

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6



Módulos Ofertas, Clasificación y Selección de Proveedores del subsistema Gestión de Proveedores de la Plataforma de Gestión de Servicios.

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

Autor:

Isael Galindo Corrales

Tutor(es):

Ing. Niurka Martínez Durán

Ing. Yandry Alberto Terry

Junio, 2011.

“Año del 53 aniversario del triunfo de la Revolución”



“Cuando se es 'hombre de ciencia', no se tiene ideal: se elaboran resultados científicos, y cuando se es hombre de partido se combate para ponerlos en práctica. Pero cuando se tiene un ideal, no se puede ser hombre de ciencia, pues se ha adoptado una decisión de antemano.”

Ernesto Guevara de la Serna

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmo la presente a los ____ días del mes de _____ del año 2011.

Isael Galindo Corrales

Firma del Tutor

Ing. Niurka Martínez Durán

Firma del Tutor

Ing. Yandry Alberto Terry

Firma del Tutor

Datos de contacto

Síntesis del Autor: Isael Galindo Corrales

Correo Electrónico: igalindo@estudiantes.uci.cu

Síntesis del Tutor: Ing. Niurka Martínez Durán.

Profesión: Ingeniero en Ciencias Informáticas.

Año de graduado: 2008

Correo Electrónico: nduran@uci.cu.

Síntesis del Tutor: Ing. Yandry Alberto Terry.

Profesión: Ingeniero en Ciencias Informáticas.

Año de graduado: 2007

Correo Electrónico: yalberto@uci.cu

Agradecimientos

A mis padres por haberme formado como la persona que soy y haberme apoyado en los momentos más duros, cuando tuve que tomar decisiones complejas.

A mi hermano por todos los consejos que me ha dado siempre y por ser mi guía en mis estudios.

A todos los que me ayudaron a confeccionar este Trabajo de Tesis, sin ellos hubiera sido más difícil realizarlo, a mis amigos Leo, Pablo, Flavio, Mario, Jorge, Felipe, Yunier, Orelvis, Camilo, a los tutores por brindarme ayuda todo momento.

Al tribunal por su ayuda en la confección del trabajo, a los integrantes de los proyectos que me ayudaron con el desarrollo del trabajo por su incondicional apoyo y las mejores intenciones.

A mis amigos que en estos cinco años siempre han estado conmigo y que me ayudaron a llegar donde estoy.

A toda esa gente que ya no está entre nosotros pero que siempre mostró interés en mi bienestar y me ayudó a seguir adelante.

A mis amigos Andrés, Humberto y Daniel en Camagüey que siempre han estado conmigo en los momentos difíciles y me han ayudado a llegar donde estoy de una forma u otra.

A mis tíos, en especial a aquellos que me apoyaron en mis estudios desde el principio, a Ramón y Herminia.

Agradezco de manera general a todos los que ayudaron a mi formación y que han compartido conmigo en algún momento de sus vidas.

A toda la gente que ha confiado en mí y me ha brindado su mano en algún momento.

Dedicatoria

A quienes han sido ejemplos a seguir siempre por mí, mis padres y mi hermano.

A los que no están conmigo físicamente pero han sido parte de la realización de este sueño.

A todos los que han compartido conmigo momentos importantes desde mis inicios hasta el momento, a todos mis amigos.

En especial a todos los que de una forma u otra me han ayudado en mi vida.

Resumen

En la actualidad debido al auge del comercio y la gran competencia de los involucrados en este, las empresas tienen la necesidad de mejorar la gestión de sus procesos. La empresa ALBET Ingeniería y Sistemas, no cuenta con un sistema de gestión para llevar el control de la información generada en su negocio. La investigación realizó y documentó el proceso de gestión de los proveedores y ofertas de la empresa, centrando la atención en la clasificación y selección de proveedores. Se concluyó que debía desarrollarse una aplicación accesible desde la web. Se definió el Proceso Unificado de Rational (RUP) como metodología de desarrollo; por sus características, su robustez y la gran cantidad de información que genera. Se tomó como lenguaje de programación Groovy y Grails como framework de desarrollo.

Además se usaron otros lenguajes tales como; AJAX, CSS HTML5 y la librería de JavaScript Dojotoolkit para el desarrollo de la interfaz de usuario. Como resultado de la investigación se cuenta con un sistema para la administración de proveedores que puede ser utilizado en empresas similares y brinda las funcionalidades requeridas por su cliente.

Palabras claves:

- ALBET.
- Gestión de proveedores.
- Selección y Clasificación de proveedores.
- Groovy.
- Grails.

Índice

AGRADECIMIENTOS	I
DEDICATORIA.....	II
RESUMEN.....	III
INTRODUCCIÓN.....	1
CAPITULO 1: FUNDAMENTACIÓN TEÓRICA.	6
1.1 APLICACIONES UTILIZADAS PARA LA GESTIÓN DE PROVEEDORES.....	6
1.1.1 SISPRO.....	6
1.1.2 ERP Manager.....	7
1.1.3 @GeSTOCK	8
1.1.4 ¿Por qué no hacer uso de las aplicaciones existentes?	9
1.2 APLICACIONES WEB.....	9
1.2.1 Arquitectura cliente/servidor.....	10
1.3 TECNOLOGÍAS Y HERRAMIENTAS A USAR EN EL DESARROLLO DE LA SOLUCIÓN.	10
1.3.1 LENGUAJES UTILIZADOS.....	10
1.3.1.1 Java.	11
1.3.1.2 JavaScript:	12
1.3.1.3 Lenguaje AJAX.....	12
1.4 FRAMEWORK.	13
1.4.1 Grails.....	13
1.4.2 Dojotoolkit.	14
1.5 METODOLOGÍA DE DESARROLLO DE SOFTWARE.	14
1.5.1 Metodología de desarrollo RUP.	14
1.5.2 UML (Unified Modeling Language).	16
1.6 HERRAMIENTAS CASE.	17
1.6.1 Visual Paradigm.	17
1.6.2 IDE de desarrollo.....	17
1.7 SISTEMA DE GESTIÓN DE BASE DE DATOS POSTGRESQL.....	18
1.8 CONCLUSIONES PARCIALES	18
CAPÍTULO 2 CARACTERÍSTICAS DEL SISTEMA.....	19
2.1 OBJETO DE ESTUDIO.	19
2.2 FLUJO ACTUAL DE LOS PROCESOS.....	19
2.3 ANÁLISIS CRÍTICO DE LA EJECUCIÓN DE PROCESOS.....	19

2.4 OBJETO DE AUTOMATIZACIÓN.....	19
2.5 MODELO DE NEGOCIO.....	20
2.5.1 Actores del negocio.....	20
2.5.2 Trabajadores del negocio.....	20
2.5.3 Diagrama de casos de uso del negocio.....	21
2.5.4 Modelo de objetos.....	23
2.6 REQUISITOS FUNCIONALES.....	23
2.7 REQUISITOS NO FUNCIONALES.....	25
2.8 MODELADO DEL SISTEMA.....	26
2.8.1 Diagrama de Casos de uso del sistema.....	27
2.9 CONCLUSIONES PARCIALES.....	35
CAPÍTULO 3 DISEÑO DEL SISTEMA.....	36
3.1 PATRONES DE DISEÑO.....	36
3.1.1 Patrones de arquitectura.....	36
3.1.2 Patrones GRASP.....	37
3.2 ARQUITECTURA.....	39
3.2.1 Arquitectura utilizada.....	40
3.3 MODELO DEL DISEÑO.....	40
3.4 DIAGRAMAS DE CLASES DEL DISEÑO.....	41
3.5 DIAGRAMAS DE INTERACCIÓN.....	42
3.6 CONCEPCIÓN DEL MAPA DE NAVEGACIÓN.....	43
3.7 DIAGRAMA DE CLASES PERSISTENTES.....	44
3.8 MODELO DE DATOS.....	46
3.8.1 Diagrama Relacional.....	46
3.9 DIAGRAMA DE DESPLIEGUE.....	47
3.10 CONCLUSIONES PARCIALES.....	47
CAPÍTULO 4 IMPLEMENTACIÓN Y PRUEBA.....	48
4.1 MODELO DE IMPLEMENTACIÓN.....	48
4.1.1 Diagrama de Componentes.....	48
4.2 CÓDIGO FUENTE.....	51
4.2.2 Ejemplo de código fuente.....	51
4.3 MODELO DE PRUEBA.....	55
4.3.1 Casos de prueba.....	55

4.3.2 Pruebas de caja negra	57
CONCLUSIONES PARCIALES	57
CONCLUSIONES GENERALES.....	59
RECOMENDACIONES	60
REFERENCIAS BIBLIOGRÁFICAS	61
BIBLIOGRAFÍA.....	62

Índice de Figuras

Figura 1 Imagen del módulo de gestión de proveedores del software ERP Manager.....	7
Figura 2 Fases e iteraciones de la metodología RUP.....	15
Figura 3 Diagrama de casos de uso del negocio.....	22
Figura 4 Modelo de objetos.....	23
Figura 5 Diagrama de Casos de uso del sistema.....	27
Figura 6 Arquitectura modelo-vista-controlador.....	40
Figura 7 Diagrama de Clases del Diseño “Gestionar Ofertas”.....	42
Figura 8 Diagrama de colaboración “Elaborar Ofertas”.....	43
Figura 9 Diagrama de colaboración "Registrar Usuario".....	43
Figura 10 Mapa de navegación.....	44
Figura 11 Diagrama de Clases Persistentes.....	45
Figura 12 Diagrama Relacional.....	46
Figura 13 Diagrama de Despliegue.....	47
Figura 14 Diagrama de componentes de Productos.....	49
Figura 15 Diagrama de componentes de Proveedores.....	50

Índice de Tablas

Tabla 1: Actores del negocio.	20
Tabla 2: Trabajadores del negocio.	20
Tabla 3: Casos de uso del negocio.	22
Tabla 4 Actores del sistema.	26
Tabla 5 Descripción del caso de uso Gestionar proveedores.	28
Tabla 8 Escenario registrar proveedores.	55
Tabla 9 Descripción de las variables.	56

Introducción

En momentos donde la información se ha colocado, junto a las materias primas y a la mano de obra, en un lugar importante, como uno de los principales recursos que poseen las empresas modernas, se necesita depositar mucha confianza en la toma de decisiones sobre sus negocios. Aquellas entidades que como principio no tienen un control adecuado de su funcionamiento así como un seguimiento sostenido de su rendimiento, terminan quebrando.

La necesidad de contar con información en tiempo y forma es un tema crucial, particularmente en las empresas enfocadas en los servicios; "... medio para entregar valor a los clientes, facilitándoles los resultados que estos quieren conseguir sin asumir costos o riesgos específicos". Tomando como valor; "...el aspecto esencial del concepto de servicio. Desde el punto de vista del cliente, el valor consta de dos componentes básicos: **utilidad** y **garantía**. La utilidad es lo que el cliente recibe, mientras que la garantía reside en cómo se proporciona". (1)

"Por su parte las empresas en la esfera de los servicios de Tecnologías Informáticas (TI), enfocan su trabajo hacia el aumento de la eficiencia y la reducción de riesgos" (1). En esta materia, los servicios son prestados por proveedores o suministradores. Los proveedores representan las ofertas o cantidad de bienes o servicios a disposición de los consumidores o clientes, los cuales representan la parte de las demandas; cantidad de bienes o servicios que se solicitan o se desean en un determinado mercado de una economía a un precio específico.

Los proveedores o suministradores son empresas u organismos especializados que proporcionan los servicios solicitados por el cliente, otorgando vital importancia a la gestión de proveedores; para determinar los aspectos que puedan resultar de interés en un momento determinado o analizar información crítica de la empresa proveedora. Un buen manejo de los proveedores de una empresa garantiza que:

La organización obtenga mayores beneficios al contratar proveedores que brinden el mejor servicio.

Su principal objetivo es alcanzar la mayor calidad a un precio adecuado. Con este fin, se encarga de definir la estrategia según la cual orientar su labor:

1. Seleccionar los nuevos suministradores para las necesidades que vayan surgiendo en el servicio.

2. Definir y negociar los nuevos contratos, garantizando que queda constancia de los acuerdos financieros y de calidad alcanzados.
3. Gestionar la relación con los proveedores, lo que incluye velar por el cumplimiento de los contratos o actualizarlos si éstos pierden vigencia.
4. Aportar valor añadido al menor costo en aquellos servicios que prestan los proveedores. (1)

Una adecuada gestión de proveedores disminuye el tiempo de respuesta de una organización ante un problema y garantiza una mejor gestión de compras para las empresas clientes, puesto que disminuye la incertidumbre del comprador cuando este debe tomar una decisión de compra, garantizando grandes ahorros a la empresa. Es un proceso que determina cuales son los proveedores mejor posicionados en el mercado y cuáles se alinean mejor a las necesidades de un cliente específico en cuanto a las características del producto o servicio en cuestión, el plazo o precio de este.

Tomándose estos rasgos de las buenas practicas definidas por la Librería de Infraestructuras de Tecnologías Informáticas en su versión tres (ITILv3); definido como un conjunto de buenas prácticas destinadas a mejorar la gestión y provisión de servicios de TI, evitar los problemas asociados a los mismos y ofrecer un marco de actuación para que puedan ser solucionados con el menor impacto y la mayor brevedad posible. Estas prácticas se remontan a la década de los ochenta cuando el gobierno británico, preocupado por la calidad de los servicios TI de los que dependía la administración, solicitó a una de sus agencias, la *Central Computer and Telecommunications Agency (CCTA)*, para que desarrollara un estándar para la provisión eficiente de servicios TI. En la actualidad es *Office of Government Commerce (OGC)* el organismo encargado de velar por este estándar y la responsable de la última versión de ITILv3 que data del año 2007.

Por su aporte tan importante en el campo de las TI, ALBET Ingeniería y Sistemas, S.A. es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI), modelo de universidad productiva. ALBET posee los derechos comerciales de todos los productos y servicios que desarrolla la UCI y mediante la alianza con otras prestigiosas entidades ofrece soluciones integrales en la esfera de las tecnologías de la información y las comunicaciones. En su principal línea, los servicios de tecnologías informáticas, su mayor atención está basada en la adecuada gestión de sus proveedores de servicios, aunque el proceso de manejo o control de los mismos; no se lleva a cabo con la calidad requerida. La información relativa a los proveedores se almacena en formato duro (Acrobat PDF), en las máquinas locales de los especialistas

de la dirección comercial de la empresa, trayendo como consecuencias:

1. Dificultad para localizar las causas y entender el por qué del rendimiento del negocio.
2. Se pierde visibilidad sobre los procesos, retardando la generación de informes y la toma de decisiones.
3. Se dificulta el seleccionar un proveedor determinado y llevar el control de los productos y servicios contratados al mismo.
4. Se dificulta identificar, interpretar y responder a los eventos del negocio tan pronto como suceden.
5. Se dificulta la comparación de información en diferentes períodos e identificar comportamientos y evoluciones excepcionales.
6. Poca confiabilidad del formato en cuando a disponibilidad y a espacio de almacenamiento necesario.

Debido a la situación antes mencionada, se plantea como **problema científico**:

¿Cómo contribuir al mejoramiento de la Clasificación de proveedores y la Elaboración de las ofertas de la empresa ALBET?

Definiéndose como **Objeto de estudio**:

Aplicaciones para gestionar la Clasificación de proveedores y la Elaboración de sus ofertas.

Campo de acción:

Aplicaciones web para la Clasificación de proveedores y Elaboración de sus ofertas del subsistema de gestión de proveedores.

Como **objetivo general** se plantea:

Desarrollar los módulos “Clasificación de proveedores” y “Elaboración de sus ofertas” del subsistema de gestión de proveedores de la Plataforma de Gestión de servicios del Centro de Soporte de ALBET.

Para lograr el objetivo general antes expuesto se plantean los siguientes **objetivos específicos**:

- Caracterizar el funcionamiento de módulos “Clasificación y Selección de proveedores” y “Elaboración de sus ofertas” del subsistema de gestión de proveedores.
- Realizar el análisis de la situación actual de las aplicaciones para realizar la “Clasificación y Selección de proveedores” y “Elaboración de sus ofertas.”

- Diseñar los módulos “Clasificación y Selección de proveedores” y “Elaboración de sus ofertas del subsistema de gestión de proveedores.
- Implementar los módulos “Clasificación y Selección de proveedores” y “Elaboración de sus ofertas” del subsistema de gestión de proveedores.

