UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

Módulos Contrato y Compra 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: Pablo Luis Abreu Fernández

Tutores: Ing. Niurka Martínez Durán

Ing. Yandry Alberto Terry

Junio 2011.

"Año del 53 Aniversario del Triunfo de la Revolución"

Declaración de autoría

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.

Pablo Luis Abreu Fernández	Ing. Niurka Martínez Durán	Ing. Yandry Alberto Terry
_____	_____	_____
Firma del Autor	Firma del Tutor	Firma del Tutor

Datos de contacto.

Síntesis del Autor: Pablo Luis Abreu Fernández.

Correo Electrónico: plabreu@estudiantes.uci.cu.

Síntesis del Tutor: Ing. Niurka Martínez Durán.

Profesión: Ingeniero en Ciencias Informáticas.

Años de graduado: 2008.

Correo Electrónico: nduran@uci.cu.

Síntesis del Tutor: Ing. Yandry Alberto Terry.

Profesión: Ingeniero en Ciencias Informáticas.

Años de graduado: 2007.

Correo Electrónico: yalberto@uci.cu.

Agradecimientos.

A mi mamá, por profesarme tanto amor y dedicación, por ser ejemplo de empeño y perseverancia, por estar siempre dispuesta cuando la he necesitado sin importar el sacrificio, pero por sobre todas las cosas por ser la madre que es.

A mi papá, por su apoyo y cariño, porque de él aprendí lo que es ser trabajador y comprometido, porque sin saberlo me ha enseñado muchas cosas que han ayudado a formar el hombre que soy hoy.

A mis abuelos queridos por ser mi refugio, por entregarme todo el amor que una persona puede necesitar, por formar en mí valores que no se borran. Por estar desde que tengo uso de razón, por darme lo que solo un abuelo puede dar.

A mi tía Alina, la mejor tía del mundo, por tenerme como su hijo, comprenderme y por ser para mí, amiga y cómplice en más de una ocasión.

A mi amigo, mi primo, mi hermano Riguito, por apoyarme siempre a pesar de todo, por estar siempre dispuesto, por no dudar y confiar en mí aunque sea en contra de las probabilidades, por ser mí escudo cuando lo he necesitado.

A mi tío Rigo por todo su amor y cariño, por ser y seguir siendo un guía, por velar por mí y ayudarme siempre.

A Adriana, por ser incondicional, por estar siempre sin importar situación, por brindarme todo su amor y apoyo.

A mi hermano Aramis, que siempre está cuando lo he necesitado, por ser el hermano que todos quieren y por estar siempre pendiente de mí.

A mi nueva familia, Berthica, Robertico, el abuelo Roberto y la abuela Nancy, por tenerme como otro hijo.

A los tutores Niurka y Yandry por estar siempre dispuestos, por hacer magia con su tiempo para ayudarnos y por ser tan preocupados.

Agradecimientos

A mi amigo y compañero de tesis Isael Galindo Corrales, por compartir conmigo todo este tiempo de desarrollo, por su ayuda y paciencia.

A mi amiga Acralis por estar siempre pendiente y por todo el tiempo que dedicó a este trabajo.

A mi amigo Marlon por ser mi otro hermano, por su confianza, su apoyo y su incondicional amistad, a su familia que es también la mía.

A ese grupo de amigos, que pusieron su tiempo a mi disposición con paciencia e interés.

Pero especialmente quiero agradecer a Javier de León Barral (jbarral) y a Luis Felipe Matos Marinas (felipao) quienes fueron nuestros "*camaroncitos duros"* en más de una ocasión sin tener en cuenta hora ni día.

A mis amigos de toda la vida, Marcos, Oscar, Odannes, Alex, Daniel, El tati, Albertico, Ojito, Raúl, Onelvis, Roberto Carlos por ser mi familia.

A mis amigos de la universidad, Leandro Carmenaty (leito), Alejandro Pérez Lara (alejo), Yasel Almenares, Rubier, Sergio, Abelito y Hector (osmany).

A mi amiga Mercedes (merci), por ayudarme tanto y por el tiempo que dedicó al desarrollo de este trabajo.

A todos ustedes y a los que no menciono aquí pero que saben que han sido parte importante de mi vida, muchas gracias…

Dedicatoria.

Al Comandante en Jefe Fidel Castro Ruz.

A mi mamá Ana Caridad Fernández Cabrera, a mi papá Jorge Luis Abreu Ávalos y a mis abuelos Ana María Cabrera Mendoza y Pablo Fernández Santos.

A mi familia y amigos.

A todos los que en un momento u otro lucharon, dieron su vida o pusieron su empeño para lograr lo que tenemos hoy.

Resumen.

Producto de la necesidad de convertirse en una empresa competitiva y de los vertiginosos avances tecnológicos en la actualidad empresarial, surge la necesidad en ALBET Ingeniería y Sistemas, de contar con un sistema para administrar la información generada de la relación con sus proveedores. De esta forma, la presente investigación realizó y documentó el desarrollo de un sistema informático para realizar la gestión de las compras y los contratos de la empresa.

Se determinó que la aplicación debía ser accesible vía web y que para realizar un proyecto de este tipo era muy conveniente tomar como metodología de desarrollo, RUP, por sus características y robustez. Se decidió que la aplicación sería desarrollada en lenguaje Groovy usando el framework Grails, por poseer estos una serie de características muy favorables al desarrollo del sistema. De la misma forma fueron usados en conjunto con Groovy, AJAX, CSS, las librerías Dojotoolkit, HTML5 y JavaScript para el desarrollo de una interfaz de usuario profesional y a la misma vez agradable a la vista.

Al término del desarrollo de la investigación se cuenta con un sistema para la administración de proveedores, funcional, con un elevado nivel de usabilidad, amigable al usuario, fácilmente adaptable a las necesidades de cualquier empresa y que brindaba las facilidades requeridas por sus usuarios finales.

Palabras Claves:

- ALBET.
- Gestión de Proveedores.
- Compra.
- Contrato.
- Groovy.
- Grails.

Abstract.

Proceeds from the need to become competitive and rapid technological advances in business today, the need arises in the company ALBET Ingeniería y Servicios, to have a system to manage the information generated by the relationship with theirs suppliers. In this way, the research conducted and documented the development of a computer system for managing purchase and contracts of the company.

It was determined that the application should be accessible by the web, and that for a project like this was very convenient to take as a development methodology, RUP, because its characteristics and robustness. It was decided that the implementation would be developed using the Groovy language, and the Grails framework, by having such a lot of features very favorable to the development of such systems. In the same way were used in conjunction with Groovy, AJAX, CSS, the Dojotoolkit libraries, HTML5 and JavaScript for the development of a professional user interface and at the same time pleasing to the eye.

At the development of the research end there had a supplier's management system, functional, with a high usability level, friendly to the user, easily adaptable to the needs of any company and provided the facilities required by final users.

Keywords:

- ALBET.
- Supplier.
- Management.
- Purchase.
- Contract.
- Groovy.
- Grails.

Índice.

Índice de figuras.

Índice de tablas.

Introducción.

Las empresas actuales están orientándose a realizar una acertada gestión de las actividades que llevan a cabo dentro de sí mismas, con el fin de mejorar su estatus en el mercado. Para lograr esto es imprescindible prestar atención a la gestión de los servicios. Como los prestados por otras instituciones especializadas, las cuales se pueden llamar proveedores. Siendo un proveedor la persona u organización que proporciona un producto o servicio, el mismo bien puede ser productor, distribuidor, vendedor de un determinado producto o bien un prestador de un servicios (1). De esta forma es de vital importancia que se mantenga bien documentado todo el proceso de relación con los proveedores.

En la presente investigación se presta especial atención a las actividades destinadas a la gestión de los contratos y las compras. La investigación se guía por las Librerías de Infraestructuras de Tecnologías Informáticas (ITIL V3). ITIL puede ser definido como un conjunto de buenas prácticas destinadas a crear avances en la gestión y provisión de servicios Tecnologías de la Informática (TI). Su objetivo es mejorar la calidad de los servicios TI ofrecidos, evitar los problemas asociados a los mismos y en caso de que estos ocurran, ofrecer un marco de actuación para que estos sean solucionados con el menor impacto y a la mayor brevedad posible. Sus orígenes 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), 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 ITIL (v3) que data del año 2007. (2)

Está comprobado que la administración o gestión de los proveedores reporta en tiempo y recursos monetarios a la empresa. Es importante conocer cuáles son los aspectos y características a tener en cuenta a la hora de aceptar un contrato o realizar una compra. La logística, o forma en que las empresas obtienen los productos y servicios necesarios para su funcionamiento, genera un alto por ciento de los gastos empresariales. Siendo esta una actividad imprescindible, se debe trabajar en la disminución de los costos que genera. La gestión de los contratos y las compras puede influir significativamente en esto. Muchas de las empresas que prestan servicios hoy en día, ya sean estos informáticos o no, comprueban que el realizar un análisis de los procesos de gestión de compras y contratos constituye parte importante

del proceso de reducción de costos. Esto puede parecer insignificante en ocasiones, pero a largo plazo puede reportar cuantiosos ahorros.

Cuba, ha llevado a cabo una serie de avances tecnológicos en diferentes campos, los productos y servicios informáticos así como la gestión empresarial, no son la excepción. En el país algunas empresas están comenzando a dar pasos en el campo de la gestión de proveedores. En estos momentos, existe un pensamiento que provoca que los esfuerzos estén orientados fuertemente a la calidad de los productos y servicios que se prestan. Pero esto, aunque de importancia para generar ingresos en una organización, no es el único factor que reporta beneficios a la misma. No solo son los productos y servicios, los que deben ser ejemplo de calidad sino también los procesos internos de las empresas. De forma que es muy importante la gestión de los proveedores. Aunque existen entidades que están realizando algunos de los procesos de la gestión de compras y contratos, muy pocas lo está haciendo con la profundidad requerida.

ALBET, es una empresa cubana, cuyo origen y desarrollo se vincula estrechamente a la Universidad de Ciencias Informáticas (UCI). Por ser dicha entidad a la cual pertenecen los derechos comerciales, de los productos y servicios que desarrolla la UCI. La misma actualmente interactúa con un gran número de proveedores de productos y servicios en cada solución integral que comercializa. Pero esto trae consigo una serie de situaciones que generan un flujo bastante acelerado de documentos. La información referente a las compras así como la información de los contratos se guarda en documentos digitales en las máquinas locales de los especialistas de la dirección comercial de la empresa. Esto puede traer como consecuencia que la información, que es crítica, en un momento determinado pueda verse comprometida. Alguna falla de seguridad o negligencia, puede hacer que la información sufra pérdidas o modificaciones. El almacenamiento en medios sin la seguridad requerida no resulta satisfactorio, en estos casos, ya que se está hablando de los registros y los contratos. Además, al estar esta información almacenada de este modo, se hacen engorrosas las acciones que toda empresa debe realizar con respecto a sus proveedores. Como realizar alguna u otra consulta a los registros de las compras y los contratos, se hace extremadamente dificultoso poder acceder a datos estadísticos de la relación de la empresa con sus suministradores. El Centro de Soporte de la empresa, cuyos especialistas son los encargados de realizar los trabajos con los registros de los contratos y las compras, presenta la imperiosa necesidad de

organizar las tareas que llevan a cabo en esta área. Es muy poco práctico el manejo actual de los datos que manejan. De lo expuesto previamente se puede identificar el siguiente **problema científico:** ¿cómo contribuir a minimizar el tiempo de búsqueda y generación de informes de los contratos y compras de la empresa ALBET?

Por lo tanto se puede declarar como **objeto de estudio** de la investigación; las aplicaciones o software para gestionar los registros de compras y contratos con proveedores, enmarcado en el **campo de acción:** aplicaciones web para realizar la gestión de los registros de las compras y los contratos en los sistemas de gestión de proveedores.

De la misma forma se ha identificado el siguiente **objetivo general:** desarrollar los módulos Contratos y Compra del subsistema de gestión de proveedores de la Plataforma de Gestión de Servicios del Centro de Soporte.

La investigación cuenta con los siguientes **objetivos específicos:**

Realizar el análisis de la situación actual de las aplicaciones web para realizar la gestión de los registros de las compras y los contratos, en los sistemas de gestión de proveedores.

Caracterizar los módulos Contrato y Compra del subsistema de Gestión de Proveedores.

Realizar el diseño de los módulos Contrato y Compra del subsistema de Gestión de Proveedores.

Implementar los módulos Contrato y Compra del subsistema de Gestión de Proveedores.

Para la realización exitosa de la investigación se han de llevar a cabo las siguientes **tareas de la investigación:**

- Estudio y análisis de los principales sistemas de gestión de proveedores actuales.
- Entrevistas con los especialistas de ALBET para la descripción del proceso de gestión de proveedores.
- Análisis con los especialistas de ALBET de los módulos Contratos y Compra.

- Diseño de los módulos Contratos y Compra.
- Implementación de los módulos Contratos y Compra.
- Pruebas de Caja Negra a los módulos Contratos y Compra.

El presente trabajo de diploma está estructurado en 4 capítulos:

Capítulo 1. Fundamentación teórica: en el capítulo se presentarán algunos de los principales sistemas que presentan funcionalidades que tratan la gestión de contratos y el registro de proveedores junto con las limitantes que impiden que las mismas puedan ser utilizadas por el Centro de Soporte de ALBET. De la misma forma se presentarán las herramientas y metodologías a utilizar en la realización de esta investigación con sus respectivas características.

Capítulo 2. Características del sistema: en el capítulo se describirán los procesos actuales a través de un modelo de negocio, el cual sentará las bases para definir que se va a desarrollar y de qué forma.

Capítulo 3. Diseño de la solución: en el capítulo se definirán las funcionalidades del sistema propuesto y se describen en detalle.

Capítulo 4. Implementación y pruebas: en el capítulo se abordarán aspectos relacionados con el desarrollo de la solución propuesta, se plasman los principales componentes y diagramas empleados, así como una descripción de las principales pruebas realizadas al sistema con los resultados obtenidos.

Capítulo 1. Fundamentación teórica.

En el presente capítulo se trata la situación actual de la gestión de proveedores en lo que a aplicaciones y soluciones informáticas se refiere. También se brinda una serie de conceptos con los cuales será más comprensible la presente investigación. Se tratan aspectos de interés sobre el uso de las tecnologías y métodos utilizados para el desarrollo de la investigación. Además se hace referencia en el marco teórico a cada uno de los recursos y herramientas utilizadas.

1.1 Tendencias mundiales.

En el mercado mundial de los servicios las empresas consultoras y los especialistas en los temas de la gestión empresarial están continuamente buscando los métodos más factibles en lo que a optimización de procesos se refiere. La gestión de proveedores no es la excepción, por lo que se han definido diferentes estándares por parte de diversos autores y desarrolladores de aplicaciones sobre el tema.

