UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

MÓDULO DE ADMINISTRACIÓN Y REPORTES PARA EL SISTEMA DE GESTIÓN DE FERIAS Y EXPOSICIONES DE LA CÁMARA DE COMERCIO DE LA REPÚBLICA DE CUBA.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

AUTOR: Alejandro Alfonso Villegas

TUTOR: Ing. Lisett de Armas Hernández

CO - TUTOR: Ing. Arnel Abad Noa

LA HABANA, JUNIO DE 2013

"AÑO 55 DE LA REVOLUCIÓN"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autor: Alejandro Alfonso Villegas

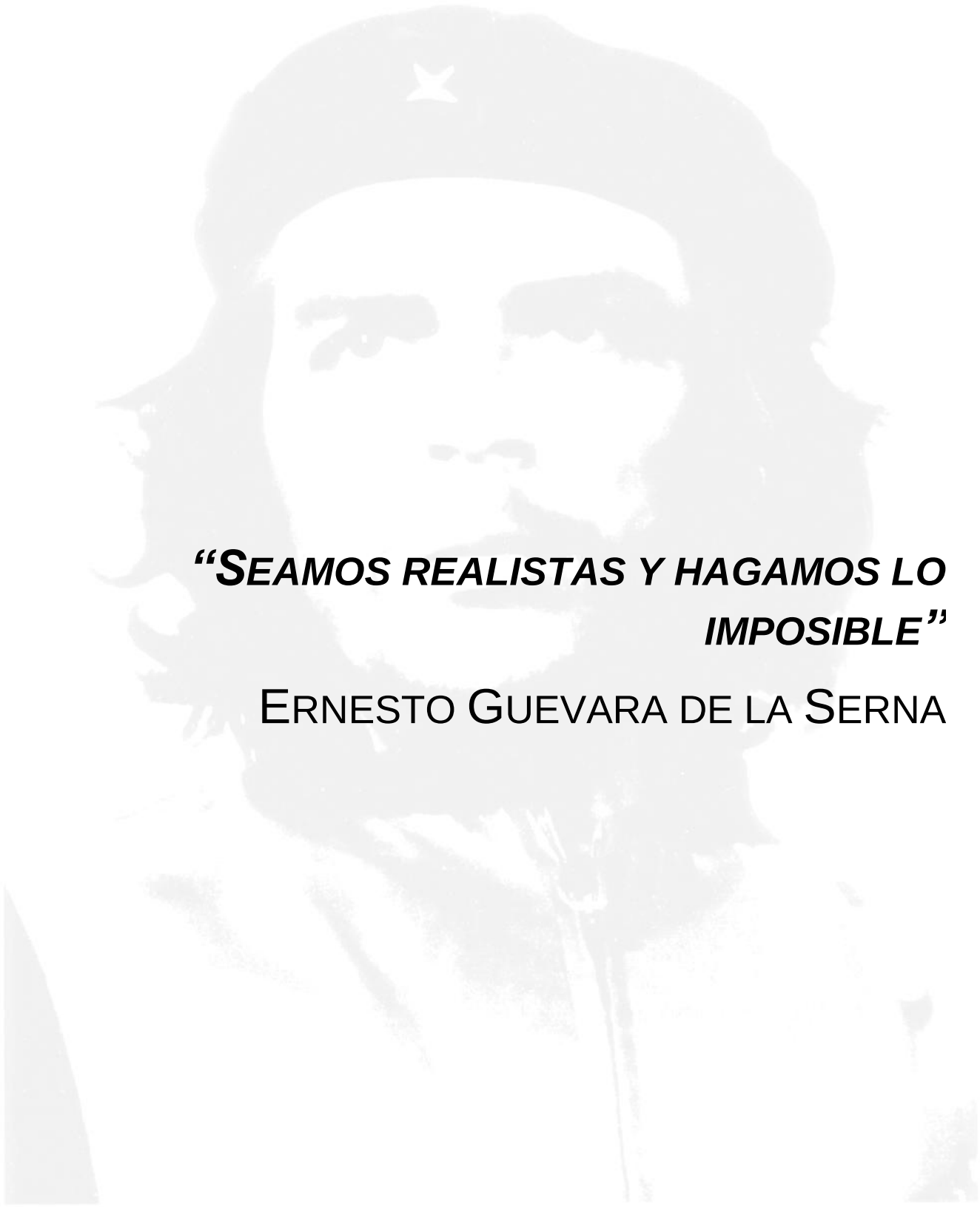
Tutor: Lisett de Armas Hernández

_____

Firma del Autor

_____

Firma del Tutor



"SEAMOS REALISTAS Y HAGAMOS LO IMPOSIBLE"

ERNESTO GUEVARA DE LA SERNA

DEDICATORIA

A mis abuelos: que hoy estarían orgullosos de mí.

A mi madre: por todo su cariño y paciencia al escucharme.

A mi padre: por sus buenos consejos y respeto a mis propias decisiones.

A mi hermano: por su confianza en mí.

AGRADECIMIENTOS

Deseo agradecer primeramente a mis padres. Ellos han sabido darme el apoyo necesario a través de estos cinco años, han sido mi razón de ser y mi soporte.

A mi hermano, a quien tengo la esperanza le sirva de patrón y ejemplo.

A Héctor, quien a pesar de haber estado lejos siempre estuvo presente.

A Yasmany, mi amigo de tantos años.

A Yaiset, mi amiga del alma.

A mis tutores, por la gran ayuda.

Y a todos los que, de una forma u otra, me aprecian y han batallado conmigo estos años de estudio.

RESUMEN

La presente investigación se desarrolla para mejorar la calidad de la información de las direcciones de ferias, exposiciones y relaciones internacionales de la Cámara de Comercio de la República de Cuba, institución que se encarga de generar estrategias para el desarrollo de sus empresas asociadas y ayuda a la reinserción de la economía nacional en el mercado extranjero.

Debido a las deficiencias con las que cuenta el Sistema Automatizado para Eventos, Ferias y Exposiciones (SAEFE), aplicación informática que utilizaban los funcionarios de la institución para gestionar toda la información de sus procesos, se decide desarrollar el Sistema de Gestión de Ferias y Exposiciones (SIGFE), dentro del cual es necesario insertar un módulo que se encargue de la configuración, la protección de los datos y la generación de reportes que ayuden a la toma de decisiones.

La solución propuesta resulta una aplicación de escritorio, desarrollada en un entorno orientado a tecnologías Java, específicamente la plataforma de desarrollo Java Enterprise Edition 6.0 (Java EE 6.0), con la utilización de la metodología de desarrollo de software Proceso Unificado de Rational, UML como lenguaje de modelado, entre otras herramientas y tecnologías que estuvieron bajo un riguroso proceso de selección. Se llevó a cabo cada flujo de trabajo ingenieril que propone la metodología, generándose los artefactos necesarios para finalmente lograr validar la solución, llegando a niveles satisfactorios de calidad.

Palabras clave: SAEFE, SIGFE, configuración, seguridad, reportes, RUP, Java, EJB.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS Y TABLAS

INTRODUCCIÓN

El mundo moderno se ha transformado de manera acelerada en los últimos 10 años, transformación que no solo ha cambiado la rutina de los hombres, la economía de las naciones o el poderío mismo de ellas, sino también la forma de hacer negocios y realizar transacciones, y en especial la forma en que se soportan tales actividades. En Cuba existe una conciencia por parte de las instituciones gubernamentales acerca de las facilidades que reporta el uso de las tecnologías de la información y las comunicaciones (TIC), que en su creciente desarrollo actual, logran propiciar una forma diferente y revolucionaria de trabajar en muchas empresas a nivel mundial y facilitan de forma considerable los procesos que en ellas se originan.

La Cámara de Comercio de la República de Cuba (en lo adelante Cámara de Comercio) representa una asociación de empresas vinculadas al comercio, la industria y los servicios, con reconocimiento ante los organismos del estado, que orienta y optimiza alternativas para el desarrollo de la actividad empresarial de sus entidades asociadas, siendo una herramienta para la reanimación de la economía cubana en las relaciones económicas internacionales (Cámara de Comercio de la República de Cuba., 2009). Esta institución potencia e intercambia información valiosa en torno a las posibilidades de negocios a escala mundial, promoviendo las ofertas exportables de productos y servicios, así como las oportunidades de inversión de la empresa cubana en beneficio de la economía nacional.

Entre los servicios que ofrece esta institución promueve la organización de ferias, exposiciones y otros eventos comerciales a entidades, tanto nacionales como extranjeras. Este proceso tiene un papel esencial para el país, al incrementar las posibilidades de intercambio comercial para Cuba en el extranjero y ampliar el diapasón económico-financiero en el ámbito nacional. Con 50 años de experiencia, la Dirección de Ferias y Exposiciones de la Cámara de Comercio ofrece una variada gama de servicios y sus eventos figuran un espacio para que las empresas asociadas exhiban sus productos al mercado cubano y mundial; además de ser considerado como punto de encuentro de socios comerciales y la cita obligada para hombres de negocios de todos los continentes.

Estas razones demandan priorizar el soporte de la organización, ejecución, cierre y seguimiento de este proceso y garantizar la calidad en la obtención y procesamiento de toda la información asociada.

Actualmente la Cámara de Comercio cuenta con un sistema incapaz de satisfacer en su totalidad las necesidades del proceso de gestión de las ferias y los eventos que se dan lugar en la institución. Desarrollado en lenguaje de programación Delphi y denominado SAEFE, esta es una aplicación creada con el fin de gestionar eventos pequeños y de poca complejidad. En la actualidad, la institución maneja eventos de gran importancia como la Feria Internacional de la Habana, que permiten abrir nuevas oportunidades para el desarrollo económico del país. SAEFE es un sistema con una capacidad de configuración y adaptación a nuevos escenarios prácticamente nula, siendo imposible el manejo dinámico de nomencladores, para lograr que el sistema se ajuste a nuevas reglas de negocio y funcione correctamente.

Así mismo no cuenta con los requerimientos mínimos de seguridad, para propiciar que la información sensible de clientes, productos y servicios quede debidamente protegida. Un ejemplo palpable que afecta esta variable es la gestión del acceso de los usuarios, que describe inconvenientes en la asignación de permisos a roles y, a su vez, de roles a los usuarios.

El proceso de gestionar ferias y eventos de este tipo demanda el uso de información sensible y la capacidad de generar informes, que puedan ser utilizados para trazar estrategias y tomar providencias. Sin embargo, la herramienta SAEFE describe inconvenientes en este sentido, pues muchos de los informes especificados presentan errores de funcionamiento, lo que hace en muchos casos imposible su visualización, y en otros impide mostrar en su totalidad los datos correspondientes. Esto provoca que no se logre el control eficiente de la información a través de los datos estadísticos que reflejan los informes, para lo cual fueron concebidos.

Lo anteriormente expuesto conduce a la formulación del siguiente **problema de la investigación:** La forma en la que está concebido el SAEFE en torno a la configuración, seguridad y salidas de informes, están afectando la funcionalidad del sistema.

Por tanto el presente trabajo centra su **objeto de estudio** en los procesos de gestión de ferias y exposiciones, delimitando como **campo de acción** la configuración, seguridad y reportes en sistemas de gestión de ferias y exposiciones.

Para dar solución al problema anterior se plantea como **objetivo general:** Desarrollar un módulo de Administración y Reportes para el Sistema de Gestión de Ferias y Exposiciones de la Cámara de Comercio, de manera tal que contribuya a la funcionalidad del sistema.

El objetivo general da lugar a los siguientes **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Realizar la captura de requerimientos del módulo.
- Diseñar e implementar los requerimientos obtenidos.
- Validar la solución propuesta.

La investigación defiende la siguiente **idea:** El desarrollo del Módulo de Administración y Reportes contribuye a la funcionalidad del Sistema de Ferias y Exposiciones de la Cámara de Comercio.

Con la intención de cumplir los objetivos específicos mencionados se propone un conjunto de **tareas de investigación**, las cuales son:

- Revisión bibliográfica para obtener antecedentes sobre el tema propuesto, como es la existencia de aplicaciones semejantes en otros países, los elementos que las componen y las herramientas más utilizadas.
- Selección y fundamentación de las herramientas a utilizar en el desarrollo de la aplicación.

- Elaboración de los artefactos necesarios según la metodología de desarrollo de software seleccionada.
- Implementación del módulo.
- Realización de pruebas al módulo.
- Validación de la solución de forma tal que se demuestre el cumplimiento del objetivo general.

En la investigación se utilizaron los siguientes **métodos científicos**:

Métodos teóricos:

- **Análisis histórico – lógico**: Este método se pone en práctica al realizar el análisis de los sistemas de gestión existentes en este contexto y utilizarlos como punto de referencia y comparación, además de comprobar de manera teórica su evolución en el tiempo.
- **Analítico – sintético**: Para resumir los componentes fundamentales que se relacionan con el proceso de desarrollo de software a partir de estilos, tendencias y documentos vinculados con el tema, expresando de manera resumida la posición del investigador.
- **Modelación:** Mediante este método científico, se crean abstracciones con la intensión de explicar la realidad. Mediante UML[1] se refleja la estructura, relaciones internas y características de la solución basándose en diagramas.

Métodos empíricos:

- **Entrevista:** Consiste en definir una reunión para intercambiar información entre una persona (el entrevistador) y otra (el entrevistado) u otras (entrevistados) (Hernández Sampieri, y otros, 2006). Este método se les realiza a los funcionarios de la Cámara de Comercio y se obtiene información relevante, para comprender cómo funcionan las tareas de administración y reportes para las ferias y exposiciones de la institución.

[1] **UML**: Unified Modeling Language (por sus siglas en inglés y en español Lenguaje Unificado de Modelado) Notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos.

- **Medición**: Permite obtener información numérica acerca de una propiedad o cualidad del objeto, donde se comparan magnitudes medibles y conocidas. Fue utilizado en el trabajo para evaluar la calidad del módulo, a través del uso de métricas y pruebas de calidad.

El presente trabajo de diploma consta de 3 capítulos, los cuales se describen brevemente a continuación:

El **capítulo 1** se refiere a la **fundamentación teórica** del trabajo, describiéndose todos los elementos de la teoría que sostienen el problema científico y los objetivos de la investigación. Como parte de este capítulo se realiza un estudio del estado del arte de los sistemas de gestión de ferias y exposiciones en otros países y también en Cuba. Se fundamentan las tecnologías, lenguajes de programación, herramientas, métricas y metodologías, para validar y guiar el proceso de desarrollo del software requerido.
Como parte del **capítulo 2** se realiza la **descripción de la propuesta de solución**, teniendo en cuenta la especificación de los requerimientos funcionales y no funcionales, diagramas de casos de uso del sistema, diagramas de clases del análisis y del diseño, así como los de secuencia correspondiente a cada uno de los escenarios de los casos de uso. También se tienen en consideración la descripción de la arquitectura y los patrones de diseño utilizados, con el objetivo de lograr un mejor entendimiento de la fase de diseño. Se muestra además como resultado el modelo de datos.
En el **capítulo 3** y último, denominado **Implementación y validación de la solución propuesta,** se exponen los diagramas de componentes y de despliegue. Además se realiza la validación de la solución desarrollada, mediante la realización de pruebas y métricas de calidad, además de la comprobación del cumplimiento del objetivo general de la investigación.

Para cada capítulo se ofrecen sus conclusiones parciales y al final del documento se exponen las conclusiones generales, y las recomendaciones.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

En este capítulo se exponen principios y conceptos que forman parte del proceso de fundamentación teórica de las tecnologías, metodologías, lenguajes de programación y herramientas que se utilizarán para dar cumplimiento a las tareas de desarrollo del módulo propuesto. Se hace además una descripción del estado del arte a escala mundial y nacional, de otros sistemas que implementan la gestión de ferias y exposiciones comerciales y se describen los conceptos fundamentales relacionados con el problema.

1.2 CONCEPTOS ASOCIADOS AL DOMINIO DEL PROBLEMA

1.2.1 CONFIGURACIÓN EN SISTEMAS DE INFORMACIÓN

Cuando se hace referencia a la configuración informática hablando de aquel grupo de datos e información que caracteriza a diferentes elementos de una computadora, como pueden ser programas, aplicaciones o elementos de hardware y software. La configuración es lo que hace que cada parte de la computadora cumpla una función específica porque es lo que eventualmente la define.

Es también importante señalar que las configuraciones pueden eventualmente llevar a errores. Si se cuenta con una configuración defectuosa, el programa o elemento funcionará de manera incorrecta.

En el ámbito comercial la configuración de sistemas informáticos se enfoca en preparar las aplicaciones de acuerdo a sus requerimientos de negocio, garantizando la gestión de nomencladores, cuentas de usuario, políticas que garanticen la seguridad y otra serie de factores que tengan que ver con el correcto funcionamiento y capacidad de adaptación de las aplicaciones.

1.2.2 SEGURIDAD EN SISTEMAS DE INFORMACIÓN

Un sistema es seguro si se puede confiar en él y se comporta de acuerdo a lo esperado (Garfinkel, y otros, 1996). La seguridad es un conjunto de soluciones técnicas, métodos,

planes, etc. con el objetivo de que la información que trata un sistema informático sea protegida. Es importante destacar que la seguridad supone un coste y que la seguridad absoluta es imposible. El término seguridad es bastante amplio, comprendiendo distintos aspectos:

- **Confidencialidad:** La información sólo puede ser accedida por aquel que esté autorizado.
- **Integridad:** La información no puede ser eliminada o modificada sin permiso.
- **Disponibilidad:** La información tiene que estar disponible siempre que sea necesario, evitando por tanto, ataques externos que puedan reducir esta disponibilidad o incluso una caída del servicio.