Para darle cumplimiento a los objetivos propuestos se plantea el siguiente conjunto de **Tareas de la investigación:**

- Estudio y análisis de los principales sistemas de gestión de proveedores actuales.
- Estudio del proceso de Gestión de Proveedores de ITILv3.
- Entrevistas con los especialistas de ALBET para la descripción del proceso de gestión de proveedores.
- Análisis con los especialistas de ALBET de los módulos Clasificación de proveedores y Elaboración de sus ofertas.
- Diseño de los módulos Clasificación de proveedores y Elaboración de sus ofertas.
- Implementación de los módulos Clasificación de proveedores y Elaboración de sus ofertas.
- Pruebas de los módulos Clasificación de proveedores y Elaboración de sus ofertas.

El trabajo de diploma se estructura en cuatro capítulos:

Capítulo 1. Fundamentación Teórica:

Hace referencia fundamental a las líneas bases hacia las que va dirigido el trabajo. Se describen las herramientas en este campo, la metodología y los lenguajes de programación a usar, así como la arquitectura de desarrollo del mismo. El capítulo muestra algunos de los sistemas que presentan funcionalidades enmarcadas en la gestión de proveedores.

Capítulo 2. Características del sistema:

El capítulo define aspectos fundamentales a tener en cuenta a la hora de realizar el diseño de la solución. Aquí se especifican los requisitos funcionales y no funcionales del sistema, se analiza toda la lógica del negocio y se documenta para el posterior análisis.

Capítulo 3. Diseño del sistema:

El capítulo describe las funcionalidades del sistema que se propone, a partir de los resultados del análisis. Crea abstracciones que ayudan a mejorar la implementación de los desarrolladores y refleja los principales componentes y diagramas de la metodología de desarrollo de software utilizada.

Capítulo 4. Implementación y prueba:

En el capítulo se abordan aspectos relacionados con la implementación del sistema por parte de los desarrolladores. Se representan los principales componentes y diagramas, y se describen las pruebas realizadas a partir de los resultados obtenidos.

CAPITULO 1: Fundamentación teórica.

En este capítulo se estudia el estado actual de las aplicaciones de gestión de proveedores existentes, peculiaridades de estas aplicaciones, herramientas de desarrollo y los lenguajes de programación determinados para dar solución al problema científico.

1.1 Aplicaciones utilizadas para la gestión de proveedores.

En el mundo las empresas están comenzando a aplicar técnicas y metodologías orientadas a mejorar la gestión de sus procesos y a facilitar la toma de decisiones, apoyando en gran medida su productividad y convirtiendo los datos en lo más beneficioso para ellas: **información útil**. Se ha llegado a entender que una buena gestión de la información es vital para cualquier organización, especialmente en las empresas dedicadas a los servicios TI y a la gestión de proveedores; por lo que a continuación se hace un recuento sobre detalles de herramientas utilizadas en entidades enmarcadas en el campo de proveedores y la gestión de la información referente a estos:

1.1.1 SISPRO.

Este sistema incorpora la información generada en los procesos de calificación u homologación, evaluación del desempeño, desarrollo y planificación de proveedores. En el sistema se establece como obligatorio que los proveedores y contratistas, antes de iniciar su relación comercial con la empresa que presta este servicio, hayan superado un proceso de calificación de acuerdo a la criticidad del bien o servicio que vayan a suministrar.

Por otra parte, SISPRO utiliza a su vez los servicios prestados por el grupo Archilles, el cual es un servicio de gestión conjunta de información y documentación de proveedores y contratistas puesto en marcha por empresas de los sectores del agua, gas, electricidad, petróleo, navales y afines. La información, completa y actualizada, se basa en los datos que cada proveedor aporta anualmente a través de un cuestionario informatizado. Estas bases de datos permiten a las empresas compradoras disponer de una herramienta que les permita clasificar a sus proveedores y contratistas de forma eficaz, permanentemente actualizada y objetiva.

A los proveedores se les solicita información de carácter general, societario y mercantil, de recursos humanos, centros de trabajo, financiera, de sistemas de gestión de calidad, medio ambiente y prevención de riesgos laborales, de prácticas laborales, de productos y servicios (descripciones, gamas, referencias comerciales), así como documentación de cumplimiento de obligaciones fiscales y

de la seguridad social, de aseguramiento por responsabilidad civil, de certificaciones de calidad, medio ambiente y prevención, de los balances y cuentas de pérdidas y ganancias, y sobre informes de riesgo financiero.

Todas sus características describen un sistema bastante completo, pero la limitante y el problema principal de este sistema es que la empresa no tiene control sobre él. La empresa interesada solo accede al servicio poniendo a disposición de la entidad contratada toda la información tanto suya como de sus proveedores. Este elemento en un momento o situación determinada puede determinarse como un problema no solo para el proveedor sino también para la empresa que contrata el servicio, porque se le está confiando una serie de información valiosa a un tercero.

1.1.2 ERP Manager.

Efectúa el alta de los proveedores, cargando todos sus datos legales e impositivos en sus respectivas fichas. Este sistema chequea y ajusta las cuentas corrientes, realizando la emisión de órdenes de pago así como la carga de notas de débito/crédito. Múltiples formas de pago y emisión de certificados. El software ERP Manager permite configurar un completo circuito de autorizaciones para los pagos a sus proveedores.

Búsqueda	
Proveedor	00121 Global Trade SA

Datos principales			
Código	00121	Razón social de búsqueda	Global Trade SA
Razón social	Global Trade SA	Fecha de alta	10/05/1992
Calificación	<input checked="" type="radio"/> A <input type="radio"/> B <input type="radio"/> C	Estado	<input checked="" type="radio"/> Activo <input type="radio"/> Suspendido <input type="radio"/> Potencial <input type="radio"/> Sin Cta.Cte.

Dirección									
Calle	Mexico	Número	2543	Piso		Depto.		C.P.	
Barrio	Centro	Localidad	Capital Federal						
Pcia.	1	Capital Federal	País	ARG	Argentina				

Datos impositivos							
Agente Percepción	<input type="radio"/> No <input checked="" type="radio"/> Si	Cond. de IVA	Inscripto	Cuit	20-11198713-1	I/Brutos	901-787878-1
% Retenc. de IVA	10.00	% Ret. I/Brutos	5.00	Ret. de SUSS	<input checked="" type="radio"/> No <input type="radio"/> Si		
Inscr. Ganancias	<input type="radio"/> No <input checked="" type="radio"/> Si	Concepto Ret. Ganancias	5	Compra de Equipos Computacion			

Cta. Contable					
Cta. Contable	2111	Proveedores Varios	CBU		
Cpto. de C/Flow	2	Egresos	Cpto. de Gastos	200	Compra Equipos Reventa

Grabar Eliminar Cancelar

Figura 1 Imagen del módulo de gestión de proveedores del software ERP Manager.

Algunas de las funcionalidades que presta son:

Archivo maestro de proveedores: Crea un registro de cada uno de los usuarios con los datos necesarios, también brinda la oportunidad de adjuntar documentos al perfil del proveedor lo que puede resultar de gran ayuda en caso de que las descripciones de los proveedores se extienda demasiado o necesite de especificaciones de otro tipo.

Consultas por proveedor: Realiza consultas sobre los proveedores en cualquier momento para si hay necesidad de conocer el estado de alguno de los campos de los proveedores.

Búsqueda de proveedores: Permite realizar una búsqueda de los proveedores, saldos de cuentas corrientes y detalles de los mismos, pedidos de cotización pendientes de recepción.

Procesos: Permite realizar ajustes a las cuentas corrientes, ingreso de comprobantes de cuenta corriente: Facturas, notas de débito, notas de crédito y facturas de compra al contado.

Reportes: Permite realizar los reportes de interés sobre los datos de los proveedores los cuales pueden ser de importancia en un momento determinado, basándose en diferentes criterios.

Este software tiene un demo comercial disponible para descarga en internet, versión esta que no cumple con todas las funcionalidades para lo cual fue desarrollada. La aplicación en toda su magnitud es privativa, hay que pagar para hacer uso de ella.

1.1.3 @GeSTOCK

Es una aplicación que gestiona de cierto modo la información referente a las empresas, pero no de empresas dedicadas a prestar servicios informáticos. Ya que sus funcionalidades no se apegan a este objetivo. Está compuesto de los módulos: agenda, almacén, albaranes de entrada, albaranes de salida, proveedores y vendedores.

Entre sus funciones principales están:

- Permite la realización de un gran número de informes y de todo tipo de documentación.
- Cada módulo con color individual para mejorar la identificación, botones activos.
- Medios para importar la información almacenada en otras bases de datos.
- Realización de copias de seguridad automática.

Es una aplicación que gestiona la información referente a las empresas, pero que se torna sencilla a

los propósitos de ALBET, debido principalmente que su uso no está orientado a empresas en el campo de los servicios informáticos.

1.1.4 ¿Por qué no hacer uso de las aplicaciones existentes?

Las aplicaciones antes mencionadas, de manera general, recogen los rasgos más significativos e importantes de los procesos de gestión de proveedores, pero cada una por separado presenta inconvenientes de uso. La mayoría de estos sistemas son privativos o su uso no garantiza una privacidad de los datos a gestionar, puesto que la empresa debe solicitar los servicios y brindar información sobre sus negocios a un tercero. Estas condiciones anulan el interés de la empresa ALBET para acceder a ellas, por lo que se decidió realizar una aplicación web que esté acorde a sus necesidades.

1.2 Aplicaciones web.

Las aplicaciones web son aquellos sistemas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Estos sistemas tienen arquitectura cliente/servidor, donde tanto el cliente como el servidor y el protocolo mediante el que se comunican (HTTP) están estandarizados y no han de ser creados por el programador de aplicaciones.¹

Entre los tipos de aplicaciones web existentes se encuentran:

- Las redes sociales, que permiten el contacto de personas individuales sobre la base de algunas relaciones de interés común.
- Generación y publicación de contenidos los cuales, formados por Blogs y Wikis.
- Herramientas para la generación de contenidos; son aplicaciones web formados por una enorme cantidad de programas y utilidades que permiten a los usuario crear y compartir información.

¿Por qué crear una aplicación web?

Ahorra tiempo: Se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.

No hay problemas de compatibilidad: Basta tener un navegador actualizado para poder utilizarlas. No ocupan espacio en el disco duro.

¹Programación de aplicaciones web: historia, principios básicos y clientes web. Autor: Sergio Luján Mora

Consumo de recursos bajo: Dado que toda (o gran parte) de la aplicación no se encuentra en el ordenador del usuario final, las tareas que realiza el software no consumen recursos de la computadora cliente porque se realizan desde otro ordenador.

Multiplataforma: Se pueden usar desde cualquier sistema operativo puesto que sólo es necesario tener un navegador en el cliente.

Portables: Es independiente del ordenador donde se utilice porque se accede a través de una página web (sólo es necesario disponer de acceso a Internet).

La **disponibilidad** suele ser alta porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.

1.2.1 Arquitectura cliente/servidor.

“La idea básica de la arquitectura cliente/servidor es que un programa, el servidor, gestiona un recurso concreto y hace determinadas funciones solo cuando las pide otro, el cliente, que es quien interactúa con el usuario. Normalmente el cliente y el servidor están en ordenadores diferentes.” (2)

Se denomina cliente al proceso que inicia el diálogo o solicita los recursos al servidor, encargado de responder las solicitudes.

Entre las características principales de esta arquitectura se pueden destacar las siguientes:

- ✓ El servidor presenta a todos sus clientes una interfaz única y bien definida.
- ✓ El cliente no necesita conocer la lógica del negocio, solo la interfaz externa.
- ✓ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni su sistema operativo.(2)

1.3 Tecnologías y herramientas a usar en el desarrollo de la solución.

Existen numerosos lenguajes de programación, herramientas y tecnologías empleados en el desarrollo de aplicaciones web entre los que se destacan los mencionados a continuación.

1.3.1 Lenguajes utilizados.

Uno de los aspectos más importantes en el desarrollo de un software está enmarcado en el lenguaje de programación que se usa. A continuación se presentan varios lenguajes de programación que se utilizan en la construcción de aplicaciones web.

1.3.1.1 Java.

La ventaja de usar una aplicación en distintas arquitecturas o sistemas operativos sin tener que recompilar, es un ahorro de desarrollo, en esto están enmarcadas las características más importantes del lenguaje de programación Java y se debe a que no es el sistema operativo el que ejecuta directamente el programa, sino su máquina virtual.

Características principales del lenguaje java:

- ✚ **Seguro:** el código de Java puede ser ejecutado en un entorno que prohíbe la introducción de virus, borrar y modificar ficheros o la ejecución de operaciones que provoquen la caída del ordenador y la pérdida de datos.
- ✚ **Multitarea:** todo lo asume de forma paralela, con varias tareas de forma simultánea. Un programa sobre Java puede procesar diferentes tareas independientes.
- ✚ **Permite** crear programas para que funcionen en un navegador web y en servicios web así como combinar aplicaciones o servicios que usan el lenguaje para crear aplicaciones totalmente personalizadas.
- ✚ **Permite** crear aplicaciones para servidores como foros en línea, tiendas, encuestas y procesamiento de formularios HTML.
- ✚ **Independiente de la plataforma:** Java se compila a un formato de código de byte que puede ser leído e interpretado por muchas plataformas.
- ✚ **Manejo automático de memoria:** el lenguaje Java se compone de objetos, esos objetos pueden y deben crearse, y tienen una vida que dura hasta su destrucción. Mientras que la creación de los objetos se deja bajo la voluntad del programador, la destrucción definitiva de un objeto ocurre cuando no es más referenciado por otros objetos del programa. De esta forma, se elimina una de las causas más comunes de error en otros lenguajes, como la destrucción por el programador de objetos aún en uso en el programa, o la falta de destrucción de objetos que ya son inútiles, pues no se usan en el resto de la ejecución, pero que molestan con empleo de recursos. Esta técnica de manejo automático de la memoria ocupada por los objetos se denomina recolección de basura (garbage collection). En una aplicación Java hay siempre un proceso, ejecutado como un hilo de ejecución separado, que se ocupa de recorrer la memoria donde se encuentran los objetos, y determina cuáles pueden liberarse y destruirse. (3)

Se usará como lenguaje de programación, **Groovy**; puesto que es un lenguaje ágil para plataformas Java, muy notable en el desarrollo de aplicaciones web orientado a objetos, que soporta los tipos de

datos estándar, pero añade características dinámicas y sintácticas presentes en otros lenguajes como Python, Smalltalk o Ruby. Groovy aporta una sintaxis que aumenta la productividad y un entorno de ejecución que permite manejar objetos de forma que en Java serían extremadamente complicados, sus códigos son similares y la mayoría de lo que se hace con Java puede hacerse también en Groovy, incluyendo: (4)

- La separación del código en paquetes.
- Definición de clases (excepto clases internas) y métodos.
- Estructuras de control excepto el bucle for (;;).
- Operadores, expresiones y asignaciones.
- Gestión de excepciones.

Además Groovy incluye mejoras sintácticas en relación con:

- La facilidad de manejo de objetos, mediante expresiones y sintaxis.
- Brevedad: el código Groovy es mucho más breve que el de Java.
- En este todo es un objeto.

1.3.1.2 JavaScript:

JavaScript es un lenguaje de programación utilizado para unir el conjunto de tecnologías usadas en la web, es un lenguaje de scripting basado en objetos. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas, se caracteriza por ser un lenguaje basado en prototipos, un lenguaje de programación interpretado; por lo que no es necesario compilar los programas para ejecutarlos. Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Es un lenguaje con el cual no se realiza un programa sino que se usa para mejorar el funcionamiento de estos. (5)

1.3.1.3 Lenguaje AJAX

El término AJAX se presentó por primera vez en el artículo "*Ajax: A New Approach to Web Applications*" publicado por Jesse James Garrett el 18 de febrero de 2005. Hasta ese momento, no existía un término normalizado que hiciera referencia a un nuevo tipo de aplicación web que estaba apareciendo. En realidad, el término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML." (6)

Ajax no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes.

Las tecnologías que forman AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías. (6)

En las aplicaciones web tradicionales, las acciones del usuario en la página desencadenan llamadas al servidor. Una vez procesada la petición al usuario, el servidor devuelve una nueva página HTML al navegador del usuario. Técnica que no crea una buena sensación al usuario, puesto que este tiene que esperar a que se recargue la página con los cambios solicitados. AJAX permite mejorar completamente la interacción usuario-aplicación evitando que la presentación al usuario de las páginas HTML se produzcan con recargas constantes de la página, ya que el intercambio de información con el servidor se realiza en un segundo plano.

1.4 Framework.

Un framework es una tecnología que simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, un framework proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y fácil de mantener, que encapsulan operaciones complejas en instrucciones sencillas.

