Universidad de las Ciencias Informáticas

Facultad 3

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Título: Módulo para el manejo de recursos en los eventos y ferias comerciales que tramita la Cámara de Comercio de Cuba.

Autor
Alexander González Santos

Tutor
Ing. Yunier Pérez Barroso

La Habana, Junio 2012

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autor: Alexander González Santos.

Tutor: Ing. Yunier Pérez Barroso.

_____

Firma del Autor

_____

Firma del Tutor

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

Dedicatoria

A las dos personas que más quiero,

Mi madre y Yudiel.

Agradecimientos

A mi madre porque sin ella no habría llegado a ser quien soy, siempre lo ha dado todo para verme feliz y triunfante en la vida.

A Yudiel, por haber brindado todo su conocimiento para realizar esta investigación además de ser mi fuente de inspiración y la persona que constantemente ha deseado lo mejor para mí.

A Risdel y mi hermano por todo el apoyo que me brindaron durante mis años de universitario.

A Maritza e Isidoro, por ser más que mis amigos, siempre me han dado muy buenos consejos y han sido incondicionales en cada momento que he necesitado de ellos.

A mis abuelos Algedo y Martha, por haber estado pendientes de cada uno de mis pasos en la universidad.

A mi abuela Minerva, que en los momentos más difíciles fue imprescindible para salir adelante.

A mi prima Dayana y su familia, que continuamente me extendieron su mano y brindaron su ayuda.

A toda mi familia, que siempre que tuvo la oportunidad puso su granito de arena para que pudiera concluir mis estudios universitarios.

A mis compañeros de grupo, especialmente a mi amiga Yudith, por su gran apoyo durante toda la carrera, fundamentalmente en los momentos que todo parecía perdido.

A los padres de Yudiel, Ovidio y Oilda, por su constante ánimo y confianza en mí.

A mis profesores, especialmente a Raúl Velázquez, Yanet Vega, Lisset de Armas y mi tutor Yunier Pérez.

A todos los que de una forma u otra hicieron posible el desarrollo de esta investigación.

Resumen

El progresivo desarrollo de las tecnologías de la información ha revolucionado la forma de trabajar de muchas empresas en el mundo y en Cuba, facilitando considerablemente los procesos que en ellas se realizan. La Cámara de Comercio es una asociación de empresas vinculadas al comercio, la industria y los servicios, con reconocimiento ante los organismos del Estado; que se encarga entre otras funciones de la gestión de las ferias y eventos que se desarrollan en el país.

Con el desarrollo de este trabajo se pretende proveer, a esta institución del Estado Cubano, un producto de software capaz de suplir las necesidades que requiere el cliente para realizar su trabajo lo mejor posible. Esta investigación consta de tres capítulos, en el primero se hace un estudio del estado del arte de los software de gestión de ferias y eventos, metodología de desarrollo, patrones arquitectónicos y de diseño, paradigmas de programación, así como de las herramientas y tecnologías a utilizar. El segundo capítulo describe minuciosamente cómo funciona el negocio, además del análisis y diseño del producto. El tercer y último capítulo abarca lo relacionado con la implementación y las pruebas de calidad de software.

Índice

Índice de Figuras

Índice de Tablas

Introducción

El creciente desarrollo de las tecnologías de la información ha propiciado una revolución en la forma de trabajar de muchas empresas en el mundo, facilitando considerablemente los procesos que en ellas se realizan. En Cuba los representantes de las instituciones gubernamentales están conscientes de las grandes facilidades que brinda la informatización de la mayoría de sus empresas, por lo que el gobierno tiene como política, el desarrollo de las soluciones informáticas necesarias para lograr este propósito.

La Cámara de Comercio de la República de Cuba, es una asociación de empresas vinculadas al comercio, la industria y los servicios, con reconocimiento ante los organismos del Estado, la que permite orientar mejores alternativas para el desarrollo de la actividad empresarial de las entidades que la integran. Además constituye una herramienta para la reinserción de la economía cubana en el mundo de las relaciones económicas internacionales, pues potencia e intercambia información valiosa en torno a las posibilidades de negocios a escala mundial. (Cámara de Comercio de la República de Cuba, 2009).

En esta institución existen dificultades con la organización, ejecución y seguimiento de las ferias y encuentros comerciales, específicamente relacionados con la recepción y procesamiento de la información asociada a estos eventos. Los problemas de mayor relevancia se sustentan en: la información que se recibe de las empresas que participan en los eventos, dificulta la identificación de intereses comunes y por tanto se pierden oportunidades de negocio; además existe un limitado seguimiento a los acuerdos comerciales, debido a que solo se mantiene un registro de los acuerdos entre entidades cubanas, lo que provoca una deficiente valoración de los resultados integrales del evento; y es inadecuada la planeación de las rondas de negocio, siendo difícil coordinar manualmente las citas cuando están involucrados varios participantes.

Actualmente la Cámara de Comercio cuenta con un sistema de gestión que no satisface completamente las necesidades. Dicha aplicación no permite ejecutar correctamente el proceso de asignación de recursos para que este sea configurable, ya que no es posible establecer la estructura de los recintos para los eventos, lo cual dificulta su asignación adecuada, por otra parte, imposibilita que se agreguen nuevos

recursos o que sean modificados los ya existentes y el número de recursos que permite asignar es limitado.

Dicho software no se integra con otros sistemas existentes en otras instituciones del país para la gestión de ferias y encuentros comerciales, ni con el sistema de gestión contable del que dispone la Cámara de Comercio, lo cual requiere la generación manual de las facturas; además no permite calcular el monto del cobro, a los participantes en los eventos, de manera automática y atendiendo a las reglas definidas al respecto, lo que obstaculiza la determinación de los costos de los eventos afectando el proceso de planificación del presupuesto para estos (Vega Miniet, 2012).

Los representantes de la Cámara de Comercio, como resultado de las carencias que presenta el actual sistema de que disponen, eventualmente han tenido que optar por el uso de software que otras empresas han desarrollado para la gestión de las ferias y eventos comerciales. La utilización de esta opción genera gastos monetarios extras.

A partir de la **problemática** descrita anteriormente se define como **problema a resolver**: ¿Cómo erradicar las ineficiencias en el control de las actividades de asignación de recursos en la Cámara de Comercio de Cuba, para que se mitigue la pérdida de oportunidades de negocio y mejore la planificación del presupuesto de las ferias y eventos comerciales?

Para dar solución al problema, se definió como **objeto de estudio:** proceso de desarrollo de software. El **objetivo general** es desarrollar un módulo de manejo de recursos en la Cámara de Comercio de manera que se mitigue la pérdida de las oportunidades de negocio y mejore la planificación del presupuesto de las ferias y eventos comerciales. El **campo de acción:** análisis, diseño e implementación de sistemas de gestión.

La investigación defiende la siguiente **idea**: el desarrollo de un módulo de manejo de recursos contribuye a mitigar la pérdida de oportunidades de negocios y mejora la planificación del presupuesto de las ferias y eventos comerciales. En función de cumplir con el objetivo general se trazaron los **objetivos específicos** siguientes:

- Fundamentar teóricamente el desarrollo de módulos de manejo de recursos.
- Efectuar la captura de requisitos del proceso de asignación de recursos de las ferias y eventos comerciales cubanos.

- Realizar el análisis diseño e implementación de los requisitos obtenidos.
- Verificar y validar la solución propuesta.

Para cumplir con los objetivos mencionados anteriormente se propone un grupo de **tareas de la investigación** que servirán de guía, las cuales son:

- Análisis de las herramientas más utilizadas a nivel mundial para el desarrollo de este tipo de aplicaciones.
- Selección de las herramientas a utilizar en el desarrollo.
- Captura de requisitos del modulo de manejo de recursos.
- Diseño del módulo para el manejo de recursos en las ferias y eventos comerciales.
- Elaboración de los artefactos necesarios para la implementación.
- Implementación del módulo para el manejo de recursos en las ferias y eventos comerciales.
- Realización de pruebas de calidad de software.

Los métodos de investigación constituyen el conjunto de acciones reglamentadas que posibilitan el avance en el conocimiento del objeto de estudio, hasta lograr resolver el problema.

Se emplean como métodos teóricos:

El método **analítico – sintético** sintetiza los elementos más importantes que se relacionan con el proceso de desarrollo de software a partir de tendencias y documentos relacionados con el tema; expresando de manera resumida la posición del investigador.

El análisis **histórico – lógico** para realizar un estudio crítico de los sistemas de gestión de recursos existentes en este contexto y utilizarlos como punto de referencia y comparación, además de constatar teóricamente cómo ha evolucionado el tema en el tiempo.

La **modelación** de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la solución que facilite así el desarrollo de la misma.

Se emplean como métodos empíricos:

La **entrevista** a funcionarios de la Cámara de Comercio con el objetivo de obtener información de interés para poder comprender el funcionamiento del proceso de contratación, específicamente de las tareas relacionadas con el manejo de los recursos.

La **observación** del proceso de contratación para lograr un mejor entendimiento, comprensión y caracterización del mismo.

La **medición** de los atributos de calidad del módulo a través del uso de pruebas y métricas de calidad.

En función de cumplimentar a los objetivos de esta investigación, la misma ha sido estructurada de la siguiente manera:

En el capítulo 1 se realiza un estudio del estado del arte de los sistemas de gestión de ferias y eventos en países foráneos y en Cuba. Se hace alusión a la metodología de desarrollo de software, los estilos arquitectónicos, patrones de diseño y paradigmas de programación, así como las herramientas, tecnologías y elementos a tener en cuenta para la verificación y validación que serán usados para desarrollar un software con la calidad requerida.

El capítulo 2 describe todos los aspectos relacionados con el modelado del negocio, el análisis y diseño del sistema. Se pueden observar el diagrama de procesos de negocio, casos de uso del sistema, los diagramas de clases del análisis y el diseño, así como los de secuencia correspondientes a cada uno de los escenarios de los casos de uso, además del modelo de datos.

En el capítulo 3 se presentan todos los elementos afines con la implementación y las pruebas, como son los diagramas de componentes y despliegue; los casos de prueba y los resultados obtenidos en las mismas.

Capítulo 1
Fundamentación teórica

1.1 Introducción

El desarrollo de un software requiere del estudio de las tendencias existentes en el mundo para la creación de sistemas con características afines a lo que necesita el cliente. También hay que tomar en consideración la metodología de desarrollo, los patrones arquitectónicos y de diseño, los paradigmas de programación, así como las herramientas y tecnologías que son usadas con el fin de obtener el producto.

1.2 Estado del arte

En el mundo existen diversas aplicaciones en su mayoría web para la gestión de ferias y eventos entre las que podemos mencionar: Perfect Table Plan, Reg Online, Inscribete, Amiando, Expocomer, facilitando considerablemente la gestión de ferias y eventos a los planificadores y organizadores permitiéndoles un control total y una perspectiva completa de cada aspecto.

Estas aplicaciones tienen funcionalidades en común:

- Gestionar y planificar eventos.
- Gestionar y organizar participantes.
- Realizar distintos tipos de reservaciones como son: alojamiento, áreas y salas de exposiciones.
- Enviar correos personalizados, mensajes de confirmación, recordatorios, datos de inscripción y alojamiento,
- Imprimir credenciales.
- Generar reportes de invitados y participantes con información detallada de cada uno de ellos.

- Relacionado con la gestión económica del evento: realizar balances económicos, generar facturas, resúmenes de facturación e informes de rentabilidad y ganancias.

La mayoría de los sistemas web que están creados para la gestión de ferias y eventos cuentan con sistemas seguros para aceptar de sus clientes tarjetas de crédito y gestionar los procesos de pago y facturación, además de cobrar bonificaciones a las empresas que utilizan este servicio.

Por las características que presenta la infraestructura tecnológica de la Cámara de Comercio, no es factible construir un software con todas las funcionalidades mencionadas anteriormente, pues no todas serían explotadas, además la adquisición de una de estas aplicaciones traería gastos económicos al país.

En Cuba se aplica el sistema SAEFE que es el encargado de realizar este tipo de funciones, el que se caracteriza por ser poco flexible a los cambios, pues no permite la modificación dinámica de sus datos, carece de un módulo que admita ejecutar correctamente la asignación de recursos, de modo que sea posible configurar la estructura de los recintos para los eventos, dificultando la asignación adecuada de los espacios dentro de los locales e imposibilitando la inserción o modificación de un determinado recurso, y además no es integrable con el sistema de gestión contable existente en la Cámara de Comercio, ni permite determinar el costo de los eventos.

1.3 Metodología de desarrollo

Durante el proceso de desarrollo de software se deben completar una serie de tareas para obtener el producto esperado; los componentes del software deben pasar por varias fases o etapas durante su ciclo de vida. Cada tarea puede ser abordada y resuelta de múltiples maneras, con diferentes herramientas y utilizando distintas técnicas, por lo que es necesario saber cuándo podemos darla por concluida, quién debe realizarla, qué tareas preceden o anteceden a una dada, qué documentación debe ser utilizada para llevar a cabo la misma.

Para obtener un producto final exitoso se hace necesario el uso de una metodología que integra un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayuda a los que lo desarrollan a realizar un nuevo software (Piattini Velthuis, 1996); por ello es muy importante definir correctamente la metodología a seguir para desplegar el nuevo producto.

La Universidad de las Ciencias Informáticas tiene como política que todos sus proyectos se desarrollen guiados por el Programa de Mejoras de la Institución, el cual está encaminado a la certificación internacional del nivel 2 del modelo CMMI[1]. Persiguiendo este fin y dada las características del sistema, así como de la documentación que se necesita generar, se decide utilizar como metodología: Rational Unified Process (RUP).

1.3.1 Rational Unified Process (RUP)

RUP es una metodología que puede especializarse para una gran variedad de sistemas de software, diferentes áreas de aplicación, tipos de organización y tamaños de proyectos.

Rational Unified Process, es un proceso de desarrollo de software dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental; utiliza Lenguaje Unificado de Modelado (UML) como lenguaje.

RUP propone un ciclo de vida para el proceso de desarrollo dividido en cuatro fases:

- Inicio: define el modelo del negocio y el alcance del proyecto.
- Elaboración: analiza el dominio del problema, establece los cimientos de la arquitectura, desarrolla el plan del proyecto y elimina los mayores riesgos. En esta fase se construye un prototipo de la arquitectura, que debe evolucionar en iteraciones sucesivas hasta convertirse en el sistema final.
- Construcción: su propósito es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto.
- Transición: su finalidad es poner el producto en manos de los usuarios finales, para lo que se requiere: completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y facilidad de uso del producto. (Kruchten, 2000)

[1] Capability Madurity Model Integration: modelo de referencia para el crecimiento de capacidades y madurez, que se enfoca tanto en procesos de Administración como de Ingeniería de Sistemas y Software.

Dentro de cada una de estas fases se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. RUP consta de nueve flujos de trabajo agrupados en seis para el desarrollo y tres de soporte:

Desarrollo:

- Modelamiento del Negocio: Permite un estudio preliminar, lograr un mejor entendimiento de la organización donde se va a implementar el producto, comprender cómo funciona el negocio y cuáles son sus necesidades.
- Requerimientos: se establece qué tiene que hacer exactamente el sistema que se construirá.
- Análisis y diseño: traduce los requisitos a una especificación que describe cómo implementar el sistema.
- Implementación: se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás.
- Prueba: evalúa la calidad del producto que se desarrolla, pero no para aceptar o rechazar el producto al final del proceso, sino que debe ir integrado en todo el ciclo de vida.
- Despliegue: su objetivo es producir con éxito reparticiones del producto y distribuirlo a los usuarios.

Soporte:

- Configuración y gestión de cambios: su finalidad es mantener la integridad de todos los artefactos que se crean en el proceso, así como la información del proceso evolutivo que han seguido.
- Gestión de proyecto: persigue lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto acorde a los requisitos de los clientes y los usuarios.
- Ambiente: su fin es dar soporte al proyecto con adecuadas herramientas, procesos y métodos.

En la Figura 1 se muestra cómo varía el esfuerzo asociado a los flujos de trabajo según la fase en la que se encuentre el proyecto.

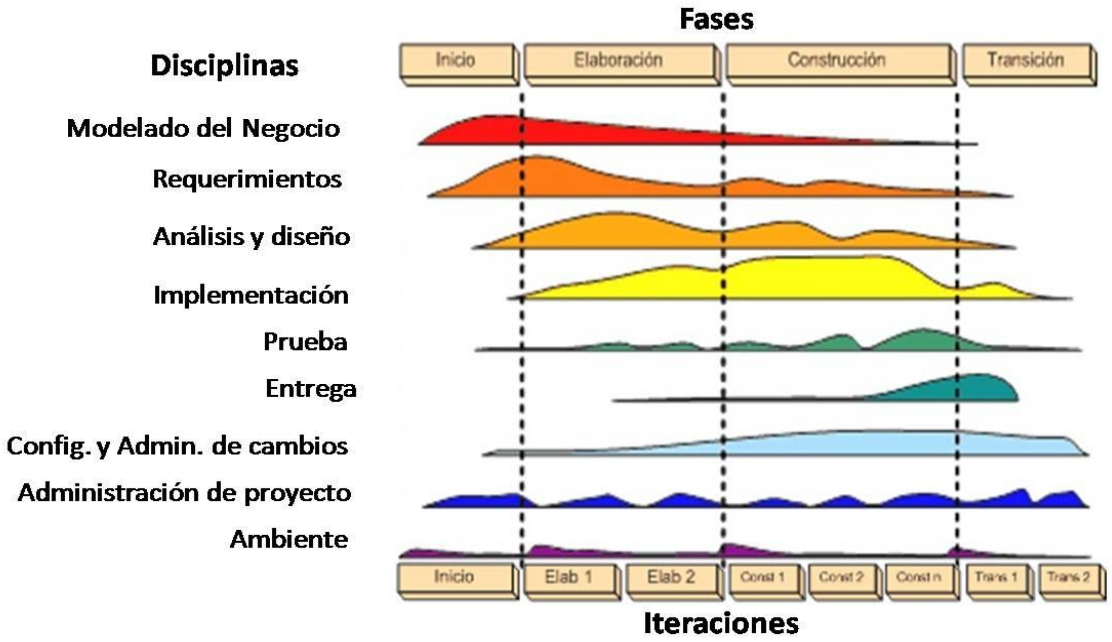

Figura 1: Proceso de desarrollo de software

1.4 Arquitectura de software

La arquitectura de software define la línea que deben seguir los procesos de análisis, diseño e implementación a través del uso de patrones, frameworks y estilos arquitectónicos que tributen a la calidad del sistema y permite a los que lo desarrollan e involucrados tener una idea clara de lo que se está implementando.

