

Universidad de las Ciencias Informáticas

Facultad 3



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

**Desarrollo de los módulos Convocatoria y Contratación para
el Sistema de Gestión de Ferias y Exposiciones de la Cámara
de Comercio de la República de Cuba**

Autores:

**Juan Carlos Mejias Cruz
Sandy Suárez Jiménez**

Tutor:

Ing. Lisett De Armas Hernández

Asesor:

MSc. Jessie Castell González

Asesor Metodológico:

Lic. Raynel Batista Téllez

La Habana, Junio 2013.



*"... No se trata solo de la informática para comunicarse, sino para saber,
aprender, enseñar, ayudar, compartir."*

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Juan Carlos Mejias Cruz.

Sandy Suárez Jiménez.

Firma del Autor

Firma del Autor

Ing. Lisett De Armas Hernández.

Firma del Tutor

Resumen

En la actualidad los sistemas de gestión de ferias y exposiciones resultan de gran utilidad para las instituciones que organizan estos eventos. En Cuba, la entidad rectora de este proceso es la Cámara de Comercio, contando con un sistema que no satisface sus necesidades, pues existen dificultades con la organización, ejecución y seguimiento de los eventos que en ella se realizan. La presente investigación pretende desarrollar los módulos Convocatoria y Contratación que requiere el Sistema de Gestión de Ferias y Exposiciones, de manera tal que contribuyan a su funcionalidad y facilidad de uso.

Para lograr el objetivo propuesto se hace un estudio de las tendencias actuales en el desarrollo de este tipo de aplicaciones. Se realiza un análisis y selección de las tecnologías adecuadas para el desarrollo de los módulos, se identifican las funcionalidades que deben cumplir y se realiza el diseño de los mismos. A partir de los artefactos obtenidos se procede a su implementación y validación. Una vez implantado el sistema se espera que los funcionarios de la institución cuenten con una herramienta, que facilite una adecuada gestión de los certámenes nacionales e internacionales.

Palabras claves: eventos, exposiciones, ferias, gestión, software.

Índice

Introducción	1
Capítulo 1. Fundamentación Teórica	6
1.1 Introducción.....	6
1.2 Conceptos fundamentales	6
1.3 Estudio del estado del arte	7
1.4 Análisis de las metodologías de desarrollo	10
1.4.1 Proceso Unificado de Rational	11
1.5 Arquitectura de software.....	12
1.5.1 Arquitectura cliente/servidor.....	13
1.5.2 Arquitectura en capas	13
1.5.3 Arquitectura basada en componentes	14
1.6 Herramientas y tecnologías	14
1.6.1 Lenguaje de modelado	14
1.6.2 Herramienta CASE	15
1.6.3 Notación para el modelado de procesos de negocio	16
1.6.4 Análisis de los lenguajes de programación	16
1.6.5 Swing y SwingX.....	18
1.6.6 Plataforma de desarrollo	19
1.6.7 Entorno de Desarrollo Integrado	20
1.6.8 Java Persistence API	21
1.6.9 Análisis de los Sistemas Gestores de Bases de Datos.....	22
1.6.10 Tecnología EJB	23
1.6.11 Sistema de control de versiones	25
1.7 Métricas y técnicas de pruebas	25
1.8 Conclusiones del capítulo	27
Capítulo 2: Descripción de la propuesta de solución.....	28
2.1 Introducción.....	28
2.2 Propuesta de solución	28
2.3 Arquitectura del sistema	28
2.4 Patrones de diseño	32
2.5 Modelo de negocio.....	34
2.5.1 Descripción de procesos de negocio.....	34
2.5.2 Modelo de proceso de negocio	35
2.6 Especificación de los requisitos de software.....	36
2.6.1 Requerimientos funcionales	37

2.6.2	Requerimientos no funcionales	38
2.7	Definición de casos de uso del sistema.....	41
2.7.1	Definición de los actores	41
2.7.2	Listado de casos de uso.....	41
2.7.3	Diagrama de casos de uso del sistema.....	41
2.7.4	Descripción textual	42
2.8	Diagrama de clases de análisis.....	42
2.9	Diagrama de secuencia.....	43
2.10	Diagrama de clases de diseño	44
2.11	Estándares del diseño	45
2.12	Diseño de la base de datos	45
2.12.1	Modelo de datos	46
2.12.2	Descripción de las tablas	47
2.13	Validación del diseño del sistema	47
2.14	Conclusiones del capítulo	50
Capítulo 3: Implementación y prueba.		51
3.1	Introducción.....	51
3.2	Modelo de implementación.....	51
3.2.1	Diagrama de componentes.....	51
3.2.2	Diagrama de despliegue	52
3.3	Estándares de codificación.....	53
3.3.1	Nomenclatura para los paquetes, métodos y clases.....	53
3.3.2	Nomenclatura para los componentes de formularios	54
3.4	Validación de la solución	54
3.5	Conclusiones del capítulo	66
Conclusiones		67
Recomendaciones		68
Bibliografía		69
Anexos		73