1.4.1 Grails.

Grails es un framework para aplicaciones web libres, desarrolladas sobre el lenguaje de programación Groovy (el cual a su vez está diseñado para correr sobre la máquina virtual de Java). Grails pretende ser un framework altamente productivo siguiendo paradigmas tales como convención sobre configuración o no te repitas (*DRY*), proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador. (7)

Grails ha sido impulsado principalmente por la empresa G2One, la cual fue adquirida por la desarrolladora de software libre *SpringSource* en noviembre del 2008. En agosto del 2009 *SpringSource* fue a su vez adquirida por *VMWare*, empresa especializada en virtualización de sistemas, el framework fue conocido como '*GroovyonRails*' (el nombre cambió en respuesta al pedido de David HeinemeierHansson, fundador de *RubyonRails*) el cual se inició en julio del 2005, con la

versión 0.1. (7)

1.4.2 Dojotoolkit.

Dojo, Librería de clases JavaScript es un framework que contiene *Apis* y *widgets* (controles) para facilitar el desarrollo de aplicaciones web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, los efectos de interfaz de usuario (UI), abstracción de eventos, almacenamiento de Apis en el cliente, e interacción de Apis con AJAX. Resuelve asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de URL para luego regresar a ellas, y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente. Es conocido como "la navaja suiza del ejército de las bibliotecas JavaScript." Proporciona una gama más amplia de opciones en una sola biblioteca JavaScript y es compatible con navegadores antiguos. (8)

1.5 Metodología de desarrollo de Software.

“Una metodología, (del griego μέθοδος, método y logía); es ciencia del método, conjunto de métodos que se siguen en una investigación científica o en una exposición doctrinal.” Las metodologías de desarrollo de software son un conjunto de procedimientos y técnicas para el desarrollo de aplicaciones que tienen como línea fundamental definir cuáles son las tareas a realizar. (9)

1.5.1 Metodología de desarrollo RUP.

“El Proceso Unificado de Rational (RUP), es un proceso de ingeniería de software planteado por Kruchten en 1996 cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido, el cual cubre el ciclo de vida y desarrollo de software”².

La metodología RUP, llamada así por sus siglas en inglés (*Rational Unified Process*), se divide en 4 fases: (10)

- **Inicio:** el objetivo en esta etapa es determinar la visión del proyecto.
- **Elaboración:** en esta etapa el objetivo es determinar la arquitectura óptima.
- **Construcción:** en esta etapa el objetivo es llegar a obtener la capacidad operacional inicial.
- **Transición:** el objetivo es obtener la liberación del proyecto.

²DÍAZ-ANTÓN et al. 2004

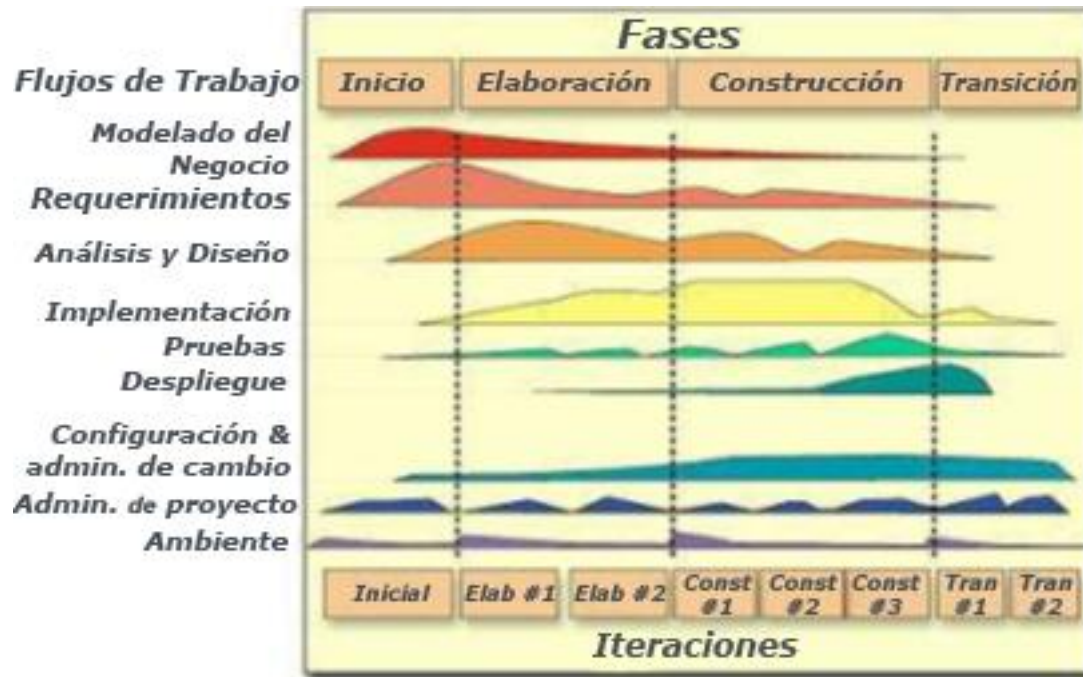


Figura 2 Fases e iteraciones de la metodología RUP.

El Proceso Unificado es una metodología pesada puesto que está guiada por una fuerte planificación durante todo el proceso de desarrollo, en el cual se establecen estrictamente las actividades involucradas, los roles definidos, los artefactos que se deben producir, las herramientas y notaciones que serán utilizadas, así como el modelado y una documentación detallada. En el mismo existe un contrato prefijado y el cliente interactúa con el equipo de desarrollo mediante reuniones y entrevistas, donde se puntualizan las precisiones del producto.

El Proceso Unificado utiliza el Lenguaje Unificado de Modelado (*Unified Modeling Language*, UML) para preparar todos los esquemas de un sistema, el cual es una parte esencial del Proceso Unificado.

El ciclo de vida de RUP se caracteriza por ser: (10)

Dirigido por casos de uso:

Los casos de uso son representaciones que reflejan lo que los usuarios futuros necesitan y desean, se capta cuando se modela el negocio y se representan a través de los requerimientos o requisitos funcionales. Los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso.

Centrado en la arquitectura:

La arquitectura permite observar los resultados del desarrollo de software antes de que este se comience a realizar, puesto que describe diferentes vistas del sistema en construcción. “La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. Debido a que lo que es significativo depende en parte de una valoración, que a su vez, se adquiere con la experiencia, el valor de una arquitectura depende de las personas que se hayan responsabilizado de su creación.”³

Iterativo e Incremental:

Cada fase se desarrolla en iteraciones, las mismas involucran actividades de todos los flujos de trabajo. En cada iteración se identifican y especifican los casos de uso relevantes, se crea un diseño utilizando la arquitectura seleccionada como guía, se implementa el diseño mediante componentes y se verifican que los componentes satisfacen los casos de uso. Se continúa con la siguiente iteración cuando la actual cumple con sus objetivos.

1.5.2 UML (Unified Modeling Language).

El UML es el lenguaje de modelado más reconocido en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. (11) UML permite modelar sistemas utilizando técnicas de POO. Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos. UML es además un método formal de modelado, que aporta las siguientes ventajas:

1. Mayor rigor en la especificación.
2. Permite realizar una verificación y validación del modelo realizado.
3. Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente, generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel de la estructura de un proyecto.

Los objetivos de UML se pueden sintetizar en: (12)

- **Visualizar:** UML permite expresar de una forma gráfica un sistema de forma que otro lo puede entender.
- **Especificar:** UML permite especificar cuáles son las características de un sistema antes de su

³JACOBSON et al. 2000

construcción.

- **Construir:** A partir de los modelos especificados se pueden construir los sistemas diseñados.
- **Documentar:** Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

1.6 Herramientas CASE.

Las herramientas CASE (*Computer Aided Software Engineering*) son un conjunto de programas que ayudan a los analistas, ingenieros de software y desarrolladores durante el ciclo de vida de desarrollo de software a llevar a cabo las tareas del modo más eficiente y efectivo posible.

1.6.1 Visual Paradigm.

“Visual Paradigm es una herramienta de UML profesional que soporta el ciclo completo de desarrollo de software. Esta herramienta ayuda a la construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Es un estándar ampliamente utilizado actualmente en las empresas para el modelado de software. Que además proporciona valiosa ayuda a los profesionales visualizando, comunicando y aplicando sus diseños. Esta herramienta ayuda a los equipos de desarrollo de software para sobresalir a todo el modelo de acumulación de trabajo y así desplegar el proceso de desarrollo de software, lo que permite maximizar y acelerar tanto las contribuciones individuales como las del equipo.” (13)

Entre las funcionalidades que lo destacan, entre las herramientas de su tipo que hace que sea tomado como herramienta de UML por el arquitecto de la Plataforma de Gestión de Servicios del Centro de Soporte de ALBET se pueden mencionar:

1. Diagramas de Procesos de Negocio, Decisión, Actor de Negocio
2. Ingeniería inversa. Modelos desde el código, diagramas desde el código.
3. Multiplataforma, soportado por (Windows/Linux/Mac OS X).
4. Generación de bases de datos. Transformación de diagramas Entidad-Relación en tablas de bases de datos.

1.6.2 IDE de desarrollo.

El NetBeans es un IDE de código abierto escrito completamente en Java, soporta el desarrollo de

todos los tipos de aplicación Java (J2SE, web y aplicaciones móviles). Cuenta con control de versiones, un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadatos en un espacio para archivos plano, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente, emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad. Puede extenderse usando otros lenguajes de programación como son C/C++ y Python y trae incluido entre sus novedades más destacadas el uso de Groovy. (14)

1.7 Sistema de gestión de Base de datos PostgreSQL.

PostgreSQL es un servidor de base de datos relacional orientada a objetos de software libre, liberado bajo la licencia BSD, de código abierto, que por definición significa que se puede obtener el código fuente, es decir, se puede utilizar el programa y modificar libremente sin los límites del software propietario, para satisfacer sus necesidades. (15)

Entre sus características se encuentran:

- Estabilidad.
- Alto rendimiento.
- Flexibilidad: puede funcionar en la mayoría de los sistemas Unix, además de que puede ser integrado a ambientes de Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.
- Se puede extender su funcionalidad.
- Su gran compatibilidad permite crear o migrar aplicaciones desde Access, Visual Basic, Visual Fox Pro, Delphi para usar PostgreSQL como servidor de base de datos. (15)

1.8 Conclusiones parciales

A partir de un análisis previo de la importancia que tiene la realización de los procesos de gestión de proveedores para las empresas enfocadas en los servicios de tecnologías de información. Se realizó una investigación de las tecnologías informáticas existentes en este campo a partir de sus condiciones de uso y sus características y se planteó la necesidad de una herramienta con las funcionalidades requeridas por ALBET para la realización de sus procesos de gestión. Se decidió desarrollar una aplicación web por sus características y ventajas y se describieron las tecnologías, herramientas y metodologías a usar en el desarrollo de la solución.

Capítulo 2 Características del sistema.

A través de este capítulo se describen el objeto de estudio y automatización, los principales problemas existentes en cuanto al proceso de gestión de proveedores de la empresa ALBET, se analizan los requisitos funcionales, los no funcionales, actores y casos de uso del negocio.

2.1 Objeto de estudio.

En la empresa cubana ALBET se desarrolla todo un proceso para la gestión de sus proveedores y la elaboración de sus ofertas. El propósito fundamental de estos procesos es lograr la selección de los proveedores con características que respondan a las necesidades primordiales de la empresa para elevar la calidad de los servicios que se brindan, así como una buena elaboración de la oferta.

2.2 Flujo actual de los procesos.

Los procesos se llevan a cabo en estos momentos mediante una serie de reuniones entre los implicados, en las cuales se brinda información sobre el negocio, se analizan las solicitudes, (requisitos funcionales y no funcionales identificados con el cliente), los acuerdos sobre los servicios que se solicitan, se realizan estudios de mercado y de pre-factibilidad o clasificación de proveedores por mercadotecnia, tiempos de respuesta al cliente y descripción de cada uno de los componentes que integran la solución, todos los análisis de estos datos recogidos en estas reuniones son guardados en formato duro para su posterior análisis.

2.3 Análisis crítico de la ejecución de procesos.

Actualmente los procesos de gestión de proveedores y elaboración de las ofertas son algo engorroso, debido a que este se realiza de forma manual, a través del análisis de datos almacenados previamente (formato PDF) en las máquinas del Centro de Soporte de ALBET, que conlleva a la pérdida de elementos de importancia a la hora de seleccionar los proveedores, evaluar o comparar a otros. La poca accesibilidad de los datos conlleva muchas veces a que, por comodidad o agobio de los participantes en el proceso se dejen de analizar elementos importantes. Estas circunstancias hacen que se tomen decisiones basándose más en la intuición que en la propia información con que cuenta la empresa, aumentando los costos y el tiempo de respuesta de la misma.

2.4 Objeto de automatización.

Se desean automatizar los procesos mencionados anteriormente, mediante un sistema que permita

recoger los datos de los proveedores y las ofertas, facilitando la inserción, evaluación y modificación de los mismos.

Ahora bien, ¿Cómo se realizan actualmente estos procesos en la empresa?

A continuación se presenta el modelado del negocio para su mejor entendimiento.

2.5 Modelo de negocio.

El modelado del negocio es una técnica para comprender los procesos de negocio de una empresa, para entender la estructura y la dinámica de la misma, los problemas actuales y asegurarse que los clientes, usuarios finales y desarrolladores tienen una idea común de la organización.

2.5.1 Actores del negocio.

Los actores del negocio son cualquier persona, individuo, grupo o entidad que se encargue de realizar las tareas del mismo, es decir, los que inician los procesos y/o las actividades del negocio. (10)

Tabla 1: Actores del negocio.

<i>Actores del negocio</i>	<i>Descripciones</i>
Director de proyecto de ALBET.	Responsable de crear la notificación.
Especialista general de Mercadotecnia.	Responsable de la elaboración de la oferta, estudio de factibilidad (Clasificación de proveedores), describe la solución técnica (Selección de proveedores)

2.5.2 Trabajadores del negocio.

El trabajador del negocio define el rol de un individuo o grupo de individuos, máquina o sistema automatizado que participa directamente en los procesos que se llevan en el negocio, realizan las actividades y son propietarios de elementos, pero no se benefician de forma alguna con los resultados del proceso. (10)

Tabla 2: Trabajadores del negocio.

<i>Trabajador</i>	<i>Descripción</i>
-------------------	--------------------

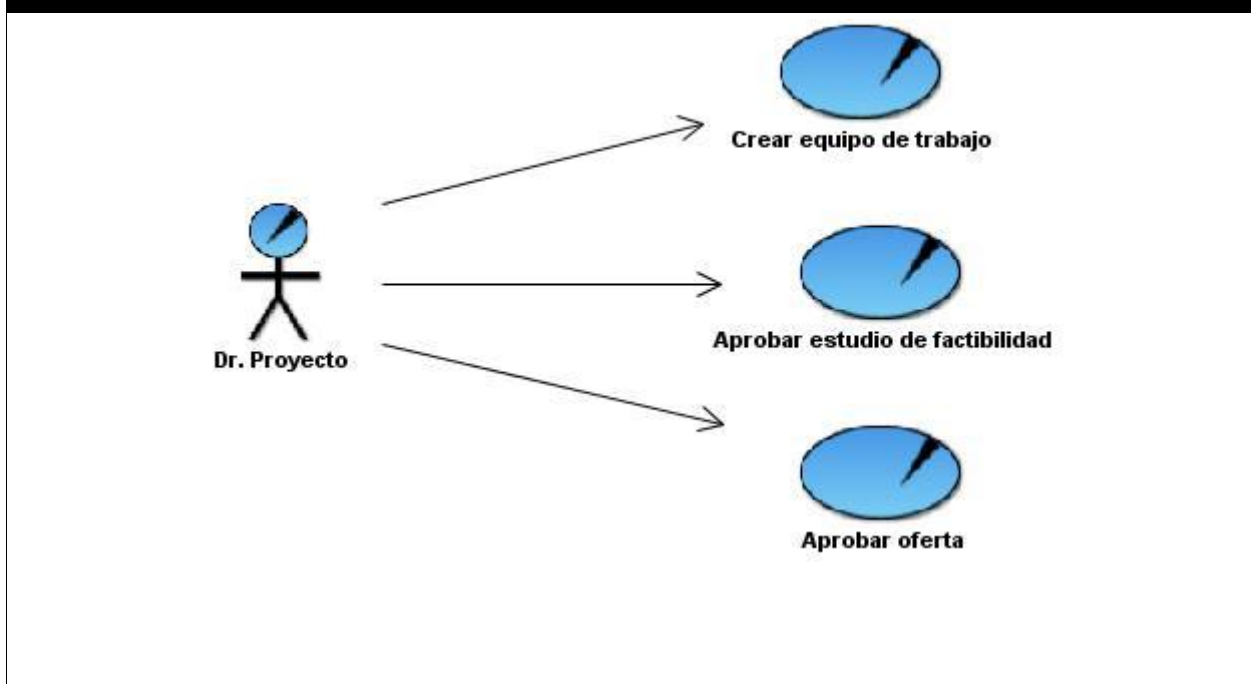
Líder de proyecto.