El Instituto de Ingenieros Eléctricos y Electrónicos conocido por sus siglas en inglés como IEEE define la Arquitectura del Software en el estándar Std 1471-2000, como: "la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución"(IEEE SA - 1471-2000).

La arquitectura de software tiene tres razones claves que demuestran su importancia: sus representaciones facilitan la comunicación entre todas las partes interesadas en el desarrollo del sistema, destaca decisiones tempranas de diseño que tendrán un gran impacto durante el ciclo de vida del proceso de desarrollo de software, constituye un elemento relativamente pequeño y comprensible de cómo está estructurado el sistema y cómo interactúan sus componentes.

Cuando se habla de construcción de software se cuenta con diversos estilos arquitectónicos, cada estilo describe una categoría del sistema, que contiene: un conjunto de componentes que realizan una función requerida, un conjunto de conectores que posibilitan la comunicación y cooperación entre los componentes,

restricciones que definen cómo se pueden integrar los componentes que forman el sistema y los modelos semánticos que permiten al diseñador entender las propiedades globales del mismo. Un sistema es una colección de componentes interrelacionados que trabajan conjuntamente para cumplir algún objetivo. (Olguín Espinoza, 2004)

"Un patrón arquitectónico expresa un esquema de organización estructural, fundamental para los sistemas de software, proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades, e incluye normas y directrices para la organización de las relaciones entre ellos"(Cunningham, 2003).

Uno de los patrones arquitectónicos más usados en el desarrollo de software es el Modelo Vista Controlador.

1.4.1 Modelo Vista Controlador (MVC)

El MVC propone un diseño que desacople la vista del modelo, con la finalidad de mejorar la reusabilidad. De esta forma las modificaciones en las vistas impactan en menor medida en la lógica de negocio o de datos(Lago Bagués, 2007).

Elementos del patrón:

- Modelo: está mayormente relacionado con los datos.
- Vista: muestra la información del modelo al usuario.
- Controlador: gestiona las entradas del usuario.

Cada uno de los elementos que componen el modelo tiene sus responsabilidades como se describen a continuación:

Modelo: Administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y a instrucciones de cambiar el estado (habitualmente desde el controlador). No depende de ninguna vista y de ningún controlador.

Vista: maneja la visualización de la información.

Controlador: Recibe los eventos de entrada (acciones del ratón y el teclado) informando al modelo y/o a la vista para que cambien según resulte apropiado, contiene las reglas de gestión de eventos.

1.4.2 Arquitectura en capas

Las arquitecturas de n-capas proporcionan una gran cantidad de beneficios para las empresas que necesitan soluciones flexibles y fiables para resolver complejos problemas inmersos en cambios constantes.

Esta arquitectura se define como una organización jerárquica donde cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Los componentes de cada capa consisten en conjuntos de procedimientos; las interacciones entre las mismas, usualmente proceden por invocación de dichos procedimientos (Reynoso, 2004). Generalmente los niveles de abajo no usan funcionalidades ofrecidas por los de arriba.

1.5 Diseño de software

El diseño es una actividad en la que se toman decisiones importantes, frecuentemente de naturaleza estructural. Comparte con la programación un interés por la abstracción de la representación de la información y de las secuencias de procedimiento, pero el nivel de detalles es muy diferente en ambos casos. El diseño construye representaciones coherentes y bien planificadas de los programas, concentrándose en las interrelaciones de los componentes de mayor nivel y en las operaciones lógicas implicadas en los niveles inferiores(Freeman, 1980).

1.5.1 Patrones de diseño

Siempre que se piensa en los patrones de diseño no se puede dejar de tener en cuenta las palabras pronunciadas por el actual director científico de Rational Software y diseñador de software Grady Booch cuando dijo: “Una arquitectura orientada a objetos bien estructurada está llena de patrones. La calidad de un sistema orientado a objetos se mide por la atención que los diseñadores han prestado a las colaboraciones entre sus objetos. Los patrones conducen a arquitecturas más pequeñas, más simples y más comprensibles”.

Los patrones que se mencionan a continuación serán usados en la arquitectura del sistema:

Singleton: asegura que una clase tiene sólo una instancia y proporciona un punto de acceso global a ella.

Inversión de Control (IoC): se delega en un componente o fuente externa la función de seleccionar un tipo de implementación concreta de las dependencias entre las clases (IoC).

Bajo acoplamiento: se basa en la idea de que la asignación responsabilidades se realice de manera que el acoplamiento sea bajo, en otras palabras que la dependencia entre las clases sea la menor posible, permitiendo así, la creación de clases más independientes y reutilizables.

Alta cohesión: las responsabilidades de una clase tiene que estar lo más estrechamente relacionadas posible, trae como ventaja un código más fácil de comprender, reutilizar y mantener, además de ser afectado lo menos posible ante los cambios.

1.6 Paradigmas de Programación

Un paradigma según el libro The Structure of Scientific Revolutions escrito por Thomas Kuhn se define como un conjunto de teorías, estándares y métodos que juntos representan una forma de organizar el conocimiento(Kuhn, 1971).

1.6.1 Paradigma Orientado a Objetos (POO)

El Paradigma Orientado a Objetos encierra una completa visión de la ingeniería del software. Los beneficios de la tecnología orientada a objetos se fortalecen si se usa antes y durante el proceso de ingeniería del software, esta tecnología debe hacer sentir su impacto en todo el proceso de desarrollo del software. Un simple uso de la programación orientada a objetos no brindará los mejores resultados.

La POO simplifica la programación con la nueva filosofía y nuevos conceptos que tiene, se basa en dividir el programa en pequeñas unidades lógicas de código llamadas objetos; los objetos son unidades independientes que se comunican entre ellos mediante mensajes.

Los conceptos básicos del modelo orientado a objetos son: Objetos, clases, herencia, polimorfismo, envío de mensajes y se caracteriza por: la abstracción, el encapsulamiento y el ocultamiento.

El uso de la POO ofrece numerosas ventajas, así como también algunas desventajas.

Ventajas:

- **Reusabilidad**. Cuando se han diseñado adecuadamente las clases, pueden ser usadas en distintas partes del programa y en numerosos proyectos.
- **Mantenibilidad**. Debido a la sencillez para abstraer el problema, los programas orientados a objetos son más sencillos de leer y comprender, pues permiten ocultar detalles de implementación dejando visibles sólo aquellos que resultan más relevantes.
- **Modificabilidad**. La facilidad de añadir, suprimir o modificar nuevos objetos permite hacer reformas de una manera muy sencilla.
- **Fiabilidad**. Al dividir el problema en partes más pequeñas se pueden probar de manera independiente y aislar mucho más fácilmente los posibles errores que puedan surgir.

Desventajas:

- Cambio en la forma de pensar de la programación tradicional a la orientada a objetos.
- La ejecución de programas orientados a objetos es más lenta.
- La necesidad de utilizar bibliotecas de clases obliga a su aprendizaje y entrenamiento.

1.6.2 Paradigma Orientado a Aspectos (POA)

La Pogramación Orientada a Aspectos (POA) es un paradigma de programación que aspira a soportar la separación de competencias para los aspectos específicos y de propósito general. Es decir, que intenta separar los componentes y los aspectos unos de otros, proporcionando mecanismos que hagan posible abstraerlos y componerlos para formar todo el sistema (Quintero, y otros, 2000).

La POA es un desarrollo que sigue al paradigma de la orientación a objetos, y como tal, soporta la descomposición orientada a objetos, además de la procedimental y la descomposición funcional. Pero, a pesar de esto, POA no se puede considerar como una extensión de la POO, ya que puede utilizarse con los diferentes estilos de programación.

"Un aspecto es una unidad modular que se disemina por la estructura de otras unidades funcionales. Los aspectos existen tanto en la etapa de diseño como en la de implementación. Un aspecto de diseño es una unidad modular del diseño que se entremezcla en la estructura de otras partes del diseño. Un aspecto de programa o de código es una unidad modular del programa que aparece en otras unidades modulares del programa"(Kiczales, y otros, 1997).

Los objetivos principales de este tipo de programación son: separar conceptos para conseguir que cada cosa esté en su sitio, es decir, que cada decisión se tome en un lugar concreto y minimizar las dependencias entre ellos buscando una pérdida del acoplamiento entre los distintos elementos.

El paradigma POA también ofrece ventajas y limitaciones.

Ventajas:

- Código más limpio, fácil de entender y mantener, menos duplicado, más natural y más reducido.
- Mayor facilidad para razonar sobre las materias, debido a que están separadas y tienen una dependencia mínima.
- Mayor facilidad para depurar y hacer modificaciones en el código.
- Logra que un conjunto grande de modificaciones en la definición de una materia tenga un impacto mínimo en las otras.
- Código más reusable que permite el acoplamiento y desacoplamiento cuando sea necesario.

Limitaciones:

- Uso de un lenguaje base existente, lo que provoca en el caso de los lenguajes de dominio específico que no es completamente libre para diseñar los puntos de enlace sino que hay que restringirse a aquellos que pueden ser identificados en el lenguaje base.
- Restricciones impuestas sobre el lenguaje base, son muy difíciles de lograr pues se deben eliminar elementos de un sistema complejo, donde cada uno tiene su lugar y responsabilidades.

- Posibles choques entre los aspectos, dos aspectos pueden trabajar perfectamente por separado pero al aplicarlos conjuntamente resultan tener un comportamiento anormal.

La mayor parte de la implementación se realizará enfocada al paradigma de programación orientado a objetos, a no ser en lo relacionado con el framework Spring que hace uso del paradigma orientado a aspectos con el fin de que cada elemento tenga la responsabilidad que le corresponde y las decisiones sean tomada en el momento preciso minimizando así las dependencias.

1.7 Herramientas y tecnologías

Entre las herramientas y tecnologías a utilizar en el desarrollo de la solución informática se encuentran:

1.7.1 CASE (Computer Aided Software Engineering)

Las herramientas CASE (en español Ingeniería de Software Asistida por Computadora) ayudan a los gestores y practicantes de la ingeniería del software en todas las actividades asociadas a los procesos de software, automatizan las actividades de gestión de proyectos, gestionan todos los productos de los trabajos elaborados a través del proceso, y ayudan a los ingenieros en el trabajo de análisis, diseño y codificación. Las herramientas CASE se pueden integrar dentro de un entorno sofisticado (Pressman, 2002). Dentro de este tipo de herramientas está Visual Paradigm la cual será usada para el modelado de los diagramas necesarios para el desarrollo del software.

Visual Paradigm (VP) es una herramienta profesional, robusta y colaborativa[2] que soporta el ciclo de vida del proceso de desarrollo de software, permitiendo organizar y controlar el desarrollo, además de representar todos los tipos de diagramas de clases, generar código y documentación desde diagramas. VP soporta notación UML, tiene la capacidad de realizar ingeniería directa e inversa, es integrable con los IDEs[3] más usados como NetBeans, posee una licencia gratuita y comercial y está disponible en múltiples versiones y plataformas. Actualmente la suite VP se encuentra en su versión 5.0 y es la que será usada para el modelado de los artefactos necesarios para el desarrollo del nuevo producto de software.

[2] soporta múltiples usuarios trabajando sobre el mismo proyecto.
[3] Integrated development environment (Entorno de Desarrollo Integrado).

1.7.2 Business Process Modeling Notation

Business Process Modeling Notation (BPMN) es un estándar para el modelado de procesos de negocio y servicios web, es una notación a través de la cual se expresan los procesos de negocio en un diagrama de éstos; agrupa la planificación y gestión del flujo de trabajo, así como el modelado y la arquitectura (Analitica). BPMN proporciona un lenguaje gráfico común, con el fin de facilitar su comprensión a los usuarios de negocios, integra las funciones empresariales. Su función fundamental es crear un mecanismo simple para realizar modelos de procesos de negocio, con todos sus elementos gráficos, y que al mismo tiempo sea posible gestionar la complejidad de los mismos, con este fin se hará uso de esta notación en su versión 2.0.

1.7.3 Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas más conocido y usado en la actualidad; está concebido como el estándar internacional aprobado por la Object Management Group (OMG). Se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software"(Booch, y otros, 1997).

UML es un conjunto de notaciones estándares destinadas a los sistemas de modelado que utilizan conceptos orientados a objetos, incluye aspectos conceptuales tales como procesos de negocio y funciones del sistema, proporciona un vocabulario y reglas que permiten la comunicación. Está compuesto por elementos que no son más que abstracciones que constituyen los bloques básicos de construcción, los cuales pueden unirse mediante relaciones para conformar los diagramas (Fowler, 1997).

UML es completamente independiente del lenguaje de implementación, de tal forma que los diseños realizados usando este tipo de modelado pueden ser implementados en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos); permite la automatización de determinados procesos y la generación de código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). En la elaboración de los diagramas que sean necesarios se hará uso de UML en su versión 8.0.

1.7.4 Lenguaje de programación. Java.

Java es un lenguaje de programación pensado de forma tal que sea sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portátil, de

gran rendimiento, multitarea y dinámico que permite a los desarrolladores realizar cualquier tipo de programación (Zukowski, 2003).

Orientado a objetos: Tiene una colección de componentes reutilizables, extensibles y sostenibles llamados bibliotecas de clases que van a permitir desarrollar software describiendo problemas mediante el uso de elementos u objetos desde el espacio del problema y no mediante un conjunto de pasos secuenciales que se ejecutarán en el ordenador.

Distribuido: Proporciona las librerías y herramientas para que Java pueda acceder a objetos repartidos mediante protocolos estándares basados en TCP/IP como HTTP, invocar métodos en un equipo remoto tan fácil e invisible como podría hacerlo en su mismo espacio de ejecución mediante protocolos comunes tales como CORBA y RMI. Para cada protocolo, el sistema se encarga automáticamente de toda la conversión y transporte.

Interpretado: El código Java en lugar de ser compilado en ejecutables nativos, es traducido en códigos de bytes que se pueden transferir a cualquier plataforma que tenga Java Runtime Environment (JRE), que consiste en una Máquina Virtual de Java (JVM) y de este modo pueden ejecutarse sin volver a compilarlos.

Robusto: Contiene varias funciones integradas que mejoran la fiabilidad de un programa como son: lenguaje basado en tipos, no existencia de punteros, recolección automática de elementos no utilizados, formatea el uso de interfaces en lugar de clases.

Seguro: Presenta varias funciones de seguridad para asegurar la correcta construcción y ejecución de programas distribuidos, entre ellas se pueden mencionar: el verificador de código de bit, el cargador de clases y el gestor de seguridad, garantizando así la certeza del código que se está ejecutando y evitando que el código no seguro realice operaciones seguras, además de las relacionadas con la autenticación, autorización y encriptación para proteger la privacidad y asegurar la integridad de los datos.

Este lenguaje será usado para desarrollar el nuevo producto de software no sólo por las ventajas que propicia al desarrollador sino porque está en consonancia con uno de los requerimientos no funcionales del sistema.

1.7.5 Entorno de Desarrollo Integrado. Netbeans

NetBeans IDE (Integrated Development Environment) es un entorno de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, Groovy y Grails, y C / C + +. Es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Existen además un número importante de módulos para extender este IDE; es un producto libre y gratuito sin restricciones de uso; su versión 7.1 tiene las características mencionadas por lo que se decide seleccionarla para desarrollar el software.

1.7.6 Plataforma de desarrollo. JEE

La plataforma de desarrollo Java EE (del inglés: Java Platform Enterprise Edition), es un estándar para el desarrollo de aplicaciones empresariales que sean portables, robustas, escalables y seguras usando tecnología Java.
Java EE consta de una API para el acceso a bases de datos y un modelo de seguridad que protege los datos incluso en aplicaciones de Internet. También apoya las nuevas tecnologías de servicios web y asegura la interoperabilidad de estos. Todo con gran sencillez, portabilidad, escalabilidad e integración.

Máquina Virtual de Java (JVM): es necesaria para poder ejecutar los programas escritos en lenguaje java, se encuentra disponible para diversos sistemas operativos, como son Mac OS X, Windows, y diversas distribuciones de Linux. La funcionalidad de la JVM es interpretar los programas de Java, transformarlos a lenguaje máquina para la PC, y así la misma puede ejecutar el programa; un aspecto que debe quedar claro es que nunca se ejecuta directamente un programa de código Java, si no que ejecuta la máquina Virtual, y esta interpreta el programa pre-compilado. Se hará uso de versión JDK 1.6.

1.7.7 Java Persistence API

Java Persistence API (JPA) proporciona un modelo de persistencia basado en POJO's (Plain Old Java Objects) para mapear bases de datos relacionales en Java. El mapeo objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML. Este tipo de notación puede ser utilizada por

varios IDEs como Netbeans pues existen las herramientas y plugin necesarios para generar clases de entidad con notaciones JPA.

JPA 2.0 es una especificación de la cual se encuentran disponibles varias implementaciones, las más populares son Hibernate, EclipseLink y OpenJPA Apache, siendo EclipseLink su implementación de referencia (Vogel, 2012) .

1.7.8 Sistema gestor de base de datos

Un Sistema Gestor de Bases de Datos (SGBD, en inglés DBMS: Data Base Management System) es una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos, de manera que sea tanto práctica como eficiente. Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización (Silberschatz, y otros, 2002).

Los SGBD permiten: un grado de abstracción que no se necesita saber detalles acerca del almacenamiento físico de la información; la independencia entre las bases de datos y las aplicaciones que se sirven de ellas; la no existencia de redundancia en la información; consistencia entre los datos que tengan relación; existencia de permisos para garantizar la seguridad de los datos; integridad de la información almacenada, sistemas de respaldo y recuperación; y minimizar el tiempo en proporcionar la información solicitada y almacenar los cambios realizados.