Índice de Tablas

<i>Tabla 1. Sub-características de calidad.</i>	7
<i>Tabla 2. Descripción del proceso de negocio Convocatoria.</i>	34
<i>Tabla 3. Descripción del proceso de negocio Contratación.</i>	35
<i>Tabla 4. Requisitos funcionales más importantes.</i>	37
<i>Tabla 5. Actores del sistema.</i>	41
<i>Tabla 6. Caso de uso Gestionar Solicitud de recursos.</i>	41
<i>Tabla 7. Tablas de la base de datos asociadas a la sección Solicitud.</i>	47
<i>Tabla 8. Aplicación la métrica TOC.</i>	48
<i>Tabla 9. Aplicación de la métrica RC.</i>	49
<i>Tabla 10. Resultados de los atributos de calidad de la métrica RC.</i>	49
<i>Tabla 11. Caso de prueba: Escenario Registrar Solicitud.</i>	59
<i>Tabla 12. Descripción de las variables.</i>	59
<i>Tabla 13. Evaluación de sub-características por usuarios.</i>	64
<i>Tabla 14. Evaluación final de cada sub-característica.</i>	65
<i>Tabla 15. Interpretación de los resultados de la métrica TOC.</i>	73
<i>Tabla 16. Medidas o umbrales de la métrica TOC.</i>	73
<i>Tabla 17. Criterios de evaluación de la métrica RC.</i>	74
<i>Tabla 18. Requisitos de los Módulos Convocatoria y Contratación.</i>	80
<i>Tabla 19. Medición de la calidad funcional para establecer nivel de relevancia.</i>	81
<i>Tabla 20. Medición de la calidad funcional para recopilar datos.</i>	81
<i>Tabla 21. Medición de la calidad funcional para cálculo de métricas.</i>	82
<i>Tabla 22. Escala para la evaluación.</i>	82
<i>Tabla 23. Medición de la calidad funcional para establecer la evaluación en cada métrica.</i>	82
<i>Tabla 24. Cálculo del porcentaje por sub-característica.</i>	83
<i>Tabla 25. Aplicación de las métricas por usuarios.</i>	83
<i>Tabla 26. Escala para la Evaluación.</i>	83
<i>Tabla 27. Cálculo de los resultados generales.</i>	83
<i>Tabla 28. Obtención del Nivel de Usabilidad existente en el software.</i>	84
<i>Tabla 29. Listado de Casos de Uso.</i>	87

Índice de Figuras

<i>Figura 1. Fases de RUP.</i>	12
<i>Figura 2. Distribución de las capas de la arquitectura.</i>	29
<i>Figura 3. Vista integrada de la arquitectura de SIGFE.</i>	32
<i>Figura 4. Modelo de proceso de negocio de la Convocatoria.</i>	35
<i>Figura 5. Modelo de proceso de negocio de la Contratación.</i>	36
<i>Figura 6. Modelo de proceso de negocio: Subproceso Negociar Contrato.</i>	36
<i>Figura 7. Diagrama de caso de uso: Módulos Convocatoria y Contratación.</i>	42
<i>Figura 8. Diagrama de clases del análisis Gestionar Solicitud de recursos.</i>	43
<i>Figura 9. Diagrama de secuencia Registrar Solicitud de recursos.</i>	43
<i>Figura 10. Diagrama de clases de diseño Gestionar Solicitud de recursos.</i>	44
<i>Figura 11. Submodelo físico de la base de datos: Sección Solicitud.</i>	46
<i>Figura 12. Resultados de la métrica TOC aplicada al diseño de la solución.</i>	48
<i>Figura 13. Diagrama de componentes de los módulos.</i>	52
<i>Figura 14. Diagrama de despliegue del sistema.</i>	53
<i>Figura 15. Código de la funcionalidad Registrar Evento.</i>	56
<i>Figura 16. Grafo de flujo asociado al método RegistrarEvento.</i>	57
<i>Figura 17. Funcionalidad de los módulos Convocatoria y Contratación.</i>	63
<i>Figura 18. Usabilidad de los módulos Convocatoria y Contratación.</i>	66
<i>Figura 19. Diagrama de Secuencia Modificar Solicitud de recursos.</i>	81
<i>Figura 20. Diagrama de Secuencia Eliminar Solicitud de recursos.</i>	81
<i>Figura 21. Diagrama de componentes de la capa de Presentación.</i>	89
<i>Figura 22. Diagrama de componentes de la capa de Lógica de Negocio.</i>	90
<i>Figura 23. Diagrama de componentes de la capa Acceso a Datos.</i>	90

Introducción

El surgimiento y evolución de la ciencia y la técnica han provocado el desarrollo de una revolución científico-técnica que está soportada por las nuevas Tecnologías de la Información y las Comunicaciones (TIC). Se están produciendo grandes cambios en todas las esferas de la vida del hombre: social, político, cultural. El modo de actuar, pensar y de asumir el mundo se transforma y adopta nuevos entornos y puntos de vistas.

El sistema empresarial se ve influenciado directamente con este creciente avance, ya que cada vez se demandan aplicaciones más rápidas, ligeras y robustas que respondan a los procesos que se llevan a cabo, permitiendo lograr un grado de eficiencia elevado al emplearlas de manera correcta. Esto se debe a que estas tecnologías pueden llegar a proporcionar recursos estratégicos, no por la tecnología en sí misma que está disponible ampliamente, sino por lo fácil que es personalizarla y construir con ella sistemas propios que se ajusten a las necesidades crecientes de los usuarios.

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

Tiene como misión promover el desarrollo de la empresa cubana asociada, en beneficio de la economía nacional y para ello también estimula la organización de ferias, eventos y exposiciones, garantizando un crecimiento de los intercambios y ampliando las oportunidades de negocio. En esta institución existen dificultades con la organización, ejecución y seguimiento de los eventos que en ella se realizan, específicamente relacionados con la recepción y procesamiento de la información de los mismos. Los problemas de mayor relevancia se sustentan en la difícil identificación de intereses comunes de la información que se recibe de las empresas que participan en los eventos, perdiendo así oportunidades de negocio.

Para afrontar esta situación la Cámara de Comercio actualmente cuenta con un sistema de gestión llamado Sistema Automatizado para Eventos, Ferias y Exposiciones (SAEFE), programado en Delphi, que no satisface completamente las necesidades de la institución. Dicho sistema no permite ejecutar correctamente el proceso de asignación de recursos para que este sea configurable, ya que no es posible establecer la estructura de los recintos para los eventos, lo cual dificulta su asignación adecuada.

Según informe del Proyecto Técnico este *sistema no es capaz de exportar información en un formato entendible para el Sistema de Gestión Contable (EXACT) del que dispone la Cámara de Comercio, lo cual requiere la generación manual de las facturas trayendo consigo lentitud en el proceso y aumentando la probabilidad de ocurrencia de errores humanos. Además no permite calcular el monto del cobro a los participantes en los eventos de manera automática atendiendo a las reglas definidas al respecto, lo que obstaculiza la determinación de los costos de los eventos y afecta el proceso de planificación del presupuesto para estos* (Vega Miniet, 2012).