Es el responsable del equipo de trabajo encargado de la realización del informe de diseño de la solución técnica y del informe de factibilidad económica y oferta de negocio.

2.5.3 Diagrama de casos de uso del negocio.

El principal objetivo que se persiguen estos diagramas es describir el flujo de procesos de forma gráfica, tratar de conocer cómo funciona todo el negocio, desde que comienza hasta que termina, describiendo la forma en que interactúa con los actores, para lograr un mayor entendimiento entre los clientes y desarrolladores. Los diagramas de casos de uso permiten identificar las responsabilidades de los actores y trabajadores del negocio.

Diagrama de casos de uso del negocio.



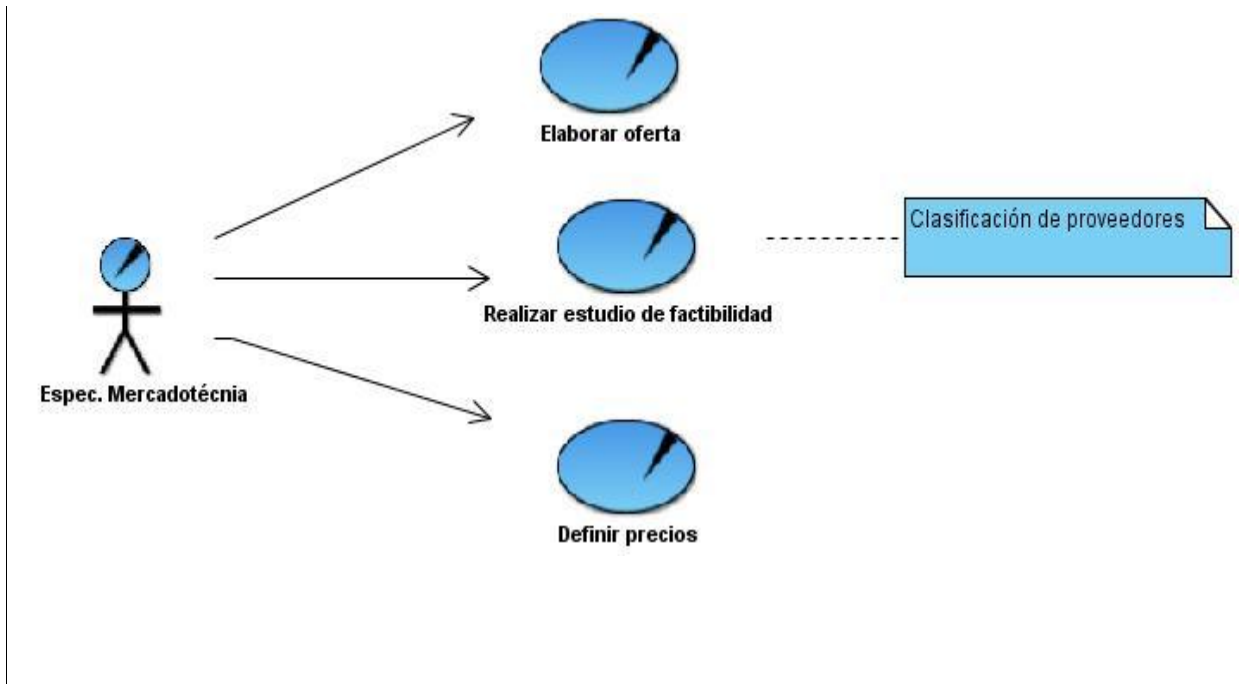


Figura 3 Diagrama de casos de uso del negocio.

A continuación se presenta una descripción sencilla de los procesos de negocio y los actores que se relacionan con estos.

Tabla 3: Casos de uso del negocio.

<i>Casos de uso del negocio</i>	<i>Descripción de casos de uso</i>	<i>Actores</i>
Elaborar oferta.	Proceso mediante el cual el responsable creará la oferta a partir de las decisiones y los análisis realizados.	Especialista de Mercadotecnia.
Realizar estudio de factibilidad.	Realiza un estudio de factibilidad o clasificación de proveedores.	Especialista de Mercadotecnia.

Describir solución técnica.	Proceso mediante el cual el responsable realizará la selección de proveedores.	Especialista de Mercadotecnia.
-----------------------------	--	--------------------------------

2.5.4 Modelo de objetos.

Se conoce como un modelo que muestra la relación entre trabajadores y entidades del negocio, en el flujo de trabajo de modelado del negocio. Describe cómo cada caso de uso del negocio es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio. (12)

Modelo de objetos.

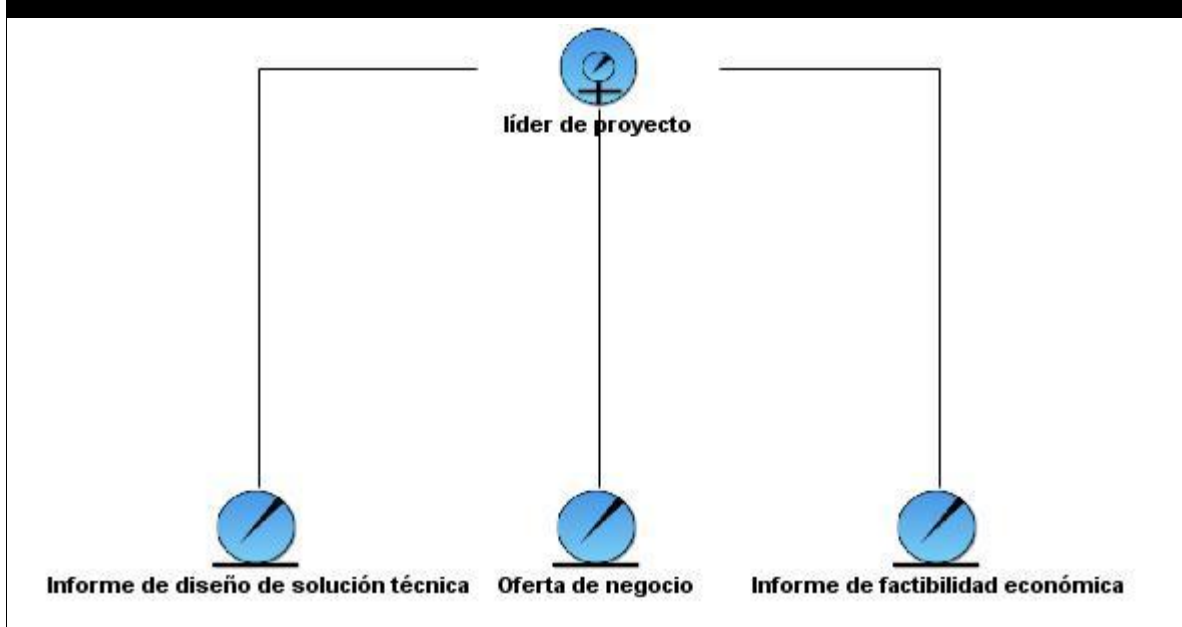


Figura 4 Modelo de objetos

2.6 Requisitos Funcionales.

Los requisitos funcionales especifican qué debe hacer el sistema en materia de funcionalidades para que cumpla con los **objetivos planteados** al inicio del trabajo.

Se identificaron como requisitos funcionales (RF) los siguientes:

RF1: Autenticar Usuario.

El usuario ingresa su usuario y contraseña y procede a autenticarse. El sistema comprueba que el usuario exista en la BD y que la contraseña esté escrita correctamente.

RF1.1 Verificar usuario y contraseña, con las existentes en la Base de datos.

RF2: Gestionar Usuarios.

El sistema debe permitir que el administrador realice acciones como crear, eliminar, actualizar o buscar un usuario.

RF3: Gestionar Proveedores.

El sistema debe permitir a los integrantes del equipo de trabajo realizar acciones como crear, eliminar, actualizar o buscar un proveedor.

RF3.1 Insertar proveedor.

RF3.2 Eliminar proveedor.

RF3.3 Modificar proveedor.

RF3.4 Visualizar proveedor.

RF4: Clasificar Proveedores.

El sistema debe analizar los datos y representar cual es el proveedor con mejores capacidades para responder a las necesidades de un cliente.

RF5: Seleccionar Proveedores.

El sistema debe mostrar los datos de los proveedores de una forma gráfica a partir de la cual se pueden seleccionar los proveedores por sus características.

RF6: Gestionar Ofertas.

El sistema debe dar opciones a los integrantes del equipo de trabajo realizar acciones como crear, eliminar, modificar o buscar una oferta.

RF6.1 Insertar Oferta.

RF6.2 Eliminar Oferta.

RF6.3 Modificar Oferta.

RF6.4 Visualizar Oferta.

RF 7: Listar proveedores.

El sistema debe dar la posibilidad de listar los proveedores previamente creados por un integrante del equipo de trabajo.

RF 8: Listar usuarios.

El sistema debe permitir listar los usuarios previamente creados por el administrador.

2.7 Requisitos no funcionales.

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, como restricciones del entorno, de implementación o rendimiento. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable.

Se identificaron como requisitos no funcionales los siguientes:

Apariencia o Interfaz Externa.

El sistema tendrá una interfaz web sencilla, fácil de usar por el usuario, además su funcionamiento será de fácil comprensión que hará que no sea necesario mucho entrenamiento para su utilización.

Usabilidad.

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web.

Rendimiento.

Como se tratará de una aplicación cliente/servidor debe ser eficaz, con la capacidad adecuada de procesamiento y cálculo, así como que requiere de un tiempo de respuesta relativamente pequeño.

Disponibilidad.

El sistema estará disponible mediante un servidor web el cual debe estar activo en el horario de trabajo del personal o de definirlo de esa manera, las 24 horas.

Portabilidad.

El sistema tendrá la posibilidad de ser multiplataforma.

Seguridad.

Para garantizar la integridad y la confidencialidad de la información que se maneja, se establece un sistema de permisos a usuarios, la entrada de estos al sistema será verificado, mediante autenticación por roles. Además de que las contraseñas se van a gestionar cifradas.

Software.

En el lado del cliente debe existir un navegador con soporte JavaScript. En el lado del servidor debe estar instalado el gestor de base de datos PostgreSQL, además de que debe contarse con servidor Apache2 y Tomcat.

Hardware en el cliente.

Procesador Pentium II o superior.

256 MB de memoria RAM o superior.

4 GB de disco duro.

Hardware en el servidor.

Procesador Pentium IV o superior

1 GB de memoria RAM o superior

80 GB de disco duro.

2.8 Modelado del sistema.

A continuación se presenta el modelado del sistema partiendo de la definición de **actor del sistema** los cuales:

“Pueden representar el rol que juegan una o varias personas, un equipo o un sistema automatizado”⁴. Representan terceros fuera del sistema, que interactúan con este. Estos suelen corresponderse con los trabajadores del negocio.

A continuación se presenta la descripción de los actores del sistema y de sus funciones dentro del mismo:

Tabla 4 Actores del sistema.

Actores	Descripción
Administrador	Posee todos los privilegios sobre la aplicación, es el responsable de manejar correctamente la información con la que contará el sistema. Responsable de realizar la gestión de usuarios, ya sea adicionar, modificar, eliminar y visualizar los datos que se manejan en la Base de datos

⁴Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. El proceso unificado de desarrollo de softwares. I.: Pearson Education, 2000.

	como otorgar los permisos necesarios a los usuarios para poder acceder a la aplicación.
Usuario	Tiene acceso a un grupo de funcionalidades que el administrador le defina, cuenta con privilegios necesarios para navegar en el sitio accediendo a los vínculos que se le definieron como accesibles y tiene acceso a realizar sobre el mismo las operaciones básicas, como visualizar parte de la información
Líder de Proyecto	Es el responsable del equipo que interactúa con las funcionalidades del sistema.

2.8.1 Diagrama de Casos de uso del sistema.

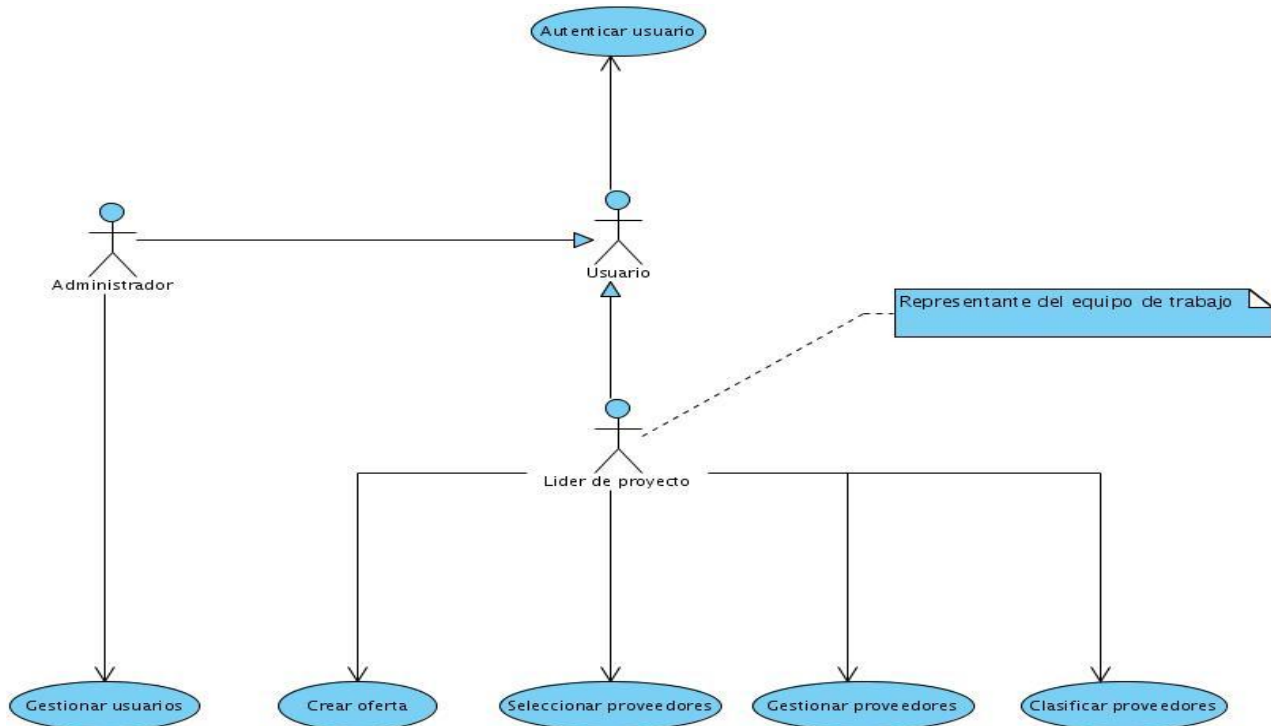


Figura 5 Diagrama de Casos de uso del sistema.

2.8.2 Descripción textual de los casos de uso del sistema.

A continuación se explican los casos de uso definidos para satisfacer los requisitos funcionales del

sistema.

Tabla 5 Descripción del caso de uso Gestionar proveedores.

Caso de Uso:	Gestionar proveedores
Actores:	Líder de proyecto.
Resumen:	<p>El CU se inicia cuando el líder de proyecto va a realizar algunas de las siguientes operaciones:</p> <ul style="list-style-type: none"> - Registrar proveedores: cuando el líder de proyecto decide registrar un nuevo proveedor en el sistema, llena los datos necesarios en un formulario y el nuevo proveedor es registrado, finalizando así el CU. - Modificar datos de proveedores: cuando el líder de proyecto necesita modificar datos de un proveedor existente, modifica los datos e indica actualizar los mismos, finalizando así el CU. - Buscar y visualizar datos de proveedores: el líder de proyecto provee los parámetros por los cuales se realizará la búsqueda, el sistema muestra los resultados para dicha búsqueda, finalizando así el CU. - Eliminar proveedores: el líder de proyecto selecciona el proveedor que desea eliminar, el sistema elimina el proveedor seleccionado, finalizando así el CU.
Precondiciones:	Los datos del proveedor serán registrados, modificados, buscados o eliminados.
Referencias	RF 2.1 RF 2.2 RF 2.3 RF 2.4
Prioridad	Crítico
Flujo Normal de Eventos	
Sección “Adicionar proveedores”	

Acción del Actor	Respuesta del Sistema
1.1 El líder de proyecto selecciona en el sistema la opción de adicionar proveedor.	1.2. La aplicación muestra un formulario solicitándole al líder de proyecto que introduzca los nuevos datos del proveedor a adicionar.
1.3. El líder de proyecto introduce en el sistema los datos.	1.4. La aplicación verifica que los datos estén escritos correctamente y que no exista un proveedor con esos datos en la BD.
	1.5 Se actualiza la BD y la aplicación muestra un mensaje de confirmación de que los datos del proveedor se han añadido correctamente.

Prototipo de Interfaz

☯ Proveedores

(*) Campos Obligatorios.