PostgreSQL: es un potente SGBD relacional de código abierto, distribuido bajo licencia BSD[4]. Se ejecuta en los principales sistemas operativos, es totalmente compatible con ACID[5], tiene soporte completo para claves foráneas, uniones, vistas, triggers y procedimientos almacenados, además soporta almacenamiento de objetos binarios grandes, incluyendo imágenes, sonidos o vídeo, cuenta con interfaces nativas

[4]Berkeley Software Distribution, licencia de software libre muy cercana al dominio público.
[5]Acrónimo de Atomicity, Consistency, Isolation and Durability: en español Atomicidad, Consistencia, Aislamiento y Durabilidad.

de programación C/C + +, Java, NET, Perl, Python, Ruby, Tcl, ODBC[6], entre otros (PostgreSQL Fundation).

Una base de datos de clase empresarial, PostgreSQL en su versión 9.1, cuenta con características avanzadas tales como control de concurrencia, replicación asincrónica, transacciones anidadas, copias de seguridad, es compatible con conjuntos de caracteres internacionales, altamente escalable tanto en la gran cantidad de datos que puede manejar como en el número de usuarios concurrentes que puede acomodar.

1.7.9 Framework

Un framework o infraestructura digital es una estructura de soporte definida en la cual otro proyecto puede ser organizado y desarrollado. Un framework puede incluir soporte de programas y librerías para ayudar a desarrollar y unir los diferentes compontes de un proyecto (Barbosa Guerrero, 2006).

El uso de frameworks durante el desarrollo de aplicaciones brinda numerosas ventajas entre las que se encuentran: una mayor velocidad en el desarrollo de aplicaciones, código optimizado y reducción de costos; la mayor parte de ellos se basan en estándares.

Spring: es un framework que proporciona una infraestructura de código abierto que actúa de soporte para desarrollar aplicaciones Java. Presenta una serie de características que lo hacen interesante a la hora de definir una arquitectura, entre ellas: se enfoca en el manejo de objetos de negocio dentro de una arquitectura en capas; es modular pudiendo usar algunos de sus módulos sin comprometerse con el resto, su Contendor de Inversión de Control es el núcleo del sistema, se encarga de instanciar los objetos y de las dependencias existentes entre ellos. Soporta la integración con la tecnología JPA.

EclipseLink: Como se mencionó con anterioridad EclipseLink es la implementación de referencia de JPA, la misma facilita el almacenamiento, la asignación, actualización y recuperación de datos de bases de datos relacionales a objetos Java y viceversa; permitiendo al desarrollador trabajar directamente con los objetos en lugar de con sentencias SQL (Vogel, 2012). La implementación de JPA se suele llamar proveedor de persistencia y normalmente define los metadatos a través de notaciones en la clase

[6]Open Data Base Connectivity: estándar de acceso a bases de datos.

Java. El lenguaje de consultas es similar al de SQL. Su versión 2.3.0 se encuentra entre las herramientas seleccionadas para el exitoso desarrollo de la aplicación.

1.8 Verificación y validación

La verificación y validación del software son procesos similares pero que persiguen intenciones diferentes, con la verificación se asegura que se construya correctamente el producto mientras que con la validación se cerciora que el producto obtenido sea el correcto.

Esta disciplina está orientada a evaluar y valorar el producto de software en cada fase del ciclo de vida, asegura que la calidad sea incorporada al producto durante el desarrollo, así como que se cumplieron los requisitos del cliente; tiene como objetivos: encontrar defectos en el software tan pronto como se pueda y determinar si las funciones y atributos requeridos se incorporaron al sistema.

La verificación es la encargada de comprobar que el producto de casa fase es: correcto, integro, consistente, que además satisface los estándares, prácticas, convenciones de la fase y establece las bases adecuadas para iniciar la siguiente fase; tiene como responsable al equipo de desarrollo. La validación se encarga de certificar que el producto totalmente finalizado cumple con los requisitos del software y el sistema, siendo su responsable el cliente. (Hernández Aguilar, 2010)

Para evaluar y validar un producto de software existen varias técnicas y métodos como los que se mencionan a continuación:

- Análisis o evaluación estática: evalúa el sistema considerando su forma, estructura, contenido o documentación.
- Pruebas estáticas: examinan un programa sin ejecutarlo a través de inspecciones y revisiones de código.
- Pruebas matemáticas: Técnica formal que prueba matemáticamente que un programa satisface sus requisitos
- Pruebas o evaluaciones dinámicas: pueden realizarse a través del análisis de código fuente y generación de casos de pruebas, considerando cobertura, pruebas de condiciones o flujo de datos y el análisis de requisitos y casos de pruebas que verifican los requisitos.
- Prototipos y simulación.

La evaluación es la determinación sistemática de hasta qué grado una entidad (empresa, software, etc.) cumple los requisitos especificados. [ISO 8402: 1994]. La evaluación va dirigida a dos elementos fundamentales: la aplicación y la documentación asociada a ella. Existen dos tipos de evaluaciones, las estáticas y las dinámicas.

Evaluación estática: busca faltas sobre el sistema en reposo, estudia los distintos modelos que componen el sistema de software buscando posibles errores en los mismos. Así pues, estas técnicas se pueden aplicar a: requisitos, modelos de análisis, diseño y código es decir a toda la documentación asociada a la aplicación. Se les conoce de modo genérico por Revisiones. Las revisiones pretenden detectar manualmente defectos en cualquier producto del desarrollo (Hernández Aguilar, 2010). Manualmente quiere decir que el producto en cuestión (sea requisito, diseño, código, etc.) es analizado mediante la lectura del mismo, sin ejecutarlo. La herramienta más usada: Listas de Chequeo.

Evaluación dinámica: genera entradas al sistema con el objetivo de detectar deficiencias, cuando el sistema ejecuta dichas entradas. Los fallos se observan cuando se detectan incongruencias entre la salida esperada y la salida real. La aplicación de técnicas dinámicas es también conocida como pruebas de software o testing y se aplican generalmente sobre código puesto que es, hoy por hoy, el único producto ejecutable del desarrollo.

Cuando se desea probar un software es muy importante tener bien definida una estrategia de prueba, la misma describe el enfoque y los objetivos generales de las actividades de prueba, incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada, las técnicas y los casos de prueba diseñados para lograr los objetivos; define: mecanismos de pruebas (manual o automática), herramientas a ser usadas, criterios de éxitos y culminación de la pruebas.

1.9 Conclusiones del capítulo

Con el desarrollo de este capítulo se define la metodología, los estilos arquitectónicos, patrones de diseño y paradigmas de programación, así como las herramientas, tecnologías y elementos a tener en cuenta para la verificación y validación que serán usados para el desarrollo de la solución informática que necesita la Cámara de Comercio de Cuba para la gestión de las ferias y eventos comerciales que se realizan en nuestro país.

Capítulo 2 Descripción de la propuesta de solución

2.1 Introducción

La creación de un nuevo producto de software requiere que exista por parte del equipo de desarrollo un amplio conocimiento del negocio que se desea informatizar. Es muy importante que se realice un adecuado levantamiento de requisitos de software que contribuya a la identificación de los casos de uso y sus actores. El diseño de clases y de la base de datos constituye también un aspecto medular para obtener una aplicación que satisfaga las necesidades del cliente.

2.2 Objeto de automatización

El objeto de automatización de la solución que se propone, está enmarcado esencialmente en el proceso de contratación, específicamente en el subproceso para negociar el contrato y en las actividades relacionadas con la asignación de recursos y generación de pre-facturas, además de los procesos para gestionar los locales y recursos feriales.

2.3 Propuesta de sistema

La propuesta de solución de este trabajo consiste en el análisis, diseño, implementación y prueba de un módulo de asignación de recursos que contribuya a mitigar la pérdida de oportunidades de negocios y mejore la planificación del presupuesto de las ferias y eventos comerciales cubanos. Dicha solución estará provista de las funcionalidades necesarias para que exista un correcto funcionamiento de las tareas relacionadas con la asignación de recursos y la generación de las pre-facturas.

2.4 Arquitectura del sistema

A partir del estudio realizado en el capítulo 1, se definió para el sistema una arquitectura en capas que tiene como base los principios y la organización del patrón arquitectónico modelo vista controlador (MVC) como se muestra en la Figura 2.

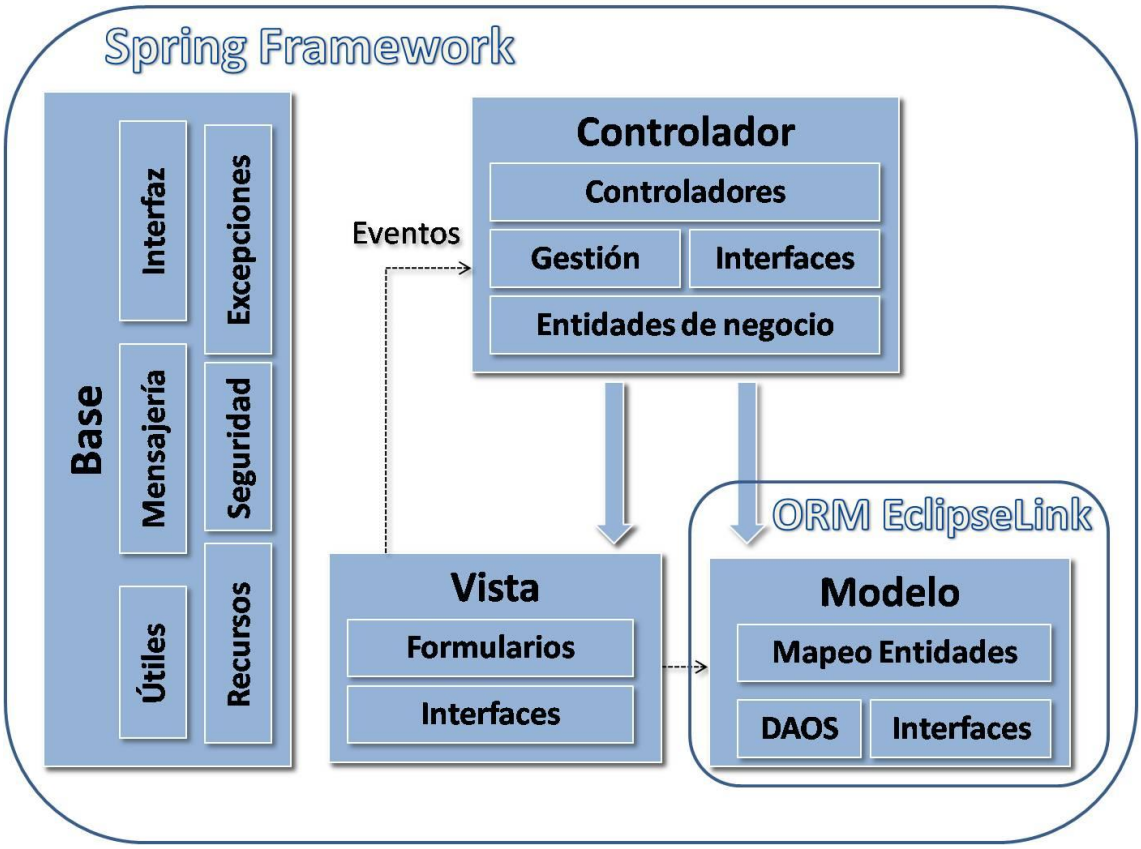

Figura 2: Arquitectura del sistema

La arquitectura del sistema está compuesta por cuatro capas nombradas: Base, Vista, Controlador y Modelo.

- En la capa base se agrupan todos los componentes comunes para las restantes capas como son: Útiles, Recursos, Excepciones, Interfaz, Mensajería y Seguridad.
- La vista agrupa los formularios e interfaces que serán implementadas por los mismos. Desde aquí se originan los eventos que serán capturados por el controlador para dar respuesta a los mismos.
- El controlador será el encargado de capturar los eventos lanzados desde la vista y proporcionar una respuesta a los mismos; contiene los controladores, las clases de gestión y sus interfaces, además de las entidades de negocio.
- El modelo está compuesto por el mapeo de entidades y los Data Access Objects (en español Objetos de Acceso a Datos) con las interfaces que implementan.

- Se hace uso del framework Spring, que desde el momento en que se inicia la aplicación carga todas sus configuraciones de los ficheros XML correspondientes, siendo el encargado de garantizar la inyección de dependencias.

Este tema será abordado con mayor profundidad en el capítulo 3.

2.5 Modelo de negocio

2.5.1 Breve descripción del negocio

La Cámara de Comercio de Cuba es una entidad que entre sus funciones tiene la de organizar algunas de las ferias y eventos que se celebran en el país, para lo cual dispone de un sistema que presenta varios problemas que no permiten que el mismo cumpla completamente con el propósito para el que fue creado. Siempre que se organiza una feria o evento es necesario establecer un contrato entre ambas partes lo que se realiza mediante el proceso de contratación que tiene como objetivo registrar los acuerdos contraídos entre la entidad participante y la Cámara de Comercio, así como las obligaciones de pago existentes.

Con este fin se efectúan varias tareas para negociar un contrato, con la finalidad de que los contrayentes estén de acuerdo con la firma del mismo. Una de las actividades antes mencionadas, es la de registrar al interesado que tiene asociada la solicitud de la cantidad y tipo de los locales y medios que el mismo requiere para su suscripción. Finalizada dicha actividad se da paso al subproceso para negociar el contrato, donde se llega a un acuerdo entre la Cámara de Comercio y el cliente, existiendo la posibilidad de que se le asigne lo solicitado o que haya que renegociar la solicitud. Una vez acordados todos los puntos del contrato se genera la pre-factura con los costos que traería a la parte interesada su participación en la feria o evento que organiza la Cámara de Comercio; concluyendo así el proceso y obteniéndose como resultado el registro de un nuevo interesado y la pre-factura asociada.

2.5.2 Diagrama de proceso de negocio

A partir de la descripción antes realizada a continuación se muestra en la Figura 3 el diagrama del proceso de contratación que es donde se encuentra enmarcada la mayor parte de la solución que propone este trabajo.

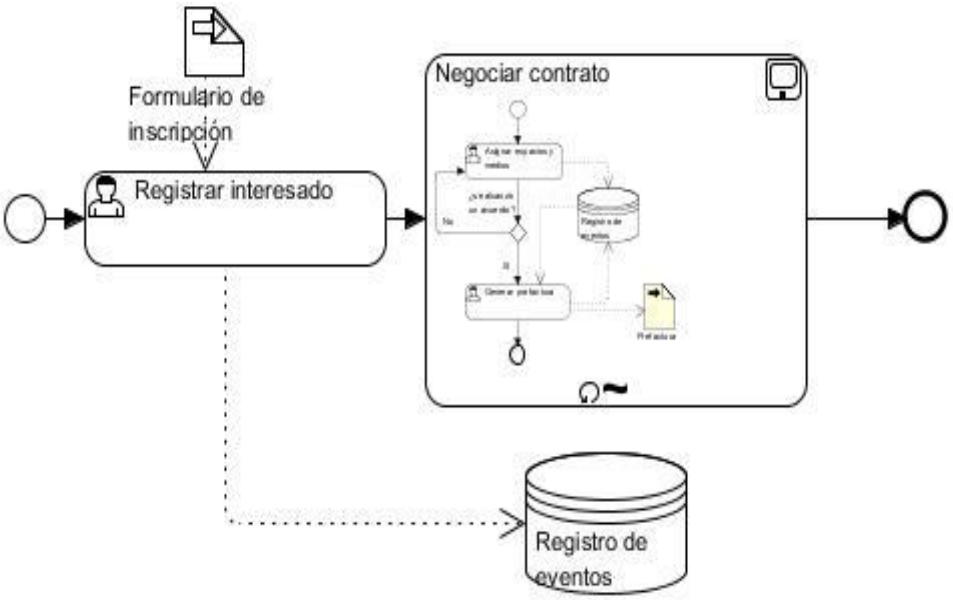

Figura 3: Diagrama de proceso de negocio: Proceso Contratación

Dentro del mismo se desarrolla el subproceso negociar contrato, del cual se muestra su diagrama en la Figura 4.

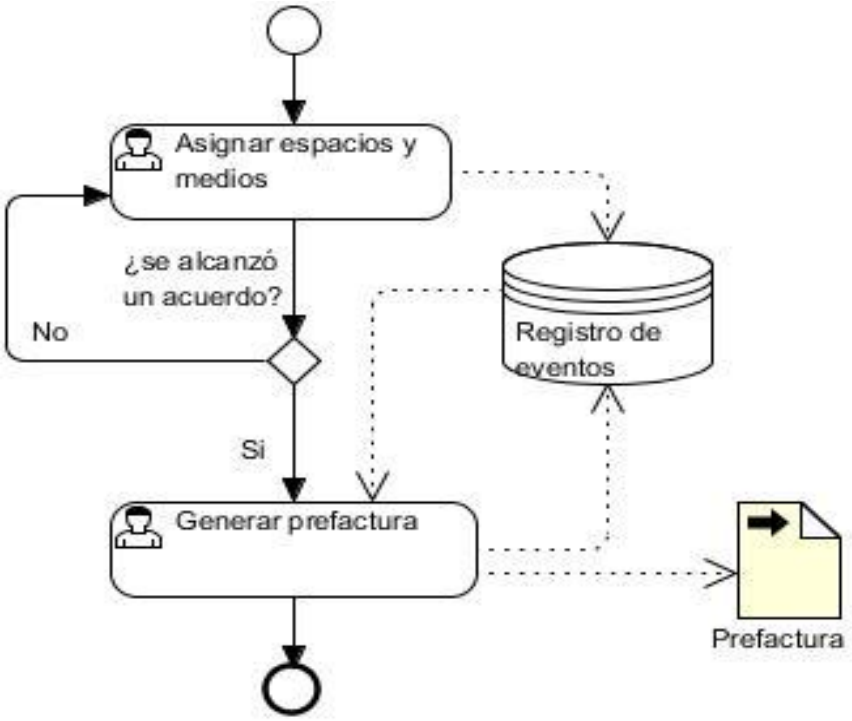

Figura 4: Diagrama de proceso de negocio: Subproceso Negociar contrato

2.6 Especificación de los requisitos de software

La captura y especificación de requisitos según el libro: “El proceso unificado de desarrollo de software” de Ivar Jacobson, Grady Booch y James Rumbaugh es: “el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe

construir" con el propósito de "guiar el desarrollo hacia el sistema correcto" (Jacobson, et al., 2000).