Al comprobar la usabilidad del SAEFE con la lista de chequeo definida por el Centro Nacional de Calidad de Software (CALISOFT) se pudo constatar que el mismo utiliza títulos descriptivos y distintivos en las pantallas, tablas e imágenes y el nombre de los botones de un formulario es adecuado. Además brinda la posibilidad de cancelar las operaciones en progreso. Sin embargo las interfaces no presentan un diseño atractivo, pues no son visualmente agradables ni amigables en su uso y el sistema carece de una ayuda que facilite su aprendizaje. Los iconos que aparecen por ser pequeños no sugieren de una manera clara las acciones que representan. Algunos carecen de funcionalidad y otros conllevan a errores en la ejecución de las tareas, trayendo consigo un incremento de los tiempos en la realización de las mismas.

Las interfaces no muestran el contenido en su totalidad, es el usuario quien debe modificar su tamaño para poder visualizar todas las acciones disponibles. No existe un diseño de los diálogos para el tratamiento de los errores por parte de los desarrolladores, careciendo de mensajes de error que den la posibilidad al usuario de retraerse antes de que se realice la acción y se comprometan los datos. Solo se validan excepciones básicas y es algo propio del lenguaje. Por todo lo antes expuesto se puede concluir que el SAEFE presenta una baja facilidad de uso.

A partir de la problemática descrita anteriormente se define como problema de la investigación: **El modo en que está concebido el SAEFE en torno a la gestión de**

ferias y exposiciones que organiza la Cámara de Comercio, no garantiza la usabilidad y funcionalidad del sistema.

Para dar solución al problema, se define como objeto de estudio: **los procesos de gestión de ferias y exposiciones** teniéndose como objetivo general de la investigación: **desarrollar los módulos Convocatoria y Contratación que requiere el SIGFE¹ para la gestión de ferias y exposiciones de la Cámara de Comercio, de manera tal que contribuyan a su funcionalidad y facilidad de uso.**

La investigación se concentra en **los procesos de convocatoria y contratación en la gestión de ferias y exposiciones**, suponiendo que si el SIGFE contara con los **módulos Convocatoria y Contratación** teniendo en cuenta la usabilidad y funcionalidad como características de calidad, contribuiría a la organización, ejecución y seguimiento de las ferias y exposiciones que realiza la Cámara de Comercio.

En función de cumplir con el objetivo general se trazaron los objetivos específicos siguientes:

- ✓ Fundamentar teóricamente la investigación sobre los procesos convocatoria y contratación en la gestión de ferias y exposiciones.
- ✓ Realizar la captura de requisitos de los procesos convocatoria y contratación.
- ✓ Diseñar e implementar los requisitos obtenidos.
- ✓ Validar la solución propuesta haciendo uso de métricas y pruebas de software.

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

1. Revisión bibliográfica para la elaboración del marco teórico de la investigación y obtener antecedentes sobre el tema propuesto, como es la existencia de aplicaciones semejantes en otros países y las herramientas más utilizadas en este sentido en el mundo.
2. Selección de las herramientas a utilizar en el desarrollo del sistema informático.
3. Identificación de las funcionalidades requeridas para desarrollar los módulos, detallando los requisitos funcionales y no funcionales detectados mediante la

¹ Por determinación de las partes, se decide realizar un nuevo sistema informático denominado Sistema de Gestión de Ferias y Eventos (SIGFE), asumiendo su realización la Universidad de las Ciencias Informáticas (UCI), en específico el Centro de Gobierno Electrónico (CEGEL) de la Facultad 3.

- ingeniería de requisitos.
4. Elaboración de los artefactos necesarios según la metodología de desarrollo de software seleccionada.
 5. Implementación de los módulos.
 6. Realización de pruebas a los módulos.
 7. Validación de los módulos de forma tal que se demuestre el cumplimiento del objetivo general.

Para la realización de las tareas antes mencionadas se emplearon métodos científicos. Se utilizaron los métodos de investigación cuantitativa y cualitativa teniendo en cuenta las referencias de Reyes (s.a.), Hernández Sampieri (2003) y Rodríguez y García (2004).

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

Se emplearon como métodos teóricos:

El método **Analítico – Sintético** sintetiza los elementos más importantes que se relacionan con los sistemas de gestión de ferias y exposiciones a partir de tendencias y documentos relacionados con el tema, expresando de manera resumida la posición del investigador.

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

El método **Hipotético – Deductivo** para el análisis y la definición de la hipótesis que será verificada o probada con el desarrollo de la investigación, arribando a conclusiones particulares.

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

Los **métodos empíricos** revelan, describen y explican las características y relaciones esenciales del objeto basando su contenido en la experiencia.

Se emplearon como métodos empíricos:

La **Entrevista** a funcionarios de la Cámara de Comercio con el objetivo de obtener información de interés para poder comprender el funcionamiento los procesos de convocatoria y contratación de participantes en las ferias y exposiciones.

La **Observación** de los procesos de convocatoria y contratación de participantes en la gestión de ferias y exposiciones para lograr un mejor entendimiento, comprensión y caracterización de los mismos.

La **Medición** a través del uso de métricas y pruebas de calidad para validar los módulos.

Para lograr la comprensión y claridad de los contenidos de la investigación realizada se ha estructurado el documento en tres capítulos.

En el **capítulo 1** se expone el estudio del estado del arte en función de los elementos esenciales para la solución del problema científico. Se fundamenta además, la selección de la metodología de desarrollo de software, los estilos arquitectónicos y paradigmas de programación, así como las herramientas, tecnologías y elementos a tener en cuenta para la verificación y validación de la solución a desarrollar.

En el **capítulo 2** se realiza un análisis crítico de todos los aspectos relacionados con el modelado del negocio, el análisis y diseño del sistema. Se describe la arquitectura, funcionalidades y requerimientos necesarios para el desarrollo de la solución informática a implementar. Se representa, a través de diagramas, los procesos de negocio, casos de uso del sistema, los diagramas de clases del análisis, secuencia y diseño, correspondientes a cada uno de los escenarios de los casos de uso, además del modelo de datos.