En términos de una empresa lograr la seguridad es primordial debido a que protege la información que la misma maneja. Cuanto más sensible es la información a procesar, mayor seguridad requieren los sistemas informáticos en las instituciones. En el ámbito de las actividades de intercambio comerciales, la protección de los datos es un pilar clave ya que actualmente muchas de las transacciones se realizan a través de las redes, donde un problema de seguridad acarrearía pérdidas incalculables. Además la información manipulada por entidades del sector suelen ser de importancia nacional y por lo tanto altamente sensible.

1.2.3 DEFINICIONES ESTADÍSTICAS

Según David Ruiz Muñoz la **estadística** en general se define como: “La ciencia que trata de la recopilación, organización, presentación, análisis e interpretación de datos numéricos con el fin de realizar una toma de decisiones más efectiva”(Ruiz Muñoz, David). La estadística atendiendo a diferentes factores se puede clasificar en dos grandes ramas:

- **Estadística Descriptiva o Deductiva:** Se encarga de analizar metódicamente los datos, simplificándolos y presentándolos en forma clara. Permite conformar cuadros, gráficos e índices bien calculados; suficientemente claros, como para disipar las dudas y la oscuridad de los datos masivos. Describe los datos que se analizan, sin hacer inferencias en cuanto a datos no incluidos en la muestra.

- **Estadística Inductiva o Inferencial:** Aporta soluciones basándose en los datos simplificados y analizados. Sobre la base de la muestra estudiada saca conclusiones, o sea, hace inferencia o inducción, en cuanto al universo o población, de donde se obtuvo dicha muestra.

La estadística, más allá que un solo conjunto de datos o valores presentados en tablas, gráficos o diagramas es una ciencia importante, que puede aplicarse a un número considerable de esferas de la sociedad, principalmente a la contabilidad, control de la calidad, análisis de resultados en la educación, cultura, salud y otros. Su ausencia llevaría a una desorganización general, dejando a los directivos sin información primordial para la toma de decisiones.

Un **reporte** es aquel documento que se utilizará cuando se quiere informar o dar noticia acerca de una determinada cuestión. Puede emplearse internamente dentro de una empresa, pero también puede ser usado en un establecimiento educativo, o cualquier rama donde se quiera comunicar un tema de interés. A través de este medio es posible reflejar y dar a conocer datos estadísticos importantes para el entorno de la empresa, asi como contribuir la toma correcta de decisiones en las mismas.

1.2.4 CONCEPTOS ASOCIADOS AL DOMINIO DEL NEGOCIO

De forma general un **nomenclador** es una forma de clasificar o agrupar diversas prácticas, actividades, tipo de especies, enfermedades, etc. En programación de sistemas, un nomenclador no suele ser otra que una tabla que permite, en principio, el ingreso de datos seleccionando un elemento, minimizando con ello la probabilidad de que el operador ingrese algún dato con errores.

Algunos de los nomencladores y reportes que serán manejados en el módulo propuesto se refieren a las siguientes definiciones, desde el punto de vista del negocio:

- **Local:** Recinto ferial donde se realiza la feria o exposición comercial.
- **Área:** Salón o espacio dentro de un local donde se agrupan un conjunto de stands.
- **Entidad:** Empresa nacional o extranjera.
- **Stand:** Espacio dentro de una feria o salón en el que una empresa expone y presenta sus productos o servicios.

- **Tipo de evento:** Clasificación que adquiere un evento (social, empresarial, institucional, comercial, entre otros).
- **Tipo de montaje:** Clasificación que adquiere un stand en relación a su forma de montaje. Esta puede ser: modular, libre diseño, empresas agrupadas, mixtos.

1.3 ESTUDIO SISTEMAS DE GESTIÓN DE FERIAS Y EXPOSICIONES

A nivel mundial existe un conjunto de sistemas capaces de gestionar la información resultante de un evento, tanto comercial como de otra índole y que actualmente son mayormente utilizados por empresas del sector. En su mayoría aplicaciones web, proveen recursos para la organización, planificación, seguimiento y toma de decisiones en la empresa. Entre los sistemas internacionales que se pueden mencionar se encuentran:

- **Amiando:** Este software ofrece a los organizadores de eventos una plataforma en Internet para planificar eventos, reuniones, conferencias, entre otros; permitiendo el ahorro de tiempo y dinero, gracias a la combinación de sus herramientas y servicios. En cuanto a la configuración permite al organizador de eventos crear y administrar su formulario de registro en línea, creación de diversas categorías y condiciones especiales a los participantes, confirmación y credenciales. Cuenta con la tecnología más novedosa el ámbito de la protección de datos: certificado SSL[2], desarrollo seguro del pago, seguridad en los pagos mediante tarjeta gracias a un certificado PCI DSS[3]. Además permite la creación de informes personalizados que se adecuen a las necesidades de los clientes (Amiando, 2013).
- **GestEvent:** Se trata de un sistema online destinado a la gestión y organización de eventos. GestEvent posee mecanismos de protección de los datos personales de los usuarios, a fin de evitar razonablemente su desviación, adulteración, pérdida, consulta o tratamiento no autorizado. Esta solución permite a los usuarios una configuración muy flexible para lograr cubrir todas sus necesidades, en torno a la gestión de recursos u otros

[2] **SSL**: Secure Sockets Layer (por sus siglas en inglés y en español Capa de conexión segura) es un protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

[3] **PCI DSS**: Payment Card Industry Data Security Standard (por sus siglas en inglés y en español Estándar de Seguridad de Datos para la Industria de Tarjeta de Pago) es un estándar guía para prevenir los fraudes que involucran tarjetas de pago débito y crédito.

elementos que puedan ser particulares en cada negocio, mientras que manejan una amplia generación de reportes con datos estadísticos confiables(GESTEVET, 2012).

- **RegOnline:** Es una solución flexible y accesible, que presenta las siguientes características: creación de sitios web de eventos a través de un sistema de control de contenidos propio que permite a interesados una fácil configuración y personalización; sistema automático de invitaciones por correo electrónico con estilo profesional; promoción de eventos a través de las redes sociales; informes tanto estándar como personalizados lo suficientemente flexibles como para permitir combinar y recopilar información en la manera que satisfaga las necesidades de un cliente. Esta herramienta posee estándar PCI nivel 1 con certificación de seguridad conforme a las normas más altas y eficaces de la industria de las transacciones, así como máximo cifrado de datos permitido por la ley (ActiveNetwork, 2012).

En nuestro país se utiliza el siguiente sistema para la gestión de las ferias y eventos comerciales desarrollados en la Cámara de Comercio:

- **SAEFE:** Esta herramienta presenta deficiencias en su configuración, que junto a la existencia de problemas en el manejo de la seguridad de la información, la convierten en una aplicación no funcional. En otro sentido, SAEFE manipula la generación de informes de una manera desorganizada y muchas veces con errores, ya que los datos no presentan la estructura correcta para ser oficialmente publicados a un usuario y en ocasiones no pueden ser visualizados debido a errores internos del sistema.

De manera general no existe un Módulo de Administración y Reportes que se encargue de gestionar y garantizar el correcto funcionamiento de la aplicación, por lo cual se ve afectada la funcionalidad del sistema.

1.3.1 CONCLUSIONES PARCIALES SOBRE LAS HERRAMIENTAS ANALIZADAS

Por las particularidades de la infraestructura tecnológica de la Cámara de Comercio no es inteligente utilizar ninguna de las aplicaciones internacionales mencionadas, exponiéndose las razones a continuación:

- Se trata de herramientas privativas o aplicaciones web cuyo uso representa un gasto económico importante.
- Presentan algunas funcionalidades que la Cámara de Comercio no utilizaría debido al entorno de trabajo en el que interactúa.
- La institución necesita que la aplicación seleccionada responda a requerimientos propios de sus funciones de negocio, con características mucho más particulares que las de las herramientas anteriormente presentadas.

La herramienta de factura nacional queda excluida por las deficiencias planteadas con anterioridad.

1.4 METODOLOGÍA DE DESARROLLO DE SOFTWARE

El desarrollo de software no es una labor fácil; las tareas que en él se dan lugar no pueden ser abordadas en desorden o a conveniencia propia de cada miembro del proyecto. Como resultado a este problema ha surgido una alternativa desde hace mucho: la metodología de desarrollo, que de forma sintetizada es el conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo software (Grupo Alarcos).

Una de las metodologías de desarrollo de software actuales es **Proceso Unificado de Rational (RUP).** Este es un proceso que proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales, con unos costos y calendario predecibles (Kruchten, 2000).

RUP se encarga de integrar todos los aspectos a tener en cuenta durante el ciclo de vida del software para abarcar todo tipos de proyectos, tanto grandes como pequeños, y presenta como características fundamentales: estar guiado/manejado por casos de uso, centrado en la arquitectura y ser iterativo e incremental.

Esta metodología define cuatro fases en cada uno de los ciclos, que constituyen la vida de un sistema (ver figura 1):

- **Inicio**: Se desarrolla una descripción del producto final y se presenta el análisis de negocio para el producto.
- **Elaboración**: Se especifican en su mayoría los casos de uso del producto y se define la arquitectura del sistema.
- **Construcción**: Durante esta fase se crea el producto, y la línea base de la arquitectura crece hasta convertirse en el sistema completo.
- **Transición**: En esta fase un reducido grupo de usuarios con experiencia prueban el producto e informan de defectos y deficiencias.

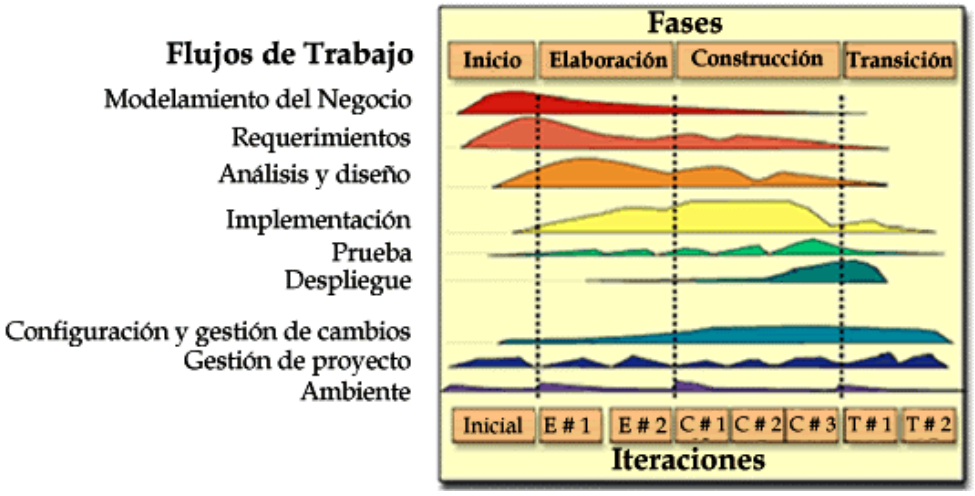


Figura 1: Fases, iteraciones y disciplinas de RUP.

En RUP la necesidad de contar con una secuencia de actividades realizadas por diferentes roles para definir el proceso, obliga al uso de los flujos de trabajo, que son la relación de actividades que producen resultados observables. Con un número total de 9 flujos, los mismos se encuentran agrupados en 6 para el desarrollo y 3 para el soporte.

1.4.1 FUNDAMENTACIÓN DE LA METODOLOGÍA SELECCIONADA

La selección de RUP como metodología de desarrollo, se basa en que una vez liberado el producto la Cámara de Comercio necesitará un control total sobre el mismo, para realizar labores de mantenimiento o cambio, a través de la información y documentación detallada que surja a lo largo de todo su proceso de desarrollo de software. Este rasgo define la

exclusión de metodologías que no pongan énfasis en la generación de una documentación exhaustiva y hace que RUP sea idónea.

1.5 ARQUITECTURA DE SOFTWARE

Al hablar de arquitectura de software, se hace alusión a la especificación de la estructura del sistema, entendida como la organización de componentes y relaciones entre ellos; los requerimientos que deben satisfacer el sistema y las restricciones a las que está sujeto, así como las propiedades no funcionales del sistema y su impacto sobre la calidad del mismo; las reglas y decisiones de diseño que gobiernan esta estructura y los argumentos que justifican las decisiones tomadas(Camacho, y otros, 2004).

A continuación se fundamentarán los diferentes estilos arquitectónicos presentes en la propuesta de solución:

1.5.1 ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. La característica central de la arquitectura cliente-servidor es la ubicación de las tareas del nivel de aplicación entre clientes y servidores. La figura 2 ilustra un caso general, donde tanto en el cliente como en el servidor, las plataformas y los sistemas operativos pueden ser diferentes.

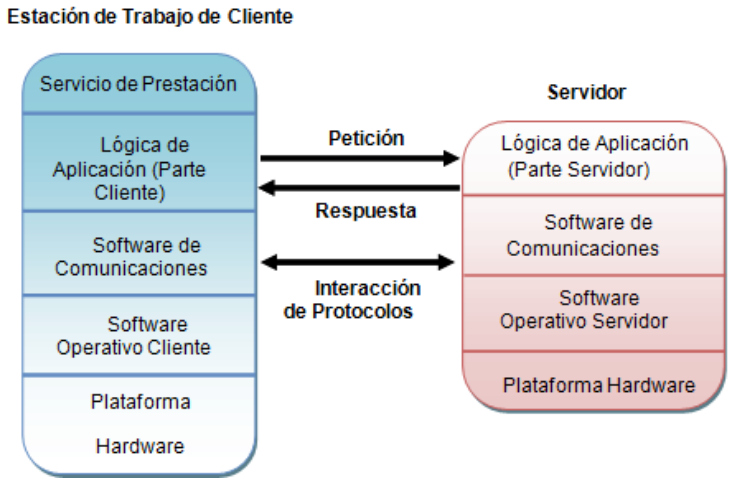


Figura 2: Arquitectura genérica cliente-servidor

Esta arquitectura proporciona las siguientes ventajas (Fadraga Artiles, y otros, 2012):

- Alta seguridad de los datos almacenados en un servidor, el cual debe ofrecer más control de la seguridad que los clientes.
- Acceso centralizado de los datos, como estos se encuentran centralizados en un solo servidor pueden ser accedidos y actualizados de mejor manera que en otros estilos arquitectónicos.
- Fácil mantenibilidad, este modelo se asegura de que se puedan mantener los sistemas sin afectar a los clientes.

1.5.2 ARQUITECTURA EN CAPAS

La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica, proveyendo una forma muy efectiva de separación de responsabilidades. El rol indica el modo y tipo de interacción con otras capas, y la responsabilidad indica la funcionalidad que está siendo desarrollada (Microsoft Corporation, 2009).

Los principales beneficios del estilo de arquitectura basado en capas son:

- **Abstracción**: Las capas permiten cambios que se realicen en un nivel abstracto.
- **Aislamiento**: La arquitectura de capas permite aislar los cambios en tecnologías a ciertas capas para reducir el impacto en el sistema total.
- **Rendimiento**: Distribuir las capas entre múltiples sistemas (físicos) puede incrementar la escalabilidad, la tolerancia a fallos y el rendimiento.
- **Mejoras en pruebas**. La capacidad de realizar pruebas se beneficia de tener interfaces bien definidas para cada capa así como de la habilidad para cambiar a diferentes implementaciones de las interfaces de cada capa.

1.5.3 ARQUITECTURA BASADA EN COMPONENTES

Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requerimientos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (Szyperski, 1998).

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación a objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Microsoft Corporation, 2009).

La arquitectura orientada al desarrollo de componentes brinda los siguientes beneficios (Oracle Corporation, 2013):

- Facilidades de mantenimiento, porque se puede reemplazar el componente con el desarrollo de nuevas versiones sin impactar al sistema en general, sino solo al componente en cuestión.
- Reducción de los costos del desarrollo, por ejemplo el uso de componentes de terceros minimiza el costo del desarrollo y de mantenimiento ya que los componentes están desarrollados y solo los mantienen sus creadores o proveedores.
- Aumenta el nivel reutilización ya que los componentes se desarrollan solo una vez y luego son utilizados donde se necesiten.

1.6 LENGUAJES UTILIZADOS

1.6.1 LENGUAJE UNIFICADO DE MODELADO (UML)

"**UML**" son las siglas de Unified Modeling Language (Lenguaje Unificado de Construcción de Modelos), notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos (Larman, 1999).

Se trata de un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas, unificando la experiencia pasada sobre técnicas de modelado e incorporando las mejores prácticas actuales en un acercamiento estándar.

UML es un lenguaje expresivo, claro y uniforme, que no garantiza el éxito de los proyectos pero sí mejora sustancialmente el desarrollo de los mismos, al permitir una nueva y fuerte integración entre las herramientas, los procesos y los dominios. Es independiente del proceso, aunque para utilizarlo óptimamente se debe usar en un proceso que sea dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental. Las herramientas pueden ofrecer generadores de código de UML para una gran variedad de lenguajes de programación, así como construir modelos por ingeniería inversa a partir de programas existentes.

1.6.2 LENGUAJES DE PROGRAMACIÓN: JAVA

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una máquina, particularmente una computadora. Consiste en un conjunto de reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos, respectivamente.

Java es un lenguaje de programación orientado a objetos ideado por Sun Microsystem. Es un lenguaje de propósito general por lo que con él se podría crear cualquier tipo de aplicación. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (Belmonte Fernández, 2005).