Es conveniente que las ideas, necesidades o deseos de los clientes y miembros del equipo de desarrollo de lo que debe hacer el sistema, sean analizadas como requisitos candidatos. Los requisitos o requerimientos se pueden clasificar en funcionales o no funcionales, siendo los funcionales las condiciones o capacidades que el sistema debe cumplir y los no funcionales las cualidades o propiedades que el producto debe tener; tratando que el mismo sea atractivo, usable, rápido y confiable entre otros.

2.6.1 Requerimientos funcionales

Los requerimientos funcionales deben verse como los procesos a informatizar, teniendo en cuenta que la satisfacción del cliente estará dada en buena medida por el grado de cumplimiento de los mismos. En el proceso de contratación fueron definidos una serie de requisitos de los cuales se mencionan a continuación los que guardan relación con las funcionalidades referidas en la propuesta de sistema:

RF_1. Registrar local
RF_2. Modificar local
RF_3. Eliminar local
RF_4. Listar locales
RF_5. Buscar local
RF_6. Registrar recurso ferial
RF_7. Modificar recurso ferial
RF_8. Eliminar recurso ferial
RF_9. Listar recursos feriales
RF_10. Buscar recurso ferial
RF_11. Registrar solicitud de recursos.
RF_12. Modificar solicitud de recursos.
RF_13. Eliminar solicitud de recursos.
RF_14. Listar solicitud de recursos.
RF_15. Consultar solicitud de recursos.
RF_16. Buscar solicitud de recursos.
RF_17. Registrar stand
RF_18. Modificar stand
RF_19. Eliminar stand
RF_20. Registrar asignación de recursos.
RF_21. Modificar asignación de recursos.
RF_22. Eliminar asignación de recursos.
RF_23. Listar asignaciones de recursos.
RF_24. Consultar asignación de recursos.
RF_25. Buscar asignación de recursos.
RF_26. Registrar pre-factura por concepto de espacios y medios.
RF_27. Modificar pre-factura por concepto de espacios y medios.
RF_28. Anular pre-factura por concepto de espacios y medios.
RF_29. Listar pre-facturas por concepto de espacios y medios.
RF_30. Consultar pre-factura por concepto de espacios y medios.

RF_31. Buscar pre-factura por concepto de espacios y medios.
RF_32. Imprimir pre-factura.
RF_33. Exportar pre-factura.
RF_34. Enviar pre-factura por correo electrónico.

Las descripciones de los requisitos mencionados pueden ser consultadas en el Anexo 1, a continuación en la Tabla 1 se muestra la descripción del requisito funcional Registrar pre-factura.

Descripción textual del requisito: Registrar pre-factura

Precondiciones	Exista una asignación de espacios y/o medios.	
Flujo de eventos		
Flujo básico registrar pre-factura		
1.	El sistema asigna un número a la pre-factura y muestra los datos restantes de la pre-factura que se genera de la solicitud de recursos: De la entidad: Código, Nombre, De la pre-factura: Número, Concepto, Si se realiza o no descuento, % de descuento, Esquema de pago, Importe total, Descuento, A pagar, Monto a pagar en correspondencia con el esquema de pago, De los recursos: Código, Descripción, U.M., P.U., Cantidad, Importe, Descuento, % de descuento, Moneda	
2.	El usuario registra la pre-factura.	
3.	El sistema valida los datos.	
4.	El sistema registra los datos y notifica al usuario.	
5.	Concluye el requisito.	
Pos-condiciones		
1.	Se registra una nueva pre-factura.	
Flujos alternativos		
Flujo alternativo 3.a Campos vacíos o erróneos.		
1	El sistema notifica que existen campos vacíos o erróneos e indica cuáles son estos.	
2	Continúa en el paso 2 del flujo básico.	
Pos-condiciones		
1	N/A.	
Flujo alternativo * El usuario cancela el requisito.		
1	Concluye el requisito.	
Pos-condiciones		
1	N/A.	
Validaciones		
1	Se validan los datos según lo establecido en el Modelo conceptual <<Referencia al modelo conceptual en cuestión>>.	
Conceptos	**Pre-factura**	Visibles en la interfaz: Número Concepto Si se realiza o no descuento % de descuento Esquema de pago Importe total Descuento A pagar Monto a pagar en correspondencia con el esquema de pago Utilizados internamente: N/A

	Recurso ferial	Visibles en la interfaz: Código Descripción U.M. P.U. Cantidad Importe Descuento % de descuento Moneda Utilizados internamente: N/A
	Entidad	Visibles en la interfaz: Código Nombre Utilizados internamente: N/A
Requisitos especiales		
Asuntos pendientes		

Tabla 1: Descripción textual RF Registrar pre-factura

2.6.2 Requerimientos no funcionales

Deben pensarse como las características que hacen al producto atractivo, usable, rápido o confiable; son fundamentales en el éxito del producto. Normalmente están vinculados a requisitos funcionales y son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto. Existen múltiples categorías para clasificar a los requisitos no funcionales, siendo las siguientes representativas de un conjunto de aspectos que se deben tener en cuenta para el desarrollo de la aplicación:

Usabilidad

RNF_1. Facilidad de aprendizaje.

- Necesidad de un fácil aprendizaje en cuanto a instalación, configuración y uso del sistema.
- Proveer manuales de instalación, configuración y uso del sistema.

RNF_2. Facilidad de uso.

- Las interfaces de usuario indican los campos requeridos y muestran ejemplos de los valores que deben introducirse en cada campo.
- Los menús permitirán una navegación sencilla, tanto a los usuarios con conocimientos avanzados de informática como a los usuarios más inexpertos

- La ayuda es sensible al contexto e incluye información sobre el proceso de negocio.

RNF_3. Mínimo impacto de los errores.

- El sistema permite cancelar los flujos y sus tareas.
- El sistema notifica a los usuarios los errores y sugiere cómo corregirlos.

Confiabilidad

RNF_4. Notificación de omisión ó errores en los datos introducidos.

- El sistema notifica al usuario los errores u omisiones en los datos introducidos.

Eficiencia

RNF_5. El sistema debe demorar como promedio en una transición un segundo y aproximadamente 5 segundos como tiempo máximo. Siendo este tiempo el correspondiente a los procesos que realizan consultas a la bases de datos y mecanismos de comunicación entre componentes

Software

RNF_6. Servidores de base de datos locales Postgre SQL versión 8.3.
RNF_7. Máquina Virtual de Java en su versión JDK 1.6.

Hardware

RNF_8. Hardware mínimo para oficinas:

- 512 MB de memoria RAM.
- 20 GB de disco duro.
- Procesador Pentium IV.

RNF_9. Hardware mínimo para servidores:

- 2 GB de memoria RAM.
- 120 GB de disco duro.
- Procesador Dual Core o superior.

Soporte

RNF_10. Portabilidad

- Se debe diseñar para ser desarrollado sin basarse en características propias de algún sistema operativo, para lograr un producto multiplataforma.

Restricciones de diseño

RNF_11. Restricciones de diseño

- **Herramienta de Modelado Visual Paradigm:** Se utilizará la herramienta CASE Visual Paradigm, teniendo en cuenta sus ventajas para modelar los diferentes artefactos que se obtienen en los flujos de trabajo y sus diferentes fases. Las restricciones propias del diseño radican en las pautas que se establecerán, así como las diferentes relaciones que se formen durante el modelado.

- **Lenguaje de Programación:** El software estará programado en Java, siguiendo una codificación estándar y organizada, haciendo uso de las potencialidades propias del lenguaje para implementar los diferentes procesos.

- **Entorno de desarrollo integrado (IDE):** El software se desarrollará sobre Netbeans 7.1 ya que contiene las herramientas para llevar a efecto la implementación de la totalidad de los componentes impuestos por la arquitectura y el diseño.

- **Restricciones Arquitectónicas:** El diseño tomará como punto de partida la Línea Base de la Arquitectura y las restricciones que esta impone para el desarrollo, tal es el caso de los componentes y sus conectores. Igualmente incorporará las bibliotecas de clases y frameworks establecidos en dicho documento para el trabajo adecuado de la arquitectura según las cualidades que se desean obtener en ella.

Seguridad

RNF_12. La autenticación estará basada en el uso de credenciales.

RNF_13. Las contraseñas se almacenarán en la base de datos haciendo uso del algoritmo hash MD5[7].

[7] MD5: abreviatura de *Message-Digest Algorithm 5*, Algoritmo de Resumen del Mensaje 5, es un algoritmo de reducción criptográfico.

RNF_14. Los usuarios estarán autorizados a realizar las acciones que se encuentran definidas para el rol al cual pertenece.

RNF_15. Solo se podrá acceder a la aplicación desde las direcciones IP[8] autorizadas.

RNF_16. Los documentos que exporta el sistema deben estar dotados de las configuraciones necesarias para que sean de solo lectura.

Interfaz

RNF_17. Proveer interfaces que posibiliten la integración con el sistema de gestión contable EXACT, para la generación de las facturas a partir de las pre-facturas generadas; esta acción debe realizarse mediante la exportación de las pre-facturas

Estándares aplicables

RNF_18. Se aplicarán los estándares legales, de calidad, internacionalización y regulatorios definidos por la empresa comercial que provee el sistema.

2.7 Definición de casos de uso

En este epígrafe se definen los casos de uso que responden a los requerimientos del cliente y se tendrán en cuenta para el posterior análisis, diseño y la implementación del sistema.

2.7.1 Definición de los actores

Actores	Justificación
Funcionario organizador	Es el encargado de la organización, tiene el control total de todos los módulos existentes en el evento o feria.
Funcionario diseñador	Es el responsable de la asignación de los espacios y medios para el desarrollo del evento o feria.
Funcionario económico	Es la persona encargada de la parte económica, su desempeño fundamental está en la elaboración de las pre-facturas.

Tabla 2: Actores del sistema

[8] Dirección IP: número que identifica a cada dispositivo dentro de una red con protocolo IP.
Protocolo IP: usado para la comunicación de datos a través de una red.

2.7.2 Listado de casos de uso

CU_1	Gestionar local
Actor	Funcionario organizador
Descripción	Permite registrar, modificar y eliminar los locales en los que la Cámara de Comercio organiza las ferias y eventos.
Referencia	RF_1, RF_2, RF_3, RF_6, RF_7, RF_8

CU_2	Gestionar recurso ferial
Actor	Funcionario organizador
Descripción	Permite registrar, modificar y eliminar los recursos feriales de los que dispone la Cámara de Comercio para celebrar las ferias y eventos.
Referencia	RF_9, RF_10, RF_11

CU_3	Gestionar solicitud de recursos
Actor	Funcionario organizador
Descripción	Permite registrar, modificar y eliminar la solicitud de los locales y recursos feriales que tiene la Cámara de Comercio para la realización de los eventos.
Referencia	RF_14, RF_15, RF_16, RF_20, RF_21, RF_22

CU_4	Gestionar asignación de recursos
Actor	Funcionario diseñador
Descripción	Permite registrar, modificar y eliminar la asignación de los locales y recursos feriales que tiene la Cámara de Comercio para la realización de los eventos.
Referencia	RF_23, RF_24, RF_25

CU_5	Gestionar pre-factura por concepto de espacios y medios
Actor	Funcionario económico
Descripción	Permite registrar, modificar y anular la pre-factura, en la misma consta el monto que hasta el momento en el que se genera debe pagar el cliente por su participación en el evento o feria.
Referencia	RF_29, RF_30, RF_31

CU_6	Listar
Actor	Funcionario organizador, Funcionario diseñador, Funcionario económico
Descripción	Puede generar un listado de las solicitudes, recursos, asignaciones de recursos o pre-facturas según sea el caso.
Referencia	RF_4, RF_12, RF_17, RF_26, RF_32

CU_7	Consultar
Actor	Funcionario organizador, Funcionario diseñador, Funcionario económico
Descripción	Muestra los detalles de las solicitudes, asignaciones de recursos o pre-facturas según sea el caso.
Referencia	RF_18, RF_27, RF_33

CU_8	Buscar local
Actor	Caso de uso extendido
Descripción	Busca y muestra el listado de los locales que cumplen con el o los criterios de búsqueda según sea el caso.
Referencia	RF_5, CU_1

CU_9	Buscar recurso ferial
Actor	Caso de uso extendido
Descripción	Busca y muestra el listado de los recursos feriales que cumplen con el o los criterios de búsqueda según sea el caso.
Referencia	RF_13, CU_2

CU_10	Buscar solicitud de recursos
Actor	Caso de uso extendido
Descripción	Busca y muestra el listado de las solicitudes de recursos que cumplen con el o los criterios de búsqueda según sea el caso.
Referencia	RF_19, CU_3, CU_4

CU_11	Buscar asignación de recursos
Actor	Caso de uso incluido
Descripción	Busca y muestra el listado de las asignaciones de recursos que cumplen con el o los criterios de búsqueda según sea el caso.
Referencia	RF_28, CU_5

CU_12	Buscar pre-factura por concepto de espacios y medios
Actor	Caso de uso extendido
Descripción	Busca y muestra el listado de las pre-facturas por concepto de espacios y medios que cumplen con el o los criterios de búsqueda según sea el caso.
Referencia	RF_34, CU_5

CU_13	Exportar pre-factura
Actor	Caso de uso extendido
Descripción	Exporta una o varias pre-facturas.
Referencia	RF_35, CU_5

CU_14	Imprimir pre-factura
Actor	Caso de uso extendido
Descripción	Imprime una pre-factura.
Referencia	RF_36, CU_5

CU_15	Enviar pre-factura por correo electrónico
Actor	Caso de uso extendido
Descripción	Envía por correo electrónico una pre-factura.
Referencia	RF_37, CU_5

Tabla 3: Resúmenes de casos de uso

2.7.3 Diagrama de casos de uso

El diagrama de casos de uso del sistema es uno de los artefactos más importantes que se genera en la fase de inicio de la metodología seleccionada, muestra los actores y sus relaciones con los casos de uso del sistema, el mismo puede ser observado en la Figura 5.

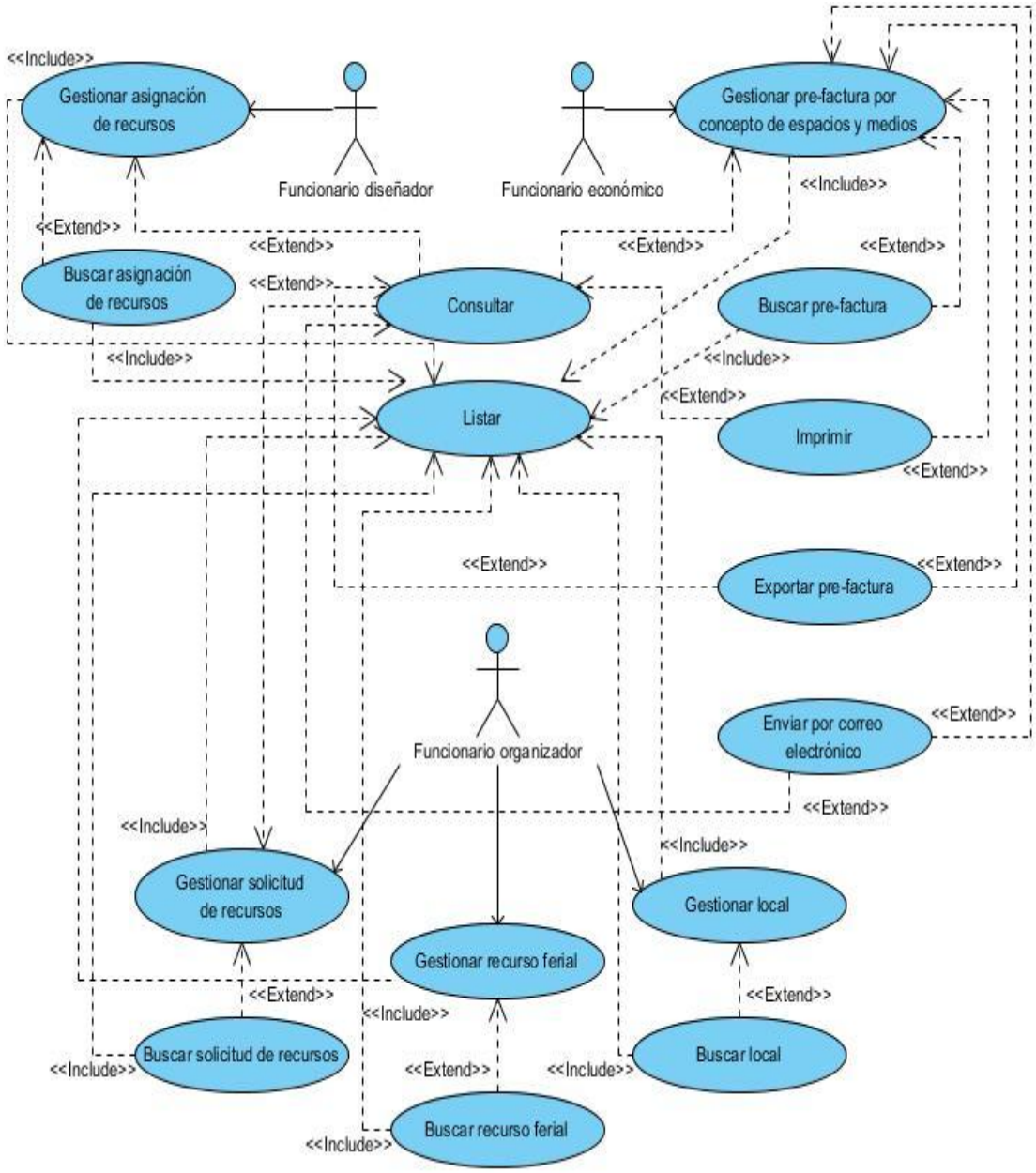

Figura 5: Diagrama de casos de uso: Módulo asignación de recursos

2.7.4 Descripción textual

Mediante la descripción textual se puede comprender mejor como es que se realiza el caso de uso, la forma en la que interactúa el actor con el sistema y cuáles son las repuestas del mismo. Por la importancia que presenta el caso de uso Gestionar pre-factura a continuación en la Tabla 4 se muestra la descripción textual del mismo, las restantes descripciones con sus prototipos de interfaz pueden ser consultadas en el Anexo 2.