En el **capítulo 3** se abordan todos los elementos relacionados con la implementación y las pruebas, como son los diagramas de componentes y despliegue y los casos de pruebas. Además se realiza la validación funcional de la solución desarrollada, mediante la realización de pruebas de calidad.

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

Capítulo 1

Fundamentación teórica

1.1 Introducción

En el capítulo se traza como objetivo realizar el estudio del estado del arte correspondiente a los procesos de convocatoria y contratación que tienen lugar en los sistemas de gestión de ferias y exposiciones. Se incluyen además, definiciones de términos significativos para lograr una mejor comprensión del contenido y se mencionan y describen las tecnologías, metodologías y herramientas seleccionadas para desarrollar la solución propuesta, con el principal objetivo de obtener, a través de estas, un software que responda a las necesidades de los clientes.

1.2 Conceptos fundamentales

Entre las funciones que tiene la Cámara de Comercio se encuentra la **gestión de ferias y exposiciones** que se celebran en el país, proceso que tiene como objetivo garantizar su satisfactorio desarrollo a partir de la adecuada organización antes de la ejecución (Vega Miniet, 2012).

Los procesos de convocatoria y contratación guían el proceso fundamental de la Cámara de Comercio: la organización de ferias y exposiciones.

El **proceso convocatoria** tiene como objetivo mantener un registro de los eventos que ha organizado la institución, así como la información fundamental asociada al mismo. Es el encargado de la realización de una correcta convocatoria del evento, desde el momento de su creación, a las entidades potencialmente interesadas en participar (Vega Miniet, 2012).

El **proceso contratación** tiene como objetivo registrar los acuerdos contraídos entre la entidad participante en el evento y la Cámara de Comercio, así como las obligaciones de pago existentes. Una vez lanzada la convocatoria de un evento determinado, se registran los datos de las entidades interesadas en participar mediante un formulario de inscripción. Luego se deriva la negociación del contrato para el cual se trata de llegar a un acuerdo entre el participante y el organizador del evento en cuanto a la asignación de espacios y medios, existiendo la posibilidad de

que se le asigne lo solicitado o que haya que renegociar la solicitud. (Vega Miniet, 2012).

Para definir las características de calidad que se ven afectadas en el problema de la investigación, los autores se acogen a la definición establecida por NC ISO/IEC 9126:

Usabilidad: capacidad del producto de software de ser comprendido, aprendido, utilizado y de ser atractivo para el usuario, cuando se utilice bajo las condiciones especificadas.

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.

Para cumplir con los atributos de calidad antes definidos, es necesario evaluar cada una de las sub-características que los componen. La Tabla 1 muestra las subcaracterísticas de cada atributo de calidad.

Tabla 1. Sub-características de calidad.

Sub-características	
Usabilidad	Funcionalidad
Comprensibilidad: Capacidad del producto de software para permitirle al usuario entender si el software es idóneo y cómo puede usarse para las tareas y condiciones de uso particulares.	Idoneidad: Capacidad del software para mantener un conjunto apropiado de funciones para las tareas y los objetivos del usuario especificados.
Cognoscibilidad: Capacidad del producto del software para permitirle al usuario aprender su aplicación.	Precisión: Capacidad del software para proporcionar efectos o resultados correctos o convenidos con el grado de exactitud necesario.
Operabilidad: Capacidad del producto del software para permitirle al usuario operarlo y controlarlo.	Interoperabilidad: Capacidad del producto de software para interactuar recíprocamente con uno o más sistemas especificados.
Atracción: Capacidad del producto de software de ser atractivo o amigable para el usuario.	Seguridad: 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.

1.3 Estudio del estado del arte

En el ámbito internacional existen varios software de gestión de eventos, entre los que se encuentran: Amiando, Inscríbete, OfiEventos, PerfectTablePlan, RegOnline. Estas

herramientas son en su mayoría aplicaciones web que facilitan la gestión de todo tipo de eventos de una forma más eficiente.

Amiando: Software para la organización de eventos online desarrollado en Alemania en el 2011. Es una herramienta online para la gestión de eventos que provee al organizador de una perspectiva completa del evento. El sistema se encarga del cobro virtual y ofrece una amplia galería de diseños para la creación de la página oficial del evento. Además brinda reportes estadísticos de visita de página y registro (Bueno, 2012).

Inscribete: Aplicación web desarrollada por la compañía Doblemente, S.L. en España. Entre las principales funcionalidades que brinda se encuentran la inscripción online a eventos, la gestión de ferias con invitados, la gestión de eventos ilimitados, la gestión económica del evento, envío de email y SMS con las actualizaciones del evento. Posibilita además, la creación de una página web donde podrán inscribirse los participantes y mediante un panel de control exclusivo permite la gestión de todas las inscripciones y reservaciones (Doblemente S.L., 2007).

PerfectTablePlan: Software de gestión de eventos, bodas, banquetes y ceremonias. Planea eventos que requieren de una gran gestión y planificación previa. Ayuda haciendo todo el trabajo en segundos: lista de invitados, lugares en cada mesa y crear las invitaciones, entre otras funciones más. Esta aplicación es fácil de manejar y ayuda eficazmente a la toma de decisiones, así como a planificar cualquier tipo de evento que se vaya a desempeñar (PerfectTablePlan, 2009).

OfiEventos: es una completísima y profesional aplicación, totalmente en español, para la gestión, administración y organización de grandes eventos y comidas para colectivos de gran envergadura. Permite generar diferentes tipos de platos, calcular costes adicionales, generar presupuestos, llevar la facturación de los cobros y un control minucioso del almacén, enviar correos a los clientes o invitados, diseñar la distribución en el salón, obtener listados de invitados, llevar un inventario, entre otros (Ofi Eventos, 2010).

