MINISTERIO DE EDUCACIÓN SUPERIOR
UNIVERSIDAD DE CIENCIAS INFORMÁTICAS

Sistema para la Inscripción, Acreditación y Creación de Catálogos de la Cámara de Comercio de Cuba.

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

AUTOR

Dina Benitez Castellanos

TUTOR

Ing. Lisett De Armas Hernández

Ciudad de la Habana

Diciembre/2012

"Si piensas que los usuarios de tus programas son idiotas, sólo los idiotas usarán tus programas".

Linus Torvalds

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconocemos 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: Dina Benitez Castellanos. **Tutor:** Ing. Lissett De Armas Hernández.

_____ _____

Firma del Autor Firma del Tutor

Datos del Contacto

Tutor: Lisett De Armas Hernández (ldearmas@uci.cu) Graduada de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI). Pertenece al Centro de Gobierno Electrónico (CEGEL). Se desempeña como líder del subproyecto de Gestión de Ferias y Eventos de la Cámara de Comercio de Cuba.

Agradecimientos

- A mi mamá Vivian, porque me apoyó sin cesar en todo momento con el único objetivo de convertirme en una profesional. Por su visión futurista y por esa gracia para dar consejos en los momentos difíciles. Por ser madre, padre y amiga.
- A Omar que ha sido como un padre para mí, pues siempre me ha dado lo máximo de sí, te quiero papi.
- A mi primo Carlitín, ese primo hermano que se parece mucho a mí y que aunque nunca se dio cuenta me ha ayudado en toda mi estancia en la universidad, siempre estaba ahí cuando pensaba que no tenía a nadie con quien hablar.
- A Aliuska que ha sido la muchacha que no me dejaba dormir las mañanas para que me pusiera en función de la Tesis.
- A Milena, Samuell, Tomy, al piquete del café entero, nunca los olvidaré. Ellos hicieron de una forma u otra que la pasara muy bien en cualquier lugar donde estuvieran.

A todos ustedes les agradezco el ser parte de mi vida, su apoyo incondicional y el que ahora gracias a cada porción de lo que me brindaron ayudó a que hoy yo sea una profesional.

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 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, que permite orientar las mejores alternativas para el desarrollo de la actividad empresarial de las entidades que a esta asociación pertenecen. Dicha institución 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, brindar a la Cámara de Comercio de Cuba, un producto de software capaz de satisfacer un conjunto de necesidades funcionales que ayudarán al cliente a realizar su trabajo lo mejor posible.

Este trabajo consta de tres capítulos, el primer capítulo se basa en un estudio del estado del arte de software de gestión de ferias y eventos, metodología de desarrollo, patrones arquitectónicos y de diseño, así como las herramientas de implementación que serán utilizadas para el desarrollo de las diferentes funcionalidades. El segundo capítulo expone cómo funciona el negocio, además del análisis y diseño del producto. El tercer capítulo describe lo referente a la implementación y las pruebas de calidad de software.

Índice

Índice de Figuras

Índice de Tablas

Introducción

La tecnología y su vertiginoso desarrollo han cambiado los estilos y modos de vida, sin dudas que las tecnologías de la información constituyen una prueba irrefutable de tal desarrollo, por su creciente y dinámico ascenso. La informatización es un proceso innegable que humaniza el trabajo y en la actualidad es cada vez más usado por el sistema empresarial en el mundo. Cuba no está al margen de tal desarrollo y su sistema empresarial hace un uso creciente de las Tecnologías de la Información y las Comunicaciones (TIC). El gobierno y el sistema empresarial tienen como política el desarrollo de las tecnologías informáticas necesarias para mejorar los sistemas de gestión de la producción, la eficiencia económica y los negocios entre otros, que les permitan adentrarse y competir en un mundo cada vez más globalizado.

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, que permite orientar las mejores alternativas para el desarrollo de la actividad empresarial de las entidades que a esta asociación pertenecen. En ella se presentan problemas relacionados con la organización, ejecución y seguimiento de las ferias que tiene como antecedentes la existencia de un sistema llamado SAEFE, que tiene como objetivo gestionar los eventos y ferias que la Cámara de Comercio organiza.

SAEFE es un sistema que no realiza de forma correcta las funcionalidades de inscripción, acreditación y creación de catálogos para los eventos que la Cámara de Comercio gestiona. Las deficiencias del SAEFE se ven reflejadas a la hora de gestionar un evento ya que no se recoge toda la información necesaria, los usuarios y entidades no pueden ser registradas correctamente dada por la vaga funcionalidad del producto, además de esto el sistema tampoco genera un catálogo del evento en el cual brinde informaciones de las empresas, imposibilitando el contacto entre diferentes entidades. Por tal situación la información que se recibe de las empresas que participan en los eventos es incompleta lo cual, dificulta la identificación de intereses comunes y por tanto se pierden oportunidades de negocio.

Los representantes de la Cámara de Comercio han optado por alquilar software que utilizan otras empresas para gestionar sus eventos con el objetivo de poder seguir llevando a cabo las ferias y eventos de dicha entidad. Todo esto incluye gastos de dinero ya que deben pagar por la utilización del software. Esto no pasara si el SAEFE no tuviera las carencias que presenta.

En el presente Trabajo de Diploma se pretende resolver la situación planteada, partiendo de las necesidades que tiene la Cámara de Comercio. Como resultado de lo anteriormente expuesto, se define como **problema a resolver:** Las ineficiencias en la inscripción, acreditación y creación de catálogos en los eventos que gestiona la Cámara de Comercio están provocando que se pierdan oportunidades de negocio y que no se realice de forma ágil y segura la gestión de las ferias y eventos comerciales.

El proceso de desarrollo de software constituye **el objeto de estudio.**

El **objetivo general** es desarrollar un sistema informático para la inscripción, acreditación y creación de catálogos, de manera tal que se mitiguen las pérdidas de oportunidades de negocio y se realicen de forma ágil y segura la gestión de las ferias y eventos comerciales.

El **Campo de acción** es el análisis, diseño e implementación de sistemas de gestión de ferias y eventos comerciales.

Para resolver el problema planteado la investigación se propone defender la siguiente **idea**: El desarrollo de un sistema informático para la inscripción, acreditación y creación de catálogos, permitirá mitigar las pérdidas de oportunidades de negocio y contribuirá a la realización de forma ágil y segura de la gestión de las ferias y eventos comerciales.

Para dar respuesta al objetivo general se proponen cumplir los siguientes **objetivos específicos**:

- Fundamentar teóricamente el desarrollo de un sistema informático para la inscripción, acreditación y la creación de catálogos.
- Realizar la captura de requisitos de los procesos de inscripción,

acreditación y creación de catálogos de las ferias y eventos comerciales.

- Diseñar e implementar los requisitos obtenidos.
- Verificar y validar la solución propuesta.

Para dar respuesta y cumplimiento a los objetivos específicos se planifican las siguientes **tareas** en el proceso de desarrollo de la investigación:

1. Revisión bibliográfica para la elaboración del marco teórico de la investigación y obtener antecedentes sobre el tema propuesto, como es la existencia de aplicaciones semejantes en otros países y las herramientas más utilizadas en este sentido en el mundo.
2. Análisis de las herramientas más utilizadas a nivel mundial para el desarrollo de este tipo de aplicaciones.
3. Selección de las herramientas a utilizar en el desarrollo del sistema informático.
4. Entrevista con el cliente para definir requerimientos del sistema informático, así como, recopilación de información (incluye todos los procesos de inscripción, acreditación y creación de catálogos de la cámara de comercio)
5. Elaboración de los artefactos necesarios los cuales son el modelo de negocio y el levantamiento de requisitos.
6. Desarrollo de las funcionalidades de inscripción, acreditación y creación de catálogos de las ferias y eventos comerciales.
7. Validación con el cliente.
8. Realización de las pruebas de calidad del software.

Para la realización de las tareas mencionadas anteriormente se emplearon métodos científicos. Los mismos se dividen en teóricos y empíricos. Los métodos teóricos posibilitan las condiciones para buscar más que las características triviales de la realidad, permiten explicar los hechos y profundizar en las principales relaciones y cualidades de los fenómenos, hechos y procesos.

Se utilizarán los métodos de investigación cuantitativa y cualitativa teniendo en cuenta las referencias de Reyes (s.a.), Hernández Sampieri (2003) y Rodríguez

y García (2004).

Los métodos de investigación que se utilizarán los siguientes:

Los teóricos:

- **Analítico – Sintético:** a partir del análisis de teorías, tendencias documentos relacionados con el tema sintetizar los elementos más importantes que se relacionan con el objeto de estudio, para expresar de manera resumida la posición del investigador.
- **Histórico – Lógico:** al hacer un estudio crítico de los trabajos anteriores en este contexto para utilizarlos como punto de referencia y comparación, además para constatar teóricamente cómo ha evolucionado el tema en el tiempo.
- **Modelación:** para la realización de los diagramas necesarios en el proceso de desarrollo de software, haciendo una representación abstracta de la solución, facilitando así el desarrollo de la misma.

Los **métodos empíricos** revelan, describen y explican las características y relaciones esenciales del objeto basando su contenido en la experiencia. Dentro de este otro grupo se utilizan los métodos:

- **Entrevista:** con el objetivo de obtener información de interés para poder comprender el funcionamiento del proceso de asignación de recursos de las ferias Comerciales y la gestión de catálogos.
- **Observación:** permite un mejor entendimiento, comprensión y caracterización del proceso de asignación de recursos de las Ferias comerciales y gestión de catálogos.
- **Medición:** a través del uso de métricas de calidad y pruebas para comprobar la funcionalidad, usabilidad y seguridad, entre otras que aseguren la calidad del sistema.

Estructura de la investigación

Capítulo 1: Fundamentación Teórica: el estudio de estado del arte en el mundo y en Cuba, así como una descripción de las tendencias, técnicas, tecnologías, metodologías y herramientas usadas para dar solución al problema.

Capítulo2: Características del Sistema: describe el flujo actual de los procesos del negocio y se realiza el análisis para definir las funcionalidades y los requisitos no funcionales del sistema a desarrollar. Análisis y Diseño del sistema: Especificación de requisitos, Diagramas de caso de uso, Diagrama de Clases, Modelo Entidad-Relación, Diagramas de Secuencia y Diagramas de Colaboración.

Capítulo3: Implementación y Pruebas: Se describe en términos de componentes, las principales clases y subsistemas definidos durante el diseño y se muestra el diagrama de componentes general, el cual organiza las dependencias entre estos. Además, se describen las clases más significativas del sistema. Se realizan las pruebas de Caja Blanca y Caja Negra para validar la propuesta de solución.

Capítulo 1
[Fundamentación teórica]

1.1. Introducción:

En este capítulo se realiza una investigación sobre sistemas que presentan similitud con la solución propuesta, lo cual es importante para la asimilación de procesos ya existentes y el nivel de desarrollo a nivel mundial y nacional. Además se realiza un estudio sobre las tecnologías existentes, seleccionando las necesarias para el cumplimiento de los objetivos.

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 se pueden mencionar: Perfect Table Plan, Reg Online, Ofi eventos, Gestiona Eventos, etc, 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 tienes varias características en común, las cuales son:

- Gestionar y planificar los eventos.
- Gestionar y registrar los participantes y organizadores implicados en el evento.
- Facilitar toda la información necesaria a cualquier empresa o particulares que tenga la necesidad de organizar un evento del tipo que sea.
- Facilitar lugares de realización de los eventos como son: centros de reuniones, hoteles, catering entre otros.
- Enviar correos personalizados, mensajes de confirmación, recordatorios, datos de inscripción y alojamiento.
- Imprimir credenciales.
- Favorecer las relaciones entre las partes implicadas ayudando a la colaboración, arribando ambos a un mutuo acuerdo.
- Realizar balances económicos, generar facturas, resúmenes de

facturación e informes de rentabilidad y ganancias.

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.

El sistema SAEFE es el que se utiliza en Cuba para gestionar las ferias y eventos de la Cámara de Comercio, dicho sistema se caracteriza por ser poco flexible a los cambios, pues no permite la modificación dinámica de sus componentes, no realiza de forma correcta las funcionalidades de inscripción, acreditación y creación de catálogos imposibilitando la inserción o modificación de un evento, de una entidad, de los participantes y por último no permite la generación del catálogo del evento.

Por lo antes expuesto se hace necesaria la creación de un sistema informático para la inscripción, acreditación y creación de catálogos, que garantice de forma ágil y segura estos procesos y que contribuya a mitigar las pérdidas de oportunidades de negocio entre las entidades participantes en los eventos.

1.3. Metodología de Desarrollo

En todas las fases del desarrollo de software se realizan una serie de tareas para obtener el producto final o resultado esperado, los componentes del software pasan por varias etapas durante su ciclo de vida. Cada una de estas tareas se ejecutan con diferentes herramientas y técnicas que ayuden a su elaboración y conclusión, para esto se tiene en cuenta que orden se le da al cumplimiento de cada una, ya que muchas de ellas dependen de otras que deben realizarse primero. No se puede pasar por alto la documentación a utilizarse y los diferentes artefactos a generar para dar solución a la misma.

Para obtener un producto final exitoso se hace necesario el uso de una metodología que integre un conjunto de procedimientos, técnicas, herramientas, y un soporte documental que ayude a los desarrolladores a realizar un software. (16)

La Universidad de las Ciencias Informáticas tiene como política que todos sus

proyectos se desarrollen guiados por el Programa de Mejoras de la Institución, por lo cual se está exigiendo una mayor calidad en el desarrollo del software, encaminado con el objetivo de la certificación internacional del nivel 2 del modelo CMMI. Para suplir esta necesidad se decide utilizar como metodología de desarrollo de software: Rational Unified Process (RUP).

1.3.1. Rational Unified Process (RUP)

RUP es un proceso iterativo e incremental, lo que facilita que sea un proceso planificado y gestionado que (10):

- Se adapta a los cambios de los requerimientos con pocas alteraciones.
- Involucra al usuario/cliente durante el proceso.
- Permite detectar y gestionar los riesgos durante todo el ciclo de vida del proyecto.
- Se basa en la construcción de prototipos ejecutables.

Además de las características propias de los procesos iterativos se caracteriza por (10):

- Dirigido por casos de uso
- Centrado en la arquitectura.
- Utiliza Lenguaje Unificado de Modelado (UML) como lenguaje.

RUP es un proceso configurable y a la vez una metodología muy extensa, en la mayoría de los casos en el momento de su implantación se considera un proceso costoso por la cantidad de entregables y actividades que se definen, pero hay que tener en cuenta, una de las principales funcionalidades es que no obliga al uso de todas las actividades y entregables definidos, sino que se puede configurar el proceso, con el fin de adaptarlo a aquellas partes que se consideran necesarias.