Detalles del proveedor

* Nombre:

* Tipo:

* Representante:

Descripción:

Información para la clasificación

* Ubicación geográfica:

* Cant. clientes:

* Formas de envío: Aéreo
 Marítima
 Terrestre

Dirección

* Calle:

* Número:

* Codigo Postal:

Municipio:

Provincia:

* País:

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
	1.4.1 Si los datos del proveedor no están escritos correctamente o este ya existe en la BD el sistema muestra un error y regresa al paso 1.2.
Prototipo de Interfaz	
<div style="border: 1px solid #ccc; padding: 10px;"> <div style="background-color: #e6f2ff; padding: 5px; border: 1px solid #add8e6; margin-bottom: 10px;"> ⊖ Proveedores </div> <p>(*) Campos Obligatorios.</p> <div style="border: 2px solid #ff0000; padding: 5px; margin: 10px 0;"> <p style="color: #ff0000; font-size: 0.9em;"> ⓘ La propiedad "codigoPostal" no puede ser nula. ⓘ La propiedad "correo" no puede ser vacía. ⓘ La propiedad "nombre" con valor "Desoft" debe ser única. ⓘ La propiedad "representante" no puede ser vacía. </p> </div> <div style="border: 1px solid #add8e6; padding: 10px; margin: 10px 0;"> <p>Detalles del proveedor</p> <p>* Nombre: <input type="text" value="Desoft"/></p> <p>* Tipo: <input style="border: none; border-bottom: 1px solid #ccc;" type="text" value="Internacional"/> ▾</p> <p>* Representante: <input type="text"/></p> <p>Descripción: <input style="width: 100%; height: 20px;" type="text"/></p> </div> </div>	
Poscondiciones	Se actualiza la Base de datos
Flujo Normal de Eventos	
Sección "Eliminar proveedores"	
Acción del Actor	Respuesta del Sistema
2.1 El líder de proyecto selecciona en el sistema la opción de eliminar proveedor a través de la interfaz Listar Proveedor .	2.2. Si el líder de proyecto selecciona un proveedor a eliminar, la aplicación muestra la interfaz donde visualiza los datos de los proveedores.
2.3 El administrador selecciona la opción Borrar.	2.3. La aplicación verifica que el líder de proyecto desea borrar el proveedor.

2.4 Si el selecciona la opción positiva. Ir al paso 2.5, si selecciona la opción negativa, ir a la opción 2.5.1

2.5 El sistema elimina el proveedor y muestra la tabla de proveedores actualizada.

Prototipo de Interfaz

Listado de proveedores

i Proveedor 117 borrado									
Id	Nombre	Tipo	Correo	Teléfono	Representante	Calle	Número	C. Postal	País
31	Fasdf	Internacional	fas@es.ci	325523	Asd	Asdf	2333	4523523	As
112	Fasd	Internacional	sad@es.ci	52345	Fasdf	Fas	234	10400	Japon
113	Fasdfg	Internacional	as@es.ci	5234	Afsd	Fasd	234	1040	Asdf
114	Fasdgdds	Internacional	ds@es.ci	452345	Gsdf	Fas	234	52345	Fasd
115	Fasdt	Internacional	fas@es.ci	42345234	Fasd	Fasd	34	10400	Fasd
116	Fasdghto	Internacional	asd@es.ci	52345	Fasd	Fasd	523	52345	arg
118	Gdf	Internacional	sf@es.ci	346345634	Fasd	Gsd	52345	52345	brb

Flujos Alternos

Acción del Actor

Respuesta del Sistema

2.5.1 El sistema muestra la interfaz Visualizar proveedor sin cambios.

Prototipo de Interfaz

Vista general del proveedor

Id	31
Nombre	Gsdfg
Tipo	Interno
Correo	sdf@s.cu
Fax	
Descripción	
Representante	Gsdf
Dir Url	
Teléfono	4
Contratos	Crear un contrato para este proveedor contrato de prueba
Calle	Fas
Número	34
Código Postal	423
Municipio	
Provincia	
País	Saf
Productos	gsddryt

 Nuevo  Modificar  Borrar



Poscondiciones Se elimina un proveedor y se actualiza la Base de datos.

Sección “Modificar proveedores”

Acción del Actor	Respuesta del Sistema
2.1 El líder de proyecto selecciona en el sistema la opción de modificar proveedores desde la interfaz Listar Proveedor .	2.2. El sistema muestra un formulario con los datos del proveedor a ser modificado.
2.3. El líder de proyecto modifica los datos que desea del proveedor.	2.4 Verifica que los datos introducidos estén correctos, si son correctos ir al paso 2.5, si no son datos correctos ir al paso 2.4.1
	2.5. La aplicación actualiza la Base de datos y se visualiza la interfaz Visualizar Proveedor actualizada.

Prototipo de Interfaz

Nombre	<input type="text" value="Gsdfg"/>
Tipo	<input type="text" value="Interno"/>
Correo	<input type="text" value="sdf@s.cu"/>
Fax	<input type="text"/>
Descripción	<div style="border: 1px solid gray; height: 100px;"></div>
Representante	<input type="text" value="Gsd"/>
Dir Url	<input type="text"/>
Teléfono	<input type="text" value="4"/>
Calle	<input type="text" value="Fas"/>
Número	<input type="text" value="34"/>
Código Postal	<input type="text" value="423"/>
Municipio	<input type="text"/>
Provincia	<input type="text"/>
País	<input type="text" value="Saf"/>

 **Actualizar**
 **Borrar**

Flujos Alternos

Acción del Actor	Respuesta del Sistema
	2.4.1 Muestra un mensaje de error de acuerdo al error de entrada: Mensaje: Los datos entrados no son correctos, y regresa al paso 2.2.

Prototipo de Interfaz

❗ El correo [correo] especificado para [class modulos.Proveedor] de valor [sfas] no es una dirección de correo válida.

Nombre	<input type="text" value="Gsdfg"/>
Tipo	<input type="text" value="Interno"/>
Correo	<input type="text" value="sfas"/>
Fax	<input type="text"/>
Descripción	<div style="border: 1px solid gray; height: 100px;"></div>
Representante	<input type="text" value="Gsdf"/>
Dir Url	<input type="text"/>
Teléfono	<input type="text" value="450823105"/>
Calle	<input type="text" value="Fas"/>
Número	<input type="text" value="34"/>
Código Postal	<input type="text" value="423"/>
Municipio	<input type="text"/>
Provincia	<input type="text"/>
País	<input type="text" value="Saf"/>

Actualizar
 Borrar

Poscondiciones

Se modifican los datos del proveedor y se actualiza la Base de datos

Flujo normal de eventos

Sección “Visualizar proveedores”

Acción del Actor	Respuesta del Sistema
2.1 El líder de proyecto selecciona en el sistema la opción de visualizar proveedor desde la interfaz Listar Proveedores .	2.2. El sistema busca en la base de datos el proveedor seleccionado y los muestra en un formulario.

Prototipo de Interfaz

Vista general del proveedor

Id	31
Nombre	Gsdfg
Tipo	Interno
Correo	sdf@s.cu
Fax	
Descripción	
Representante	Gsdf
Dir Url	
Teléfono	4
Contratos	Crear un contrato para este proveedor contrato de prueba
Calle	Fas
Número	34
Código Postal	423
Municipio	
Provincia	
País	Saf
Productos	gsddryt

Nuevo
 Modificar
 Borrar

Flujos Alternos	
Acción del Actor	Respuesta del Sistema
Prototipo de Interfaz	
Poscondiciones	Se visualizan los datos de un proveedor específico.

2.9 Conclusiones parciales

En este capítulo se llevó a cabo una representación del modelo del negocio para garantizar un mejor entendimiento del dominio del problema en cuestión. Se definieron ocho requisitos funcionales y ocho no funcionales del sistema, para dar respuestas a las necesidades del cliente, además se definieron los casos de uso, seis casos de uso del negocio y seis del sistema, representados mediante los diagramas de casos de uso del negocio y sistema respectivamente. Se mostró además una descripción de los casos de uso, sus actores y la descripción de sus responsabilidades.

Capítulo 3 Diseño del sistema.

En este capítulo se pone de manifiesto cómo se utilizaron los patrones de diseño, se hace referencia a todo lo relacionado con traducir los requisitos a una descripción de cómo desarrollar el sistema, o sea se evidenciarán los artefactos generados correspondientes al flujo de trabajo en cuestión para la solución desde el punto de vista informático. Se muestra el modelado visual de su desarrollo y se presenta además la arquitectura, así como una adaptación del diseño.

3.1 Patrones de diseño.⁵

“Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.” (16)

3.1.1 Patrones de arquitectura.

Los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Estos proveen un conjunto de subsistemas predefinidos, especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Ayudan a especificar la estructura fundamental de una aplicación. (17)

El presente trabajo utilizó el patrón Modelo-Vista-Controlador (MVC) el cual es un patrón de arquitectura de software en el que todo el proceso está dividido en 3 capas; el Modelo, la Vista y el Controlador.

El **Modelo** incorpora la capa del dominio y persistencia, es la capa encargada de guardar los datos en un medio persistente (ya sea una base de datos, un archivo de texto o XML. En el modelo es donde se hace el levantamiento de todos los objetos que el sistema debe utilizar, es el proveedor de recursos, contiene los componentes que representan y gestionan los datos manejados por la aplicación. En el caso más típico, los objetos encargados de leer y escribir en la base de datos.

La **Vista** se encarga de presentar la interfaz al usuario, en las cuales se deben hacer operaciones simples. Los componentes de esta capa son responsables de mostrar al usuario el estado actual del modelo de datos, y presentarle las distintas acciones disponibles.

⁵Cito en <http://web2development.blogspot.com/2007/05/patron-mvc.htm>

El **Controlador** es el que escucha los cambios en la vista y se los envía al modelo, el cual le regresa los datos a la vista, es un ciclo donde cada acción de usuario causa que se inicie un nuevo ciclo. La forma más sencilla de implementar este patrón es pensando en capas, como regla, los accesos a la base de datos se hacen en el modelo, la vista y el controlador no deben de saber si se usa o no una base de datos. El controlador es el que decide que vista se debe de imprimir y que información es la que se envía. Además la capa de control es la que gestiona la lógica de los casos de uso.

3.1.2 Patrones GRASP

“Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.” (12)

Creador.

Este patrón se basa en asignarle la responsabilidad a una clase B de crear instancias de A en una de las siguientes condiciones:

- B agrega los objetos A.
- B contiene los objetos A.
- B registra las instancias de los objetos A.
- B utiliza específicamente los objetos A.
- B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado (así que B es un Experto respecto a la creación de A).

“La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El diseño, bien asignado, puede soportar un bajo acoplamiento, una mayor claridad y la reutilizabilidad”. La principal intención de este patrón es encontrar una clase que necesite crear una instancia de un objeto en alguna situación, si cumple las condiciones antes expuestas.

“El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear la cosa contenida o registrada”. (12)

En la aplicación este patrón se pone de manifiesto, puesto que la clase Proveedor, la cual contiene productos, es la que tiene la responsabilidad de crear instancias de Producto.

Experto en información.

Este patrón responde a la interrogante que se cita: ¿Quién debería ser el responsable de conocer la información?

“Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. Si las asignaciones de responsabilidad se hacen en forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones.” (12)

Este patrón se pone de manifiesto en la aplicación mediante las clases entidades las cuales son las encargadas de conocer su información y no las de otras clases; así en la clase Proveedor solo se manejan datos referentes a los proveedores y en las demás: Ofertas, Productos, TiposProductos los datos que ellas engloban por sí solas. Las clases entidades (*Domain Classes*) son las que tienen la responsabilidad de conocer la información propia y no otra. En la aplicación, un Proveedor es el encargado de conocer su capacidad de envío, para lo cual debe hacer uso de la cantidad de productos con que cuenta, valor que está contenido en la clase entidad Proveedor_productos la cual es también un experto en información puesto que esta y no otra es la que conoce la cantidad de productos de un proveedor.

Alta cohesión.

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Una baja cohesión hace muchas cosas no afines o realiza trabajo excesivo. (12)

La alta cohesión maneja el problema: ¿Cómo mantener la complejidad dentro de los límites manejables?, planteando como solución: Realizar las acciones correspondientes a una clase determinada en dicha clase únicamente, es decir que cada clase realice una labor única no realizada por ninguna otra.

En resumen, se observa cuando una clase tiene la responsabilidad de realizar una labor dentro del sistema, no desempeñada por el resto de los componentes del diseño. Este patrón se evidencia en conjunto con el patrón bajo acoplamiento, de forma tal que cada clase realiza sus acciones y se evita que otra clase realice acciones correspondientes a la clase con la que está relacionada.

Bajo Acoplamiento.

“El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce o con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de

muchas otras.

Este patrón responde la interrogante: ¿Cómo mantener un bajo acoplamiento entre las clases?

Plantea como solución: Para mantener un bajo acoplamiento entre clases se debe garantizar la alta cohesión, con el fin de que las clases no dependan en gran medida unas de las otras (que cada clase realice sus labores y más ninguna otra). De esta forma las clases se vuelven más fáciles de comprender, mantener y reutilizar.

De manera contraria una clase con alto o fuerte acoplamiento recurre a muchas clases. Este tipo de clases no es conveniente: presentan los siguientes problemas:

- Los cambios de las clases afines ocasionan cambios locales.
- Son más difíciles de entender cuando están aisladas.
- Son más difíciles de reutilizar porque requieren la presencia de otras clases de las que dependen.” (12)

En resumen, en la aplicación se hace uso de este patrón cuando las clases entidades manejan todas las operaciones referentes a ellas por sí sola y las controladoras de cada entidad solo hacen uso de estas operaciones, cosa que; de haber un cambio en el diseño de una entidad no implique cambios en la controladora, y de implicar cambios estos sean mínimos. Este patrón se traduce a tener pocas dependencias entre las clases.

Patrón Controlador.

Responde la interrogante: ¿Quién debería encargarse de atender un evento del sistema?

Plantea como solución: Asignar la responsabilidad del manejo de una serie de eventos determinados a un controlador. En la aplicación, el framework utilizado denomina estos controladores con la estructura [Nombre_Entidad]Controller.groovy. En el sistema cada controlador es el encargado de manejar los eventos y la lógica del negocio, relacionados a la clase entidad que maneja.

3.2 Arquitectura.

“La arquitectura del software proporciona una visión global del sistema a construir. Describe la estructura y la organización de los componentes del software, sus propiedades y las conexiones entre ellos. Los componentes del software incluyen módulos de programas y varias representaciones de datos que son manipulados por el programa. Además, el diseño de datos es una parte integral para la derivación de la arquitectura del software. La arquitectura marca decisiones de diseño tempranas y

proporciona el mecanismo para evaluar los beneficios de las estructuras de sistema alternativas.” (12) La arquitectura no es un software operacional, sino es la representación que provee al ingeniero de software analizar la efectividad del diseño para la consecución de los requisitos fijados, además permite considerar las diferentes variantes arquitectónicas y ayuda a disminuir los posibles riesgos que pudiera correr el desarrollo de software.

3.2.1 Arquitectura utilizada.

La arquitectura que se utilizó para la realización de la aplicación fue la arquitectura Modelo Vista Controlador (MVC). Diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

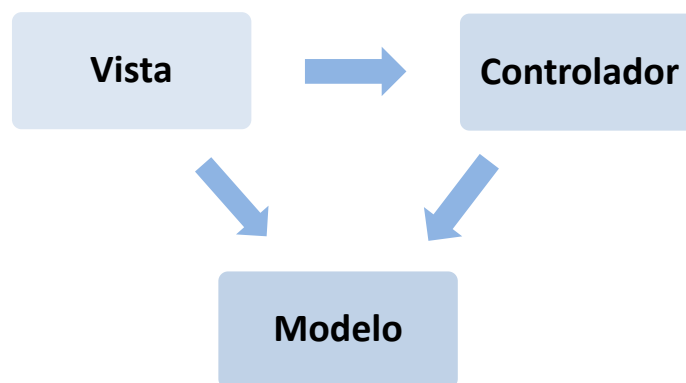


Figura 6 Arquitectura modelo-vista-controlador.

3.3 Modelo del diseño.

“El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar. Sirve de abstracción de la implementación y es utilizada como entrada fundamental de las actividades de implementación.”⁶

⁶Jacobson,Booch, Grady y Rumbaugh 2000.”El proceso unificado de desarrollo de software”

Para realizar los Diagramas de clases del diseño se utilizó el patrón Modelo Vista Controlador (MVC) puesto que el framework utilizado (Grails) utiliza este patrón; permitiendo la separación de la vista de la aplicación de las clases del modelo, usando una capa controladora que es la encargada de realizar los cambios en dicha vista.

