del presente trabajo, por su completa colaboración para comprender muchos aspectos que carecían de claridad y que con su apoyo fueron comprendidos de manera satisfactoria para la culminación de este proyecto y aprender muchas cosas en la etapa de mi formación profesional y personal.

Al Lic. Rubén Alcón López, por los consejos y sugerencias, por el tiempo dedicado a la revisión y corrección para el desarrollo y culminación del presente trabajo.

Al Dr. Francisco Belzu Alarcón por darme la oportunidad de desarrollar e implementar el presente proyecto en instalaciones del Estudio Jurídico Belzu & Ortuño.

jacarowen@hotmail.com

A todos mis amigos por estar en las buenas y en las malas conmigo, gracias por su apoyo, a la familia BAP por sus enseñazas y experiencia que pude obtener dentro de la organización, a la Lic. Ruth Limachi por su constante apoyo y sugerencias.

jacarowen@hotmail.com

RESUMEN

El presente trabajo tiene por objetivo el desarrollo e implementación de una aplicación web para un estudio jurídico que se dedica a la representación de procesos judiciales. Los principales problemas que se encontraron son: la desinformación e inseguridad de los clientes, la presencia del abogado para informar al cliente, pérdida de documentación y la falta de comunicación entre el estudio jurídico y sus clientes.

La aplicación web fue desarrollada aplicando la metodología Objetory que está orientado a objetos, tecnología basada en el lenguaje unificado de modelación (UML) que permite acercar al patrón modelo vista controlador (MVC) a las necesidades para este tipo de proyectos, el diseño de la aplicación web fue aplicada con la metodología OOHDM que permite la elaboración de aplicaciones multimedia.

Las pruebas finales del producto nos muestran que la aplicación es altamente confiable con respecto a los datos que ofrece.

Palabras Clave: Aplicación web, estudio jurídico online, pruebas de calidad de software.

jacarowen@hotmail.com

INDICE DE FIGURAS

Descripción **Pág.**

INDICE DE TABLAS

I N D I C E
SISTEMA DE GESTION JURIDICO

jacarowen@hotmail.com

CAPITULO III
Análisis y Diseño del Sistema

Capitulo IV
Conclusiones y Recomendaciones

CAPITULO I

PRESENTACIÓN

1.1 INTRODUCCIÓN.

En la actualidad el uso de la tecnología dentro de las diferentes organizaciones, instituciones públicas y privadas es un hecho muy relevante, por el gran avance tecnológico dentro del campo de la información. Este hecho se plasma precisamente en la implementación de sistemas informáticos dentro de las organizaciones, dando de esta forma una gran importancia al control, seguimiento y análisis de la información que para muchas organizaciones constituyan un producto más, que incluso puede llegar a determinar el fracaso o el éxito de una institución.

Por estas características, el hecho de implementar un sistema informático que sea adecuado a las necesidades de la organización o institución pública o privada, ya es prácticamente una obligación de quienes tienen a su cargo facilitar el trabajo de su personal, haciendo que estos disfruten de las ventajas en cuanto a información, comunicación y más aun en el tiempo y eficiencia que le brinda un sistema informático.

Por estas razones se propone el desarrollo de un Sistema para facilitar la gestión de un despacho jurídico, que sea capaz de dar a conocer de un modo rápido sencillo el estado actual o pasado de cualquier caso o cliente. Además tener una comunicación constante con sus clientes vía Internet, ofreciendo servicios más eficientes para los usuarios.

1.2 ANTECEDENTES.

1.2.1 ANTECEDENTES DEL PROYECTO.

En cuanto a sistemas relacionados se revisó la siguiente documentación:

- **Sistema de Rigoberto Paredes,** Es un sistema en el que ofrece información sobre los casos que patrocina como ser casos en contra de la integridad física del ser humano, entre ellos: Negligencia Médica, Responsabilidad de Productos Farmacéuticos y Valoración de Daños Personales, etc.

  De cada uno de estos casos sus clientes pueden acceder a la información de su caso. También ofrece información sobre sus oficinas.

- **Sistema Lino - Sandy,** Es un sistema en el que ofrece información sobre las áreas que asesora: Derecho Civil, Mercantil, Administrativo, Tributario, Aduanero, Laboral, Familiar, Constitucional, Medioambiental etc. Lino – Sandy asesora a las empresas en el correcto cumplimiento de las formalidades requeridas por el Gobierno para la importación, tránsito y exportación de mercaderías y en la búsqueda de mejores prácticas en operaciones aduaneras cotidianas, contribuyendo a reducir el impacto negativo en los negocios producidos por la detención de las mercaderías por eventuales incumplimientos a las leyes aduaneras.
- **SILEG**, Es un sistema informático que consiste en una base de datos en la que están registradas los textos íntegros y totalmente concordados de las disposiciones normativas fundamentales y activas que se han aprobado desde la fundación de la República, así como tratados y convenios internacionales, legislación comparada, doctrina de calidad y textos ordenados de códigos y normas básicas. Con el "SILEG en Línea", podrá realizar consultas mediante búsquedas entre más de 40000 disposiciones (textos íntegros) desde la creación de la República a la fecha; Códigos, Lees, Decretos Supremos, Reglamentos, etc. a una velocidad extraordinaria. Puede acceder al SILEG en Línea desde cualquier lugar simplemente con una conexión a Internet.
- **Sistema de Administración de un Estudio Jurídico,** Este sistema desarrollado en la carrera de Informática de la U.M.S.A. propone la gestión de casos jurídicos dentro un estudio de Abogados, donde se realizan transacciones legales en el quehacer jurídico, ayudando al abogado en el registro de estrategias planes de trabajo, paralelamente a esto se realiza una contabilidad del movimiento económico de los casos. El análisis del sistema expresado no es del todo completo en cuanto a todo el área jurídica puesto que se ha descartado el análisis de la informática jurídica documental de sistemas expertos aplicados al área jurídica.

1.3 PLANTEAMIENTO DEL PROBLEMA.

1. 3.1 ANÁLISIS DEL PROBLEMA.

La solidez, eficiencia y eficacia de una empresa, en nuestro caso un despacho de abogados, se produce en parte cuando todos los involucrados se encuentran

satisfechos con la información adecuada sobre el caso que lleva en curso, así como también información sobre el abogado que se encuentra a cargo de dicho caso y obtener información sobre el costo que deberá invertir.

Al inicio del nuevo milenio, la legislación boliviana toma un nuevo mecanismo de proceso judicial que lleva el nombre de Juicio Oral, este nuevo mecanismo impulsará la aceleración de los tramites judiciales cuyo objetivo es el de disminuir la retardación de justicia. La falta de comunicación que existe hoy en día entre el abogado y su cliente lleva muchas consecuencias como: falta de decisiones oportunas, pérdida de dinero, casos inconclusos, etc. También se puede detectar que cuando el abogado este con trabajo excesivo (muchos casos a su cargo) y cuando un cliente va a solicitar información este no pueda darle una buena información o ninguna por diversas razones: falta de registro del caso, descuido de su secretaria, pérdida de bienes, casos perdidos, etc. Toda esta falta de información completa, adecuada y a un determinado tiempo podría llevarnos a varios tropiezos y consecuentemente a pérdida de tiempo y dinero.

La información requerida se la puede obtener consultando o pasando por el despacho o consultoría de abogados personalmente, pero todo esto requiere valioso tiempo y dinero. Hoy en día no podemos perder tanto tiempo ni dinero ya que podríamos invertirlo en otras actividades. Ya que el cliente talvez vive lejos o no cuente con tiempo para seguir su caso, no cuente con mucho recurso económico y el retraso del mismo podría obligarle a abandonar el caso.

En resumen los problemas que se detectaron son los siguientes:

- Se requiere la presencia del abogado responsable cuando un cliente requiere saber en que estado anda su caso.
- Se desconoce el proceso cronológico de un caso en específico.
- No existe un medio por el cual el Director sepa en tiempo real el estado de las cuentas de un cliente, así mismo del caso del cliente.

- Los documentos se mezclan o pierden, tanto de actividades o gastos, sin poder tener la información adecuada en un tiempo oportuno para alguna consulta o informe que se requiera.

1.3.2 PROBLEMA PRINCIPAL.

Por la evaluación efectuada existen un alto índice de procesos inconclusos, retrasados o archivados debido a que no existe tiempo suficiente para el seguimiento de un caso en especial, o por el simple hecho de que existe desinterés por parte del cliente, esta situación deriva a la poca disponibilidad de tiempo para indagar o estudiar extensos artículos de ley con el afán de tomar un conjunto de ellos y amparar el caso bajo los mismos.

Existe desinformación en los clientes acerca de sus casos, pérdida de tiempo y dinero, abogados con mucho trabajo y no pueden atender ni informar como quisieran a sus clientes, todo esto provoca inseguridad en sus clientes que no saben lo que deben hacer ni que tácticas se van llevar a cabo o el siguiente paso que deben seguir.

Habiendo analizado los distintos problemas del estudio jurídico Belzu & Ortuño en el seguimiento de casos que atienden, se propone:

Incorporar un Sistema de Gestión Jurídico, en el que se enfatiza el Seguimiento de Casos y se optimicen los procesos de control de las actividades que realiza el Estudio Jurídico Belzu & Ortuño.

1.4 OBJETIVOS.

A partir de lo desarrollado anteriormente, se formulan los siguientes objetivos correspondientes al presente proyecto de grado:

1.4.1 OBJETIVO GENERAL.

Desarrollar un Sistema de Información de un Despacho Jurídico validado con normas de calidad ISO 9126.

1.4.2 OBJETIVOS ESPECÍFICOS.

- Desarrollar procedimientos para automatizar procesos de búsqueda;
- Desarrollar procedimientos de altas de clientes y de sus casos, del/los abogado(s) que se haga cargo de dicho caso y cliente, así como también el costo.
- Desarrollar procedimientos de altas, bajas, modificaciones de clientes y casos.
- Desarrollar procedimientos de registro de los casos, fechas, abogado que atendió, etc.
- Facilitar el acceso a la información sobre cualquier caso que requiera el cliente previa verificación de identidad;
- Brindar información actualizada sobre las áreas que se atienden en el despacho de abogados;
- Permitir a los usuarios obtener toda la información de una manera sencilla, ágil y completa, previa verificación de identidad;
- Facilitar consultas a todas aquellas personas que tienen responsabilidad en la toma de decisiones;
- Establecer privilegios y niveles de acceso definiendo políticas y procedimientos de seguridad.
- Facilitar información de las direcciones de oficinas y horario en que se les puede atender.
- Facilitar información de los abogados que se encuentran asociados en nuestro despacho de abogados.

1.5 ALCANCES y APORTES.

El alcance del proyecto comprende lo siguiente:

- Desarrollar e implementar un sistema informático en el control y seguimiento de casos de un despacho de abogados.
- Difundir información corporativa, mantener informados a los clientes mediante servicios Web.

Los aportes del sistema son:

- La implantación del presente proyecto es de gran aporte ya que este será de gran ayuda para la toma de decisiones de los cliente y/o usuarios del Sistema de Gestión Jurídico;
- Este sistema será elaborado bajo factores que permitan medir la calidad del mismo como ser: exactitud, fiabilidad, usabilidad y flexibilidad, permitiendo de esta forma la satisfacción del usuario.

1.6 JUSTIFICACIONES.

Es necesario automatizar el seguimiento de casos de un despacho de abogados, así mismo poder gestionar el despacho de abogados, obteniendo de manera rápida y sencilla la información del estado actual o pasado del cliente y su caso. De esta manera ahorrar tiempo y dinero, facilitando la comunicación entre el cliente y su abogado obteniendo información actualizada, oportuna.

1.6.1 JUSTIFICACIÓN TÉCNICA.

Se cuenta con la tecnología para poder desarrollar el sistema de información y el software necesario para su desarrollo. En la actualidad los medios que ofrecen información acerca de seguimiento de casos son manuales o no ofrecen información completa, por lo que el uso de herramientas informáticas se hace necesario para posibilitar un acceso ágil y eficiente a la información requerida.

1.6.3 JUSTIFICACIÓN ECONÓMICA.

El Sistema ayudará de gran manera, ya que la información se la tendrá de manera ágil, actualizada y de manera oportuna esto influye de gran forma en la realización de trámites y decisiones. Por tanto el beneficio es intangible ya que se refleja indirectamente a partir de hechos intangibles, ya que se expresan a través de mejorar el servicio a sus clientes.

1.7METODOLOGÍA Y HERRAMIENTAS.

La metodología a ser utilizada será Object Oriented Software Engineering (Metodología Objetory) [Jacobson, 1992] (OOSE/Objetory), que es una metodología orientado a

objetos, ya que los sistemas orientados a objetos se desarrollan alrededor de entidades del dominio del problema, lo cual resulta desarrollos bastante estables. El análisis orientado a objetos, a diferencia del estructurado, considera comportamiento y datos de forma separada, combina ambos. La característica principal de la tecnología orientada a objetos es que ofrece una forma de pensar más que una forma de programar. Además, reducen la complejidad en el diseño de software, y permiten atacar los errores durante el diseño en lugar de durante la implementación, donde el costo de reparación es bastante mayor.

Para el análisis el diseño nos basaremos en el Lenguaje Unificado de Modelado (UML) que permite acercar al patrón Modelo Vista Controlador (MVC) a las necesidades para este tipo de sistemas informáticos.

También se utilizará las herramientas como:

- PostgreSQL, ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones.
- Java Server Faces (JSF), la tecnología JSF constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java y en el patrón MVC. Con el entorno NetBeans que es un entorno de desarrollo, de código abierto, una herramienta para programadores para escribir, compilar, corregir errores para ejecutar programas.
- OOHDM, es una metodología de desarrollo propuesta por Rossi y Schwabe (ROSSI 1996) para la elaboración de aplicaciones multimedia y tiene como objetivo simplificar y a la vez hacer más eficaz el diseño de aplicaciones hipermedia.

1.8 ARQUITECTURA DE SISTEMA.

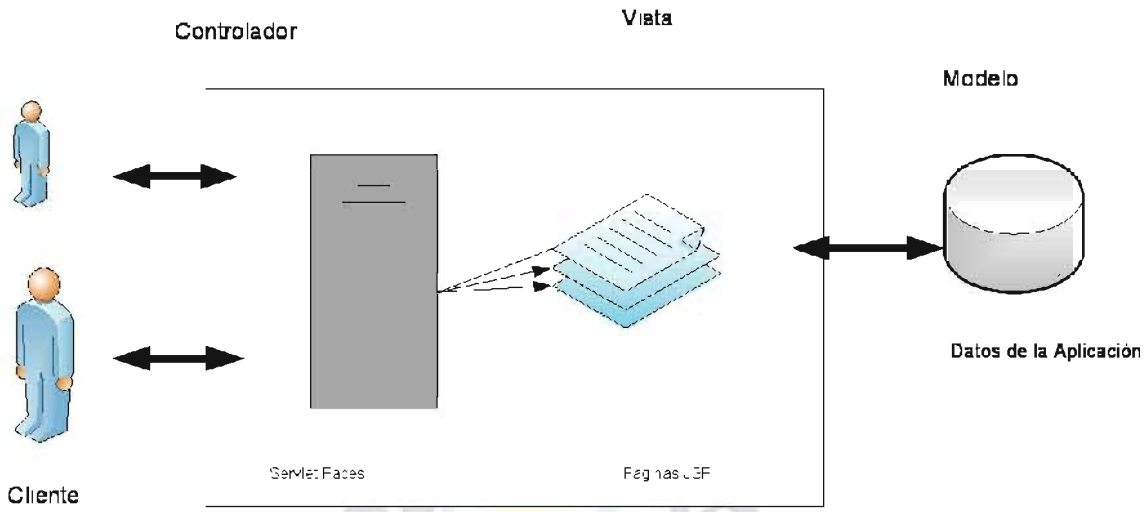


Figura 1.1: Arquitectura Sistema: Sistema de Gestión Jurídico
Fuente: [Elaboración Propia]

El uso del patrón MVC ver figura 1.1 nos permite separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) la lógica de presentación (cómo interaccionar con el usuarios).

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, perder el miedo al folio en blanco (existe un patrón de partida por el que empezar un proyecto), etc. al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar estandarizar el desarrollo de Software.

2.1 SISTEMA DE GESTION.

Un Sistema de Gestión es un conjunto de etapas unidas en un proceso continuo, que permite trabajar ordenadamente una idea hasta lograr mejoras y su continuidad.

Se establecen cuatro etapas en este proceso, que hacen de este sistema, un proceso circular virtuoso, pues en la medida que el ciclo se repita recurrente y recursivamente, se logrará en cada ciclo, obtener una mejora.

Las cuatro etapas del sistema de gestión son:

- Etapa de Ideación;
- Etapa de Planeación;
- Etapa de Implementación y
- Etapa de Control.

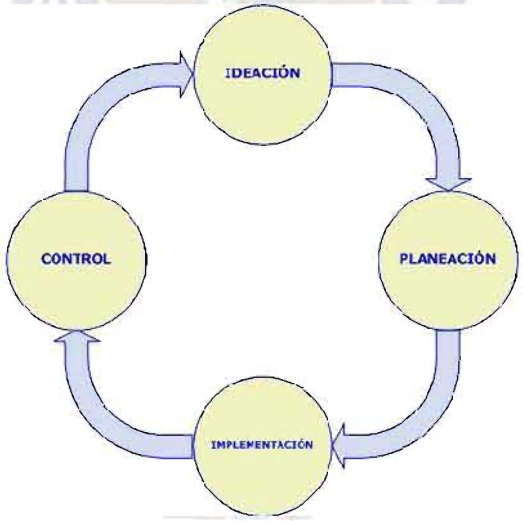


Figura 2.1: Sistema de Gestión: Etapas
Fuente: [Internet]

- **ETAPA DE IDEACIÓN.**

  El objetivo de esta etapa es trabajar en la idea que guiará los primeros pasos del proceso de creación que se logra con el sistema de gestión propuesto.

- **ETAPA DE PLANEACIÓN (PLANIFICACIÓN).**

Dentro del proceso, la planificación constituye una etapa fundamental y el punto de partida de la acción directiva, ya que supone el establecimiento de sub-objetivos y los cursos de acción para alcanzarlos.

En esta etapa, se definen las estrategias que se utilizarán, la estructura organizacional que se requiere, el personal que se asigna, el tipo de tecnología que se necesita, el tipo de recursos que se utilizan y la clase de controles que se aplican en todo el proceso.

- **ETAPA DE IMPLEMENTACIÓN (GESTIÓN).**

En su significado más general, se entiende por gestión, la acción y efecto de administrar. Pero, en un contexto empresarial, esto se refiere a la dirección que toman las decisiones y las acciones para alcanzar los objetivos trazados. Es importante destacar que las decisiones y acciones que se toman para llevar adelante un propósito, se sustentan en los mecanismos o instrumentos administrativos (estrategias, tácticas, procedimientos, presupuestos, etc.), que están sistémicamente relacionados y que se obtienen del proceso de planificación. (Véase la figura: 2.2).

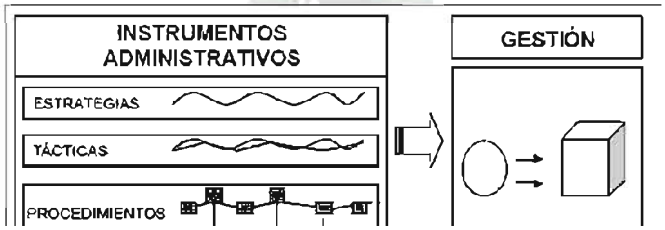


Figura 2.2: Esquema de Gestión
Fuente: [Internet]

- **ETAPA DE CONTROL.**

Para este concepto se han desarrollado varias definiciones [CABRERA, 2005] a lo largo de su evolución, sin embargo, todas se centran en la siguiente idea general:

El control es una función administrativa, esencialmente reguladora, que permite verificar (o también constatar, palpar, medir o evaluar), si el elemento seleccionado (es decir, la actividad, proceso, unidad, sistema, etc.), está cumpliendo sus objetivos o alcanzando los resultados que se esperan.

Es importante destacar que la finalidad del control es la detección de errores, fallas o diferencias, en relación a un planteamiento inicial, para su corrección y/o prevención. Por tanto, el control debe estar relacionado con los objetivos inicialmente definidos, debe permitir la medición y cuantificación de los resultados, la detección de desviaciones y el establecimiento de medidas correctivas y preventivas.

2.2 MODELO VISTA CONTROLADOR.

Este patrón fue descrito por primera vez por Trygve Reenskaug en 1979, y la implementación original fue realizada en Smalltalk en los laboratorios Xerox.