Descripción textual del caso de uso: Gestionar pre-factura

Objetivo	Registrar, modificar o anular una pre-factura.	
Actores	Funcionario económico.	
Resumen	El caso de uso se inicia cuando se registra, modifica o anula una pre-factura, termina con el cumplimiento de la funcionalidad deseada por el actor.	
Complejidad	Alta	
Prioridad	Crítico	
Precondiciones	El usuario debe estar registrado y con los permisos necesarios asignados.	
Postcondiciones	Se registra, modifica o anula una pre-factura.	
Referencias	RF_26, RF_27, RF_28	
Flujo de eventos		
Flujo básico "Gestionar pre-factura"		
	Actor	**Sistema**
1.	Selecciona la opción: Gestionar pre-factura.	
2.		Lista todas las pre-facturas. Ver caso de uso "Listar".
3.		Se puede consultar una pre-factura.
4.		Permite realizar varias acciones con una pre-factura: - Registrar una nueva pre-factura. Ver Sección 1: "Registrar pre-factura" - Modificar una pre-factura. Ver Sección 2: "Modificar pre-factura" - Anular una pre-factura. Ver Sección 3: "Anular pre-factura"
Flujos alternos		
1a. El usuario cancela el caso de uso		
	Actor	**Sistema**
1.		Termina el caso de uso
Sección 1: "Registrar pre-factura"		
Flujo básico Registrar pre-factura		
1.		Muestra la interfaz correspondiente al registro de la pre-factura. Se visualizan todos los campos necesarios para el mismo: De la entidad: Código, Nombre De la pre-factura: Número, Concepto, Si se realiza o no descuento, % de descuento, Esquema de pago, Importe total, Descuento, A pagar, Monto a pagar en correspondencia con el esquema de pago. De los recursos: Código, Descripción, U.M., P.U., Cantidad, Importe, Descuento, % de descuento, Moneda.
2.		Carga los códigos y los nombres de las entidades.
3.	Selecciona el código o la entidad que desea	

	pre-facturar.	
4.		Genera automáticamente el número de la pre-factura.
5.		Muestra habilitada la opción realizar descuento solo si la empresa es asociada; el campo descuento permanece deshabilitado hasta que se marque la opción realizar descuento si es que procede.
6.		Buscar los recursos contratados. Ver CU Buscar asignación de recursos.
7.	Inserta los datos correspondientes y selecciona la opción Aceptar	
8.		Valida los datos.
9.		Registra la pre-factura y muestra un mensaje notificando el registro de la misma.
10.		Habilita las opciones Anular, Imprimir, Exportar, Enviar por correo, permitiendo realizar varias acciones con una pre-factura: - Anular una pre-factura. Ver Sección 3: "Anular pre-factura" - Imprimir pre-factura - Exportar pre-factura - Enviar pre-factura
11.		Termina el caso de uso
Flujos alternos		
8a. Campos vacios o erróneos		
	Actor	**Sistema**
1.		Notifica que existen campos vacíos o erróneos e indica cuáles son estos.
2.		Continúa en el paso 7 del flujo básico.
Sección 2: "Modificar pre-factura"		
Flujo básico Modificar pre-factura		
1.	Se puede aplicar un filtro de búsqueda a las pre-facturas.	
2.	Selecciona una pre-factura.	
3.		Muestra la interfaz correspondiente a la modificación de la pre-factura. Se visualizan todos los campos necesarios para la misma: De la entidad: Código, Nombre De la pre-factura: Número, Concepto, Si se realiza o no descuento, % de descuento, Esquema de pago, Importe total, Descuento, A pagar, Monto a pagar en correspondencia con el esquema de pago. De los recursos: Código, Descripción, U.M., P.U., Cantidad, Importe, Descuento, % de descuento, Moneda.
4.	Modifica los valores correspondientes y selecciona la opción Aceptar.	

5.		Valida los datos.
6.		Modifica la pre-factura.
7.		Muestra un mensaje notificando la modificación de la pre-factura.
8.		Habilita las opciones Imprimir, Exportar, Enviar por correo, permitiendo realizar varias acciones con una pre-factura: - Imprimir pre-factura - Exportar pre-factura - Enviar pre-factura
9.		Termina el caso de uso
Flujos alternos		
5a. Campos vacios o erróneos		
	Actor	**Sistema**
1.		Notifica que existen campos vacíos o erróneos e indica cuáles son estos.
2.		Continúa en el paso 4 del flujo básico.
Sección 3: "Anular pre-factura"		
Flujo básico Anular pre-factura		
1.	Se puede aplicar un filtro de búsqueda a las pre-facturas.	
2.	Selecciona una pre-factura y la opción anular.	
3.		Muestra un mensaje para confirmar la acción.
4.	Confirma la acción.	
5.		Anula la pre-factura y muestra un mensaje de notificación.
6.		Termina el caso de uso.
Flujos alternos		
4a. Negar acción		
	Actor	**Sistema**
1.		Continúa en el paso 2 del flujo básico.

Relaciones	**CU Incluidos**	CU 6. Listar CU 11. Buscar asignación de recursos
	CU Extendidos	Consultar pre-factura. Paso 3 del flujo básico. Buscar pre-factura. Paso 1 del flujo básico. Secciones 2 y 3 Imprimir pre-factura. Paso 10 del flujo básico Sección 1. Paso 8 del flujo básico Sección 2. Exportar pre-factura. Paso 10 del flujo básico Sección 1. Paso 8 del flujo básico Sección 2. Enviar pre-factura. Paso 10 del flujo básico Sección 1. Paso 8 del flujo básico Sección 2.
Requisitos no funcionales		
Asuntos pendientes		

Tabla 4: Descripción textual CU gestionar pre-factura

2.8 Diagrama de clases de análisis

El diagrama de clases de análisis constituye una vista estática de las clases que conforman el modelo del análisis y las asociaciones entre las mismas, es una vista de la futura composición de clases del software. En la Figura 6 se muestra el diagrama correspondiente al caso de uso número cinco descrito en el capítulo dos. Los restantes diagramas podrán ser consultados en el Anexo 3.

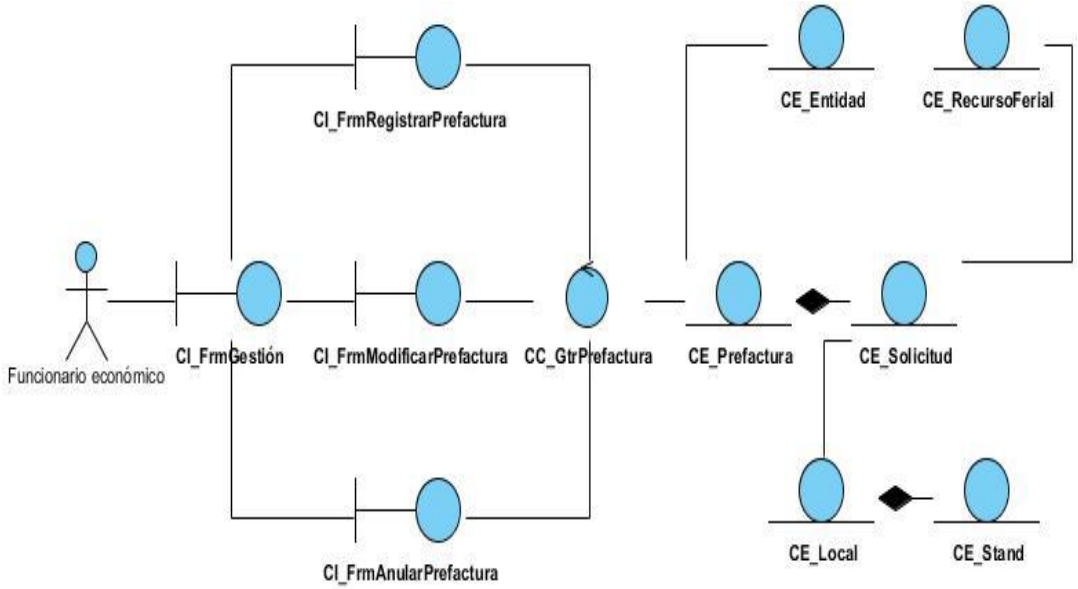

Figura 6: Diagrama de clases de análisis CU gestionar pre-factura

2.9 Diagrama de secuencia

El diagrama de secuencia se utiliza para ilustrar la realización de un caso de uso, incluye una secuencia cronológica de los mensajes intercambiados entre los objetos. El diagrama correspondiente a la sección registrar pre-factura del caso de uso Gestionar pre-factura descrito con anterioridad, para su mejor observación el diagrama será dividido en dos partes como se muestra en las Figuras 7 y 8. Los restantes diagramas podrán ser consultados en el Anexo 3.

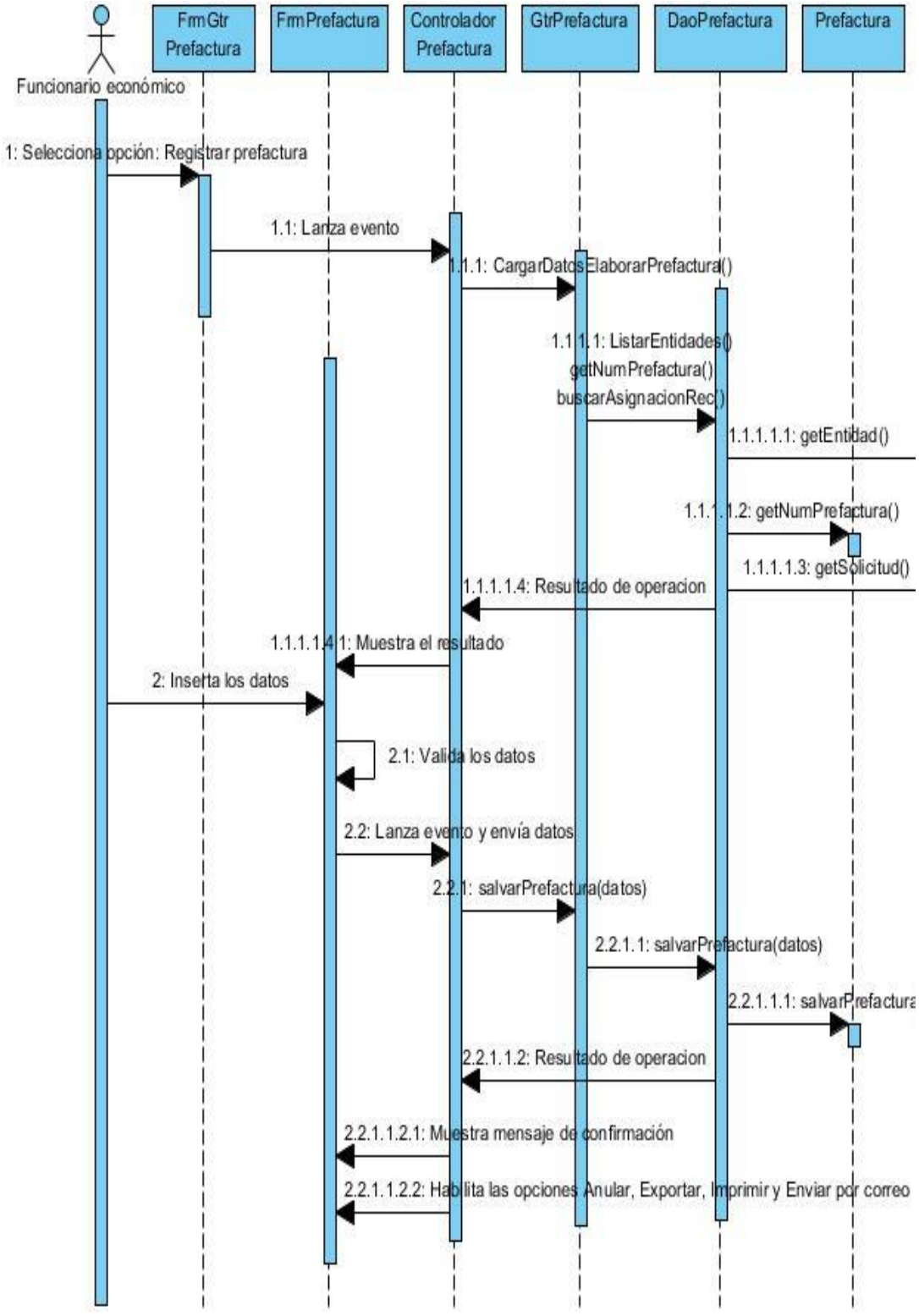

Figura 7: Diagrama de secuencia. Sección registrar pre-factura. Parte 1

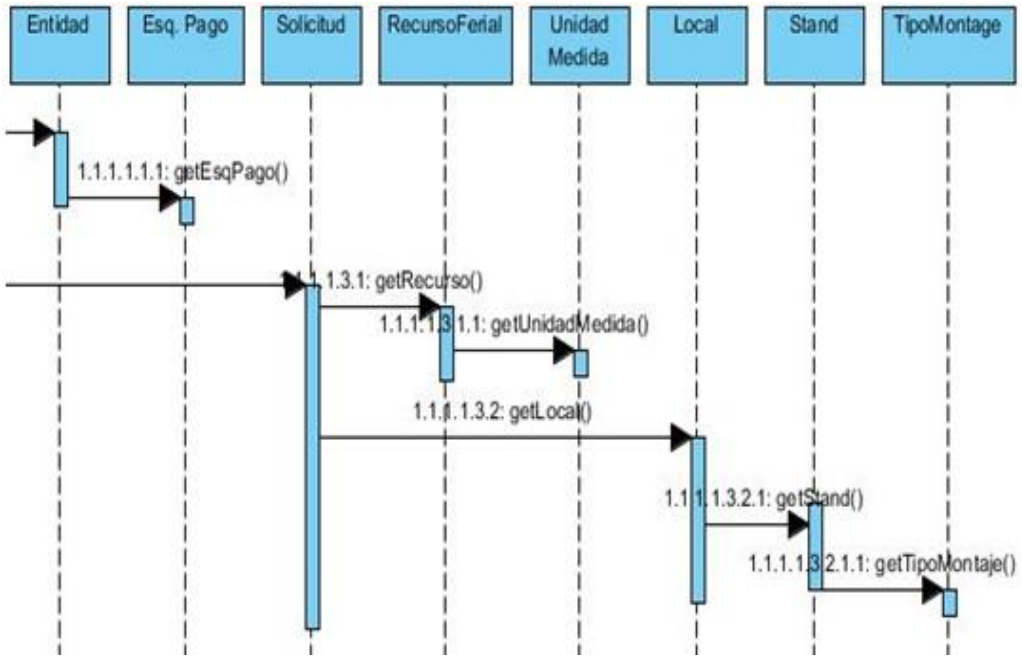

Figura 8: Diagrama de secuencia. Sección registrar pre-factura. Parte 2

2.10 Diagrama de clases de diseño

El diagrama de clases del diseño describe la realización de un caso de uso y al mismo tiempo es una abstracción del modelo de implementación y el código fuente; tiene gran importancia para comenzar con la implementación convirtiéndose en una entrada esencial para esta actividad. En la Figura 9 se observa de forma conceptual el diagrama de clases de diseño del caso de uso gestionar pre-factura descrito en este capítulo. Los restantes diagramas así como las descripciones de las clases que conforman el diagrama pueden ser consultados en el Anexo 4.

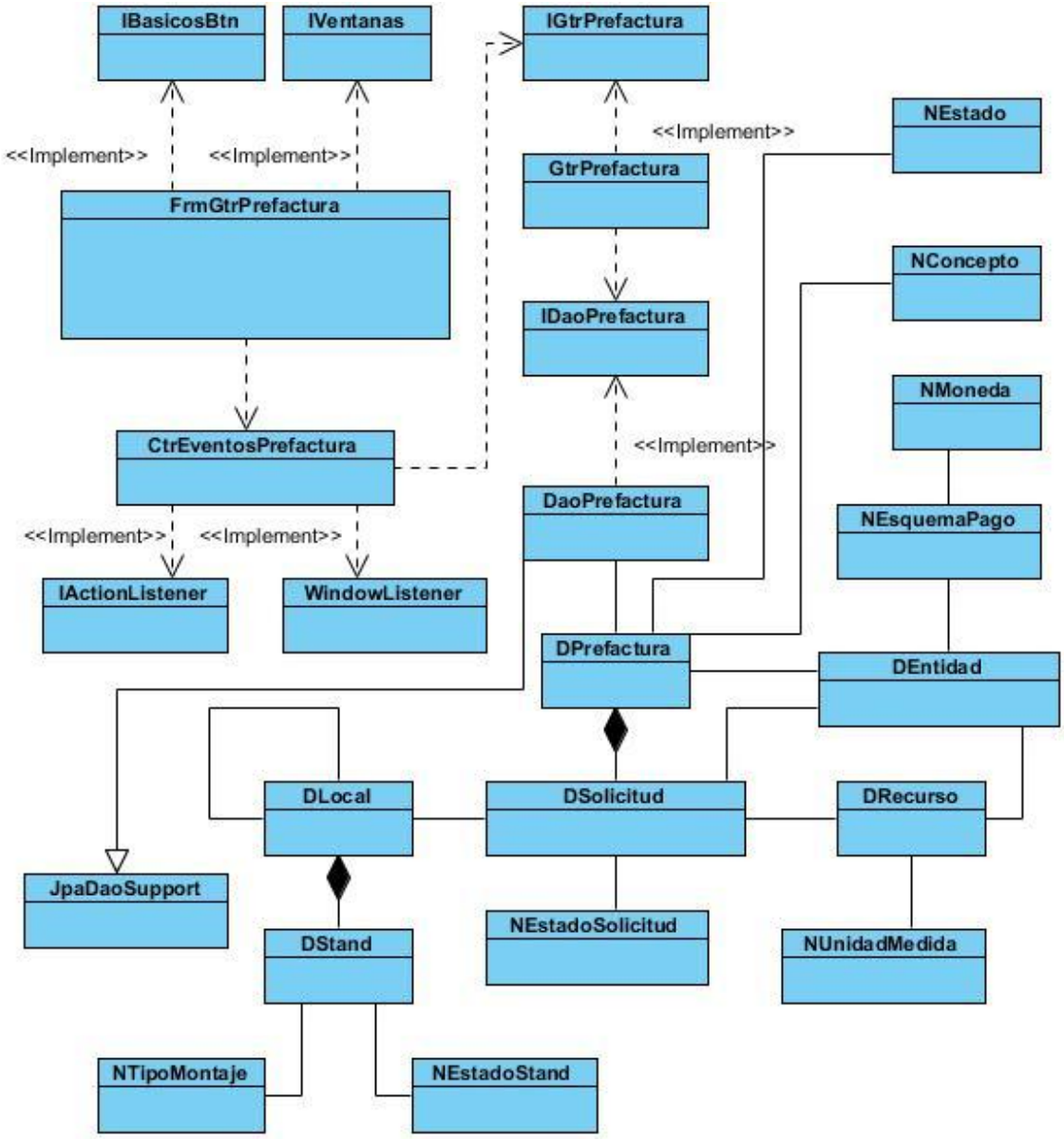

Figura 9: Diagrama de clases de diseño CU Gestionar pre-factura

2.10.1 Descripción de las clases

En este sub-epígrafe se describen las clases más importantes que conforman el diagrama que se muestra en la Figura 9, el mismo abarca la totalidad de las clases persistentes del módulo.