El proceso RUP se repite en una serie de ciclos. Cada ciclo concluye con una versión del producto (release) y cada ciclo está dividido por 4 fases: Inicio (Concepción), Elaboración, Construcción y Transición. Cada una de las fases está dividida a su vez por iteraciones, y en cada una se realizan 5 procesos o

flujos de trabajo principales: requerimientos, análisis, diseño, implementación, pruebas y entrega o preparación del reléase (10).

- 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. (17)

RUP consta de nueve flujos de trabajo seis de ellos son para el desarrollo y tres para soporte, estos flujos se desarrollan en todas las fases del desarrollo de software unos con más fuerza que otros. A continuación se describen los flujos de trabajos orientados al desarrollo con esta metodología.

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.

Se escogió como metodología de desarrollo de software a RUP ya que los artefactos que genera permiten un mejor entendimiento del negocio a los desarrolladores, además de que permite realizar varios flujos a la vez logrando con esto capturar errores en etapas iniciales. Genera una gran cantidad de documentación que es muy útil para proyectos de grandes dimensiones, no siendo así para proyectos pequeños para los que se dispone de poco tiempo y de un equipo de desarrollo pequeño como es el caso del sistema para la inscripción, acreditación y creación de catálogos de la Cámara de Comercio de Cuba q tiene como objetivo la gestión de los eventos. A esto se le adiciona que es la única metodología que puede detectar las vulnerabilidades y riesgos del software incluso antes de que este sea implementado.

En la siguiente figura (Figura 1.) se muestra como se aplica RUP en las diferentes fases y flujos del desarrollo de software.

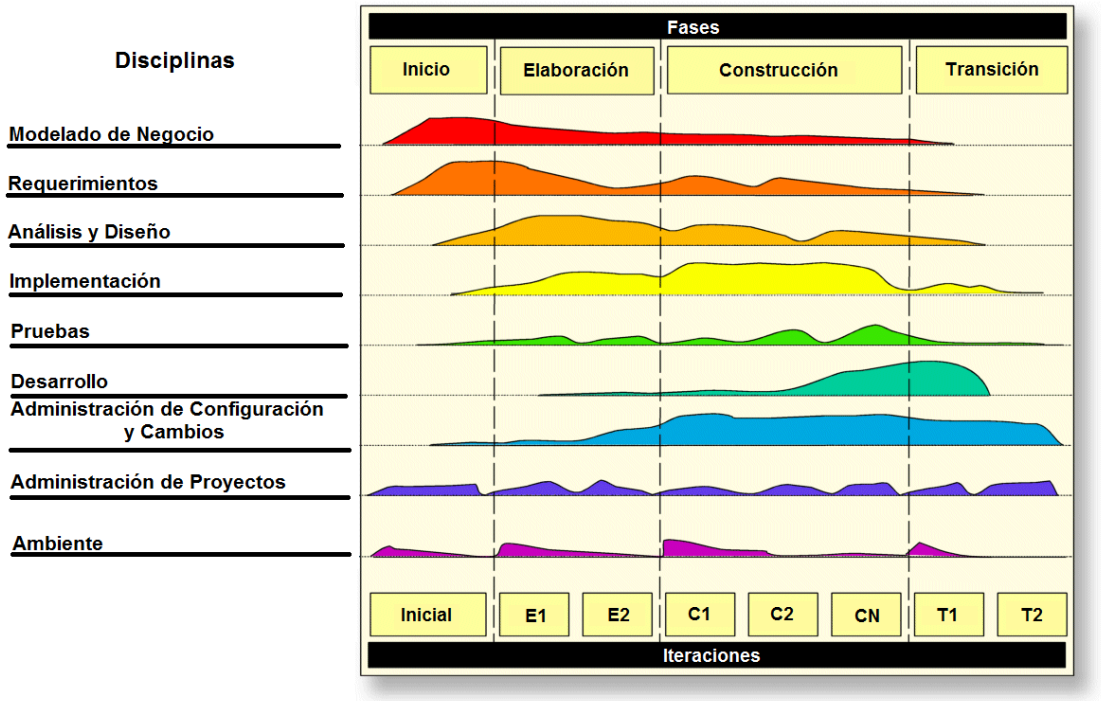


Figura 1. Proceso Unificado de Desarrollo de Software.

1.4 Herramientas y tecnologías

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

1.4.1. CASE (Computer Aided Software Engineering)

Las herramientas CASE (en español Ingeniería de Software Asistida por Computadora) se puede definir como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software (investigación Preliminar, Análisis, Diseño, Implementación y Despliegue.).

Dentro de este tipo de herramientas se encuentra Visual Paradigm la cual será usada para el modelado de los diagramas necesarios para el desarrollo del software.

Visual Paradigm (VP): Se utilizará VP 5.0 ya que es una herramienta que utiliza notación UML. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y

despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Está diseñado para una amplia gama de usuarios, incluidos los Ingenieros de Software, Analistas de Sistema, Analistas de Negocios, Sistema de Arquitectos igual, que estén interesados en la construcción de sistemas de software a gran escala mediante el uso fiable, es Orientado a Objetos y facilita la comunicación entre ellos. Facilita la interoperabilidad con otras herramientas CASE, la mayoría de los IDE's principalmente y permite la integración de todos los componentes. (6)

Esta versión 5.0 es la que será usada para el modelado de los artefactos necesarios en el desarrollo del nuevo producto de software.

1.4.2. Business Process Modeling Notation (BPMN)

BPMN que traducido al español significa Notación para el Modelado de Procesos de Negocio. Es una notación gráfica estandarizada que permite el modelado de procesos de negocio en un formato de flujo de trabajo. Tiene como principal objetivo proveer una notación estándar que sea fácilmente leíble y entendible por parte de todos los involucrados e interesados del negocio.

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. (11)

Con el fin de modelar los procesos de negocio de forma tal que los desarrolladores entiendan rápidamente el negocio del sistema, se optó por utilizar la notación BPMN, ya que brinda una perspectiva de la complejidad de los diferentes procesos y permite una mayor comprensión de estos.

1.4.3. Lenguaje Unificado de Modelado

Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado de sistemas más conocido y usado en la actualidad. Se define como un "lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de

software".

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. (26)

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). (26)

En la elaboración de los diagramas se hará uso de la notación UML en su versión 8.0.

1.4.4. Lenguaje de Programación. Java

Se utilizó el lenguaje JAVA para el desarrollo de la aplicación y para justificar su uso se hace énfasis en lo que expresa el libro blanco de Java al referir que los objetivos de su diseño eran ser "un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portátil, de gran rendimiento, multitarea y dinámico". A continuación se presentarán las ideas básicas que explican cada una de las mencionadas características de Java, que fundamentan su elección.

- **Sencillo**: Porque incorpora nuevas tareas como un recolector de elementos no utilizados. Otro aspecto de la simplicidad de Java es que nada es realmente nuevo, es decir, su conjunto de funciones procede de algún otro lenguaje.(13)
- **Orientado a objetos**: “Esencialmente, la programación orientada a objetos (POO) es un modo de 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. Un buen diseño conlleva a componentes reutilizables, extensibles y sostenibles." Java cuenta con bibliotecas de clases previamente diseñadas que se van enriqueciendo de una a otra versión y constituyen la base sobre la cual el programador trabaja. (13)

- **Distribuido**: Este lenguaje reconoce la red y admite el acceso a objetos distribuidos tan fácilmente como a los objetos locales. Mediante protocolos comunes y los servicios Web, se invocan métodos en un equipo remoto tan fácil e invisiblemente como puede hacerlo en su mismo espacio de ejecución.(13)
- **Interpretado**: "Los programas de Java son interpretados. En lugar de ser compilado en ejecutables nativos, el código de Java es traducido en códigos de bytes no asociados a una plataforma. Estos códigos de bytes 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 o revincularlos."(13)
- **Robusto**: La robustez es la medida de la fiabilidad de un programa y Java contiene varias funciones integradas que la garantizan, por ejemplo es un lenguaje basado en tipos, no tiene punteros, realiza automáticamente la recolección de elementos no utilizados y fomenta el uso de interfaces.(13)
- **Seguro**: La seguridad está ligada a la robustez, por ejemplo, al no tener punteros no se corrompe la memoria. Además tiene integradas otras funciones de seguridad como son: el verificador de código de bytes que proporciona el primer nivel de defensas, a continuación el cargador de clases y luego el administrador de seguridad, que refuerza las normas de seguridad para el entorno de ejecución. (13)
- **De arquitectura neutral**: Los programas de Java se realizan para que se ejecuten en cualquier equipo sin recompilarlos. (13)
- **Portátil e independiente de la plataforma**: Significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un

programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, write once, run everywhere. La portabilidad viene dada por la capacidad que tiene Java para ejecutar un mismo programa en cualquier arquitectura. Por otra parte el espacio que ocupan los programas es relativamente pequeño. (13)

- **De gran rendimiento**: Los códigos de bytes de plataforma neutral realmente pueden convertirse en tiempo de ejecución en código maquina específico de la CPU, ejecutándose rápidamente. Hay dos herramientas de traducción incluidas con Java que lo hacen automáticamente: el compilador Justo a tiempo (JIT) y HotSpot. (13)
- **Multitarea**: Un programa que ejecuta varias tareas o subprocesos simultáneamente, se dice que es multitarea, en Java se incluyen múltiples recursos con el objetivo de facilitar la comunicación garantizando la seguridad de los subprocesos. (13)
- **Dinámico**: Las bibliotecas de Java se encuentran evolucionando constantemente, sin embargo eso no implica que los programas anteriores dejen de funcionar. Otro elemento que explica el dinamismo es la preferencia de Java de las interfaces sobre las clases. (13)

Las características del leguaje anteriormente explicadas, unidas a que la plataforma Java forma parte de la familia de software libre, en la universidad existe previa explotación de dicho lenguaje además de una abundante bibliografía, estas fueron las causas que motivaron a la selección de este lenguaje.

1.4.5. Ambiente de Desarrollo Integrado. Netbeans.

Un Ambiente de Desarrollo Integrado es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs

pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, etc. (5)

Existen varios IDEs en la actualidad entre los cuáles se pueden mencionar C++Builder, Eclipse, Netbeans, Visual Studio, etc.

Para el proceso de codificación del sistema será utilizado Netbeans 7.1 por ser un producto de código abierto, proporciona herramientas para la construcción de todos los componentes J2EE, incluidas páginas web, servlets, necesarios para la confección de aplicaciones profesionales, provee igualmente soporte para el framework Spring. (5)

Es un producto libre y gratuito sin restricciones de uso; su versión 7.0.1 tiene las características mencionadas por lo que se decide seleccionarla para desarrollar el software.

1.4.6. Plataforma de Desarrollo. JEE (Java Enterprise Edition)

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de la compañía Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el Lenguaje de programación Java u otros lenguajes. En este caso, la plataforma no es un hardware específico o un sistema operativo, sino más bien una máquina virtual encargada de la ejecución de aplicaciones, y un conjunto de librerías estándar que ofrecen funcionalidad común. Hoy en día esta plataforma ha evolucionado en concordancia con el avance tecnológico y se ha convertido en una de las plataformas de programación más usadas por los desarrolladores. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que estén operando. Es toda una tecnología orientada al desarrollo de software con la cual se puede realizar cualquier tipo de programa. 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 integrabilidad. (27)

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 que puedan ser interpretados por el sistema operativo (SO), y así este pueda ejecutar el programa. 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.

1.4.7. Java Persistence API

Java Persistence API (JPA) proporciona un estándar para gestionar datos relacionales en aplicaciones Java SE o Java EE, de forma que además se simplifique el desarrollo de la persistencia de datos. En su definición, ha combinado ideas y conceptos de los principales frameworks de persistencia, como Hibernate, Toplink y JDO, y de las versiones anteriores de EJB. Todos estos cuentan actualmente con una implementación JPA.

El mapeo objeto-relacional (es decir, la relación entre entidades Java y tablas de la base de datos, queries con nombre, etc) se realiza mediante anotaciones en las propias clases de entidad. No se requieren ficheros descriptores XML. También pueden definirse transacciones como anotaciones JPA. (28)

1.4.8. Sistema gestor de base de datos (SGBD)

Debido a la necesidad de manipular datos persistentes y de larga durabilidad en el sistema se hace necesario la aplicación de un SGBD, a través del cual sea posible recuperar y almacenar información, interpretable y útil para el usuario, con facilidad y fiabilidad, de ahí que se ha convertido en el instrumento o soporte básico de mayor despliegue en la actualidad para la gestión de los sistemas informáticos por todo tipo de empresas, independientemente de su tamaño.

PostgreSQL: Para el desarrollo de este sistema se decidió utilizar PostgreSQL entre otras cosas porque permite la aproximación de los datos a un modelo objeto-relacional, siendo capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transacciones, optimización de consultas y herencia. Es altamente extensible: PostgreSQL soporta operadores, funciones, métodos de acceso y tipos de datos definidos por los usuarios, además soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92. Tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, Java, o TCL como lenguaje procedural embebido. (12)

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. (12)

Se utilizó como sistema gestor de base de datos ya que su código fuente se está disponible libremente. Además de que un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando.

1.4.9. Framework

Framework es un concepto sumamente genérico, se refiere a “ambiente de trabajo, y ejecución”. En general los framework son soluciones completas que contemplan herramientas de apoyo a la construcción (ambiente de trabajo o desarrollo) y motores de ejecución (ambiente de ejecución).

Framework puede ser algo tan grande como “.NET”, pero también el concepto se aplica a ámbitos más específicos, por ejemplo; dentro de Java en el ámbito especifico de aplicaciones Web se tiene los framework: Struts, “Java Server Faces”, o Spring. Estos frameworks de Java en la práctica son conjuntos de

librerías (API's) para desarrollar aplicaciones Web, librerías para su ejecución (o motor), y un conjunto de herramientas para facilitar esta tarea (debuggers, ambientes de desarrollo como Eclipse, Netbeans etc).

Como una de sus ventajas se encuentra la velocidad en el desarrollo de aplicaciones, código optimizado y reducción de costos; la mayor parte de ellos se basan en estándares. (29)

Spring: es un Framework que proporciona una infraestructura de código abierto que ayuda en el soporte y desarrollo de aplicaciones. Ayuda a estructurar aplicaciones completas en una manera consistente y productiva para crear arquitecturas coherentes.

Es el único framework que interviene en todas las capas arquitectónicas de una aplicación J2EE. Además está diseñado para facilitar una flexibilidad arquitectónica. Posee disímiles módulos que facilitan el desarrollo de una aplicación en Java/J2EE los cuales pueden ser usados 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. (7)

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. La implementación de JPA se suele llamar proveedor de persistencia y normalmente define los metadatos a través de notaciones en la clase 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. (30)

1.5. Arquitectura de software

La arquitectura de software propone las pasos que deben seguirse para realizar los procesos de análisis, diseño e implementación utilizando de forma eficaz los patrones, frameworks y estilos arquitectónicos que ayuden a desarrollar el sistema con calidad permitiendo de esta forma que los implicados

dígase desarrolladores tengan una idea clara y estén todo el tiempo orientados en lo que se debe implementar. (18)

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 como 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. (18)

"Un patrón arquitectónico expresa un esquema de organización estructural, fundamental para los sistemas de software, proporciona un conjunto de subsistemas predefinidos[1], especifica sus responsabilidades, e incluye normas y directrices para la organización de las relaciones entre ellos" (19).