RegOnline: Está reconocido en todo el mundo como el software de gestión de eventos líder del sector y ofrece a los planificadores y organizadores de eventos un control total y una perspectiva completa de cada aspecto de sus eventos y asistentes. Es una solución flexible y accesible que permite realizar tareas de registro y gestión en línea para organizaciones de cualquier tipo y tamaño (RegOnline, 2010).

De forma general estas aplicaciones realizan los procesos de convocatoria y contratación de manera online. Permiten crear la página oficial del evento mediante la cual se inscriben las entidades interesadas en participar, definiendo los recursos que necesitan a través de un formulario de solicitud. Una vez que se tienen las entidades registradas, los organizadores del evento les asignan los recursos solicitados o negocian con los clientes por medio del correo electrónico especificado en el momento de la inscripción. Dichos sistemas establecen las opciones de pago, procesan automáticamente las tarjetas de crédito y permiten mantener actualizados a los participantes del evento mediante el envío de mensajes automáticos de confirmación y recordatorio.

Al aplicar a estos sistemas la lista de chequeo definida por Calisoft para medir la usabilidad en sitios web, resaltan de manera general algunos indicadores como la existencia de un Localizador de Recurso Uniforme (URL, por sus siglas en inglés Uniform Resource Locator) correcto, claro y fácil de recordar; los iconos que aparecen se identifican claramente con lo que representan; la información está organizada en categorías lógicas, fácilmente memorizables para el usuario; reflejan la identidad de la empresa; cuentan con un mapa o buscador que facilita el acceso directo a los contenidos; presentan un buen diseño de las interfaces de usuario; la manera de navegar por la web o ejecución de tareas asignadas se aprenden de forma rápida; se mantiene una tipografía coherente en todo el sitio web y son compatibles con navegadores antiguos y modernos.

Por todo lo antes expuesto se puede concluir que los sitios analizados presentan un elevado nivel de usabilidad, pueden ser comprendidos, aprendidos, utilizados por usuarios con un mínimo de experiencia y su diseño atractivo los hacen ser sitios de referencia a nivel mundial.

A pesar de existir diversas soluciones informáticas para la gestión de eventos a nivel mundial, por las características que presenta la infraestructura tecnológica de la Cámara de Comercio, no es factible construir un software con todas las funcionalidades mencionadas anteriormente, pues no todas serían explotadas. Además estos sistemas tienen el impedimento de ser privativos y para poder usarlos la Cámara de Comercio debe pagar licencias.

En Cuba existe actualmente la herramienta SAEFE, un sistema para gestionar las ferias comerciales que realiza la Cámara de Comercio, pero no permite ejecutar correctamente los procesos de convocatoria y contratación ya que no cuenta con un módulo para la asignación de recursos, de modo que sea posible configurar la estructura de los locales de los eventos, dificultando la asignación adecuada de

espacios a los participantes y la gestión de los recursos. Además presenta deficiencias en las funcionalidades de inscripción y acreditación, dificulta la determinación del costo de los eventos y tiene un bajo nivel de usabilidad.

1.4 Análisis de las metodologías de desarrollo

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar y documentar un software. Se encarga de elaborar estrategias; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega. Su objetivo es elevar la calidad del software a través de un mayor control sobre el proceso (Sommerville, 2007).

En la actualidad existen dos grandes tendencias de metodologías: ágiles y tradicionales.

Las metodologías ágiles se caracterizan por tener un desarrollo incremental para producir tempranamente pequeñas entregas en ciclos rápidos y disposición para el cambio y adaptación continua (Claro Sánchez, y otros, 2011). Entre las más utilizadas se encuentran eXtreme Programming (XP) y Scrum.

Las metodologías tradicionales o pesadas se centran en el control del proceso y establecen una rigurosa planificación de tareas y responsabilidades, generando gran cantidad de documentos y artefactos. Estas se distinguen por especificar las herramientas y notaciones que deben ser utilizadas durante la construcción del producto (Claro Sánchez, y otros, 2011). Entre las metodologías tradicionales o pesadas se encuentra RUP llamada así por sus siglas en inglés Rational Unified Process (Proceso Unificado de Rational) y MSF (Microsoft Solution Framework).

Como el software a desarrollar cuenta con requisitos definidos y es orientado al cliente, se requiere una documentación detallada del mismo porque una vez liberado el producto el cliente es el responsable de su administración y mantenimiento, necesitando conocer a profundidad los artefactos relevantes del proceso de desarrollo de software. Con esta premisa se desecha el uso de las metodologías ágiles pues de forma general estas se caracterizan por estar centradas en desarrollar productos funcionales, más que en conseguir una buena documentación.

Entre las metodologías tradicionales, se puede afirmar que no es factible la utilización de MSF ya que aunque está orientada para proyectos pequeños o medianos, la documentación que genera no es muy detallada. Además otra característica a tener en cuenta es que MSF presenta una gran cantidad de retroalimentación, ya que en cada etapa se desarrollan aplicativos funcionales. Los mismos son revisados y validados por el equipo de pruebas y finalmente por el usuario, pudiendo este último sugerir

cambios o en el caso extremo, solicitar cambios en la funcionalidad generando costos y tiempos de retraso en el cronograma del proyecto (Villarroel González, y otros, 2008).

Por tanto para regir el proceso de desarrollo de esta investigación, se selecciona como metodología de desarrollo a RUP porque es aplicable a proyectos a largo plazo, intentando reducir la complejidad del software y la realización del mismo. Permite la obtención de una gran cantidad de artefactos que describen el proceso de desarrollo de software y que pueden ser de gran utilidad para futuras versiones de la aplicación. Esta metodología es capaz de adaptarse a las características y complejidad de cualquier proyecto de software realizando una buena captura de los requisitos y siendo apropiada para aplicar a proyectos complejos. Además se contempla en las nuevas metas que se ha trazado la universidad, de desarrollar un programa de mejora de sus procesos basado en el modelo CMMI².