1.2 Los sistemas de gestión de proveedores en la actualidad.

En el mundo las empresas han comprendido que una gestión interna es la clave, no solo para el ahorro de recursos monetarios sino también de tiempo. De la misma forma estos procesos garantizan la calidad de los productos y servicios prestados. Además se consigue una mayor fidelidad por parte de los clientes.

Son varias las aplicaciones informáticas que se dedican a la gestión de servicios que cuentan con módulos de gestión de proveedores. Es de vital importancia para una organización tener controlados y accesible sus datos. A continuación se hace referencia a algunos de estos sistemas.

1.2.1 Sistema ISIS ERP Manager, módulo de proveedores.

El sistema ISIS es una aplicación de gestión de servicios la cual cuenta con un módulo destinado a la gestión de los suministradores. En este módulo se efectúa el alta de los proveedores, cargando todos sus datos legales e impositivos en sus respectivas fichas.

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, emisión de certificados, trazabilidad total de

operaciones. El software permite configurar un completo circuito de autorizaciones para los pagos a sus proveedores.

Algunas de las funcionalidades que presta son:

- Archivo maestro de proveedores: se 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: se realizan 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 etc.
- Procesos: permite realizar ajustes a las cuentas corrientes, ingreso de comprobantes de cuenta corriente: facturas, notas de débito, notas de crédito, facturas de compra al contado, etc.
- Reportes: permite realizar los reportes de interés sobre los datos de los proveedores lo cuales pueden ser de importancia en un momento determinado, basándose en diferentes criterios.

Para este software existe un demo que se puede descargar libremente, pero no tiene presente todas las funcionalidades del sistema, es un software privativo y en el sitio que se promociona se puede adquirir mediante su compra.

1.2.2 Sistema para la gestión unificada 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, que 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. 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. Por otra parte este sistema 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.

El mencionado sistema es en realidad muy completo para algunas de las tareas relacionadas con la gestión de los proveedores. Sin embargo es un sistema privativo el cual no se puede adquirir libremente.

1.2.3 @GeSTOCK.

@GeSTOCK es una completa aplicación para administrar la información. Está compuesto por algunos módulos acoplados, como son: agenda, almacén, 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 muy simple que gestiona de cierto modo la información referente a las empresas. Pero con un inconveniente, no se ajusta a las características de las empresas dedicadas a prestar servicios informáticos.

Todos estos sistemas analizados tienen una característica en común, son propietarios, hasta los más sencillos, los que incluso fueron libres en determinado momento, como es el caso de @GeSTOCK, hoy están siendo privatizados.

En Cuba existen empresas utilizando este tipo de herramientas para la gestión de sus servicios, un ejemplo de estas empresas son: **Copextel, Etecsa, Softel, Desoft**, todas empresas de prestigio en el país, con un sistema bien organizado y una buena calidad de los servicios que prestan.

En la universidad no existe ningún sistema de este tipo que se pueda utilizar libremente. Se cuenta con algunos sistemas afines pero son privados. La situación que existe en el mundo con estos sistemas es complicada ya que cada organización crea sus propios sistemas con sus herramientas para realizar sus tareas de gestión. Esto hace difícil que estas aplicaciones puedan ser utilizadas de una empresa a otra por la singularidad presente en cada una de las mismas. Además en estas herramientas, están definidas las estrategias que cada empresa sigue con respecto a la gestión de los datos de sus suministradores. Por otra parte para las empresas no es factible compartir sus aplicaciones ya que estarían abriendo sus estrategias internas, cosa que no es para nada común en el mundo de la competencia empresarial. Esta es una de las razones por las que surge la idea por parte de los especialistas de ALBET de crear dentro de la plataforma de gestión de servicios un subsistema que gestione sus proveedores.

1.3 Tecnologías a usar en el desarrollo de la solución.

La empresa ALBET está desarrollando un sistema de gestión de servicios el cual está en parte concluido solo con la excepción de algunas funcionalidades entre ellas el Subsistema de Gestión de proveedores. Este sistema ha sido desarrollado por los especialistas de ALBET, por lo que su arquitecto definió de previo al comienzo de su desarrollo una serie de especificaciones y herramientas. El subsistema de Gestión de Proveedores, el cual es el objetivo de la presente investigación, es parte de un Sistema de Gestión de Servicios. Por tal razón algunos de los aspectos del desarrollo del Subsistema de Gestión de Proveedores van a estar sujetos a una serie de especificaciones definidas con anterioridad.

Independientemente de lo expuesto anteriormente, las tecnologías que se describen a continuación fueron seleccionadas para el desarrollo de la investigación por sus características.

1.3.1 Lenguaje de modelado.

Para el desarrollo de la solución propuesta se utiliza como lenguaje de modelado UML que es un lenguaje estándar para escribir planos de software. Usado para visualizar, especificar, construir y documentar los artefactos de un mismo sistema que involucra una gran cantidad de software. Su alfabeto está constituido por elementos y relaciones, los cuales al combinarse cobran sentido al armar una colección de palabras formando diferentes tipos de diagramas. Los elementos de UML se clasifican en estructurales (clases, interfaces. colaboraciones, casos de uso, clases activas, componentes y nodos), de comportamiento (interacciones y máquinas de estado), de agrupación (paquetes) y de anotación (notas). A su vez, hay cuatro tipos de relaciones: de dependencia, de asociación, de agrupación y de realización. Para construir un plano de software que tenga sentido, lo que se hace es combinar los elementos estructurales con sus respectivas relaciones, según sea el caso, obteniendo como resultado uno de los nueve diagramas que existen en UML: de clases, de objetos, de casos de uso, de secuencia, de colaboración, de estados, de actividades, de componentes y de despliegue.

1.3.2 Lenguajes de programación.

Dado a que el sistema debe ser accesible vía web y funcional, independientemente del sistema donde se encuentre alojado el servidor, se decidió que el lenguaje a utilizar, aparte de ser libre, debe ser multiplataforma. Además, interfaz de usuario debe ser sencilla pero agradable, funcional, con buena arquitectura de la información, donde los usuarios encuentren las funcionalidades necesarias fácilmente, lo cual es muy útil en este tipo de sistema. Para la conformación de la vista se hace necesario utilizar herramientas que permitan al sistema brindar cada una de las características antes mencionadas. A continuación se describen de forma breve los lenguajes a utilizar en el desarrollo de la aplicación.

Java.

Java fue desarrollado por *Sun Microsystems* en un momento cuando las páginas web eran estáticas. Java revolucionó la Internet convirtiendo las páginas en dinámicas e interactivas. Java es un lenguaje definido

por sus creadores como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portable, multitarea y dinámico. Java permite hacer cosas de interés como son los gráficos dinámicos los cuales proporcionan información visual que resulta de mayor comprensión e interés. (1)

Groovy.

Lenguaje ágil para plataformas Java, muy notable en el desarrollo de aplicaciones web orientado a objetos lo que resulta de mucha utilidad para el desarrollo del sistema.

Independiente de la plataforma: Java que es el lenguaje sobre el cual está basado Groovy, se compila a un formato de código de byte que puede ser leído e interpretado por muchas plataformas.

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: Java todo lo asume de forma paralela, con varias tareas de forma simultánea. Un programa sobre Java puede procesar diferentes tareas independientes, además de que permite la creación de aplicaciones distribuidas.

Este lenguaje 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.

Crear aplicaciones para servidores como foros en línea, tiendas, encuestas y procesamiento de formularios *Híper Text Markup Language* (HTML). (4)

JavaScript.

Lenguaje script utilizado para unir el conjunto de tecnologías usados en la web. JavaScript es un lenguaje de scripting basado en objetos, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza

integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. *JavaScript* se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. Todos los navegadores modernos interpretan el código *JavaScript* integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje *JavaScript* de una implementación del *Document Object Model* (DOM). Su principal importancia radica en que tradicionalmente, se venía utilizando en páginas web HTML, para realizar operaciones y en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML. (2)

1.3.3 Marco de trabajo (Framework).

Con el desarrollo de las aplicaciones informáticas y la necesidad de agilizar el proceso de desarrollo de las mismas, se han desarrollado marcos de trabajo o como se lo conoce por el término en inglés *(Framework)*. Los marcos de trabajos tienen como función fundamental el proveer al desarrollador de una serie de funcionalidades desarrolladas anteriormente no solo a nivel de código sino también a nivel organizativo. Estos marcos de trabajo, implementan de la misma forma, arquitecturas dependiendo del tipo de aplicación para las cuales han sido creados. Con el objetivo de contar con un marco de trabajo que se ajuste a la arquitectura que debe, al lenguaje a utilizar y a los requerimientos de interfaz, se han decidido la utilización de los que se describen más adelante.

Grails.

Grails es un marco de trabajo para aplicaciones web libre desarrollado 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, no repetir, proporcionando un entorno de desarrollo estandarizado y ocultando gran parte de los detalles de configuración al programador.

Grails ha sido impulsado principalmente por la empresa G2One, la cual fue adquirida por la desarrolladora de software libre *Spring Source* en noviembre del 2008. En agosto del 2009 *Spring Source* fue a su vez adquirida por *VMWare*, empresa especializada en virtualización de sistemas.

Grails fue conocido como 'Groovy on Rails' (el nombre cambió en respuesta al pedido de David Heinemeier Hansson, fundador de Ruby on Rails). Se inició en julio del 2005, con la versión 0.1 el 29 de marzo del 2006 y la versión 1.0 anunciada el 18 de febrero de 2008. En diciembre del 2009 se publicó la versión 1.2, y en mayo del 2010 la versión 1.3.

Características:

Grails se ha desarrollado con una serie de objetivos presentes:

- Ofrecer un framework web de alta productividad para la plataforma Java.
- Reutilizar tecnologías Java ya probadas como *Hibernate* y *Spring* bajo una interfaz simple y consistente.
- Ofrecer un framework consistente que reduzca la confusión y que sea fácil de aprender.
- Ofrecer documentación para las partes del framework relevantes para sus usuarios.
- Proporcionar lo que los usuarios necesitan en áreas que a menudo son complejas e inconsistentes.
- Framework de persistencia potente y consistente.
- Patrones de visualización potente y fácil de usar con Groovy *Server Pages* (GSP).
- Bibliotecas de etiquetas dinámicas para crear fácilmente componentes web.
- Buen soporte de Ajax que sea fácil de extender y personalizar.
- Proporcionar aplicaciones ejemplo que muestren la potencia del framework.
- Proporciona un entorno completo de desarrollo, incluyendo un servidor web y recarga automática de recursos.

Grails se ha diseñado para ser fácil de aprender, fácil para desarrollar aplicaciones y extensible. Intenta ofrecer el balance adecuado entre consistencia y funcionalidades potentes. (6)

Dojotoolkit.

Librería de clases JavaScript. Dojo es un marco de trabajo que contiene interfaces de programación aplicadas (API) y controles para facilitar el desarrollo de aplicaciones Web que utilicen tecnología AJAX. Contiene un sistema de empaquetado inteligente, abstracción de eventos, almacenamiento de Apis en el cliente, e interacción de Apis con AJAX. Resuelve asuntos de usabilidad que son generalmente comunes como pueden ser la navegación y detección del navegador,

soportar cambios de URL en la barra de URLs 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. (3)

1.3.4 Persistencia de los datos.

Dada la función del sistema, gestionar los datos de las compras y los contratos, es imprescindible que existan registros de estas operaciones. Para garantizar la persistencia de los datos se utilizará el servidor de bases de datos que se describe a continuación:

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 las necesidades. Es desarrollado por una comunidad de desarrolladores, entre sus características se encuentran:

- Estable.
- Alto rendimiento.
- Flexibilidad puesto que puede funcionar en la mayoría de los sistemas Unix, además de que puede ser integrado a ambiente de Windows permitiendo de esta manera a los desarrolladores, generar nuevas aplicaciones o mantener las ya existentes.
- Se puede extender su funcionalidad.

Gran compatibilidad: permite crear o migrar aplicaciones desde Access, Visual Basic, Visual Fox Pro, Delphi para usar PostgreSQL como servidor de base de datos. (5)

1.4 Metodología y software.

Todo proceso de desarrollo de una aplicación informática debe estar regido y orientado por una metodología de desarrollo de software, que guie los procesos y permita tener un registro detallado del avance de la investigación. Las metodologías pueden ser ágiles o robustas. Las metodologías ágiles están orientadas a la realización del software de una manera más objetiva, están generalmente orientadas al

desarrollo no haciendo mucho énfasis en la documentación y su principal objetivo es garantizar la agilidad en el proceso de desarrollo. Por su parte las metodologías robustas o pesadas están concebidas para guiar el proceso de desarrollo de los software de gran envergadura, cuando un proyecto requiere de gran cantidad de documentación por la complejidad de sus funcionalidades, cuando este proyecto va a ser realizado en un tiempo considerablemente largo y existe la posibilidad de que pase por las manos de varios equipos de trabajo, en efecto es necesario que exista una correcta documentación para que en un momento determinado los procesos de desarrollo puedan ser continuados por otras personas, entonces es que se debe utilizar una metodología robusta que garantice que todos los elementos estén en un orden perfectamente comprensible. Por lo que se va a utilizar *Rational Unified Process* (RUP) para el proceso de desarrollo de la solución. Para dar una idea de la metodología a utilizar, se describen a continuación sus principales características.

1.4.1 Metodología RUP.

La metodología RUP, llamada así por sus siglas en inglés *Rational Unified Process*, divide en 4 fases el desarrollo del software:

• **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 llevar a obtener la capacidad operacional inicial.

• **Transición**, el objetivo es llegar a obtener el despliegue del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

Disciplina de desarrollo.

La disciplina de desarrollo comprende, como su nombre lo dice los flujos de trabajo en los que se desarrolla el proyecto, estos flujos son:

• Modelado del negocio: entendiendo las necesidades del negocio.

• Requisitos: Traslada las necesidades del negocio a un sistema automatizado.

• Análisis y diseño: Trasladando los requerimientos dentro de la arquitectura de software.

• Implementación: creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

• Pruebas: asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de soporte.

La disciplina de soporte abarca los flujos de trabajo posteriores al desarrollo del proyecto, una vez que el proyecto ha sido probado se ejecutan los flujos que se listan a continuación:

• Configuración y administración del cambio: guarda todas las versiones del proyecto.

• Administrando el proyecto: administrando horarios y recursos.

• Ambiente: Administrando el ambiente de desarrollo.

• Distribución: hacer todo lo necesario para la salida del proyecto.

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entregable o en cada iteración.

Los elementos del RUP son:

• **Actividades,** son los procesos que se llegan a determinar en cada iteración.

• **Trabajadores**, vienen hacer las personas o entes involucrados en cada proceso.