Clases persistentes

Todas las clases que comienzan por **N** se refieren a los nomencladores, que tienen por responsabilidad establecer los valores que puede tener un atributo en la clase con la cual presentan una relación, tienen como atributos un nombre y una descripción. Los nomencladores se relacionan a continuación: NEstado, NMoneda,

NEsquemaPago, NConcpeto, NEstadoSolicitud, NUnidadMedida, NTipoMonate, NEstadoStand y pueden ser observados en la Figura 10.

Las clases que se describen a continuación pueden ser observadas en las Figura 10.

- DPrefactura tiene como propósito modelar el objeto pre-factura, la cual se asocia a una entidad, y un listado de solicitudes, además de un estado y un concepto.
- DEntidad modela el objeto entidad, se asocia a una pre-factura, una solicitud, un esquema de pago y recursos. Esta entidad juega varios roles dentro del negocio en este caso particular puede comportarse como un entidad que participa en el evento y tiene solicitudes de recursos o como un suministrador de recursos.
- DSolicitud su propósito es modelar el objeto solicitud, tiene asociada una entidad, un recurso ferial y un local que son las posibles opciones de solicitudes, además de un estado. Esta clase se comporta como una solicitud en el momento en que se solicitan los recursos y pasaría a comportarse como una asignación en el momento en que se asignan los recursos.
- DLocal puede estar compuesta por varios locales y tiene un listado de los Stands que pueda tener.
- DStand tiene asociada un estado y un tipo de montaje.
- DRecursoFerial modela el objeto pre-factura, tiene asociada una entidad, una solicitud y una unidad de medida.

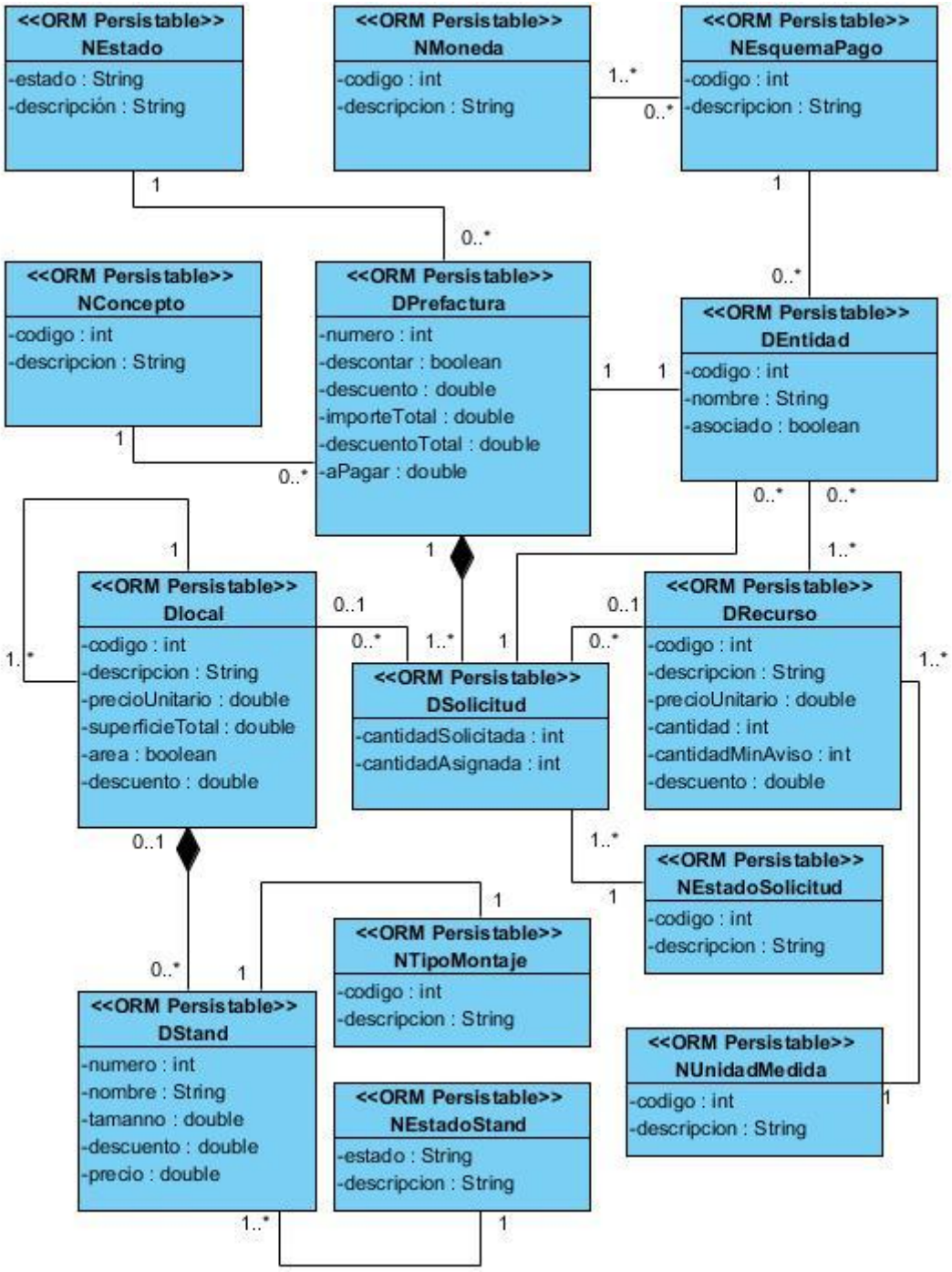

Figura 10: Clases persistentes

Clase controlador de eventos

Todas las clases que comienzan por **CtrEventos** se refieren a los controladores, cada caso de uso tiene asociado un controlador que es el encargado de dar una respuesta a cada evento que es originado desde la vista (Formularios). Los controladores asociados se mencionan a continuación: CtrEventosLocal, CtrEventosRecursoFerial,

CtrEventosSolicitud, CtrEventosAsignación, CtrEventosPrefactura. En la Figura 11 puede observarse el controlador correspondiente a la gestión de pre-facturas.

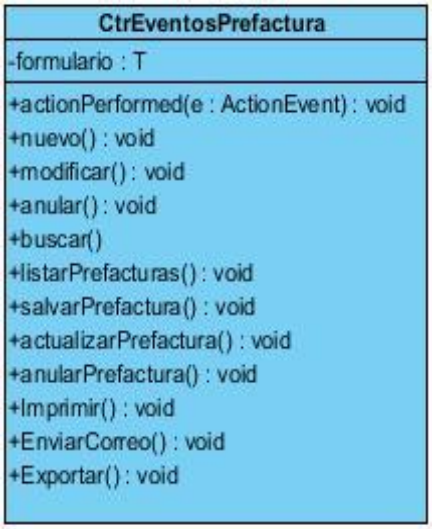

Figura 11: Controlador de eventos pre-factura

Clases de gestión

Las clases de gestión comienzan por **Gtr**, las mismas se llaman GtrLocal, GtrRecursoFerial, GtrSolicitud, GtrAsignacion, GtrPrefactura y en la Figura 12 se muestra una para mejor comprensión.

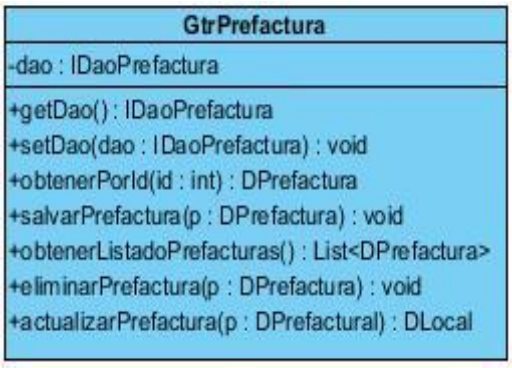

Figura 12: Gestionar pre-factura

2.11 Diseño de la base de datos

Una de las tareas más importantes a la hora de construir un nuevo producto de software es el diseño de la base de datos, este epígrafe estará dedicado al tema en cuestión.

2.11.1 Diagrama Entidad-Relación de la base de datos

El diagrama Entidad-Relación constituye un medio de representación conceptual de los problemas y la visión de un sistema de forma global, sus elementos fundamentales

son las entidades y las relaciones. En este epígrafe en la Figura 13 se muestra el diagrama entidad relación de la base de datos del módulo de asignación de recursos.

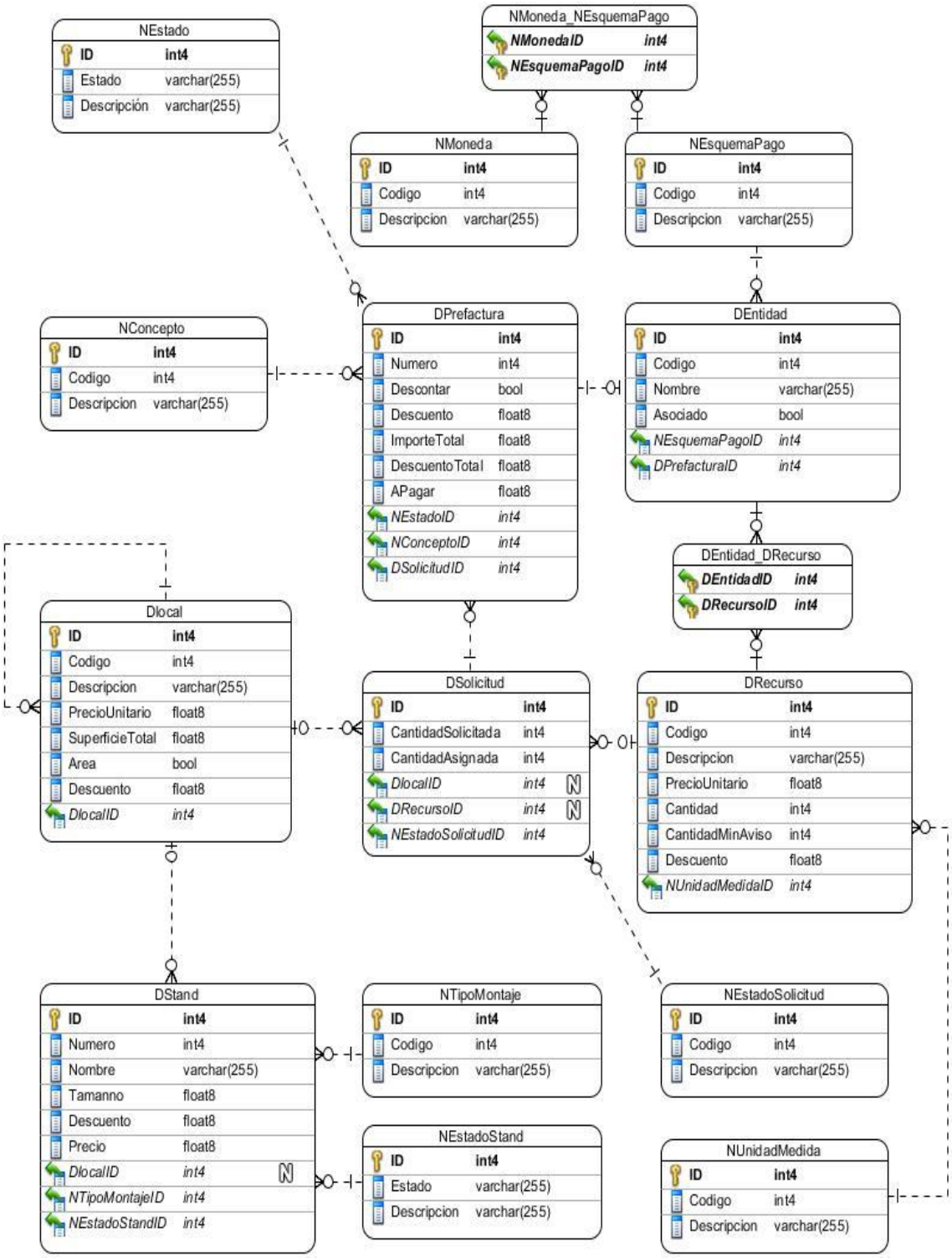

Figura 13: Diagrama entidad relación

2.12 Conclusiones del capítulo

En el presente capítulo se aborda una descripción de la solución propuesta para informatizar el subproceso, negociar contrato, y las actividades del proceso de contratación que están directamente relacionadas a la asignación de recursos y la gestión de pre-facturas. Así mismo se elaboraron los artefactos comprendidos en las fases de modelado y elaboración de la metodología escogida y argumentada en el capítulo 1.

Capítulo 3
Implementación y prueba

3.1 Introducción

En este capítulo serán abordados los temas relacionados con la implementación y las pruebas del sistema; ambos aspectos son muy importantes para satisfacer las expectativas del cliente y desarrollar un software de calidad.

3.2 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones, muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos del software que entran en la fabricación de aplicaciones informáticas, pueden ser archivos, paquetes, bibliotecas cargadas dinámicamente, etc. El diagrama que se visualiza en la Figura 14 muestra los componentes que conforman el sistema, en el mismo se pueden observar cuatro componentes genéricos que contienen otro número de paquetes que serán mencionados a continuación.

- La funcionalidad del componente base es similar a la de un framework está formado por los recursos, las excepciones, mensajería, seguridad e interfaces que son implementadas por los útiles.

  Recursos: se encuentran los XML con la configuración básica del framework Spring.

  Excepciones: componente que se encarga de interpretar las excepciones que serán manejadas en el sistema, pueden ser de tres tipos: de negocio BOException acceso a datos DAOException o de aplicación ApplicationException, con el objetivo de que se realice un correcto tratamiento de las excepciones.

Mensajería: interpreta los mensajes que serán mostrados en la aplicación, para así lograr una gestión más eficiente de los mismos.

Seguridad: aquí se encuentran las configuraciones para proteger la conexión a la base de datos y además las funcionalidades necesarias para hacer uso del algoritmo de encriptación MD5.

Útiles: puede verse como la fachada del componente base, a través del mismo se establece la comunicación de las restantes capas de la arquitectura con cada uno de los paquetes de la base.

- En la vista se pueden apreciar los formularios y las interfaces que son implementadas por los mismos, así como una librería para facilitar el trabajo con las tablas que son mostradas al usuario.

- El controlador está compuesto por los controladores, los gestores con las interfaces que implementan y las entidades de negocio.

  Controladores: son los encargados de la captura de cada uno de los eventos que se originan en la vista para posteriormente darle respuesta a los mismos, en este paquete se encuentra una clase java con la responsabilidad de iniciar la aplicación y cargar todas las configuraciones que necesita Spring, además de comportarse como un controlador frontal, encargado de recibir cada uno de los eventos e instanciar al controlador correspondiente para satisfacer los mismos.

  Gestores: se encuentran las clases de gestión que son las encargadas de resolver los problemas del negocio, además de inicializar las transacciones de Spring para el acceso a datos.

  Entidades de negocio: agrupa las entidades que son necesarias para dar respuesta a una determinada funcionalidad pero que las mismas no persisten en la base de datos.

- En el modelo se encuentran las interfaces que son implementadas por los DAOs y el mapeo de entidades.

DAOs: componente que recoge las clases que garantizan el acceso a datos utilizando el EntityManager de Spring para interactuar con la base de datos.

Mapeo de entidades: componente donde pueden ser encontradas las entidades persistentes, las cuales cuentan con las notaciones que propone la tecnología JPA, para en conjunto con el framework EclipseLink garantizar el mapeo relacional de objetos.