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

1.5.1. Modelo Vista Controlador (MVC)

Para el desarrollo de este sistema se propone el modelo vista controlador (MVC) porque es un patrón de arquitectura de software que separa los datos

[1] Son el esqueleto de futuras aplicaciones.

de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. (37)

Este patrón es muy común a la hora de construir software debido a las facilidades que brinda, pues permite la reutilización y la independencia entre las capas, se pueden realizar cambios en capas sin tener que modificar las otras, facilita la estandarización y la utilización de los recursos. Las capas que componen el sistema se listan a continuación:

Datos (Modelo): Esta capa representa la estructura de datos. Las clases que define contendrán funciones que lo ayudarán a recuperar, insertar y actualizar información en la base de datos.

Presentación (Vista): Es la capa que ve el usuario, hay quien la denomina "capa de usuario", presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Negocio (Controlador): Esta capa sirve como un intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para procesar la petición HTTP y generar una página web. (20)

1.5.2. Arquitectura en capas

Este patrón define cómo organizar el modelo de diseño a través de capas, que pueden estar físicamente distribuidas, lo que significa que los componentes de una capa sólo pueden hacer referencia a componentes en capas inmediatamente inferiores. Además, dicho patrón simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. (20)

1.6. Patrones de diseño

Es recomendable realizar un estudio de los distintos patrones de diseño teniendo en cuenta la estrategia que se pretenda seguir arquitectónicamente. A continuación se especifican algunos de los que serán aplicados:

DAO (Data Access Object): se utiliza para abstraer y encapsular todos los accesos a la fuente de datos.

Singleton: Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

Proxy: Brinda un sustituto o representante de otro objeto para controlar el acceso a éste.

Facade: Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

Reflection: Proporciona un mecanismo para cambiar la estructura y comportamiento de sistemas de software dinámicamente, soporta la modificación de aspectos fundamentales, como tipos y mecanismos de invocación (Ávila Mojica, y otros).

Inversion of 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.

1.7. Paradigmas de Programación

Los paradigmas son marcos de referencia que imponen reglas sobre cómo se deben hacer las cosas, indican qué es válido dentro del paradigma y qué está fuera de sus límites. Un paradigma distinto implica nuevas reglas, elementos, límites y maneras de pensar, o sea implica un cambio. Los paradigmas pueden ser considerados como patrones de pensamiento para la resolución de problemas. Desde luego siempre teniendo en cuenta los lenguajes de programación.

Un paradigma de programación es un modelo básico de diseño y desarrollo de programas, que permite producir programas con unas directrices específicas, tales como: estructura modular, fuerte cohesión, alta rentabilidad, etc. (22)

1.7.1 Paradigma Orientado a Objetos (POO)

Es una metodología de diseño de software y un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que combinan estado (es decir, datos) y comportamiento (esto es, procedimientos o métodos). La programación orientada a objetos expresa un programa como un conjunto de estos objetos, que se comunican entre ellos para realizar tareas. Esto difiere de los lenguajes procedurales tradicionales, en los que los datos y los procedimientos están separados y sin relación. Estos métodos están pensados para hacer los programas y módulos más fáciles de escribir, mantener y reutilizar (22).

Ventajas:

- ✓ **Flexibilidad:** Partiendo del hecho que mediante la definición de clases se establecen módulos independientes, a partir de los cuales se definen nuevas clases, entonces se puede pensar en estos módulos como bloques con los cuales construir diferentes programas.
- ✓ **Reusabilidad:** Por medio de la reusabilidad se pueden utilizar clases definidas previamente en las aplicaciones en el momento que sea conveniente.
- ✓ **Mantenibilidad:** Las clases que conforman una aplicación, vistas como módulos independientes entre sí, son fáciles de mantener sin afectar a los demás componentes de la aplicación.
- ✓ **Extensibilidad:** Gracias a la modularidad y a la herencia una aplicación diseñada bajo el paradigma de la programación orientada a objetos puede ser fácilmente extensible para cubrir necesidades de crecimiento de la aplicación.(23)

Desventajas

A pesar de que las ventajas de la programación orientada a objetos superan a las limitaciones de la misma, se encuentran algunas características no deseables como son:

- ✓ Cuando se heredan clases a partir de clases existentes se heredan de forma implícita todos los miembros de dicha clase aun cuando no todos se necesiten, lo que produce aplicaciones muy grandes que no siempre encajan en los sistemas con los que se disponga.
- ✓ Velocidad de ejecución; la programación orientada a objetos crea aplicaciones innecesariamente pesadas, en muchas ocasiones es más lenta de ejecutar que una aplicación conformada únicamente por los módulos necesarios.(23)

1.7.2. Paradigma Orientado a Aspectos (POA)

La POA ha revolucionado en gran medida la forma de desarrollar aplicaciones informáticas. Al igual que otros tantos paradigmas, marca una manera diferente de ver el proceso de construcción de un sistema. Deviene además en una excelente opción cuando se desea ganar en flexibilidad y facilidad de mantenimiento en los diseños de sistemas. De modo que para muchos constituye una mejor vía de obtener un producto informático, en tanto para otros es solo un paradigma o técnica más de programación. Lo cierto es que es una útil herramienta y que, como es lógico, la decisión de emplearla o no está en función del análisis de otros elementos que permitan evaluar los beneficios de su introducción. (24)

La Programación Orientada a Aspectos (POA) es un paradigma de programación relativamente reciente cuya intención es permitir una adecuada modularización de las aplicaciones y posibilitar una mejor separación de conceptos. Gracias a la POA se pueden encapsular los diferentes conceptos que componen una aplicación en entidades bien definidas, eliminando las dependencias entre cada uno de los módulos. De esta forma se consigue razonar mejor sobre los conceptos, se elimina la dispersión del código y las implementaciones resultan más comprensibles, adaptables y reusables. Varias tecnologías con nombres diferentes se encaminan a la consecución de los mismos objetivos y así, el término POA es usado para referirse a varias tecnologías relacionadas como los métodos adaptivos, los filtros de composición, la programación orientada a sujetos o la separación multidimensional de competencias. (24)

Ventajas

- Como resultado se obtiene un código más limpio, menos duplicado, más fácil de entender y de mantener.
- La separación de conceptos permite agregar nuevos aspectos, modificar y / o remover aspectos existentes fácilmente.
- Puede retrasar las decisiones de diseño sobre requerimientos actuales o que surjan en el futuro, ya que permite, luego, implementarlos separadamente, e incluirlos automáticamente en el sistema.
- Al ser implementados separadamente, tiene mayor probabilidad de ser reusados en otros sistemas con requerimientos similares. (24)

Desventajas

- Posibles choques entre el código funcional (expresado en el lenguaje base) y el código de aspectos (expresados en los lenguajes de aspectos). Usualmente estos choques nacen de la necesidad de violar el encapsulamiento para implementar los diferentes aspectos, sabiendo de antemano el riesgo potencial que se corre al utilizar estas prácticas.
- Posibles choques entre los aspectos. El ejemplo clásico es tener dos aspectos que trabajan perfectamente por separado pero al aplicarlos conjuntamente resultan en un comportamiento anormal.
- Posibles choques entre el código de aspectos y los mecanismos del lenguaje. Uno de los ejemplos más conocidos de este problema es la anomalía de herencia. Dentro del contexto de la POA, el término puede ser usado para indicar la dificultad de heredar el código de un aspecto en la presencia de herencia. (24)

1.8. Verificación y Validación (V&V)

La verificación y validación es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes. La V&V es un proceso de ciclo de vida completo. Inicia con las revisiones de los requerimientos y continúa con las revisiones del diseño y las inspecciones del

código hasta la prueba del producto. Existen actividades de V&V en cada etapa del proceso de desarrollo del software.

La verificación y la validación no son la misma cosa, aunque es muy fácil confundirlas, Boehm (1979) expresó la diferencia entre ellas de forma sucinta:

- **Verificación**: ¿Se está construyendo el producto correctamente?

  El papel de la verificación comprende comprobar que el software está de acuerdo con su especificación. Se comprueba que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado.

- **Validación**: ¿Se está construyendo el producto concreto?

  La validación es un proceso más general. Se debe asegurar que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado.

Es importante llevar a cabo la validación de los requerimientos del sistema de forma inicial. Es fácil cometer errores y omisiones durante la fase de análisis de requerimientos del sistema y, en tales casos, el software final no cumplirá las expectativas de los clientes. Sin embargo, en la realidad, la validación de los requerimientos no puede descubrir todos los problemas que presenta la aplicación. Algunos defectos en los requerimientos solo pueden descubrirse cuando la implementación del sistema es completa.

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. 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. (31)

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. (31)

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. (31)

1.9. Conclusiones del Capítulo

Con el estudio profundo de la bibliografía consultada sobre diferentes aplicaciones de gestión de ferias y eventos comerciales en el mundo y en Cuba, se vio reflejada la abundante variedad de aplicaciones exitosas, en donde casi todas utilizan herramientas de software libre para su desarrollo. Una vez estudiadas las distintas herramientas y metodologías aplicables para la propuesta de la solución informática, dio como resultado que se escogiera como metodología a utilizar RUP, como notación para el modelado de los procesos de negocio BPMN, como gestor de bases de datos el PostgreSQL, como herramienta Case el Visual Paradigm, como lenguaje de programación Java, integrado en la plataforma JEE, y como patrón arquitectónico el MVC.

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 en los subprocesos de convocatoria y contratación, específicamente los procesos de inscripción y acreditación, además del proceso para generar el catálogo del evento en las ferias y eventos comerciales.

2.3. Propuesta de sistema

La propuesta de solución de este trabajo consiste en el análisis, diseño, implementación y prueba de un sistema para la inscripción, acreditación y creación de catálogos que contribuya a mitigar la pérdida de oportunidades de negocios y realice de forma ágil y segura la gestión de las ferias y eventos comerciales. Dicha solución estará provista de las funcionalidades necesarias para que exista un correcto funcionamiento de las tareas relacionadas con la generación de catálogos y la correcta gestión de los usuarios y entidades.

2.4. Breve descripción del Negocio

La Cámara de Comercio de Cuba de la República de Cuba es una entidad que entre sus funciones tiene la de organizar muchas de las ferias y eventos que se

celebran en el país, dicha entidad dispone para estas funciones de un sistema que presenta varios problemas no permiten que se cumpla el objetivo para el que fue creado. Para llevar a cabo una feria o evento es necesario que se inscriban usuarios participantes y organizadores, que se inserten eventos y que se genere un catálogo de dicho evento formando parte de los procesos de convocatoria y contratación que tienen como objetivo registrar a los usuarios y llevar a cabo el evento con los participantes en este.

Con este fin se efectúan varias tareas para gestionar un evento y que finalmente para llevarse a cabo debe estar registrada o registrarse una entidad organizadora de lo contrario no se puede llevar a cabo el evento.

2.4.1. Diagrama de proceso de negocio

A partir de la descripción antes realizada a continuación se muestra en la Figura 2 el diagrama del proceso de contratación que es donde se encuentra enmarcada la mayor parte de la solución que propone este trabajo.

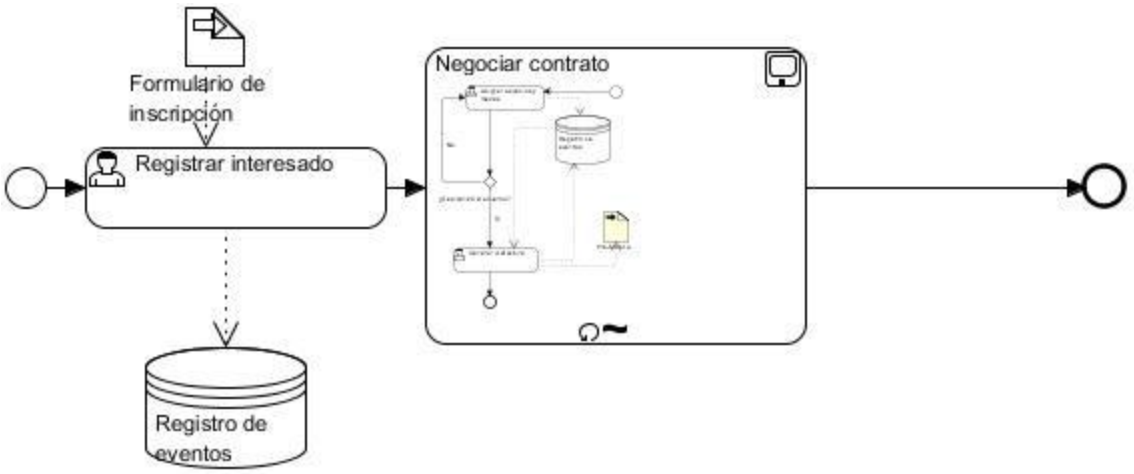


Figura 2: Diagrama de proceso de negocio: Proceso Contratación.

En la figura 3 se muestra el proceso convocatoria en el cual se crea un evento y por último es lanzada la convocatoria para que los usuarios interesados se inscriban en el evento.

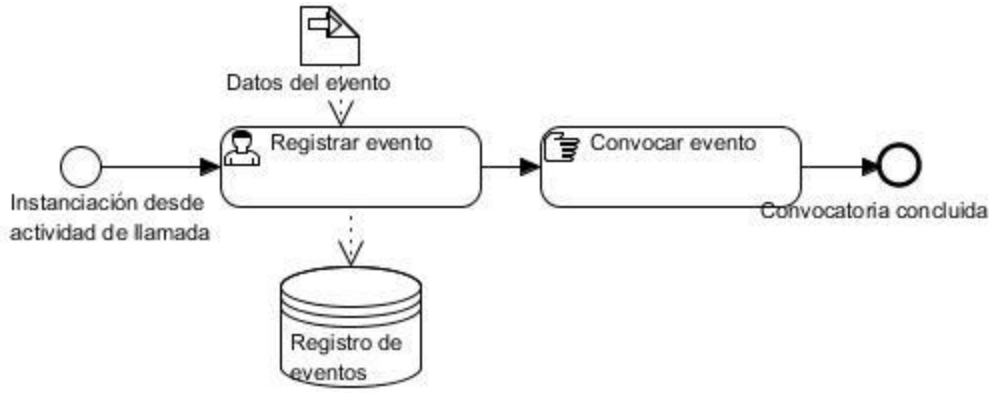


Figura 3: Diagrama de proceso de negocio: Proceso Convocatoria.

Por último se muestran todos los procesos que deben efectuarse en función de la organización del evento que luego de haber lanzado la convocatoria y de que los interesados dígase entidades participantes u organizadores hayan sido registrados en el sistema, se generará un catálogo del evento que almacenará informaciones importantes de las entidades favoreciendo así las oportunidades de negocio entre las partes implicadas. Ver figura 4.