• **Artefactos,** un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (10)

Herramientas.

La metodología RUP ha sido escogida por sus características, las cuales le son favorables al desarrollo de la solución propuesta por esta investigación. La mencionada metodología soporta una serie de herramientas para realizar las actividades referentes al proceso de desarrollo, a continuación se describen las herramientas con las que se va a desarrollar el sistema:

NetBeans 6.9.

NetBeans es un IDE de código abierto escrito completamente en Java usando la plataforma NetBeans. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. Cuenta con 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 Phyton y trae incluido entre sus novedades más destacadas el uso de Groovy.

Visual Paradigm.

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software. Permite crear todos los tipos de diagramas necesarios para demostrar los procesos de desarrollo. Además brinda una serie de facilidades por lo que ha sido seleccionado para realizar el modelado de los diferentes diagramas, algunas de las características se relacionan a continuación:

- Soporte de UML versión 2.1.
- Diagramas de Procesos de Negocio-Proceso, Decisión, Actor de Negocio, Documento.
- Modelado colaborativo con Subversion.
- Interoperabilidad con modelos UML2 (meta modelos UML 2.x para plataforma Eclipse) a través de XML.
- Ingeniería de ida y vuelta.
- Ingeniería inversa – Código a modelo, código a diagrama.
- Ingeniería inversa Java.
- Plataforma Java (Windows/Linux/Mac OS X). (11)

Conclusiones parciales.

En el capítulo se realizó un análisis de la situación actual de los sistemas encargados de gestionar la información relacionada con los contratos y las compras. Se han determinado una serie de elementos que dejan sentadas las bases para la realización de la investigación. También se han mostrado las herramientas con las que se va a desarrollar la solución (NetBeans 6.9 y Visual Paradigm), las cuales son en su totalidad de carácter libre no teniendo que estar sujeto el desarrollo a ningún tipo de presión legal. De la misma forma se han descrito los lenguajes a utilizar para el desarrollo del sistema, tanto el lenguaje de modelado (UML), como los lenguajes de programación (Java, JavaScript, Groovy y HTML). Además de describe ampliamente la metodología (RUP), por la cual se va a regir el proceso de desarrollo y los marcos de trabajo o Frameworks a utilizar (Grails y Dojotoolkit).

Capítulo 2. Características del sistema.

En el presente capítulo se describe el flujo actual de los procesos involucrados, lo que hará posible conocer las particularidades del marco donde se desarrollará, el objeto de automatización, las principales características de la solución a desarrollar y los elementos principales de su arquitectura, de la misma forma quedarán descritos los requisitos funcionales y no funcionales.

2.1 Flujo Actual de procesos.

A falta de un sistema que informatice las tareas de gestión de registros de contratos y compras, los especialistas de ALBET se ven en la tarea de realizarlas de forma manual. Como es llevado hasta el momento el flujo actual de procesos se describe a continuación.

2.1.1 Proceso de compra.

Para realizar el proceso de compra de la empresa, primeramente se realiza la asignación del presupuesto de compras para las direcciones de ALBET, para ello la responsable del Proceso Económico Financiero (Directora Económica) solicita a los responsables de los procesos de la empresa (Directores de la Empresa) el plan de presupuesto. Se realiza la propuesta referente al mismo a nivel de empresa. Los Directores de la empresa llenan los modelos que les son entregados para que realicen las solicitudes de sus respectivas áreas, las cuales son aprobadas por el Director General. Seguido a esto se procede a realizar la gestión del Contrato en caso que proceda.

2.1.2 Proceso de gestión de contrato.

Primeramente se le entregan al Director Jurídico una preforma del Contrato para que sea revisada y aprobada por él, para que el Director General haga la aprobación final del Contrato del Proveedor. Además de esto se procede a llenar una ficha de cliente a la cual se le adjuntan una serie de documentos. Cuando estos documentos son avalados por el Director Jurídico de la Empresa este realiza un Certifico de los documentos y son llevados al proveedor. El suministrador revisa los documentos, los aprueba y se entrega una copia del Contrato firmada por él, al Responsable del Proceso Legal de la Empresa.

2.3 Objeto de la automatización.

La empresa ALBET para realizar la compra realiza un conjunto de procesos que requieren ser automatizados, entre estos se encuentran la confección de un modelo de solicitud de compras el cual es entregado a los directores de la empresa para que realicen las solicitudes por áreas. Al ser llenado el mismo, por parte de los directores son enviados a la Dirección Administrativa para que sean aprobados por el Director General. Después se realiza la gestión del contrato en caso de que proceda. Al ser aprobada la compra se pasa a la revisión y aceptación de los contratos con el proveedor, por parte del Director Jurídico. Después de esto el contrato es aprobado por el Director General. La aprobación de la compra se realiza a través del Director Administrativo el cual presenta el contrato y la compra correspondiente al Comité de Aprobación para que este tome la decisión final.

2.4 Información que se maneja.

En el proceso se manejan una serie de documentos relacionados con los procesos de Compra como es el caso de los Contratos con los proveedores y las fichas de clientes que a su vez llevan asociados una serie de documentos como son:

- Resolución del nombramiento del Director.
- Licencia para operar en divisas.
- Objeto social de la empresa.
- Resolución de creación de la entidad.

2.5 Modelo de negocio.

2.5.1 Actores del negocio.

Actor del Negocio	Justificación
Proveedor	Realiza la revisión de los documentos referentes al contrato.
Directores de ALBET	Realizan la solicitud de Compra y confeccionan la ficha del cliente.

Director General	Se encarga de aprobar la solicitud de Compra y el Contrato
Director Jurídico	Se encarga de aprobar la preforma de Contrato y de avalar los documentos de la ficha del cliente.
CAD(Comisión de Aprobación de Divisa)	Se encarga de aprobar el Contrato y la Compra.

Tabla 1 Actores de negocio.

2.5.2 Trabajadores del negocio.

Trabajadores del Negocio	Justificación
Directores de ALBET	Realizan la solicitud de Compra y confeccionan la ficha del cliente.
Director General	Se encarga de Registrar la solicitud de Compra y el Contrato
Director Jurídico	Se encarga de Realizar el Certifico de los documentos presentados al proveedor.
CAD(Comisión de Aprobación de Divisa)	Se encarga de Registrar el Contrato y la Compra.

Tabla 2 Trabajadores del negocio.

2.5.3 Diagrama de caso de uso del negocio.

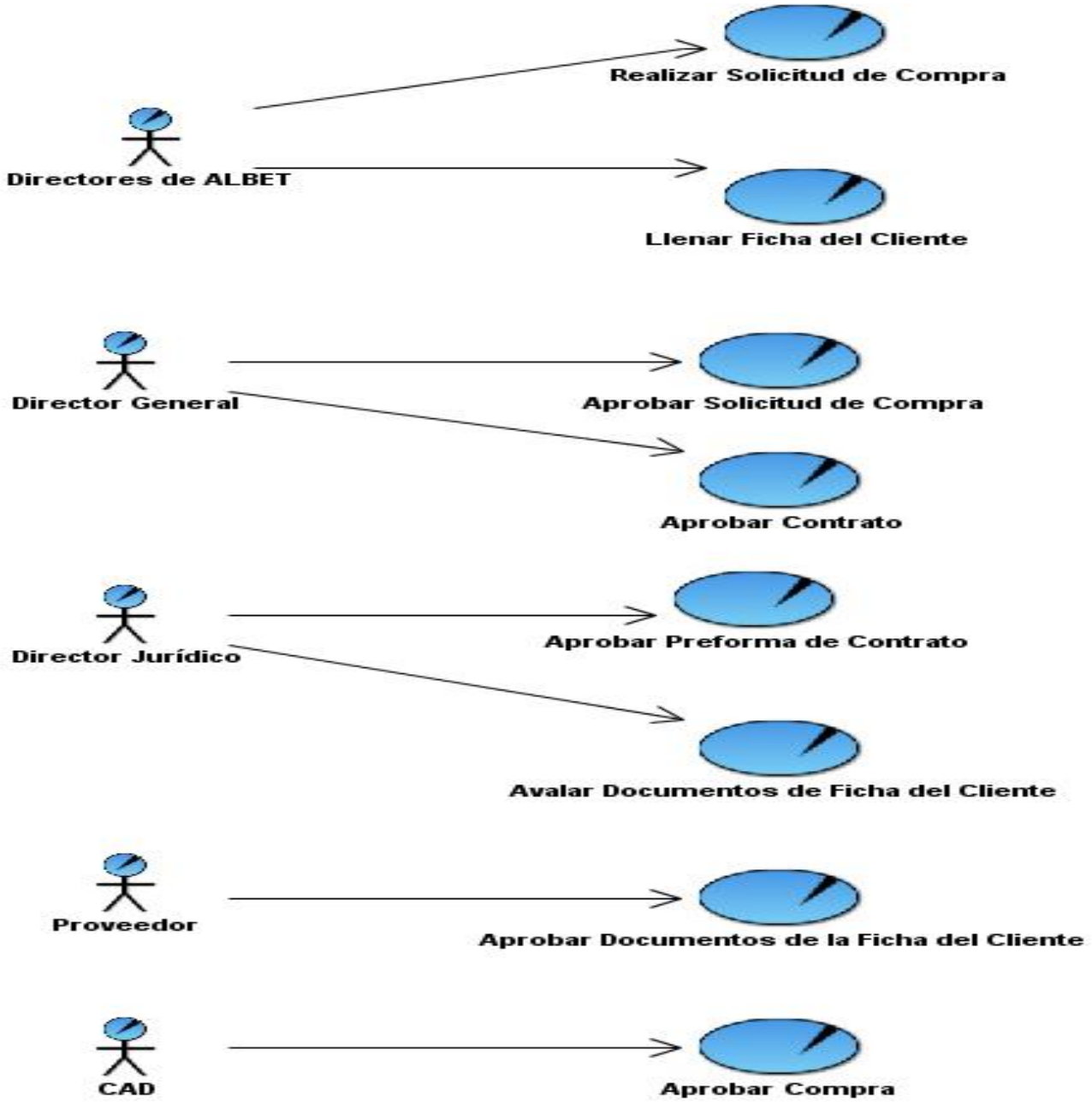


Figura 1 Diagrama de caso de uso del Negocio.

2.5.4 Descripciones de los caso de uso del negocio.

Caso de Uso del Negocio	Descripción del Caso de Uso	Actores
Realizar Solicitud de Compra	Se realiza un Modelo de Solicitud de Compras para sus áreas.	Directores de ALBET
Aprobar preforma de Contrato	Se revisa y prueba al preforma de Contrato y se pasa al Director General para ser Aprobado.	**Director Jurídico**
Aprobar Solicitud de Compra	Se aprueban las solicitudes de Compra realizadas anteriormente por los Directores de ALBET.	Director General
Aprobar Contrato	Se revisa y aprueba el contrato con el proveedor.	Director General
Avalar Documentos del Cliente	Se avalan los documentos que se adjuntan a la Ficha de Cliente para ser entregados al proveedor.	Director Jurídico
Aprobar Compra	Se presenta la Oferta y el Contrato y este procede a aprobar la Compra.	Comité de Aprobación de Divisas (CAD).

Tabla 3 Descripciones de los Caso de Uso del Negocio.

2.5.5 Diagrama de objetos.

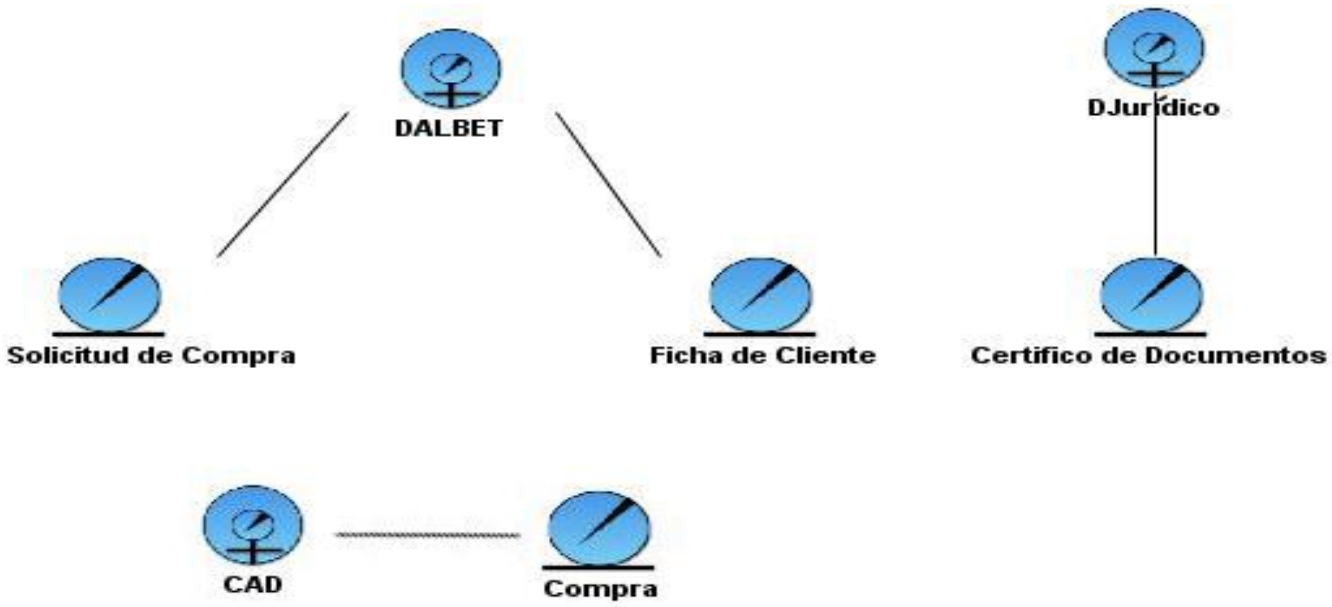


Figura 2 Diagrama de Objetos.

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, para ello se enumeran las funcionalidades que el sistema deberá ser capaz de realizar.

RF 1: 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.

RF 2: Gestionar usuario.

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

- RF2.1: Crear Usuario.

- RF2.2: Eliminar Usuario.
- RF2.3: Actualizar Usuario.
- RF2.4: Buscar Usuario.

RF 3: Realizar preforma de contrato.

El sistema debe mostrar una forma con los datos de un contrato y con la posibilidad de ingresar nuevos campos.

RF 4: Modificar preforma de contrato.

El sistema debe mostrar una forma con los datos de un contrato y con la posibilidad de ingresar nuevos campos. En caso de que el contrato deba ser modificado después de creado.

RF 5: Aceptar preforma de contrato.

El sistema debe dar la opción de aceptar la preforma de Contrato y debe enviar la misma al Director General.

RF 6: Aprobar contrato.