1.4.1 Proceso Unificado de Rational

El Proceso Unificado de Desarrollo es un proceso de ingeniería de software planteado por Kruchten (1996) cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido. Cubre el ciclo de vida y desarrollo de software (Kruchten, 2000). Es una metodología muy utilizada por equipos de desarrollo de software y adaptable prácticamente a cualquier proyecto. Toma en cuenta las mejores prácticas en el modelo de desarrollo de software entre las que se encuentran el desarrollo de software de forma iterativa, el manejo de requerimientos, utiliza la arquitectura basada en componentes, modela el software visualmente, verifica la calidad del software y controla los cambios (Jacobson, y otros, 2000).

Su ciclo de vida se caracteriza por ser iterativo e incremental, dirigido por casos de uso y centrado en la arquitectura. Consta de cuatro fases o etapas:

1. Comienzo o Inicio: Se describe el negocio y se delimita el proyecto, describiendo sus alcances con la identificación de los casos de uso del sistema.
2. Elaboración: Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen.

² Capability Maturity Model Integration: modelo de referencia para la calidad en los procesos de desarrollo y mantenimiento de software, que incrementa la satisfacción de las necesidades de los usuarios internos del sistema (trabajadores), permitiendo la elaboración de productos de calidad, dentro del tiempo y costos previstos.

3. Construcción: Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario. Se obtiene 1 o varias entregas del producto que han pasado las pruebas.
4. Transición: Con la liberación, ya el producto está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Además de las fases, el ciclo de vida de esta metodología contiene flujos de trabajos o disciplinas que agrupan de forma lógica las actividades por naturaleza (Jacobson, y otros, 2000). En la Figura 1 se muestra el esfuerzo que conlleva cada una de las fases en los diferentes flujos de trabajo.

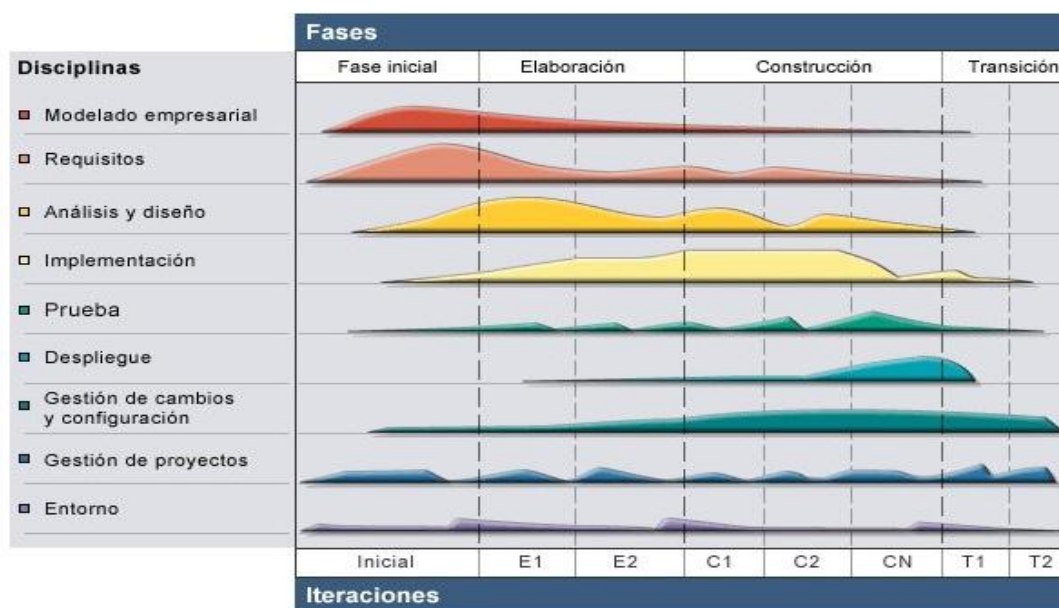


Figura 1. Fases de RUP.

En cada una de las iteraciones se obtendrá una versión del producto entregable al cliente.

1.5 Arquitectura de software

La Arquitectura de Software es una disciplina que tiene sus orígenes en los años sesenta del siglo pasado. Teniendo en cuenta la variedad de las definiciones y los grandes aportes a esta rama se decidió estandarizar un concepto que se ha establecido por IEEE³. *La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución* (Arquitectura de Software- IEEE 1471-2000).

³IEEE corresponde a las siglas de The Institute of Electrical and Electronics Engineers.

1.5.1 Arquitectura cliente/servidor

La arquitectura cliente/servidor describe la relación de dos programas de computadoras donde un programa, el cliente, hace una llamada a otro programa, el servidor. El uso de esta arquitectura se ha hecho muy popular debido a los grandes beneficios que brinda para el desarrollo de aplicaciones, entre los que se encuentran (J.D. Meier, 2002):

- ✓ 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.
- ✓ Delegación de la mayor carga de procesamiento posible hacia el servidor y mantener los clientes libres de gran parte de la lógica de negocio de las aplicaciones es una tendencia y buena práctica en estos entornos.
- ✓ Posibilidad de escalar horizontal y verticalmente el servidor, que significa clusterizar el servidor y añadir mayor cantidad de recursos respectivamente. Esto se traduce en ofrecer mayor disponibilidad de servicios y de aplicaciones y obtener mejores índices de rendimiento.

1.5.2 Arquitectura en capas

La arquitectura 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. Este estilo se identifica por las siguientes características (Microsoft Corporation, 2009):

- ✓ Describe la descomposición de servicios de forma que la mayoría de la interacción ocurre solamente entre capas vecinas.
- ✓ Las capas de una aplicación pueden residir en la misma máquina física (misma capa) o puede estar distribuido sobre diferentes computadores (n-capas).
- ✓ Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces bien definidas.

Las principales ventajas del estilo de arquitectura en capas son:

- ✓ Abstracción: Las capas permiten cambios que se realicen en un nivel abstracto.

- ✓ Aislamiento: El estilo de arquitectura de capas permite asilar 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 dichas interfaces.

1.5.3 Arquitectura basada en componentes

Se basa en la división de la lógica de la aplicación en componentes de software, que más tarde elevarán los niveles de reutilización, mantenibilidad, integración y seguridad. Un componente de software no es más que: *una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, 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).