De acuerdo con el libro blanco de Java, este lenguaje de programación se encamina a ser "*un lenguaje sencillo, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutral, portátil, de gran rendimiento, multitarea y dinámico*" (Zukowski, 2003).

Como parte de las tendencias de lenguajes de programación actuales, Java presenta los siguientes beneficios:

- Es independiente de la plataforma de desarrollo.
- Existen dentro de su librería, clases gráficas que permiten crear objetos visuales comunes altamente configurables y con una arquitectura independiente de la plataforma.

- Java permite a los desarrolladores aprovechar la flexibilidad de la Programación Orientada a Objetos en el diseño de sus aplicaciones.
- El manejo de las bases de datos es uniforme, es decir, transparente y simple.
- El sistema de Java es seguro ya que tiene muchas funciones de seguridad integradas, que garantizan la seguridad del código que se está ejecutando.

1.6.2.1 FUNDAMENTACIÓN DEL LENGUAJE DE PROGRAMACIÓN SELECCIONADO

Java es el lenguaje apropiado para la implementación de la solución ya que es un lenguaje que se compila para correr en una máquina virtual (JVM) y esta máquina virtual corre en casi cualquier sistema operativo: Windows, Linux, UNIX, Solaris, hasta en dispositivos móviles. Para desarrollar en Java no es necesario comprar licencias de ningún tipo, es completamente gratuito y libre. Java restringe el uso de aspectos críticos del sistema para evitar la codificación de virus.

1.7 HERRAMIENTAS Y TECNOLOGÍAS

Actualmente existen muchas herramientas que brindan al implementador la comodidad para desarrollar software y productos. Para la realización, desarrollo y concepción de la propuesta de solución presentada se tuvo en cuenta la utilización de las siguientes herramientas y tecnologías:

1.7.1 HERRAMIENTAS CASE: VISUAL PARADIGM UML 8.0 ENTERPRISE EDITION (VP)

Las herramientas denominadas como CASE (por sus siglas en inglés, Computer Aided Software Engineering, en español Ingeniería de Software Asistida por Ordenador) permiten organizar y controlar el desarrollo de software, especialmente en proyectos grandes y complejos que involucran múltiples componentes de software y personas (Pressman, 2002).

Dentro de las herramientas CASE más usadas se encuentra Visual Paradigm UML 8.0 Enterprise Edition. Esta herramienta presenta facilidades para el trabajo colaborativo, la integración, el trabajo con modelos relacionales de bases de datos, modelado de

procesos de negocio a través de BPMN[4] así como el modelado de todos los artefactos propuestos por la metodología a través de UML, en una sola herramienta. Permite a los equipos de desarrollo el diseño, comunicación e implementación de sus requerimientos, maximizando y acelerando las contribuciones, tanto individuales como en equipo, facilitando a las organizaciones visualizar y diseñar diagramáticamente. Visual Paradigm es integrable además con los entornos de desarrollo más utilizados a nivel mundial, entre ellos NetBeans.

Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Este software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo (Booch, y otros, 1999). Entre sus características principales se encuentran:

- Multiplataforma.
- Ofrece un diseño centrado en casos de uso y enfocado al negocio.
- Posee capacidades de ingeniería directa e inversa.
- Puede integrarse a los principales entornos de desarrollo.
- Generación de bases de datos y conversiones de diagramas entidad-relación a tablas de bases de datos, además de mapeos de objetos y relaciones.

1.7.1.1 FUNDAMENTACIÓN DE LA HERRAMIENTA CASE SELECCIONADA.

Se selecciona Visual Paradigm Enterprise Edition debido a que es una herramienta de modelado multiplataforma que brinda una interfaz de usuario amigable y fácil de usar. VP proporciona al desarrollador un ambiente completamente funcional y permite ahorrar tiempo durante el proceso de desarrollo del software, principalmente durante la fase de construcción. De manera general se considera un estándar ampliamente utilizado en el mundo por las empresas, para el modelado en el proceso de desarrollo de software.

[4] **BPMN**: Business Process Modeling Notation (por sus siglas en inglés y en español 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.

Suministra facilidades en el trabajo colaborativo y la integración, soportando la confección de todos los artefactos que propone la metodología de software seleccionada.

1.7.2 PLATAFORMA DE DESARROLLO: JAVA EE 6

Java Platform Enterprise Edition o Java EE, es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java, con la cual el desarrollo de aplicaciones empresariales Java es fácil y rápido. El objetivo de la plataforma Java EE es proporcionar a los desarrolladores un potente conjunto de API (por sus siglas en ingles Application Programming Interface, en español Interfaz de programación de aplicaciones), mientras acorta el tiempo de desarrollo, reduce la complejidad de las aplicaciones y mejora el rendimiento de la aplicación (Oracle Corporation, 2013).

Java EE tiene varias especificaciones de API, tales como JDBC[5], RMI[6], e-mail, Java Message Service, Servicios Web, entre otros; y define cómo coordinarlos. También configura algunas especificaciones únicas para Java EE que incluye Enterprise JavaBeans, Servlets, Java Server Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una aplicación de empresa portable entre plataformas, escalable, e integrable con las tecnologías anteriores. Otros beneficios añadidos son, por ejemplo, que el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados, significando que los desarrolladores pueden concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

1.7.3 ENTORNO DE DESARROLLO INTEGRADO (IDE): NETBEANS IDE 7.2

Un Entorno Integrado de Desarrollo (IDE, Integrated Development Environment) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la

[5] **JDBC**: Java Database Connectivity, (por sus siglas en inglés) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java utilizando el dialecto SQL del modelo de base de datos que se utilice.

[6] **RMI**: Java Remote Method Invocation, (por sus siglas en inglés) es un mecanismo ofrecido por Java para invocar un método de manera remota.

edición orientada al lenguaje, la compilación o interpretación, la depuración, las medidas de rendimiento, etc., normalmente de forma modular (González Barahona, y otros, 2003-2007).

Uno de los IDE actuales más usados por las facilidades que reporta es NetBeans, considerándose libre y hecho principalmente para el lenguaje de programación Java, siendo un producto gratuito y sin restricciones de uso. Permite el rápido y fácil desarrollo de aplicaciones Java de escritorio, móviles y aplicaciones web, mientras que también proporciona una gran herramienta para PHP y C/ C++. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE[7], web, EJB[8] y aplicaciones móviles) (Oracle Corporation NetBeans, 2010).

A continuación, algunas ventajas de este IDE:

- Es un IDE multilenguaje y adaptable.
- Es software libre y gratuito.
- Intuitivo y fácil de utilizar.
- Se puede desarrollar todo tipo de aplicaciones.
- Es poderoso y extensible.
- Fácil integración con el servidor de aplicaciones Glassfish.

1.7.3.1 FUNDAMENTACIÓN DEL IDE SELECCIONADO.

Se decide utilizar NetBeans como IDE ya que cuenta con una curva de aprendizaje corta y una interfaz amigable e intuitiva, que mejora la usabilidad del software, es más fácil de navegar y posee un constructor de interfaz gráfica de usuario que viene integrado con la plataforma. Además NetBeans funciona en múltiples sistemas operativos, es gratis y libre, compatible para muchos lenguajes de programación además de Java. Este IDE es

[7] **J2SE**: Java Platform, Standard Edition o Java SE (conocido anteriormente hasta la versión 5.0 como Plataforma Java 2, Standard Edition o J2SE), es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java.

[8] **EJB**: Enterprise Java Beans (por sus siglas en inglés) son una de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE (ahora JEE 6.0) de Oracle Corporation (inicialmente desarrollado por Sun Microsystems).

ampliamente integrable con la plataforma Java EE 6, con todas sus tecnologías, y las demás herramientas seleccionadas para el desarrollo.

1.7.4 ENTERPRISE JAVA BEAN 3.0 (EJB)

Enterprise Java Beans (EJB) es una plataforma para la construcción portátil, reutilizable y escalable de aplicaciones de negocio, utilizando el lenguaje de programación Java. Desde sus inicios, EJB ha sido promocionado como un componente modelo o marco de referencia que le permite construir aplicaciones Java empresariales sin tener que reinventar los servicios, como transacciones, seguridad, persistencia automatizada, entre otros. EJB permite a los desarrolladores de aplicaciones centrarse en la lógica de construcción de negocio sin tener que gastar tiempo en la construcción de infraestructura de código (Reka, y otros, 2007).

Desde la perspectiva de un desarrollador, un EJB consiste en un fragmento de código Java que se ejecuta en un entorno especializado llamado contenedor EJB, que proporciona un número de servicios de componentes que serán de gran ayuda para la solución, evitando así la pérdida de tiempo desarrollando los mismos.

A continuación se muestran alguno de los servicios implementados por el contenedor EJB: manejo de transacciones, seguridad, concurrencia, servicios de red, gestión de recursos, persistencia, gestión de mensajes, escalabilidad y adaptación en tiempo de despliegue (Sierra, y otros, 2003).

Existen tres tipos de EJB (Subrahmanyam, y otros, 2002):

- EJB de **Entidad:** Se caracterizan por presentar persistencia gestionada por el contenedor (**CMP**) y persistencia gestionada por el Bean (**BMP**)
- EJB de **Sesión:** Pueden catalogarse en dos tipos: con estado (**stateful**) y sin estado (**stateless**).
- EJB **dirigidos por mensajes:** Son una clase especial de EJB que no están destinados a la invocación directa de cliente.

1.7.5 SERVIDOR DE APLICACIONES: GLASSFISH SERVER 3.1

Se denomina servidor de aplicaciones a un servidor en una red de computadoras que ejecuta ciertas aplicaciones. Usualmente se trata de un dispositivo de software que proporciona servicios de aplicación a las computadoras cliente. Un servidor de aplicaciones generalmente gestiona la mayor parte (o la totalidad) de las funciones de lógica de negocio y de acceso a los datos de la aplicación.

Glassfish Server 3.1 es un servidor de aplicaciones que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación, tomando como base de desarrollo el Application Server de Sun Microsystems (Sun Microsystems). Es gratuito y de código libre.

Glassfish soporta las últimas versiones de tecnologías como: JSP[9], JSF[10], Servlets, EJB, Java API para Servicios Web (JAX-WS), Arquitectura Java para Enlaces XML (JAXB), y muchas otras tecnologías. El servidor Glassfish está rodeado por una gran comunidad de desarrolladores con el mismo nombre, que están en continuo desarrollo detectando y corrigiendo errores a la vez que dotan al servidor de nuevas funcionalidades.

Entre los diferentes beneficios de utilizar esta tecnología se encuentran:

- Flexible, extensible y personalizable.
- Mejora de la productividad del desarrollador.
- Alta disponibilidad para JAX-WS, JMS, EJB.
- Clúster y balance de carga.

1.7.5.1 FUNDAMENTACIÓN DEL SERVIDOR DE APLICACIONES SELECCIONADO

La selección de un servidor de aplicaciones es una decisión estratégica y crucial para el desarrollo de cualquier aplicación. Desde una perspectiva de negocios y como consecuencia de la selección de Java como lenguaje de programación es necesario

[9] **JSP**: Java Server Pages (por sus siglas en inglés) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

[10] **JSF**: Java Server Faces (por sus siglas en inglés) es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.

elegir un servidor que pueda utilizarse para aplicaciones de escritorio ya que el módulo propuesto así lo define. Por esta razón Glassfish Server 3.1 es el servidor de aplicaciones seleccionado. En primer lugar Glassfish Server es el primer servidor de aplicaciones que apoya completamente a Java EE 6, plataforma de desarrollo seleccionada para la solución. En cuanto a rendimiento, Glassfish ofrece buenos tiempos de respuesta, es altamente reconocido por los desarrolladores por su facilidad de uso y administración, ya que proporciona tareas orientadas a consola de administración y asistentes de configuración, que simplifican las tareas administrativas rutinarias. Glassfish Server 3.1 proporciona pre-configuración para NetBeans IDE.

1.7.6 SISTEMA GESTOR DE BASES DE DATOS: POSTGRESQL 9.1

Un SGBD (Sistema Gestor de Bases de Datos) es un sistema computarizado cuya finalidad general es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones. La información en cuestión puede ser cualquier cosa que sea de importancia para el individuo u organización; en otras palabras, todo lo que sea necesario para auxiliarle en el proceso general de su administración (Date, 2010). **PostgreSQL** es un avanzado SGBD relacional, orientado a objetos y libre, disponible en una amplia gama de plataformas y publicado bajo la licencia BSD. Uno de los beneficios más claros de PostgreSQL es que es de código abierto, lo que significa que su licencia es muy permisiva para instalar, usar y distribuir. Es totalmente compatible con ACID[11], tiene soporte completo para claves foráneas, uniones, vistas, disparadores y procedimientos almacenados (en varios idiomas).Es un sistema compatible con conjuntos de caracteres internacionales, codificaciones de caracteres multibyte y Unicode, altamente escalable tanto en la enorme cantidad de datos que puede manejar como en el número de usuarios simultáneos que puede acomodar (PostrgreSQL Fundation).

[11] **ACID**: Término utilizado en bases de datos para referirse a un conjunto de características necesarias para que una serie de instrucciones puedan ser consideradas como una transacción. En concreto **ACID** es un acrónimo de **A**tomicity, **C**onsistency, **I**solation and **D**urability; en español Atomicidad, Consistencia, Aislamiento y Durabilidad.

1.7.6.1 FUNDAMENTACIÓN DEL SISTEMA GESTOR DE BASES DE DATOS SELECCIONADO

Se selecciona PostgreSQL Server como SGBD debido a que posee una gran escalabilidad. Es capaz de ajustarse al número de CPUs y a la cantidad de memoria que posee el sistema de forma óptima, haciéndole capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta. PostgreSQL implementa el uso de rollback's[12], sub-consultas y transacciones, haciendo su funcionamiento mucho más eficaz. De forma general PostgreSQL es un SGBD confiable, robusto y estable, que corre en la mayoría de los sistemas operativos actuales, posee un buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas, así como gran capacidad de almacenamiento.

1.7.7 HERRAMIENTA ADMINISTRATIVA PARA POSTGRESQL: PGADMIN III

Para administrar las bases de datos en PostgreSQL 9.1, se utiliza la herramienta PgAdmin III en su versión 1.14.0. Se trata de un software libre, que accede a todos los objetos del PostgreSQL y responde a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, y un agente para lanzar scripts programados (Antonio Aliaga Ibarra, 2008). Está escrito en C++ usando la librería gráfica multiplataforma, lo que permite que se pueda usar en Linux, Mac OS y Windows.

1.7.8 MAPEO OBJETO RELACIONAL (ORM): JAVA PERSISTENCE API 2.0 Y ECLIPSELINK 2.0

El mapeo objeto-relacional (por sus siglas en ingles ORM) es una técnica de programación para convertir datos entre el sistema de tipos de un lenguaje de programación orientada a objetos y el de una base de datos relacional, utilizando un motor de persistencia. Esto posibilita el uso de las características propias de la orientación a objetos.

[12] **Rollback**: En tecnologías de bases de datos es una operación que devuelve a la base de datos a algún estado previo.

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE. Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). Java Persistence API estandariza mapeo objeto-relacional, proporcionando un modelo de persistencia cimentado en POJO's (Plain Old Java Objects) para mapear bases de datos relacionales en Java. El mapeo objeto-relacional, o sea, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML. NetBeans puede ser configurado para programar con notación JPA a través de herramientas y plug-ins.
En su versión 2.0 JPA incluye funciones adicionales de mapeo y flexibles maneras de determinar la forma en que el proveedor da acceso al estado de la entidad, las extensiones a la persistencia Java Query Language (JPQL), y criterios para la creación de consultas dinámicas (Keith, y otros, 2009). Entre sus principales ventajas suministra independencia de la base de datos, bajo acoplamiento entre negocio y persistencia, y un desarrollo rápido.

Es importante conocer que JPA es solo la especificación de Java que proporciona un estándar para ORM, por lo que es necesario utilizar una implementación, en este caso EclipseLink en su versión 2.0.

EclipseLink provee un marco para que los desarrolladores de Java interactúen con diversos servicios de datos y ofrece a objetos relacionales servicios de mapeo y soporte completo para la especificación JPA, de manera sofisticada y de alto rendimiento (Eclipse Foundation, 2012).

EclipseLink aborda la disparidad entre los objetos Java y fuentes de datos. Contiene un framework de persistencia que le permite crear aplicaciones, que combinan los mejores aspectos de la tecnología de objetos con una fuente de datos específica, permitiendo la persistencia de objetos Java a prácticamente cualquier base de datos relacional y además mapear cualquier modelo de objetos para cualquier esquema relacional o no relacional. Un amplio conjunto de características se proporcionan en EclipseLink (Eclipse

Foundation, 2012): no intrusivo, flexible, basada en arquitectura metadatos, optimizado para un rendimiento altamente escalable y concurrencia con amplias opciones de ajuste del rendimiento, amplia capacidad de consulta que incluye JPQL, SQL nativo, entre otros.

1.7.9 HERRAMIENTAS DE CREACIÓN DE INFORMES: JASPER REPORT E IREPORT 3.1

Jasper Report es una biblioteca que permite el manejo y creación de informes, con la habilidad de entregar contenido enriquecido al monitor, a la impresora o a ficheros PDF, HTML, XLS, CSV[13] y XML. Está escrito completamente en Java y puede ser usado en gran variedad de aplicaciones de Java, incluyendo J2EE o aplicaciones web, para generar contenido dinámico. Su propósito principal es ayudar a crear documentos de tipo páginas, preparados para imprimir en una forma simple y flexible. Se encuentra bajo licencia libre GNU, por lo que es software libre. Jasper Report es una biblioteca que puede ser embebida en cualquier aplicación Java y sus funciones incluyen (Swenson, 2002):

- Scriptlets, que pueden acompañar a la definición del informe, y ser invocados en cualquier momento por la definición para realizar un procesamiento adicional. El scriptlet se basa en Java, y tiene muchos hooks (ganchos) que se pueden invocar antes o después de las etapas de la generación de informes, como el informe, página, columna o grupo.
- Sub-informes.