3.4 Diagramas de Clases del Diseño.

“El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces (las de Java, por ejemplo) en una aplicación. Normalmente contiene la información: (12)

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.
- Dependencias

En el modelado de la aplicación se representa el subsistema “Componentes Grails” que representa los componentes agregados por el framework Grails, además del objeto relacional llamado GORM (*Grails Object Relational Mapping*). GORM es un gestor de persistencia escrito en Groovy, para controlar el ciclo de vida de las entidades y proporcionar una serie de métodos dinámicos (se crean en tiempo de ejecución para cada entidad) que facilitan enormemente las búsquedas. “GORM está construido sobre Hibernate, una herramienta de mapeo objeto-relacional, que se encarga de relacionar las entidades de las clase con tablas de una base de datos, y las propiedades de las entidades con campos en las tablas. Cada operación que se realice sobre los objetos del modelo de datos será traducida por Hibernate en las sentencias SQL necesarias para quedar reflejado en la base de datos.”⁷

⁷Nacho Brito 2009 “Manual de desarrollo web con Grails”.

A continuación se representan algunos diagramas de clases de diseño de la aplicación:

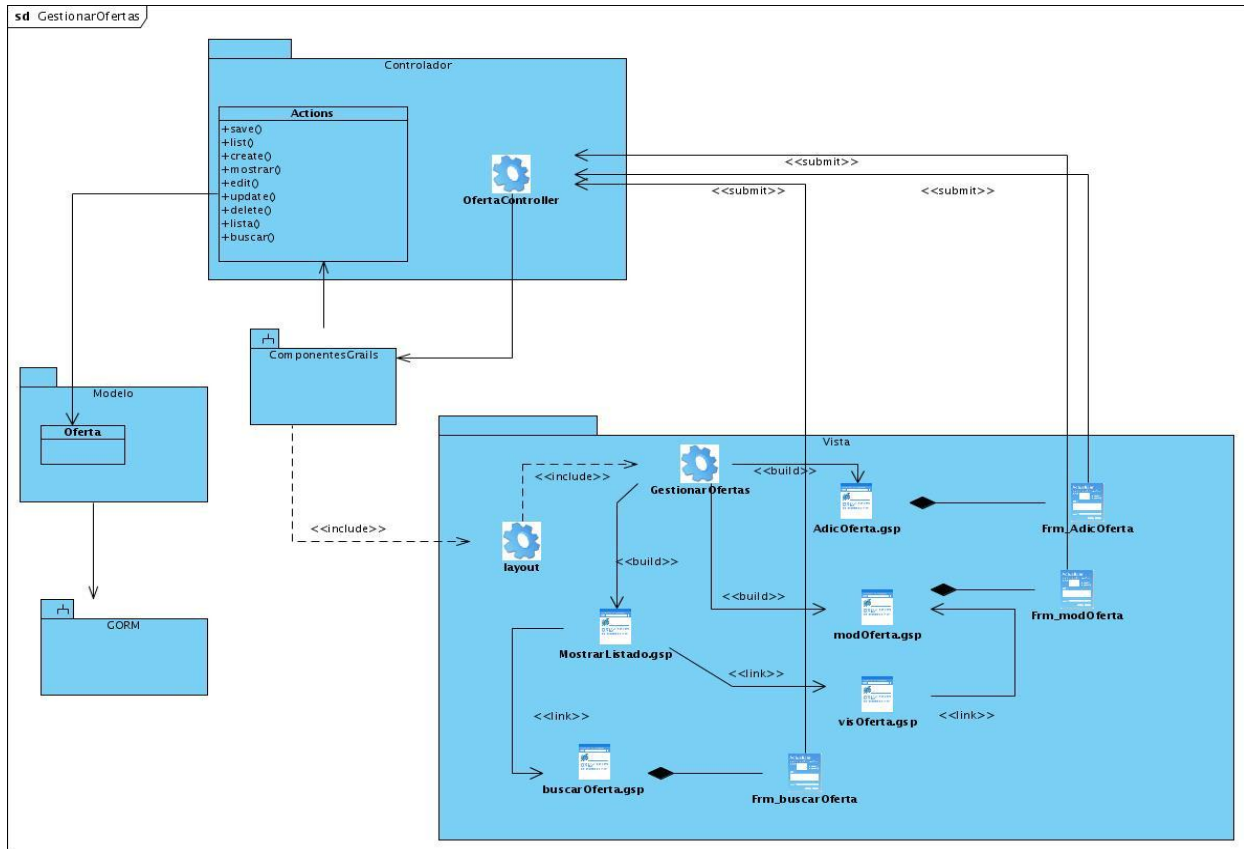


Figura 7 Diagrama de Clases del Diseño “Gestionar Ofertas”.

3.5 Diagramas de interacción.

Un diagrama de interacción explica gráficamente las interacciones existentes entre las instancias (y las clases) del modelo de estas. “Según UML se definen dos tipos de estos diagramas:

1. Diagramas de colaboración.
2. Diagrama de secuencia.”

Los diagramas de colaboración describen las interacciones entre los objetos. El cual se modela para cada método de la clase, además contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y los mensajes pasados entre los objetos. (12)

A continuación se presentan algunos diagramas de colaboración elaborados:

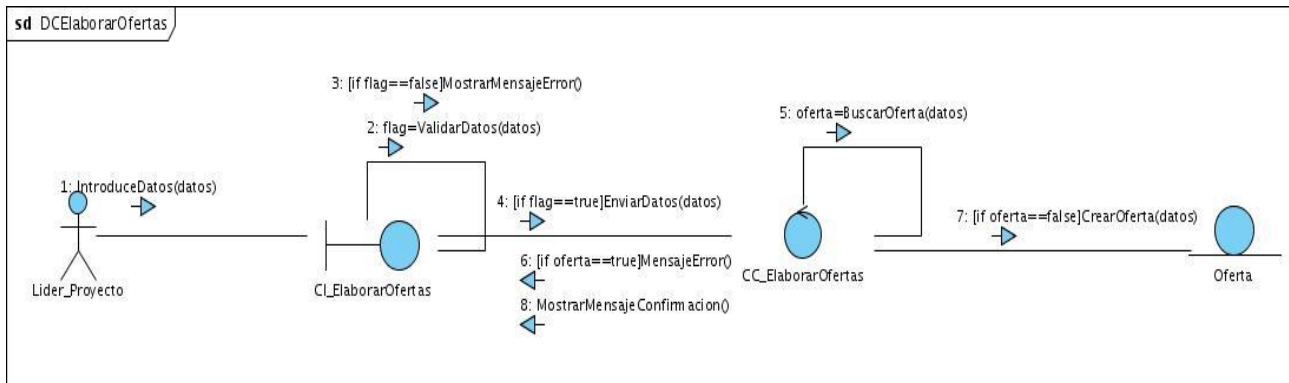


Figura 8 Diagrama de colaboración "Elaborar Ofertas".

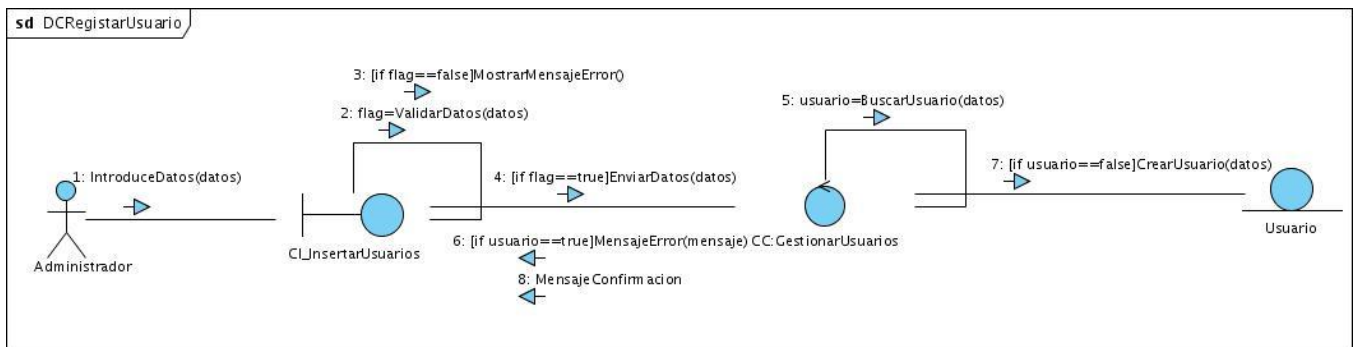


Figura 9 Diagrama de colaboración "Registrar Usuario".

3.6 Concepción del Mapa de Navegación.

Los mapas de navegación proporcionan una representación esquemática de la estructura del hipertexto, indicando los principales conceptos incluidos en el espacio de la información y las interrelaciones que existen entre ellos. La principal importancia del mapa de navegación es aportar al usuario una referencia a la cual acudir cuando en la página principal no ha encontrado la información que está buscando.

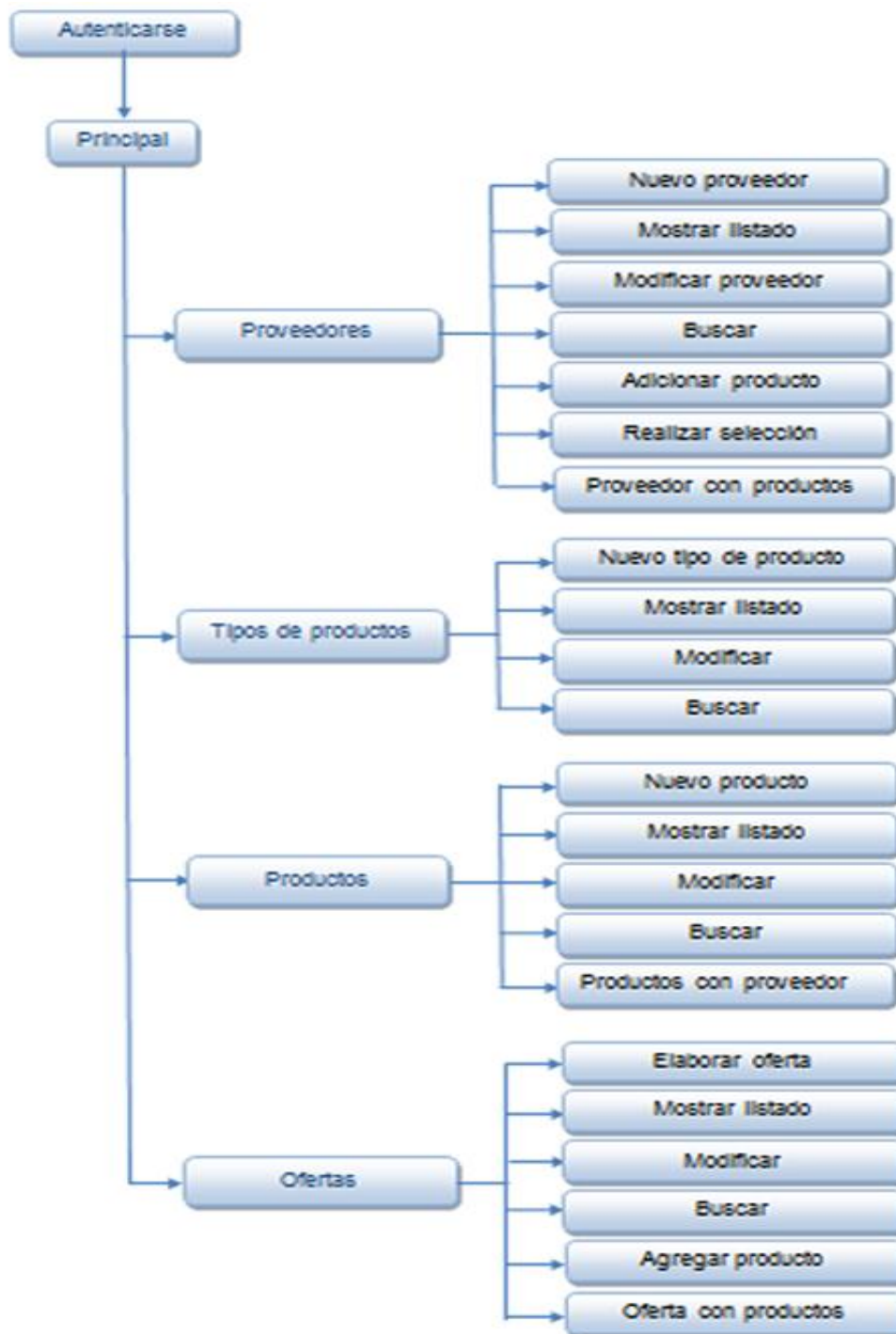


Figura 10 Mapa de navegación.

3.7 Diagrama de clases persistentes.

La persistencia de los datos es la capacidad que tienen de mantener su valor en el espacio y en el

tiempo. Las clases que deben ser persistentes se definen durante el diseño, responsabilidad del diseñador. Por lo general las clases persistentes tienen como origen las clases clasificadas como entidad porque ellas modelan la información del sistema y el comportamiento asociado de algún fenómeno o concepto. El diagrama de clases persistentes describe la estructura del sistema mostrando clases, atributos y las relaciones entre ellos.

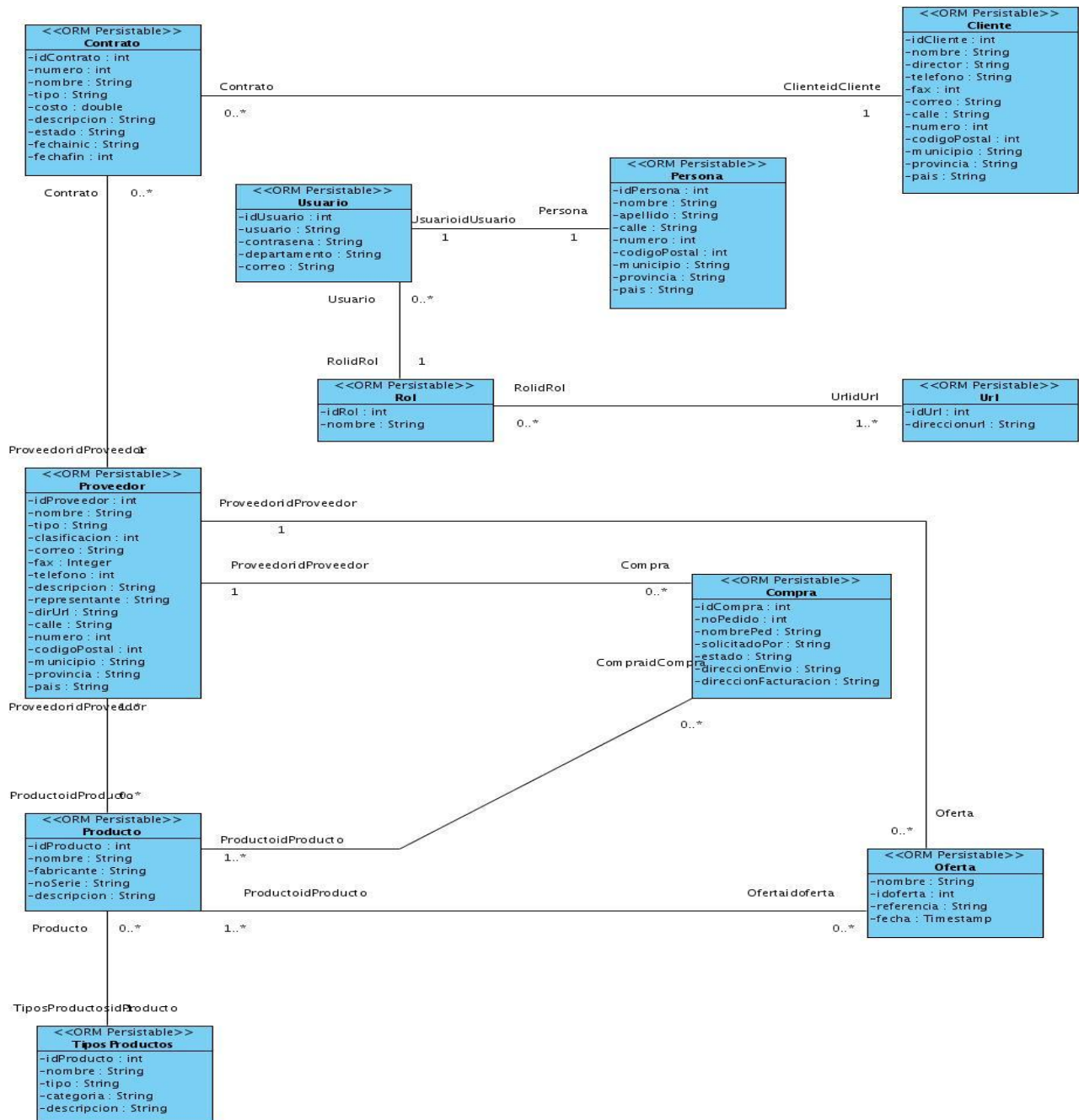


Figura 11 Diagrama de Clases Persistentes.