La arquitectura orientada al desarrollo de componentes brinda los siguientes beneficios:

- ✓ 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 (Fadruga Artiles, y otros, 2012).

1.6 Herramientas y tecnologías

Para el desarrollo de los módulos Convocatoria y Contratación del Sistema de Gestión de Ferias y Eventos de la Cámara de Comercio es necesario seleccionar las herramientas y tecnologías de desarrollo que cumplan con las necesidades requeridas. A continuación se abordan las que fueron utilizadas.

1.6.1 Lenguaje de modelado

Un lenguaje de modelado es un conjunto estandarizado de símbolos dispuestos para modelar parte de un diseño de software orientado a objetos. Estos se combinan con

las metodologías de desarrollo para avanzar de una especificación inicial a un plan de implementación, para comunicar dicho plan a todo un equipo de desarrolladores (Velázquez Corona, 2008). Los modelos proporcionan un mayor nivel de abstracción, permitiendo trabajar con sistemas mayores y más complejos facilitando el proceso de codificación e implementación del sistema de forma distribuida y en distintas plataformas (Catalá Matienzo, 2010).

1.6.1.1 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés Unified Modeling Language) es un lenguaje gráfico para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (Rumbaugh, y otros, 2000). Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Es importante resaltar que es un "lenguaje" para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir.

Además es un método formal de modelado que aporta las siguientes potencialidades (Hernández Orallo, 2002):

- ✓ Mayor rigor en la especificación.
- ✓ Permite realizar una verificación y validación del modelo realizado.
- ✓ Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de la estructura de un proyecto. Se utilizará en su versión 2.0.

1.6.2 Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés, Computer Aided Software Engineering) son utilizadas con el fin de automatizar los aspectos claves de todo un proceso de desarrollo de software de un sistema (Pérez Chirino, 2010). Permiten modelar procesos de desarrollo de software, y tienen como objetivo fundamental solucionar y afrontar los problemas de una mala calidad de software y una documentación inadecuada (Périssé, 2001).

1.6.2.1 Visual Paradigm for UML 8.0

Es una herramienta CASE que utiliza UML como lenguaje de modelado, con el uso del acercamiento orientado al objeto. Contiene varios productos o módulos que facilitan el

trabajo durante la confección de un software, procurando garantizar la calidad en el producto final. Se caracteriza por (Visual Paradigm, 2009):

- ✓ Permitir un diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad.
- ✓ Permitir la sincronización entre Diagramas de Entidad Relación y Diagramas de Clases.
- ✓ Ser una herramienta CASE que soporta las últimas versiones del Lenguaje Unificado de Modelado y del modelado de procesos de negocio.

Se selecciona Visual Paradigm for UML 8.0 por soportar el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue; permitiendo la elaboración de aplicaciones con calidad. Es una herramienta multiplataforma y amigable en su entorno, de la que se tiene conocimiento precedente, lo que facilita su uso e interoperabilidad con otras aplicaciones. A pesar de ser una herramienta propietaria, la UCI cuenta con su licencia.

1.6.3 Notación para el modelado de procesos de negocio

BPMN denominada así por sus siglas en inglés Business Process Modeling Notation (Notación para el Modelado de Procesos de Negocio) es utilizada por ser un estándar internacional de modelado de procesos, independiente de cualquier metodología de modelado de procesos. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. Proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente (Yáñez Martínez , 2008).

1.6.4 Análisis de los lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos, caracteres y reglas que son entendibles y ejecutables por un computador. Tienen un conjunto de instrucciones que permiten realizar operaciones de entrada / salida, manipulación de textos, lógica / comparación y almacenamiento / recuperación.

Para la implementación de los módulos se realiza un análisis de los lenguajes de programación más utilizados a nivel mundial en el desarrollo de aplicaciones de escritorio, C#, C++ y Java.

El lenguaje C++ no es recomendable porque la programación se realiza a bajo nivel, implicando un importante incremento de la dificultad en cuanto a la declaración de eventos, visualización, punteros, control de la memoria y una disminución de la productividad. Aunque es muy bueno en cuanto al rendimiento y la protección del

código presenta una limitada portabilidad en diferentes sistemas operativos porque depende de bibliotecas que en ocasiones son dependientes del sistema operativo (Martínez Prieto, 2012).

Teniendo como premisa las políticas de migración seguidas por el país, C# no es una alternativa viable porque está atado a Microsoft y a Windows. Vale aclarar que existe una plataforma llamada Mono⁴ la cual es inestable, no está a la par con la que corre bajo Windows y presenta problemas de rendimiento pues requiere de un buen número de recursos para funcionar (Álvarez Román, y otros, 2005). Al C# estar estandarizado por el ECMA 334⁵, las nuevas características que se incorporen, pueden que no se definan en el estándar y queden como propiedad de Microsoft, limitando así el uso del lenguaje (Paredes Galarza, y otros, 2011).

Se decide hacer uso de Java para el desarrollo de los módulos porque es un lenguaje robusto, simple, poderoso y fácil de aprender. Es un lenguaje multiplataforma de código abierto, distribuido, que proporciona un conjunto de clases para su uso en aplicaciones de red, permitiendo abrir sockets y establecer conexiones con servidores o clientes remotos. Java fue diseñado para crear software altamente fiable (Lara Bello, 2010).

A diferencia de los lenguajes convencionales, que generalmente están diseñados para ser compilados a código nativo, este es compilado a un código intermedio o código byte, el cual es interpretado por la máquina virtual de Java. La estrategia de no compilar en código nativo hasta la ejecución de la aplicación tiene el objetivo principal de mejorar la portabilidad de las aplicaciones desarrolladas.

1.6.4.1 Java

Es un lenguaje de desarrollo de propósito general desarrollado por un grupo de ingenieros de Sun Microsystems en 1995. Entre sus características fundamentales se encuentran (Ken, y otros, 2001):

- ✓ Es intrínsecamente orientado a objetos y tiene una gran funcionalidad gracias a sus bibliotecas (clases).
- ✓ Aprovecha características de la mayoría de los lenguajes modernos y el manejo de la memoria no es un problema, la gestiona el propio lenguaje y no el programador.