En otro sentido, **iReport 3.0.0** es un programa de código abierto que puede crear complejos reportes, que cada aplicación java puede utilizar mediante la librería Jasper Report. Esta escrito 100% en lenguaje de programación Java y es distribuido con código fuente, acorde con la licencia GNU. A través de una interfaz gráfica intuitiva y rica en recursos, iReport permite la creación rápida de cualquier informe (Toffoli, 2007).

[13] **CSV:** Comma-Separated Values (por sus siglas en inglés) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por y las filas por saltos de línea.

Es utilizado para simplificar el diseño de reportes y exportar ficheros, que la biblioteca Jasper Report es capaz de integrar a las aplicaciones Java. La lista siguiente describe algunos de los rasgos más importantes de iReport (Toffoli, 2007):

- 100% escrito en JAVA y además libre y gratuito.
- Maneja el 98% de las etiquetas de Jasper Report.
- Soporta internacionalización nativamente.
- Soporta JDBC.
- Soporta JavaBeans como orígenes de datos.

1.8 CALIDAD DE SOFTWARE. DEFINICIONES

Según Roger S. Pressman, la calidad del software es la: "*Concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo explícitamente documentados y con las características implícitas que se espera de todo software desarrollado profesionalmente*"(Pressman, 2002).
El único instrumento adecuado para determinar el status de la calidad de un producto software es el proceso de pruebas. En este proceso se ejecutan pruebas dirigidas a componentes del software o al sistema de software en su totalidad, con el objetivo de medir el grado en que el software cumple con los requerimientos (Pruebas de software, 2005).

Los **niveles de pruebas** se clasifican en cuanto a 2 criterios, el primero de ellos define a qué elemento se le realiza la prueba, estos son Nivel de unidad o Prueba unitaria, Nivel o Prueba de integración y Nivel o Prueba de sistema. El otro criterio define quién es el encargado de hacer las pruebas, dentro de estos se enumeran Pruebas de desarrollador,Pruebas Independientes y Pruebas de Aceptación. (Hernández, 2009)

El **nivel de unidad** conforma la prueba enfocada a los elementos probables más pequeños del software. Es aplicable a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera.

Las pruebas que se realizan en este nivel pueden ser mediante la utilización de dos métodos: **caja blanca** o **caja negra**.

La primera de estaspruebas, se selecciona en función del conocimiento que se tiene de la implementación del módulo. Se suelen aplicar a módulos pequeños, donde el probador analiza el código y deduce cuántos y qué conjuntos de valores de entrada han de probarse para que al menos se ejecute una vez cada sentencia del código.

Las pruebas de caja negra se encargan de seleccionar casos de pruebas basados en la especificación funcional del componente o programa y no su estructura interna.

La validación de la solución se centrará en la técnica de **camino básico** (método de caja blanca) que permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

Por otra parte la técnica de **particiones equivalentes** (método de caja negra) fracciona el dominio de entrada de un programa en clases de datos (con valores por encima, debajo, y en el rango establecido) de los que se pueden derivar casos de prueba.

La calidad del software es determinada por una serie de factores o características, según el modelo que se utilice. El modelo ISO/IEC 9126 categoriza los atributos de calidad de software en seis características: funcionalidad, confiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad. Según este mismo modelo de calidad la **funcionalidad** es la capacidad del software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas cuando el software se usa bajo las condiciones especificadas (ISO, 2005).Esta característica se desglosa en sub-características, tales como:

- **Idoneidad:** Capacidad del software para mantener un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados.

- **Precisión:** Capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario.
- **Interoperabilidad:** Capacidad del producto de software para interactuar recíprocamente con uno o más sistemas especificados.
- **Seguridad (informática):** Capacidad del producto de software para proteger información y los datos, para que personas o sistemas desautorizados no puedan leer o modificar los mismos, y las personas o sistemas autorizados tengan el acceso a ellos.

El IEEE Standard Glossary of Software Engineering Terms define métrica como "una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado" (Drake, y otros, 2009). Las métricas de software tienen un papel fundamental al obtener un producto de alta calidad, ya que determinan mediante estadísticas basadas en la experiencia, el avance del software y el cumplimiento de parámetros requeridos.

Existen dos tipos de métricas: las **internas** y las **externas**. Las métricas internas se aplican a productos de software no ejecutables durante el diseño y la codificación, con el objetivo de asegurar que se logre la calidad externa y la calidad de uso requerida. Este tipo de métrica permite evaluar la calidad del producto de software y lo referido a problemas de calidad antes que el producto de software sea puesto en ejecución. Las métricas externas usan medidas de un producto de software, derivadas del comportamiento del mismo, a través de la prueba, operación y observación del software, proporcionando el beneficio de que puedan evaluar la calidad del producto de software durante las pruebas o el funcionamiento (Fernández Díaz, y otros, 2009).

Con el objetivo de evaluar los requerimientos obtenidos se emplearán las métricas: estabilidad de los requerimientos y especificidad de los requerimientos que se detallan en el **Anexo 1**. Así mismo para medir la calidad del diseño de la solución propuesta, se utilizarán las siguientes métricas internas, que pueden consultarse en el **Anexo 2**: tamaño operacional de clase **(TOC)** y relaciones entre clases **(RC).**

Para medir lacaracterística de calidad **funcionalidad** se utilizarán métricas externas, aportadas por la ISO/IEC 9126 y orientadas a cada una de las sub-características de calidad evaluadas. Estas son: adecuación funcional, precisión, capacidad de revisión de cuentas de acceso y capacidad de control de acceso. Las métricas mencionadas se podran consultar en el **Anexo 3**.

1.9 CONCLUSIONES DEL CAPÍTULO

En el presente capítulo se logró identificar, mediante el análisis de los sistemas existentes en el contexto, que ninguno permitía satisfacer el proceso en cuestión. Además se identificaron los estilos arquitectónicos claves para la solución después de una búsqueda exhaustiva. A través del estudio de metodologías de desarrollo de software se identificó a RUP como la más idónea, así mismo se consiguió definir las herramientas y tecnologías más factibles en el contexto contemporáneo, de manera que cumpliera con las políticas de nuestro país y brindara la mayor cantidad de ventajas al desarrollo.

CAPÍTULO 2: DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

2.1 INTRODUCCIÓN

En el presente capítulo se dará a conocer una propuesta general del módulo que será desarrollado para satisfacer las necesidades del proceso de ferias y exposiciones de la Cámara de Comercio, en relación a la administración y generación de reportes. Se realiza un estudio detallado del dominio, donde se describen los actores, trabajadores, casos de uso y se muestran los diagramas correspondientes. También se especifican los requerimientos funcionales y no funcionales del software, así como los estándares del diseño, el modelo de datos y los diagramas de clases del diseño. Además se precisan los patrones de diseño utilizados y se realiza una descripción de la arquitectura.

2.2 DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN

Con el desarrollo del capítulo 1 se concluyó que ninguna de las herramientas analizadas para la gestión de ferias y eventos comerciales de la Cámara de Comercio eran adecuadas, por lo era necesario una nueva propuesta capaz de llevar a cabo el proceso poniendo énfasis en aspectos de configuración, seguridad y generación de reportes. El módulo propuesto será responsable de garantizar la seguridad de los datos, la gestión dinámica de nomencladores y la salida de informes para el Sistema de Gestión de Ferias y Exposiciones de la Cámara de Comercio.

2.3 MODELO DE DOMINIO

Un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno en el que trabaja el sistema. El modelo de dominio se describe mediante diagramas UML (especialmente mediante diagramas de clases). De manera general el modelo del dominio ayuda a los usuarios, clientes, desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se sitúa el sistema. En este caso se

decidió realizar un modelo de dominio, ya que no se logró identificar con claridad los procesos del negocio, con fronteras bien establecidas.

Los conceptos fundamentales que se encuentran en el dominio se detallan en el artefacto **0012_CEGEL_SIGFE_Modelo de Dominio_Administración y Reportes.doc**:

2.3.1 DIAGRAMAS DE CLASES DEL DOMINIO

Los diagramas de clases de dominios correspondientes al módulo se muestran diferenciados por los escenarios: eventos, seguridad, recursos y entidades.

A continuación, el diagrama de clases del dominio del escenario seguridad, para ver los restantes diagramas consultar artefacto **CEGEL_SIGFE_0012_Modelo de Dominio_Administración y Reportes.doc**:

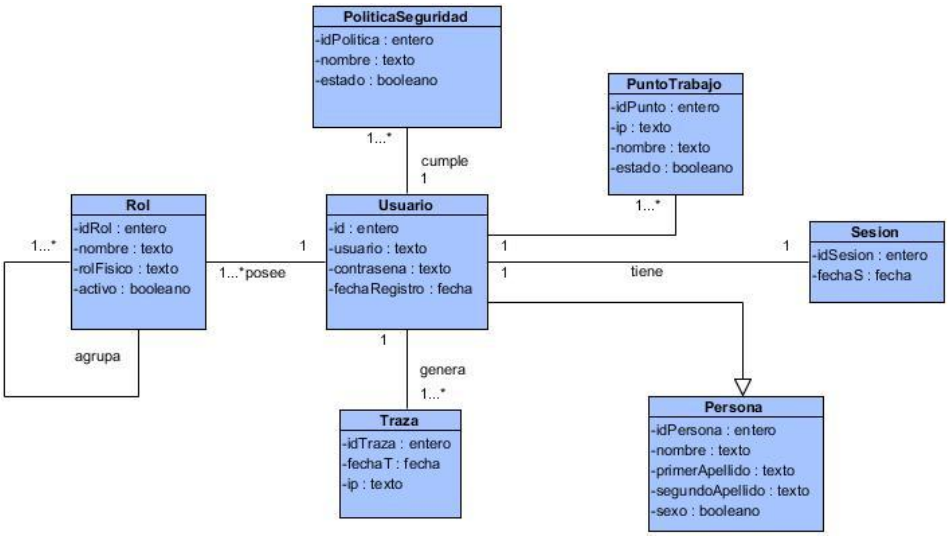


Figura 3: Diagrama de clases del dominio del escenario seguridad.

2.4 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Según el libro: "El proceso unificado de desarrollo de software" de Ivar Jacobson, Grady Booch y James Rumbaugh la captura y especificación de requerimientos es: "*el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir*" para lograr "*guiar el desarrollo hacia el sistema correcto*".

Los mismos se pueden clasificar en funcionales, conciderando las funciones o capacidades que el sistema debe cumplir; y no funcionales, teniendo en cuenta las cualidades o propiedades que debe poseer el producto.

2.4.1 REQUERIMIENTOS FUNCIONALES

En el artefacto **CEGEL_SIGFE_0113_ERS_Administración y Reportes.doc** se puede observar la especificación de requerimientos del módulo; sin embargo, los requerimientos funcionales mas importantesdefinidos para el sistema son los siguientes:

- RF_1. Activar aplicación en estación de trabajo.
- RF_2. Registrar cuenta de usuario.
- RF_3. Modificar cuenta de usuario.
- RF_4. Eliminar cuenta de usuario.
- RF_5. Buscar cuenta de usuario.
- RF_6. Adicionar rol a usuario.
- RF_7. Eliminar rol a usuario.
- RF_8. Registrar local.
- RF_9. Modificar local.
- RF_10. Eliminar local.
- RF_11. Buscar local.
- RF_12. Mostrar entidades Inscritas.

2.4.2 REQUERIMIENTOS NO FUNCIONALES

- RnF 1. Facilidad de aprendizaje:
  - ✓ Proveer manuales de instalación, configuración y uso del sistema.
- RnF 2. Facilidad de uso:
  - ✓ Los menús permitirán una navegación sencilla, para todos los usuarios. La ayuda incluye información sobre el proceso de negocio.
- RnF 3. Mínimo impacto de los errores.

- ✓ El sistema permite cancelar las instancias de los procesos y las tareas, y notifica a los usuarios los errores.

- RnF 4. Notificación de omisión o errores en los datos introducidos.
  - ✓ El sistema notifica al usuario los errores u omisiones en los datos introducidos.

- RnF 5. Eficiencia.
  - ✓ El sistema debe demorar como promedio en una transición entre un segundo y 6 segundos como tiempo máximo aproximadamente.

- RnF_6. Servidores de base de datos locales PostgreSQL versión 9.1.
- RnF_7. Máquina Virtual de Java en su versión JDK 1.6.
- RNF_8. Hardware mínimo para clientes:
  - ✓ 512 MB de memoria RAM.
  - ✓ 20 GB de disco duro.
  - ✓ Procesador Pentium IV.

- RNF_9. Hardware mínimo para servidores:
  - ✓ 2 GB de memoria RAM.
  - ✓ 120 GB de disco duro.
  - ✓ Procesador Dual Core o superior.

- RnF_10. Portabilidad.
  - ✓ El sistema es una aplicación de escritorio que se utiliza en la Cámara de Comercio de Cuba y en la sede del evento.

- RnF_11. Restricciones de diseño.
  - ✓ Herramienta de Modelado Visual Paradigm: Se utilizará la herramienta CASE Visual Paradigm for UML 8.0 Enterprise Edition, teniendo en cuenta sus ventajas para modelar los diferentes artefactos que se obtendrán en los flujos de trabajoy sus diferentes fases.

- ✓ Lenguaje de Programación: El software estará programado en Java, siguiendo una codificación estándar y organizada, haciendo uso de las potencialidades propias del lenguaje.
- ✓ Entorno de desarrollo integrado (IDE): El software se desarrollará sobre NetBeans 7.2 ya que contiene las herramientas para llevar aefecto la implementación de la totalidad de los componentes impuestos por la arquitectura y el diseño.

- RNF_12. La autenticación estará basada en el uso de credenciales.
- RNF_13. Las contraseñas se almacenarán en la base de datos haciendo uso del algoritmo hash MD5[14].
- RNF_14. Los usuarios estarán autorizados a realizar las acciones que se encuentran definidas para el rol al cual pertenecen.
- RNF_15. Solo se podrá acceder a la aplicación desde las direcciones IP autorizadas.

2.5 DEFINICIÓN DE CASOS DE USO

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

2.5.1 DEFINICIÓN DE LOS ACTORES

Actores	Justificación
Administrador del sistema	Es la persona encargada de los procesos administrativos, como creación de cuentas de usuario, asignación de roles y otros elementos de la seguridad del sistema. También gestiona los nomencladores.
Organizador	Es el encargado de la organización, tiene el control total de todos los módulos existentes en el evento o feria. Consulta informes relacionados con la

[14] **MD5**: Abreviatura del inglés Message-Digest Algorithm 5 (Algoritmo de Resumen del Mensaje 5), es un algoritmo de reducción criptográfico.

de feria	organización.
Diseñador de feria	Es el responsable de la asignación de los espacios y recursos para el desarrollo del evento o feria, consultando los informes relacionados.
Usuario	Se trata de cualquier usuario que acceda al sistema.
Sistema	Se encarga de las funcionalidades automáticas que se ejecutan en el sistema.

Tabla 1: Actores del sistema.

2.5.2 LISTADO DE CASOS DE USO

A continuación se muestran los resúmenes de los casos de uso más importantes del módulo, para ver detalles de los mismos consultar el artefacto **CEGEL_SIGFE_0114_Especificación de casos de uso_Administración y Reportes**.

CU_1	**Activar aplicación en estación de trabajo**
Actor	Administrador
Descripción	Permite activarla aplicación en una estación de trabajo que no ha interactuado anteriormente con el sistema.
Referencia	RF_1.
CU_2	**Gestionar cuentas de usuario**
Actor	Administrador del sistema
Descripción	Permite el registro, modificación, eliminación o búsqueda de una cuenta de usuario en el sistema.
Referencia	RF_2, RF_3, RF_4, RF_5
CU_3	**Gestionar rol a usuario**
Actor	Administrador del sistema
Descripción	Permite la asignación o eliminación de roles a un usuario determinado.
Referencia	RF_6, RF_7.
CU_4	**Gestionar local**
Actor	Diseñador de la feria
Descripción	Permite registrar, modificar, eliminar y buscar los locales en los que se organizan las ferias y eventos.
Referencia	RF_8, RF_9, RF_10, RF_11.
CU_5	**Mostrar reporte: Entidades inscritas**
Actor	Organizador de la Feria
Descripción	Permite mostrar un informe donde se ofrecen datos estadísticos de las

	entidades inscritas en algún evento registradas en el sistema.
Referencia	RF_12.

Tabla 2: Fragmento de los resúmenes de los casos de uso del sistema.