3.8 Modelo de Datos.

Un modelo de datos es básicamente... “un conjunto de conceptos, reglas y convenciones que nos permiten describir y en ocasiones manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos” (18).

3.8.1 Diagrama Relacional.

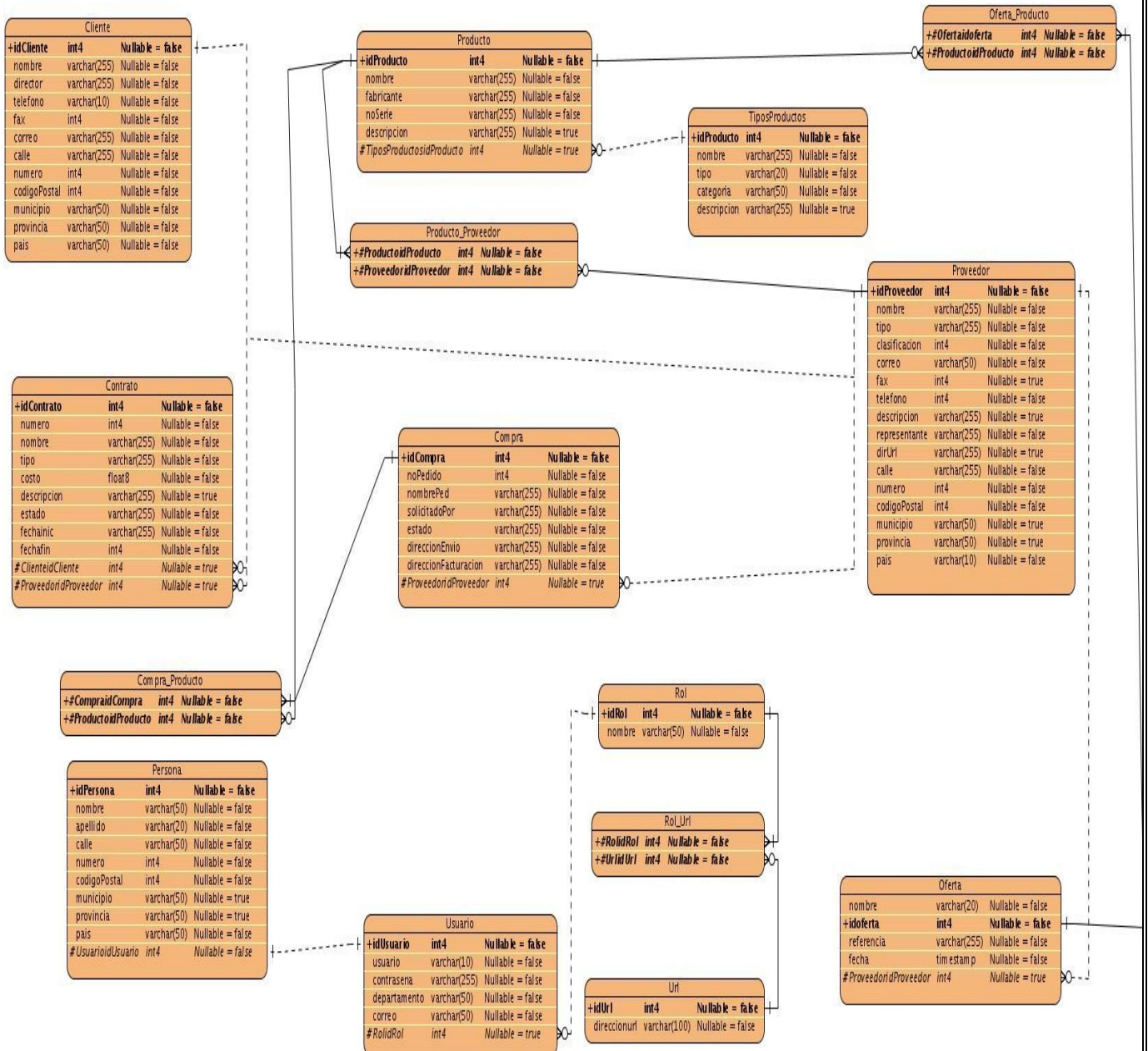


Figura 12 Diagrama Relacional.

3.9 Diagrama de despliegue.

Un diagrama de despliegue es un tipo de diagrama de UML que se utiliza para modelar el hardware utilizado en las implementaciones de sistemas y las relaciones entre sus componentes.

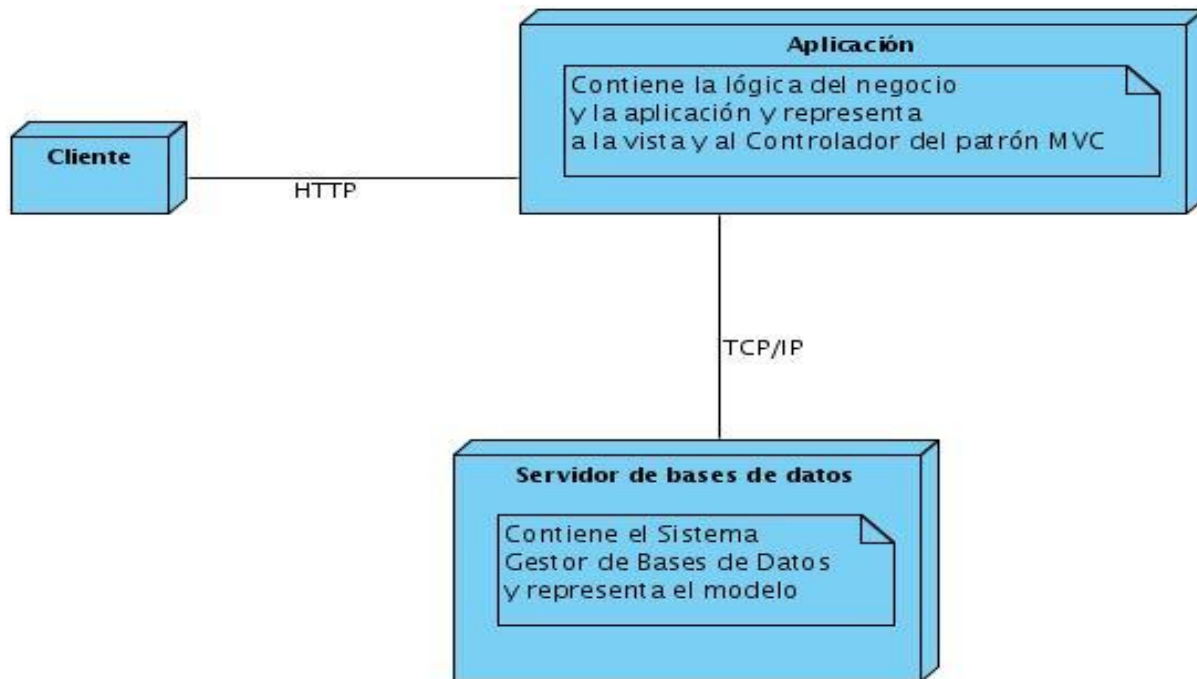


Figura 13 Diagrama de Despliegue.

3.10 Conclusiones parciales.

En el capítulo se abordó lo relacionado al diseño del sistema, para lo cual se identificaron los patrones de diseño utilizados; patrón de arquitectura MVC y los patrones GRASP (patrones de asignación de responsabilidades), fundamentales en el desarrollo del sistema. Se representaron los diagramas de clases del diseño, los diagramas de colaboración de los casos de uso más significativos, se definió la arquitectura del sistema y se representó el mapa de navegación.

Capítulo 4 Implementación y prueba.

En el presente capítulo se presenta el modelo de implementación, especificando las clases y objetos utilizados en la implementación como componentes y cómo dependen unos de otros, para lograr una mejor idea o entendimiento de las mismas. Además se especifican los casos de prueba realizados sobre la aplicación.

4.1 Modelo de Implementación.

“El Modelo de implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe como los elementos del diseño se implementan en componentes. Se considera el artefacto más significativo del flujo de trabajo de implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Este modelo está conformado por el diagrama de componentes.” (10)

4.1.1 Diagrama de Componentes.

“Un diagrama de componentes representa las dependencias entre componentes de software, incluyendo componentes de código abierto, componentes de código binario y componentes ejecutables. Un módulo de software se puede representar como componente. Algunos componentes existen en tiempo de compilación, algunos en tiempo de enlace y algunos en tiempo de ejecución, otros en varias de estas.” (12)

Los diagramas de componentes expuestos a continuación han sido definidos por casos de uso, para que estos sean más comprensibles, según el patrón arquitectónico Modelo-Vista-Controlador que usa Grails para crear una estructura organizada en tres capas.

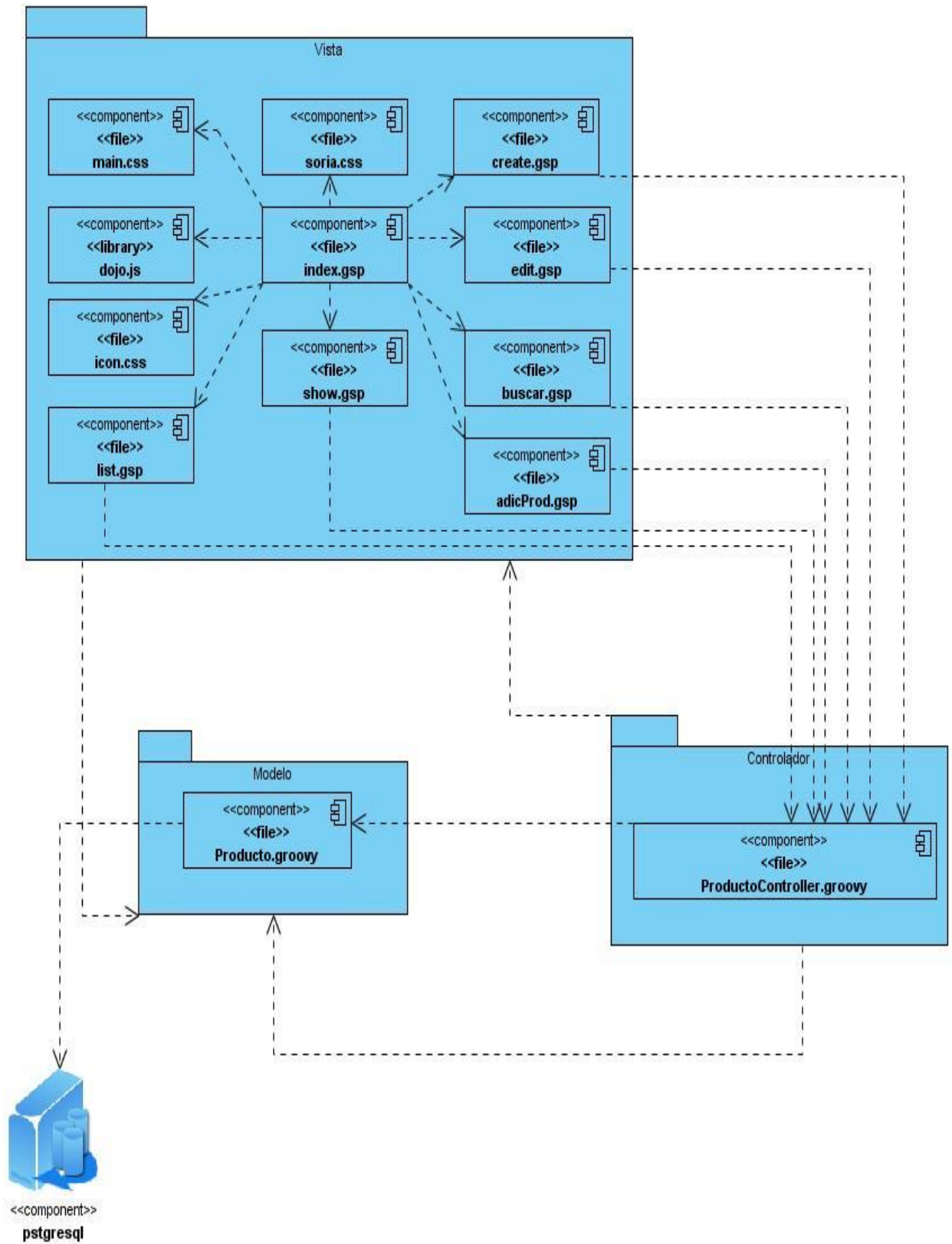


Figura 14 Diagrama de componentes de Productos.

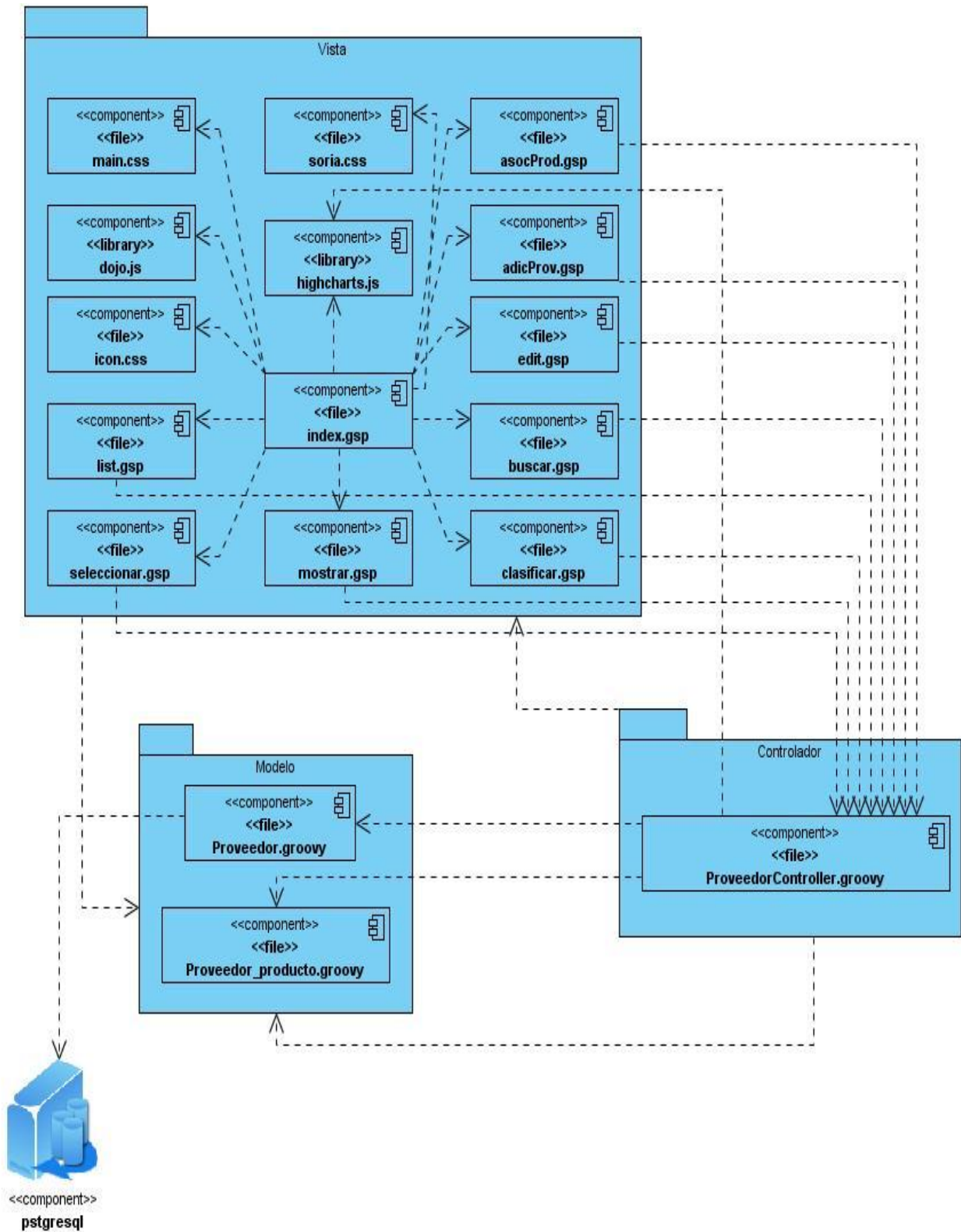


Figura 15 Diagrama de componentes de Proveedores.

4.2 Código fuente.

“En inglés (*Source Code*). El Código Fuente es un conjunto de líneas que conforman un bloque de texto, escrito según las reglas sintácticas de algún lenguaje de programación destinado a ser legible por humanos.

Es un programa en su forma original, tal y como fue escrito por el programador, no es ejecutable directamente por el computador, debe convertirse en lenguaje de maquina mediante compiladores, ensambladores o intérpretes.” (19)

4.2.2 Ejemplo de código fuente.

La gran mayoría de los métodos implementados en el desarrollo de la aplicación se basan en funciones básicas de registro, modificación, actualización y eliminación de los datos, debido fundamentalmente a las facilidades de desarrollo que brinda Grails para realizar el mapeo con la base de datos.