El sistema debe dar la opción de aceptar el Contrato y debe enviar una notificación a los implicados.

RF7: Gestionar compra.

El sistema debe permitir realizar las acciones de crear, buscar, modificar y eliminar una compra.

- RF3.1: Crear Compra.
- RF3.2: Buscar Compra.
- RF3.3: Modificar Compra.
- RF3.4: Eliminar Compra.

RF8: Registrar solicitud de compra.

El sistema debe permitir el registro de una solicitud de compra.

RF9: Aceptar solicitud de compra.

El sistema debe permitir que el responsable de compras acepte la solicitud de compra.

RF10: Aprobar compra.

El sistema debe permitir aprobar la Compra a un miembro del CAD autenticado previamente. El miembro del CAD se autentica en la aplicación y accede al listado de compras, para así elegir una compra y aprobarla o rechazarla.

2.7 Requisitos no funcionales.

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

Apariencia o interfaz externa.

El sistema deberá poseer una interfaz web sencilla, amigable, lo más atractiva y clara posible para el usuario, además su funcionamiento debe ser de fácil comprensión.

Usabilidad.

El sistema podrá ser usado por cualquier persona que posea conocimientos básicos en el manejo de la computadora y de un ambiente Web. Para garantizar la usabilidad del sistema el mismo cuenta con buena arquitectura de la información. Permitirá al usuario acceder mediante un menú localizado en la parte izquierda de la aplicación a las opciones a las cuales le sea necesario.

Rendimiento.

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

Portabilidad.

El sistema debe ser multi-plataforma.

Seguridad.

Para garantizar la integridad y la confidencialidad de la información que se maneja se pretende establecer un sistema de permisos a usuarios para el acceso, la entrada de los usuarios al sistema debe ser verificado si el mismo ya está autenticado, si no lo está, brindarle el servicio de autenticación. Además se utilizará el cifrado de contraseñas. De la misma manera se establecerá la gestión de permisos por roles para los usuarios registrados.

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.

Software.

En el lado del Cliente debe existir un navegador con soporte JavaScript, el cual puede estar en cualquier sistema operativo. En el lado del Servidor debe estar instalado el gestor de base de datos PostgreSQL, además de que debe contarse con servidor Apache Tomcat, por soportar este código java lo cual es imprescindible para el funcionamiento de la aplicación.

Hardware.

Cliente.

- Procesador Pentium II o superior.
- 256 MB de memoria RAM o superior.
- 4 GB de HDD.

Servidor.

- Procesador Pentium IV o superior.

- 1 GB de memoria RAM o superior.
- 80 GB de HDD (Este valor puede variar dependiendo de la carga de información).

2.8 Definición de los casos de uso del sistema.

A continuación quedan descritos los actores del sistema con cada una de sus responsabilidades. De la misma forma se muestras los diagramas de casos de uso del sistema y se describen algunos de los casos de uso críticos para el desarrollo de la aplicación.

2.8.1 Actores del sistema.

A continuación se muestran los roles y se describen los actores del sistema, dando una idea del nivel de acceso de cada uno de ellos a la aplicación. De acuerdo a lo que se describe en la tabla # 4 es que quedarán establecidos los permisos a la aplicación.

Actores	Descripción
Administrador	El administrador es el encargado de Gestionar los usuarios del sistema.
Director Jurídico	Aprueba la preforma de contrato.
Director General	Encargado de aprobar los contratos y aceptar las compras.
Directores de ALBET	Son los encargados de realizar las solicitudes.
CAD	Aprueban las compras.

Tabla 4 Actores del sistema.

2.8.2 Diagramas de casos de uso del sistema.

2.8.2.1 Módulo contrato.

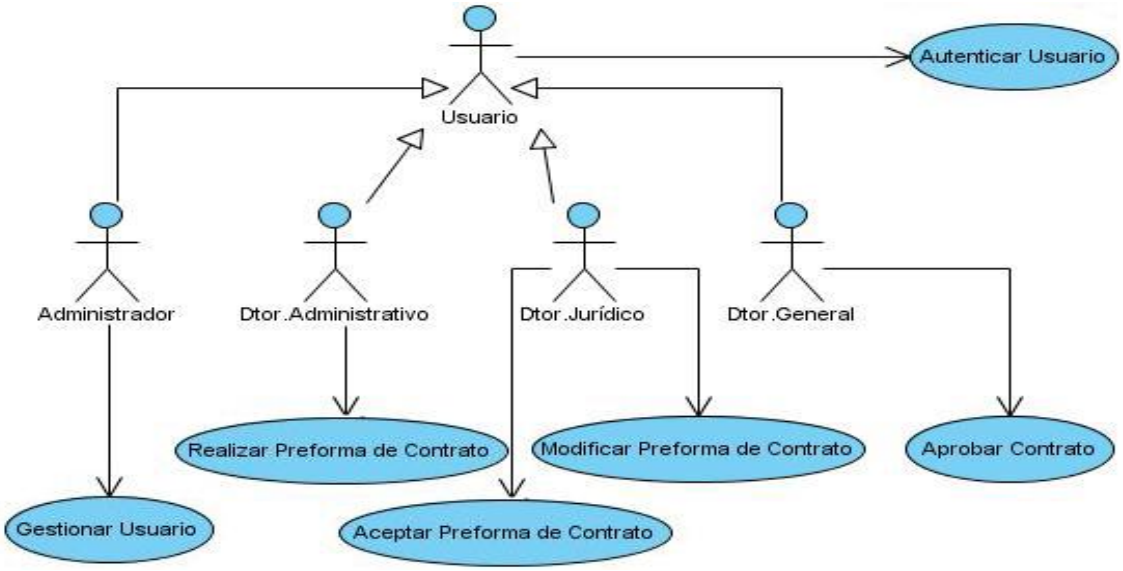


Figura 3 Diagrama de CUS-Módulo Contrato.

2.8.2.2 Módulo compra.

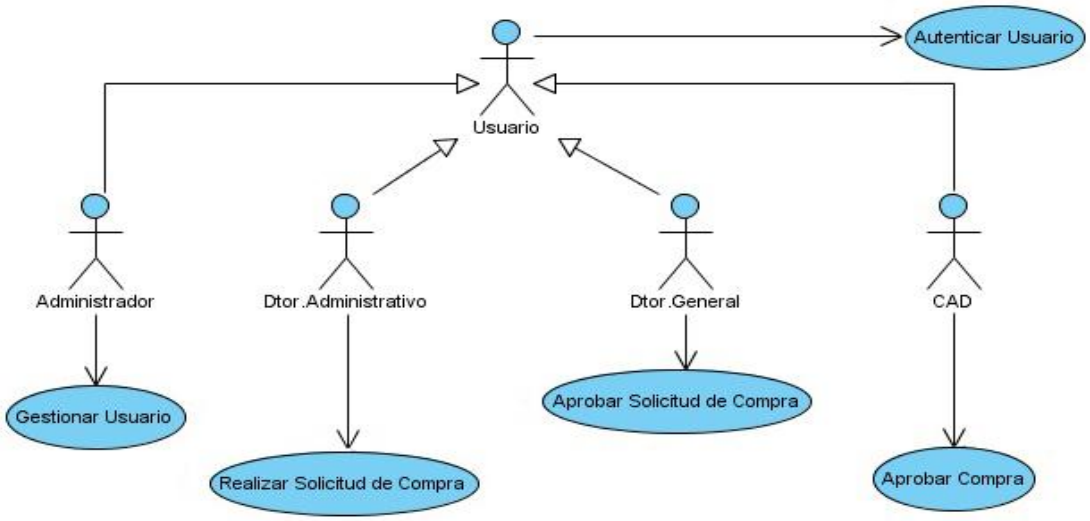


Figura 4 Diagrama de CUS-Módulo Compra.

2.8.3 Listado de los casos de uso del sistema.

2.8.3.1 Realizar preforma de contrato.

Caso de Uso:	Realizar Preforma de Contrato.
Actores:	Director Administrativo.
Resumen:	El Caso de Uso se inicia cuando el Director Administrativo accede al modelo de Contrato, se muestra la interfaz correspondiente para llenar la preforma de Contrato y el caso de uso termina cuando introduce todos los datos.
Precondiciones:	El usuario debe estar autenticado como Director Administrativo.
Referencias	RF 3.
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Director Administrativo selecciona la opción “Realizar Preforma de Contrato”	2. El sistema muestra el formulario de Contrato con los siguientes datos: • # • Nombre • Cliente • Estado • Proveedor • Costo • Tiempo Activo o Duración • Tipo

3. El Director Administrativo introduce los datos necesarios para la creación de la preforma.	4. El sistema verifica que no existan campos obligatorios en blanco y que los datos estén correctos.
	5. El sistema crea un Contrato.
	6. El sistema envía la preforma de Contrato a la lista para que pueda ser accedida por el Director Jurídico.
	7. El sistema da la posibilidad de crear otra preforma de Contrato.
	8. El sistema va a la acción 2 de esta sección.

Prototipo de Interfaz

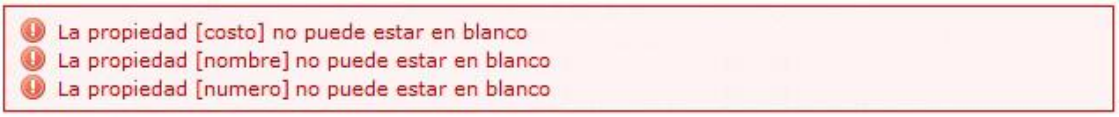
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1. En caso que los datos no estén correctos muestra un mensaje de error “Datos Incorrectos”.
8.1 Si el Director Administrativo no desea realizar otra preforma de Contrato finaliza el caso de uso.	
Prototipo de Interfaz 	
Poscondiciones	Se crea un Contrato.

Tabla 5 Listado de los CUS- Realizar Preforma de Contrato.

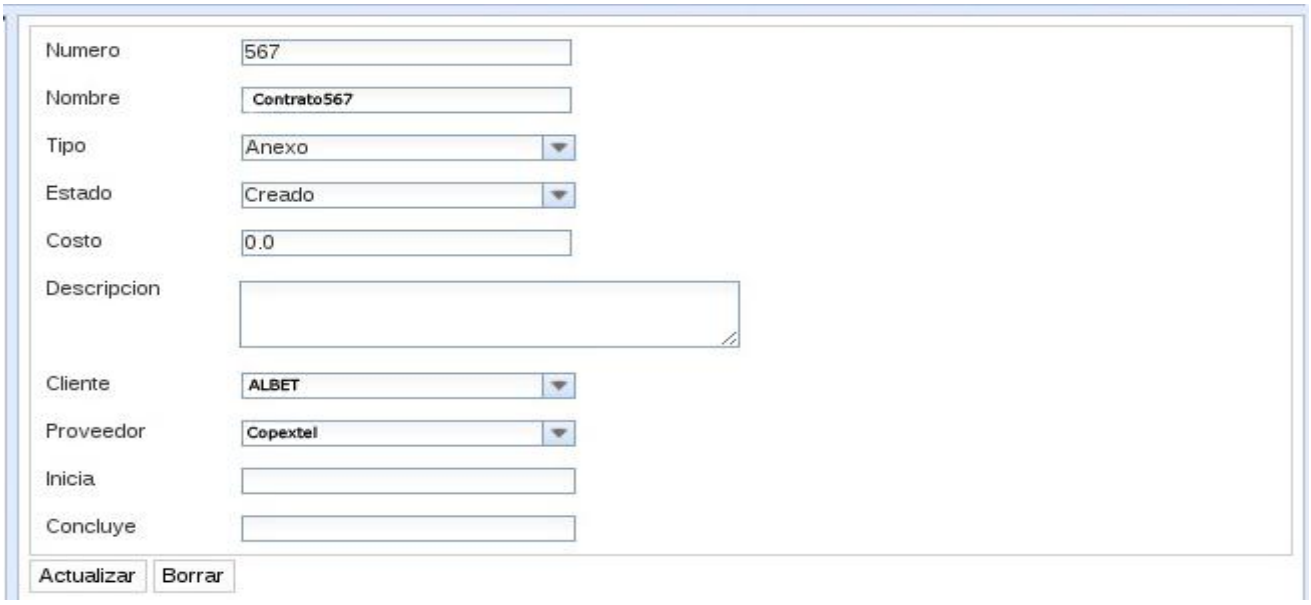
2.8.3.2 Aprobar contrato.

Caso de Uso	Aprobar Contrato.
Actores	Director General.

Resumen	El Caso de Uso inicia cuando el Director General accede a la interfaz de Contratos donde se encuentran todos los Contratos existentes, el caso de uso termina cuando el Director General Acepta el Contrato.
Precondiciones	El usuario debe estar autenticado como Director General.
Referencias	RF 6.
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Director General elige la opción “Contratos”.	2. El sistema muestra los Contratos existentes.
3. El Director General presiona el id del contrato que desea mostrar.	4. El sistema muestra el Contrato y brinda las opciones “Aprobar” y “Aplazar”.
5. El Director General selecciona la opción “Aprobar”	6. El sistema cambia el estado del contrato a “Aprobado”.
	7. El sistema muestra el contrato modificado y da la posibilidad de regresar a la lista de contratos.
8. El Director General selecciona la opción “Aprobar otro”.	9. El sistema va a la acción 2 de esta sección.

Prototipo de Interfaz



Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5.1. El Director General selecciona la opción “Aplazar”.	6.1. El sistema cambia el estado del Contrato a “Aplazado”.
8.1. Si el Director General no desea consultar algún otro Contrato finaliza el caso de uso.	