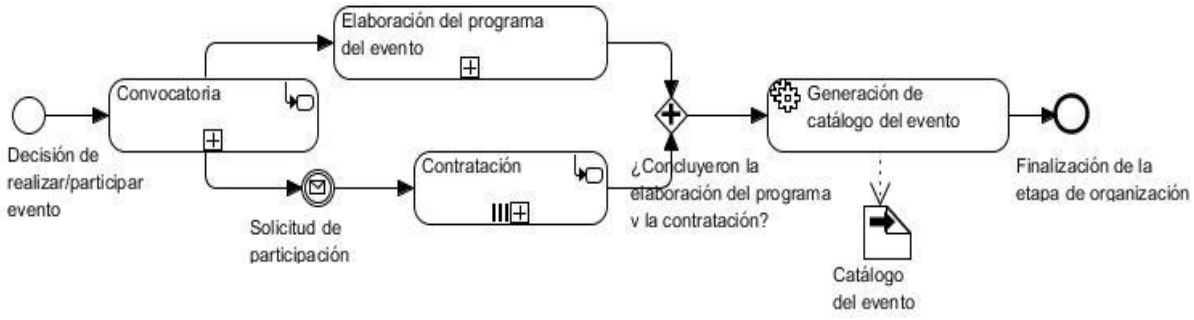


Figura 4: Diagrama de proceso de negocio: Ferias y Eventos.

2.5. 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 5.

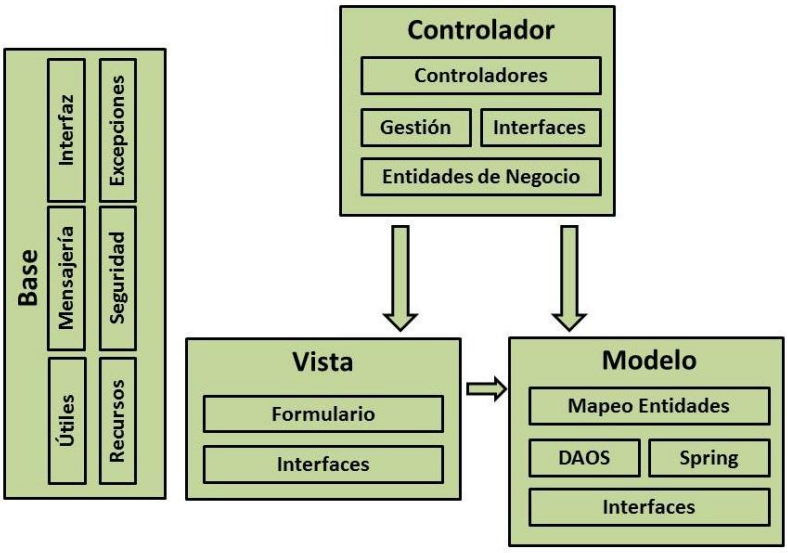


Figura 5. Arquitectura del Sistema

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 contiene las clases que se encargan de mostrar al usuario la información contenida en el modelo. 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 se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. Contiene los controladores, gestores y sus interfaces, además de las entidades de negocio.
- El modelo está compuesto por un conjunto de clases que representan la

información del mundo real que el sistema debe procesar sin tomar en cuenta ni la forma en la que esa información va a ser mostrada ni los mecanismos que hacen que esos datos estén dentro del modelo, es decir, sin tener relación con ninguna otra entidad dentro de la aplicación. El modelo está compuesto por el mapeo de entidades, el framework spring 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.

En el capítulo 3 será abordado este tema con mayor profundidad.

2.6. Especificación de los requisitos de software

Requerimientos Funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir; permiten expresar específicamente las responsabilidades del sistema que se propone, determinar de una manera clara lo que el sistema debe hacer y no alteran las funcionalidades del producto. (33)

Gestionar Evento	RF1.1: Registrar evento
	RF1.2: Listar eventos
	RF1.3: Consultar evento.
	RF1.4: Modificar evento.
	RF1.5: Eliminar evento
	RF1.6: Buscar evento.
Gestionar Entidad	RF2.1: Registrar entidad.
	RF2.2: Listar entidades.
	RF2.3: Consultar entidad
	RF2.4: Modificar entidad.
	RF2.5: Eliminar entidad.
	RF2.6: Buscar entidad.
	RF3.1: Registrar datos de contacto.
	RF3.2: Listar datos de contacto.

Gestionar Datos de Contacto	RF3.3: Modificar datos de contacto.
	RF3.4: Eliminar datos de contacto.
	RF3.5: Buscar datos de contacto.
Gestionar Catálogo	RF4.1: Preparar catálogo del evento.
	RF4.2: Generar catálogo del evento.
	RF4.3: Exportar catálogo del evento.
Gestionar Participante	RF5.1: Acreditar participante.
	RF5.2: Listar participante.
	RF5.3: Consultar participante.
	RF5.4: Modificar participante.
	RF5.5: Eliminar participante.
	RF5.6: Buscar participante

Tabla 1: Muestra los requisitos funcionales asociado a sus casos de uso.

Las descripciones de los requisitos mencionados pueden ser consultadas en el Anexo 1, a continuación en la Tabla 2 se muestra la descripción del requisito funcional Registrar evento.

Precondiciones	N/A.
Flujo de eventos	
Flujo básico Registrar Evento	
1.	El sistema permite registrar los datos del evento: código del evento, nombre, siglas, fecha de inicio, fecha de fin, tipo de evento. Además, permite seleccionar una entidad de las registradas en el sistema como organizadora o adicionar una nueva entidad.
2.	El usuario registra los datos del evento.
3.	El usuario selecciona una entidad registrada como organizadora.
4.	El sistema valida los datos.
5.	El sistema registra los datos y notifica al usuario.
6.	Concluye el requisito.
Pos-condiciones	
1.	Se registró un nuevo evento.
2.	Se relacionó la entidad seleccionada como organizadora del evento.
Flujos alternativos	
Flujo alternativo 3.a El usuario requiere adicionar una nueva entidad como organizadora.	
1	El sistema permite registrar el nombre, país de la entidad, dirección de la entidad y gestionar los datos de contacto (ver descripción del requisito Gestionar datos de contacto).
2	El usuario registra los datos.
3	Continúa en el paso 4 del flujo básico.
Pos-condiciones	
1	Se registró un nuevo evento.
2	Se registró una nueva entidad.
3	Se relacionó la entidad registrada como organizadora del evento.
Flujo alternativo 5.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	**Evento**	Visibles en la interfaz: código nombre siglas fecha de inicio fecha de fin tipo de evento Utilizados internamente: N/A
	Entidad	Visibles en la interfaz: nombre país dirección datos de contacto (teléfonos, fax, correo electrónico) Utilizados internamente: rol
	Cuenta Bancaria	Visibles en la interfaz: número moneda título titular sucursal dirección sucursal banco Utilizados internamente: N/A
	Sucursal	Visibles en la interfaz: sucursal dirección sucursal Utilizados internamente: N/A
	Banco	Visibles en la interfaz: nombre Utilizados internamente: N/A
Requisitos especiales	N/A.	
Asuntos pendientes	N/A.	

Tabla 2: Descripción textual RF Registrar evento.

2.6.2. Requerimientos No Funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener (33). Entre los requisitos no funcionales definidos se

encuentran:

Usabilidad

RnF1: 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.

RnF2: 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.

RnF3: Mínimo impacto de los errores.

- El sistema debe permitir cancelar los flujos y sus tareas.
- El sistema debe notificar a los usuarios los errores y sugerir cómo corregirlos.

Confiabilidad

RnF4: Notificación de omisión ó errores en los datos introducidos.

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

Eficiencia

RnF5: 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

RnF6: Servidores de base de datos locales PostgreSQL versión 9.1.

RnF7: Máquina Virtual de Java en su versión JDK 1.6.

Hardware

RnF8: Hardware mínimo para oficinas:

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

RnF9: Hardware mínimo para servidores:

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

Soporte

RnF10: Portabilidad

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

Restricciones de diseño

RnF11: 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

RnF12: La autenticación estará basada en el uso de credenciales.

RnF13: Las contraseñas se almacenarán en la base de datos haciendo uso del algoritmo hash MD5[2].

RnF14: Los usuarios estarán autorizados a realizar las acciones que se encuentran definidas para el rol al cual pertenece.

RnF15: Los documentos que exporta el sistema deben estar dotados de las configuraciones necesarias para que sean de solo lectura.

Estándares aplicables

RnF16: 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

Un caso de uso define la secuencia de transacciones desarrolladas por un sistema en respuesta a un evento que inicia un actor al interactuar con él, este actor representa el rol que juega una o varias personas, un equipo u otro sistema automatizado en el propio sistema. Por la marcada importancia de estos en aras de favorecer el funcionamiento de un sistema software, se hace imprescindible definir y describir específicamente cuáles interactúan con el sistema en cuestión.

2.7.1. Definición de los actores

La siguiente tabla muestra el actor que interactúa con el sistema el cual será el encargado de hacer uso de las funcionalidades descritas con anterioridad. Ver

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

Tabla 3.

Actor	Objetivo
Organizador.	Registrar y consultar la información relativa a la organización, ejecución y seguimiento a las ferias y exposiciones comerciales.

Tabla 3: Descripción de los actores del sistema.

2.7.2. Listado de casos de uso.

En la siguiente tabla se muestra la relación de los requisitos funcionales con el actor, permitiendo una vista clara de la cantidad de requisitos pertenecientes a cada caso de uso. Para ello ver la Tabla 4.

CU 1	Gestionar Evento
Actor	Organizador
Descripción	Permite registrar, modificar, eliminar, buscar, consultar y mostrar los eventos a realizar en la Cámara de Comercio de Cuba.
Referencia	RF 1.1, RF 1.2, RF 1.3, RF 1.4, RF 1.5, RF 1.6

CU 2	Gestionar Entidad
Actor	Organizador
Descripción	Permite registrar, modificar, eliminar, buscar, consultar y mostrar las entidades que tiene la Cámara de Comercio para la realización de los eventos.
Referencia	RF 2.1, RF 2.2, RF 2.3, RF 2.4, RF 2.5, RF 2.6

CU 3	Gestionar Datos de Contacto
Actor	Organizador
Descripción	Permite registrar, modificar, eliminar, buscar y listar los datos de contacto de las entidades y participantes que tiene la Cámara de Comercio de Cuba para la realización de los eventos.
Referencia	RF 3.1, RF 3.2, RF 3.3, RF 3.4, RF 3.5

CU 4	Gestionar Catálogo
Actor	Organizador
Descripción	Permite preparar, generar y exportar el catálogo del evento que tiene la Cámara de Comercio de Cuba para la realización de los eventos.
Referencia	RF 4.1, RF 4.2, RF 4.3

CU 5	Gestionar Participante
Actor	Organizador
Descripción	Permite registrar, listar. Consultar, modificar, eliminar y buscar a los participantes en los eventos que realiza la Cámara de Comercio de Cuba.
Referencia	RF 5.1, RF 5.2, RF 5.3, RF 5.4, RF 5.5, RF 5.6

Tabla 4. Resúmenes de casos de uso

2.7.3. Diagrama de casos de uso

La forma en que interactúa cada actor del sistema con el sistema se representa con un Caso de Uso. Los Casos de Uso son los que conducen toda la arquitectura del sistema. A continuación se representa el Modelo de Casos de Uso del Sistema (MCUS), modelo que contiene actores, casos de uso y sus relaciones. En la figura 6 se muestra la descripción del Modelo de Casos de Uso.

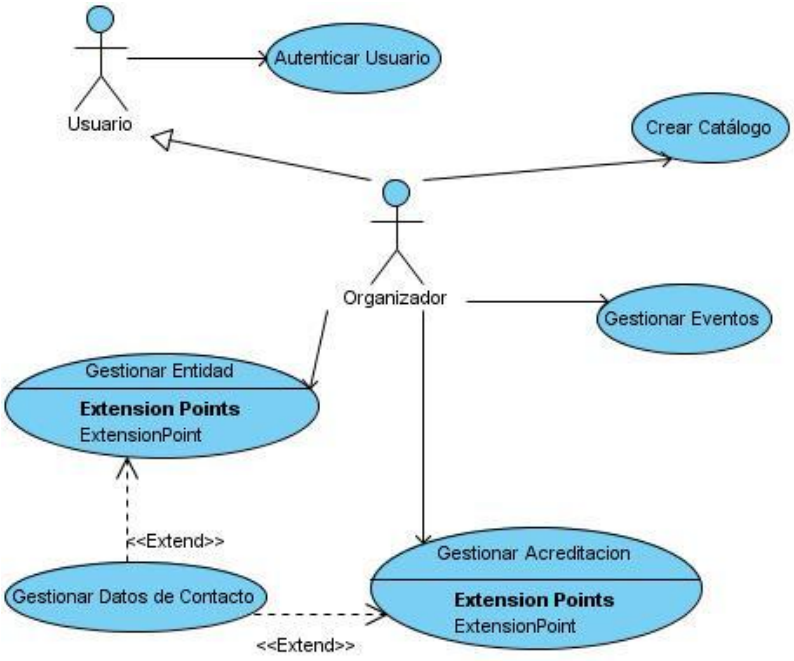


Figura 6: Modelo de Casos de Uso del Sistema: Funcionalidades de inscripción, acreditación y creación de catálogos de los procesos convocatoria y contratación.

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

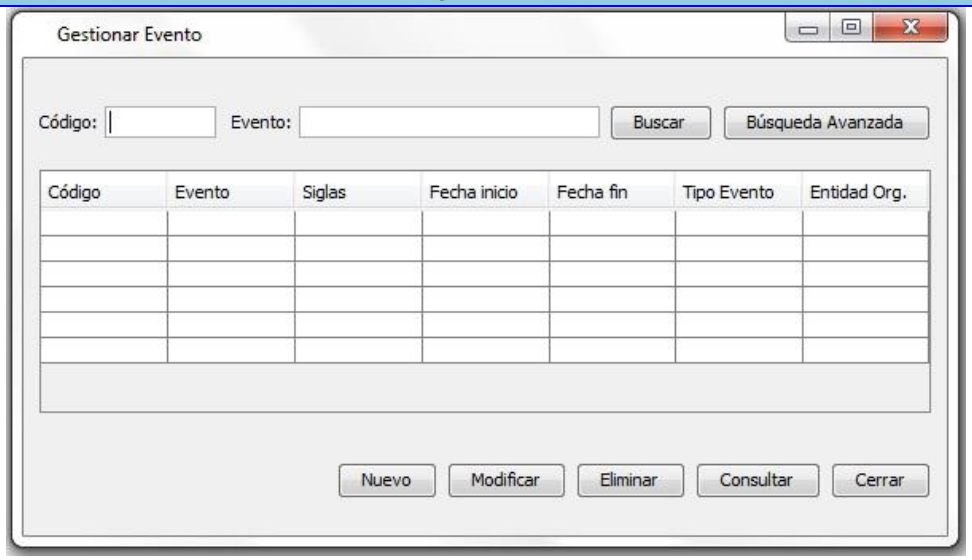
cuáles son las repuestas del mismo. Por la importancia que presenta el caso de uso Gestionar evento a continuación en la Tabla 5 se muestra la descripción textual del mismo, las restantes descripciones con sus prototipos de interfaz pueden ser consultadas en el Anexo 2.