2.5.3 DIAGRAMA DE CASOS DE USO

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario, describiéndolo bajo la forma de acciones y reacciones y permitiendo definirlos límites del sistema y sus relaciones con el entorno. En la fase de inicio de la metodología seleccionada el diagrama de casos de uso del sistema es uno de los artefactos más importantes que se genera. A continuación se puede observar el diagrama de casos de uso del sistema (ver figura 8):

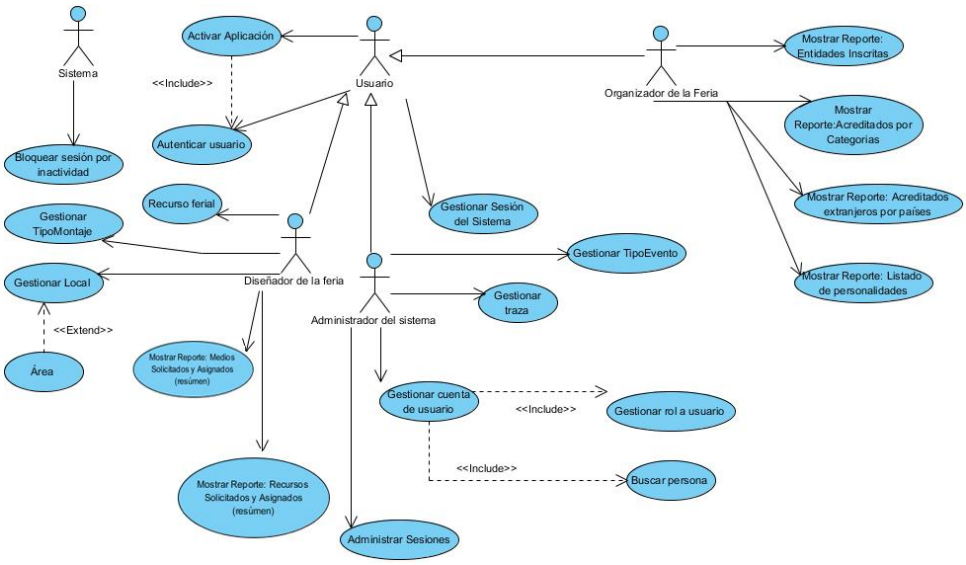


Figura 4: Diagrama de casos de uso del sistema

2.5.4 DESCRIPCIÓN TEXTUAL

Mediante la descripción textual se puede comprender mejor cómo es que se realiza el caso de uso, las relaciones entre el actor y el sistema, además de cómo responde este último. Por la importancia que presenta el caso de uso Gestionar rol a usuario a continuación se ofrece un fragmento de la descripción textual del mismo, las restantes descripciones, junto a sus prototipos de interfaz pueden ser consultadas en el artefacto**CEGEL_SIGFE_0114_Especificación de casos de uso_Administración y Reportes.doc**

Objetivo	Asignación o eliminación de roles a un usuario. RF_6, RF_7
Actores	Administrador del sistema.
Resumen	El caso de uso inicia cuando es invocado desde el caso de uso Gestionar cuentas de usuario. Consiste en mostrar el listado de roles asignados y no asignados a un usuario del sistema, con el objetivo de seleccionar aquellos que deben retirársele como mecanismo de restricción de permisos para el uso de la aplicación informática, o seleccionar los roles disponibles que permitan proveer al usuario de nuevos permisos para realizar satisfactoriamente su trabajo. El caso de uso termina al hacer efectiva la nueva configuración de los roles.
Complejidad	Baja
Prioridad	Crítico
Precondiciones	El Administrador debe estar autenticado con los permisos necesarios.
Postcondiciones.	

Flujo de eventos

Flujo básico Gestionar rol a usuario

	Actor	**Sistema**
1.		El sistema muestra la interfaz “Asignación de roles a usuario” con los siguientes datos: **Datos generales de la persona propietaria y el nombre de usuario asignado.** • Primer nombre. • C.I. • Nombre de usuario. • Estado. **Roles de usuario.** • Listado de Roles asignados. • Listado de Roles disponibles. Además muestra las opciones Adicionar y Eliminar deshabilitadas inicialmente, hasta que no se seleccione un rol. Además muestra las opciones:

		✓ Aceptar ✓ Cancelar
2.	Selecciona una de las siguientes opciones: • Si selecciona un rol de la lista de roles disponibles (Ver sección "**Adicionar rol**"). • Si selecciona un rol de la lista de roles asignados (Ver sección "**Eliminar rol**").	

Tabla 3: Fragmento de la descripciónтextual del CU_3: Gestionar rol a usuario.

2.6 MODELO DE ANÁLISIS

El análisis se preocupa de ver que hace el sistema, interesándose solo por los requerimientos funcionales, permitiendo así estructurar los requerimientos de manera que facilite su comprensión (Jacobson y otros, 2000).

2.6.1 DIAGRAMAS DE CLASES DEL ANÁLISIS

El diagrama de clases del análisis está compuesto por clases del análisis y sus relaciones. Los tipos de clases utilizados en el modelo de análisis son:

- **Clase Interfaz (CI):** Modelan las formas de interacción entre los actores y el sistema.
- **Clase Controladora (CC):** Encapsulan el comportamiento de cada caso de uso y coordinan el trabajo de las clases interfaz y entidad.
- **Clase Entidad (CE):** Modelan toda la información del sistema que posee una vida larga y que puede ser persistente.

A continuación se observan los diagramas de clases del análisis del caso de uso críticoGestionar rol a usuario.Los restantes diagramas de pueden consultar en el artefacto: **CEGEL_SIGFE_0114_Diagramas clases análisis_Administración y Reportes.rar**

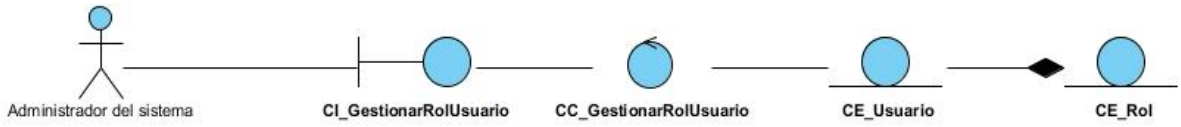


Figura 5: Diagrama de clases del análisis CU_3: Gestionar rol a usuario.

2.6.2 DIAGRAMAS DE SECUENCIA DEL ANÁLISIS

Los diagramas de secuencia muestran cómo las instancias específicas de las clases trabajan juntas para alcanzar un objetivo común. A continuación se presenta el diagrama de secuencia del análisis correspondiente al escenario adicionar rol del caso del uso crítico Gestionar rol a usuario:

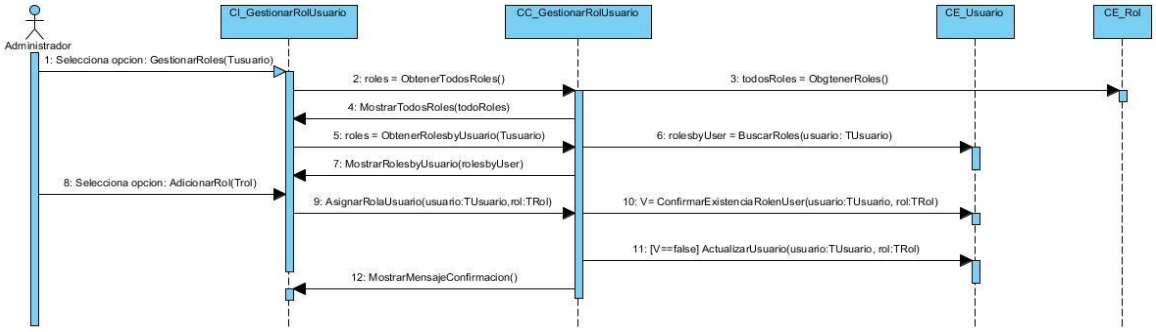


Figura 6: Diagrama de secuencia del análisis. Escenario adicionar rol del CU 3: Gestionar rol a usuario.

2.7 MODELO DE DISEÑO

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requerimientos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar.

2.7.1 DIAGRAMAS DE CLASES DEL 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.

A continuación, el diagrama de clases de diseño del CU_3: Gestionar rol a usuario.Los restantes diagramasse pueden consultaren el artefacto **CEGEL_SIGFE_0121_Modelo de diseño_Administración y Reportes.doc**

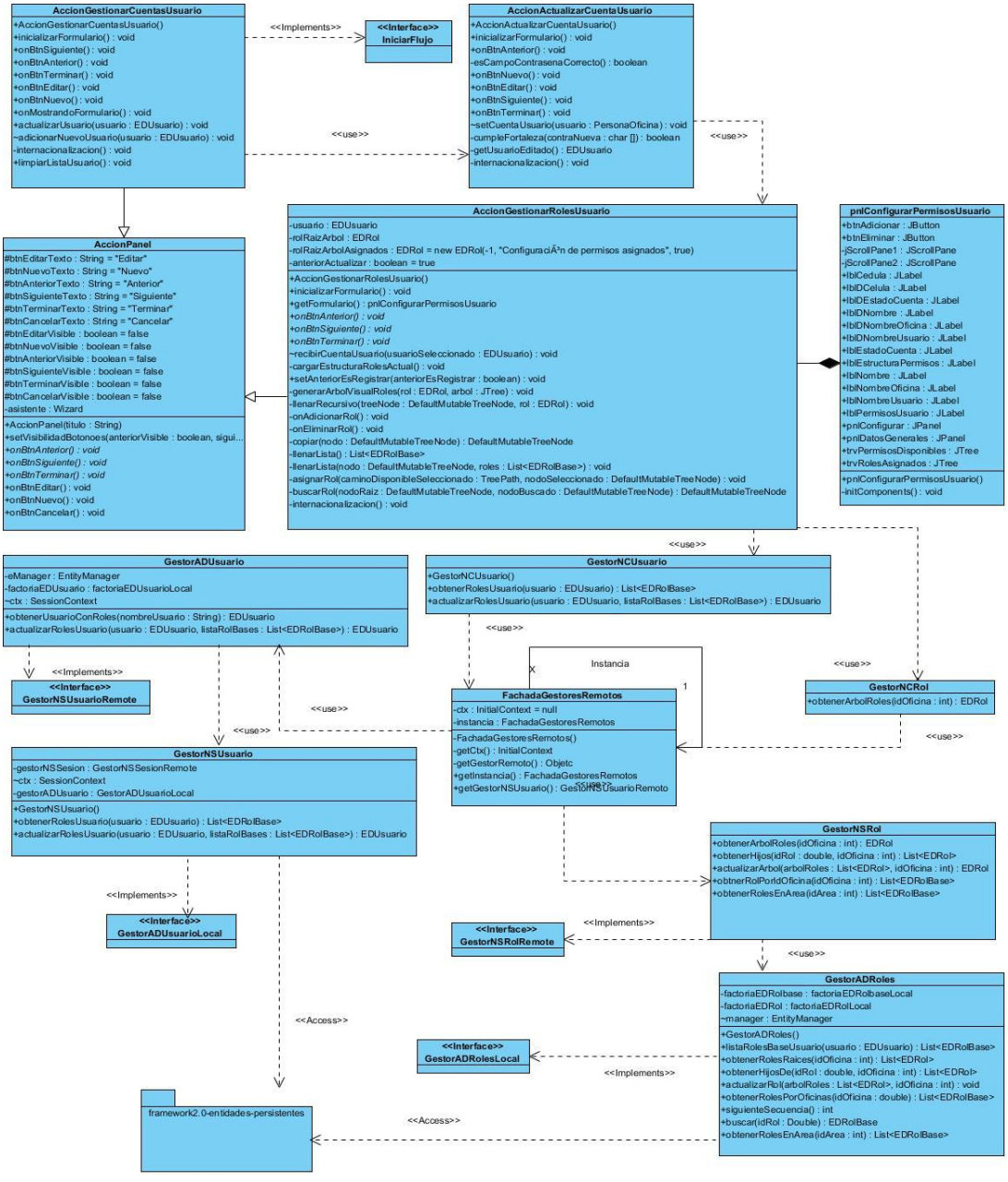


Figura 7: Diagrama de clases del diseño CU_3: Gestionar rol a usuario.

2.7.2 ESTÁNDARES DEL DISEÑO

Los estándares de diseño definen un conjunto de reglas para diseñar las interfaces del sistema, los cuales permiten obtener un diseño de alta calidad y que cumpla con las buenas prácticas establecidas en la Ingeniería de Software.

A continuación se describen los estándares de diseño que se utilizarán para el desarrollo del módulo propuesto:

- La separación de los componentes respecto a los bordes del área de trabajo (margen izquierdo, derecho, superior e inferior) debe ser de 10 píxeles como mínimo.
- En un formulario, los datos se agruparán utilizando paneles con título. El título del panel se escribe en formato de oración, en negrita y sin dos puntos al final.
- Cuando el formulario tiene varios componentes. El nombre de las etiquetas se escribe al lado de los componentes. El texto en las etiquetas se escribe en formato de oración, sin utilizar negrita y terminado en dos puntos. Los componentes irán al lado, a una distancia de 3 píxeles, cada componente de entrada de datos debe tener una altura de 23 píxeles. La separación horizontal entre componentes debe ser de 20 píxeles y vertical de 15 píxeles.
- Si el formulario tiene pocos componentes, se pondrá primero la etiqueta y al lado el componente, ambos siguiendo las pautas anteriores. Además se alinean las etiquetas a la izquierda y los componentes también a la izquierda. El texto de las etiquetas, títulos de columnas de tablas, botones, entre otros debe ser siempre en formato de oración.
- Los combos deben mostrar como texto inicial --Seleccione--. Los ítems para seleccionar deben escribirse en formato de oración.
- Los botones siempre estarán situados en la parte inferior derecha de los formularios, con una altura de 25 píxeles, siempre dejando un espacio de 8 píxeles delante y detrás de la palabra o el ícono. Sólo en el caso de los mensajes de confirmación, avisos y error los

botones irán en el centro. El texto que contienen debe estar en formato de oración y sin utilizar negrita.

- Los botones no llevan íconos, el Anterior y Siguiente llevan los caracteres <,> respectivamente. Los botones de SI y NO en los mensajes de confirmación deben llevar íconos y tendrán un ancho de 50 píxeles.

2.8 DISEÑO DE LA BASE DE DATOS

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

2.8.1 MODELO DE DATOS

El Modelo de datos constituye el principal elemento de diseño de bases de datos que consiste en representar: objetos (entidades que existen y que se manipulan), atributos (características básicas de estos objetos) y relaciones (forma en que se enlazan los distintos objetos entre sí). El modelo de datos del módulo fue dividido en los siguientes escenarios: seguridad y ferias–exposiciones.

A continuación se presenta el modelo de datos correspondiente al escenario seguridad, el restante modelo pueden consultarse en el **Anexo 6.**

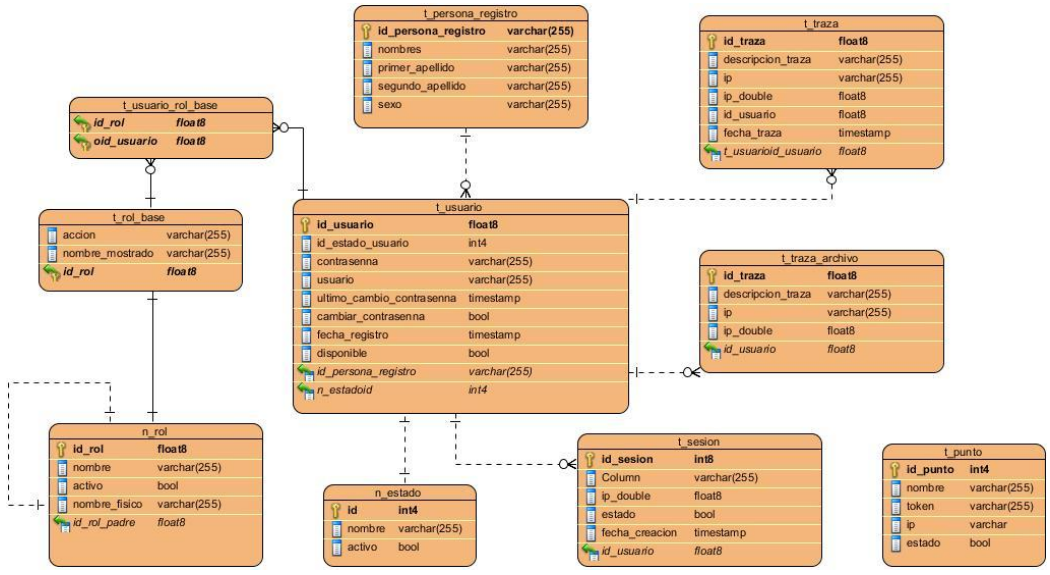


Figura 8: Modelo de Datos. Escenario seguridad.

2.8.2 DESCRIPCIÓN DE LAS TABLAS

Nombre	Descripción
n_rol	Representa todos los roles existentes en el sistema.
n_estado_usuario	Nomenclador que almacena todos los estados de los usuarios.
t_rol_base	Almacena los datos generales que representan los roles desempeñados por los usuarios del sistema.
t_usuario	Almacena los datos generales de los usuarios del sistema.
t_usuario_rol_base	Representa la relación entre la tabla usuario y rol_base, especificando que los usuarios pueden tener muchos roles base y los roles base, pueden ser desempeñados por varios usuarios.
t_persona_registro	Representalos datos de todas las personas que se encuentran registradas en el sistema que pueden ser usuarios del sistema.
t_sesion	Almacena las sesiones presentes en el sistema.
t_traza	Almacena las trazas que surgen en el sistema.
t_punto	Almacena todos los puntos de los puestos de trabajo en el sistema.

Tabla 4: Descripción de las tablas de la base de datos.

2.9 ARQUITECTURA DEL SISTEMA

Atendiendo al análisis realizado en la fundamentación teórica de la investigación (capítulo 1), se definió para el sistema una combinación del estilo Cliente/Servidor, distribuida en n niveles, y basada en el desarrollo de componentes de software.