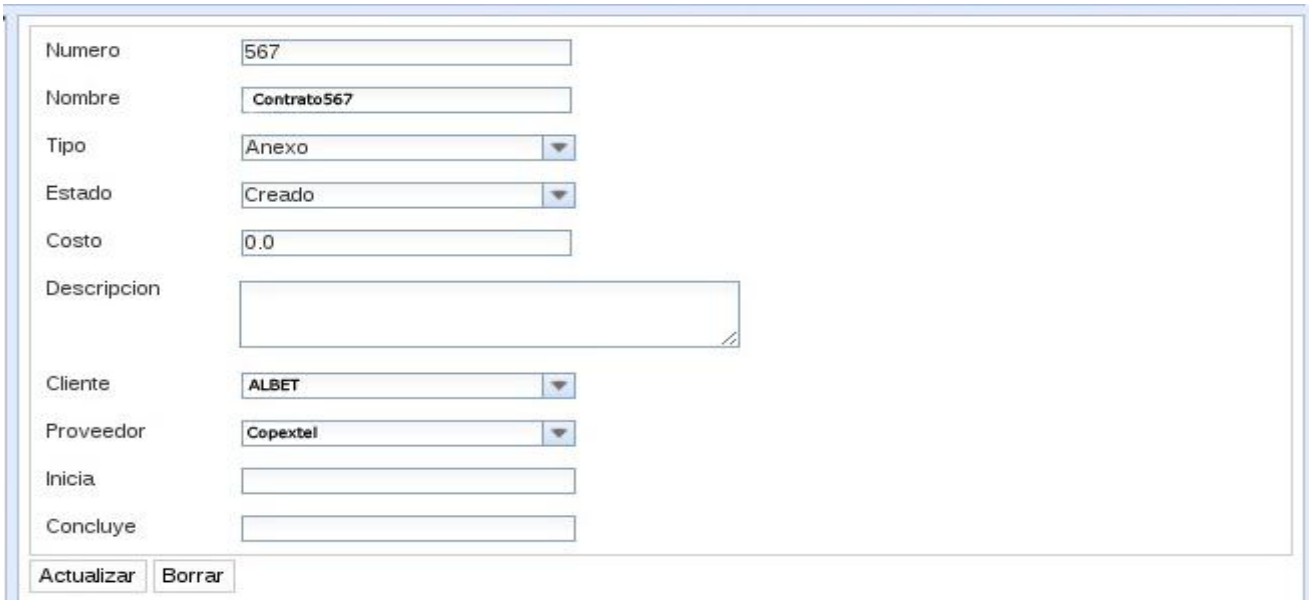
	Prototipo de Interfaz 
Poscondiciones	El Contrato queda modificado, pasando su estado de “Aceptado” a “Aprobado” o a “Aplazado”.

Tabla 6 Listado de los CUS-Aprobar Contrato.

2.8.3.3 Realizar solicitud de compra.

Caso de Uso	Realizar Solicitud de Compra.
Actores	Director Administrativo.
Resumen	El Caso de Uso se inicia cuando el Director Administrativo accede al modelo de solicitud de Compra, se muestra la interfaz correspondiente para llenar la solicitud, el caso de uso termina cuando introduce todos los datos.

Precondiciones	El usuario debe estar autenticado como Director Administrativo.
Referencias	RF 8
Prioridad	Crítico.

Flujo Normal de Eventos

Acción del Actor	**Respuesta del Sistema**
1. El Director Administrativo selecciona la opción “Realizar Solicitud de Compra”	2. El sistema muestra el formulario de solicitud de compra.
3. El Director Administrativo introduce los datos necesarios para la creación de la solicitud.	4. El sistema verifica que no existan campos obligatorios en blanco.
	5. El sistema crea una nueva solicitud.
	6. El sistema va a la acción 2 de esta sección.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	4.1. En caso que existan campos obligatorios en blanco muestra un mensaje de error “El campo “campo” no puede estar en blanco”.
8.1 Si el Director Administrativo no desea realizar otra preforma de Contrato finaliza el caso de uso.	

Prototipo de Interfaz

Poscondiciones	Se crea una solicitud de compra.

Tabla 7 Listado de los CUS-Realizar Solicitud de Compra.

2.8.3.4 Aprobar compra.

Caso de Uso	Aprobar Compra.
Actores	CAD.
Resumen	El Caso de Uso inicia cuando el miembro del CAD accede a la interfaz de Compras donde se encuentran todos los Compras por aprobar, el caso de uso termina cuando el miembro del CAD Aprueba la Compra.
Precondiciones	El usuario debe estar autenticado como miembro del CAD.
Referencias	RF 10.
Prioridad	Crítico.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El representante del CAD elige la opción “Aprobar/Aplazar”.	2. El sistema muestra las compras existentes.
3. El representante del CAD presiona el id de la que desea mostrar.	4. El sistema muestra la compra y brinda las opciones “Aprobar” y “Aplazar”.
5. El representante del CAD selecciona la opción “Aprobar”	6. El sistema cambia el estado de la compra a “Aprobado.

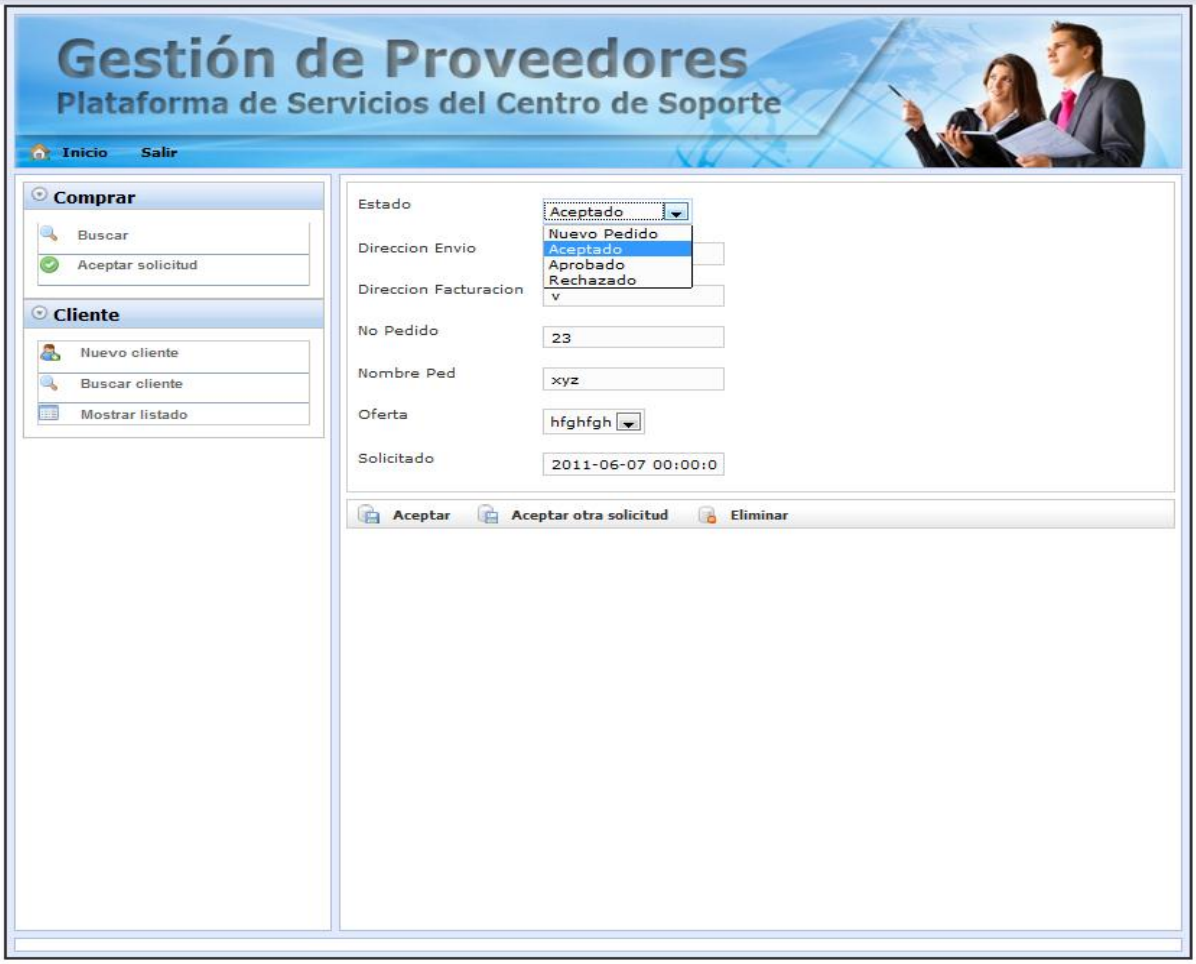
	7. El sistema brinda la posibilidad de consultar otra compra.
8. El representante del CAD selecciona la opción “Aprobar otra solicitud”.	9. El sistema va a la acción 2 de esta sección.

Prototipo de Interfaz

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5.1. El representante del CAD selecciona la opción “Aplazar”.	6.1. El sistema cambia el estado de la Compra a “Aplazado”.

8.1. Si el representante del CAD no desea consultar alguna otra Compra finaliza el caso de uso.	

Prototipo de Interfaz



Poscondiciones	La compra pasa su estado de “Aceptado” a “Aprobado” o a “Aplazado”.

Tabla 8 Listado de los CUS- Aprobar Compra.

Conclusiones parciales.

En el presente capítulo se identificaron los actores (Administrador, Director Jurídico, Director General, Directores de ALBET, CAD) y trabajadores del negocio (Director Jurídico, Director General, Directores de ALBET, CAD) y se representaron en el Diagrama de Casos de Uso del Negocio, el mismo cuenta con ocho casos de uso, de igual forma se representó el Diagrama de Objetos.

También se identificaron los siguientes requerimientos funcionales: Autenticar usuario, Gestionar usuarios, Realizar preforma de contrato, Modificar preforma de contrato, Aceptar preforma de contrato, Aprobar contrato, Gestionar compra, Registrar solicitud de compra, Aceptar solicitud de compra, Aprobar compra, Registrar ficha de cliente, Avalar documento de ficha de cliente. Como requerimientos no funcionales del sistema: Apariencia o interfaz externa, Usabilidad, Rendimiento, Portabilidad, Seguridad, Disponibilidad, Software, Hardware, así como los actores y casos de uso del sistema. Se presentaron los diagramas de casos de uso del sistema, para cada módulo, mostrando las relaciones entre los actores del sistema y los casos de uso. Se describieron cada uno de los casos de uso del sistema.

Capítulo 3. Diseño de la solución.

En el presente capítulo se crea el diseño de la solución y se generan toda una serie de artefactos para guiar el proceso de implementación de la solución. Siempre atendiendo a lo recogido en los requisitos funcionales y no funcionales sobre como el sistema debe responder en cada caso para dar respuesta a las necesidades del usuario.

3.1 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". 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. (14)

3.1.1 Modelo Vista Controlador (MVC).

El MVC fue descrito por primera vez en 1979 por Trygve Reenskaug, trabajador de *Smalltalk*, en unos laboratorios de investigación de Xerox. MVC es un principio de diseño arquitectónico que separa los componentes de la aplicación web. Esta separación ofrece más control sobre las partes individuales de la aplicación, lo cual permite desarrollarlas, modificarlas y probarlas más fácilmente.

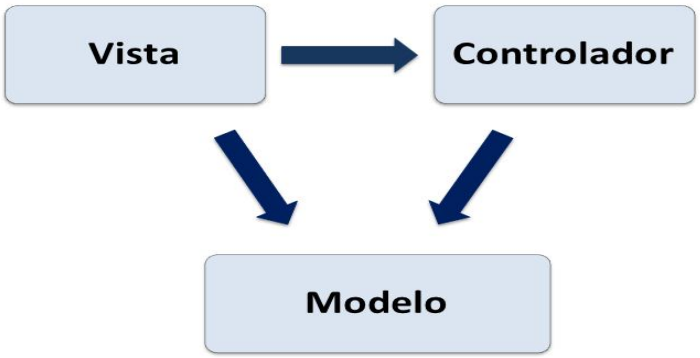


Figura 5 Modelo Vista Controlador.

Los fundamentos básicos del MVC son los siguientes:

Modelo: sirve como representación específica de toda la información con la cual el sistema va a trabajar. La lógica de datos puede llegar a asegurar la integridad de ellos y permitirá derivar nuevos datos.

Vista: presenta el modelo con el que va a interactuar el usuario, más conocida como interfaz.

Controlador: el controlador responde a eventos, normalmente son acciones que el usuario invoca, implica cambios en el modelo y también en la vista. (8)

3.1.2 Patrones de diseño.

Los patrones de diseño son soluciones a problemas comunes, basados en la experiencia y que está demostrado que son efectivos a la hora de corregir malas prácticas en el diseño de aplicaciones.

El MVC como patrón de diseño.

En el patrón de diseño de software MVC todo el proceso está dividido en 3 capas, típicamente estas capas son el Modelo, la Vista y el Controlador. El Modelo incorpora la capa del dominio y persistencia, es la encargada de guardar los datos en un medio persistente (en la presente solución los datos se almacenan en una base de datos.).

La vista se encarga de presentar la interfaz al usuario, en el presente sistema, la vista está compuesta por elementos HTML, con sentencias groovy y la utilización de las librerías Dojotoolkit. El controlador es el que escucha los datos enviados desde la vista y realiza los cambios en esta, de la misma forma los envía al modelo, por otra parte cuando son requeridos estos, regresa los datos a la vista. Este patrón es pensando en capas y así es como lo implementa el framework Grails, 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. (13)

Patrones GRASP.

Los patrones GRASP están orientados a lograr aplicaciones web más robustas y seguras. Estos patrones tienen en cuenta cada una de las necesidades tecnológicas para la creación de aplicaciones. Los patrones constituyen soluciones a problemas comunes de diseño, estas soluciones ya han sido probadas y está demostrado que funcionan en sistemas orientados a objetos. (2)

Patrón Controlador.

Problema: ¿Quién debería encargarse de atender un evento del sistema?

Solución: asignar la responsabilidad del manejo de una serie de eventos determinados a un controlador.

En la aplicación: en el caso de Grails, estos controladores son denominados [Nombre_Entidad]Controller. En el sistema cada controlador (*Controller)* es el encargado de manejar los eventos y la lógica del negocio del sistema, relacionados a la clase entidad que maneja.

Patrón Experto.

Problema: ¿Quién debiera ser el responsable de conocer la información?

Solución: asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad.

En la aplicación: el sistema implementa este patrón en forma de clases Entidades *(Domain Classes).* De esta forma cuando son necesarios datos para realizar alguna operación, estos son llamados de la clase entidad que los contiene, las cuales son las encargadas del acceso a datos.

Patrón Creador.

Problema: ¿Quién debería ser responsable de crear una nueva instancia de alguna clase? Solución: asignarle a la clase B la responsabilidad de crear una instancia de clase A en uno de los siguientes casos:

B agrega los objetos A.
B contiene los objetos A.
B registra las instancias de los objetos A.
B utiliza especialmente 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). B es un creador de los objetos A.

En la aplicación: Las clases [Nombre_Entidad]Controller, son responsables de crear instancias de las clases Entidad para realizar las funciones de insertar, modificar y eliminar datos en la base de datos. De la misma forma una clase entidad puede tener la responsabilidad de crear un objeto de otra clase entidad.

Patrón Alta cohesión.

Problema: ¿Cómo mantener una alta cohesión dentro de los límites manejables?

Solución: realizar las acciones correspondientes a una clase determinada en dicha clase únicamente.

En la aplicación: una clase será cohesionada cuanto más enfocado sea su comportamiento. Esto quiere decir que, a la hora de implementar las clases, se asignarán los métodos de forma coherente, completa y relacionada. De esta forma todos los métodos que se puedan afectar con un cambio o toda la información referente a la clase estarán a la vista en el mismo fichero. El sistema implementa este patrón en conjunto con el patrón de bajo acoplamiento, de forma tal que cada clase realiza sus acciones y se evita que una

clase realice acciones correspondientes a la clase con la que está relacionada. Esto genera bajo acoplamiento haciendo mínima la dependencia entre una clase y otra. La relevancia y utilidad de este patrón, como de todos los otros se hace visible a la hora de realizar cambios. Al realizar una modificación a una clase cohesionada y que por consiguiente tenga una bajo acoplamiento con otra clase, solo se afecta una de las dos, siendo la modificación transparente a la otra.