Descripción del Caso de Uso: Gestionar Evento

Objetivo	Insertar un evento al sistema con su entidad organizadora.	
Actores	Organizador	
Resumen	Este caso de uso inicia cuando el organizador necesita gestionar los eventos. Dicha gestión está enmarcada en registrar eventos, listar eventos creados hasta el momento, modificarlos en caso de que se desee, actualizar algún dato, eliminarlos, buscarlos y ver sus detalles.	
Complejidad	Baja	
Prioridad	Crítico	
Precondiciones	El organizador debe estar autenticado con los permisos necesarios.	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	El Organizador selecciona Gestionar eventos.	
2.		El sistema muestra la interfaz Gestionar eventos donde se pueden realizar las siguientes opciones: • Registrar evento. • Modificar evento. • Eliminar evento. • Consultar evento. • Buscar. • Búsqueda avanzada. En la interfaz se muestran los criterios de búsqueda: • código • nombre Además se muestra un listado de eventos, visualizándose de los mismos los siguientes datos, ordenados por fecha de creación del evento. • código • nombre • siglas • fecha de inicio • fecha de fin • tipo de evento • entidad organizadora. a. Al mostrar la interfaz por primera vez aparecerán todas las opciones deshabilitadas con excepción de:

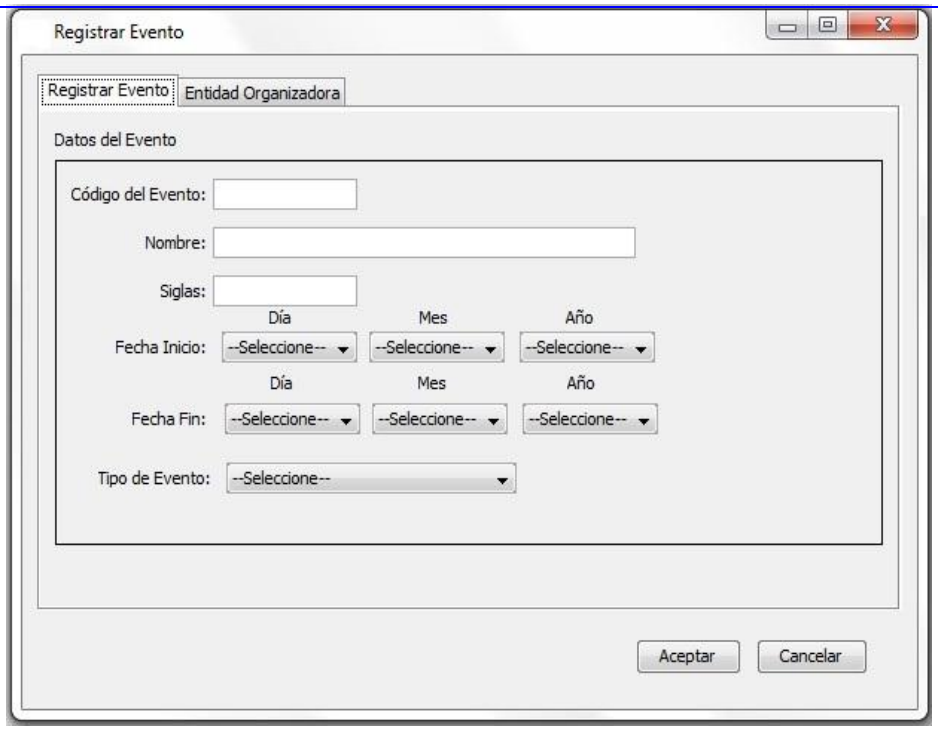
		Registrar evento, Búsqueda y Búsqueda avanzada permaneciendo así hasta que el organizador seleccione un evento. b. Si el organizador selecciona un evento se habilitan las opciones de Modificar evento, Consultar evento y Eliminar evento.
3.	• Si selecciona la opción “Registrar evento” ver sección “Registrar evento” • Si selecciona un evento y selecciona la opción Modificar evento, ver sección “Modificar evento”. • Si selecciona un evento y selecciona la opción Eliminar evento, ver sección “Eliminar evento”. • Si selecciona un evento y selecciona la opción “Consultar evento” ver sección “Consultar evento”. • Si selecciona la opción “Buscar”, ver sección “Buscar”. • Si selecciona la opción “Búsqueda avanzada”, avanzada, ver sección “Búsqueda avanzada”,	
4.		Se termina el caso de uso.

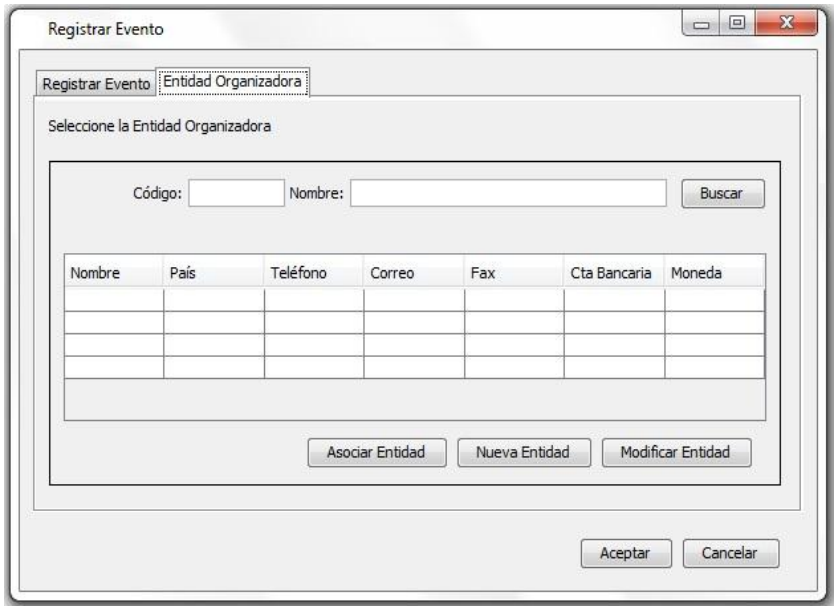
Prototipo de Interfaz



Sección		
Sección “Registrar evento”		
Acción del Actor		**Respuesta del Sistema**
1		El sistema muestra la interfaz **Registrar evento** en el cual se muestran los siguientes datos: • nombre • siglas • fecha de inicio • fecha de fin • tipo de evento • entidad organizadora

		Esta interfaz ofrece además las opciones habilitadas: ✓ Aceptar. ✓ Cancelar. ✓ Seleccionar Entidad Organizadora. ✓ Crear nueva Entidad Organizadora.
2	El Organizador introduce todos los datos solicitados y elige la opción **Aceptar**. Si selecciona la opción Seleccionar Entidad Organizadora ver **Flujo Alterno 4.** Si selecciona la opción Crear nueva Entidad Organizadora ver **Flujo Alterno 5.** Si selecciona la opción de Cancelar, ver **Flujo alterno 1.** Si cierra la interfaz, ver **Flujo alterno 2.**	
3		Si los datos son válidos, el sistema guarda los datos. Si los datos insertados son inválidos, ver **Flujo alterno 3**.
Prototipo de Interfaz		



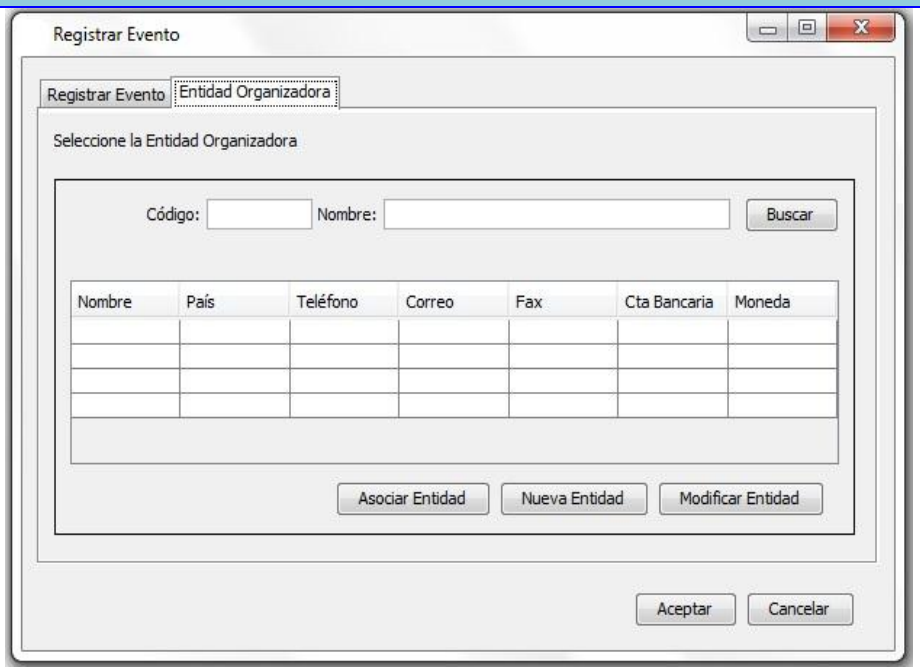


Flujos Alternos		
Flujo alterno 1		
Acción del Actor		**Respuesta del Sistema**
1		El sistema cierra la interfaz sin realizar ningún cambio. Regresa al paso 3 del **Flujo normal de eventos.**
Flujo alterno 2		
Acción del Actor		**Respuesta del Sistema**
1		El sistema cierra la interfaz correspondiente. Regresa al paso 3 del **Flujo normal de eventos.**
Flujo alterno 3		
Acción del Actor		**Respuesta del Sistema**
1		El sistema muestra un mensaje de error indicando

		que hay datos incorrectos o campos obligatorios vacíos.
2	El organizador modifica o introduce los datos especificados por el sistema como incorrecto y se sigue el **Flujo normal de eventos**.	

Flujo alterno 4		
Acción del Actor		**Respuesta del Sistema**
1		El sistema muestra una interfaz en la cual se puede seleccionar o crear la entidad organizadora que debe ser solo una entidad organizadora por evento.
2	El organizador selecciona o crea una nueva entidad organizadora para el evento.	
3		El sistema guarda la entidad como organizadora.
	Regresa a la interfaz principal en la que se termina de agregar un evento.	

Prototipo de Interfaz



Flujo Alterno 5		
Acción del Actor		**Respuesta del Sistema**
1		El sistema muestra una interfaz preparada para crear una nueva entidad siendo esta la entidad organizadora del evento.
2	El organizador introduce los datos y selecciona aceptar.	
		El sistema adiciona una nueva entidad lista para terminar de crear el evento.

Prototipo de Interfaz

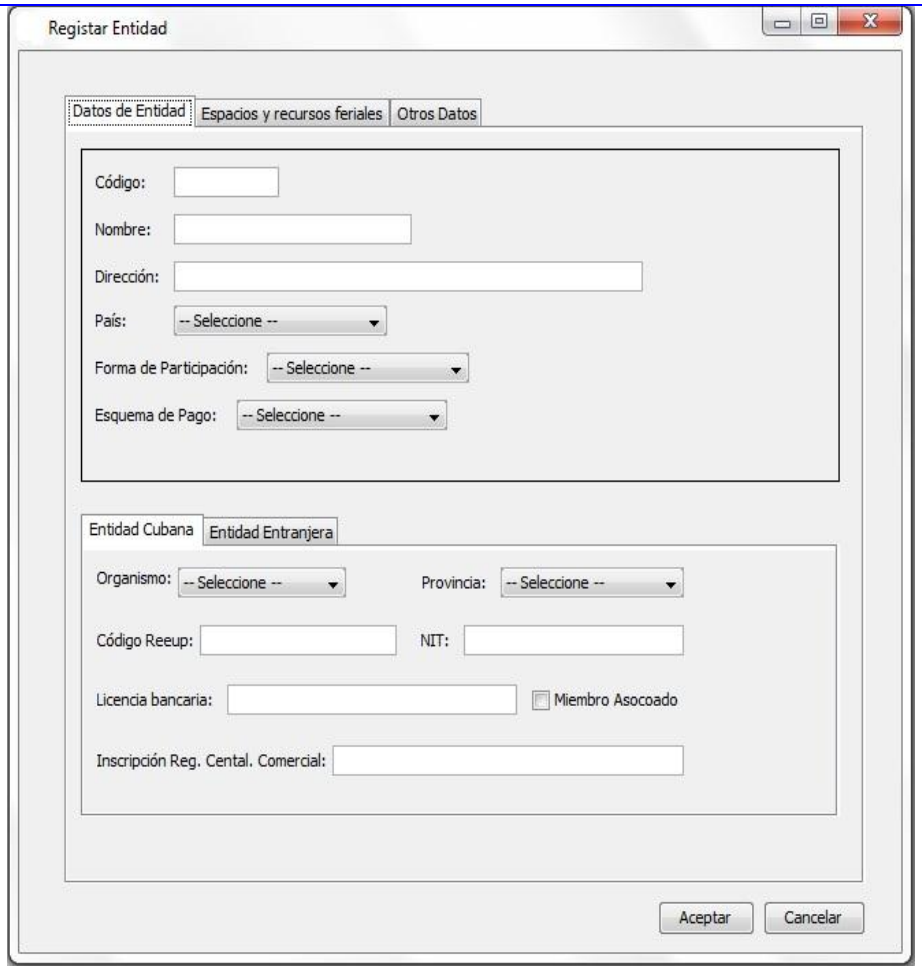


Tabla 5. Descripción Textual del Caso de Uso: Gestionar evento.

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 7 se muestra el diagrama correspondiente al caso de uso número 1 descrito en el capítulo dos. Los restantes diagramas podrán ser consultados en el Anexo 3.