MVC se basa en la separación de la aplicación en tres capas principales: Modelo, Vista y Controlador. Se usa (él o alguna de sus variantes) en la gran mayoría de las interfaces de usuario.

El Modelo-Vista-Controlador (Model-View-Controller) MVC es un patrón de desarrollo que separa la parte lógica de una aplicación de su presentación. Básicamente sirve para separar el lenguaje de programación del HTML lo máximo posible y para poder reutilizar componentes fácilmente. El patrón MVC nos permite separar la lógica de control (qué cosas hay que hacer pero no cómo), la lógica de negocio (cómo se hacen las cosas) y la lógica de presentación (como interaccionar con el usuario, ver Figura 2.3.

- El **Modelo** representa las estructuras de datos. Típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos. Será la encargada de hacer el trabajo real de la aplicación. "Es el que

gestiona los datos y controla todas sus transformaciones". El modelo no tiene conocimiento específico de los diferentes controladores y/o vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el modelo y sus vistas, y notifica a las vistas cuándo deben reflejar un cambio en el modelo;

- La **Vista** es la información presentada al usuario. Una vista puede ser una página web o una parte de una página. Será la encargada de gestionar la interfaz y todo lo relacionado con la interacción entre la aplicación y el usuario. "Es el que gestiona como se muestran los datos". Interacciona con el modelo a través de una referencia al propio modelo;
- El **Controlador** actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página. Sirve para desacoplar ambas, de forma que, si fuera necesario, una vista pueda ser actualizada por varios modelos, o un modelo pueda proveer datos de varias vistas. "Es el que determina que modificaciones hay que hacer cuando se interacciona con el elemento".

El controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el modelo. Entra en acción cuando se realiza alguna operación, ya sea un cambio en la información del modelo o una interacción sobre la Vista. Se comunica con el modelo y la vista a través de una referencia al propio modelo.

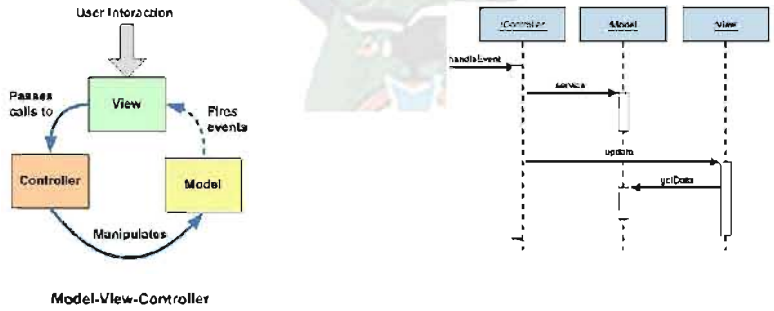


Figura 2.3: División en capas del M.V.C.
Fuente: [Bermúdez & Martínez & González & et al]

2.2.1 VENTAJAS DE UTILIZAR MVC.

Utilizando este tipo de patrón es posible conseguir más calidad, un mantenimiento más fácil, perder el miedo al folio en blanco (existe un patrón de partida por el que empezar un proyecto), etc. Al margen de todo esto, una de las cosas más importantes que permite el uso de este patrón consiste en normalizar y estandarizar el desarrollo de Software.

Obviamente una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc. sin alterar la lógica de negocio. La separación de capas como presentación, lógica de negocio, acceso a datos es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y más fácilmente mantenibles, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas;
- Agregar nuevas formas de recolectar las ordenes del usuario (interpretar sus modelos mentales);
- Modificar los objetos de negocios bien sea para mejorar el performance o para migrar a otra tecnología;
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones solo se deben hacer en un solo lugar y no en varios como sucedería si tuviésemos una mezcla de presentación e implementación de la lógica del negocio;
- Es posible construir nuevas vistas sin modificar el modelo subyacente;
- Proporciona un mecanismos de configuración a componentes complejos muchos más tratable que el puramente basado en eventos (el modelo puede verse como una representación estructurada del estado de interacción);
- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado;

- Hay una API muy bien definida; cualquiera que use la API, podrá reemplazar el modelo, la vista o el controlador, sin demasiada dificultad;
- La conexión entre el modelo y sus vistas (ya que puede haber varias vistas para un mismo modelo) es dinámica: se produce en tiempo de ejecución, no en tiempo de compilación.

Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se paralice. Adicionalmente el patrón MVC propone a la especialización de cada rol del equipo, por tanto en cada liberación de una nueva versión se verán los resultados

2.3 HERRAMIENTAS.

Las herramientas que se empleará son: POSTGRES, PHP y HTML, que se describen a continuación:

- **PostgreSQL,** los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS,s) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos.

  El modelo relacional sustituyó modelos previos en parte por su "simplicidad espartana". Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema: clases, herencia, tipos y funciones.

  Otras características aportan potencia y flexibilidad adicional: Restricciones (Constraints), Disparadores (triggers), Reglas (rules) e integridad transaccional.

Estas características colocan a Postgres en la categoría de las Bases de Datos identificadas como objeto-relacionales. Nótese que éstas son diferentes de las referidas como orientadas a objetos, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales. Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos. De hecho, algunas Bases de Datos comerciales han incorporado recientemente características en las que Postgres fue pionera;

- **JSF** Java Server Faces, la tecnología JSF constituye un marco de trabajo (framework) de interfaces de usuario del lado de servidor para aplicaciones Web basadas en tecnología Java y en el patrón MVC. Los principales componentes de la tecnología JSF son:

  - Una API y una implementación de referencia para:
  - Representar componentes de interfaz de usuario (UI –User Interface) y manejar su estado;
  - Manejar eventos, validar en el lado del servidory convertir datos;
  - Definir la navegación entre páginas;
  - Soportar internacionalización y accesibilidad, y;
  - Proporcionar extensibilidad para todas estas características;
  - Una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP.

Como se puede apreciar en la figura 2.4, la interfaz de usuario que se crea con la tecnología JSF (representada por miUI en el grafico) se ejecuta en el servidor y se renderiza en el cliente.

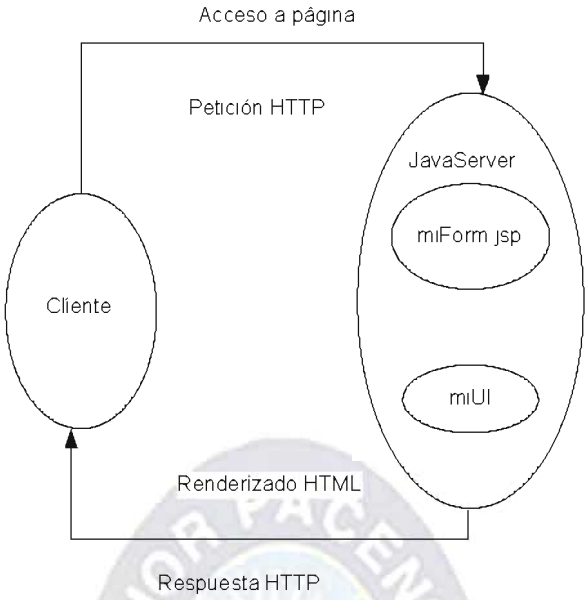


Figura 2.4: Diagrama de una aplicación JSF
Fuente: [Tutorial JavaServer Faces]

JSF, es muy flexible, ofrece una clara separación entre el comportamiento y la presentación.

- **OOHDM,** OOHDM es una metodología de desarrollo propuesta por Rossi y Schwabe (ROSSI 1996) para la elaboración de aplicaciones multimedia y tiene como objetivo simplificar y a la vez hacer más eficaz el diseño de aplicaciones hipermedia. OOHDM está basada en HDM, en el sentido de que toma muchas de las definiciones, sobre todo en los aspectos de navegación, planteadas en el modelo de HDM. Sin embargo, OOHDM supera con creces a su antecesor, ya que no es simplemente un lenguaje de modelado, sino que define unas pautas de trabajo, centrado principalmente en el diseño, para desarrollar aplicaciones multimedia de forma metodológica. Sus características son:
  - OOHDM está basada en el paradigma de la orientación a objetos;
  - propone un proceso predeterminado para el que indica las actividades a realizar y los productos que se deben obtener en cada fase del desarrollo;
  - Fundamentalmente OOHDM toma como partida el modelo de clases que se obtiene en el análisis del Proceso Unificado de UML. A este modelo lo denomina modelo conceptual;
  - Partiendo de este modelo conceptual, OOHDM propone ir añadiendo características que permitan incorporar a esta representación del sistema todos

los aspectos propios de las aplicaciones multimedia. En una segunda etapa de diseño, se parte de ese modelo conceptual y se añade a éste todos los aspectos de navegación, obteniéndose un nuevo modelo de clases denominado modelo navegacional. Por último, este modelo sirve como base para definir lo que en el argot de OOHDM se denomina modelo de interfaz abstracta. El modelo de interfaz abstracta representa la visión que del sistema tendrá cada usuario del mismo;

- OOHDM como técnica de diseño de aplicaciones hipermedia, propone un conjunto de tareas que según Schwabe, Rossi y Simone (s. f.) pueden resultar costosas a corto plazo, pero a mediano y largo plazo reducen notablemente los tiempos de desarrollo al tener como objetivo principal la reusabilidad de diseño, y así simplificar el coste de evoluciones y mantenimiento;

Esta metodología plantea el diseño de una aplicación de este tipo a través de cinco fases que se desarrollan de un modo iterativo. Estas fases son:

FASES DE OOHDM: EN OOHDM SE PROPONEN 5 FASES DE DESARROLLO:

- Determinación de Requerimientos;
- Diseño Conceptual;
- Diseño Navegacional;
- Diseño de Interfaz Abstracto;
- Implementación.

2.4 UML (UNIFIED MODELING LANGUAJE) COMO HERRAMIENTA PARA EL DESARROLLO DE SISTEMAS.

El Lenguaje Unificado de Modelado (UML) se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software" [BJR 97] El UML es un estándar incipiente de la industria para construir modelos orientados a objetos, nació en 1994 por iniciativa de Grady Booch y Jim Rumbaugh para combinar sus dos famosos métodos: el de Booch y el OMT (Object Management Technique,

Técnica de Modelado de Objetos). Mas tarde se les unió Ivar Jacobson, creador del Método OOSE (Ingeniería del software Orientado a Objetos) para definir un lenguaje y una notación estándar del Lenguaje de Construcción de Modelos.

UML describe un conjunto de notaciones y diagramas estándar para visualizar, especificar, construir y documentar sistemas orientados a objetos, describiendo la semántica esencial de lo que estos diagramas y símbolos significan.

- Diagrama de Estructura Estática
- Diagrama de Casos de Uso
- Diagrama de Secuencia

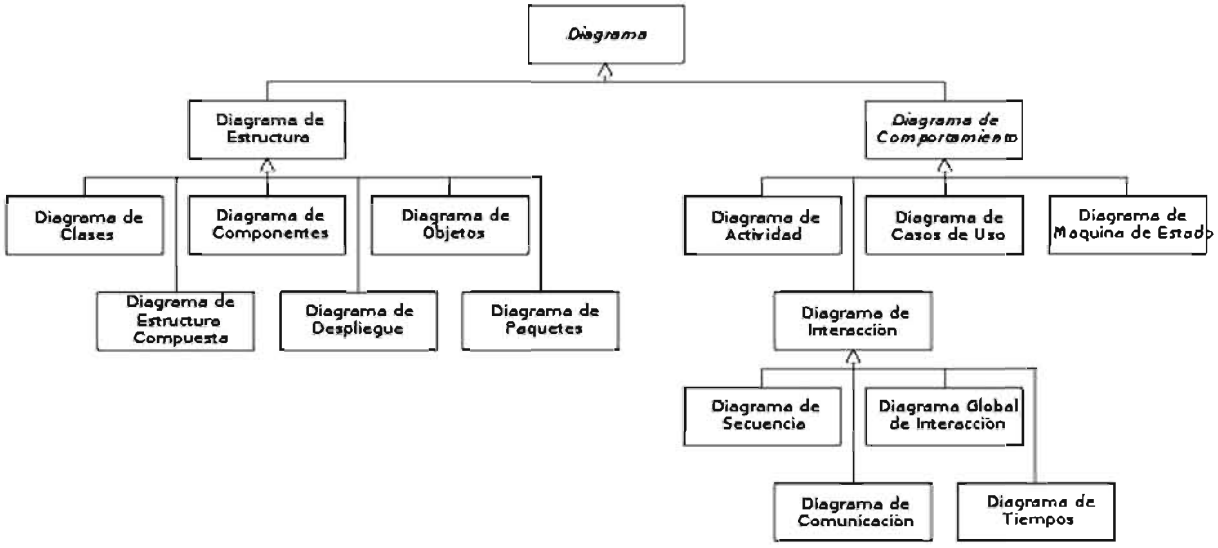
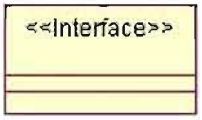


Figura 2.5: Jerarquía de los Diagramas UML 2.0 mostrado como un diagrama de clases
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

En UML, se propone para el desarrollo del modelo de análisis de las aplicaciones, tres tipos de clases fundamentales, con las cuales podemos expresar todas las funciones de cualquier software, con sus respectivas responsabilidades.

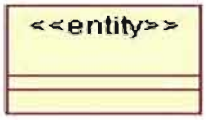
CLASE INTERFAZ <<INTERFACE>>:



- Recepcionar peticiones al sistema.
- Mostrar respuestas del sistema.

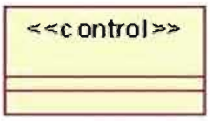
CLASE ENTIDAD <<ENTITY>>:

- Gestionar datos (información) necesaria para el sistema.
- Almacenar datos (información) persistentes del sistema.
- Provee la funcionalidad principal de la aplicación



CLASE CONTROLADOR <<CONTROLLER>>:

- Procesar Información del sistema.
- Gestionar visualización de respuesta del sistema.
- Obtiene los datos del modelo.



2.4.1 DIAGRAMA DE ESTRUCTURA ESTÁTICA.

Con el nombre de Diagrama de Estructura Estática se engloba tanto el Modelo Conceptual de la fase de Análisis como el Diagrama de Clases de la fase de Diseño. Ambos son distintos conceptualmente, mientras el primero modela elementos del dominio el segundo presenta los elementos de la solución del software.

2.4.2 DIAGRAMAS DE CASOS DE USO.

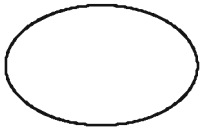
Los casos de uso fueron introducidos por Ivar Jacobson en 1994 y permiten realizar la especificación de un sistema centrada en el usuario. En esencia un caso de uso representa una interacción típica entre un usuario y un sistema o aplicación computacional. Para ilustrar lo anterior, supongamos un usuario utilizando un procesador de texto. Dos casos de uso típicos podrían ser "asignar negrita a una palabra" o "crear un índice". Con los ejemplos anteriores se pueden identificar algunas de las propiedades de los casos de uso:

- Capturan alguna funcionalidad visible para el usuario.
- Pueden ser pequeños o grandes.
- Satisfacen un objetivo del usuario

En su forma más simple, los casos de uso son identificados mediante conversaciones con el usuario y discutiendo las distintas operaciones que el sistema debe proveer. Cada funcionalidad u objetivo discreto debe ser documentado asignándole un nombre y una breve descripción.

La funcionalidad que describe un caso de uso determinado puede ser interpretada como una Interacción de Sistema u Objetivo de Usuario. La interacción de sistema permite describir las operaciones que el usuario realiza para satisfacer un objetivo. El objetivo del usuario, como su nombre lo indica, refleja un objetivo que el usuario desea satisfacer. Por ejemplo, siguiendo con el ejemplo del procesador de texto, algunas interacciones de sistema podrían ser "asignar formato a un párrafo", "definir un nuevo estilo" o "modificar un estilo". Sin embargo, éstas no reflejan el objetivo del usuario que podría ser "mantener un formato consistente para todos los documentos". Es necesario tomar en cuenta esta diferencia al momento de modelar los casos de uso.

Los casos de uso incluyen, además, Actores. Un actor representa una entidad externa que se relaciona directamente con el sistema. Los actores representan humanos, máquinas u otros sistemas. En definitiva, un actor corresponde al rol que juega alguna de las entidades anteriores frente al sistema. Puede haber muchos usuarios con un mismo rol. Un usuario también puede poseer distintos roles. Por esta razón, al considerar los actores del sistema se debe centrar el análisis en los roles y no en las personas.



Actor Caso de uso

Figura 2.6: El actor y el caso de uso son las entidades básicas del modelo de casos de uso]
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld]

Los actores son los que realizan los casos de uso. Un actor puede realizar varios casos de uso; alternativamente, un caso de uso puede ser realizado por varios actores.

Los actores apoyan la identificación de los casos de uso. Enfrentado a un sistema muy grande, intentar identificar la lista de casos de uso puede ser una tarea muy compleja. La alternativa adecuada es identificar la lista de actores del sistema y luego intentar identificar los casos de uso de cada actor.

2.4.3 DIAGRAMA DE SECUENCIA.

Muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo.

El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Se pueden colocar etiquetas (como restricciones de tiempo, descripciones de acciones, etc.) bien en el margen izquierdo o bien junto a las transiciones o activaciones a las que se refieren.

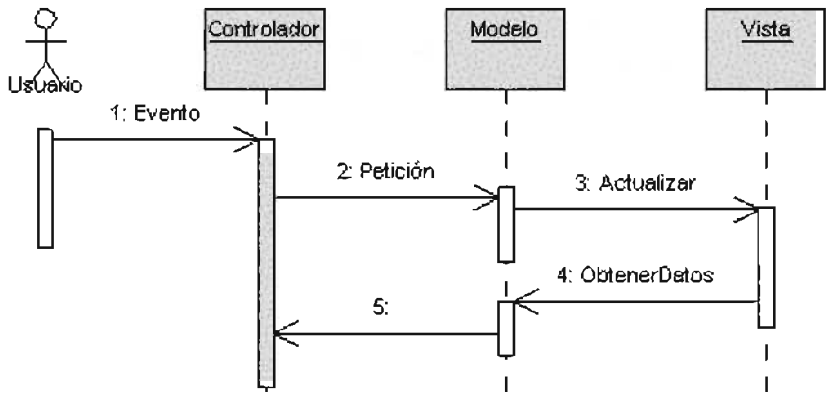


Figura 2.7: Pasos a seguir de una secuencia
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

Pasos:

- El usuario introduce el evento;
- El Controlador recibe el evento y lo traduce en una petición al Modelo (aunque también puede llamar directamente a la vista);
- El modelo (si es necesario) llama a la vista para su actualización;
- Para cumplir con la actualización la Vista puede solicitar datos al Modelo;
- El Controlador recibe el control.

2.5 METODOLOGIA OBJETORY (JACOBSON ET AL 1992).

La metodología a ser utilizada será Object Oriented Software Engineering [Jacobson, 1992] (OOSE/Objetory), que es una metodología orientado a objetos, ya que los sistemas orientados a objetos se desarrollan alrededor de entidades del dominio del problema, lo cual resulta desarrollos bastante estables. El análisis orientado a objetos, a diferencia del estructurado, considera comportamiento y datos de forma separada, combina ambos. Las características principales de los orientados a objetos son que ofrecen una forma de pensar más que una forma de programar. Además, reducen la complejidad en el diseño de software, y permiten atacar los errores durante el diseño en lugar de durante la implementación, donde el costo de reparación es bastante mayor.

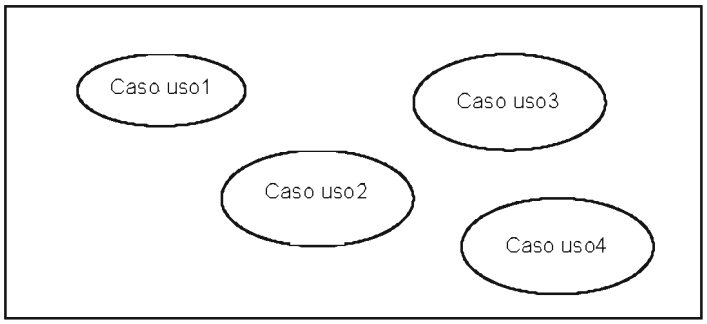
La metodología Objetory involucra las siguientes fases:

- **Requisitos,** el modelo de casos de uso sirve para expresar el modelo de requisitos, el cual se desarrolla en cooperación con otros modelos como se verá más adelante;

jacarowen@hotmail.com

- **Análisis,** la funcionalidad especificada por el modelo de casos de uso se estructura en el modelo de análisis, que es estable con respecto a cambios, siendo un modelo lógico independiente del ambiente de implementación;
- **Diseño,** la funcionalidad de los casos de uso ya estructurada por el análisis es realizada por el modelo de diseño, adaptándose al ambiente de implementación real y refinándose aún más;
- **Implementación,** los casos de uso son implementados mediante el código fuente en el modelo de implementación;
- **Pruebas,** los casos de uso son probados a través de las pruebas de componentes y pruebas de integración;
- **Documentación,** el modelo de casos de uso debe ser documentado a lo largo de las diversas actividades, dando lugar a distintos documentos como son los manuales de usuario, manuales de administración, etc.