Patrón Bajo acoplamiento.

Problema: ¿Cómo mantener un bajo acoplamiento entre las clases?

Solución: Para mantener un bajo acoplamiento entre clase se debe garantizar la alta cohesión, con el fin de que las clases no dependan en gran medidas unas de las otras. De esta forma las clases se vuelven más fáciles de comprender, mantener y reutilizar.

En el sistema: se implementa este patrón asegurando que cada clase que está relacionada con otro no implemente métodos pertenecientes a esta clase. La clase contratoController.groovy, a la hora de retornar a una vista cuando concluye un contrato, no implementa este método, simplemente lo llama de la clase contrato.

3.3 Representación de la arquitectura.

Es determinante definir una arquitectura y describirla detalladamente, parte fundamental del proceso de descripción de la arquitectura es la representación de la misma. A continuación se presenta el modelo de diseño, elemento principal para comprender la arquitectura del sistema.

3.4 Modelo de 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.

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 y en el modelo. (14)

3.5 Diagrama 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:

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

En el modelado de la aplicación se representa un subsistema como “Componentes Grails” que representa los componentes agregados por el *framework Grails*, además del objeto relacional llamado *Grails Object Relational Mapping* (GORM), 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 la 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. (6)

A continuación se presentan diagramas de Clases del Diseño Utilizados para el desarrollo de la solución:

3.5.1 Diagrama de clases del diseño: CU Realizar Preforma de Contrato.

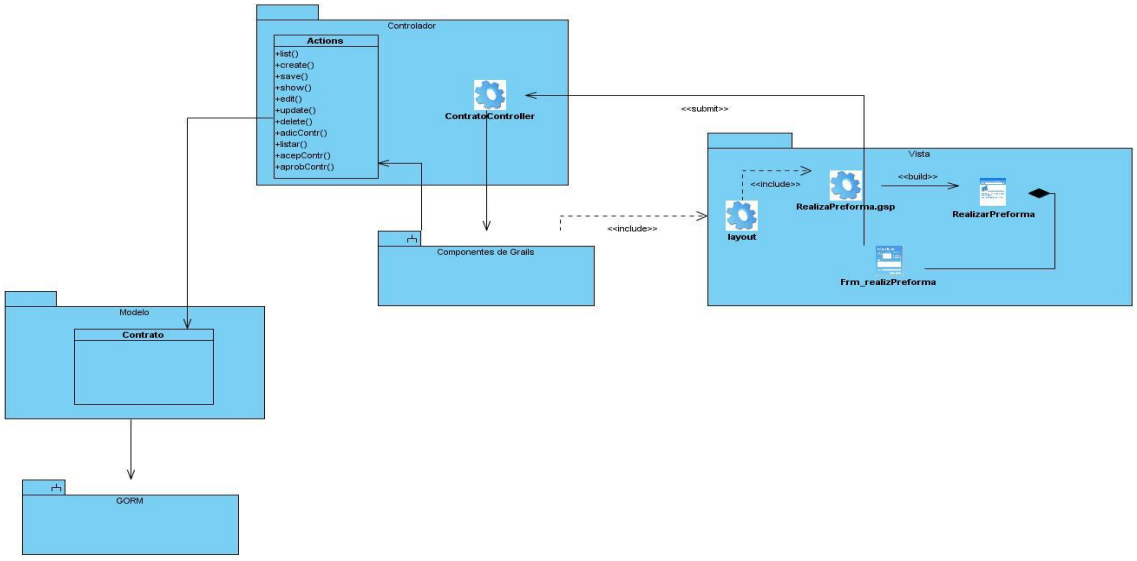


Figura 6 Diagrama de CD: CU Realizar Preforma de Contrato.

3.5.2 Diagrama de clases del diseño: CU Aprobar Contrato.

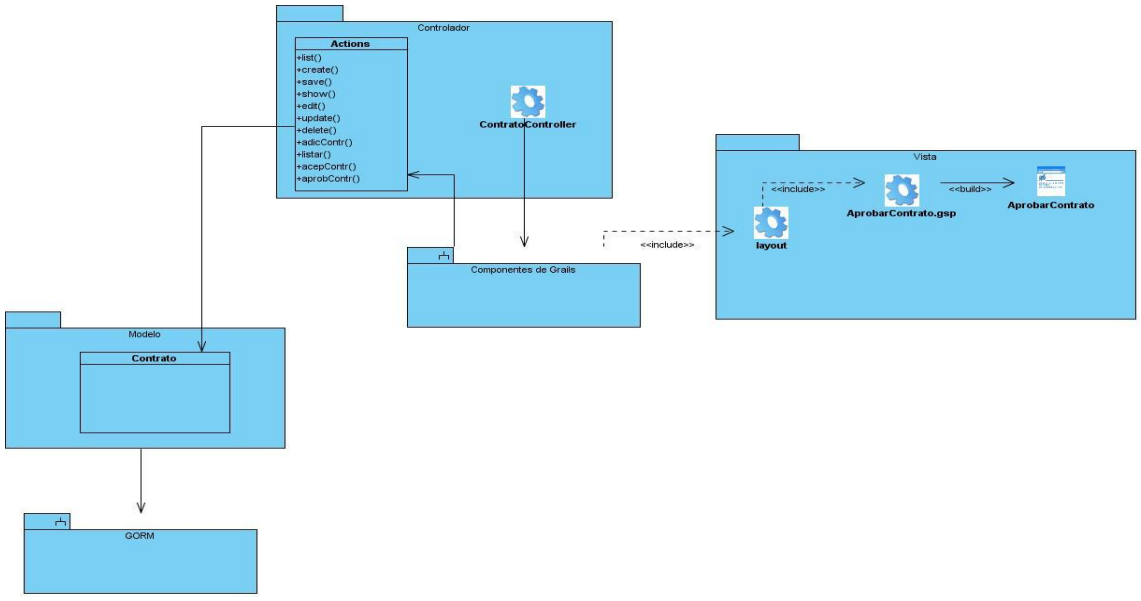


Figura 7 Diagrama de CD: CU Aprobar Contrato.

3.6 Diagramas de colaboración.

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. (16)

A continuación se presentan diagramas de clases del diseño utilizados para el desarrollo de la solución:

3.6.1 Diagrama de colaboración: CU Realizar Solicitud de Compra.

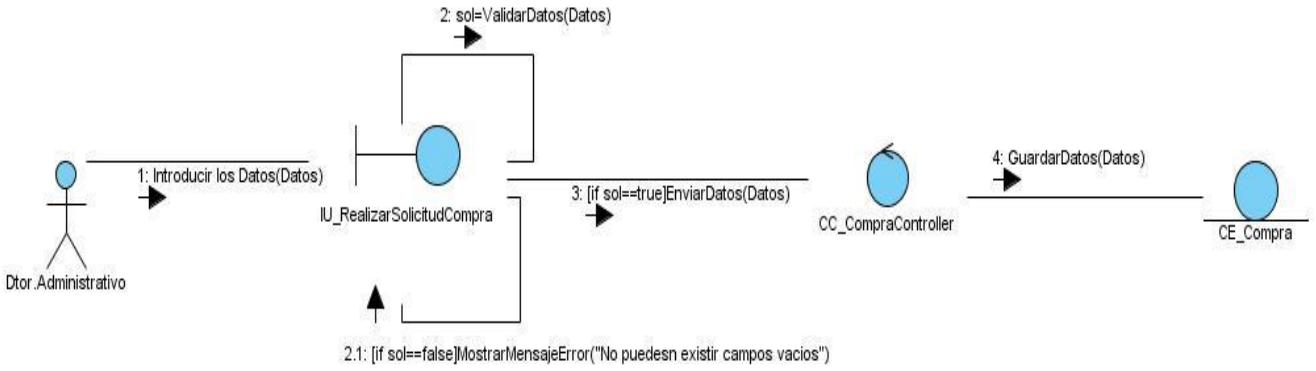


Figura 8 Diagrama de colaboración: CU Realizar Solicitud de Compra.

3.6.2 Diagrama de colaboración: CU Aprobar Compra.

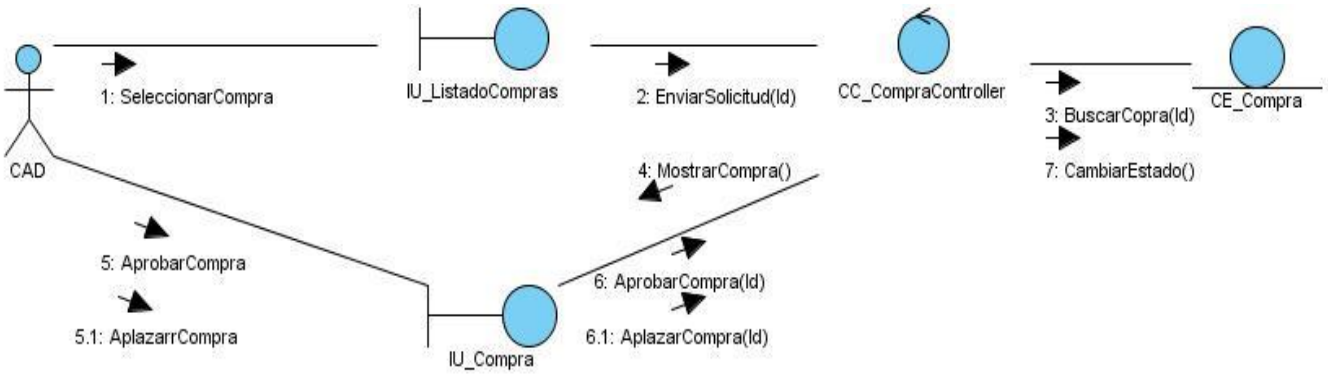


Figura 9 Diagrama de colaboración: CU Aprobar Compra.

3.7 Mapa de navegación.

El mapa de navegación constituye una representación gráfica de la constitución del sitio. En él esta descrito de forma clara como es que se relacionan las diferentes vistas que están presentes en la aplicación y como llegar a ellas partiendo del punto de inicio. Es una herramienta a disposición del usuario que brinda una guía dentro de la aplicación. A continuación se muestra el mapa de navegación de los módulos contrato y copra del sistema.

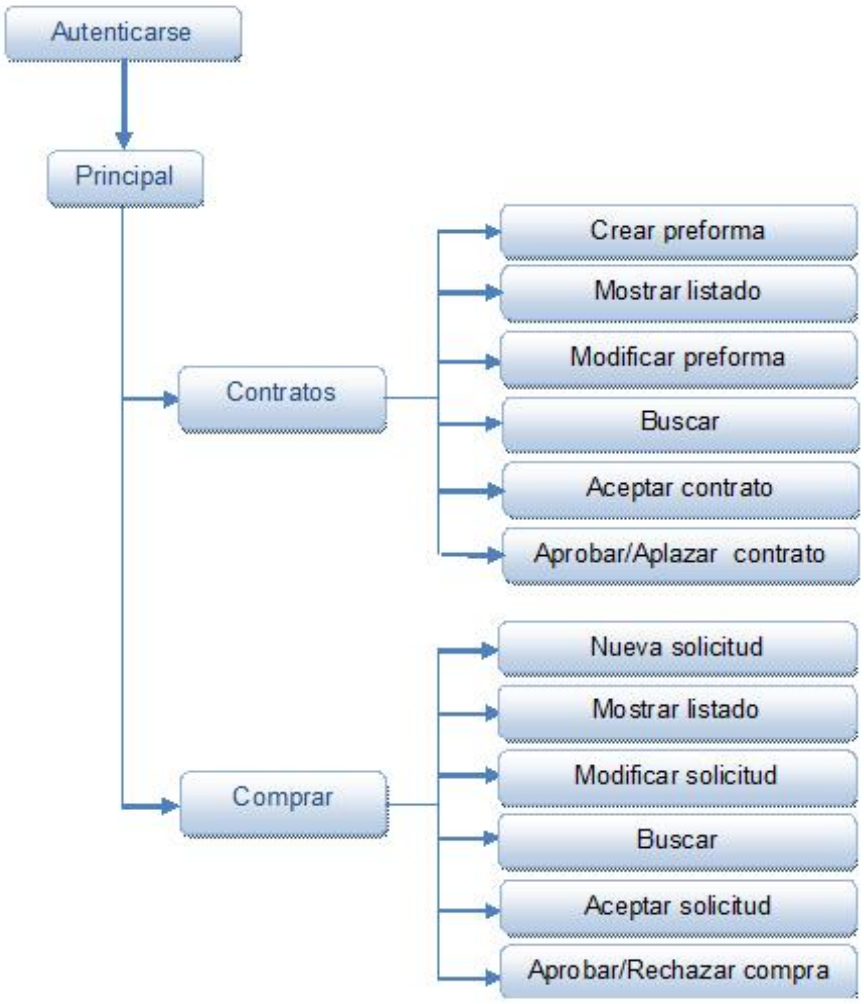


Figura 10 Mapa de navegación.

3.8 Modelo de datos.

Un modelo de datos es básicamente... “un conjunto de conceptos, reglas y convenciones que permiten describir y en ocasiones manipular los datos de un cierto mundo real que deseamos almacenar en la base de datos”. (17)

3.8.1 Diagrama relacional.

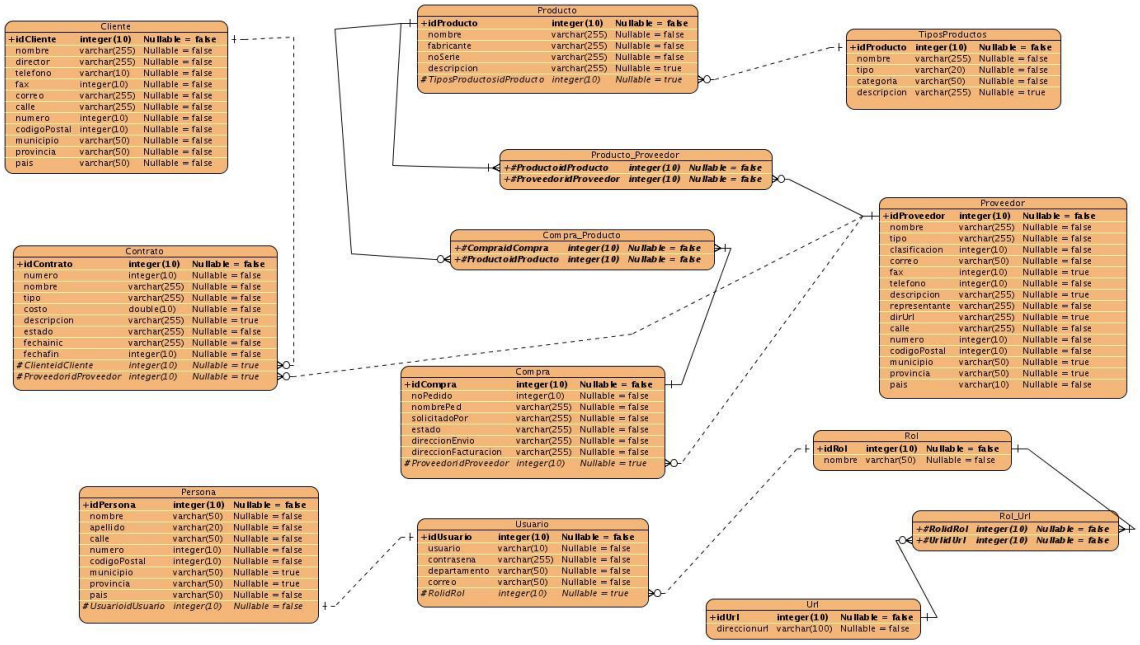


Figura 11 Diagrama relacional.