Empleando la arquitectura Cliente-Servidor, el sistema cuenta con un cliente que presenta llamadas a un servidor de aplicaciones y este retorna el resultado de la llamada hacia el cliente que se encarga de la presentación de los datos en una interfaz de usuario, usando el patrón de diseño Modelo Vista Presentador.

Además la arquitectura se encuentra dividida en tres niveles físicos por la distribución o localización de las capas lógicas involucradas en la solución (ver Figura 9):

Figura 9: Distribución de las capas de la arquitectura.

A continuación en la tabla, una descripción de las capas de la arquitectura:

- **Lógica de Presentación:** Es responsable de la lógica de presentación, mostrado los datos, que obtiene de la capa Lógica de Negocio Cliente en la interfaz de usuario. Además controla los eventos que se generan en la misma. Solo se relaciona con la capa de Lógica de Negocio Cliente atendiendo al principio de que cada capa solo se comunica con otra vecina a ella, proveyéndola de los elementos necesarios para su procesamiento de datos.
- **Lógica de Negocio Cliente:** Gestiona la lógica de negocio relacionada en la parte cliente, y el acceso a las funcionalidades de los componentes en el servidor. En ella se encapsularán lógicas de acceso a dispositivos externos y la lógica de comunicación con el servidor de aplicaciones. La lógica de negocio está implementada sobre un concepto que se agrega a la arquitectura y es el concepto de gestor de negocio, que no es más que la agrupación de funcionalidades que se refieren a una misma entidad en un gestor con su nombre.
- **Lógica de Negocio Servidor:** Es responsable de presentar una fachada a todos los componentes que se encuentran en el servidor a modo de gestores de negocio, físicamente alojada en un servidor de aplicaciones al cual se accede de forma remota.

Los gestores de negocio servidor presentan una interfaz y su implementación contiene la lógica de negocio de las funcionalidades y hace llamadas a su capa inmediata superior (Lógica de Negocio de Acceso a Datos) de la cual obtiene los datos y se los retorna a la capa de Lógica de Negocio Cliente.

- **Lógica de Negocio Acceso a Datos:** Recibe los datos de la capa de Lógica de Negocio Servidor y los puede consultar, persistir, actualizar, y eliminar en la base de datos, mediante mecanismos que ofrecen las especificaciones utilizadas, que se realizan en conjunto con la capa de Entidades Persistentes. También se administran aspectos como las propiedades de las conexiones a los SGBD, y los conjuntos de tablas que se desean manejar, configuraciones para mejorar el rendimiento y la velocidad de las transacciones.
- **Entidades Persistentes:** Representan una relación entre la base de datos y los objetos de la programación, donde cada entidad se corresponde con una tabla. Esta capa abstrae del uso de un SGBD determinado, ya que estas entidades se refieren a elementos genéricos del modelo de datos y no a las especificaciones de cada uno de los gestores.
- **Datos:** La capa de datos se encuentra ubicada físicamente en el SGBD, en ella se encuentran los datos almacenados listos para ser consultados y actualizados, también se encuentran mecanismos para facilitar el trabajo con los datos como las funciones y secuencias.
- **Seguridad:** La capa de Seguridad está implícita a través de todas las capas de la arquitectura, muy relacionada a las tecnologías que se seleccionen para la implementación pero tiene aspectos generales que son fundamentales y deben ser llevados en cuenta, como el acceso y uso de recursos del servidor; operaciones que se pueden realizar en los clientes; mecanismos de seguridad. Cada nivel físico tiene sus especificaciones de seguridad porque los elementos que tiene son distintos.

La arquitectura se basa además en la división de la lógica de la aplicación en componentes de software para elevar los niveles de reutilización, mantenibilidad, integración y seguridad. La figura 10 muestra la interacción de los componentes:

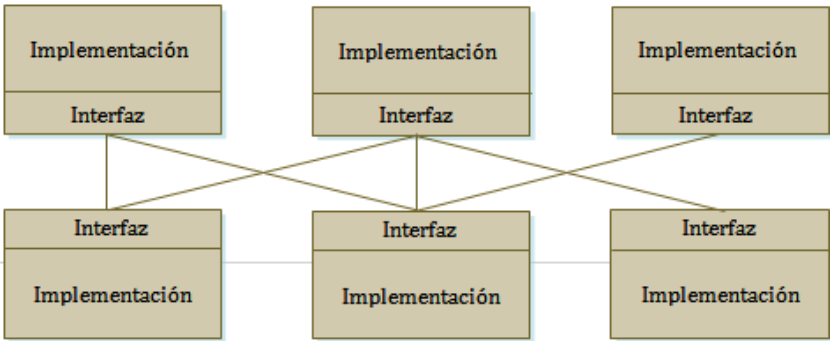


Figura 10: Interacción entre componentes a través de interfaces.

2.10 PATRONES DE DISEÑO

Un patrón de diseño provee un esquema para refinar los subsistemas o componentes de un sistema de software, o las relaciones entre ellos. Describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular (Buschmann, y otros, 1996).

Los patrones más importantes a usar en el desarrollo de software de la solución presentada son los siguientes:

- **Experto:** Se utilizará para que cada objeto realice la funcionalidad de acuerdo a la información que domina. Se evidencia en las clases gestoras de negocio, las cuales agrupan funcionalidades y toman decisiones con información afín a una entidad determinada.
- **Alta cohesión:** Se evidencia en el diseño de cada una de las clases del sistema, garantizando que solo exista en ellas las características y funcionalidades necesarias. Entre estas clases se encuentran las clases gestoras de negocio cliente y servidor.
- **Bajo acoplamiento:** Permitirá estimular la asignación de responsabilidades de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Al lograr una alta cohesión en diseño de las clases del sistema se garantiza un bajo acoplamiento ya que permite tener clases más

independientes, donde la realización de cambios sea más sencilla. Este patrón se observa en las clases gestoras de negocio cliente y servidor.

- **Controlador:** Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que recibe los datos del usuario y los envía a las distintas clases según el método llamado. En este caso las clases Acciones son las encargadas de implementar este patrón, mediante el control de los eventos del usuario a través de los formularios.
- **Instancia única:** Permitirá la creación de una única instancia de las clases que sean necesarias, Se observa este patrón en la clase de internacionalización y la FachadaGestoresRemoto.
- **Fachada:** Proporcionar una interfaz para la creación de familias de objetos interdependientes o interrelacionados, sin especificar sus clases concretas. El patrón se utilizó al diseñar interfaces con funcionalidades bien definidas que van a ser implementadas por todas las clases que las necesiten.
- **Proxy:** Se utiliza como intermediario para acceder a un objeto, permitiendo controlar el acceso a él. Se observa en la creación de la clase FachadaGestoresRemotos para facilitar la conexión entre los gestores del lado del cliente y del servidor.

2.11 CONCLUSIONES DEL CAPÍTULO

Durante el transcurso de este capítulo se logró obtener una visión general de las principales características del módulo propuesto, partiendo de los requerimientos funcionales. La realización de los artefactos fundamentales que propone la metodología seleccionada como por ejemplo el modelo de dominio, modelo de análisis y de diseño, junto al modelo de datos, permitieron sentar las bases para comprender el contexto del sistema en todas sus variantes y comenzar la implementación del mismo. De igual manera detallar la arquitectura del sistema, los patrones de diseño utilizados y los estándares de diseño lograron dotar al equipo de trabajo de los conocimientos necesarios para las posteriores etapas del desarrollo.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 INTRODUCCIÓN

En el presente capítulo se abordarán temas relacionados con la implementación y las pruebas del sistema, aspectos que son fundamentales para cumplir con las necesidades y espectativas del cliente y desarrollar un producto con la calidad requerida. Como parte de esta sección, se desarrollará el Modelo de implementación y se describirán los estándares de codificación empleados, así como el procedimiento para medir la funcionalidad del sistema, mediante la aplicación de pruebas de caja negra, caja blanca y métricas de calidad.

3.2 MODELO DE IMPLEMENTACIÓN

Está conformado por los diagramas de componentes y de despliegue, describiendo cómo los elementos del modelo de diseño se desarrollan en términos de componentes, ficheros de código fuente o ejecutables. Este modelo describe también la forma en que se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización, disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y cómo dependen los componentes unos de otros.

Seguidamente se muestran los diagramas de componentes y de despliegue, además de los estándares de codificación de la solución.

3.2.1 DIAGRAMA DE COMPONENTES

Los diagramas de componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema (Sparx Systems Pty Ltd., 2000-2007). Permite modelar la vista estática del sistema mostrando dependencias lógicas entre un conjunto de componentes de software. A continuación, el diagrama de componentes relativo al módulo, dividido por capas:

Capa Presentación:

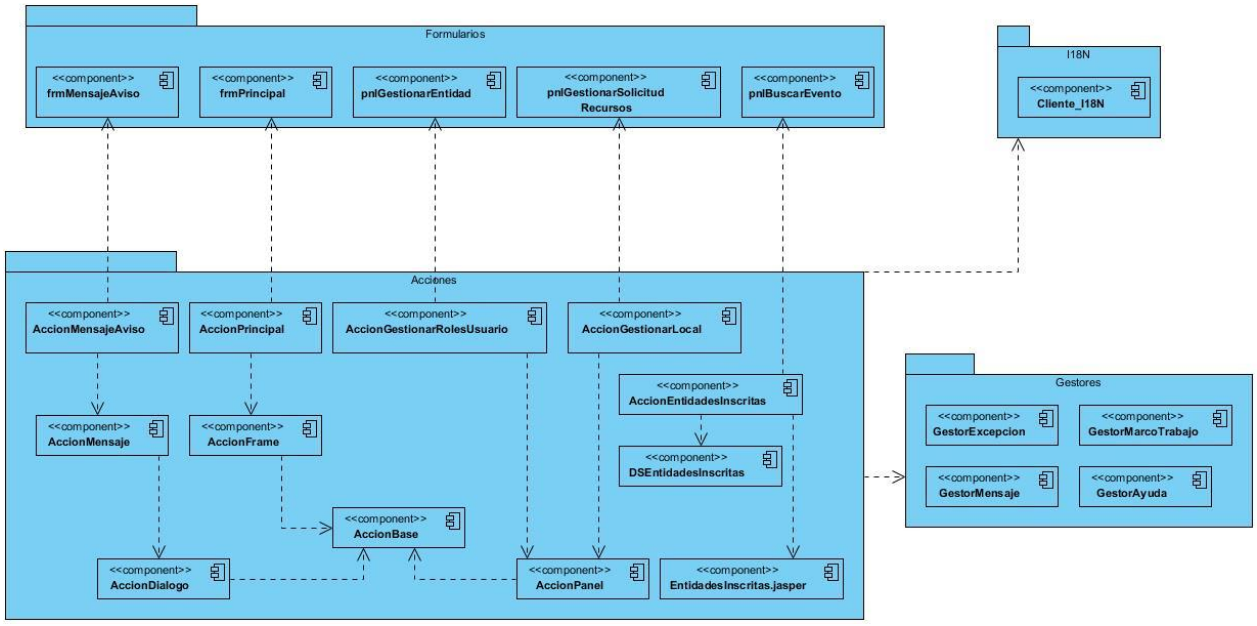


Figura 11: Diagrama de Componentes de la Capa de Presentación

Capa Lógica de Negocio:

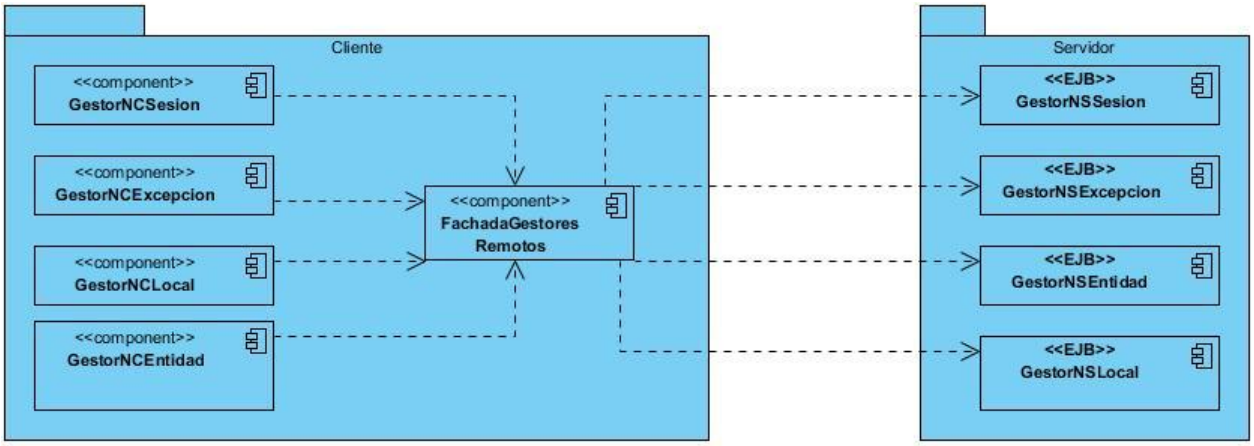


Figura 12: Diagrama de Componentes de la Capa Lógica Negocio

Capa de Acceso a Datos:

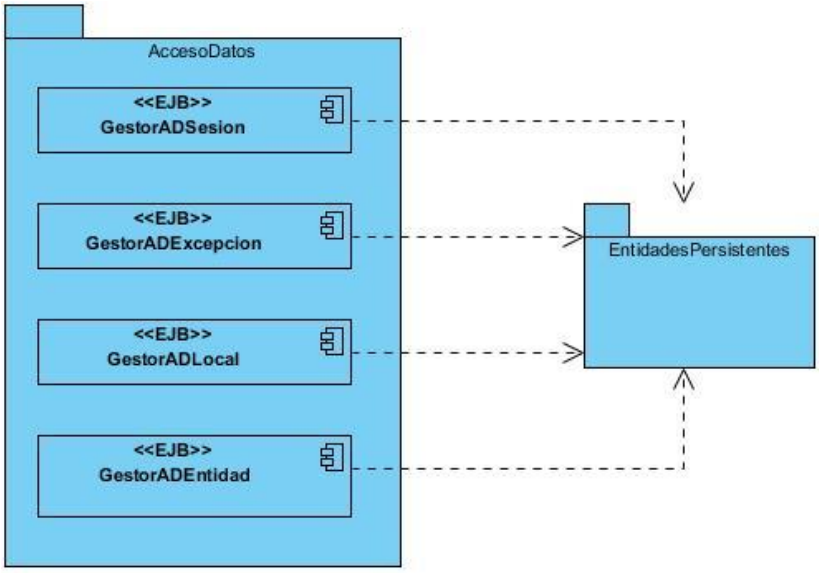


Figura 13: Diagrama de Componentes de la Capa Acceso a datos.

En el siguiente diagrama de integración se puede apreciar cómo interactúan los componentes de las capas descritas con anterioridad:

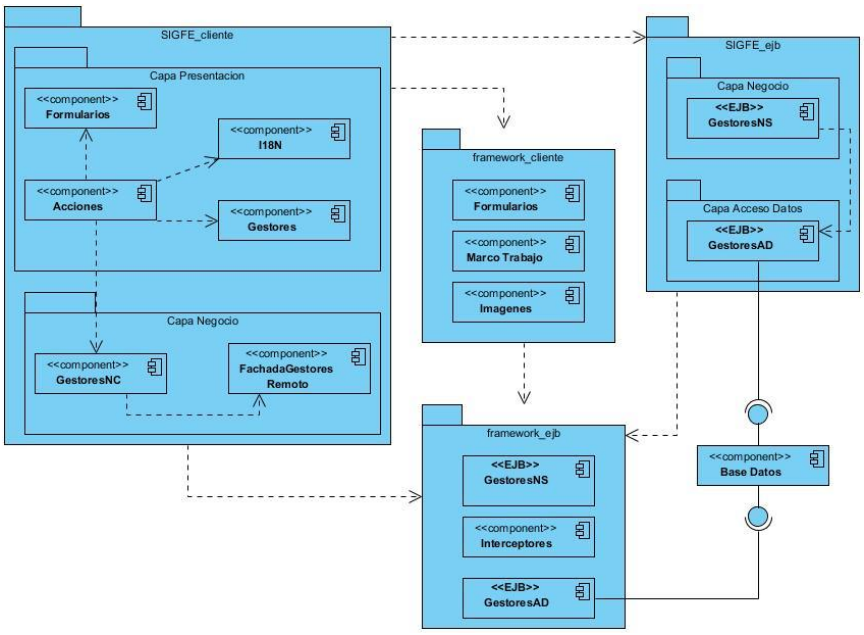


Figura 14: Diagrama de integración de componentes

3.2.2 DIAGRAMA DE DESPLIEGUE

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Al mismo tiempo un nodo es un objeto físico en tiempo de ejecución que representa un recurso computacional, generalmente con memoria y capacidad de procesamiento.

A continuación se muestra en la figura 16 el Diagrama de Despliegue para el módulo Administración y Reportes para el Sistema de Gestion de Ferias y Exposiciones de la Cámara de Comercio.

- **Nodo_Cliente:** Representan las estaciones de trabajo donde los usuarios interactúan con la aplicación, estableciendo una comunicación con el Nodo_Servidor a través del protocolo RMI-IIOP (Remote Method Invocation, en español Método de Invocación Remota) – (Internet Inter-ORB Protocol, en español Protocolo de Internet Inter-ORB), que brinda una comunicación remota en aplicaciones distribuidas.
- **Nodo_Servidor:** Agrupa el servidor de aplicaciones Glassfish en su versión 3.1 y el Servidor de Base de datos PostgreSQL en su versión 9.1.
- **Nodo Impresora:** Representa las impresoras, que serán empleadas en caso de imprimir algún reporte.