El diagrama de la figura 2.8 ilustra las relaciones entre los distintos modelos. Describiremos los detalles y la notación más adelante. Nótese que los modelos dependientes del modelo de casos de uso tienen una dependencia entre si. Existe también dependencia entre los propios modelos.



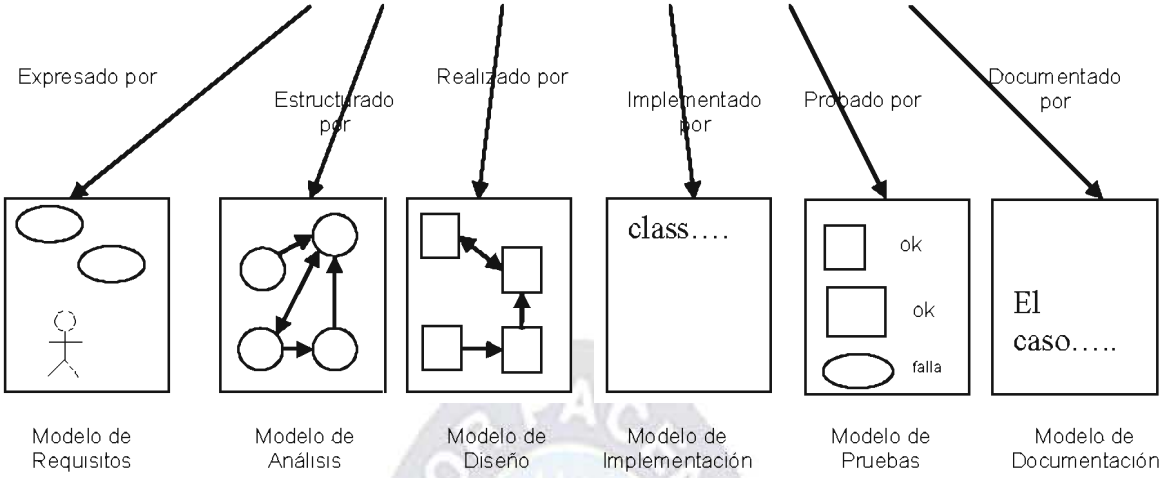
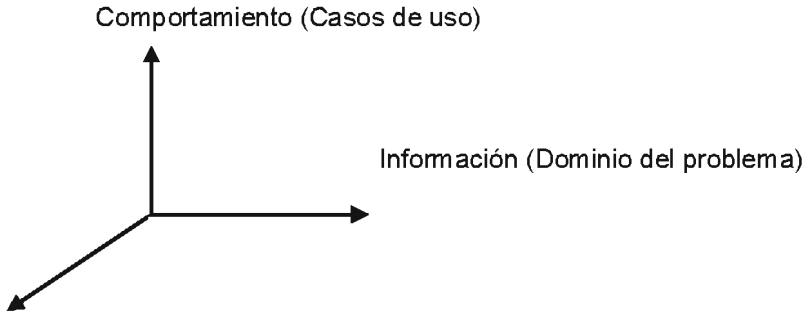


Figura 2.8: Dependencia de los distintos modelos de proceso de software del modelo de casos de uso
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld]

2.5.1 MODELO DE REQUISITOS.

Tiene como objetivo delimitar el sistema y capturar la funcionalidad que ofrecerá desde la perspectiva del usuario. Es el primero en desarrollarse, y es la base para formar los demás modelos en el desarrollo del software. El propósito del modelo de requisitos es comprender en su totalidad el problema y sus implicaciones. Los demás modelos, análisis, diseño, implementación y pruebas dependen directa o indirectamente del modelo de requisitos. Así mismo, este modelo sirve para el desarrollo de las instrucciones operacionales y los manuales, ya que todo lo que el sistema debe hacer se describe aquí desde la perspectiva del usuario.

En la metodología de Objetory, el modelo de requisitos consiste de tres modelos principales, visualmente representado por un diagrama de tres dimensiones como se muestra en la figura 2.9.



Presentación (Interfaces)

Figura 2.9: Los tres ejes del modelo de requisitos
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld]

- **EL MODELO DE COMPORTAMIENTO,** basado directamente en el modelo de casos de uso, especifica la funcionalidad que ofrece el sistema desde el punto de vista del usuario;
- **EL MODELO DE PRESENTACIÓN O MODELO DE INTERFACES,** el cual especifica cómo interactúa el sistema con actores externos al ejecutar los casos de uso, en particular, especifica cómo se verán visualmente las interfaces gráficas y que funcionalidad ofrecerán cada una de ellas;
- **EL MODELO DE INFORMACIÓN,** el cual especifica los aspectos estructurales del sistema. Este modelo conceptualiza el sistema en base a los objetos que representan las entidades básicas de la aplicación.

La separación en tres ejes de modelado independientes es la base para una mayor estabilidad en el desarrollo del sistema, permitiendo minimizar los efectos entre cada uno de ellos.

2.5.2 MODELO DE ANÁLISIS.

El objetivo del modelo de análisis es comprender y generar una arquitectura de objetos para el sistema en base a lo especificado en el modelo de requisitos. En otras palabras el análisis pretende modelar el sistema bajo condiciones ideales, garantizando que la arquitectura de software sea robusta y extensible para servir de base a la estructura lógica de la aplicación.

El modelo de análisis genera una representación conceptual del sistema, consistiendo de clases de objetos. Cada una de las clases de objetos contribuye de manera especial para lograr la robustez de la arquitectura, como se muestra conceptualmente en la figura 2.10.

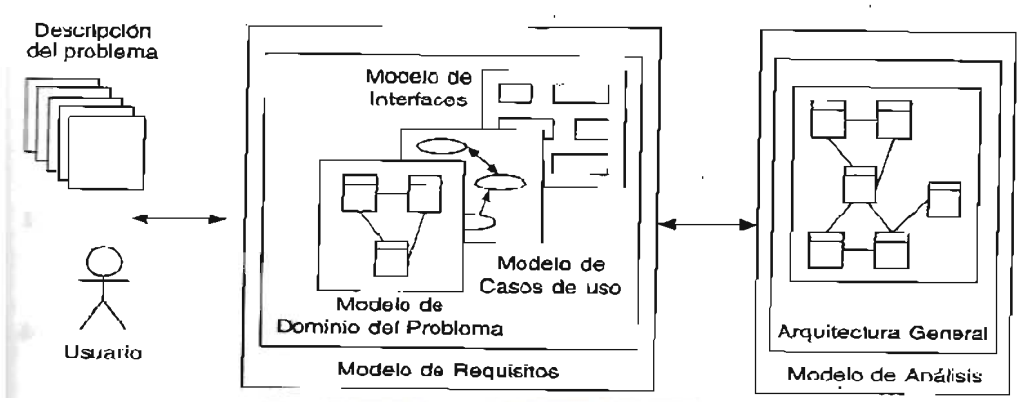


Figura 2.10: Modelo de análisis junto con la arquitectura general de objetos en relación al modelo de requisitos
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld]

ARQUITECTURA DE CLASES.

El modelo de análisis tiene como objetivo generar una arquitectura de objetos que sirva como base para el diseño del sistema. Las arquitecturas se distinguen según la organización de la funcionalidad que ofrecen los objetos dentro de ellas o la dimensión de los objetos. Esta dimensión corresponde a los diferentes tipos de funcionalidad que manejan los objetos dentro de la arquitectura.

En el caso de los sistemas de información, una de las arquitecturas más utilizadas es la de Modelo, Vista, Control (MVC – Model, View, Control) popularizada por los ambientes de desarrollo para los lenguajes de programación de Smalltalk. Esta arquitectura se basa en tres dimensiones principales Modelo correspondiente a la información, Vista correspondiente a la presentación o interacción con el usuario y Control correspondiente al comportamiento como se ilustra en la figura 2.11.

Control (comportamiento)

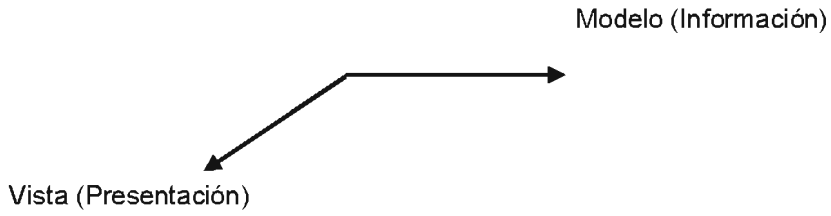


Figura 2.11: Diagrama de tres dimensiones correspondiente a la arquitectura Modelo, Vista, Control (MVC)
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

CLASES CON ESTEREOTIPOS.

El tipo de funcionalidad o "la razón de ser" de un objeto dentro de una arquitectura se conoce como su estereotipo. Siguiendo la metodología de casos de uso, la arquitectura del sistema para el modelo de análisis se basará en tres estereotipos:

- El estereotipo **entidad** (entity) para los objetos que guardan información sobre el estado interno del sistema a corto y largo plazo. Estos objetos corresponden al dominio del problema. Un ejemplo de objeto entidad es un registro de usuario con sus datos y comportamiento asociados.
- El estereotipo **borde** (boundary) para objetos que implementan las interfaces del sistema con el mundo externo, correspondientes a todos los actores, incluyendo a aquellos que no son humanos. Un ejemplo de objeto borde es una interface de usuario o pantalla para insertar o modificar información en el registro de usuario o pantalla para insertar o modificar información en el registro de usuario. Otro ejemplo es un objeto que se comunica con una base de datos externa al sistema.
- El estereotipo **control** (control) para objetos que implantan el comportamiento o control de la lógica de los casos de uso, especificando cuándo y como el sistema cambia de estado. Los objetos control modelan la funcionalidad que no se asocia naturalmente con un solo objeto. Un ejemplo típico de objeto control es validar un usuario existente o insertar uno nuevo. Este comportamiento requiere la interacción de múltiples objetos y actores.

Los tres estereotipos, Entidad, Borde y Control, correspondientes a las tres dimensiones de la arquitectura de análisis se muestran en la Figura 2.12

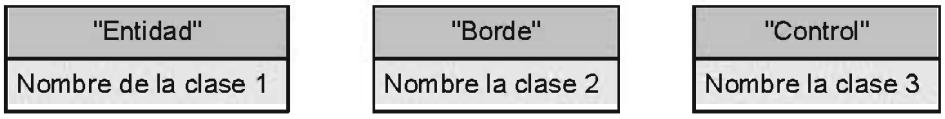


Figura 2.12: Diagrama de clases para los tres estereotipos
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld]

CLASES PARA CASOS DE USO.

Cuando se desarrolla el modelo de análisis, normalmente se trabaja con un caso de uso a la vez.

En cada caso de uso se identifican los objetos necesarios para su implementación. Los objetos se identifican según sus estereotipos de manera que correspondan con la funcionalidad ofrecida en cada uno. Se define explícitamente qué objeto es responsable de cuál comportamiento dentro del caso de uso. Se comienza identificando los objetos borde necesarios, continuando con los objetos entidad y, finalmente, los objetos control. Este proceso se continúa a los demás casos de uso. Dado que los mismos objetos pueden participar en varios casos de uso, durante este proceso es necesario identificar aquellos objetos comunes. Esto significa que cuando un conjunto de objetos ya existe, éstos pueden modificarse para ajustarlos al nuevo caso de uso. La meta es formar una arquitectura que reutilice el mayor número de objetos posible. De tal manera, la descripción original de los casos de uso se transforma en una descripción con base en los tres estereotipos de objetos, como se muestra en la figura 2.13.

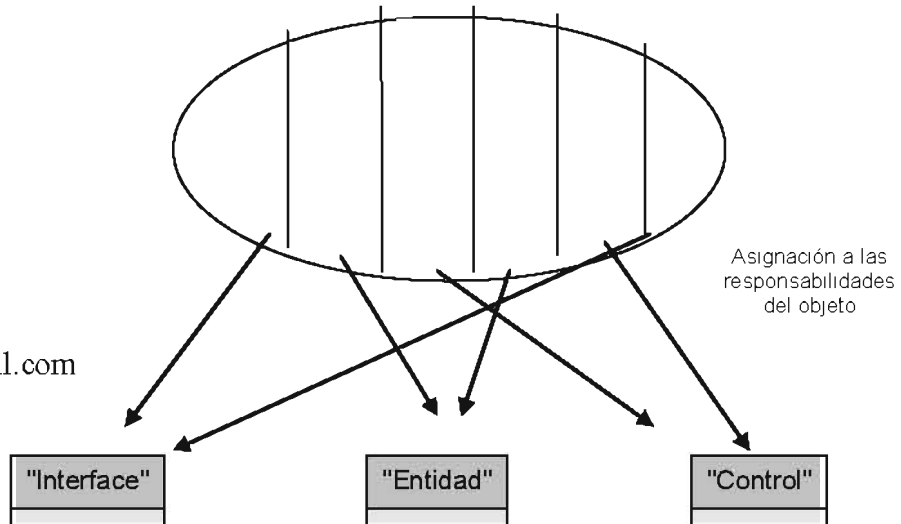


Figura 2.13: La funcionalidad de cada caso de uso se asigna a objetos distintos y de acuerdo con sus estereotipos
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

La asignación de objetos a cada caso de uso se hace de acuerdo con los siguientes principios:

- La funcionalidad de los casos de uso que depende directamente de la interacción del sistema con el mundo externo se asigna a los objetos borde;
- La funcionalidad relacionada con el almacenamiento y manejo de información del dominio del problema se asigna a los objetos entidad;
- La funcionalidad específica a uno o varios casos de uso y que afecta a múltiples objetos a la vez, o que no se relaciona naturalmente con ningún objeto borde o entidad, se asigna a los objetos control. En general, se asigna un solo objeto control por caso de uso. Si el caso de usos es muy complejo, se pueden asignar objetos de control adicional.

2.5.3 MODELO DE DISEÑO.

El modelo de diseño es un refinamiento y formalización adicional del modelo de análisis, donde se toman en cuenta las consecuencias del ambiente de implementación. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objeto, incluyendo sus operaciones y atributos. El modelo de diseño se basa en el diseño por responsabilidades.

Se requiere un modelo de diseño, ya que el modelo de análisis no es lo suficientemente formal para alcanzar el código fuente. Por tal motivo se refinan los objetos, incluyendo

las operaciones y atributos. El sistema real también debe adaptarse al ambiente de implementación.

Se considera el modelo de diseño como una formalización del espacio de análisis, extendiéndolo para incluir una dimensión adicional correspondiente al ambiente de implementación, como se puede ver en la siguiente figura 2.14:

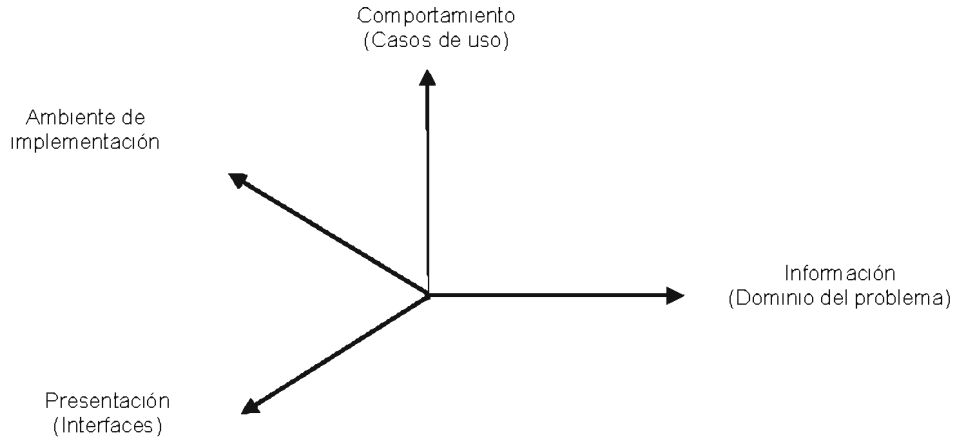


Figura 2.14: El diseño añade el ambiente de implementación como un nuevo eje de desarrollo.
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

Esta nueva dimensión, correspondiente al ambiente de implementación, se considera al mismo tiempo que se refina el modelo. La meta es refinarlo hasta que sea fácil escribir el código fuente. Como el modelo de análisis define la arquitectura general del sistema, se busca obtener una arquitectura detallada como resultado del modelo de diseño, de manera que haya una continuidad de refinamiento entre los dos modelos, como se ve en el diagrama de la Figura 2.15.

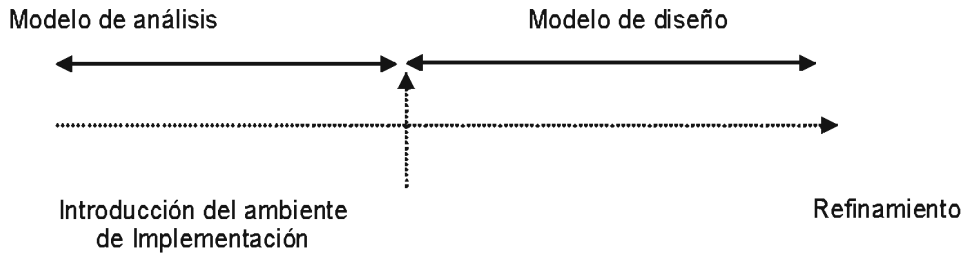


Figura 2.15: El modelo de diseño es una continuación del modelo de análisis
Fuente: ["Ingeniería de Software Orientada a Objetos" Dr. Alfredo Weitzenfeld**]**

La transición de análisis a diseño debe decidirse por separado para cada aplicación particular. Aunque es posible continuar trabajando sobre el modelo de análisis, incluso durante la incorporación del ambiente de implementación, no es recomendable, ya que aumenta su complejidad. Por tanto, es conveniente tener un modelo de análisis ideal del sistema durante el ciclo de vida del sistema, dado que muchos de los cambios del sistema provienen de cambios en el ambiente de implementación. Tales cambios se incorporan fácilmente, ya que el mismo modelo de análisis sirve de base para el nuevo modelo de diseño. De esta manera, el modelo de diseño se ve como una especialización del modelo de análisis según el ambiente de implementación específico.

2.5.4 MODELO DE IMPLEMENTACIÓN.

El modelo de implementación toma el resultado del modelo de diseño para generar el código final. Durante el modelo de implementación se adapta al lenguaje de programación y/o la base de datos, según la especificación del diseño y las propiedades del lenguaje de implementación y base de datos. Aunque el diseño de objetos es bastante independiente del lenguaje actual, todos los lenguajes tienen sus particularidades, las cuales deben adecuarse durante la implementación final. La elección del lenguaje influye en el diseño, pero éste no debe depender de los detalles

de aquél. Si se cambia de lenguaje de programación, no debe requerirse el rediseño del sistema.

Este flujo de trabajo implementa y lleva a cabo las pruebas de todos los componentes. El resultado, después de varias iteraciones, de la integración y pruebas del sistema, es la versión operativa inicial, que representa el 100 por cien de los casos de uso. En este flujo de trabajo en el que el proyecto lleva a cabo la mayor parte del trabajo en la fase de construcción. El mismo rellena cada componente con código, hasta que al finalizar la construcción de todos los componentes se encuentran con código.

También se realiza la vista de despliegue, que representa la disposición de las instancias de componentes de ejecución.

2.5.5 MODELO DE PRUEBAS.

Probar un producto es relativamente independiente de la metodología de desarrollo utilizada para construirlo.

Existen diversos tipos de pruebas aplicados durante las diferentes actividades del proceso de desarrollo, las cuales requieren de tiempo y presupuesto adicionales, que pueden llegar a significar un alto porcentaje del costo total. Por tal motivo, el modelo de pruebas debe ser planificado con anticipación y de manera integral junto con el desarrollo, ya que no se puede lograr software de alta calidad sólo mediante pruebas finales y depuraciones. Las mismas deben hacerse simultáneamente con el desarrollo del sistema. Además, las pruebas finales deben tener como objetivo la certificación final de la calidad del producto y no la búsqueda de errores. Detectarlos al final del desarrollo es bastante problemático, dado que ello requiere regresar a etapas anteriores para resolverlos. Se considera que “evitar defectos” es más importante que “removerlos”.