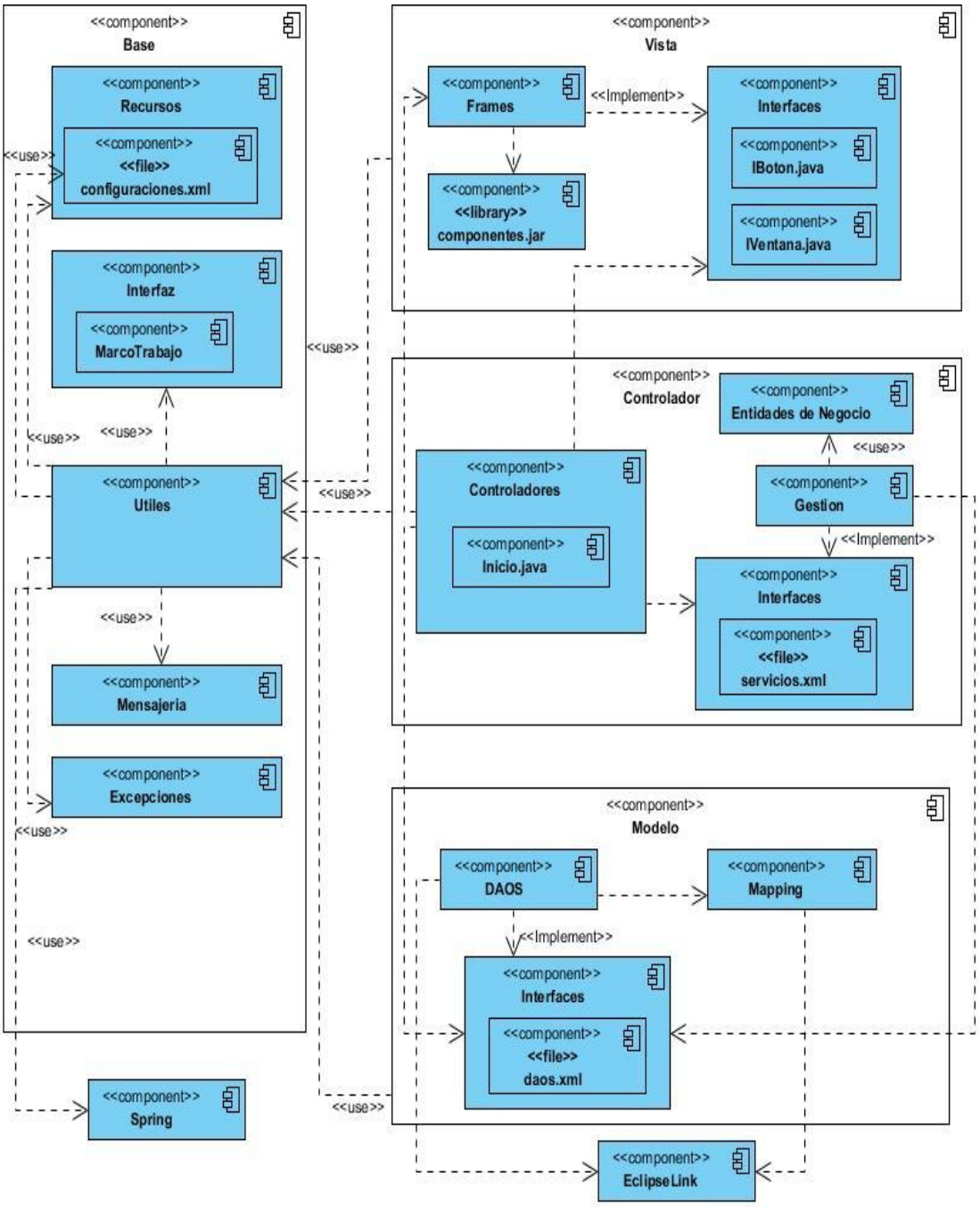

Figura 14: Diagrama de componentes

3.3 Diagrama de despliegue

Los Diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, que generalmente tiene algo de memoria y, a menudo, capacidad de procesamiento. Los nodos se utilizan para modelar la topología del hardware sobre el que se ejecuta el sistema. Representan típicamente un procesador o un dispositivo sobre el que se pueden desplegar los componentes. A continuación en la Figura 15 puede observarse el diagrama de despliegue de la aplicación.

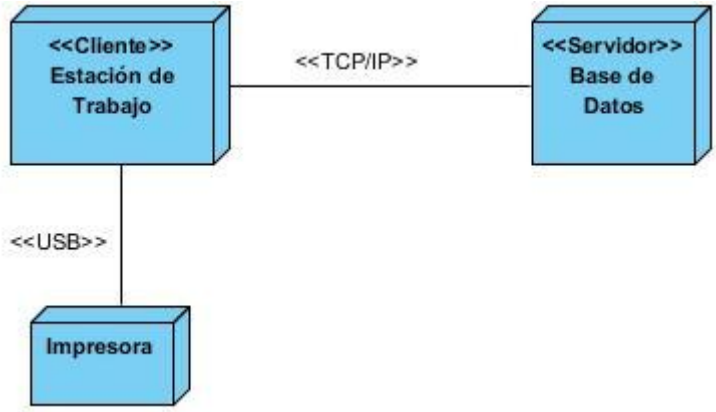

Figura 15: Diagrama de despliegue

Las prestaciones de los nodos cliente y servidor pueden ser consultadas en los requerimientos no funcionales números 8 y 9 respectivamente. La impresora es HP Laserjet P1006 y estará conectada por USB, con posibilidades de que la conexión se realice por TCP/IP.

3.4 Validaciones

Con el propósito de confirmar que todos los requisitos están incluidos en los casos de uso y que ninguno queda fuera de la solución propuesta, se decide hacer una matriz de trazabilidad, la cual puede ser consultada a continuación:

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10	CU11	CU12	CU13	CU14	CU15
RF1	X														
RF2	X														
RF3	X														
RF4						X									
RF5								X							
RF6		X													
RF7		X													
RF8		X													
RF9						X									
RF10									X						
RF11			X												
RF12			X												
RF13			X												
RF14						X									
RF15							X								
RF16										X					
RF17			X												
RF18			X												
RF19			X												
RF20				X											
RF21				X											
RF22				X											
RF23						X									
RF24							X								
RF25											X				
RF26					X										
RF27					X										
RF28					X										
RF29						X									
RF30							X								
RF31												X			
RF32													X		
RF33														X	
RF34															X

Tabla 5: Matriz de trazabilidad

3.5 Métricas de software

Uno de los objetivos de la ingeniería del software es producir un sistema, aplicación o producto de alta calidad. Para lograr este objetivo, los ingenieros de software deben emplear métodos efectivos junto con herramientas modernas dentro del contexto de un proceso maduro de desarrollo del software.

En la Ingeniería de Software la acción de medir es de vital importancia pues a través de ella se caracteriza, evalúa, predice y mejora el proceso de desarrollo del software

desde fases tempranas de su ciclo de vida. Con el fin de determinar la calidad[9] de un sistema informático se emplean las métricas que proporcionan a los desarrolladores una forma cuantitativa de valorar la calidad de los atributos internos de la aplicación que se construye.

"Una métrica de software relata de alguna forma las medidas individuales sobre algún aspecto (…). Un ingeniero de software desarrolla métricas para obtener indicadores. Un indicador es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso que permite al gestor de proyectos (…) ajustar el producto, el proyecto o el proceso (…). La única forma racional de mejorar cualquier proceso es (…) utilizar las métricas para proporcionar indicadores que conducirán a una estrategia de mejora". (Pressman, 2005)

3.5.1 Métricas para requisitos

La métrica relacionada con la calidad de la especificación del requisito proporciona una visión de cuan entendible por parte de los revisores es la descripción del requerimiento, está basada en la consistencia de la interpretación de cada requisito, la misma propone la siguiente fórmula:

$$Q1 = \frac{nui}{nr}$$

Donde:

nr: es la sumatoria de los requisitos funcionales y no funcionales del sistema:
$nr = nf + nnf$

nf : número de requisitos funcionales

nnf : número de requisitos no funcionales.

nui : número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

El resultado de la métrica es el valor obtenido en Q1 y mientras más cerca de 1 esté, menor será la ambigüedad de la especificación.

[9] Definida por la norma ISO 9000 como el grado en el que un conjunto de características inherentes cumple con los requisitos.

El equipo de revisión estuvo integrado por las personas que se relacionan a continuación en la Tabla 6.

Nombre y Apellidos	Rol que desempeña
Jorge Yuniel Jorrín Perdomo	Líder del proyecto
Lisset de Armas Hernández	Analista principal
Arnel Abad Noa	Arquitecto
Raúl Velázquez Álvarez	Asesor de calidad del centro CEGEL
Lianne Quincoces	Jefe Dpto. Ferias y Eventos

Tabla 6: Equipo de revisión

A continuación se muestra el resumen de los resultados obtenidos:

Atributo de calidad	Tipo de requisito	Interpretaciones	
		Iguales	Desiguales
Especificidad	Funcional	30	4
	No funcional	17	1
	Total:	**47**	**5**

Tabla 7: Interpretación de los requisitos

$nr = 34 + 18 = 52$

$Q1 = \frac{47}{52} = 0{,}90$

Como se puede apreciar el resultado de la métrica tiene a uno lo que refleja la existencia de poca ambigüedad en la especificidad de los requisitos.

3.6 Pruebas de software

Cuando se busca un instrumento que determine el status de la calidad de un producto de software, se piensa rápidamente en el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos del cliente. Las pruebas de software tiene gran importancia porque no solo propiciarán la obtención de un producto que satisfaga las necesidades del cliente, también con la detección de errores en las distintas etapas del ciclo de vida ayudarán a reducir los costos del proyecto y el tiempo que debe dedicar el equipo de desarrollo en la construcción del software.

Las pruebas son una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, los resultados se observan y se registran para realizar una evaluación del mismo (DTSI). En otras palabras es el proceso mediante el cual se ejecuta un programa con el fin de encontrar errores. El objetivo fundamental de las pruebas es detectar defectos en el sistema y se dice que es exitosa cuando descubre algún error; siempre que se revela un fallo se contribuye a elevar la calidad del producto.

Existen varias técnicas de prueba, entre las que se pueden mencionar estructurales o de caja blanca y funcional o de caja negra. La técnica de caja blanca se centra en la estructura interna del programa y la de caja negra en las funciones, las entradas y las salidas.

Actualmente existen numerosas herramientas para llevar a cabo una prueba y pueden ser concentradas en dos grupos: las manuales y las automatizadas; en el primero estarían las listas de chequeo y los casos de prueba, y en el segundo los software que automatizan este proceso.

Las listas de chequeo son formulario de preguntas, las cuales dependen del objetivo para el cual son usadas; son fáciles de aplicar, resumir y comparar. Proporcionan un apoyo mayor mediante preguntas que los probadores deben responder mientras leen el artefacto. Esta técnica proporciona listas que ayudan al probador a saber qué tipo de faltas buscar. En el Anexo 5 puede ser consultada la lista de chequeo aplicada a las descripciones de requisitos y casos de uso.

Los casos de prueba, en lo adelante CP, no son más que un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada; siempre es ejecutado como una unidad, desde el comienzo hasta el final. Los CP deben verificar:

- Si el producto satisface los requerimientos del usuario, tal y como se describe en la especificación de requisitos.
- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

En las Tablas 8 a la 13 puede ser consultado el caso de prueba correspondiente al caso de uso Gestionar local, los restantes CP pueden ser consultados en el Anexo 6. Al fragmento de funcionalidad Gestionar local le fueron realizadas tres iteraciones de

pruebas encontrándose 4 no conformidades que ya fueron resueltas, las mismas se muestran en el registro de no conformidades.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
Sección 1: Registrar local	EC 1.1: Registrar un local correctamente.	Se registra correctamente un local.	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar opción “Nuevo”.
	EC 1.2: Registrar un local con campos vacios o erróneos.	No se registra el local	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar opción “Nuevo”.
	EC 1.3: Cancelar la operación	No se registra el local	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar opción “Nuevo”.
Sección 2: Modificar local	EC 1.1: Modificar un local correctamente.	Se modifica correctamente un local.	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar la opción “Modificar”.
	EC 1.2: Modificar un local con campos vacios o erróneos.	No se modifica el local	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar la opción “Modificar”.
	EC 1.3: Cancelar la operación	No se modifica el local	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar opción “Modificar”.
Sección 3: Eliminar local	EC 1.1: Eliminar un local correctamente.	Se elimina correctamente un local.	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar un local y la opción Eliminar”.
	EC 1.2: Eliminar un local sin previa selección del mismo.	No se elimina el local	1. Seleccionar opción “Recursos” en el menú principal. 2. Seleccionar opción “Gestionar local”. 3. Seleccionar opción “Eliminar”.

Tabla 8: Secciones del caso de prueba

ID	Esc.	Código	Descripción	Superficie total	Sub-local	Local Ppal	Área	P.U	Respuesta del sistema	Resultado de la prueba
EC 1.1:	Registrar un local correctamente.	V	V	V	N/A	N/A	N/A	N/A	Se registra el local y se muestra un mensaje notificando el registro.	Satisfactorio
		V	V	V	N/A	V	N/A	N/A		
		V	V	V	N/A	N/A	N/A	V		
EC 1.2:	Registrar un local con campos vacios o erróneos.	V	V	I	N/A	N/A	N/A	I	Muestra un mensaje para que sean llenados los campos obligatorios.	Satisfactorio
		V	V	V	N/A	V	N/A	V		
		V	I	I	N/A	I	N/A	V		
		V	I	I	N/A	I	N/A	I		
EC 1.3:	Cancelar la operación	V	V	V	N/A	N/A	N/A	V	Se cancela el registro.	Satisfactorio

Tabla 9: Sección 1: Registrar local

ID	Esc.	Código	Descripción	Superficie total	Sub-local	Local Ppal	Área	P.U.	Respuesta del sistema	Resultado de la prueba
EC 1.1:	Modificar un local correctamente.	V	V	V	N/A	N/A	N/A	N/A	Se modifica el local y se muestra un mensaje notificando la modificación	Satisfactorio
		V	V	V	N/A	V	N/A	N/A		
		V	V	V	N/A	N/A	N/A	V		
EC 1.2:	Modificar un local con campos vacios o erróneos.	V	V	I	N/A	N/A	N/A	I	Muestra un mensaje para que sean llenados los campos obligatorios.	Satisfactorio
		V	V	V	N/A	V	N/A	V		
		V	I	I	N/A	I	N/A	V		
		V	I	I	N/A	I	N/A	I		
EC 1.3:	Cancelar la operación	V	V	V	N/A	N/A	N/A	V	Se cancela el registro.	Satisfactorio

Tabla 10: Sección 2: Modificar local

ID	Esc.	Código	Descripción	Superficie total	Sub-local	Local Ppal	Área	P.U	Respuesta del sistema	Resultado de la prueba
EC 1.1:	Eliminar un local correctamente.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Se elimina el local y muestra un mensaje de notificación.	Satisfactorio
EC 1.2:	Eliminar un local sin selección previa	N/A	N/A	N/A	N/A	N/A	N/A	N/A	No se elimina el local	Satisfactorio

Tabla 11: Sección 3: Eliminar local

No.	Nombre del campo	Clasificación	Valor nulo	Descripción
1.	Código	Campo de texto	No	Es un valor (número) generado por el sistema
2.	Descripción	Campo de texto	No	Cadena de caracteres (números y/o letras)
3.	Superficie total	Campo de texto	No	Sólo números.
4.	Sub-local	Campo de selección		
5.	Local principal	Lista desplegable	No (Si se selecciona la opción anterior)	Listado de todos los locales principales existentes.
6.	Área	Campo de selección		
7.	Precio unitario	Campo de texto	No (Si se selecciona la opción anterior)	Sólo números.

Tabla 12: Descripción de variables

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Resp. equipo de desarrollo
Botón Aceptar	1	Error ortográfico	Falta la r en la palabra Aceptar.	Pruebas en el proyecto	x			R	
Funcionalidad Insertar	2	Error al registrar el local	No muestra mensaje de confirmación.	Pruebas en el proyecto	x			R	
Funcionalidad Insertar	3	Cuando seleccionas sub-local no se habilita la lista desplegable	Campo de selección sub-local, lista desplegable sub-local.	Pruebas en el proyecto	x			R	
Funcionalidad Modificar	4	Error al visualizar información	Lista desplegable sub-local.	Pruebas en el proyecto	x			R	

Tabla 13: Registro de defectos y dificultades encontradas

3.7 Conclusiones del capítulo

En este capítulo fueron tratados los temas relacionados con la implementación y las pruebas del módulo, se obtuvieron los artefactos: diagrama de componente y despliegue, así como el resultado satisfactorio de las pruebas aplicadas, además se puede señalar que:

- El uso de métricas definidas para los requisitos contribuyó a la corrección y mejor calidad de los mismos, obteniéndose resultados satisfactorios.
- La verificación a través de las diferentes técnicas y métodos de calidad, hizo posible la corrección de los errores existentes en los artefactos que se generan durante el proceso de desarrollo de software.

Conclusiones

Con el desarrollo de esta investigación se pudo llegar a las siguientes conclusiones:

- El estudio previo de las características que presentan los software de gestión de eventos y ferias comerciales, específicamente en lo relacionado al manejo de recursos, sirvió de base para el desarrollo del nuevo producto de software que se presenta.
- El trabajo comprendió varios pasos, se seleccionó la metodología de desarrollo, definió la arquitectura, las herramientas y tecnologías a utilizar para desarrollar la aplicación.
- Del resultado de las entrevistas con el cliente fueron identificados un total de 34 requisitos funcionales y 18 requisitos no funcionales.
- El diseño y la implementación fueron llevados a cabo de manera satisfactoria obteniéndose en cada momento los artefactos propuestos por la metodología de desarrollo seleccionada.
- La realización de las pruebas de caja negra y listas de chequeo, arrojaron resultados satisfactorios.
- Como resultado del proceso de desarrollo de software llevado a cabo se obtuvo un nuevo producto que cumple con los requisitos establecidos por el cliente y los estándares de calidad.

Recomendaciones

Una vez terminada la investigación y cumplidos sus objetivos, teniendo en cuenta la experiencia y los resultados obtenidos, se recomienda:

- Utilizar el sistema en la Cámara de Comercio de Cuba y en otras instituciones del Estado cubano que se dediquen a la gestión de ferias y eventos.
- Obtener una nueva versión del producto que disponga de un módulo de reportes para aumentar así la satisfacción del cliente.

Bibliografía

Analitica. Manual de Diagramacion de Procesos Bajo Estandar BPMN. [En línea] [Citado el: 13 de enero de 2012.] http://www.analitica.com.co/website/images/stories/documentosTecnicos_SGP/Manual%20de%20Diagramacion%20de%20Procesos%20Bajo%20Estandar%20BPMN.pdf.

Apache Foundation. 2011. Apache Tomcat Wiki. [En línea] 2011. [Citado el: 20 de Febrero de 2012.] http://wiki.apache.org/tomcat/.

Avila Mojica, Jamir Antonio y Bustacara Medina, César Julio. Pattern Oriented Software Architecture. Reflection. *Pontificia Universidad Javeriana.* [En línea] [Citado el: 23 de ferero de 2012.] http://sophia.javeriana.edu.co/~cbustaca/Patrones_Arquitectura/Clases/reflection2.pdf.