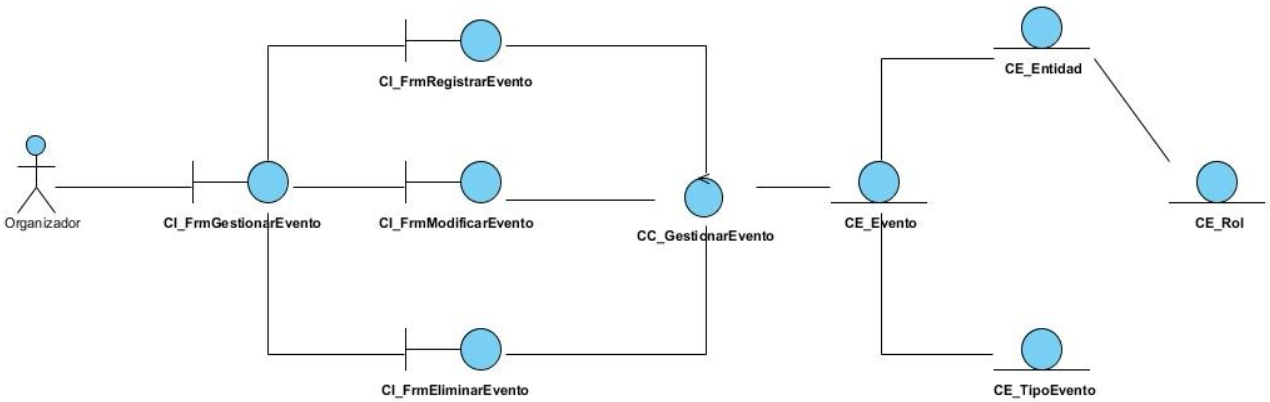


Figura 7. Diagrama de clases del análisis. CU Gestionar evento.

2.9. Diagrama de secuencia

Los diagramas de interacción muestran las interacciones entre objetos mediante transferencia de mensajes entre objetos o subsistemas, por lo que son empleados para modelar aspectos dinámicos del sistema. Los diagramas de colaboración y de secuencia son dos tipos de diagramas de interacción que son semánticamente equivalentes pero sin embargo los diagramas de colaboración destacan el orden estructural de los objetos que interactúan y los de secuencia destacan el orden temporal de los mensajes. Para la realización de los casos de uso del diseño es más factible el empleo de los diagramas de secuencia ya que representan con más claridad el flujo de las acciones que debe realizar el sistema. A continuación en la figura 8 se muestra el diagrama de secuencia cuyo escenario es el de registrar evento del caso de uso gestionar evento, los demás diagramas de secuencia generados por cada escenario se encuentran en el Anexo 4.

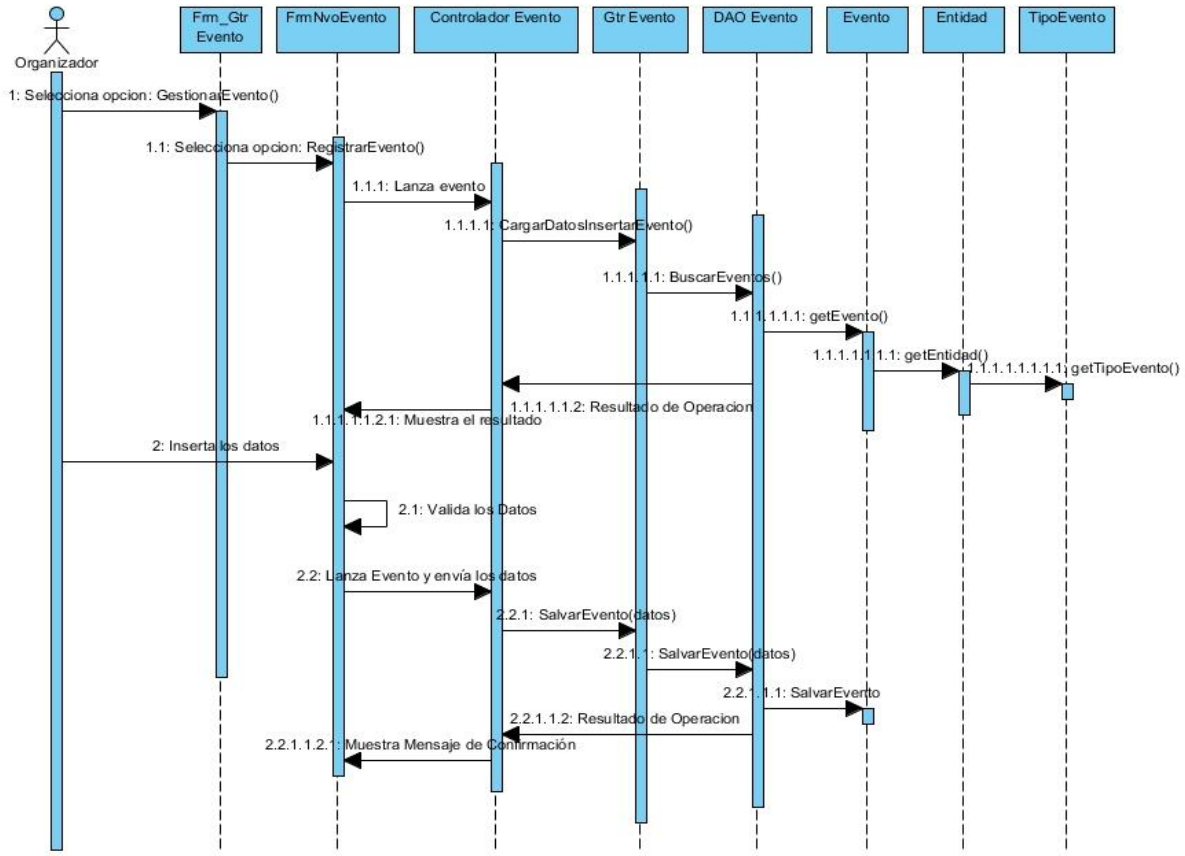


Figura 8. Diagrama de secuencia. Sección registrar evento.

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 evento descrito en este capítulo. Los restantes diagramas así como las descripciones de las clases que conforman el diagrama pueden ser consultados en el Anexo 5.

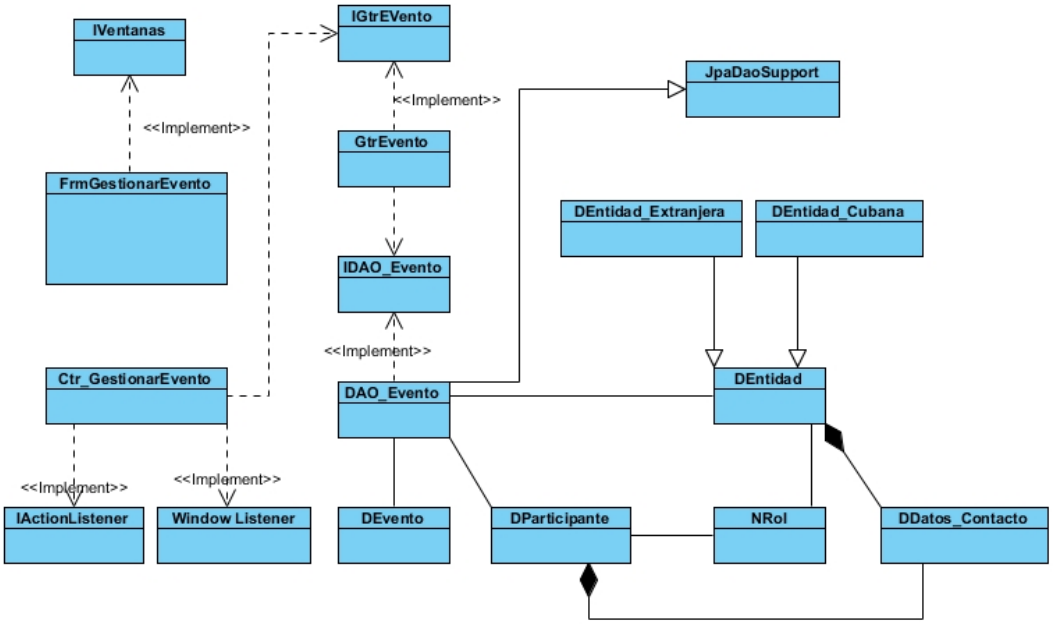


Figura 9. Diagrama de clases de diseño CU Gestionar evento.

2.10.1. Descripción de las clases

A través del diagrama de clases persistentes se puede representar en forma de clases toda aquella información que debe persistir como información duradera dentro de la base de datos. A continuación se muestra la representación gráfica de dicho diagrama.

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. El único nomenclador existente en el diagrama de clases persistentes es el Rol (NRol) puede ser observados en la Figura 10.

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

- DEvento tiene como propósito modelar el objeto evento, el cual se asocia a una entidad, a un participante y a catálogo.
- DEntidad modela el objeto entidad, se asocia a la clase rol, evento, a los datos de contacto y de ella heredan las clases entidad extranjera y entidad cubana. Esta entidad juega varios roles dentro del negocio en

este caso particular puede comportarse como un entidad que participa en el evento o como una entidad organizadora del evento.

- DParticipante modela el objeto participante y se asocia con las clases datos de contacto y evento, además de que el participante puede desempeñar el rol de organizador del evento o simplemente participante del evento.
- DDatosContacto: modela el objeto datos de contacto del cual hacen uso las clases participante y entidad.
- DCatalogo: modela el objeto catalogo el cual se asocia a la clase evento. Este se genera cuando se crea un evento.

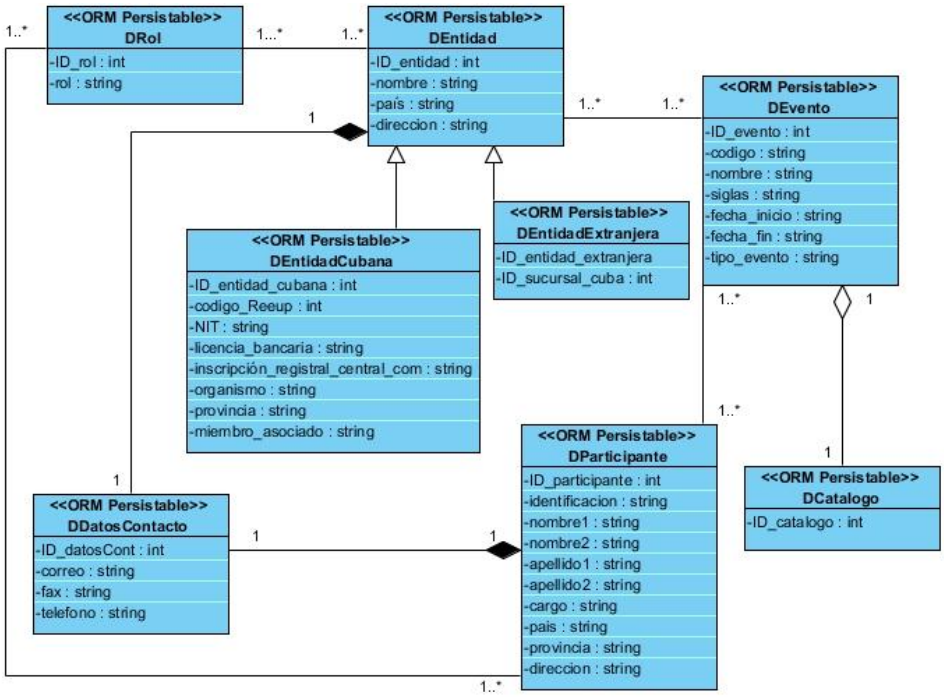


Figura 10. Diagrama de Clases persistentes

Clase controlador de eventos

Todas las clases que comienzan por **Ctr** 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: CtrEventoEvento, CtrEventosCatalogo, CtrEventosParticipante, CtrEventosEntidad,

CtrEventosDatosContacto. En la Figura 11 puede observarse el controlador correspondiente a la gestión de eventos.

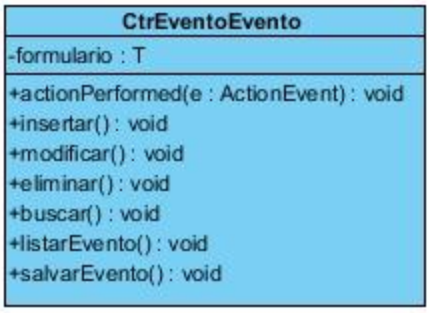


Figura 11. Controlador de eventos.

Clases de gestión

Las clases de gestión comienzan por Gtr, las mismas se llaman GtrEvento, GtrCatalogo, GtrEntidad, GtrParticipante, GtrDatosContacto y en la Figura 12 se muestra dicha clase.

Figura 12. Gestionar evento.

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.

2.11.1. Diagrama Entidad-Relación de la base de datos

A través de este modelo se puede establecer de una manera más específica las relaciones entre las clases que se han definido en el diagrama de clases

persistentes. A continuación en la figura 13 se muestra la representación gráfica de dicho modelo.

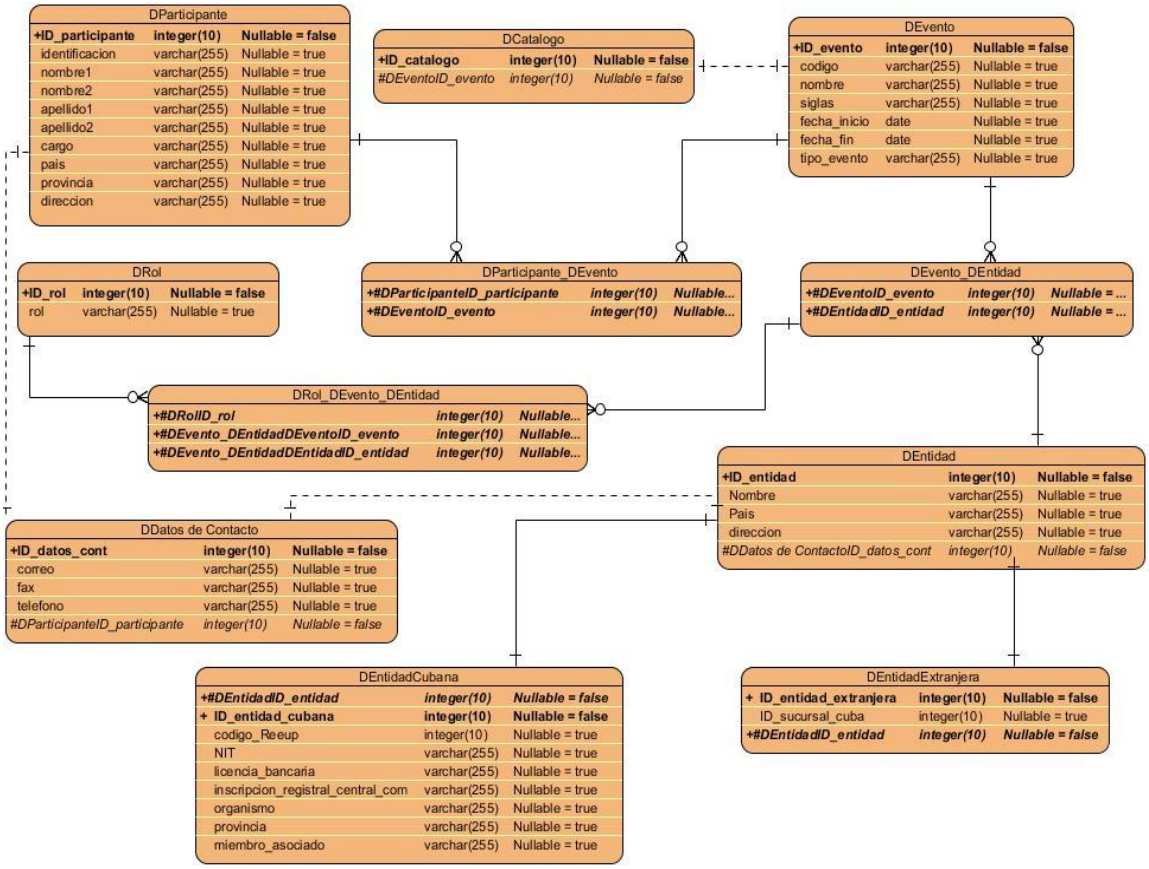


Figura. 13. Diagrama Entidad Relación.