2.6 CALIDAD EN LA WEB.

Pressman define la calidad como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecido, los estándares de desarrollo explícitamente documentados y las características implícitas que se esperan de todo software desarrollado profesionalmente [PRE,1998].

La calidad de software es una compleja mezcla de factores que varían de acuerdo a la aplicación que se desarrolla y a los requerimientos de los clientes. El estándar ISO 9126 identifica cinco factores clave de calidad para el software, estos factores coinciden con los definidos para la evaluación de la calidad de aplicaciones web. Estos factores son:

Tabla 2.1 Árbol de requisitos de calidad

Factores de Calidad	Requisitos de Calidad
Usabilidad	Capacidad de comprensión del sitio global Servicios de ayuda y realimentación en línea Capacidades estéticas y de interfaz Servicios especiales
Fiabilidad	Proceso correcto de enlace Recuperación de errores Validación y recuperación de la entrada de usuario
Eficiencia	Rendimiento del tiempo de respuestas Velocidad de generación de páginas Velocidad de generación de gráficos
Capacidad de mantenimiento	Facilidad de corrección Adaptabilidad Extensibilidad
Portabilidad	Facilidad de instalación Facilidad de ajuste Facilidad de adaptación al cambio

Un gestor de proyectos de software debe evaluar la calidad objetivamente y no subjetivamente con el uso de métricas.

2.6.1 USABILIDAD.

La facilidad con que se usa el software de acuerdo con los siguientes subatributos: facilidad de comprensión, facilidad de aprendizaje operabilidad.

Es el esfuerzo necesario para comprender el sitio general, reparar los datos de entrada e interpretar las salidas. Es el intento por medir lo amigable que puede ser un programa con el usuario.

Patty [PAT,2003] define una tabla de requerimientos de usabilidad correspondiente a dominios web que detalla sub-características y atributos derivados de las características de Usabilidad (tabla 2.2). Estas características proveen un marco conceptual para especificar requerimientos de usabilidad.

Tabla 2.2 Características, sub-características y atributos de usabilidad [PAT03]

	Esquema de	Mapa de sitio		
	Organización	Tabla de contenidos		
Comprensibilidad	Global	Índice alfabético		
	Calidad en el sistema de etiquetado			
	Visita guiada orientada al visitante			
	Mapa de imagen (Campus /Edificio)			
	Calidad de la ayuda	Ayuda explicatoria Orientada al usuario		
		Ayuda de la Búsqueda		
	Indicador de última	Global (de todo el sitio web)		
	actualización	Restringido (por sub-sitio o página)		
Mecanismos de	Directorio de	Directorio e-mail		
ayuda y	direcciones	Directorio Tele-Fax		

retroalimentación	Facilidad FAQ			
		Cuestionario		
	Retroalimentación	Libro de visitantes		
		Comentarios/Sugerencias		
	Permanencia y	Permanencia de controles directos		
	estabilidad en la	Permanencia de controles indirectos		
	presentación de los			
Aspectos de	controles principales	Estabilidad		
interfaces y	Aspectos de estilo	Uniformidad en el color de enlaces		
Estéticos		Uniformidad en el estilo global		
	Preferencia estética			
	Indicador de resolución de pantalla			
	Performance	Páginas de acceso rápido		
			Soporte a versión solo texto	
Eficiencia		Accesibilidad de	Legibilidad al desactiv	Imagen con título
	Accesibilidad	Información	prop de imag de nave	Legibilidad global
		Accesibilidad de	Número de visitas considerando	
		ventanas	Marcos (frames)	
			Versión sin marcos	
		Orientación	Indicador del camino	
	Navegabilidad		Etiqueta de la posición actual	
		Promedio de enlaces por pagina		
Aspectos de		Permanencia y estabilidad	Permanencia de los controles	
navegación y	Objetos de control	en la presentación de	textuales	
Exploración	navegacional	los controles textua	Estabilidad	
		les (sub-sitio)		
	Precicción	Enlace con título (enlace con texto explicatorio)		
	navegacional	Calidad de la frase de enlace		

	Errores de enlaces	Enlaces rotos		
		Enlaces no implementados		
Errores		Deficiencias o cualidades ausentes debido a diferentes		
	Errores o	navegadores		
	deficiencias	Nodos destinos (inesperadamente) en construcción		
	varias	Nodos Web muertos (sin enlaces de retorno)		

Para cada uno de los atributos A_i definidos, se establecen diversos valores, parámetros y criterios elementales de evaluación cuantificables.

Se pueden representar los criterios elementales a través de una notación gráfica para cada atributo en la que presenten los puntos de mayor interés su preferencia correspondiente. En la figura 2.16 se aprecia la escala de preferencia para un atributo denominado Tiempo promedio de respuesta.

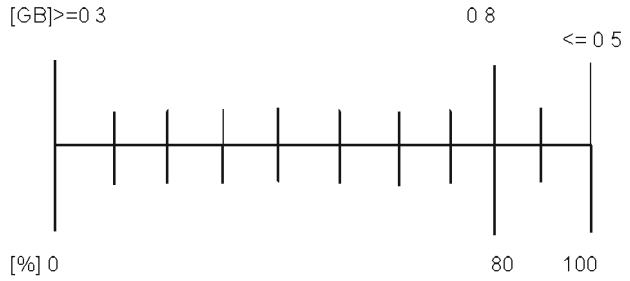


Figura 2.16: Escala de preferencia para el atributo Tiempo promedio de respuesta
Fuente: [Patty & Cruz, et al]

Es posible establecer una representación rotacional de las coordenadas de los puntos más relevantes a través de la interpolación lineal entre dos puntos. Una representación para el criterio de la figura 10

$$CrE(t_i) = \{(t_{i\,min}, 100), (t_{i\,max}, 0)\} \text{ ó } CrE(t_i) = \{(0.5, 100), (0.8,80) (3,0)\}$$

Una interpretación de dichos valores y sus preferencias asociadas es: "un tiempo promedio de respuesta menor o igual a 0.5 segundos satisface totalmente el

requerimiento, 0.8 segundos satisface un 80% del requerimiento, y un tiempo mayor o igual a 3 segundos es totalmente inaceptable.

Luego de computar el valor de X_i que modela el requerimiento del valor A_i a través de la métrica $m = A_i \rightarrow X_i$ el valor de preferencia caerá en uno de los tres niveles de aceptabilidad o barras de calidad. Esto es, insatisfactorio (de 0 a 40%) y satisfactorio (desde 60% a 100%).

2.6.2 EFICIENCIA.

Eficiencia se entiende como la capacidad del sistema para proporcionar tiempos de respuesta, tiempos de proceso y potencia apropiados bajo condiciones determinadas. Desde el punto de vista de la utilización de recurso, se considera también como la capacidad para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas. Para la obtención del grado de eficiencia del sistema, han de considerarse los cuatro atributos establecidos en la tabla 2 las cuales son: páginas de acceso rápido, soporte a versión solo texto, imagen con título, legibilidad global.

El grado en que el software emplea en forma óptima los recursos del sistema, como lo indican los siguientes subatributos: comportamiento en el tiempo, comportamiento de los recursos.

2.6.3 MANTENIMIENTO.

La facilidad con se repara el software de acuerdo con los siguientes subatributos: facilidad de análisis, facilidad de cambio, estabilidad facilidad de prueba.

jacarowen@hotmail.com

2.6.4 PORTABILIDAD.

La portabilidad se entiende como la facilidad de transferir un producto a diferentes entornos hardware/software, sin necesidad de aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito pro el propio software considerado. También es considerado como la capacidad del producto de software para ser usado en lugar de otro producto software, para el mismo propósito, dentro del entorno. Las características más importantes que se consideran para este factor son: la facilidad de instalación, facilidad de ajuste y la facilidad al cambio.

CAPITULO 3
ANALISIS Y DISEÑO DEL SISTEMA

3.7 ANALISIS DEL SISTEMA.

3.7.1 REQUERIMIENTOS DEL SISTEMA.

El sistema a desarrollar debe cumplir con los siguientes requerimientos:

- Registro de los clientes que tienen un proceso judicial abierto y activo.
- Registro de nuevos clientes asignando una clave al registrarse.
- Registro de abogados al Estudio Jurídico Belzu & Ortuño.
- Seguimiento y control de pagos que efectúa el cliente.
- Registro de los distintos documentos que se elaboran y reciben en torno a un caso en proceso judicial.
- Registro de personal asignado al caso.
- Seguimiento de casos de los clientes.

El sistema se dividirá en los siguientes procesos:

- Proceso de Registro de Clientes.
- Proceso de Registro de Abogados.
- Proceso de registro de Usuarios.
- Proceso de Registro de Apuntes.
- Proceso de Seguimiento a Casos
- Proceso de Asignación de permisos.
- Proceso de Registro de Casos.
- Proceso de Asignación de Servicios,
- Proceso de Preguntas Frecuentes y Respuestas.

Cada proceso contará con procedimientos de registro de información así como las consultas y reportes correspondientes.

3.7.2 **MODELO DE OBJETOS.**

Este modelo nos permitirá conocer las distintas tareas y roles que cumplen los actores vinculados al estudio jurídico.

3.7.2.1 DIAGRAMAS DE CASOS DE USO.

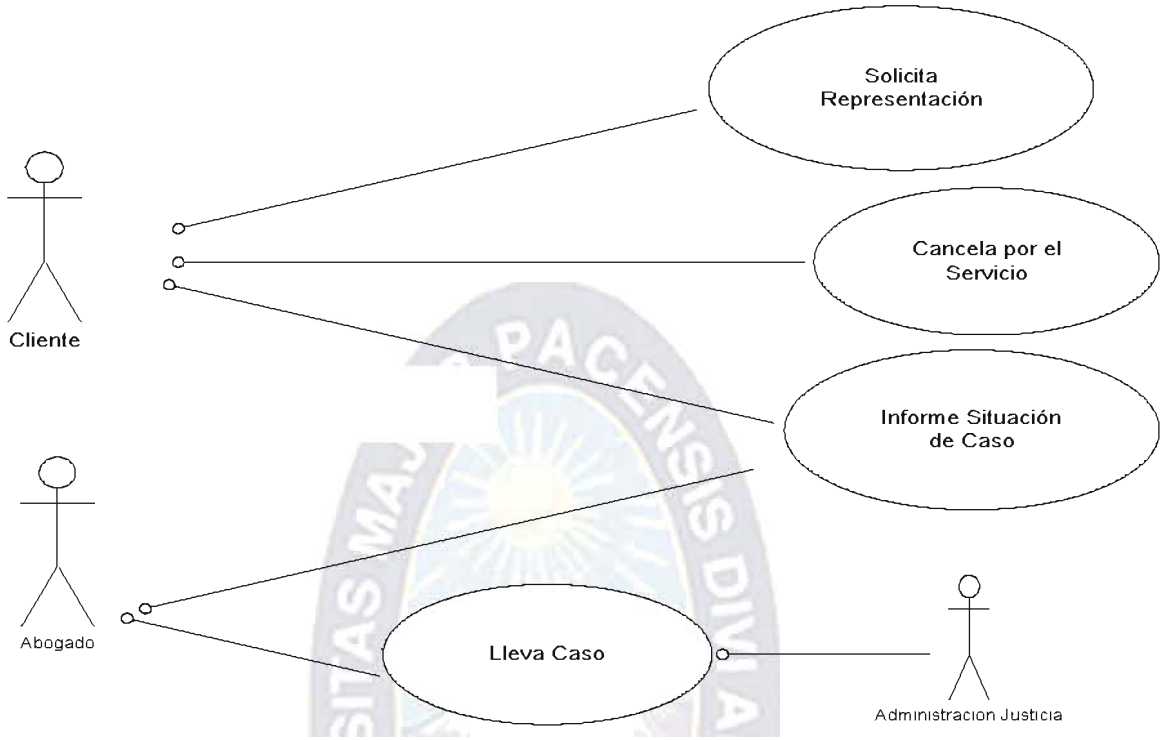


Figura 3.1: Diagrama de Casos de Uso, Nivel General
Fuente: [Elaboración Propia]

Un cliente interactúa con el Estudio Jurídico de la siguiente forma:

- Solicita representación jurídica mediante el Estudio Jurídico Belzu & Ortuño ante la Justicia Ordinaria.
- Efectúa cancelaciones de pago por concepto del servicio jurídico.
- Obtiene información completa detallada acerca de la situación y estado de su caso previa identificación en la aplicación.

El staff de abogado tiene por lo general los siguientes roles fundamentales:

- El abogado puede gestionar su despacho de abogados, obteniendo información de una manera rápida y sencilla, además conocer el estado actual o pasado de cualquier caso o cliente.

- Representa a un cliente frente a la Administración de Justicia.
- Informa al cliente la situación estado del caso que representa.

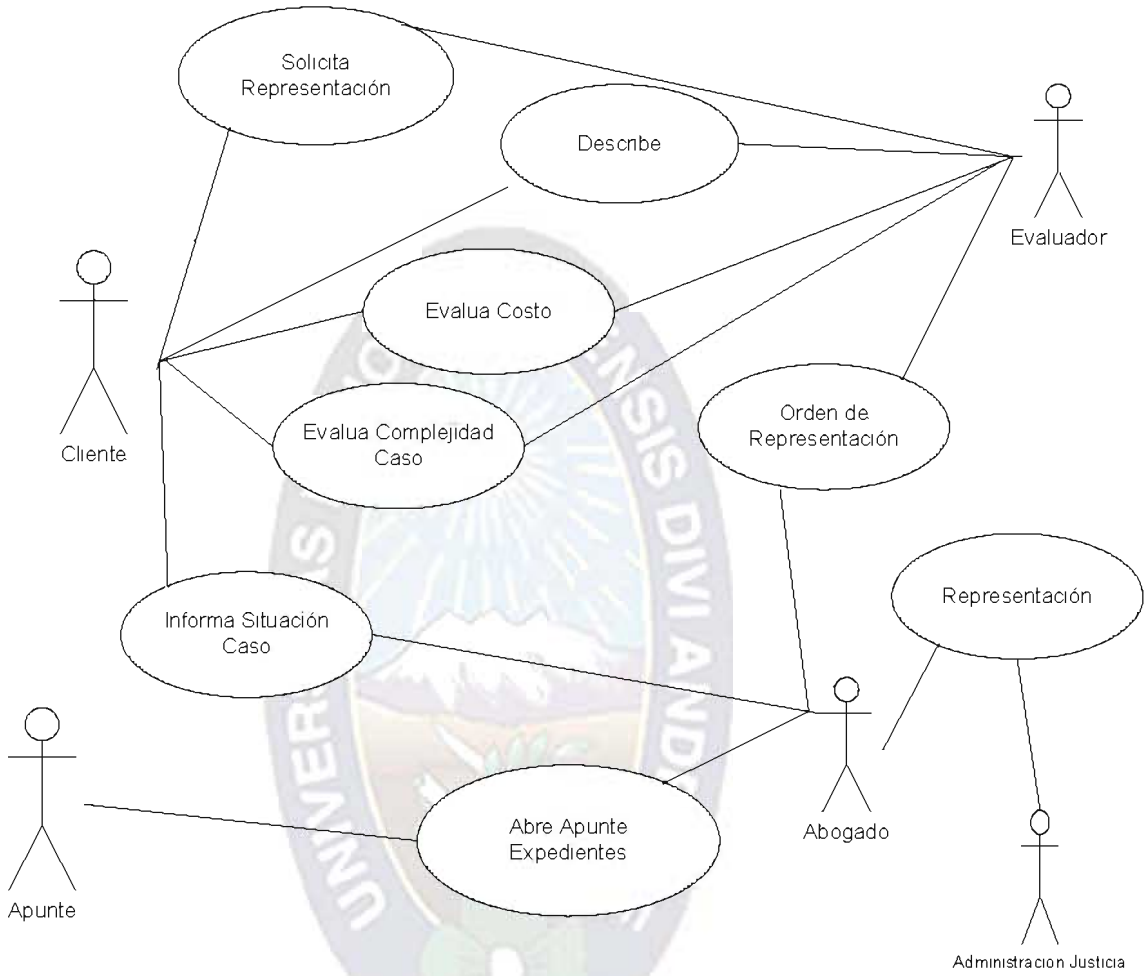


Figura 3.2: Diagrama de Casos de Uso, Solicitud de Representación
Fuente: [Elaboración Propia]

Cuando el cliente solicita la Representación de Servicios:

- Describe el caso o problema ante el abogado evaluador.
- El abogado evalúa la complejidad del caso y el costo en función a la complejidad, las mismas que informa el cliente.
- Si la representación es aprobada, asigna el caso a uno o más abogados del staff que tiene el Estudio Jurídico.
- El abogado obtiene la responsabilidad de la representación del cliente ante la Administración de Justicia.

- El abogado informa sobre los detalles de cada paso de cómo se llevará adelante el caso del cliente.

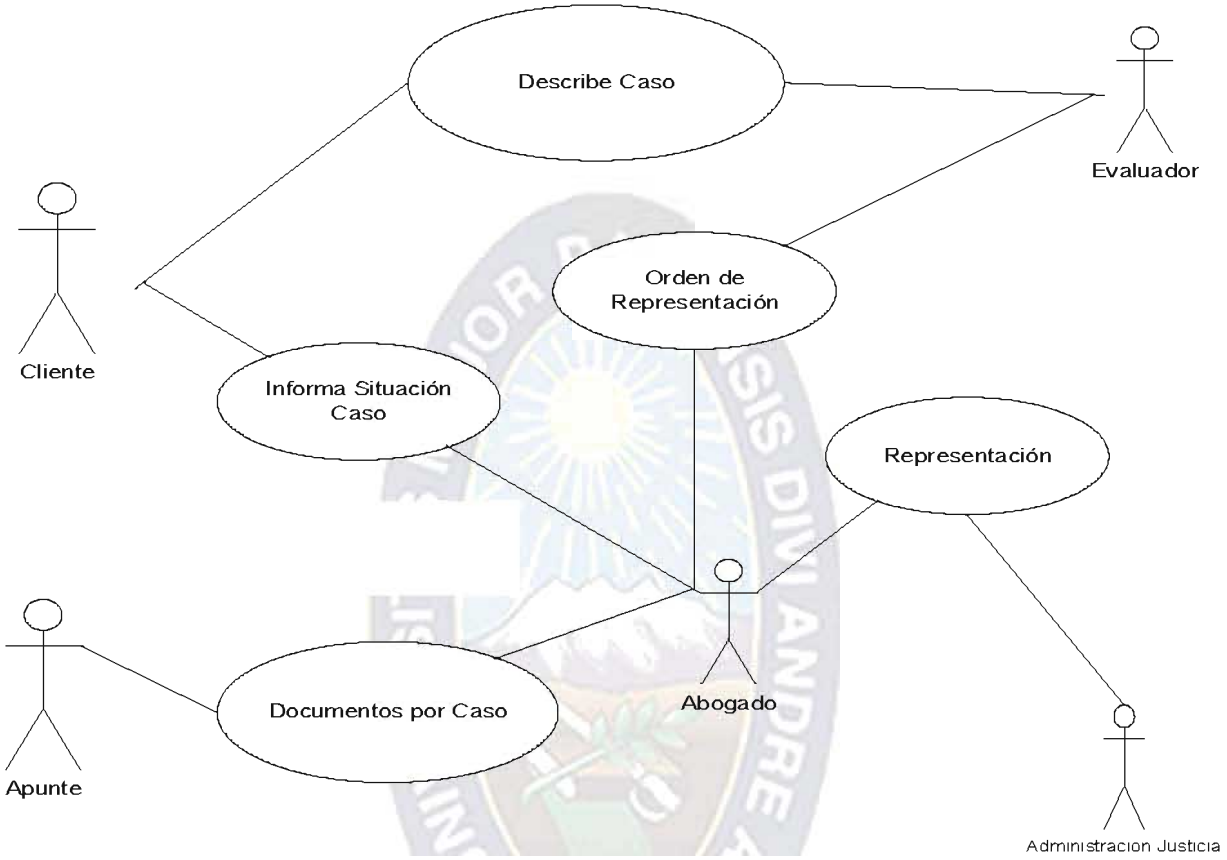
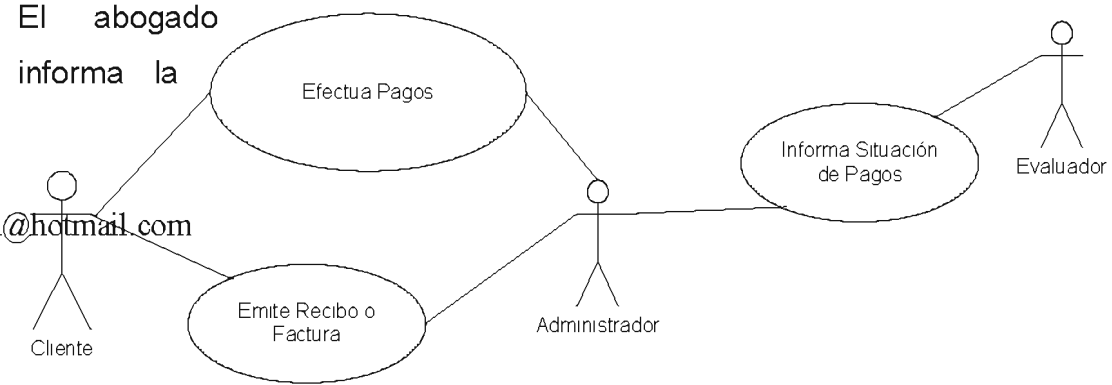


Figura 3.3: Diagrama de Casos de Uso, Abogado Obtiene la Representación.
Fuente: [Elaboración Propia]

Cuando el abogado esta autorizado para representar el caso:

- El abogado evaluador aprueba la representación del cliente.
- El Staff de abogados recibe la orden de representación.
- El abogado hace el trabajo de representación ante la administración de justicia.
- El abogado elabora resume los documentos necesarios, estos son archivados en el expediente del caso.
- El abogado archiva en el expediente las notificaciones de audiencia.
- El abogado informa la



situación y estado del caso al cliente.