Conclusiones parciales.

En el presente capítulo quedó descrita la arquitectura del sistema (MVC), se identificaron los patrones de diseño a utilizar para el desarrollo del sistema como: (patrón Modelo Vista Controlador y los patrones GRASP), sirviendo de base para la construcción de los diagramas de clases básicos para obtener un correcto diseño. Para describir la arquitectura se utilizó el modelo de diseño donde se presentaron los diagramas de clases del diseño, los diagramas de interacción (colaboración). Con vistas a crear una representación panorámica de la aplicación se muestra el mapa de navegación y finalmente para que se pueda apreciar la relación entre las entidades se presentan el diagramas de clases persistentes y el modelo relacional.

Capítulo 4. Implementación y pruebas.

A través de la realización de este capítulo se describe cómo se llevó a cabo la implementación de la aplicación en términos de componentes. Se describen algunos fragmentos de códigos, se exponen los principales resultados obtenidos a lo largo del trabajo, a la vez que se discute sobre la importancia y relevancia de los mismos. También en este capítulo se realiza la construcción de los casos de pruebas para las principales funcionalidades.

4.1 Modelo de implementación.

El Modelo de Implementación describe cómo los elementos del Modelo de Diseño son implementados en términos de componentes. Describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponible en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y la dependencia entre los componentes. (18)

El diagrama de componentes mostrará las organizaciones y dependencias lógicas entre componentes software, sean estos componentes de código fuente, archivos, paquetes, bibliotecas cargadas dinámicamente o ejecutables.

Gestionar Compra.

El caso de uso comprende los procesos referentes a la manipulación de las compras, lo que se traduce en el registro, la búsqueda, la modificación, la aceptación, aprobación o rechazo de una compra determinada. Todas las acciones que se realizan sobre y en función de las compras, están comprendidas y son ejecutadas por un controlador, el cual se ocupa de desplegar las vistas necesarias para su funcionamiento y de hacer persistir los datos en el momento requerido, se trata del *CompraController.groovy.* El caso de uso está descrito en términos de componentes más adelante en la figura 19.

Gestionar Contrato.

El caso de uso comprende los procesos referentes a la manipulación de los contratos, la creación de una nueva preforma, la aceptación de esta preforma, la búsqueda, la aprobación o el rechazo de un contrato determinado. Estas acciones son ejecutadas por el controlador *ContratoController.groovy* el cual es el encargado de desplegar las vistas necesarias para realizar dichas acciones, de la misma forma es el encargado de hacer persistir los datos en el momento indicado. El caso de uso está descrito más adelante en términos de componentes en la figura 20.

4.2 Diagrama de componentes.

4.2.1 Compra.

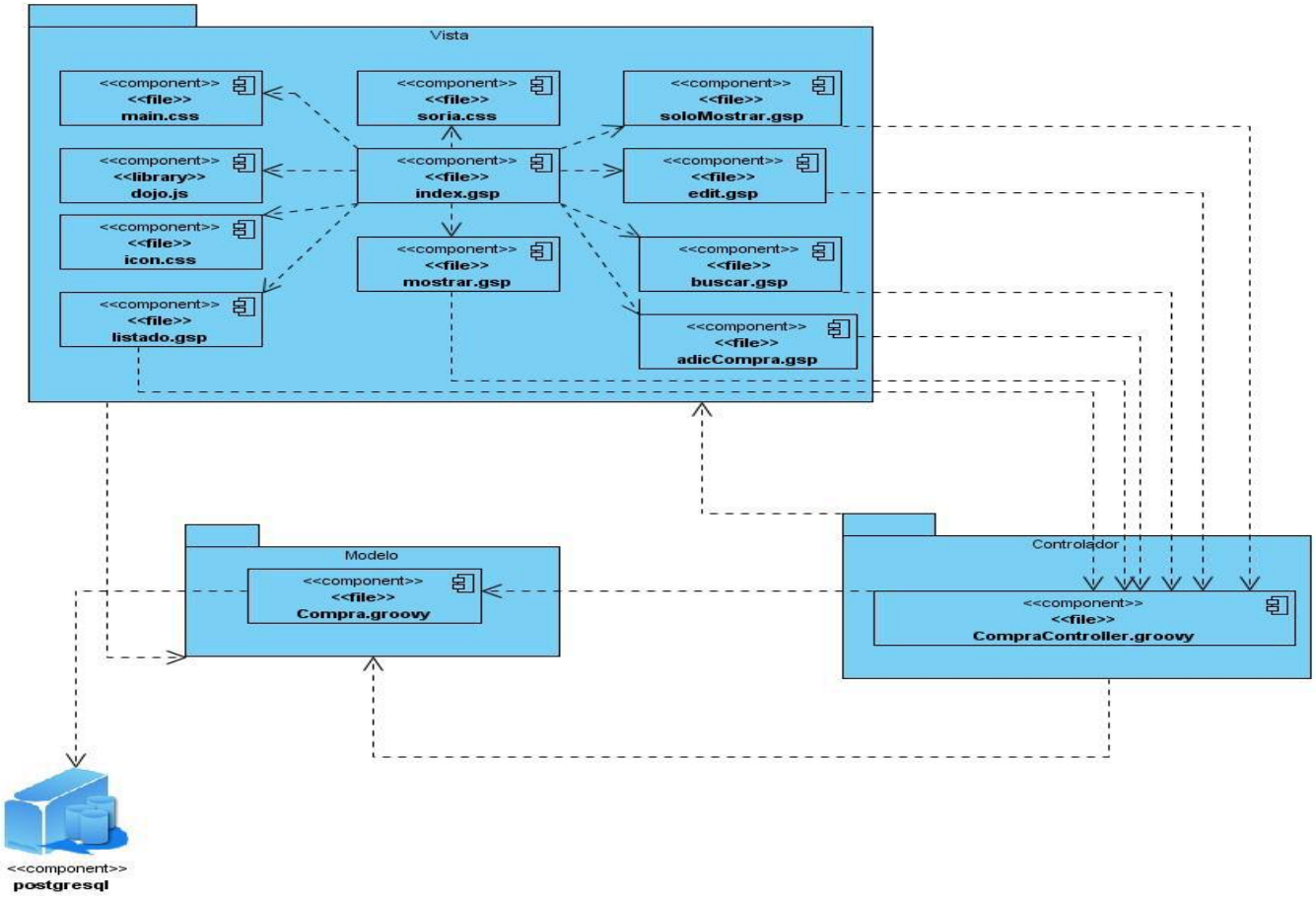


Figura 12 Diagrama de componentes: Compra.

4.3 Código.

A continuación se muestra el código de uno de los componentes referidos en la figura 12.

adicCompra.gsp

```
<div dojoType="dijit.layout.ContentPane" title="<a class='tituloTab'>Pedido</a>" >
<g:formRemoteurl="[action:'save', controller:'compra']" name="formcompra" method="POST">
<div style="margin-bottom: 1.3%">
<fieldset><legend class="legen">Detalles de la Compra</legend>
<table  border="0" ><tr><td>
<label   class="h" for="noPedido"><b style="color: red">* </b>Pedido NÂº:</label><br>
</td>
<td><input  size="30" dojoType="dijit.form.ValidationTextBox" type="text" name="noPedido" id="noPedido"
promptMessage="Introduzca el número del Pedido"invalidMessage="El campo no puede estar vacío"
required="true"></td></tr>
<tr><td><label class="h" align="right" for="nombrePed"><b style="color: red">* </b>Nombre del Pedido: </label>
</td>
<td><input class="formContainer"  dojoType="dijit.form.ValidationTextBox" type="text" name="nombrePed"
id="nombrePed"promptMessage="Introduzca el nombre del Pedido"invalidMessage="El campo no puede estar vacío"
required="true"></td></tr>
<tr><td><label align="right" class="h" for="solicitadoPor">Solicitado: </label></td>
<td><input class="formContainer" size="30" dojoType="dijit.form.DateTextBox"type="text" name="solicitado"id="solicitadoPor"></td></tr>
<tr><td><label align="right" class="h"  for="estado">Estado: </label></td><td><select  dojoType="dijit.form.FilteringSelect" readonly="true"
name="estado" id="estado">
<optionvalue="Nuevo Pedido">Nuevo Pedido</option>
<optionvalue="Aceptado">Aceptado</option>
<optionvalue="Aprobado">Aprobado</option>
<option value="Rechazado">Rechazado</option></select></td></tr></table></fieldset>
<fieldset><legend class="legen">Oferta</legend>
<table ><tr><td width="38%"><label class="h" align="right" for="oferta"><b style="color: red">* </b>Oferta:</label>
</td>
<td><g:select from="${modulos.Oferta.list()}"  optionKey="id" optionValue="nombre"dojoType="dijit.form.FilteringSelect"
name="oferta"/></td></tr></table></fieldset>
<fieldset><legend class="legen">DirecciÃ³n</legend>
<table><tr><td><label class="h" align="right" for="direccionEnvio"><b style="color: red">* </b>Dirección de Envío: </label></td>
<td><input class="formContainer" size="30" dojoType="dijit.form.SimpleTextarea" type="text" style="width:215px; height: 38px"
name="direccionEnvio" id="direccionEnvio">
</td></tr>
<tr><td><labelalign="right" class="h" for="direccionFacturacion"><b style="color: red">* </b>Dirección de Facturación: </label></td>
<td><input class="formContainer" size="30" dojoType="dijit.form.SimpleTextarea" style="width:215px; height: 38px" type="text"
name="direccionFacturacion" id="direccionFacturacion">
</td></tr></table></fieldset>
</div>
<div dojoType="dijit.Toolbar">
<button type="submit" iconClass="icoGoadd" dojoType="dijit.form.Button">Enviar</button></div></g:formRemote></div>
```

4.4 Pruebas.

No es posible la validación de un sistema sin realizar pruebas al mismo, de esta forma a la solución propuesta se le han realizado una serie de pruebas, en primer lugar se le han venido realizando pruebas a nivel de desarrollador a lo largo de todo el proceso de implementación del sistema. Además de las mencionadas pruebas, se le realizaron pruebas de interfaz enfocadas a probar el desempeño de la aplicación.

A continuación se describe unas de las pruebas realizadas a la solución.

Pruebas por CU.

Estas pruebas se realizan a las vistas y la ejecución del caso de uso referido, ingresando una serie de datos, correctos o incorrectos y registrando los resultados obtenidos. De esta forma puede comprobarse si los componentes encargados de ejecutar el CU están realizándolo de forma correcta.

4.4.1 CU Realizar Preforma de Contrato.

El Caso de Uso se inicia cuando el Director Administrativo accede al modelo de Contrato, se muestra la interfaz correspondiente para llenar la preforma de Contrato y el caso de uso termina cuando introduce todos los datos.

Condiciones de ejecución.

Deben estar creados el cliente y al menos un proveedor.

4.4.1.1 SC Realizar Preforma de contrato.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 Creación exitosa.	El escenario consiste en la entrada de todos los datos válidos y la creación de una preforma de contrato.	V 1	V Contrato de Prueba 1	V ALBET	El sistema crea una nueva preforma de contrato y muestra un mensaje de confirmación.	1-El usuario ingresa los datos correctamente. 2-El usuario presiona el botón “Enviar”.
		Variable 4	**Variable 5**	**Variable 6**		
		V Proveedor 1	V Anexo	V 600		
		Variable 7	**Variable 8**	**Variable 9**		
		NA Contrato de…	V Creado	V 25/3/10-25/3/11		

EC 1.2 Campos obligatorios en blanco.	El escenario consiste en que faltan algunos campos obligatorios.	I [vacío]	I Contrato de Prueba 1	I [vacío]	El sistema muestra un mensaje de error, por los campos en blanco.	1-El usuario no ingresa todos los datos. 2-El usuario presiona el botón "Enviar".
		I [vacío]	I [vacío]	I [vacío]		
		NA [vacío]	I [vacío]	I [vacío]		

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Tabla 9 Prueba: SC Crear Preforma de contrato.

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	No. Contrato	Campo de texto.	No.	El campo puede contener tanto letras como números.
2	Nombre del Contrato	Campo de texto.	No.	El campo puede contener tanto letras como números.
3	Cliente	Lista de selección.	No.	El campo solo va a contener el nombre de la empresa cliente.
4	Proveedor	Lista de selección.	No.	El campo va a contener todos los proveedores que estén registrados en el sistema.
5	Tipo de contrato	Lista de selección.	No.	El campo va a contener los tipos de Contrato.
6	Costo	Campo de texto.	No.	Contiene el costo del contrato.
7	Descripción	Área de texto.	Si.	Contiene una breve descripción.
8	Estado	Lista de selección	No.	Estado inicial del contrato.

Tabla 10 Variables.

4.4.2 CU Realizar Solicitud de Compra.

El Caso de Uso se inicia cuando el Director Administrativo accede al modelo de solicitud de compra, se muestra la interfaz correspondiente para llenar la solicitud, el caso de uso termina cuando introduce todos los datos.

Condiciones de ejecución.

Debe estar creada al menos una oferta.