2.12. Conclusiones del Capítulo

Al identificar los requerimientos funcionales y no funcionales, los actores del sistema, el modelo de casos de uso y sus descripciones textuales, se puede afirmar que la metodología seleccionada y sus herramientas han permitido la generación de los artefactos necesarios comprendidos en las fases de Modelado y Elaboración, obteniéndose una idea precisa de lo que en realidad se desea implementar, para lo cual se cumple el objetivo definido desde el principio.

Capítulo 3
[Implementación y prueba]

3.1. Introducción

En este capítulo se aborda el flujo de trabajo de implementación, donde se describen sus principales artefactos generados por el rol de implementador, destacando el modelo de implementación que incluye componentes, subsistemas de implementación y diagramas de componentes. Además en este capítulo se describen las pruebas del software que constituyen un elemento crítico para la garantía de la calidad del software. Con el diseño de pruebas se descubren diferentes clases de errores y se asegura que los defectos encontrados sean corregidos.

3.2. Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Se utiliza para modelar la vista estática de un sistema. Muestra la organización y dependencias lógicas entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, binarios o ejecutables. El diagrama que se visualiza en la Figura 14 muestra los componentes que conforman el sistema, en el mismo se pueden observar los componentes genéricos que contienen otro número de paquetes que serán mencionados a continuación.

- El componente **base** está formado por los recursos, las excepciones, mensajería, seguridad e interfaces que son implementadas por los útiles. El componente base se encarga de dar respuestas e intervenir en todas las funciones que se realicen en la aplicación.
- Dentro de **recursos** se encuentran los XML con la configuración básica del framework Spring.
- El componente **excepciones** agrupa las excepciones que serán

manejadas en el sistema, que pueden ser: de negocio BOException, acceso a datos DAOException, o de aplicación ApplicationException, con el objetivo de que las excepciones sean controladas.

- Dentro de **mensajería** contiene todos los mensajes que se mostraran en la aplicación.
- Dentro de **seguridad** almacena todas las configuraciones para proteger la conexión a la base de datos y además las funcionalidades necesarias para hacer uso del algoritmo de encriptación MD5.
- Dentro de **útiles** puede verse como la fachada del componente base 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, así como una librería **jar** que otorga un nivel de compresión y reduce la carga administrativa al distribuir clases en el lenguaje ya que permite agrupar todas las clases diseñadas en el lenguaje Java.
- El **controlador** está compuesto por los controladores, los gestores (clases que tienen toda la información referente al negocio) con las interfaces que implementan y las entidades de negocio. Los **controladores** están encargados de la captura de cada uno de los eventos que se originan en la vista para luego darle respuesta, en este paquete se encuentra el controlador frontal que se encarga de iniciar la aplicación.
- Los **gestores** contienen 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.
- Las 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.
- El componente **DAOs** recoge las clases que garantizan el acceso a datos utilizando el EntityManager ya que es la interfaz principal de JPA utilizada para la persistencia de las aplicaciones.

- El Mapeo de entidades almacena las entidades persistentes.

Se utilizan las librerías de EclipseLink pues contribuye a la persistencia de los datos, además de que las notaciones son entendibles por el mapeo relacional de objetos de Spring. Ya que se necesita un lenguaje entendible que sea reconocido por Spring y por el JPA.

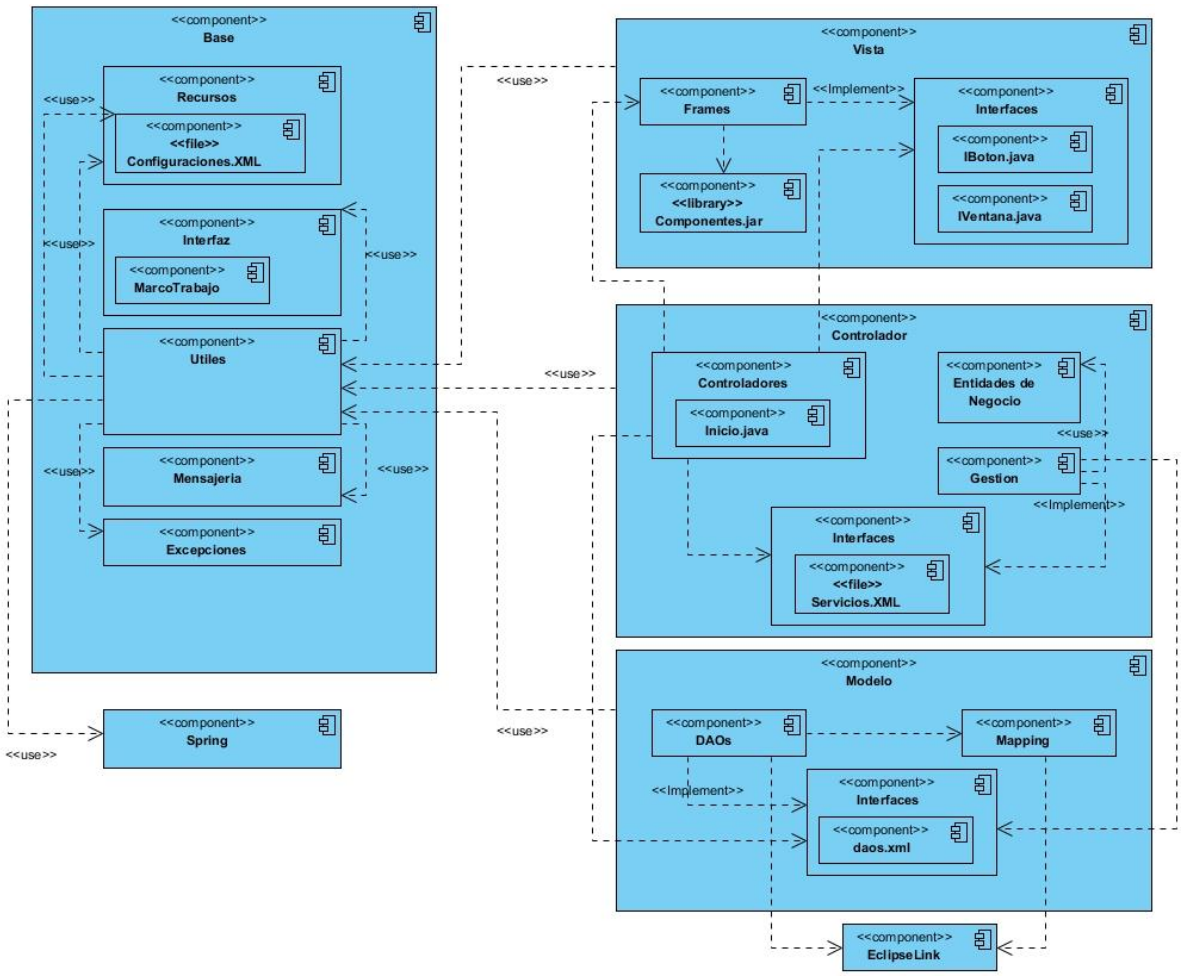


Figura 14. Diagrama de componentes.

3.3. Diagrama de despliegue

El diagrama de Despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Es un grafo de nodos unidos por conexiones de comunicación donde cada nodo puede contener instancias de componentes. En general un nodo puede ser una unidad de computación de algún tipo, desde un sensor hasta un mainframe y las instancias de componentes de software pueden estar unidas por relaciones de dependencia. (32)

El diagrama de despliegue para esta aplicación consta de una estación de trabajo (PC_cliente) el cual se conecta mediante los protocolos TCP/IP al servidor de base de datos (donde se almacenan los datos de los procesos de inscripción, acreditación y creación de catálogos). Para imprimir los catálogos generados se tendrá una impresora conectada a la estación de trabajo por USB. Con esto se garantiza el buen funcionamiento de la aplicación.

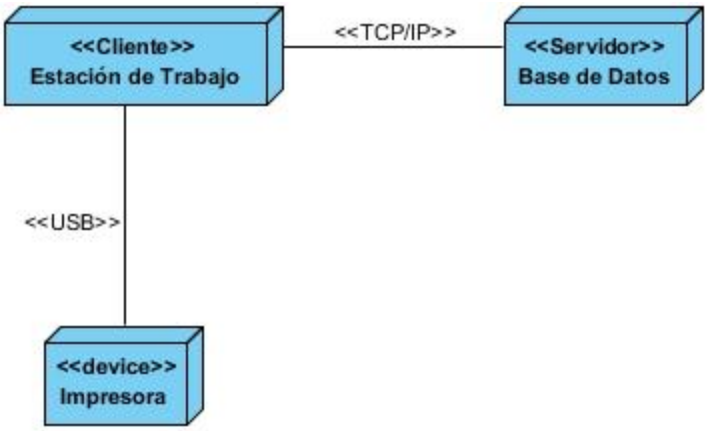


Figura 15. Diagrama de Despliegue.

3.4. Pruebas de software

Las pruebas del software constituyen un elemento crítico para la garantía de la calidad del software. Además de que se asegura de que la aplicación satisfaga los requisitos del cliente. Con el diseño de pruebas se descubren diferentes clases de errores y se asegura que los defectos encontrados sean corregidos. En este capítulo se abordará sobre métodos de pruebas aplicados al sistema.

Pruebas

La prueba es el proceso de ejecución de un programa con la intención de descubrir un error. Las mismas son un elemento crítico para la garantía de calidad del software y representan una visión final de las especificaciones, del diseño y de la codificación. Las pruebas comienzan desde la fase de inicio del software hasta la fase de construcción, siendo esta última fase donde tiene mayor volumen el flujo de trabajo de prueba. (15) La etapa de pruebas implica:

- Verificar la interacción de componentes.

- Comprobar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos normales encontrados se han corregidos antes de entregar el software al cliente.
- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

Las pruebas se enfocan sobre la lógica interna del software y las funciones externas para lo cual se definen los siguientes métodos:

3.4.1. Pruebas de caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal o pruebas unitarias, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales;
- Ejerciten las estructuras internas de datos para asegurar su validez. (15)

Uno de los métodos o técnicas de prueba unitarias, es la **prueba del camino básico**. Esta técnica permite obtener una medida de la complejidad de un procedimiento o algoritmo y un conjunto básico de caminos de ejecución de este, los cuales luego se utilizan para obtener los casos de prueba. Esta técnica asegura que durante la prueba se ejecute al menos una vez cada sentencia del código que se está probado (4). Esta será la técnica utilizada para desarrollar los casos de pruebas unitarias a la herramienta desarrollada.

De igual manera existen varias métricas de software para realizar pruebas unitarias, entre estas se encuentra la **complejidad ciclomática**, la cual será

utilizada junto a la técnica explicada anteriormente. Esta métrica proporciona una medición cuantitativa de la complejidad lógica de un procedimiento. La complejidad ciclomática cuando se utiliza en el contexto del método de prueba del camino básico, el valor que se calcula como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de casos de prueba que se deben realizar para asegurar que cada sentencia de código se ejecuta al menos una vez.(4)

Para realizar las pruebas unitarias o de caja blanca al código de la herramienta se seleccionaron varios algoritmos que son los fundamentales y los que más comunes se encuentran. A continuación se explica todo el procedimiento de obtener los casos de prueba y poner a pruebas estos a través de un ejemplo, utilizando la técnica prueba del camino básico junto con la métrica complejidad ciclomática. Para esto se hará uso de una notación para representación de flujo de control, denominada grafo de flujo, en la figura se muestra como se utiliza.

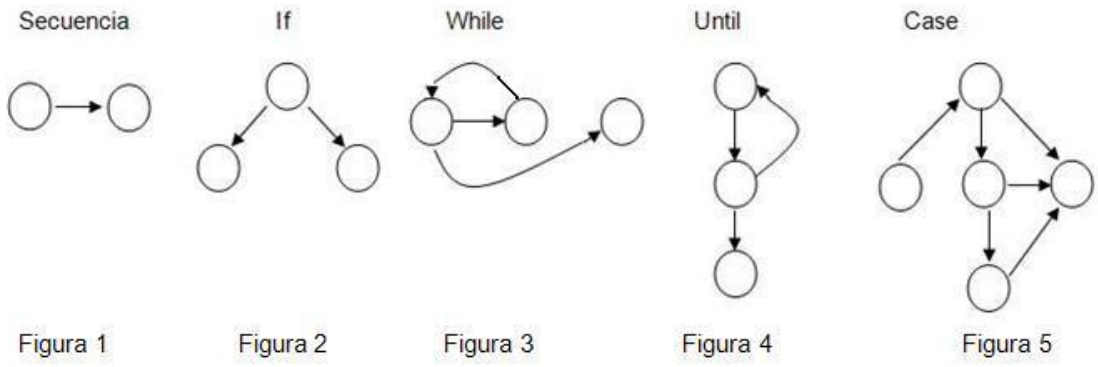


Figura 16. Construcciones estructurales en forma de grafo de flujo

El algoritmo seleccionado lleva como nombre **modificar** y tiene como finalidad modificar un evento seleccionado. En caso de que ya exista un evento con las mismas características que fueron insertadas, se lanza un error describiendo lo sucedido. De lo contrario es modificado el evento con los valores insertados.

```
private void modificar() {
    try { #1
        IGtrEvento gtr = (IGtrEvento) Base.getContexto().getBean("IGtrEvento");#2
        List<Ntipoevento> tiposEventos = gtr.tiposEventos(); #3
        Integer idEvento = (Integer) formulario.capturarObjet #4
        Devento evento = gtr.obtenerPorId(idEvento); #5
        EEntidad e = gtr.tmpEntidad(evento.getDentidadroleventoList().get(0).getDentidad());#6
        FrmNvoEvento modificar = new FrmNvoEvento(evento, tiposEventos, e);#7
        modificar.setControladorAction(this); #8
        modificar.setControladorVentana(this); #9
        modificar.iniciar(); #10


    } catch (Exception ex) { #11
        Logger.getLogger(CtrEventosEvento.class.getName()).log(Level.SEVERE, null, ex); #12
    }
}
```

Figura 17. Prueba unitaria método modificar.

Para obtener los casos de prueba a partir de este algoritmo se debe construir inicialmente el grafo de flujo correspondiente al código.