Figura 3.4: Diagrama de Casos de Uso, Cliente Efectúa Pagos por el Servicio.
Fuente: [Elaboración Propia]

Cuando el cliente efectúa sus pagos por el servicio prestado:

- El cliente realiza un pago que puede ser total o parcial por los servicios prestados.
- El administrador extiende recibos por los pagos parciales y factura por el pago total, los cuales se le extiende al cliente.
- El administrador informa sobre la situación de un cliente referente a sus pagos por los servicios prestados.

De acuerdo a las actividades descritas se tiene los siguientes actores:

Cliente. Persona o entidad que se presenta en el Estudio Jurídico para solicitar su representación ante la justicia ordinaria como demandado o demandante.

Abogado Evaluador. Es el abogado en jefe, profesional al frente del staff de abogados, tiene la facultad de supervisar todos los casos que se representan o llevan a cabo en el estudio jurídico. Lleva también la administración de la entidad.

Abogado. Es la persona miembro del grupo jurídico, tiene la responsabilidad de ejercer su profesión para representar al cliente ante la justicia ordinaria de modo imparcial defendiendo los derechos del demandado o demandante.

Apunte. Ente que mantiene expedientes de casos en orden, cuya responsabilidad es la de no extraviar ningún documento, permite efectuar consultas en relación al seguimiento del caso e estado en que se encuentra el mismo.

Administrador. Profesional encargado de realizar un seguimiento de todas las cuentas pendientes de pago por cobrar pagar, elabora informe de financiero del Estudio Jurídico.

Administrador de Justicia. Ente externo al Estudio Jurídico, hace el papel oponente, dueño de los códigos de artículos de los códigos de procedimiento judicial. Esté se encuentra distribuido en distintas materias tiene varios representantes en distintas instancias.

3.7.3 DIAGRAMA DE CLASES.

Se definen las siguientes generalizaciones:

CLASE DOCUMENTOS.

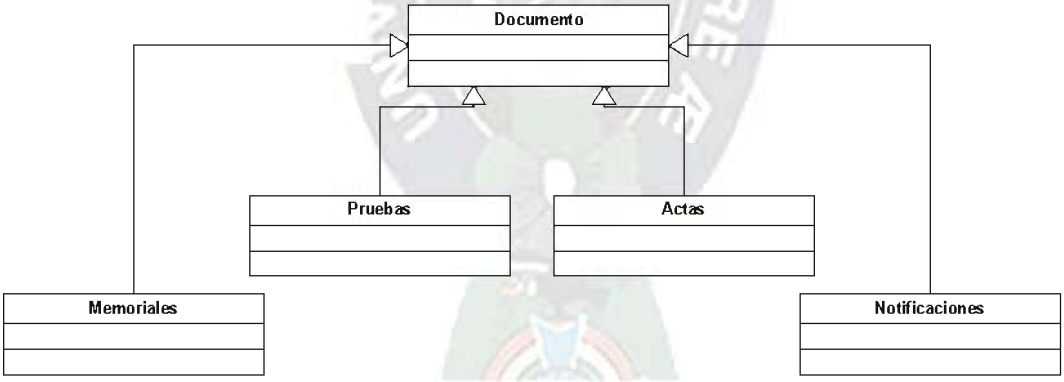


Figura 3.5: Diagrama de Clases: Generalización documentos.

Por lo general existen cuatro tipos de documentos que se archivan en un expediente de casos:

- Memoriales;
- Pruebas de cargo o descargo;

- Notificaciones y
- Actas.

CLASES SERVICIOS Y TIPO DE CASOS.

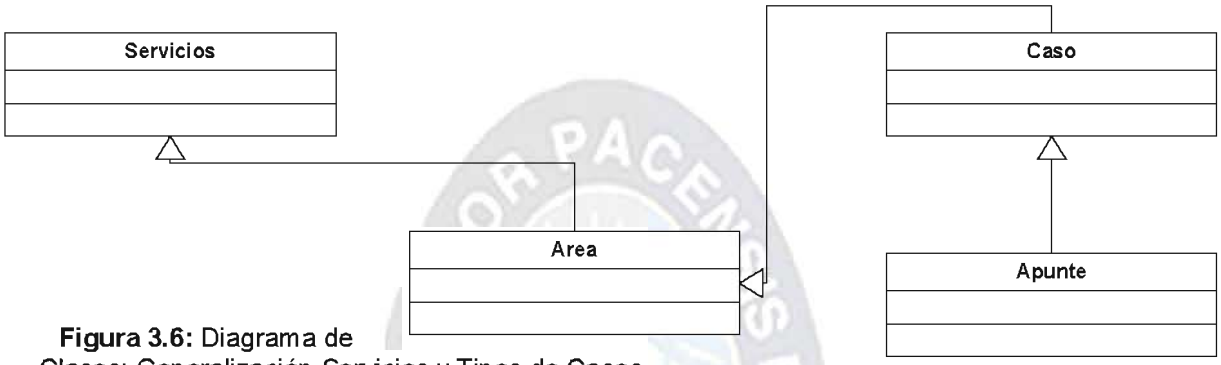


Figura 3.6: Diagrama de Clases: Generalización Servicios y Tipos de Casos.
Fuente: [Elaboración Propia]

Cuando se habré un expediente de caso el mismo pertenece a un tipo de caso o área en particular, el mismo permitirá conocer el procedimiento a seguir, dependiendo al servicio al cual corresponda o pertenezca (Derecho Familiar, Civil, Penal, etc.)

DIAGRAMA DE CLASE EN GENERAL.

A partir de lo desarrollado se puede brindar una primera versión del diagrama de clases correspondiente a la problemática del Estudio Jurídico Belzu & Ortuño.

Figura 3.7: Diagrama de Clases en General: Estudio Jurídico Belzu & Ortuño.

IDENTIFICACION DE OBJETOS.

jacarowen@hotmail.com

- **Servicios.** Contiene los distintos servicios del entorno judicial: Familiar, Civil, Penal, Procesos Administrativos, etc.
- **Áreas.** Contiene las áreas en que se dividen los servicios (Ejemplo Derecho Familiar se divide en: Divorcio, Divorcio Mutuo, etc.).
- **Casos.** Contiene los distintos casos registrado por un cliente.
- **Etapa.** Contiene registradas las distintas etapas en que se encuentra un caso.
- **Evolución.** Registra la evolución o desarrollo de cada caso de un cliente en particular.
- **Cliente.** Registra los datos generales de un cliente.
- **Abogado.** Registra los datos de un abogado.
- **Permisos.** Registra los permisos de cada usuario al ingreso del sistema.
- **Proceso.**
- **Tipo Apunte.** Registra todos los documentos que el cliente o abogado adjunten en un caso en particular.
- **Usuario.** Registra los usuarios que pueden ingresar al sistema.
- **Preguntas.** Registra las preguntas de todos los usuarios que ingresen a la pagina sean clientes o no.
- **Respuestas.** Registra las respuestas que el abogado responde a cada pregunta de los usuarios.

3.7.4 **DIAGRAMA DE SECUENCIAS**.

CLIENTE SOLICITA REPRESENTACION.

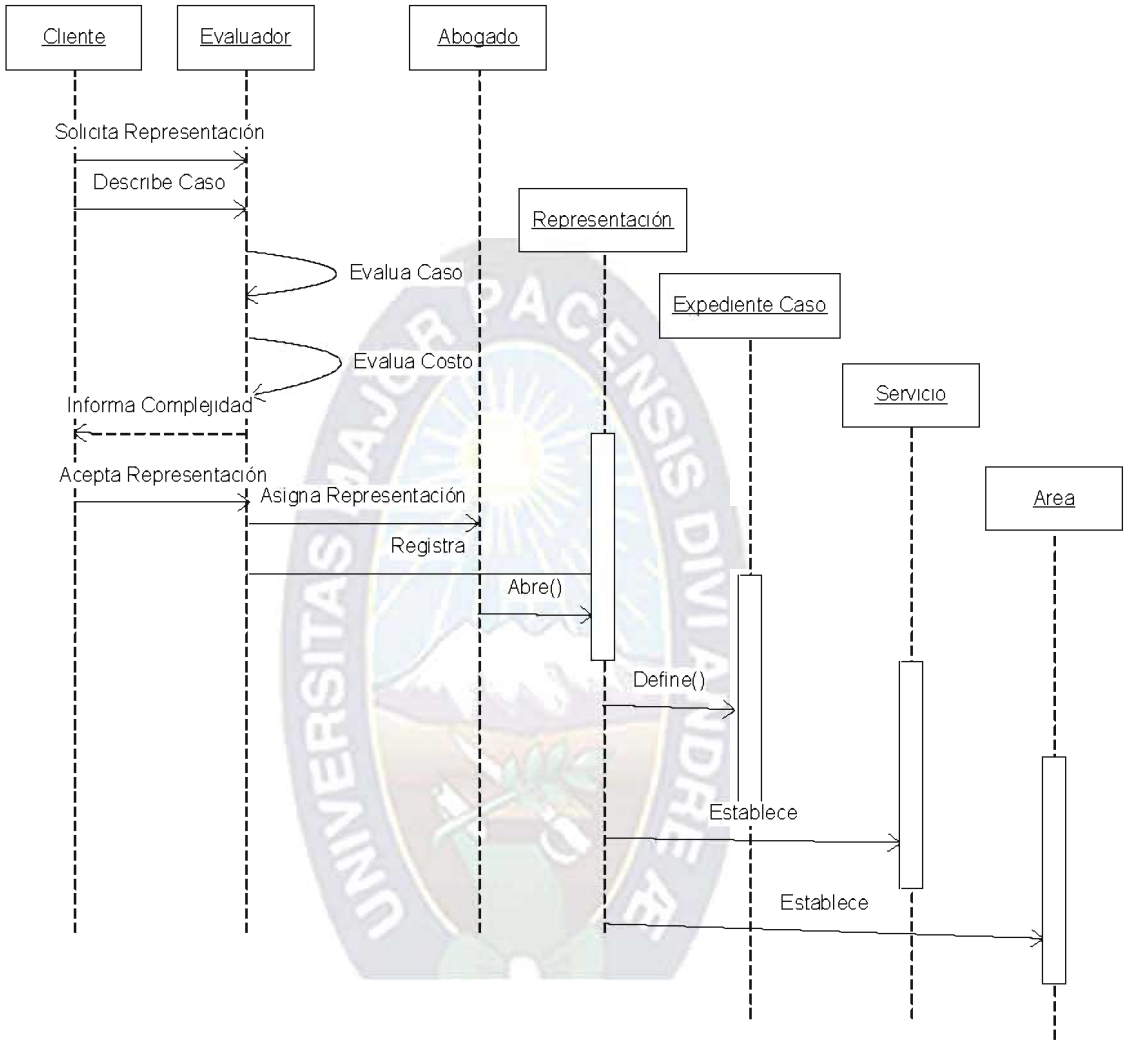


Figura 3.8: Diagrama de Secuencia: Cliente Solicita Representación.
Fuente: [Elaboración Propia]

- El Cliente solicita la representación al abogado Evaluador.
- El Cliente describe el caso a representar.
- El Evaluador evalúa el caso.
- El Evaluador evalúa el costo.
- El Evaluador informa al cliente la complejidad el costo del caso.
- El cliente acepta la representación al Evaluador.

- El Evaluador asigna la representación a uno o mas Abogados.
- El Evaluador registra la representación.
- El Abogado define el Tipo de Caso al cual corresponde.
- El Abogado establece el servicio que corresponde.
- El Abogado estable el Área la cual corresponde.
- El Abogado estudia el caso.
- El Abogado informa la situación del caso al Cliente.

SEGUIMIENTO DE CASO.

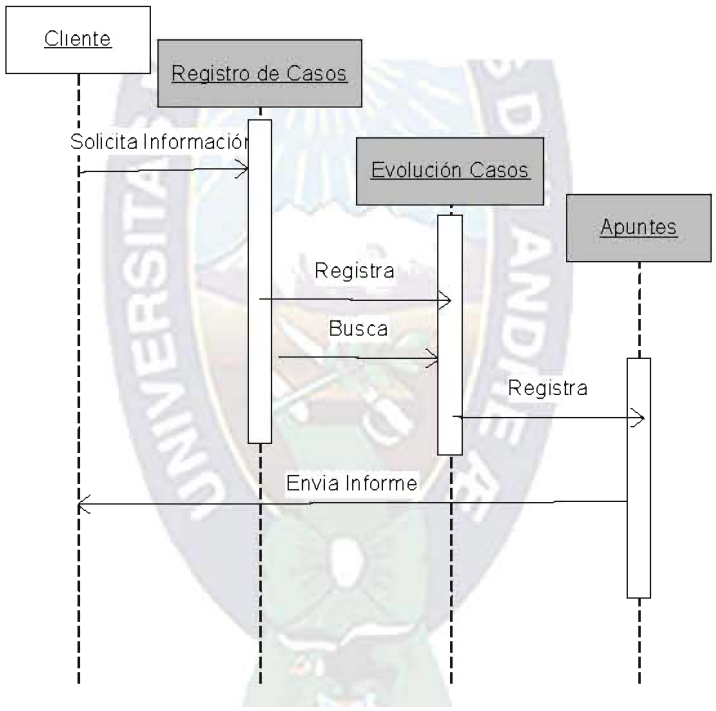


Figura 3.9: Diagrama de Secuencia: Seguimiento de Caso.
Fuente: [Elaboración Propia]

- Cliente solicita Información de su caso en particular.
- Registro de Casos registra la solicitud busca información del caso.
- Evolución de Caso registra busca información en Apuntes.
- Apuntes envía informe a cliente de su caso.

3.7.5 DIAGRAMA DE ACTIVIDADES.

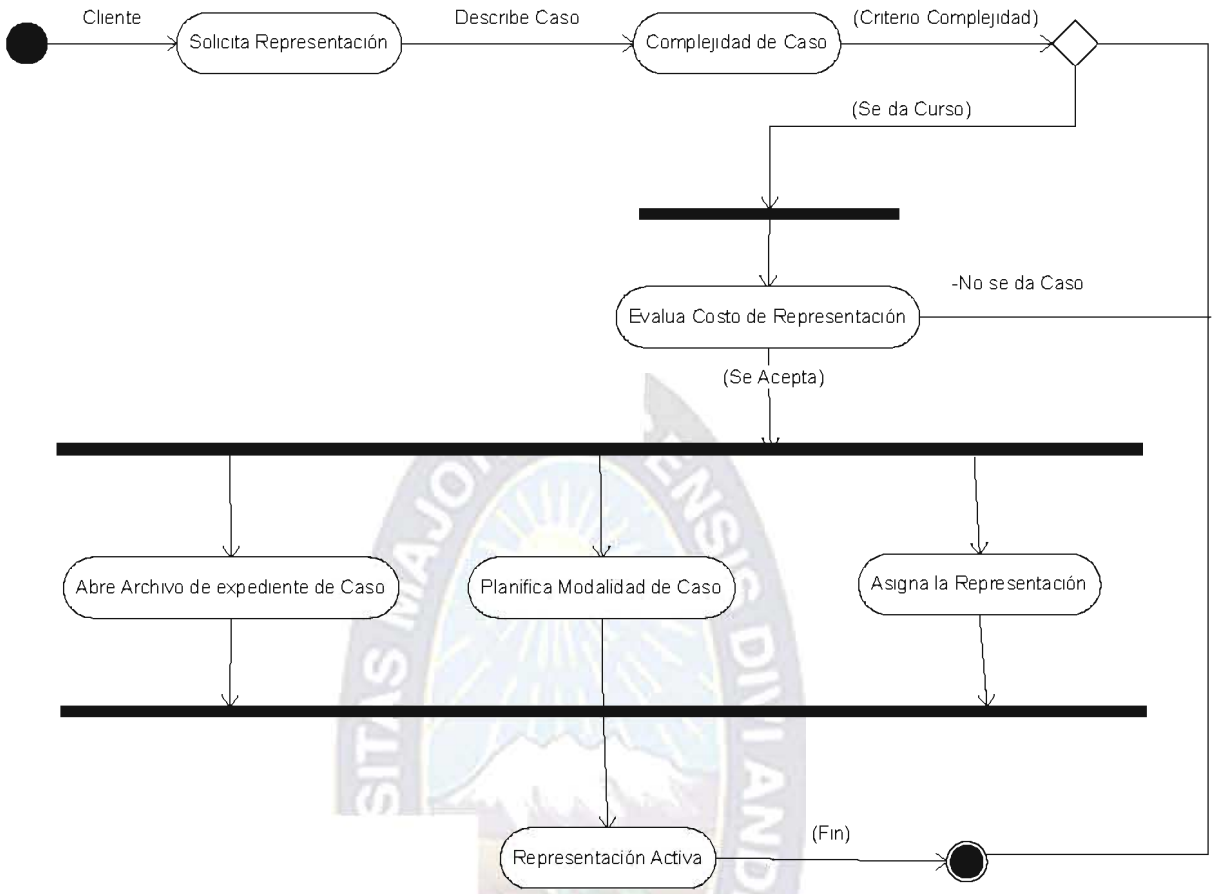


Figura 3.10: Diagrama de Actividades: Solicitud de Representación Jurídica.
Fuente: [Elaboración Propia]

Para tener una **Representación Activa** se requiere que estén conclusas las actividades. Abre Archivo de Expediente, Planifica Modalidad de Pago, Asigna la Representación. Las actividades Solicita Representación, Complejidad del Caso Evalúa costo son iniciales tienen la característica de un criterio de evaluación profesional, en las distintas etapas puede dar a lugar el abandonar el proceso de Representación Activa.

PAGO DE CLIENTES.

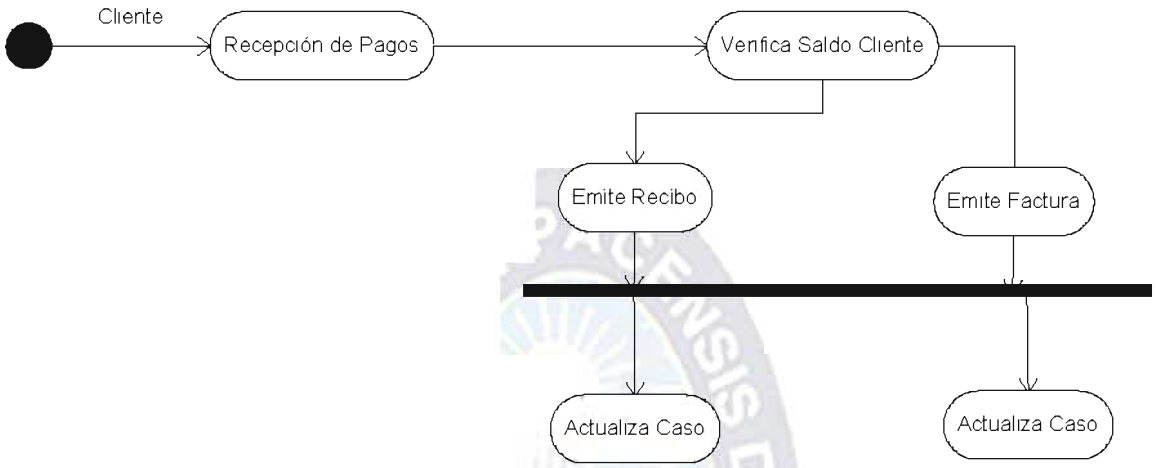


Figura 3.11: Diagrama de Actividades: Pago de Clientes.
Fuente: [Elaboración Propia]

La Actividad Pago de Clientes concluye con la Actualización del Caso y la emisión de un recibo por el pago efectuado o en su defecto una Factura si es que el saldo del cliente es igual a cero.