A continuación se presentan fragmentos de código y se ofrecen breves descripciones de estos.

```
def busqueda = {
    if(params.nombre==" &&params.representante==" &&params.tipo==" &&params.correo==" &&
params.pais==""){
        flash.message = "Debe llenar al menos un campo para la búsqueda."
        render(view:'buscar')
    }
    def locurrencias = Proveedor.withCriteria {
        or {
            ilike("nombre","${params.nombre}")
            ilike("representante","${params.representante}")
            ilike("tipo","${params.tipo}")
            ilike("correo","${params.correo}")
            ilike("pais","${params.pais}")
        }
    }
    render(view:"buscar",model:[locurrencias:locurrencias])
}
```

El método expuesto anteriormente presenta una forma de construir consultas a bases de datos a través de una sintaxis avanzada que se basa en el Builder de Groovy y la Criteria Api de Hibernate, muy usado en el framework (Grails), para hacer búsquedas avanzadas que incluyen muchos campos y comparaciones complejas.

```
def delete = {
    def proveedorInstance = Proveedor.get(params.id)
    if (proveedorInstance) {
        try {
            proveedorInstance.delete(flush: true)
            flash.message = "${message(code: 'default.deleted.message', args: [message(code:
'proveedor.label', default: 'Proveedor'), params.id])}"
            redirect(action: "list")
        }
        catch (org.springframework.dao.DataIntegrityViolationException e) {
            flash.message = "${message(code: 'default.not.deleted.message', args: [message(code:
'proveedor.label', default: 'Proveedor'), params.id])}"
            redirect(action: "mostrar", id: params.id)
        }
    }
    else {
        flash.message = "${message(code: 'default.not.found.message', args: [message(code:
'proveedor.label', default: 'Proveedor'), params.id])}"
        redirect(action: "list")
    }
}
```

El código expuesto antes representa un ejemplo de funcionalidades que brinda el framework usado en cuanto a funcionalidades básicas se refiere, una sintaxis de fácil comprensión, donde se observa el tradicional tratamiento de excepciones de Java mediante las cláusulas try-catch, el método que lo sigue representa como se asocian productos a un proveedor existente, además se trata una de las excepciones un tanto conocidas por los programadores de (Java) (NumberFormatException) cuando no puede realizar la conversión de determinados valores de entrada representados por los parámetros que vienen en la solicitud que se hace a la controladora, además se observan otros métodos nativos

del framework:

```
1 def proveedor_productosInstance = new Proveedor_productos(params)
2 def pr = proveedor_productosInstance.getProveedor ()
3 def pd = proveedor_productosInstance.getProducto ()
4 def p = Proveedor_productos.findByProveedorAndProducto (pr, pd)
```

La línea 1 representa la creación de un objeto de tipo `Proveedor_productos`, resultado de la relación entre las entidades `Proveedor` y `Productos` que se gestionan, al cual se le pasan una serie de parámetros, los cuales tienen la estructura `[clave: valor]`, donde la clave es el nombre del campo del elemento HTML, con el mismo nombre de los atributos del objeto a crear, y el valor viene con el valor del elemento HTML.

Las líneas 2 y 3 representan métodos (`getEntidad`) que retorna la instancia de un objeto de una entidad de nuestra base de datos. Y en la línea 4 se presenta otro método muy usable, los llamados buscadores dinámicos que usa hasta dos propiedades de las clases entidades para realizar consultas simples a bases de datos, es decir expresiones Grails, que tendrán como resultado, la primero ocurrencia que cumpla con los dos valores que se pasan como parámetro, en este caso, la primera ocurrencia de una asociación entre un `Proveedor` y un `Producto`, en la tabla, resultado de la relación entre las entidades anteriores, es decir `proveedor_productos` representada con un objeto llamado `defproveedor_productosInstance`.

```
def asocProductos = {

    printlnparams.tiempoEntrega
    try {
        params.solicitud = params.solicitud.toFloat()
    }catch(java.lang.NumberFormatException e){
        flash.error = 'La propiedad "<b>solicitud</b>" debe ser un valor numérico.'
        redirect(action: "asocProd")
    }
    try {
        params.cantprod = params.cantprod.toInteger()
    }catch(java.lang.NumberFormatException e){
```

```
flash.error = 'La propiedad "<b>Cant productos</b>" debe ser un valor numérico'
redirect(action: "asocProd")
}
try {
  params.precio = params.precio.toFloat()
}catch(java.lang.NumberFormatException e){
  flash.error = 'La propiedad "<b>precio</b>" debe ser un valor numérico'
  redirect(action: "asocProd",controller:"proveedor")
}
def proveedor_productosInstance = new Proveedor_productos(params)
if (proveedor_productosInstance.validate()) {

def pr = proveedor_productosInstance.getProveedor()
def pd = proveedor_productosInstance.getProducto()
def p = Proveedor_productos.findByProveedorAndProducto(pr,pd)
if(! p){
  proveedor_productosInstance.save(flush: true)
  flash.message = "La asociación proveedor-producto se realizó con éxito."
  redirect(action: "asocProd", id: proveedor_productosInstance.id)
}else {
  def prov = Proveedor.list()
  def prod = Producto.list()
  flash.error = "Ya existe el proveedor seleccionado con ese producto"
  render(view: "asocProd", model: [prov:prov,prod:prod])
}
}
else {
  defprov = Proveedor.list()
  defprod = Producto.list()
  render(view: "asocProd", model: [proveedor_productosInstance:
proveedor_productosInstance,prov:prov,prod:prod])
}
}
```

4.3 Modelo de Prueba.

“En el flujo de trabajo de la prueba se verifica el resultado de la implementación...” (10). Mediante las pruebas el sistema es ejecutado bajo condiciones o requerimientos especificados, los resultados son observados, analizados y registrados, con lo cual se hace una evaluación de los aspectos del sistema.

4.3.1 Casos de prueba.

Los casos de prueba son definidos como “formas de probar el sistema, incluyendo la entrada o resultado con la que se ha de probar y las condiciones bajo las que ha de probarse”. (10)

“Un caso de prueba puede especificar cómo probar un caso de uso o un escenario específico de un caso de uso. Un caso de prueba de este tipo incluye la verificación del resultado de la interacción entre los actores y el sistema, que se verifican las precondiciones especificadas por el caso de uso y que se sigue la secuencia de acciones especificadas por el caso de uso. Los casos de prueba basados en casos de uso especifica típicamente una prueba del sistema como “**caja negra**”; una prueba del comportamiento observable externamente del sistema.” (19)

Tabla 6 Escenario registrar proveedores.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 "Registro exitoso"	Insertar los datos del proveedor correctamente	V Proveedor 1	V prov@emp.cu	V 10400	El sistema muestra un mensaje de confirmación	1 Se registra el usuario como líder de proyecto. 2 El líder de proyecto selecciona la opción de adicionar proveedor. 3 Introduce los datos correctamente.
		Variable 4	Variable 5	Variable 6		
		N/A http://diremp.cu	V Interno	V 078305620		
		Variable 7	Variable 8	Variable 9		
		N/A 85654215485	V Representante 1	V Descripción		
		Variable 10	Variable11	Variable 12		
		V Calle 1ra	V #120A	V Municipio-zona		
		Variable 13	Variable14	Variable 15		
		V Cuba	V 2	V 6		
		Variable 16	Variable17	Variable 18		
V 8	V 2	V Habana				

Tabla 7 Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Campo de texto	No	El campo acepta datos que contengan letras, números y caracteres especiales.
2	Correo electrónico	Campo de texto	No	El campo solo acepta
3	Código postal	Campo de texto	No	El código postal debe cumplir que sea solo números.
4	Dirección de url	Campo de texto	Si	Dirección url de la web válida
5	Tipo	Lista de selección	No	Los tipos de proveedores que pueden existir (Interno, Externo, Nacional, Internacional).
6	Teléfono	Campo de texto	No	Debe cumplir que sea un número telefónico válido, solo aceptarán números.
7	Fax	Campo de texto	Si	Debe cumplir que sea un fax válido, solo aceptarán números.
8	Representante	Campo de texto	No	El campo acepta datos que contengan letras, números y caracteres especiales.
9	Descripción	Campo de texto	Si	El campo acepta datos que contengan letras, números y caracteres especiales.
10	Calle	Campo de texto	No	El campo acepta datos que contengan letras, números y caracteres especiales.
11	Número	Campo de texto	No	El campo acepta datos que contengan letras, números y caracteres especiales.
12	Municipio	Campo de texto	Si	El campo acepta datos que contengan letras, números y caracteres especiales.
13	País	Campo de texto	No	El campo acepta datos que contengan letras, números y caracteres especiales.
14	Cantidad de clientes.	Lista de selección	No	El campo acepta los rangos de selección definidos para cantidades de clientes.

15	Ubicación geográfica.	Lista de selección	No	El campo acepta las ubicaciones geográficas definidas.
16	Forma de envío.	Campos de selección	No	Las formas de envío que tiene el proveedor del que se van a insertar los datos.
18	Provincia	Lista de selección	Si	El campo acepta los países que se listan.

4.3.2 Pruebas de caja negra.

Son pruebas de software que se realizan sobre la interfaz del mismo, por lo que los casos de pruebas, pretenden demostrar que las funciones definidas previamente como requisitos son operativas, que las entradas se aceptan de la forma adecuada y que se produce el resultado correcto, estas pruebas:

- Verifican las especificaciones funcionales y no consideran la estructura interna del programa.
- Es hecha sin el conocimiento interno del producto.
- No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

Las pruebas que se le realizaron al sistema, tanto a nivel de desarrollador como por parte del cliente, arrojaron una serie de resultados, dichos resultados permitieron tomar acciones para perfeccionar funcionalidades e interfaces del mismo. Al sistema se le realizó un total de cuatro casos de prueba por casos de uso.

Estas pruebas arrojaron el resultado esperado, de ellas resultaron 2 no conformidades las cuales fueron solucionadas y 2 solicitudes de cambio, las cuales reflejan un sistema completamente funcional y acorde a las necesidades del cliente.

Conclusiones parciales.

En el capítulo se describieron los aspectos referentes a la implementación del sistema. Se presentaron los diagramas de componentes de los casos de uso gestionar Ofertas, Productos y Proveedores respectivamente. Se representaron fragmentos de código de la implementación. Se presentaron además, las pruebas realizadas con sus resultados y se mostró un caso de prueba realizado mediante el uso de los modelos de casos de prueba por casos de uso, en los cuales han quedado descritas las

variables utilizadas para la realización de las mismas y se muestra de forma gráfica el resultado obtenido en la etapa de pruebas del sistema.

Conclusiones generales

En el Trabajo de Diploma se realizó un estudio del estado actual de los sistemas de gestión de proveedores, a partir de los cuales se tomaron referencias de sus principales enfoques en este campo. Se analizó el funcionamiento de los procesos de clasificación y selección de proveedores de la empresa ALBET. Después de realizados los estudios antes mencionados, se concluyó como necesaria una aplicación para gestionar la información de la empresa, referente a la selección, clasificación de los proveedores y la elaboración de las ofertas de la empresa. Se llevó a cabo el diseño de los módulos “Clasificación y Selección de proveedores” y “Elaboración de sus ofertas” a partir de los requerimientos planteados. Se realizó la implementación de estos módulos y se cumplió el principal objetivo del trabajo; la creación de un sistema informático para llevar el control y análisis de los datos de los Proveedores de ALBET.

Recomendaciones

Con el objetivo de mejorar y continuar la aplicación se plantean las siguientes recomendaciones:

- ✓ Realizar versiones posteriores donde se implementen nuevas funcionalidades.
- ✓ Integrar la aplicación a la plataforma de gestión de servicios.
- ✓ Integrar a la aplicación un módulo que permita la autenticación de un usuario mediante un LDAP.

Referencias bibliográficas

1. Osiatis. [En línea] [Citado el: 5 de 12 de 2010.] <http://itilv3.osiatis.es>.
2. Falgueras, Benet Campderrich. *Ingeniería de software*.
3. López, Angel. *Java la programación del futuro*.
4. Kenneth Barclay, John Savage. *Groovy Programming an Introduction for Java Developers*.
5. Pérez, Javier Eguíluz. *Introducción a JavaScript*.
6. —. *Introducción a AJAX*.
7. Brito, Nacho. *Manual de desarrollo web con grails*. 2009.
8. Almaer, Dion. *The Dojo Toolkit in Practice: An AJAX library for more than 'Prototype-ing'*.
9. Real academia española. *Diccionario de la lengua española*. [En línea] [Citado el: 11 de junio de 2011.] <http://www.rae.es/rae.html>.
10. Jacobson, Booch, Grady y Rumbaugh. *El proceso unificado de desarrollo de software*. 2000.
11. TM, Unified Modeling Language. [En línea] <http://www.uml.org>.
12. Larman, Craig. *UML y Patrones. Introducción al Análisis y Diseño Orientado a Objetos*. 2009.
13. Visual Paradigm Design Group. *visualparadigm*. [En línea] <http://www.visual-paradigm.com>.
14. Sitio Oficial de NetBeans. [En línea] [Citado el: 10 de 11 de 2010.] www.netbeans.org.
15. PostgreSQL, Comunidad de. Comunidad de PostgreSQL. [En línea] [Citado el: 12 de 12 de 2010.] <http://www.postgresql.org>.
16. Welicki, L. *Patrones y Antipatrones*. .
17. Web2development. [En línea] [Citado el: 20 de 2 de 2011.] <http://web2development.blogspot.com/2007/05/patron-mvc.html>.
18. [En línea] <http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/> .
19. Pergamino Virtual. *Pergamino Virtual*. [En línea] [Citado el: 2 de 6 de 2011.] http://www.pergaminovirtual.com.ar/definicion/Codigo_Fuente.html.

Bibliografía

1. Osiatis. [En línea] [Citado el: 5 de 12 de 2010.] <http://itilv3.osiatis.es>.
2. Falgueras, Benet Campderrich. Ingeniería de software.
3. López, Angel. Java la programación del futuro.
4. Kenneth Barclay, John Savage. Groovy Programming an Introduction for Java Developers.
5. Pérez, Javier Eguíluz. Introducción a JavaScript.
6. —. Introducción a AJAX.
7. Brito, Nacho. Manual de desarrollo web con grails. 2009.
8. Almaer, Dion. The Dojo Toolkit in Practice: An AJAX library for more than 'Prototype-ing'.
9. Real academia española. Diccionario de la lengua española. [En línea] [Citado el: 11 de junio de 2011.] <http://www.rae.es/rae.html>.
10. Jacobson, Booch, Grady y Rumbaugh. El proceso unificado de desarrollo de software. 2000.
11. TM, Unified Modeling Language. [En línea] <http://www.uml.org>.
12. Larman, Craig. UML y Patrones.Introducción al Análisis y Diseño Orientado a Objetos. 2009.
13. Visual Paradigm Design Group. visualparadigm. [En línea] <http://www.visual-paradigm.com>.
14. Sitio Oficial de NetBeans. [En línea] [Citado el: 10 de 11 de 2010.] www.netbeans.org.
15. PostgreSQL, Comunidad de. Comunidad de PostgreSQL. [En línea] [Citado el: 12 de 12 de 2010.] <http://www.postgresql.org>.
16. Welicki, L. Patrones y Antipatrones. .
17. Web2development. [En línea] [Citado el: 20 de 2 de 2011.] <http://web2development.blogspot.com/2007/05/patron-mvc.html>.
18. [En línea] <http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/> .
19. Pergamino Virtual. Pergamino Virtual. [En línea] [Citado el: 2 de 6 de 2011.] http://www.pergaminovirtual.com.ar/definicion/Codigo_Fuente.html.
20. Draper, Dave. Dojo concepts for Java Developers.
21. Addison Wesley Dojo Using the Dojo JavaScript Library to Build Ajax Applications. 2008.
22. Orallo, Enrique Hernández. El lenguaje Unificado de Modelado(UML).
23. Pressman. Ingeniería del Software. Un enfoque práctico. 2005.
24. Ingenieros de Software. [En línea] [Citado el: 25 de 1 de 2011.] <http://www.ingenierossoftware.com>.
25. Taylor, Sharon, Iqbal, Majid y Nieves, Michael. ITIL v3 - Libro 1 - Service Strategy. 2008.
26. Russell, Mathew A. O'Reilly Dojo The Definitive Guide. 2008.

27. Mora, Sergio Luján. Programación de aplicaciones web: historia, principios básicos y clientes web.
28. Sitio Oficial de Dojo Toolkit. [En línea] <http://dojotoolkit.org/>.
29. Murray, Alex Russell-Greg. Using the Dojo Toolkit to Develop AJAX-Enabled Java EE Web Applications.