4.4.2.1 SC Realizar solicitud de compra.

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Respuesta del sistema	Flujo central
EC 1.1 Creación exitosa.	El escenario consiste en la entrada de todos los datos válidos y la creación de una solicitud.	V 1	V Pedido de Prueba 1	V Oferta 1	El sistema crea una nueva solicitud de contrato y muestra un mensaje de confirmación.	1-El usuario ingresa los datos correctamente.2-El usuario presiona el botón “Enviar”.
		Variable 4	**Variable 5**	**Variable 6**		
		NA 1/02/11	V Creado	V UCI, La Habana Cuba		
		Variable 7				
		V UCI,La habana,Cuba				
EC 1.2 Campos obligatorios en blanco.	El escenario consiste en que faltan algunos campos obligatorios.	I [vacío]	I Pedido de Prueba 1	I [vacío]	El sistema muestra un mensaje de error, por los campos en blanco.	1-El usuario no ingresa todos los datos. 2-El usuario presiona el botón “Enviar”.
		NA [vacío]	I [vacío]	I [vacío]		
		I [vacío]				

[Las celdas de la tabla contienen V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante.]

Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	*No. Pedido*	*Campo de texto.*	*No.*	*El campo puede contener tanto letras como números.*
2	*Nombre del Pedido*	*Campo de texto.*	*No.*	*El campo puede contener tanto letras como números.*

3	*Oferta*	*Campo de selección.*	*No.*	*El campo solo va a contener las ofertas que estén creadas.*
4	*Solicitado*	*Campo de fecha.*	*Si.*	*Contiene la fecha en que fue solicitado el pedido.*
5	*Estado*	*Campo de selección.*	*No.*	*El campo va a contener los posibles estados del pedido.*
6	*Dirección de Envío*	*Área de texto.*	*No.*	*Contiene la dirección de envío de la compra.*
7	*Dirección de Facturación*	*Área de texto.*	*No.*	*Contiene la dirección de facturación de la compra.*

Tabla 11 Descripción de las variables.

4.5 Resultados obtenidos.

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 diez (10) pruebas por CU. De estas pruebas, seis (6), un 60%, arrojaron el resultado esperado, tres (3), un 30%, resultaron en no conformidades y una (1), un 10%, generó una sugerencia de cambio. A continuación se representa de forma gráfica, para su mejor comprensión los resultados obtenidos.

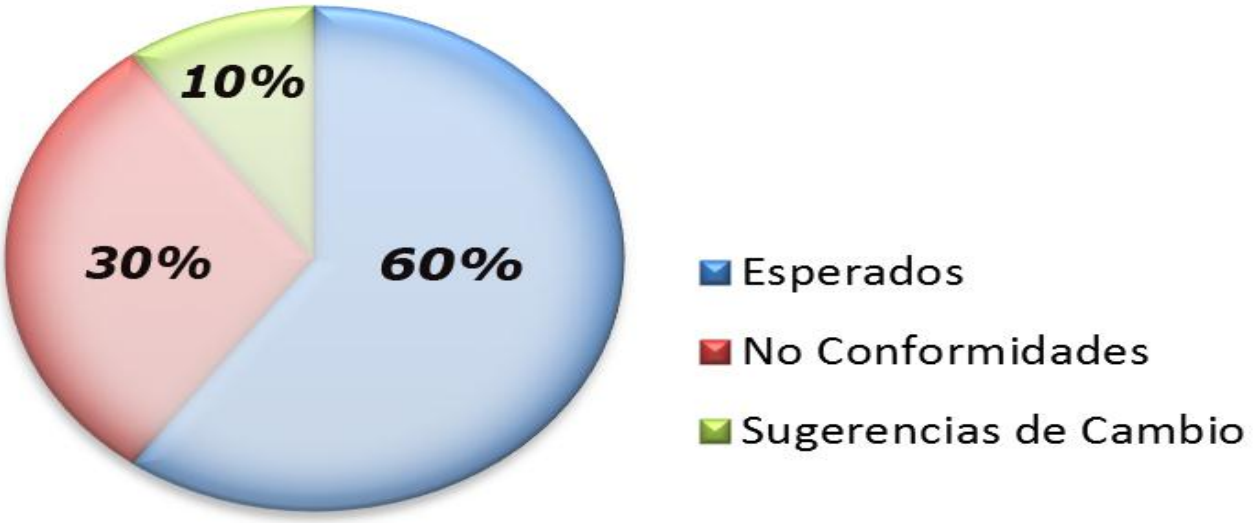


Figura 13 Resultados de las pruebas.

Conclusiones parciales.

En el capítulo concluido se han descrito aspectos referentes a la implementación de la solución. Se han presentado los diagramas de componentes de los dos CU principales del sistema. Se representa un fragmento de código de un componente referido en el diagrama de componentes del módulo Compra. Además se describen las pruebas realizadas, mediante el uso de los modelos de pruebas por casos de uso. Han quedado descritas de la misma forma las variables utilizadas para la realización de las pruebas. También se muestra de forma gráfica el resultado obtenido en la etapa de pruebas del sistema.

Conclusiones generales.

En el desarrollo de la presente investigación se realizó un estudio del estado del arte de los sistemas de gestión de proveedores en la actualidad. Tomando algunos de estos como referencia para el desarrollo de la solución, con el fin de crear una aplicación informática que, basada en los requisitos especificados, además contara con los beneficios de incluir características que los grandes sistemas de gestión estudiados no poseen.

Para desarrollar la solución que se propone, se realizó para esto un análisis en conjunto con los especialistas interesados para conformar el diseño del sistema según sus necesidades. Luego se llevó a cabo la implementación, respetando los requerimientos y realizando pruebas a nivel de desarrollador a lo largo de todo el proceso, para mantener la aplicación libre de errores.

Al concluir la etapa de implementación del sistema, este fue probado usando el método de pruebas mediante casos de uso el cual arrojó los resultados esperados.

La investigación ha cumplido con todos los objetivos planteados al inicio, se desarrolló y probó el sistema y quedó como resultado de la misma una aplicación funcional, lista para realizar las tareas por las cuales fue concebida.

Recomendaciones.

Siendo la aplicación un sistema de gestión de proveedores, es importante recomendar algunos aspectos para la obtención de un producto de mayor envergadura. A continuación se listan estos aspectos.

- ✓ Llevar a cabo el despliegue de la aplicación en el Centro de Soporte.
- ✓ Continuar con el desarrollo de una siguiente versión de la aplicación.
- ✓ Integrar la aplicación a la plataforma de gestión de servicios.
- ✓ Integrar un módulo a la aplicación para gestionar las notificaciones, no solo las de los contratos.
- ✓ Integrar a la aplicación un módulo que permita la autenticación de un usuario mediante un LDAP.

Referencias Bibliográficas

1. **EnterpiseDB.** ItilV3. *ItilV3.* [En línea] 2009. [Citado el: 16 de Febrero de 2011.] http://itilv3.osiatis.es/.

2. **Commerce, Ofice of Government.** www.itil.osiatis.com. *www.itil.osiatis.com.* [En línea] 2007. [Citado el: 6 de Diciembre de 2010.] http://wwwitil.osiatis.com.

3. **Javier Garcia de Jalón, José Ignacio Rodriguez, Iñigo Mingo, Aitor Imaz, Alfonzo Brasales, Alberto Larzabal, Jesús Calleja, Jon García.***Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000.

4. **Pérez, Javier Eguíluz.***Introducción a JavaScript.*

5. **Shing, Sang.***Introduction to Dojo Toolkit.* s.l. : Sun MicroSystems.

6. **Kenneth Barclay, John Savage.***Groovy Programming an introduction for java developers.* San Francisco : Morgan Kaufmann, 2007.

7. **PostgreSQL, Comunidad de.** www.postgresql.org. *PostgreSQL*. [En línea] Comunidad de PostgreSQL, 15 de 12 de 2010. [Citado el: 20 de 12 de 2010.] http://www.postgresql.org.

8. **Brito, Nacho.***Manual de desarrollo web con GRAILS.* 2010.

9. *Arquitectura_y_diseño_de_sistemas_web_modernos.*

10. Tu Función. [En línea] [Citado el: 14 de 1 de 2011.] http://www.tufuncion.com/mvc.

11. **Microsoft.***Service_Oriented_Architecture_ (SOA) _in_the_real_World). .* 2007.

12. **Sanchez, Maria A. Mendoza.***Metodologias de Desarrollo de Software.*

13. Sitio Oficial de Visual Paradigm. [En línea] [Citado el: 5 de 11 de 2010.] www.visual-paradigm.com.

14. Web2development. [En línea] 2007. [Citado el: 20 de 2 de 2011.] http://web2development.blogspot.com/2007/05/patron-mvc.html.

15. **Visconti, Marcello y Astudillo, Hernán.** 2010.

16. **Pressman, R.S.***Ingienería del Software. Un enfoque práctico. V ed. Vol. I. .* 2005.

17. **Jacobson, Booch,Gradu y Rumbaugh.***"El proceso unificado de desarrollo de software".* 2000.

18. **Larman, Craig.**UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.

Bibliografía

1. **Javier Garcia de Jalón, José Ignacio Rodriguez, Iñigo Mingo, Aitor Imaz, Alfonzo Brasales, Alberto Larzabal, Jesús Calleja, Jon García.** *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000.

2. **Larman, Craig.** *UML Y PATRONES. Introducción al análisis y diseño orientado a objetos.*

3. MSDN Home Page. [En línea] http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx. [En línea]

4. **Visual Paradigm Design Group.** Visual Paradigm. [En línea] 12 de Noviembre de 2009. [Citado el: 18 de 01 de 2010.] http://www.visual-paradigm.com/.

5. **PostgreSQL Fundation.** PostgreSQL - Batabase Server. [En línea] 15 de 12 de 2009. http://www.postgresql.org/.

6. **Jacobson, I., G. Booch, and J. Rumbaugh,.** *El Proceso Unificado de Desarrollo de Software.* 2000.

7. **Microsoft.** MSDN Home Page. [En línea] http://msdn.microsoft.com/es-es/library/aa291591%28VS.71%29.aspx. [En línea] 2009.

8. **Connalen, J.** *Building Web Applications with UML.* s.l. : Adison Wesley, 1999.

9. **Wordware Publishing, Inc.** *Advanced Javascript.* 2001. ISBN 1-55622-852-X.

10. **EnterpiseDB.** ItilV3. *ItilV3.* [En línea] 2009. [Citado el: 16 de Febrero de 2011.] http://itilv3.osiatis.es/.

11. **Commerce, Ofice of Government.** www.itil.osiatis.com. *www.itil.osiatis.com.* [En línea] 2007. [Citado el: 6 de Diciembre de 2010.] http://wwwitil.osiatis.com.

12. **Pérez, Javier Eguíluz.** *Introducción a JavaScript.*

13. **Shing, Sang.** *Introduction to Dojo Toolkit.* s.l. : Sun MicroSystems.

14. **Kenneth Barclay, John Savage.** *Groovy Programming an introduction for java developers.* San Francisco : Morgan Kaufmann, 2007.

15. **PostgreSQL, Comunidad de.** www.postgresql.org. *PostgreSQL.* [En línea] Comunidad de PostgreSQL, 15 de 12 de 2010. [Citado el: 20 de 12 de 2010.] http://www.postgresql.org.

16. **Brito, Nacho.** *Manual de desarrollo web con GRAILS.* 2010.

17. Unified Modeling Language TM. [En línea] www.uml.org.

18. **Microsoft.** *Service_Oriented_Architecture_ (SOA) _in_the_real_World). .* 2007.

19. **Sanchez, Maria A. Mendoza.** *Metodologias de Desarrollo de Software.*

20. Sitio Oficial de NetBeans. [En línea] [Citado el: 5 de 11 de 2010.] www.netbeans.org.

21. Sitio Oficial de Visual Paradigm. [En línea] [Citado el: 5 de 11 de 2010.] www.visual-paradigm.com.

22. **Pressman, R.S.** *Ingienería del Software. Un enfoque práctico. V ed. Vol. I. .* 2005.

23. Mundo Geek. [En línea] [Citado el: 25 de 2 de 2011.] http://mundogeek.net/archivos/2004/08/26/modelo-de-datos/.

24. *Arquitectura_y_diseño_de_sistemas_web_modernos.*

Anexos

Anexo 1 Interfaces del Sistema.

Autenticación del sistema.

Figura 14 Autenticación del sistema.

Entrada al sistema como Director Administrativo.

Figura 15 Entrada al sistema como Dtor. Administrativo.

Realizar Preforma de Contrato.

Figura 16 Realizar Preforma de Contrato.

Realizar Solicitud de Compra.

Figura 17 Realizar Solicitud de Compra.

Anexo 2 Código de las entidades Compra.groovy y Contrato.groovy.

Código de la clase entidad Compra.groovy.

```
class Compra {

    String noPedido
    String nombrePed
    String solicitado
    String estado    //empieza en Nuevo Pedido

    String direccionEnvio
    String direccionFacturacion


    Oferta oferta

    static constraints = {

        noPedido(blank:false, unique:true)
        nombrePed(blank:false)
        solicitado(blank:true,nullable:true)
        estado(inList:['Nuevo Pedido','Aceptado','Aprobado','Rechazado'])
        direccionEnvio(blank:false)
        direccionFacturacion(blank:false)
        oferta(blank:false)
    }
}
```

Figura 18 Código de la clase entidad Compra.groovy.

Código de la clase entidad Contrato.groovy.

```
1
2  class Contrato {
3      String numero
4      String nombre
5      Cliente cliente
6      Proveedor proveedor
7      String tipo
8      double costo
9      String descripcion
10     String estado
11     Date inicia
12     Date concluye
13     List notificaciones //Personas, los nombres insertados
14     int dias //dias antes de los que se debe notificar al vencerce el contrato
15
16     static constraints = {
17         numero(blank:false)
18         nombre(blank:false)
19         tipo(inList:['Anexo','Suplemento','Acuerdo de nivel de servicio','Ordinario','ALBA'])
20         estado(inList:['Creado','Aceptado','Aprobado','Aplazado','Cerrado'])
21         costo(blank:false)
22         notificaciones(blank:true,nullable:true)
23         descripcion(blank:true,nullable:true,maxSize:255)
24         cliente(blank:true,nullable:true)
25         proveedor(blank:true,nullable:true)
26         file(blank:true,nullable:true)
27         inicia(blank:false, nullable:false)
28         concluye(blank:false, nullable:false, validator:{val, obj->
29             if(val.equals(obj.inicia)){
30                 return false}})
31         dias(blank:true,nullable:true)
32     }
33
34     String toString(){
35         nombre
36     }
37     boolean notificable(user){
38         Date hoy = new Date()
39         def resta = concluye.minus(hoy)
40         if(resta<=dias && esta(user))
41            return true
42         return false
43     }
44
45     void paraConcluir(){
46         Date hoy = new Date()
47         conc = concluye.minus(hoy)
48         if(conc == 0)
49            estado = "Cerrado"
50     }
51
52     boolean esta(cont){
53         for(i in notificaciones){
54             if(i == user){
55                 return true
56                 break
57             }else{
58                 return false
59                 break
60             }
61         }
62     }
63
```

Figura 19 Código de la clase entidad Contrato.groovy.