SEGUIMIENTO DE CASO.

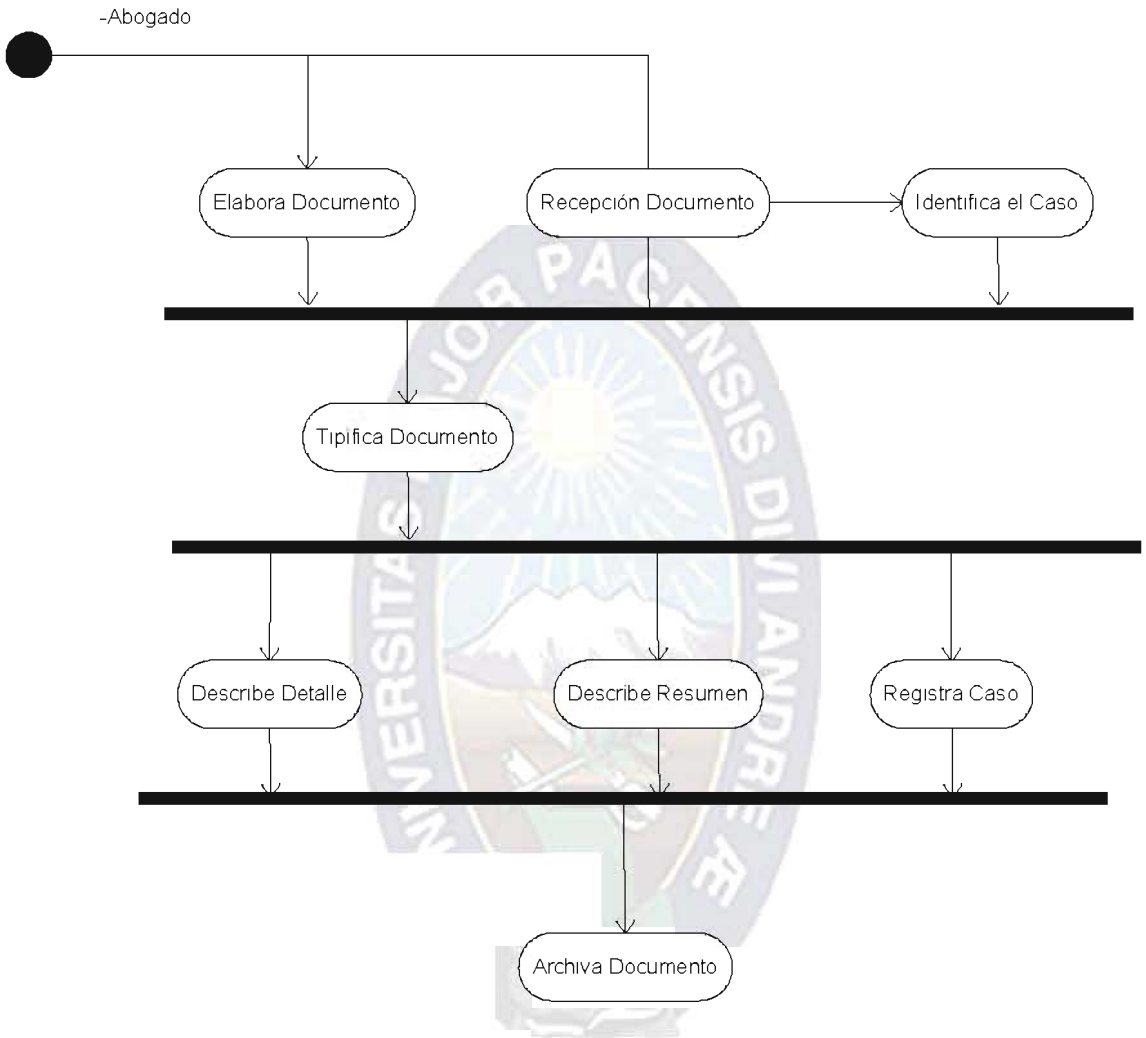


Figura 3.12: Diagrama de Actividades: Seguimiento de Casos.
Fuente: [Elaboración Propia]

El Estudio Jurídico puede elaborar o recepcionar documentos vinculados a un caso en proceso judicial. Cuando el documento viene del administrador de justicia se debe identificar el caso al cual pertenece, en ambos casos se tipifica el documento para tener un documento en archivo de expediente o apunte de documentos, se debe describir al detalle al registrar el mismo.

3.8 DISEÑO DEL SISTEMA.

Descomposición del Sistema en Subsistemas.

El Sistema para el Estudio Jurídico Belzu & Ortuño contiene los siguientes procesos:

- Clientes;
- Abogado;
- Casos;
- Apuntes,
- Usuarios,
- Permisos.
- Preguntas Frecuentes y Respuestas.

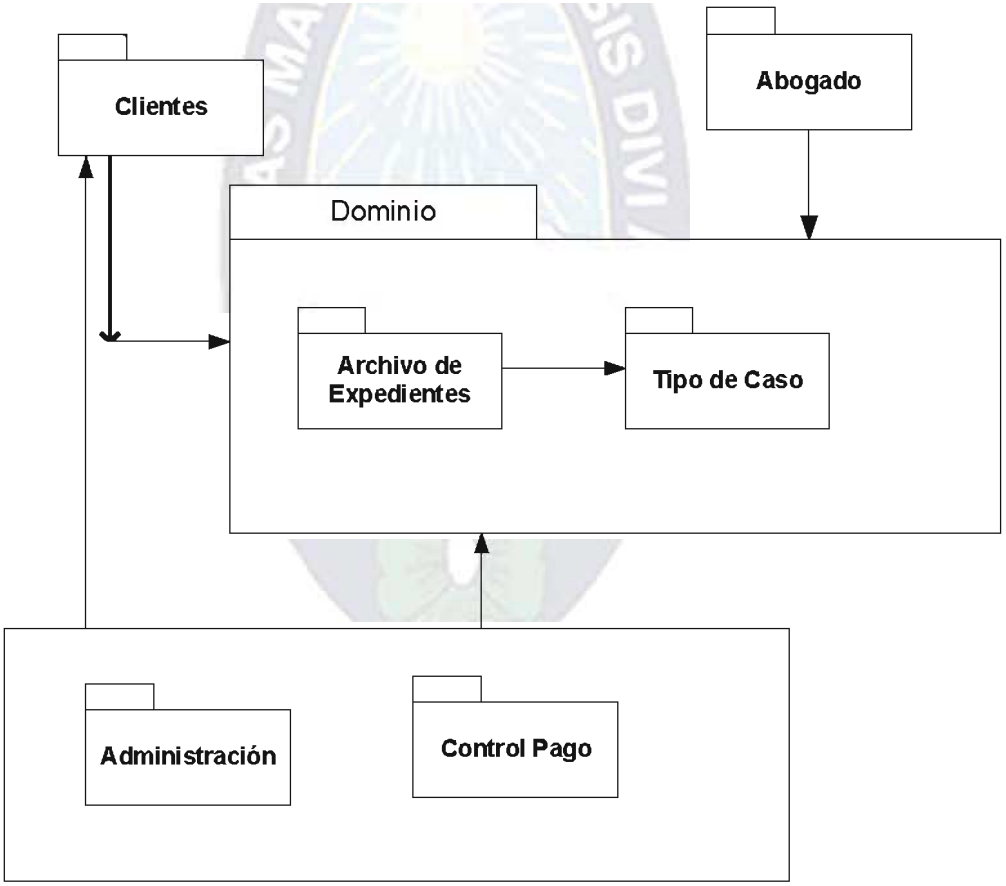


Figura 3.13: Diagrama de Paquetes: Sistema Estudio Jurídico Belzu & Ortuño
Fuente: [Elaboración Propia]

En el diagrama se puede observar las dependencias que existen entre cada paquete o subsistema, a continuación se detallan los procesos de cada paquete:

CLIENTES.

- Registro de Clientes.
- Altas, Bajas y Modificaciones de un Cliente en particular.

En la figura 3.14 se puede observar el procedimiento de registro de clientes, ABM de clientes el mismo que permite adicionar un registro con todos los datos de un cliente, modificarlos o eliminarlos.

```
public void BMCliente(clienteEntidad cliente, Integer caso) {
    jdbc manager = new jdbc(jndiName.BUFETE.getNombreJndi());
     String query;
    try {
        manager.conectar();
        query = "{call BM_Clientes(?,?,?,?,?,?,?,?,?,?,?" +
              ",?,?,?,?,?)}";
        manager.createCallableStatement(query);
        manager.setParameterInCallableStatement(1, cliente.getIdCliente());
        manager.setParameterInCallableStatement(2, cliente.getNombre());
        manager.setParameterInCallableStatement(3, cliente.getApaterno());
        manager.setParameterInCallableStatement(4, cliente.getAmaterno());
        manager.setParameterInCallableStatement(5, cliente.getDireccion());
        manager.setParameterInCallableStatement(6, cliente.getTelefono());
        manager.setParameterInCallableStatement(7, cliente.getCelular());
        manager.setParameterInCallableStatement(8, cliente.getEmail());
        manager.setParameterInCallableStatement(9, cliente.getContacto());
        manager.setParameterInCallableStatement(10, cliente.getDocidentidad());
        manager.setParameterInCallableStatement(11, cliente.getExpedido());
   manager.setParameterInCallableStatement(12, cliente.getProfesion());
        manager.setParameterInCallableStatement(13, cliente.getFechareg(), Types.DATE);
        manager.setParameterInCallableStatement(14, cliente.getCodigocli(), Types.CHAR);
        manager.setParameterInCallableStatement(15, new Integer(caso));
        manager.executeCallableStatementUpdate();
        manager.commit();
    } catch (Exception e) {
        System.out.println("Error al modificar el cliente...");
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
    } finally {
       manager.close();

    }   }
```

Figura 3.14 Procedimiento ABMCliente

ABOGADO.

- Registro de Abogados.

- Altas, Bajas y Modificaciones de un Abogado en particular.

CASOS.

- Registro de Casos.
- Apertura de un Caso.
- Altas, Bajas y Modificaciones de un Caso en particular.
- Asignación a un Servicio y Área
- Consulta de seguimiento de un Caso en particular por un Cliente.

En la figura 3.15 se puede observar el procedimiento de registro de casos, en el que el abogado administrador tiene acceso registra los datos que el cliente le proporcione o describa, así mismo puede realizar el ABM.

```
public void abmCasos(Integer caso, Casos casos) {
    jdbc manager = new jdbc(jndiName BUFETE getNombreJndi()),
    try {
        manager conectar(),
        String query = "{call sp_abm_casos(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)}",
        manager createCallableStatement(query),
        manager setParameterInCallableStatement(1, caso, Types INTEGER),
        manager setParameterInCallableStatement(2, casos getIdCaso(), Types INTEGER),
        manager setParameterInCallableStatement(3, casos getNroCaso(), Types CHAR),
      manager setParameterInCallableStatement(4, casos getIdCliente(), Types INTEGER),
        manager setParameterInCallableStatement(5, casos getIdServicio(), Types INTEGER),
        manager setParameterInCallableStatement(6, casos getIdArea(), Types INTEGER),
        manager setParameterInCallableStatement(7, casos getIdAbogado(), Types INTEGER),
        manager setParameterInCallableStatement(8, casos getReferencia()),
        manager setParameterInCallableStatement(9, casos getFechaInicio(), Types DATE),
        manager setParameterInCallableStatement(10, casos getFechaFin(), Types DATE),
        manager setParameterInCallableStatement(11, casos getMontoPlanificado()),
        manager setParameterInCallableStatement(12, casos getMontoReal()),
        manager setParameterInCallableStatement(13, casos getAntecedentes()),
        int res = manager executeCallableStatementUpdate(),
        manager commit(),
    } catch (Exception e) {
        e printStackTrace(),
    } finally {
        manager close(),
    } }
```

Figura 3.15: Procedimiento de ABMcasos

ADMINISTRACION.

- Registro de Pagos.

- Emisión de recibos y facturas.

APUNTES O ARCHIVO DE EXPEDIENTE.

- Apertura de Apunte.
- Elaboración de Apunte.
- Registro de recepción de documentos.
- Consultas sobre el seguimiento de un caso en particular.

3.9 DISEÑO NAVEGACIONAL.

La figura 3.4.1 ilustra la construcción del nodo a partir de un requerimiento http proveniente de un cliente remoto. Los objetos navegacionales pueden actuar como observadores, para construir vistas de objetos conceptuales como adaptadores, para extender la actividad navegacional de un nodo poder aprovechar el comportamiento conceptual de un objeto. Estas dos perspectivas se implementan en la tecnología JSF para observar y los Servlets para adaptar. La páginas JSF son las encargadas de construir los nodos de la capa navegacional, esto se logra instanciando los objetos del diseño conceptual necesarios para mostrar la información de nodo y utilizando los datos solicitados a dichas instancias para generar árboles de elementos XML. Con este procedimiento se compone un nodo concentrando información relacionada en un documento XML generado dinámicamente con cada requerimiento, lo que permite personalizar el contenido a partir de infinitas configuraciones, tales como un perfil de usuario, la sobrecarga de requerimientos de la aplicación, la historia registrada de la navegación, políticas de protección de contenidos, seguridad, etc.

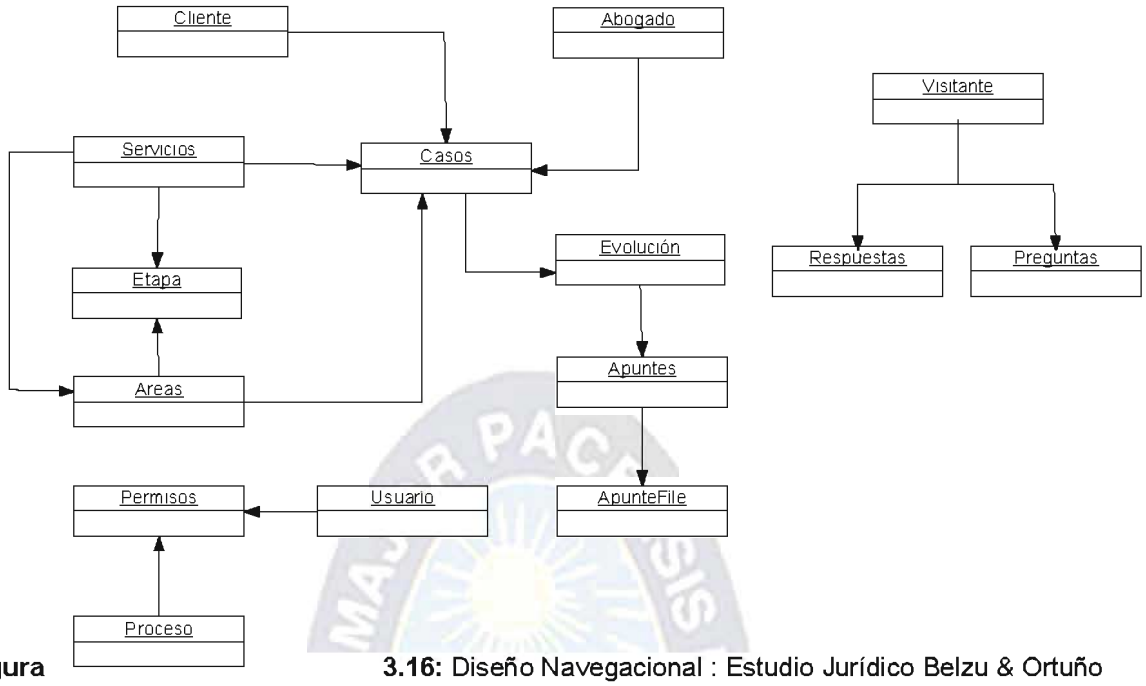


Figura 3.16: Diseño Navegacional : Estudio Jurídico Belzu & Ortuño
Fuente: [Elaboración Propia]

3.9.1 REGLAS DE NAVEGACION ESTUDIO JURÍDICO BELZU & ORTUÑO

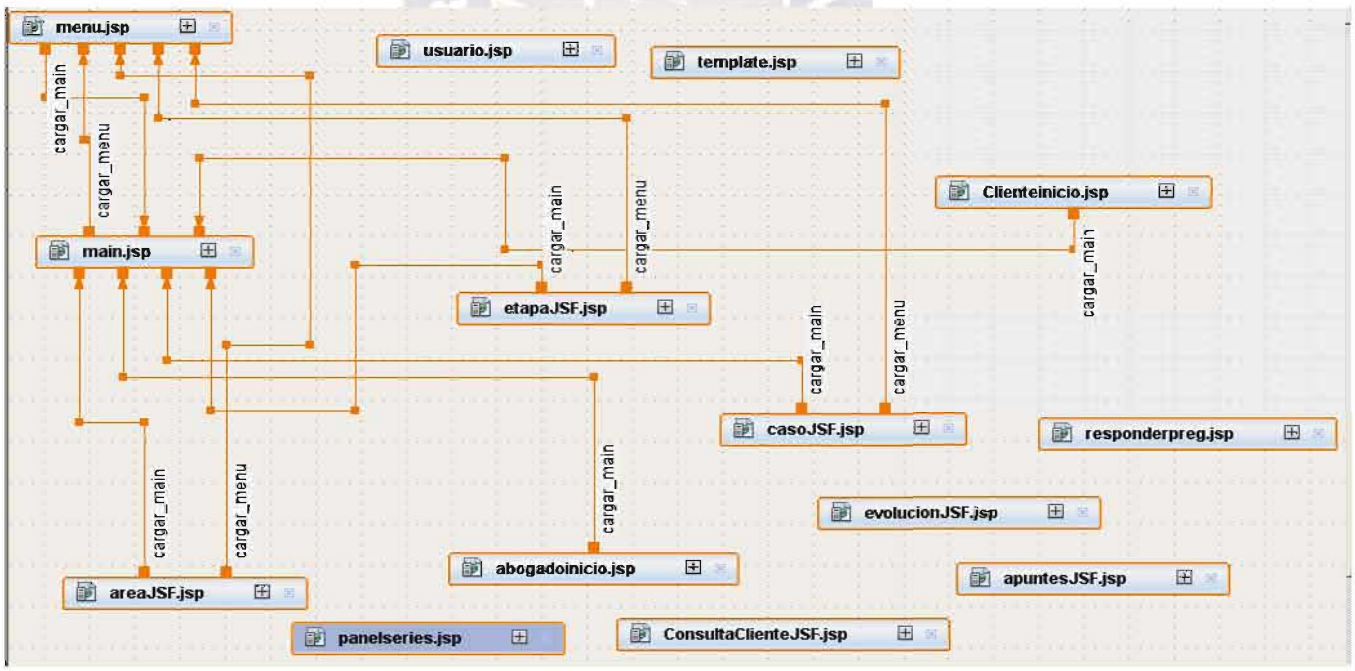


Figura 3.17: Diseño Navegacional: Estudio Jurídico Belzu & Ortuño
Fuente: [Elaboración Propia]

3.10 CALIDAD DEL SOFTWARE.

Para evaluar la calidad del software se utilizó métricas para medir los factores que definen la ISO 9126: funcionalidad, confiabilidad, usabilidad, eficiencia y mantenibilidad, también se utilizó las métricas que sugiere Roger Pressman.

- **FUNCIONALIDAD.**

En la tabla 3.1 se observa la suma del espacio o peso que ocupa cada archivo funcional del sistema.