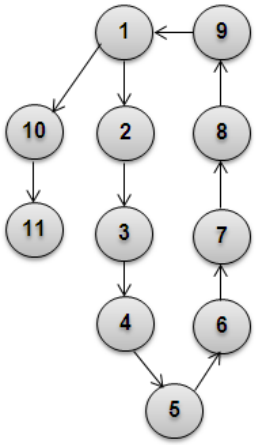


Figura 18. Prueba unitaria grafo de flujo.

Luego se determina la complejidad ciclomática V(G) del grafo resultante G, para esto se utilizan las siguientes fórmulas:

El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

$V(G) = A – N + 2$

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.

V(G) = P + 1

Donde P es el número de nodos predicados contenidos en el grafo de flujo G.

Del grafo obtenido se tiene que:

Número de regiones del grafo de flujo es 2. V(G) = 2	A = 11 N = 11 V(G) = A – N + 2 V(G) = 11 – 11 + 2 V(G) = 2	P = 1 V(G) = P + 1 V(G) = 1 + 1 V(G) = 2

El número de caminos independientes de la estructura del programa es igual a 2 por lo que se obtiene el conjunto básico.

Camino 1: 1-2-3-4-5-6-7-8-9
Camino 2: 1-10-11

Luego se definen 2 casos de prueba para el código del algoritmo. Estos se realizaron en JUnit el cual posibilita realizar pruebas unitarias en aplicaciones Java. JUnit brinda un conjunto de librerías que se integran fácilmente al IDE de desarrollo seleccionado: NetBeans. Permite la realización de las pruebas a los métodos de las clases implementadas en todas las capas. Para ello se definieron los siguientes pasos:

- Crear un caso de prueba por cada clase implementada.
- Configurar en el caso de prueba, los ficheros que permiten comunicar las clases de las capas de presentación y lógica de negocio.
- Definir los métodos a probar dentro de cada caso de prueba, incluyendo los parámetros de entrada.
- Realizar pruebas a los métodos.

Es importante destacar que aunque la implementación supere todas las pruebas, no necesariamente implica que todo este correcto; sólo infiere que funciona conforme a los casos de prueba que se han diseñado o para las pruebas realizadas. Los desarrolladores de la capa de persistencia al ser los

encargados de manejar directamente todos los componentes para el acceso a datos, deberían cerciorarse accediendo al gestor de base de datos y verificar visualmente si la operación se realizó satisfactoriamente. En caso contrario, lo recomendable sería revisar la prueba, la implementación del método, y mucho más importante aún, los elementos y atributos de los archivos de mapeo correspondientes a las entidades involucradas en la prueba.

La siguiente tabla muestra los aspectos que se tuvieron en cuenta en la realización de las pruebas a los métodos para la persistencia de objetos. Para ello se le realizó la prueba al método **modificar** del caso de uso gestionar evento.

Caso de Prueba	test_Gtr_evento	
Método	modificar	
Condiciones	Debe haber seleccionar un evento.	
Objetivo de la prueba	**Resultado esperado**	**Resultado de la prueba**
Salvar los valores que sean modificados en el evento seleccionado.	Si los valores insertados fueron correctos se espera que el evento modificado con todos los datos que hayan sido cambiados se guarden en la base de datos persistentes.	Se guardaron los datos correctamente y fue modificado el evento.

Tabla 6. Aspectos para realizar las pruebas a los métodos implementados.

3.4.2. Pruebas de caja negra

Pruebas funcionales que aplican el método de caja negra se llevan a cabo sobre la interfaz de usuario y se centran principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Examina aspectos del modelo, principalmente del sistema, sin tener en cuenta la estructura interna del software.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

- Funciones incorrectas o ausentes.

- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

En la Tabla 7 puede ser consultado el caso de prueba correspondiente al caso de uso gestionar evento, las secciones y los restantes casos de prueba pueden ser consultados en el Anexo 6.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo central**
Sección 1: Registrar evento	EC 1.1: Registrar un evento correctamente.	Se registra correctamente un evento.	1. Seleccionar opción “Gestionar Evento” en el menú principal. 2. Seleccionar opción “Nuevo”.
	EC 1.2: Registrar un evento con campos vacíos o erróneos.	No se registra el evento	1. Seleccionar opción “ Gestionar Evento” en el menú principal. 2. Seleccionar opción “Nuevo”.
	EC 1.3: Cancelar la operación	No se registra el evento	1. Seleccionar opción “Gestionar Evento” en el menú principal. 2. Seleccionar opción “Nuevo”.
Sección 2: Modificar evento	EC 1.1: Modificar un evento correctamente.	Se modifica correctamente un evento.	1. Seleccionar opción “Gestionar Evento” en el menú principal. 2. Seleccionar un evento y la opción “Modificar”.
	EC 1.2: Modificar un evento con campos vacíos o erróneos.	No se modifica el evento	1. Seleccionar opción “Gestionar Evento” en el menú principal. 2. Seleccionar un evento y la opción “Modificar”.
	EC 1.3: Cancelar la operación	No se modifica el evento	1. Seleccionar opción “ Gestionar Evento” en el menú principal. 2. Seleccionar un evento y la opción “Modificar”.
Sección 3: Eliminar evento	EC 1.1: Eliminar un evento correctamente.	Se elimina correctamente un evento.	1. Seleccionar opción “Gestionar Evento” en el menú principal. 2. Seleccionar un evento y la opción “Eliminar”.
	EC 1.2: Eliminar un	No se elimina el evento	1. Seleccionar opción “ Gestionar Evento”

	evento sin una previa selección del mismo.		en el menú principal. 2. Seleccionar opción “Eliminar”.

Tabla 7. Secciones del caso de prueba.

Elemento	No.	No conformidad	Aspecto correspondiente	Etapa de detección	Significativa	No Significativa	Recomendación	Estado NC	Resp. equipo de desarrollo
Interfaz Gestionar evento	1	El campo de búsqueda **código** le falta la tilde.	Pruebas a la interfaz de la aplicación.	1ra Etapa	X		Rectificar dicha palabra.	Resuelta	
Interfaz Gestionar evento	2	Los datos que muestra la búsqueda siempre son los mismos.	Pruebas a la interfaz de la aplicación.	1ra Etapa	X		Rectificar	Resuelta	
Interfaz Gestionar evento	3	La interfaz consultar evento no se muestra.	Pruebas a la interfaz de la aplicación.	1ra Etapa	X		Rectificar	Resuelta	
Interfaz Gestionar evento	4	El botón Eliminar evento, no elimina el evento seleccionado	Pruebas a la interfaz de la aplicación.	1ra Etapa	X		Rectificar	Resuelta	
Interfaz Gestionar evento	5	La interfaz Adicionar Entidad Organizadora no muestra todos los datos requeridos	Pruebas a la interfaz de la aplicación.	1ra Etapa	X		Rectificar	Resuelta	

Tabla 8. Registro de las No Conformidades encontradas.

3.5. Conclusiones del Capítulo

Se logró con este capítulo mostrar la organización y dependencias lógicas entre un conjunto de componentes de software evidenciado en el diagrama de componentes siendo este diagrama una exposición de los componentes de

código fuente, librerías, etc. que fueron usados para la implementación del sistema. Se describieron las pruebas del software y se realizaron casos de pruebas que guiaron la calidad del sistema con el único objetivo de eliminar la mayor cantidad de errores posibles. De forma general en este capítulo se ve la importancia de realizar un buen diseño de la aplicación y de asegurar la calidad del software.

Con la realización de la implementación se logra mostrar la organización y dependencias lógicas entre los componentes de software, esto a su vez se evidencia en el diagrama de componentes en el cual se logra una exposición de los componentes de código fuente, librerías, etc. que fueron usados para la implementación. Con la realización de los casos de pruebas de software se logra guiar la calidad del sistema con el único objetivo de eliminar la mayor cantidad de errores posibles. Se manifiesta la importancia de realizar un buen diseño de la aplicación y de asegurar la calidad del software.

Conclusiones Generales

- Como parte de la investigación se estudiaron otros sistemas tanto nacionales como internacionales, los cuales no respondían a todas las necesidades que se requerían, pero sirvieron de apoyo para la confección de este.
- Se analizaron los métodos, herramientas o materiales más óptimos para lograr un software con la calidad necesaria.
- En el actual trabajo, se puede afirmar que con su desarrollo se han cumplido los objetivos trazados, gestionando la información de las funcionalidades de inscripción, acreditación y creación de catálogos de los procesos Convocatoria y Contratación de las ferias y eventos de la Cámara de Comercio lo que permitirá realizar las actividades necesarias de forma ágil y en el tiempo previsto, y que puedan ser controladas en su transcurso o una vez terminadas.
- Se probaron las funcionalidades una vez concluida la implementación, para verificar el correcto funcionamiento de cada uno en los puntos más críticos.

Con la terminación de este software se da un paso más en la informatización de los distintos procesos para gestionar eventos, demostrando que muchas de las actividades pueden ser computarizadas, lo que traería consigo mayor calidad de gestión de la información y un mejor desarrollo del trabajo en menor tiempo, facilitándole el trabajo al hombre.

Recomendaciones

- Mejorar la aplicación con el objetivo de que no sirva simplemente para gestionar los eventos de la Cámara de Comercio de la Republica de Cuba sino que otros organismos, entidades o personas a nivel internacional y nacional puedan utilizarlo para gestionar sus eventos.
- Mantener actualizado el sistema para propiciar una mejor gestión de la información y de los servicios con el fin de brindar un mejor servicio a los usuarios.

Bibliografía

1. [En línea] http://www.gestionaeventos.com/es/41-agencias_organizacion_eventos/la_rioja/510-nueva_imagen_comunicacion_integral..

2. [En línea] http://www.programastop.com/perfecttableplan-software-de-gestion-de-eventos-bodas-banquetes-ceremonias..

3. [En línea] http://www.regonline.com.es/__articles/products/software~de~gesti%C3%B3n~de~eventos.

4. [En línea] http://ofi-eventos.programas-gratis.net/.

5. [En línea] http://es.wikipedia.org/wiki/NetBeans_IDE.

6. Visual Paradigm for UML. [En línea] 2010. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

7. **ROD JOHNNSON, J. H., ALEF ARENDSEN, THOMAS RISBERG, COLIN SAMPALEANU.** *Professional Java Development with the Spring Framework.* s.l. : Wiley Publishing Inc, 2005. ISBN 13:978-0-7645-7483-2.

8. **Rodríguez, Jorge.** Spring y el principio de Hollywood. [En línea] 10 de Enero de 2009. http://blog.continuum.cl/wp-content/uploads/2008/08/spring_y_el_principio_de_hollywood.pdf.

9. **Deepak Alur, John Crupi, Dan Malks.** Core J2EE Patterns Best Practices and Design Strategies. [En línea] http://books.google.com.cu/books?id=1dx34EMVyi8C&dq=Core+J2EE%E2%84%A2+Patterns:+Best+Practices+and+Design+Strategies&printsec=frontcover&source=bn&hl=es&ei=zKAUSvKADuGrtgeFirSZBA&sa=X&oi=book_result&ct=result&resnum=4#PPP1,M1.

10. **Tejada, L.** Metodología TenStep. *Proceso Unificado de Rational. Babo's blog.* [En línea] 2007. http://www.tenstep.es/01_Publico/00-10_ContenidoPrincipal/Castellano/0.0.4_Vision_general_de_la_metodologia.htm.

11. milestone.com. [En línea] http://www.milestone.com.mx/CursoModeladoNegociosBPMN.htm.

12. **DOMINICANO, F. C. L.** Gestor de Base de Datos: MySQL, PostgreSQL, SQLite. [En línea] 2009. http://www.eaprende.com/gestor-de-basededatos-mysql-postresql-sqlite.html#post.

13. Sitio Oficial JAVA. [En línea] http://java.sun.com/people/jag/OriginalJavaWhitepaper.pdf.

14. **Dorado, Andrés.** Patrones de diseño. [En línea] 26 de Febrero de 2009. http://www.slideshare.net/2008PA2Info3/tema-de-revision-2003.

15. **Larman, Craig.** *UML y Patrones.* Mexico : Editorial Mexicana, 1999. 970-17-0261-1..

16. **Piattini Velthuis, Mario G.** *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Madrid : s.n., 1996.

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

18. **Olguín Espinoza, José Martín.** *Análisis Orientado a Objetos.* 2004.

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

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

21. Tecnología y Synergix. [En línea] 31 de Julio de 2008. [Citado el: 7 de Febrero de 2010.] http://synergix.wordpress.com/2008/07/31/las-4-mas-1-vistas.

22. **otros, Bonaparte U y.** Universidad Tecnológica Nacional Facultad Regional Tucumán. *Universidad Tecnológica, Facultad Regional Tucumán.* [En línea] 2008. http://www.frt.utn.edu.ar/sistemas/paradigmas/poo.htm.

23. [En línea] http://www2.topografia.upm.es/pdi/m.manso/docencia/Informatica_plan92/Curso-2002-2003/poo.pdf.

24. [En línea] 24. http://www.slideshare.net/yamili7/paradigmas-de-programacion.

25. [En línea] http://t06nocturno.blogspot.com/2011/11/definicion-de-herramientas-case.html.

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

27. JEE. [En línea] 2009. http://cea-fidelmazo.blogspot.com/.

28. **holgazán, El.** JPA: Java Persistence API. [En línea] 30 de Agosto de 2007.

29. [En línea] http://www.soaagenda.com/journal/articulos/que-son-los-frameworks/.

30. **Vogel, Lars.** JPA 2.0 con EclipseLink. Tutorial. [En línea] 16 de Marzo de 2012.

http://www.vogella.com/articles/JavaPersistenceAPI/article.html.

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

32. **Vilas, Ana Fernandez.** *Diagrama de Despliegue.* 2001.

33. **Pressman.** *Ingeniería de software para un enfoque práctico.* 2005.

34. **Hernández Sampieri, R Tomo 1.** *Metodología de la Investigación.* La Habana : Félix Varela, 2003.

35. **Rodríguez, G., J. Gil y E, García.** *Metodología de la investigación cualitativa.* La Habana : Félix Varela, 2004.

36. **Reyes, T. (s.a.).** *Métodos cualitativos de investigación: los grupos focales y el estudio de caso.*

37. [En línea] http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

ANEXOS

ANEXO 1. Descripción de Requisitos.

Este documento contiene la descripción de los requisitos funcionales. (Consultar documento con el nombre “0129_Descripcion 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 clases del 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. Diagrama de secuencia o interacción.

Este documento contiene los diagramas de secuencia para cada uno de los escenarios de cada caso de uso. (Consultar documento con el nombre “Diagrama de Secuencia.doc” en la carpeta Artefactos).

ANEXO 5. 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 6. Diseño de caso 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).

Glosario de Términos

API: (Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

RUP: El Proceso Unificado Racional (Rational Unified Process en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

V&V: Verificación y validación.

Calidad: Medida de la cierta característica deseable

SGBD: Sistema Gestor de Base de Datos.