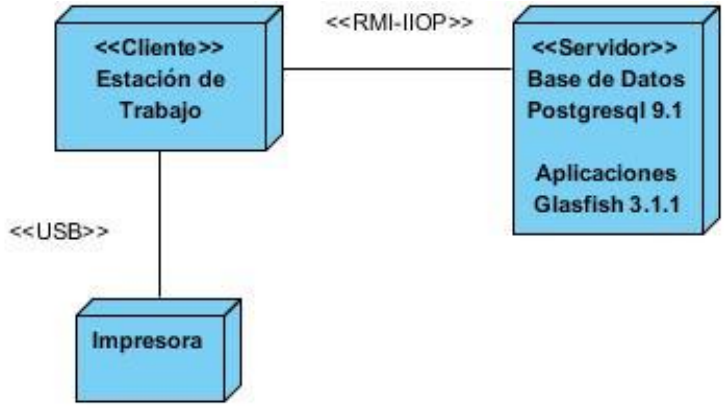


Figura 15: Diagrama de Despliegue

3.3 ESTÁNDARES DE CODIFICACIÓN

Un estándar de codificación es una regla que se sigue para la escritura del código fuente. Se definen estándares de codificación porque un estilo de programación homogéneo en un proyecto permite que todos los participantes lo puedan entender en menos tiempo y que el código en consecuencia sea mantenible.
A continuación se describen los estándares de codificación que se utilizarán para el desarrollo del sistema:

Nomenclatura para los paquetes, métodos y clases:

- Los paquetes (o package) deberán comenzar con el nombre del sistema (SIGFE) seguido la capa lógica a la que pertenece.
- Se utilizará la notación Camello en los nombres de las entidades persistentes, entidades de dominio, propiedades y funcionalidades del sistema.
- Los formularios de tipo pop-up comienzan con las siglas Frm, y los de tipo panel con las siglas Pnl.
- Los gestores de negocio del cliente comienzan con las siglas GestorNC y los del servidor con las siglas GestorNS.
- Las clases de la lógica de negocio de acceso a datos comienzan con las siglas GestorAD.
- Las interfaces publicadas en el servidor terminan con el prefijo Remoto.

Nomenclatura para los componentes de formularios:

- Los botones (JButton) comienzan con las siglas btn.
- Los campos de texto (JTextField) comienzan con las siglas tfd.
- Los campos de texto (Formateador) comienzan con las siglas frt.
- Para el componente JYearChooser con las siglas jyc.
- Para el componente JCalendar con la letra c.
- Las cajas de texto (JComboBox) comienzan con las siglas cmb.
- Los labels (JLabel) comienzan con las siglas lbl.

- Los cuadros de chequeo (JCheckBox) comienzan con las siglas chb.
- Los radiobutton (JRadioButton) comienzan con las siglas rbt.
- Las tablas (JTable) comienzan con las siglas tbl.

3.4 VALIDACIÓN DE LA SOLUCIÓN

En la validación de la solución propuesta se emplearon pruebas de software, además de métricas para las disciplinas de requerimientos y diseño. También se utilizó un procedimiento para comprobar el objetivo general. Todos los elementos anteriormente mencionados se abordarán a continuación.

3.4.1 PRUEBAS DE SOFTWARE

3.4.1.1 PRUEBAS DE CAJA BLANCA

El desarrollo de pruebas de caja blanca, utilizando pruebas unitarias como nivel de pruebas, se le aplicó a varios métodos de casos de uso críticos con el objetivo de validar el código. Se empleó la técnica del cálculo de camino básico.

Los pasos para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba para probar los caminos obtenidos.

Siguiendo dichos pasos se calcula la complejidad ciclomática del algoritmo o fragmento de código a analizar. En la figura 16 se enumeran las sentencias de código del procedimiento realizado sobre el método **onBtnTerminar()** que se encarga de insertar o modificar un local según sea el caso.
Luego se realiza el grafo de flujo asociado al código anterior quedando como se muestra en la figura 17.

```
public void onBtnTerminar()                    {

        if (modificando == false) { 1
            int codigo = Integer.parseInt(this.getFormulario().txtCodigo.getText()); 2
            String nombre = this.getFormulario().txtDescripcion.getText(); 2
            float superficieT = Float.parseFloat(this.getFormulario().txtSupT.getText()); 2
            float superficieD = Float.parseFloat(this.getFormulario().txtSupD.getText()); 2
            this.local = new Dlocal(); 2
            this.local.setCodigo(codigo); 2
            this.local.setNombre(nombre); 2
            this.local.setSuperficiedisponible(superficieD); 2
            this.local.setSuperficietotal(superficieT); 2
            if (ValidarCodigo() && ValidarSuperficie()) { 3
                GestorNCLocal.insertarLocal(local); 4
                GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia()
                        .getMensaje("SIGFE.guardado")); 4
                GestorMarcoTrabajo.getInstancia().mostrarFormularioExistente(GestorMarcoTrabajo.getInstancia()
                        .getPanelExistente(AccGestionarLocal.class.getName())); 4
            }5
        } else {
            local.setNombre(this.getFormulario().txtDescripcion.getText()); 6
            local.setSuperficietotal(Float.parseFloat(this.getFormulario().txtSupT.getText())); 6
            local.setSuperficiedisponible(Float.parseFloat(this.getFormulario().txtSupD.getText())); 6

            GestorNCLocal.actualizarLocal(local); 6
            GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia().getMensaje("SIGFE.modificado")); 6
            GestorMarcoTrabajo.getInstancia().mostrarFormularioExistente(GestorMarcoTrabajo.getInstancia()
                    .getPanelExistente(AccGestionarLocal.class.getName())); 6
        } 7
}8
```

Figura 16: Representación del algoritmo onBtnTerminar().

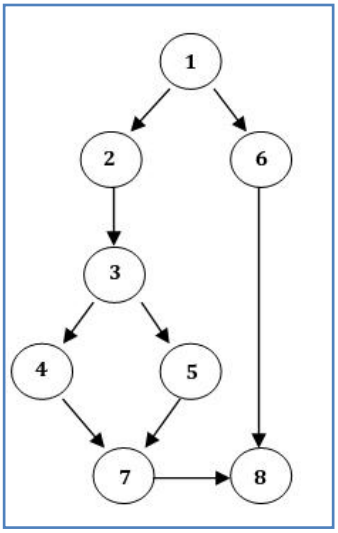


Figura 17: Grafo de flujo asociado al algoritmo onBtnTerminar()

Seguidamente se calcula de la complejidad ciclomática, mediante las tres fórmulas siguientes, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto:

1. **V (G)** = (A-N) +2 (Donde "**A**" la cantidad total de aristas y "**N**" la cantidad total de nodos.)
   V(G) = (9-8) +2

V (G) = 3

2. **V(G)** = P+1 (Siendo "**P**" la cantidad total de nodos predicados[15])
   V(G) = 2+1
   V(G) = 3
3. **V(G)** = R (Siendo "**R**" la cantidad total de regiones, se incluye el área exterior)
   V(G) = 3

Como puede apreciarse todas las fórmulas dan el mismo valor, concluyéndose que el algoritmo presentado tiene un complejidad ciclomática de 3, que da la visión de que existe el mismo número de caminos lógicos por donde recorrer el algoritmo. Es necesario entonces representar los caminos básicos por los que puede recorrer el flujo:

- **Camino 1:** 1-2-3-5-7-8
- **Camino 2:** 1-2-3-4-7-8
- **Camino 3:** 1-6-8

Tras extraer los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para consultar los casos de pruebas correspondientes ver **Anexo 5**.

Luego de aplicar los diferentes casos de pruebas, se logró evidenciar que el flujo de trabajo de las funciones es correcto, puesto que todos cumplen con las condiciones necesarias que se habían planteado.

3.4.1.2 PRUEBAS DE CAJA NEGRA

Con el objetivo de llevar a cabo las pruebas de caja negra para la validación técnica de la solución se confeccionaron casos de prueba a los distintos casos de uso presentes en el alcance del módulo. El el **Anexo 5** se pueden observar los casos de puebas diseñados para cada uno de los escenarios o flujos de los casos de uso del módulo, así como la descripción de las variables que intervienen.

Se realizaron un total de tres iteraciones para poder alcanzar los resultados satisfactorios desde el punto de vista funcional del módulo, atendiendo a la técnica de particiones

[15] **Nodos predicados**: Son los nodos de donde sale más de una arista.

equivalentes. Seguidamente se describen las no conformidades detectadas durante la ejecución de las pruebas de caja negra:

En la primera iteración se encontraron un total de 12 no conformidades (NC):

- Faltas ortográficas **5 NC**
- Errores en las interfaces **3 NC**
- Errores de validación **4 NC**

En la segunda iteración se encontraron un total de 4 no conformidades(NC):

- Faltas ortográficas **0 NC**
- Errores en las interfaces **2 NC**
- Errores de validación **2 NC**

En la tercera iteración no se encontraron no conformidades (NC).

A continuación los resultados del proceso:

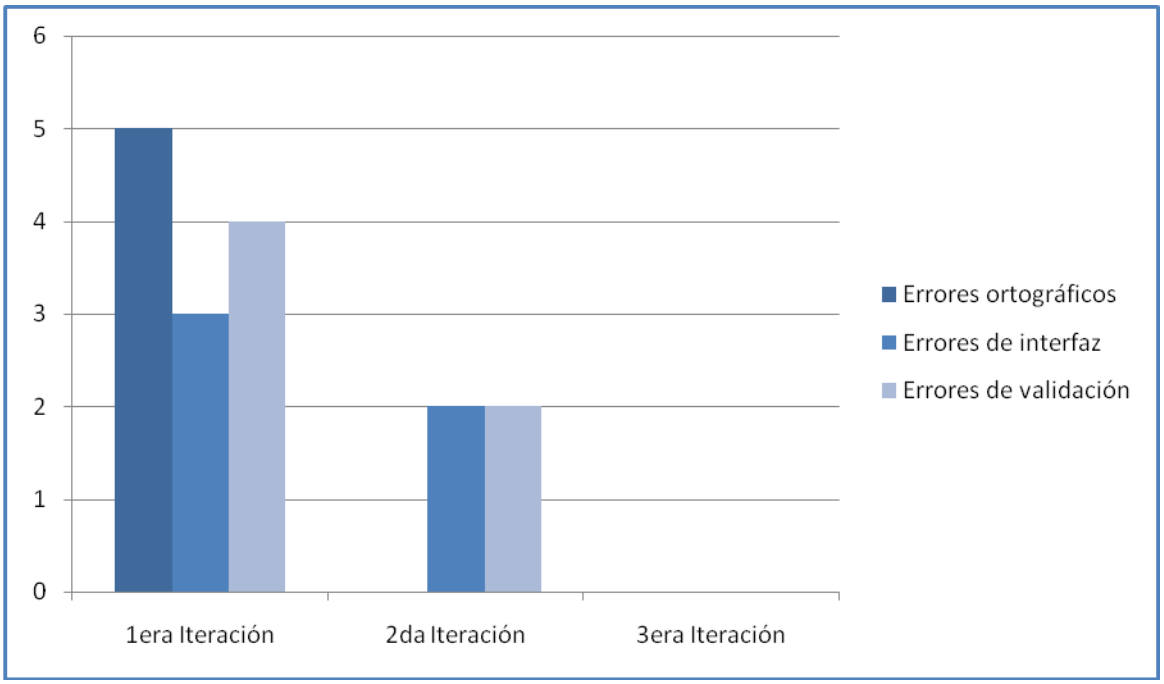


Figura 18: Resultados de las pruebas de caja negra

3.4.2 MÉTRICAS DE SOFTWARE

3.4.2.1 MÉTRICAS PARA VALIDAR REQUERIMIENTOS

Para la disciplina de requerimientos, se utilizaron las métricas de estabilidad de los requerimientos y especificidad de los requerimientos (ver **Anexo 1**). Al aplicarlas se obtuvo los siguientes resultados:

1. Estabilidad de los requerimientos:

   Se tiene un total de 43 requerimientos funcionales de los cuales 4 fueron cambiados (2 por actualización y 2 por eliminación)

   ETR = ((RT-RC)/ RT) *100

   ETR =((39)/43) * 100

   ETR = 90.69

   Se obtiene un valor de 90.69 lo que señala que no han habido cambios importantes en los requerimientos, considerándolos estables para realizarles análisis y diseño.

2. Especificidad de los requerimientos:

   Se cuenta con 43 requerimientos funcionales y 15 no funcionales.

   $Q1 = nu_i / n_i$ $\quad n_i = n_f + n_{nf}$

   Q1 = 58/58 $\quad n_i = 43 + 15$

   Q1 = 1

   Se consigue un valor de 1, lo que demuestra que la especificación no presenta ambigüedades.

3.4.2.2 MÉTRICAS PARA VALIDAR EL DISEÑO

Como se definió en el capítulo 1, se decide utilizar las siguientes métricas para validar diseño:

- Tamaño operacional de clase (**TOC**): se basa en el número de métodos asignados a una clase.
- Relaciones entre clases (**RC**): se basa en el número de relaciones de uso de una clase con otra.

Para comenzar la validación de las clases del diseño, se emplea la métrica TOC a una muestra representativa y crítica de los casos de usos de la solución. Es importante señalar que los parámetros responsabilidad y la complejidad son inversamente proporcionales a la reutilización, es decir a mayor responsabilidad y complejidad de implementación, menor será la reutilización de la clase y viceversa. De acuerdo con los resultados obtenidos, se puede llegar a la conclusión de que las clases del diseño no se encuentran sobrecargadas en cuanto a responsabilidad, mostrando un nivel de complejidad bajo, lo que favorece la reutilización.

RC se considera otra de las métricas empleadas. Luego de aplicar la misma se obtiene como resultado que las clases del diseño promueven el bajo acoplamiento, la compejidad de mantenimiento y cantidad de pruebas no es alta y en consecuencia, el grado de reutilización es alto.

La matriz de cubrimiento o matriz de inferencia permite conocer si el resultado obtenido de la relación atributo/métricas para cada componente es positivo o negativo. Si los resultados son positivos obtendrá un valor de 1, si son negativos de 0y si no existe relación alguna se tomará como nula, otorgándole un valor de -1(ver Tabla 5).

Luego de completar dicha relación se realiza un cálculo promediando la sumatoria de los valores obtenidos de un atributo por cada métrica evaluada, y la división de dicha sumatoria por la cantidad de métricas evaluadas (solo se promedian las que arrojan un resultado, las nulas no).

Este valor es el que va a tener el atributo dentro de una tabla que medirá si los atributos fueron buenos, regulares o malos. Para darle un valor cualitativo a la evaluacion se utiliza la siguiente escala: Malo (valor menor de 0.4), Regular (valor mayor de 0.4 y menor e igual a 0.7), Bueno (valor mayor que 0.7).

Atributos/Métricas	TOC	RC	Promedio
Responsabilidad	1	-1	1
Complejidad de implementación	1	-1	1
Reutilización	1	1	1

Complejidad del mantenimiento	-1	1	1
Cantidad de pruebas	-1	1	1
Acoplamiento	-1	1	1

Tabla 5: Matriz de cubrimiento relativa a la solución.

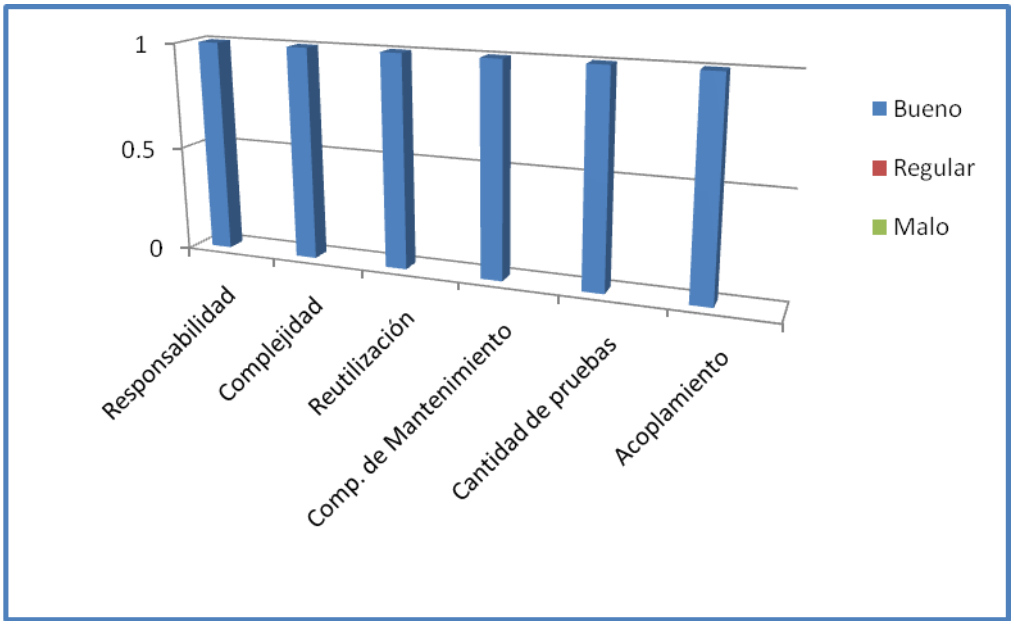


Figura 19: Resultados de los atributos evaluados en las métricas

3.4.2.3 MÉTRICAS PARA VALIDAR LA CARACTERÍSTICA FUNCIONALIDAD

Para medir la característica de calidad funcionalidad, se decide seguir el procedimiento descrito en la tesis de diploma "*Procedimiento para evaluar la característica de funcionalidad de componentes de software*"(Estrada Peña, 2012).