Tabla 3.1 Tipos de Archivos que componen una aplicación web

Tipo de archivo	Extensiones de archivo	Descripción	Peso en kb
Estrtucturales	jspx, fiase	En esta categoría se incluyen los archivos que describen lógicamente las interfaces de los objetos que componen una aplicación web.	10 kb
Funcionales(servidor)	jsp, bean.java	Estos son los archivos que implementan las funcionalidades de la aplicación web desde el lado del servidor.	596 kb
Funcionales(cliente)	css,ajax,xml	Son archivos que implementan funcionalidades o apariencia de la aplicación web desde el lado del cliente.	127 bk
Funcionales (Incrustados)	class,war	Estos son archivos de funcionalidad externa que se visualizan mediante una funcionalidad agregada al navegador.	1093 kb
Imágenes	jpg, gif, png,jpeg	Son archivos binarios de imágenes en varios formatos.	896 kb
Documentos	pdf, xls,rtf	Este tipo de archivos, comprende los documentos que pueden ser descargados o visualizados de la aplicación web por el cliente.	0 kb

El índice de funcionalidad se calcula como:

$$IFtipo = (\sum PAFtipo) / (\sum PAE + \sum PAFtipo)$$

Donde: IFtipo : índice funcional según el tipo de archivo funcional (servidor, cliente o incrustado)

PAFtipo: peso del tipo de archivo funcional.

PAE: peso de los archivos estructurales.

Entonces se procede al cálculo (con los datos obtenidos de la tabla 2.3 que son la suma del peso en kb de los archivos del sistema):

IFservidor = (∑ PAFservidor) / (∑ PAE + ∑ PAFservidor) = (596) / (10 + 596) = 0.9770

IFcliente = (∑ PAFcliente) / (∑ PAE + ∑ PAFcliente) = 127 / (10 + 127) = 0.927

IFincrustados = (∑ PAFincrustados) / (∑PAE + ∑ PAFincrustados)

IFincrustados = 1093 / (10 + 1093) = 0.99

Para clasificar una aplicación web en funcional o no, se debe calcular el **Índice de Funcionalidad Neto (IFN)** como una combinación lineal de los valores anteriores:

$$IFN = \alpha IFservidor + \beta IFcliente + \Omega IFincrustados$$

Con $\alpha + \beta + \Omega = 1$

Para los valores de α, β, Ω se realizó un promedio de todos los archivos servidor, cliente e incrustados dando como resultado $\alpha = 0.5$, $\beta = 0.3$ y $\Omega = 0.2$

Entonces IFN = (0.5)(0.9770) + (0.3)(0.927) + (0.2)(0.99) =

IFN = 0.4885 + 0.2781 + 0.198 = 0.9646

Por tanto el **IFN = 0.9646**

jacarowen@hotmail.com

Según los criterios siguientes:

Si IFN < 0.2, entonces hablamos de un sitio web no funcional.

Si 0.2 ≤ IFN < 0.5, entonces hablamos de un sitio web funcional.

Si 0.5 ≤ IFN, entonces hablamos de aplicación web.

*Como **IFN = 0.9646** entonces se puede decir que la aplicación es funcional.*

- **CONFIABILIDAD**

Para medir la confiabilidad se utilizo la siguiente ecuación medido en minutos:

$$TMEF = TMPF + TMPR$$

Donde:

TMEF = tiempo medio entre fallas

TMPF = tiempo medio para la falla

La falla puede darse un aproximado de cada 4 horas = 240 minutos

TMPR = tiempo medio para la reparación

En caso de que la falla sea fácil de reparar entonces: TMPR = 5 a 10 minutos

En caso de que la falla sea grave o difícil de reparar entonces: TMPR = 30 minutos.

Reemplazando las variables por los valores correspondientes se tiene:

$$TMEF = 240 + 30 = 270 \text{ min.}$$

Entonces el **TMEF = 270 min.**

De este cálculo se puede obtener la disponibilidad del software mediante la siguiente fórmula:

Disponibilidad = ((TMPF) / (TMPF + TMPR)) * 100

Disponibilidad = (240/ (240 + 30)) = 0.8888 * 100

luego se tiene **Disponibilidad = 88,88%**

- **USABILIDAD**

Para la usabilidad se utilizó datos de cuatro usuarios en tres tareas el tiempo medido en minutos: En la tabla 3.2 se muestra el resumen de los datos obtenidos utilizando la métrica SUM (Single Usability Metric) .

Tabla 3.2 Datos colectados para la métrica SUM

Tarea 1: Registrar cliente				
Usuario	**Satisfacción**	**Conclusión**	**Errores**	**Tiempo**
1	5	1	1	5
2	4	1	0	7
3	5	1	1	8
4	4	1	1	10
Tarea 2: Registrar Caso				
Usuario	**Satisfacción**	**Conclusión**	**Errores**	**Tiempo**
1	4,33	1	1	15
2	3,99	1	3	12
3	4	1	2	14
4	5	1	1	10
Tarea 3: Consultar Caso				
Usuario	**Satisfacción**	**Conclusión**	**Errores**	**Tiempo**
1	4,33	1	2	3
2	3,7	1	10	4
3	3,9	1	1	5
4	5	1	3	3

Tabla 3.3 Resultados para la evaluación de la usabilidad

SUM							
Tarea	**Bajo**		**Alto**	**Conclusión**	**Satisfacción**	**Tiempo**	**Errores**
1	65,80%	86,60%	95,60%	100%	86,30%	100%	60%
2	60,70%	82,70%	93,60%	100%	70,70%	91,30%	68,60%
3	63,40%	83,70%	93,00%	100%	64,30%	99%	71,40%
	63,30%	84,30%	94,10%				

Dado que:

Excelente 90 – 100% ◆ Muy bien 80 – 89% ◣ Bien 70 – 79% ▲

Pasable 60 – 69% ● Pobre < 60% ○

Se puede concluir que la aplicación web tiene una apreciación de usabilidad buena.

- **MANTENIBILIDAD.**

Para el cálculo de la mantenibilidad se utilizará las siguientes métricas:

t_{cola} = Tiempo transcurrido(hrs.) desde el momento en el que se hace una petición hasta que la evaluación está completa.

W_{eval} = Esfuerzo (hrs. - persona) para realizar la evaluación.

W_{cambio} = Esfuerzo (hrs. - persona) para realizar el cambio.

t_{cambio} = Tiempo requerido (hrs.) para hacer el cambio.

E_{cambio} = Errores descubiertos durante el trabajo para hacer el cambio.

D_{cambio} = Defectos descubiertos después de liberar el cambio al cliente.

Los datos se tomaron para cuatro peticiones de cambio en donde el cambio se evaluó en dos casos:

En el caso de que sea fácil de cambiar la hora minima de cambio es **1 hora.**

En el caso de que sea difícil el cambio entonces las horas a requerir son de **8 horas como** máximo para cambiar algo en el sistema según requerimiento. En caso de que en esas 8 hrs. requeridas no se pueda hacer el cambio entonces el problema se subdividir el problema y analizar más a fondo.

Tabla 3.4 Datos para la medición de la mantenibilidad

	t_{cola}	W_{eval}	W_{cambio}	t_{cambio}	E_{cambio}	D_{cambio}
cambio 1	1	1,5	1	1	2	1
cambio 2	2,15	3	2,5	2,5	3	2
cambio 3	3	4,5	3,5	3,15	5	1
cambio 4	2	3,15	8	8	4	1

$Tiempo_{total} = (\sum^{n}_{i=1} (t_{cambio} + t_{cola})) / n = (14.65 + 8.15) / n = 22.8 / 4 = 5.7$ hrs.

$Esfuerzo_{total} = (\sum^{n}_{i=1} (W_{eval} + W_{cambio})) / n = 27.15 / 4 = 6.78$ (hrs. - persona)

$ERD = E_{cambio} / (E_{cambio} + D_{cambio}) = 3.5 / (3.5 + 1.25) = 0.77$ es la eficiencia de remoción de defectos.

Entonces: **ERD = 0.77 es la eficiencia de remoción de defectos**

Por los resultados obtenidos se puede observar que la ERD tiende a 1 en un 77% por lo cual se puede concluir que el software es mantenible.

- **PORTABILIDAD**

Para medir la portabilidad se siguió los siguientes criterios:

1º. El primer problema de portabilidad es la elección del lenguaje en el que será implementada la aplicación web. Problemas de portabilidad tienden a aparecer no en el núcleo del lenguaje sino en las librerías de soporte en el grado de integración con el ambiente local. Para este proyecto se eligió el lenguaje de programación JAVA.

2º. La portabilidad de la base de datos, las consultas escritas es otro problema de portabilidad. Para este proyecto se eligió PostgreSQL, además que se esta utilizando JSF (Java Server Faces) como Framework para la persistencia.

Tabla 3.5 Porcentajes de Portabilidad

Características	Porcentaje evaluado
La portabilidad de java es excelente, después de todo fue diseñada con el objetivo de "escribe una vez, se ejecuta en cualquier lugar" Sin embargo la portabilidad falla para ser perfecta, las dificultades que existen son mayormente en las diferencias de versiones entre JDK 1.3 y los GUI toolkit.	70%
Las aplicaciones en JSF son fáciles de migrar de una base de datos a otra sin hacer mayores cambios en el código más que un pequeño cambio en el archivo XML de configuración (faces-config.xml) que contiene los detalles de los accesos a la base de datos.	90%
Las consultas están escritas en SQL que se traduce al dialecto de la base de datos elegida por lo tanto no es necesario hacer cambios en estas consultas.	90%

Por lo tanto el porcentaje promedio de portabilidad es de **83.33%**, lo cual nos dice que la aplicación es altamente portable a otros ambientes de ejecución.

El grado de portabilidad puede ser calculado mediante la formula:

DP = 1 – (costo de transportar / costo de redesarrollar)

TT = TCC + TCW + TCL = horas de transportar la aplicación al nuevo Framework

En caso de querer cambiar al Framework Spring que es más difícil por que en este caso el modelo y el controlador son los que más cambian por tanto se tiene un valor aproximado:

TCC = número de horas de transportar la capa controlador = 70 hrs.

TCW = número de horas de transportar la capa web = 10hrs.

TCL = número de horas de transportar la lógica del negocio = 90 hrs.

TT = 70 + 10 + 90 = 170 hrs.

TT = TDCC + TDCW + TDCL = horas de desarrollar la aplicación

TDCC = número de horas de redesarrollar la capa controlador = 500 hrs.

TDCW = número de horas de redesarrollar la capa web = 350hrs.

TDCL = número de horas de redesarrollar la lógica del negocio = 350 hrs.

Luego se tiene: TT = 500 + 350 + 350 = 1200 hrs.

Entonces: DP = 1 – (170/ 1200) = 1 – 0.141666

DP = 0.858 grado de portabilidad

Por tanto, como DP es mayor que 0 se concluye que es mucho más económico transportar toda la aplicación que desarrollarla de nuevo.

jacarowen@hotmail.com

3.11 DISEÑO DE SEGURIDAD

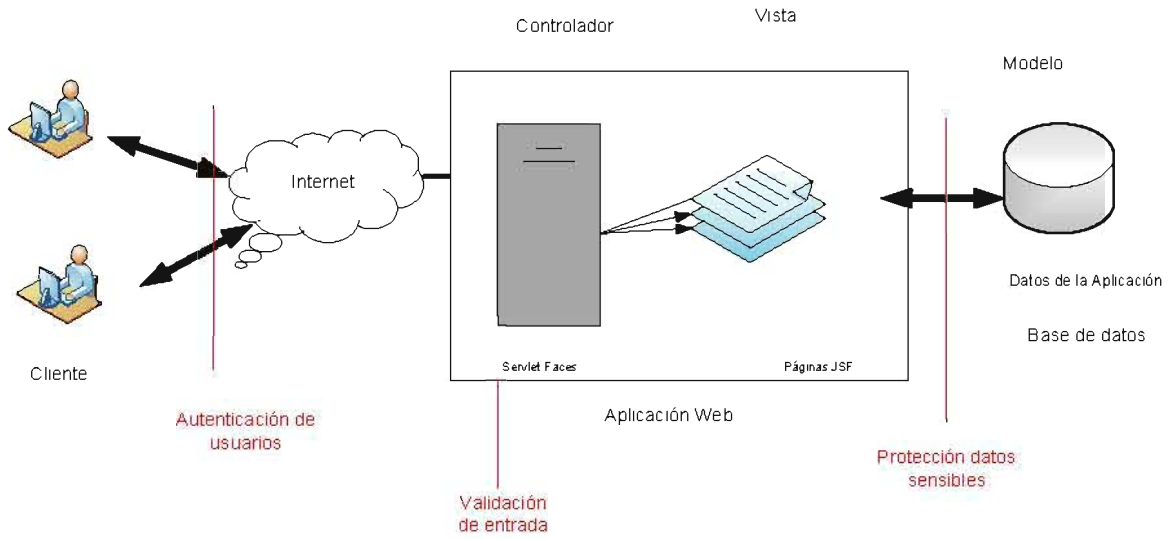


Figura 3.18: Diseño Seguridad: Estudio Jurídico Belzu & Ortuño
Fuente: [Elaboración Propia]

La aplicación debe permitir identificar a un usuario usando algún mecanismo de autenticación. Dado que todas las subsecuentes decisiones de autorización estarán basadas en la identidad del usuario. Lo que se debe considerar también se presenta en el paso de datos de entrada y salida a través de redes públicas. Prevenir el acceso a datos sensibles es un aspecto también de prioridad.

En el sistema se aplicará políticas de seguridad que determinen que es lo que está permitido hacer a la aplicación y usuario, esto define las restricciones que determinan que es lo que no puede hacer una aplicación o usuario.

3.11.1 **SEGURIDAD EN LA BASE DE DATOS.**

Es la protección contra:

- Revelación no autorizada (*confidencial*)
- Alteración no autorizada (*integridad*)
- Destrucción intencional o involuntaria

La protección está dirigida a dos tipos de usuarios

- Los que no tienen derecho a acceso
- Los que tienen derechos de accesos limitados a ciertas acciones

Seguridad ≡ ¿Qué datos? + ¿Qué operaciones? + ¿Cuáles usuarios?

Ya que los datos son el activo más valioso de una empresa u organización

3.11.1.1 OBJETIVOS DE SEGURIDAD.

- **INTEGRIDAD.** Sólo los usuarios deberían tener acceso para modificar datos.
- **DISPONIBILIDAD.** Los datos deben estar disponibles para usuarios y programas de actualización autorizados.
- **CONFIDENCIALIDAD.** Protección de los datos de su revelación no autorizada.

En la tabla 3.6 se observa lo que se aplicará para la seguridad de la base de datos el seguimiento de rastros, se refiere a si algún usuario ingresa a la BD lo que quedará registrado en una tabla log a que datos ingreso y que hizo con los mismo esto se logrará con el uso de triggers o disparadores.

Tabla 3.6 Procedimiento log_abm_caso triggers

```
Creación de log abm_casos()
CREATE OR REPLACE FUNCTION abm_casos()
  RETURNS trigger AS
$BODY$
BEGIN
    IF (TG_OP='INSERT') THEN
          INSERT INTO log_casos (logid,idCasos,logmomento,logip,logtipope,ctauid)
          VALUES(default,NEW.idCasos,now(),TG_IP,TG_OP,10),
        ELSEIF( TG_OP='UPDATE' OR TG_OP='DELETE') THEN
          INSERT INTO log_casos logid,idCasos,logmomento,logip,logtipope,ctauid)
          VALUES(default,OLD.idCasos,now(),TG_IP,TG_OP,10),
    END IF,
RETURN NULL,
END,
```

```
  $BODY$
    LANGUAGE 'plpgsql' VOLATILE
    COST 100,

creación de triggers para log abm_casos()

CREATE TRIGGER function_trigger
  AFTER INSERT OR UPDATE OR DELETE
  ON "Casos"
  FOR EACH ROW
  EXECUTE PROCEDURE abm_casos(),
```

AUTORIZACIÓN BASADA EN ROLES.

Los privilegios son asignados realmente a identificadores que pueden representar a un usuario aislado o a un grupo de ellos. Para esto los privilegios son asignados a roles, como se observa en la tabla 3.7.

Tabla 3.7 Privilegios asignados a roles

```
CREATE ROLE <nombre_rol> [WHIT ADMIN <quien>]
GRANT <rol_concedido>[{<como><rol_concedido>}……]
TO <a-quien>[{<como><a-quien>}……..]
[WHIT ADMIN OPTION] [GRANTED BY <a-quien>]
```

En el mundo real, los privilegios de acceso están asociados con el rol de una persona en la organización, cada rol debe ser creado con determinados privilegios y cada usuario es asociado a un rol.

3.12 INTERFAZ CON EL USUARIO.

El sistema tiene la capacidad de ser ejecutado desde un equipo, administrado a partir de una red local o administrada en la Web, de esta manera el sistema será aprovechado en su explotación.

3.12.1 CARACTERÍSTICAS DEL NUEVO SISTEMA.

El sistema trabaja bajo un entorno multiusuario. Cada uno de los usuarios tendrá la posibilidad de poder acceder al sistema. El cliente tendrá la posibilidad de consultar el estado de sus casos, de esta forma podrá realizar un seguimiento del mismo previa identificación en la aplicación, de esta manera mejorar la comunicación entre el Estudio Jurídico Belzu & Ortuño y sus Clientes. Se podrá restringir el acceso a información confidencial para cualquier usuario, según el nivel de acceso o permisos del usuario.

Se puede registrar a los abogados que trabajan en el Estudio Jurídico Belzu & Ortuño con sus respectivos permisos de acceso al sistema.

Así como también facilitarle al mismo abogado, gestionar su despacho de abogados, obteniendo la información de manera rápida y sencilla además de conocer el estado actual o pasado de cualquier caso o cliente. De esta manera modernizar y reducir el tiempo en el acceso de información.

Cualquier usuario podrá realizar Preguntas al Estudio Jurídico Belzu & Ortuño, obteniendo respuestas a su debido tiempo, Así mismo, el Cliente podrá realizar observaciones o sugerencias cuando consulte su Caso.

Las características más destacables son:

- Es un navegador de entorno sencillo y conocido por todos los usuarios.
- No requiere introducir gran cantidad de datos.
- Posibilidad de consulta de los clientes sin necesidad de instalación, a cualquier hora y lugar que se encuentre.

- Cada reunión, conversación, documento respecto a un caso quedan registrados en un apunte nuevo con la fecha y abogado que lo atiende, de esta forma los abogados y administrador saben en que estado se encuentra cada caso en particular.

3.12.2 **INGRESO AL SISTEMA.**

Figura 3.19: Pantalla Principal, Ingreso al Sistema

En la figura 3.19 se observa la pantalla principal que nos permite navegar sin ingresar ningún password para ver lo que es y ofrece el Estudio Jurídico Belzu & Ortuño.

En la parte superior de la página el cliente tiene acceso al sistema previa identificación en la aplicación. De esta manera facilitarle la posibilidad de consultar sobre el estado

de su caso obteniendo una ventaja competitiva respecto a la información y comunicación entre el despacho y sus clientes.

Se pueden definir distintos usuarios, cada uno con distintos niveles de acceso a la información y registro del sistema.

3.12.3 **OPCIONES DEL SISTEMA.**

En la figura 3.20, se presenta la pantalla principal de acuerdo al permiso que tenga el cliente o usuario para registrar o consultar las opciones del sistema.