Bagües, Ramiro Lago. 2007. Proactiva. [En línea] Abril de 2007. [Citado el: 20 de 02 de 2012.] http://www.proactiva-calidad.com/java/patrones/proxy.html.

Barbosa Guerrero, Lugo Manuel. 2006. Arquitectura de Software como eje temático de investigación. 2006. Vol. 4.

Bass, L, P, Clements y R, Kazman. 1998. *Software Architecture in Practice.* s.l. : Addison-Wesley, 1998.

Bizagi. 2009. Bizagi. [En línea] 2009. [Citado el: 12 de febrero de 2012.] "http://wiki.bizagi.com/es/index.php?title=Patrones".

Bonillo, Pedro. 2006. *Metodología para la gerencia de los procesos de negocio sustenda en el uso de patrones.* Venezuela : s.n., 2006.

Bonitasoft. [En línea] [Citado el: 16 de febrero de 2012.] http://es.bonitasoft.com/productos/beneficios.

—. Benefits | Bonitasoft. [En línea] [Citado el: 16 de febrero de 2012.] http://es.bonitasoft.com/productos/beneficios.

BonitaSoft. 2011. BonitaSoft. [En línea] 2011. [Citado el: 05 de 02 de 2012.] http://es.bonitasoft.com/vision-general/bonita-studio.

Booch, G, Jacobson, I y Rumbaugh. 1997. *The UML specification documents.* Santa Clara, CA : Rational Software Corp., 1997.

Booch, Grady, Rumbaugh, James y Jacobson, Ivar. 1998. *The Unified Modeling Language User Guide.* Massachusetts : Addison-Wesley Longman Inc., 1998. 0-201-57168-4.

Budd, Timothy A. *Introducción a la Programación Orientada a Objetos.* Oregon State University : s.n.

Buschmann, Frank. 1996. *Pattern-Oriented Software Architecture.* s.l. : John Wiley & Sons Ltd,, 1996.

Cámara de Comercio Bilbao. Cámara de Comercio Bilbao. [En línea] [Citado el: 21 de Febrero de 2012.] http://www.camarabilbao.com.

Cámara de Comercio de Cuba. 2009. Cámara de Comercio de Cuba. [En línea] 2009. [Citado el: 20 de 10 de 2011.] http://www.camaracuba.cu/.

Cámara de Comercio de Florencia para el Caquetá. CAE - Centro de Atención Empresarial. [En línea] [Citado el: 17 de 02 de 2012.] http://www.ccflorencia.org.co/servicio-al-cliente/cae-centro-de-atencion-empresarial.html.

Cámara de Comercio de la República de Cuba. 2009. *Portal Cámara de Comercio de la República de Cuba.* [En línea] 2009. [Citado el: 4 de noviembre de 2011.] http://www.camaracuba.cu/index.php?option=com_content&view=article&id=44&Itemid=54.

Cámara de Comercio de Madrid. Cámara de Comercio de Madrid. [En línea] [Citado el: 21 de Febrero de 2012.] http://www.camaramadrid.es.

Cámara de Comercio de Medellín para Antoquia. Cámara de Comercio de Medellín. [En línea] [Citado el: 17 de 02 de 2012.] http://www.camaramedellin.com.co/site/Tramites-Virtuales/Inscripciones-virtuales.aspx.

Cámara de Comercio de Quito. Cámara de Comercio de Quito. [En línea] [Citado el: 21 de Febrero de 2012.] http://www.lacamaradequito.com.

CCIA. 2009. Sistemas Cliente/Servidor. [En línea] 27 de 09 de 2009. http://ccia.ei.uvigo.es/docencia/SCS.

Ciberaula. 2010. Ciberaula. [En línea] 2010. [Citado el: 16 de 02 de 2012.] http://www.ciberaula.com/curso/java2/que_es/.

Citas de Negocios. *Expocomer 2012.* [En línea] [Citado el: 15 de noviembre de 2011.] http://www.expocomer.org/index.html?js=1&why=.

COPLEC. COPLEC -Comunidad de Programadores de Software libre del Ecuador. [En línea] [Citado el: 16 de 02 de 2012.] http://www.coplec.org/?q=book/export/html/240.

Cunningham, Ward. 2003. *Enterprise Solution Patterns Using Microsoft .NET.* s.l. : Microsoft Corporation, 2003.

Delgado, Andrea. *Desarrollo de Software con enfoque en el Negocio.*

Desarrollo de n-capas. [En línea] [Citado el: 23 de febrero de 2012.] http://html.rincondelvago.com/desarrollo-de-n-capas.html.

Dpto. Sistemas Inf. y Comp. Rational Unified Process https://pid.dsic.upv.es

DSI, Departamneto de Sistemas Informáticos. Modelo de Implementación: Diagramas de Componentes y Despliegue. [En línea] [Citado el: 25 de marzo de 2012.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

DTSI, Departamento de Tecnologías y Sistemas de Información. Pruebas del Software. [En línea] [Citado el: 29 de marzo de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema09.pdf.

E, Gamma, y otros. 2003. *Design Patterns. Elements of Reusable Object-Oriented.* s.l. : Adisson Wesley, 2003.

EJIE S.A. JUnit. Manual de Usuario. [En línea] [Citado el: 14 de febrero de 2012.] http://www.ejie.net/documentos/Herramientas/JUnit.%20Manual%20de%20Usuario%20v1.0.pdf.

Enciclopedia Libre Universal en Español. Enciclopedia Libre Universal en Español. [En línea] [Citado el: 16 de 02 de 2012.] http://enciclopedia.us.es/index.php/Java_2_Enterprise_Edition.

Estevez, Isabel Peréz. 2002. *Métricas para el control de proyectos de software.* Ciudad de la Habana : s.n., 2002.

Fowler, Scott. 1997. *UML gota a gota.* s.l. : PRENTICE HALL, 1997.

Freeman, P. 1980. *The Context of Desing in Software Desing Techniques.* 3ra. s.l. : IEEE Computer Society Press, 1980.

Funciones y ventajas. *Doblemente · Soluciones de comercio electrónico.* [En línea] [Citado el: 21 de noviembre de 2011.] http://www.doblemente.com/es/productos/funciones-y-ventajas,f33456520299793f.html.

Gamma, y otros. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

García, Joaquín. Patrones de diseño. Análisis y Diseño. *Ingenieria del Software.* [En línea] [Citado el: 9 de febrero de 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

Gracía, Felix Oscar y Vizcaíno, Aurora. Trabajando con Visual Paradigm. [En línea] [Citado el: 7 de febrero de 2012.] http://personales.unican.es/ruizfr/is1/doc/lab/01/is1-p01-trans.pdf.

Gracia, Joaquin. 2003. IngenieroSoftware. *IngenieroSoftware.* [En línea] 2003. [Citado el: 08 de 04 de 2012.] http://www.ingenierosoftware.com/analisisydiseno/casosdeuso.php.

Grupo de investigación eumednet (SEJ-309) de la Universidad de Málaga. Enciclopedia virtual. [En línea] [Citado el: 08 de Abril de 2012.] http://www.eumed.net/libros/2010b/698/Requisitos%20funcionales.htm.

Grupo Kaizen Calidad . 2005. *Cómo desarrollar el enfoque de procesos.* 2005.

Grupo Satelite s.a de c.v. 2011. Visual paradigm. [En línea] 2011. [Citado el: 06 de 02 de 2012.] http://www.gruposatelite.net/index.php?option=com_content&view=article&id=66&Itemid=69.

Gunnar Övergaard, Karin Palmkvist. 2004. *Use Cases Patterns and Blueprints.* s.l. : Addison Wesley, 2004.

Hernández Aguilar, MSc. Violena. 2010. *Validación y verificacion de software.* La Habana : Calisoft. Centro para la excelencia en el desarrollo de proyectos tecnológicos , 2010.

Hurtado González, Yudiel E. y Durán Vienes, Annia. 2008. Sistema de Métricas para Requisitos Funcionales. La Habana : Universidad de Ciencias Informáticas, 2008.

I. Jacobson,Booch, G and Rumbaugh J. 2000. *El proceso unificado de desarrollo de software.* Madrid : s.n., 2000.

IEEE SA - 1471-2000. *IEEE Recommended Practice for Architectural Description for Software-Intensive Systems.* [En línea] [Citado el: 28 de enero de 2012.] http://standards.ieee.org/findstds/standard/1471-2000.html.

IEEE, 1028. *IEEE Std 1028™ (1997) .* s.l. : IEEE Standard for Software Reviews.

IEEE, 829. *IEEE Std 829™ (1998/2007).* s.l. : IEEE Standard for Software Test Documentation.

Introducción a Java. [En línea] [Citado el: 23 de febrero de 2012.] http://codigoprogramacion.com/java/47-introjava.html.

Inyección de Dependencias. **Torres, Jose Miguel.** s.l. : msdn, dotNetMania.

IoC. Inyección de dependencias e Inversión de control. [En línea] [Citado el: 23 de enero de 2012.] http://es.scribd.com/doc/60751645/21/Inyeccion-de-dependencias-e-Inversion-de-control.

ISO 9126. 2005. "Software product evaluation – Quality characteristics and guidelines for their use". 2005.

ISO. 2001. *Orientación acerca del enfoque basado en procesos para los sistemas de gestión de la calidad.* 2001.

ISTQ. 2007. *Certified Tester Foundation Level Syllabus.* s.l. : International Software Testing Qualifications Board, 2007.

Jacobson, Ivar, Booch, Grady y Rambaugh, James. 2000. *El Proceso Unificado de Desarrollo de Sostware.* Madrid : Addison Wesley, 2000.

John Metsker, Steven y C Wake, William. 2006. *Design Patterns in Java.* s.l. : Addison-Wesley, 2006.

Jorge, Rodríguez. 2005. *Spring y el principio de Hollywood.* 2005.

Jurídica, Directora. 2009. *Manual de la Dirección Jurídica.* 2009.

Keith, Mike y Schincariol, Merrick. Pro JPA 2.

Kiczales, G., y otros. 1997. *Aspect-Oriented Programming.* s.l. : Xerox Palo Alto Research Center, 1997.

Kruchten, P. 2000. *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley, 2000.

Kuhn, Thomas S. 1971. *La estructura de las revoluciones científicas.* México : Fondo de Cultura Econámica, 1971.

Labrador, Ramón M. Gómez. 2005. *Tipos de Licencias de Software.* 2005.

Lago Bagués, Ramiro. 2007. Patrón "Modelo-Vista-Controlador". *Proactiva.* [En línea] 2007. [Citado el: 23 de febrero de 2012.] http://www.proactiva-calidad.com/java/patrones/mvc.html.

Larman, Craig. 1999. *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999.

Lourdes, Zaragoza Santiago María de. UTVM Calidad Académica de Primer Nivel. [En línea] [Citado el: 26 de 02 de 2012.] http://www.utvm.edu.mx/OrganoInformativo/orgJul07/RUP.htm.

Marañón, Gonzalo Álvarez. 1997-1999 . Departamento de Tratamiento de la Información y Codificación,CSIC. [En línea] 1997-1999 . [Citado el: 25 de 02 de 2012.] http://www.iec.csic.es/criptonomicon/java/quesjava.html.

MasterMagazine. 2004. MasterMagazine. [En línea] 2004. [Citado el: 12 de Enero de 2012.] http://www.mastermagazine.info/termino/4720.php.

Merseguer, José. 2010. *Ingeniería del software II.* 2010.

Modeling tool for business process in BPMN with simulation. *Visual Paradigm Fundation.* [En línea] [Citado el: 7 de febrero de 2012.] http://www.visual-paradigm.com/product/bpva/provides/.

Morena, Verónica de la. 2008. Connexions. [En línea] 27 de Noviembre de 2008. http://cnx.org/content/m17457/latest/.

NetBeans. *Portal del IDE Java de Código Abierto.* [En línea] [Citado el: 8 de febrero de 2012.] http://netbeans.org/index_es.html.

Olguín Espinoza, José Martín. 2004. *Análisis Orientado a Objetos. Ingeniería del Software.* 2004.

Oracle Corporation y / o sus filiales. 2012. NetBeans IDE. [En línea] 2012. [Citado el: 17 de 02 de 2012.] http://netbeans.org/community/releases/70/.

Pereda, Héctor Fernández. *Enfoque a Procesos.*

PerfectTablePlan. *PerfectTablePlan: software de gestión de eventos.* [En línea] [Citado el: 15 de noviembre de 2011.] http://www.programastop.com/perfecttableplan-software-de-gestion-de-eventos-bodas-banquetes-ceremonias/.

Piattini Velthuis, Mario G. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Madrid : Ra-Ma, 1996.

PostgreSQL Fundation. PostgreSQL. [En línea] [Citado el: 8 de febrero de 2012.] http://www.postgresql.org.es/sobre_postgresql.

Pressman, Roger. 2005. *Ingeniería del Software. Un enfoque práctico.* 5ta. La Habana : Felix Varela, 2005.

—. 2002. *Ingeniería del Software. Un enfoque práctico.* s.l. : McGraw Hill, 2002.

Pressman, Roger S. 2002. *Ingeniería de Software.Un enfoque práctico.* 2002.

Protocolo FTP: Clientes y Servidores. *MundoPC.NET.* [En línea] [Citado el: 30 de enero de 2012.] http://mundopc.net/protocolo-ftp-clientes-y-servidores/.

Protocolo TCP/IP. *EcuRed.* [En línea] [Citado el: 30 de enero de 2012.] http://www.ecured.cu/index.php/Protocolo_TCP/IP.

Pruebas de Software. *Gestión de calidad y pruebas de software.* [En línea] [Citado el: 02 de abril de 2012.] http://pruebasdesoftware.com/laspruebasdesoftware.htm.

Quintero, Reina y Antonia. 2000. Visión General de la Programación Orientada a Aspectos. Universidad de Sevilla : s.n., 2000.

R. Pascual, Inmaculada. 1996. elmundo.es. [En línea] 30 de Junio de 1996. http://www.elmundo.es/sudinero/noticias/act-37-4.html..

Reynoso, Carlos Billy. 2004. Profundizando en Estilos de Arquitectura de Software. Universidad de Buenos Aires : s.n., 2004.

Rondon Grados, Luis. 2009. JPA. *Java Persistence API.* [En línea] 2009. [Citado el: 21 de febrero de 2012.] http://luchorondon.blogspot.com/2009/04/jpa-java-persistence-api.html.

Silberschatz, Abraham, Korth, Henry F. y Sudarshan, S. 2002. *Fundamentos de Bases de Datos.* 4ta. Madrid : McGRAW-HILL, 2002.

Software de gestión. [En línea] [Citado el: 21 de noviembre de 2011.] http://www.regonline.com.es/__articles/products/software~de~gesti%C3%B3n~de~eventos.

Software para organización de eventos online. *Amiando.es.* [En línea] [Citado el: 21 de noviembre de 2011.] http://es.amiando.com/.

SQLite. *AplicacionesEmpresariales.com.* [En línea] [Citado el: 23 de febrero de 2012.] http://www.aplicacionesempresariales.com/sqlite-el-motor-de-base-de-datos-agil-y-robusto.html.

SQLite Fundation. SQLite. [En línea] [Citado el: 23 de febrero de 2012.] http://www.sqlite.org/.

UML, BPMN and Database Tool for Software Development. *Visual Paradigm Fundation.* [En línea] [Citado el: 7 de febrero de 2012.] http://www.visual-paradigm.com/.

Una Introducción a Apache. [En línea] [Citado el: 17 de febrero de 2012.] http://linux.ciberaula.com/articulo/linux_apache_intro/.

Universidad de Valladolid. Departamento de Informática. [En línea] [Citado el: 13 de 02 de 2012.] http://www.infor.uva.es/~jmrr/tgp/java/JAVA.html.

Universidad Salesiana de Bolivia. 2009. Universidad Salesiana de Bolivia. [En línea] 09 de 06 de 2009. virtual.usalesiana.edu.bo/web/practica/archiv/compon.doc.

Vega Miniet, Yanet. 2012. Proyecto Técnico. Proyecto de Gestión de Ferias y Eventos. 2012.

Vogel, Lars. 2012. JPA 2.0 con EclipseLink. *Tutorial.* [En línea] 16 de marzo de 2012. [Citado el: 2012 de marzo de 30.] http://www.vogella.com/articles/JavaPersistenceAPI/article.html.

Weske, Mathias. 2007. *Business Process Management. Concepts, Languages, Architectures.* Berlin : Springer, 2007.

Zamitiz, Ing. Carlos Alberto Román. Temas especiales de computación. [En línea] [Citado el: 26 de 02 de 2012.] http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html.

Zukowski, John. 2003. *Programación Java 2 J2SE 1.4.* Madrid : Anaya Multimedia, 2003.

Anexos

ANEXO 1. DESCRIPCIÓN DE REQUISITOS

Este documento contiene la descripción de los requisitos funcionales. (Consultar documento con el nombre “0129 Descripción de Requisitos.doc” en la carpeta Artefactos).

ANEXO 2. ESPECIFICACIÓN DE CASOS DE USO

Este documento contiene la especificación de los casos de uso del módulo. (Consultar documento con el nombre “0114 Especificación de Casos de Uso.doc” en la carpeta Artefactos).

ANEXO 3. MODELO DE ANÁLISIS

Este documento contiene el modelado del análisis para cada uno de los casos de uso del módulo. (Consultar documento con el nombre “Modelo de análisis.doc” en la carpeta Artefactos).

ANEXO 4. MODELO DE DISEÑO

Este documento contiene el modelado del diseño para cada uno de los casos de uso del módulo. (Consultar documento con el nombre “0121 Modelo de diseño v2.0.doc” en la carpeta Artefactos).

ANEXO 5. LISTA DE CHEQUEO

Este documento contiene las listas de chequeo aplicadas a las descripciones de los requisitos y los casos de uso. (Consultar documento con el nombre “Lista de chequeo.doc” en la carpeta Artefactos).

ANEXO 6. DISEÑO DE CASOS DE PRUEBA

Este documento contiene el diseño de los casos de prueba para cada uno de los casos de uso del módulo. (Consultar documento con el nombre “0122 Modelo de Casos de Prueba.doc” en la carpeta Artefactos).