En la misma se establecen los siguientes pasos lógicos para lograr la medición:

1. **Establecer nivel de relevancia**: El evaluador otorga a cada sub-característica un nivel de importancia sobre el sistema que puede ser: alto, medio o nulo. Si es nulo la sub-característica no será medible por lo que no se evalúa (ver Tabla 8 del Anexo 4).
2. **Recopilación de datos***:* Se colocan los datos de las variables implicadas en las fórmulas de medición de las métricas (ver Tabla 9 del Anexo 4).
3. **Calcular métricas**: Se calculan las métricas basándose en las fórmulas y los datos introducidos. (ver Tabla 10 del Anexo 4)

4. **Evaluación por cada métrica**: Una vez el evaluador posea los resultados de las métricas pasará a evaluarlas cualitativamente, mediante la tabla de escala. (ver Tabla 11 del Anexo 4). Los resultados se ubican en la tabla medición de la calidad funcional para establecer la evaluación alcanzada en cada métrica. (ver Tabla 12 del Anexo 4)
5. **Evaluación por sub-característica***:* El evaluador dará una medición cuantitativa y cualitativa por sub-característica, calculando el porciento de sus métricas y evaluándolo de bien, regular o mal según la tabla de escala. Dicho cálculo de porciento se realiza mediante la fórmula y las variables que se muestran en la tabla 13 del Anexo 4.
6. **Evaluación de la funcionalidad***:* Se muestra la evaluación final de la funcionalidad, calculando un promedio entre todas las sub-características, que luego se lleva a la tabla de escala para darle una evaluación cualitativa.

3.4.2.4 APLICACIÓN DEL PROCEDIMIENTO PARA EVALUAR LA CARACTERÍSTICA DE FUNCIONALIDAD

Este procedimiento se aplica tanto al SIGFE como al SAEFE, con el objetivo de obtener una comparación concreta de ambas herramientas. Siguiendo los pasos mencionados en el acápite anterior (ver acápite 3.4.2.3) el procedimiento se desarrolla de la siguiente manera:

1. **Proceso de medicion:**

Nivel de relevancia: Para ambas herramientas los niveles de relevancias son similares.

- Idoneidad: Alta
- Precisión: Media
- Interoperabilidad: Nula
- Seguridad: Alta

2. **Ejecución de la medicion:**

SAEFE

- Idoneidad

a) Adecuación funcional

X = 1-17/30

X = 1 – 6/43

X = 0.44 Evaluación: Mal

Se evaluaron 30 funcionalidades, de las cuales 17 presentaron problemas.

Métrica general de la sub-característica Idoneidad:

Psub = (0.44) *100

PSub = 44 %

- Precisión

a) Precisión

X = 1-(A/T)

X =1 - (2/3)

X = 0.33 Evaluación: Mal

En un tiempo de operación de 3 segundos el módulo mostro imprecisión en 2 de los reportes revisados.

Nota: Se ajusta la fórmula a la tabla de escala.

Métrica general de la sub-característica Presición:

Psub = (0.33) *100

PSub = 33 %

- Seguridad

a) Capacidad de revisión de cuentas de usuario

X = A/B

X =0/10

X = 0 Evaluación: Mal

Se realizaron 10 accesos al sistema, de los cuales ninguno resultó registrado como traza en la base de datos histórica de acceso.

b) Capacidad de control de acceso

X = A/B

X =4/4

X = 1 Evaluación: Bien

Se realizaron 4 intentos con el objetivo de acceder sin permiso al sistema, como son: usuario y contraseña aleatorios, usuario real y contraseña incorrecta y

usuario y contraseña sin completar. Todos los intentos fallaron sin brindar información sensible.

Métrica general de la sub-característica Seguridad:

Psub = (0 + 1)/2 *100

PSub = 1/2 * 100

PSub =50 %

SIGFE

- Idoneidad

b) Adecuación funcional

X = 1-A/B

X = 1 – 6/43

X = 0.86 Evaluación: Bien

Se evaluaron 43 funcionalidades, de las cuales 6 presentaron problemas.

Métrica general de la sub-característica Idoneidad:

Psub = (0.86) *100

PSub = 86 %

- Precisión

b) Precisión

X = 1-(A/T)

X =1- (0/3)

X = 1 Evaluación: Bien

En un tiempo de operación de 3 segundos el módulo fue capaz de generar todos los reportes con la precisión requerida.

Nota: Se ajusta la fórmula a la tabla de escala.

Métrica general de la sub-característica Precisión:

Psub = (1) *100

PSub = 100 %

- Seguridad

c) Capacidad de revisión de cuentas de usuario

X = A/B

X =9/10

X = 0.9 Evaluación: Bien

Se realizaron 10 accesos al sistema, de los cuales 9 resultaron registrados como trazas en la base de datos histórica de acceso.

d) Capacidad de control de acceso

X = A/B

X =3/3

X = 1 Evaluación: Bien

Se realizaron 3 intentos con el objetivo de acceder sin permiso al sistema, como son: usuario y contraseña aleatorios, usuario real y contraseña incorrecta y usuario y contraseña sin completar. Todos los intentos fallaron sin brindar información sensible.

Métrica general de la sub-característica Seguridad:

Psub = (0.9+1)/2 *100

PSub = 95 %

3. Conclusión de la evaluación

SAEFE

Evaluación por subcaracterística:

PFun= (44 + 33 + 50)/ 3

PFun = 42.33 %

SIGFE

Evaluación por subcaracterística:

PFun= (86 + 100 + 95) / 3

PFun = 93.66 %

Como se puede evidenciar en los resultados obtenidos la herramienta SAEFE presenta un por ciento muy bajo de funcionalidad con un 42.33. En cambio SIGFE, presenta altos valores para este característica de calidad, lo cual permite comprobar el cumplimiento del objetivo general planteado.

A continuación se muestra un gráfico con resultados generales del procedimiento:

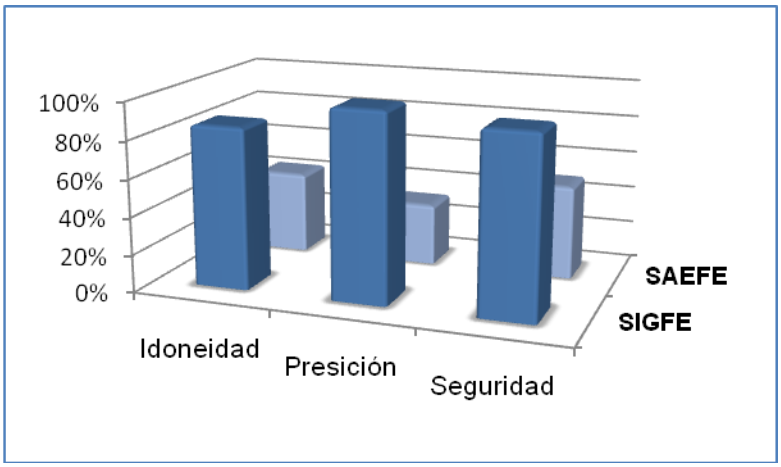


Figura 20: Resúmen comparativo de los sistemas SAEFE y SIGFE atendiendo a parámetros para evaluar Funcionalidad

3.5 CONCLUSIONES DEL CAPÍTULO

El desarrollo del presente capítulo permitió abordar los temas relacionados con la implementación y validación de la solución propuesta, generándose en el proceso los diagramas de componentes y de despliegue competentes al módulo, además del diseño de casos de prueba, para desarrollar pruebas de caja blanca y de caja negra. alcanzándose resultados satisfactorios en la corrección y calidad de las funcionalidades del módulo. Así mismo, la propuesta de un procedimiento para evaluar la característica de calidad funcionalidad, permitió la validación del cumplimiento del objetivo general de la investigación.

CONCLUSIONES GENERALES

Con la investigación realizada se logró el desarrollo de un Módulo de Administración y Reportes para el Sistema de Gestión de Ferias y Exposiciones de la Cámara de Comercio de la República de Cuba que perseguía resolver los problemas presentados en materia de configuración, garantía de la seguridad de la información, y generación de reportes. Los objetivos propuestos fueron cumplidos con satisfacción, generando cada uno de ellos los siguientes resultados:

- La elaboración del marco teórico de la investigación permitió identificar la necesidad de crear un Módulo de Administración y Reportes para el Sistema de Gestión de Ferias y Exposiciones de la Cámara de Comercio de la República de Cuba, determinando qué herramientas, metodologías y tecnologías eran factibles a utilizar en el desarrollo del mismo.
- La captura de requerimientos competentes al módulo permitió al equipo de desarrollo obtener los elementos necesarios para cumplir con las exigencias del cliente de manera satisfactoria, centrándose en las funcionalidades críticas a desarrollar y dotando al producto de características y propiedades para su buen funcionamiento en la institución.
- Los artefactos generados durante el flujo de trabajo de análisis y diseño sirvieron para conformar la primera visión de la implementación del sistema, defendiéndose durante la misma una estrategia para lograr una construcción flexible y robusta a través de la utilización de una arquitectura afín, patrones de diseño bien establecidos y estándares de codificación.
- Al validar la solución, utilizando métricas para requerimientos y diseño, la realización de pruebas de software, además de la aplicación de un procedimiento para evaluar la característica de funcionalidad de componentes de software, se logró alcanzar altos niveles de calidad en la solución propuesta y se comprobó el cumplimiento del objetivo general trazado.

RECOMENDACIONES

Después del desarrollo del Módulo de Administración y Reportes para el Sistema de Gestión de Ferias y Exposiciones de la Cámara de Comercio de la República de Cuba, teniendo en cuenta la experiencia y los resultados obtenidos, se proponen las siguientes recomendaciones:

- Emplear el sistema en la Cámara de Comercio de la República de Cuba y en otras empresas o instituciones que se dediquen a la gestión de ferias y exposiciones comerciales.
- Trabajar en nuevas versiones del producto para contribuir a su mejora, partiendo de nuevos requerimientos del cliente.

REFERENCIAS BIBLIOGRÁFICAS

- **Antonio Aliaga Ibarra, Marcos Agustin Miani Flores. 2008.** *PostgreSQL.* 2008.
- **ActiveNetwork. 2012.** *RegOnline.* [En línea] 2012. [Citado el: 3 de Abril de 2013.] http://www.regonline.com.es.
- **Amiando. 2013.** *Funciones-Amiando*. [En línea] 2013. [Citado el: 3 de Abril de 2013.] http://es.amiando.com/features.html.
- **Belmonte Fernández, Oscar. 2005.** Introducción al lenguaje de programación Java. [En línea] 2005. [Citado el: 24 de Enero de 2013.] http://www3.uji.es/~belfern/pdidoc/IX26/Documentos/introJava.pdf.
- **Booch, Grady, Rumbaugh, Jim y Jacobson, Ivar. 1999.** *El Lenguaje Unificado de Modelado.* s.l. : Addison Wesley, 1999.
- **Buschmann, F y otros. 1996.** *Pattern – Oriented Software Architecture. A System of Patterns.* s.l. : John Wiley & Song, 1996.
- **Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004.** *Arquitecturas de Software. Guia de Estudio.* 2004.
- **Cámara de Comercio de la República de Cuba. 2009.** Portal de la Cámara de Comercio de la República de Cuba. [En línea] 2009. [Citado el: 15 de Octubre de 2012.] http://www.Cámaracuba.cu/index.php?option=com_content&view=article&id=44&Itemid=54.
- **Carrera, R. 2002.** *Apuntes de la materia de sistemas integrales de información.*
- **Date, C. J. 2010.** *Introducción a los Sistemas de Bases de Datos.* México : Pearson Education, 2010.
- **Drake, José M. y López, Patricia. 2009.** [En línea] 2009. [Citado el: 14 de Noviembre de 2012.] http://www.ctr.unican.es/Asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf.
- **Eclipse Foundation. 2012***. EclipseLink/FAQ/JPA.* [En línea] 2012. [Citado el: 31 de Enero de 2013.] http://wiki.eclipse.org/EclipseLink/FAQ/JPA.
- **—. 2012.** *Understanding EclipseLink Release 2.4*. [En línea] 2012. [Citado el: 31 de Enero de 2013.] http://www.eclipse.org/org/documents/epl-v10.php.

- **Estrada Peña, Yuneisis. 2012.** *Procedimiento para evaluar la caracterisitaca de funcionalidad de componentes de software*. Habana : s.n., 2012.
- **Fadraga Artiles, Lissuan y Lizama Mué, Yadira. 2012.** *Propuesta de una arquitectura de referencia para el desarrollo de aplicaciones empresariales.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2012.
- **Fernández Díaz, Suammy y Ramírez Morales, Yoslaine de la Caridad. 2009.** *Propuesta de Métricas para medir Calidad en Portales WAP.* Habana : s.n., 2009.
- **Garfinkel, S., & Spafford, G. 1996.** Pra*ctical Unix and Internet Security***.** O'Reilly and Associates, Inc.
- **GESTEVET. 2012.** *GestEvent***.** [En línea] 2012. [Citado el: 3 de Abril de 2013.] http://www.gestevent.es.
- **González Barahona, Jesús M., Seoane Pascual, Joaquin y Robles, Gregorio. 2003-2007.** *Introducción al software libre*. [En línea] 2003-2007. [Citado el: 14 de Enero de 2013.] http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/book1.html.
- **Grupo Alarcos.***Alarcos*. [En línea] [Citado el: 17 de Octubre de 2012.] http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema04.pdf.
- **Hernández, A. V. 2009.** *Estrategia de Evaluación de Software, Calisoft. La Habana.*
- **ISO. 2005.** *Ingeniería de software – Calidad del producto-Parte 1: Modelo de la calidad*. Habana : s.n., 2005.
- **Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000.** *El proceso unificado de desarrollo de software.* s.l. : Addison Wesley, 2000.
- **Keith, Mike y Schincariol, Merrick. 2009.** *Pro JPA 2 Mastering the Java™ Persistence API.* s.l. : Apress, 2009.
- **Koontz, H., & Weinrichy, H. (1998).** *Elementos de la Administración***.** México**.**
- **Kruchten, Philippe. 2000.** *The Rational Unified Process An Introduction.* s.l. : Addison Wesley, 2000.
- **Larman, Craig. 1999.** *UML y Patrones. Introducción al Diseño Orientado a Objetos.* México : PRENTICE HALL, 1999.
- **Microsoft Corporation. 2009.** Microsoft Application Architecture Guide. [En línea] 2009. [Citado el: 28 de Noviembre de 2012.] http://msdn.microsoft.com/en-us/library/ff650706.aspx.
- **Murdick, R. 1998.** *Sistemas de Información administrativa***.** México**.**

- **Nagel, William. 2005.** *Subversion Version Control. Using The Subversion Version Control System in Development Projects.* Massachusetts : Pearson Education, 2005.
- **Oracle Corporation NetBeans. 2010.** NetBeans IDE 6.9 Release Information. [En línea] 2010. [Citado el: 15 de Enero de 2013.] http://netbeans.org/community/releases/67/.
- **Oracle Corporation. 2013.** The Java EE 6 Tutorial. [En línea] 2013. [Citado el: 4 de Febrero de 2013.] http://docs.oracle.com/javaee/6/tutorial/doc/p1.html.
- **PostrgreSQL Fundation.** PostgreSQL. [En línea] [Citado el: 13 de Noviembre de 2012.] http://www.postgresql.org/about/.
- **Pressman, Roger S. 2002.** *Ingeniería de Software, un enfoque práctico.* s.l. : Mc Graw Hill, 2002.
- **Pruebas de software. 2005.** Gestión de Calidad y Pruebas de Software [En linea] [Citado el: 5 de Mayo de 2013]: http://pruebasdesoftware.com/laspruebasdesoftware.htm
- **Reka, Rahman, Debu, Panda y Lane, Derek. 2007.** *EJB in Action.* s.l. : Greenwich : Manning, 2007.
- **Ruiz Muñoz, David.** *Eumed.net. Manual de Estadística* [En línea] [Citado el: 29 de Mayo de 2013] http://www.eumed.net/cursecon/libreria/drm/index.htm
- **Sierra, Kathy y Bates, Bert. 2003.** *Head First EJB.* s.l. : O'Reilly Media, 2003.
- **Sparx Systems Pty Ltd. 2000-2007.** Sparx System. [En línea] 2000-2007 . [Citado el: 5 de Mayo de 2013.]http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html**.**
- **Stallings, William. 1997.** *Sistemas Operativos.* Madrid : PRENTICE HALL, 1997.
- **Subrahmanyam, Allamaraju y otros. 2002.** *Programación Java Server con J2EE Edition.* s.l. : Anaya Multimedia, 2002.
- **Sun Microsystems.** *GlassFish*. [En línea] [Citado el: 14 de Noviembre de 2012.] https://glassfish.dev.java.net.
- **Swenson, Erik. 2002.** *Reports made easy with Jasper Reports*. [En línea] 2002. [Citado el: 23 de Noviembre de 2012.] http://www.javaworld.com..
- **Szyperski, C. 1998.** *Component Software. Beyond Object-Oriented Programming.* 1998.

- **Toffoli, Giulio. 2007.** *The Definitive Guide to IReport.* s.l. : Apress, 2007.
- **USI. 2013.** *Ungerboeck Software International***.** [En línea] 2013. http://ungerboeck.com.
- **Zukowski, Jonh. 2003.** *Programación JAVA 2 JSE 1.4.* [ed.] Ignacio Luca de Tena. Madrid: ANAYA Multimedia, 2003.