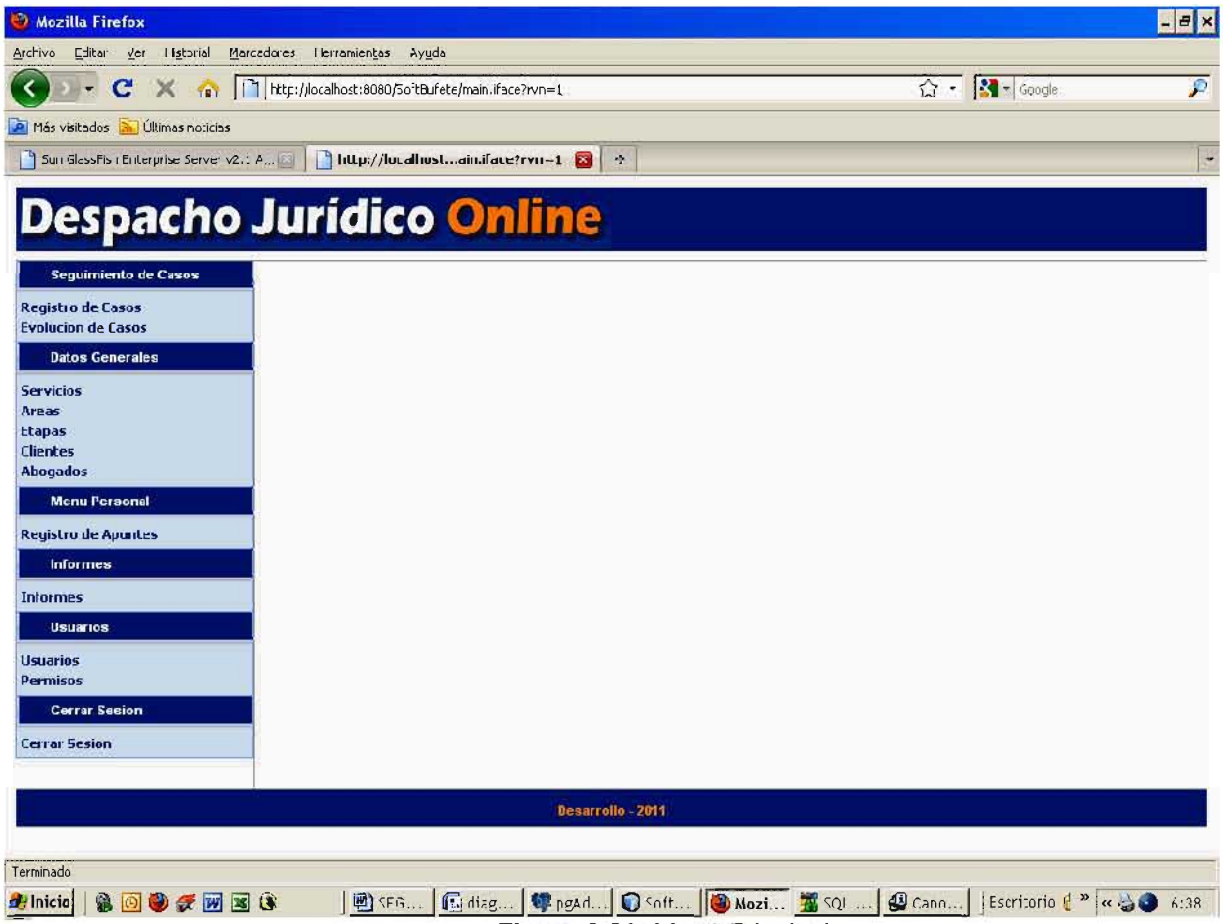


Figura 3.20: Menú Principal

3.12.4 CLIENTE

Figura 3.21: Registro de Clientes

En la figura 3.21, se observar la captura de datos para registrar a un cliente, al registrarse el mismo en el momento de grabar el sistema le asigna un código de cliente que es formado por las iniciales de su apellido paterno, materno y nombre más el numero de cedula de identidad, esto se observa en la figura 3.20

Figura 3.22: Asignación código cliente

3.12.5 **REGISTRO DE ABOGADOS.**

En la figura 3.23 se observar la captura de los datos de un abogado que trabajara en el Estudio Jurídico Belzu & Ortuño, el mismo que podrá representar a un cliente. El administrador puede realizar el ABM (Altas, Bajas y Modificaciones) del mismo según el nivel de acceso que tenga al sistema.

Figura 3.23: Registro de Abogados

3.12.6 **REGISTRO DE CASOS**

En la figura 3.24 vemos el registro de un caso, en el mismo se jalan los datos del cliente que ya esta registrado, se le asigna un servicio que corresponda al caso, también se le asigna un área, una referencia y se adjunta o registra las pruebas o documentos que el cliente proporcione al abogado. El mismo cuenta con el ABM respectivo. También se puede realizar consultas del mismo solo por el abogado.

Figura 3.24: Registro de Casos

3.12.7 **EVOLUCION DE CASOS.-**

En la figura 3.25 se nos permite observar la evolución de los casos, las etapas que paso y en la que actualmente se encuentra, se puede realizar consultas, ABM del mismo.

Figura 3.25: Evolución de Casos

3.12.8 **REGISTRO DE SERVICIOS.**

Nos permite realizar el ABM de los servicios que ofrece el Estudio Jurídico Belzu & Ortuño.

Figura 3.26: Registro de Servicios

3.12.9 **REGISTRO DE AREAS.**

Nos permite realizar el ABM de un área respecto al servicio que corresponda

Figura 3.27: Registro de Áreas

3.12.10 REGISTRO DE ETAPAS.

En la figura 3.28, podemos ver el registro de una etapa según el servicio y área que corresponda para relacionar el caso correspondiente.

Figura 3.28: Registro de Etapas

3.12.11 CONSULTA CLIENTE (SEGUIMIENTO DE CASOS)

En la figura 3.29 nos permite observar el ingreso de un cliente para consultar sobre su caso previa identificación para su aplicación.

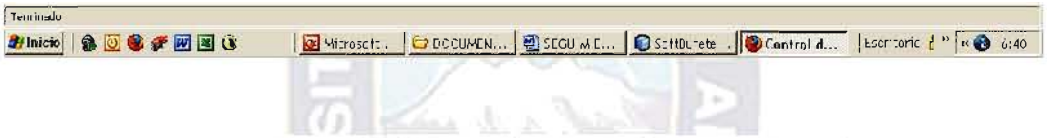


Figura 3.29: Consulta Cliente: Seguimiento de Caso

En la figura 3.30 se puede observar los datos de consulta a los que tiene acceso el cliente y estos son todos los datos de su caso: se puede observar los datos del cliente, el servicio, el área al que corresponde, el o los abogados que atienden su caso, la etapa en la que se encuentra, la táctica a seguir, los documentos que tiene registrados, etc.

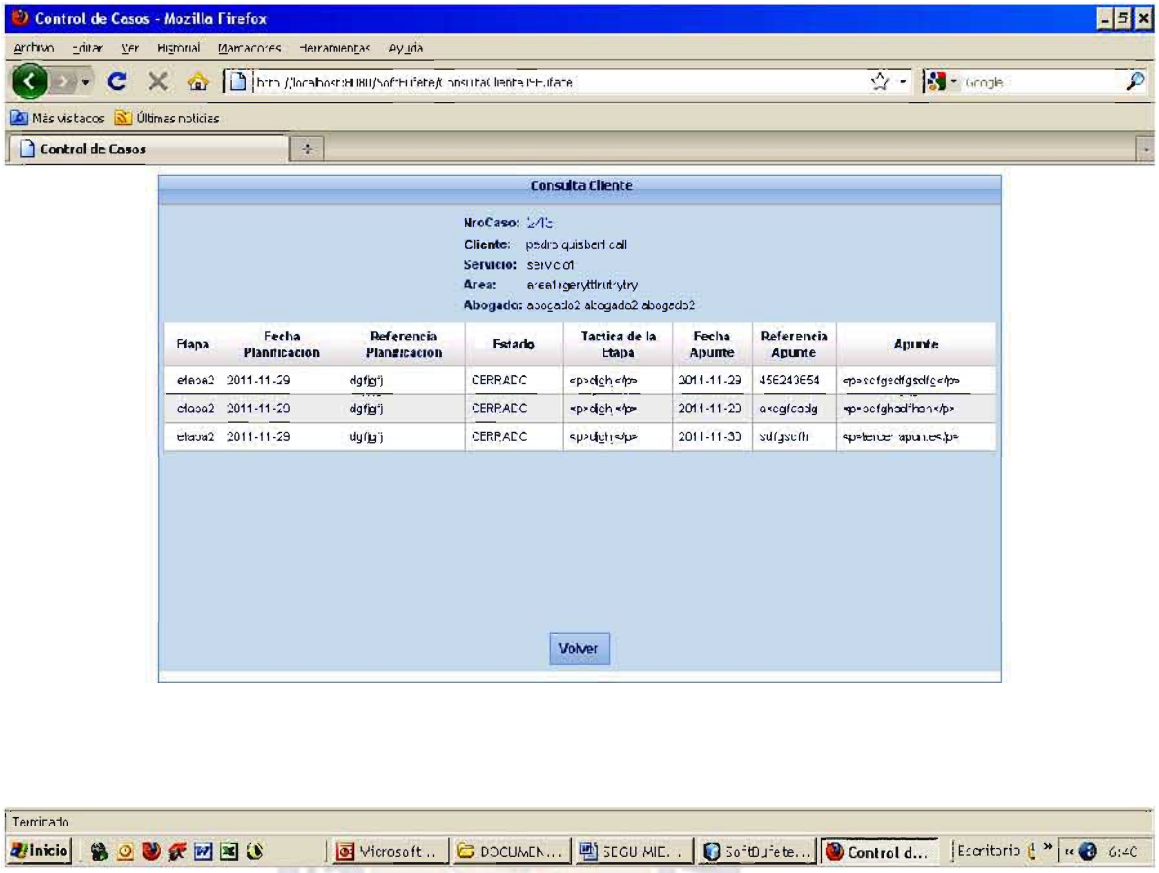


Figura 3.30: Consulta Cliente: Seguimiento de Caso

CAPITULO IV
CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

El presente trabajo logró el cumplimiento de sus objetivos satisfactoriamente gracias a las herramientas y al proceso utilizado. Es importante utilizar un proceso de desarrollo, planificar un proyecto y evaluarlo en todo el ciclo de vida de un proyecto.

El trabajo logró los objetivos específicos los mismos que se pueden apreciar en el diseño del sistema en las figuras 3.14 y 3.15 que detallan los procesos del sistema.

Así mismo, se concluye que el presente trabajo logró sus objetivos respecto a la calidad con el patrón ISO 9126, ya que midiendo cada uno de los factores dieron los siguientes resultados que se muestran en la tabla 3.6.

Tabla 4.1 Resultados de Calidad

Funcionalidad IFN	Confiabilidad	Mantenibilidad ERD	Portabilidad DP
0.9646	88.88%	0.77	0.858

Midiendo el índice de funcionalidad neto se observó que ***IFN = 0.9646*** y según uno de los criterios utilizados nos dice que: si ***0.5 ≤ IFN se habla de una aplicación web,*** por tanto se puede concluir que la aplicación es funcional. Respecto a la confiabilidad se tiene el siguiente resultado: la aplicación tiene un **88.88% de disponibilidad**. Midiendo la usabilidad se observó que la aplicación tiene una apreciación buena de usabilidad. Con respecto a la mantenibilidad se observó que la ***eficiencia de remoción de defectos ERD = 0.77*** y como tiende a 1 se puede deducir que la aplicación es mantenible. Finalmente midiendo la portabilidad se observó que el ***grado de portabilidad DP = 0.858***, como el resultado es mayor a cero se concluye que es más económico transportar toda la aplicación que desarrollarla de nuevo. De esta manera se logró la satisfacción de los usuarios, así mismo de los clientes.

También se debe tomar en cuenta el uso de las herramientas de uso libre que se eligieron, a excepción de java y JSF que están bien documentados, se encontró problemas en las herramientas como glasfish ya que les falta documentación de algunos problemas que se pudieron detectar.

El sistema le ofrece al cliente la posibilidad de consultar el estado de sus casos, previa identificación en la aplicación, obteniendo así una ventaja competitiva respecto a la información y comunicación entre el despacho y sus clientes, ganando tiempo al saber las dos partes en que situación se encuentra el caso antes de una reunión, o pudiendo consultar el estado a cualquier hora del día y en cualquier lugar.

Por último se concluye que el presente trabajo puede ser instalado y utilizado en cualquier otra empresa que esté dedicada al rubro, para este propósito se deberá configurar algunos detalles propios para cada empresa.

4.2 RECOMENDACIONES.

En el proyecto Sistema de Gestión Jurídico se enfatiza en el seguimiento de casos, se recomienda en ampliar el sistema con un módulo de contabilidad que permita obtener resultados de balance general, cierre de gestión, presupuesto estimado al inicio del caso y el presupuesto pactado del caso etc. de está manera ayudar en las tareas del Estudio Jurídico obteniendo además la rentabilidad de cada caso, como por ejemplo el volumen de las horas presupuestadas, el costo global del caso, el beneficio actual del mismo, etc. Relacionando al volumen de horas presupuestadas también el sistema podría de alguna manera automática mostrar los casos deficitarios o cuales están próximos a agotar las horas que se presupuestaron, de esta manera el abogado podría tomar las previsiones correspondientes.

A la hora de realizar la comprobación de la calidad de software se emplearon el patrón ISO 9126 y métricas para medir la calidad descritas en el libro de "Ingeniería de Software, un Enfoque Práctico" de Roger Pressman, en este mismo texto sugiere otras opciones para medir la calidad de software, las cuales se recomiendan sean analizados y aplicados en otros proyectos de similares características.

Finalmente se recomienda utilizar otras herramientas de programación orientado a objetos para el desarrollo de un sistema de software para realizar comparaciones que permitan desarrollar una mejor selección de software de aplicación para futuras aplicaciones.

jacarowen@hotmail.com

BIBLIOGRAFIA

[PRESSMAN, 2002] "Ingeniería del Software, Un Enfoque Práctico", Pressman, Roger S; 343 -434, Mc. Graw Hill, España 2002

" UML Gota a Gota", Fowler, Martin; Pearson Educación; México; 1999

[Weitzenfeld,2000] "Ingeniería de Software Orientada a Objetos", Dr. Alfredo Weitzenfeld

[Jacobson, 1992] Metodología Objetory OOSE/ Objetory, Jacobson, 1992

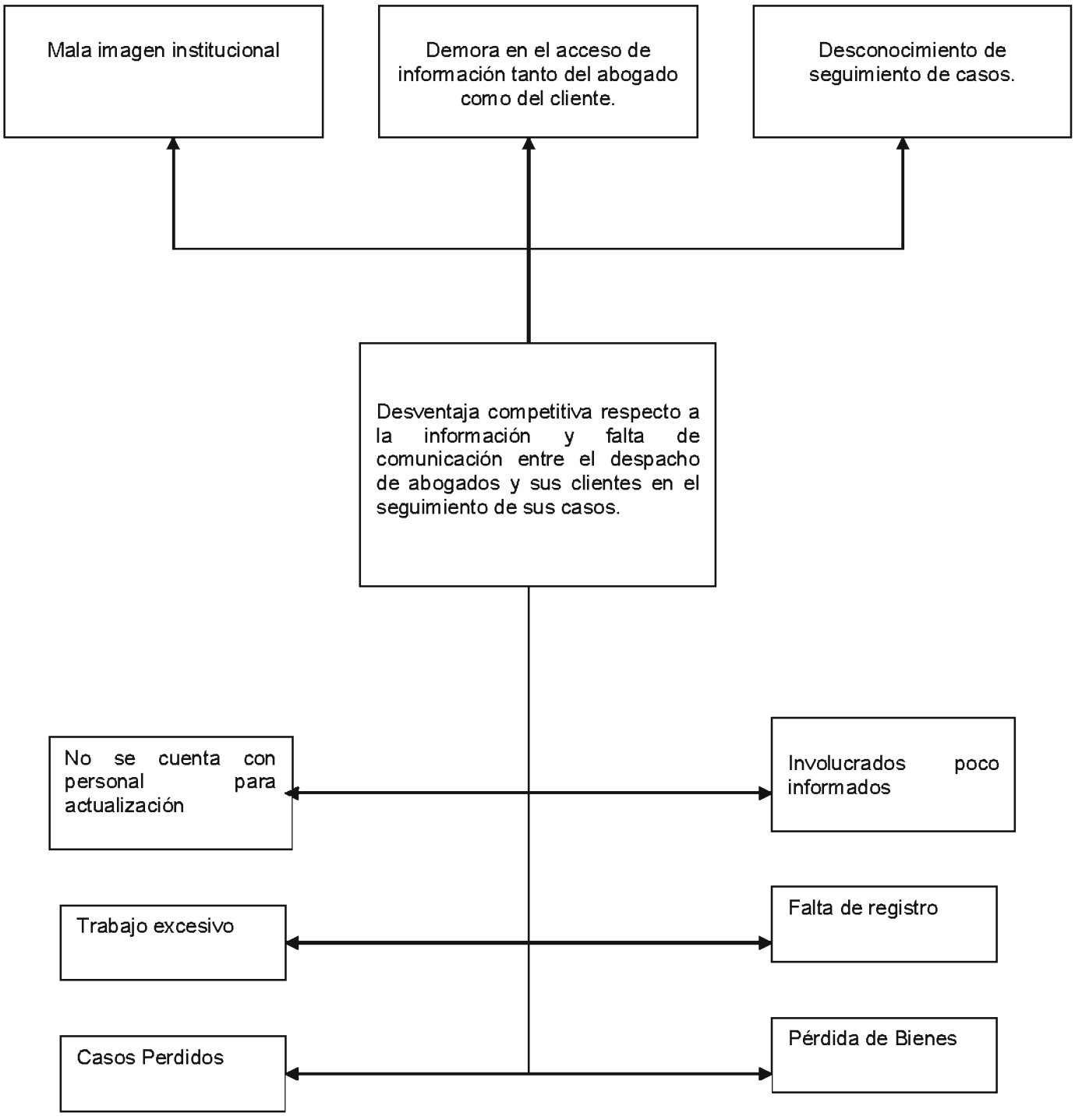
Lenguaje Unificado de Modelado, Disponible en http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado

http://www.monografias.com/trabajos14/control/control.shtm

jacarowen@hotmail.com

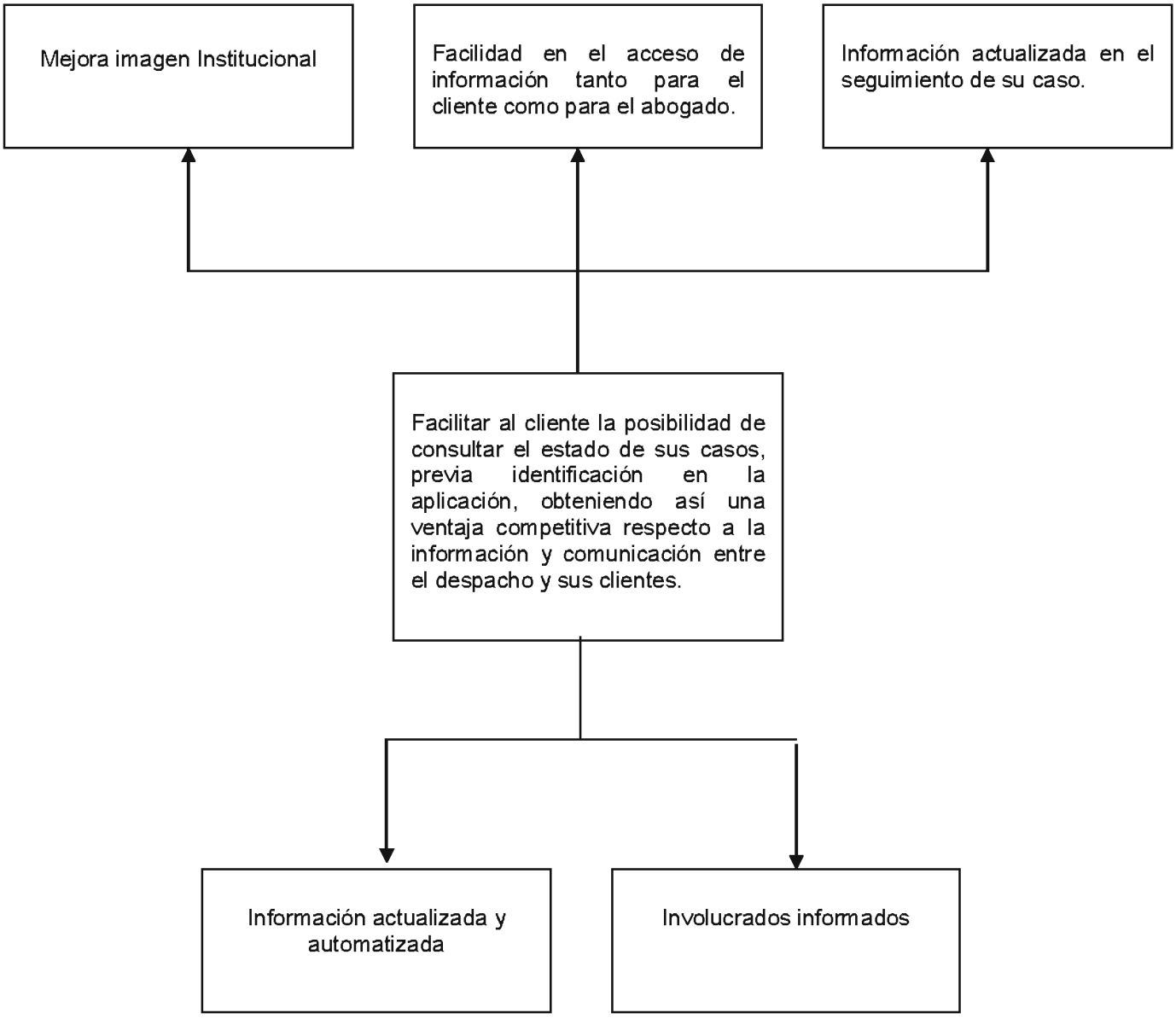
ANEXOS

ARBOL DE PROBLEMAS



jacarowen@hotmail.com

ARBOL DE OBJETIVOS



DOCUMENTACIÓN