⁴Plataforma de software, patrocinada por Novell. Constituye una implementación de código abierto de .NET Framework de Microsoft, basada en el estándar ECMA para C# y el Common Language Runtime.

⁵ Estándar internacional de la organización ECMA International que especifica la forma y establece la interpretación de los programas escritos en el lenguaje de programación C#.

- ✓ Incorpora Multi-Threading (Multihilo en español) es un paradigma conceptual de la programación que permite la ejecución de tareas concurrentes dentro de un mismo programa.
- ✓ Contiene estructuras para la detección de excepciones (errores de ejecución previstos) y permite obligar al programador a escribir código fiable mediante la declaración de excepciones posibles para una determinada clase reusable.

1.6.5 Swing y SwingX

Las aplicaciones de escritorio son programas que se utilizan como herramientas para una operación o tarea específica. Ayudan a la solución de problemas en diferentes campos, convirtiéndose en recursos estratégicos en el sector empresarial. Los componentes visuales juegan un papel importante a la hora de construir las interfaces gráficas que permitirán el intercambio con el usuario. En el lenguaje de programación Java se encuentran dos bibliotecas que hacen de esta tarea un trabajo más simple. Por su importancia en el desarrollo de la solución propuesta se brinda a continuación una breve descripción de cada una.

Swing: es el paquete gráfico que apareció en la versión 1.2 de Java. Está compuesto por un amplio conjunto de componentes de interfaces gráficas de usuario multiplataforma. Cada uno de los componentes de este paquete puede presentar diversos aspectos y comportamientos en función de una biblioteca de clases.

La arquitectura Swing presenta una serie de ventajas respecto a su antecedente AWT⁶ (Manzanedo del Campo, y otros, 1999):

- ✓ Amplia variedad de componentes
- ✓ Arquitectura Modelo-Vista-Controlador: Esta arquitectura da lugar a todo un enfoque de desarrollo muy arraigado en los entornos gráficos de usuario realizados con técnicas orientadas a objetos. Cada componente tiene asociado una clase del modelo de datos y una interfaz que lo utiliza.
- ✓ Objetos de acción: Estos objetos cuando están activados controlan las acciones de varios objetos componentes de la interfaz gráfica de usuario.
- ✓ Contenedores anidados
- ✓ Diálogos personalizados

⁶ Abstract Window Toolkit (AWT, en español Kit de Herramientas de Ventana Abstracta) es un kit de herramientas de gráficos, interfaz de usuario, y sistema de ventanas independiente de la plataforma original de Java.

SwingX: es un proyecto que amplía las capacidades de escritorio de Swing con características tales como tablas de árboles, la función de autocompletar, punta del día, paneles plegables, un componente de selector de fecha, clasificación, filtrado, y poner de relieve las tablas, árboles y listas. El código fuente y binario se basa en componentes SwingX que está disponible en SwingLabs, proporcionando un conjunto de componentes en su biblioteca que mejoran la visualización de aplicaciones de escritorio (Haines, 2009).

1.6.6 Plataforma de desarrollo

La plataforma Java es el nombre de un entorno o plataforma de computación originaria de Sun Microsystems, capaz de ejecutar aplicaciones desarrolladas usando el lenguaje de programación Java u otros lenguajes que compilen a código byte. Su principal ventaja es que su entorno de desarrollo es independiente de la plataforma sobre la que se trabaje, es decir, sus aplicaciones son funcionales independientemente del sistema operativo sobre el que estén operando.

Como los módulos a desarrollar forman parte de un aplicación con un fin empresarial la plataforma seleccionada es Java Platform, Enterprise Edition. Java EE permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. Tiene varias especificaciones de API, tales como JDBC⁷, JMS⁸, Servicios Web, etc y define cómo coordinarlos. También configura algunas especificaciones únicas para componentes. Estas incluyen Enterprise JavaBeans (EJB), JavaServer Pages y varias tecnologías de servicios web. Ello permite al desarrollador crear una aplicación de empresa portable entre plataformas (Bodoff, 2004).

Como uno de los componentes de esta plataforma se utiliza la máquina virtual de Java (JVM) en su versión JDK- 6u26, que es un programa que se encarga de interpretar código intermedio (código byte) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuando las llamadas pertinentes al Sistema Operativo (S.O).

Así, la JVM proporciona al programa Java independencia de plataforma respecto al hardware y al S.O subyacente. (Guadalinfo, 2011).

⁷ **Java Database Connectivity:** es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java

⁸ **Java Message Service:** es una API que facilita la comunicación entre aplicaciones.

1.6.7 Entorno de Desarrollo Integrado

Un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés Integrated Development Environment) es un conjunto de herramientas utilizadas por los programadores, que incluye por lo general, un editor de código, administrador de proyectos y archivos, enlace a compiladores e integración con sistemas controladores de versiones o repositorios, además de brindar facilidades para la construcción de interfaces gráficas de usuario (Bradshaw Venzant, y otros, 2010). Entre los entornos de desarrollo más utilizados para la creación de aplicaciones en el lenguaje Java a nivel mundial, se encuentran Eclipse y Netbeans.

Teniendo como punto de partida el conocimiento precedente del equipo de desarrollo sobre el trabajo con Netbeans, se decide hacer uso del mismo para el desarrollo de los módulos porque es más fácil e intuitivo a la hora de crear aplicaciones empresariales. Además presenta una alta modularidad, proporcionando mediante módulos todas las funciones del IDE, tales como el soporte de Java, edición, o soporte para el sistema de control de versiones, entre otras; conteniendo todos los módulos necesarios para el desarrollo de aplicaciones Java, permitiéndole al usuario comenzar a trabajar inmediatamente.

A diferencia de Eclipse que se caracteriza por presentar una dependencia total de complementos (plugins en inglés) siendo algunos gratis y otros comerciales, lo que dificulta en gran medida la solución de problemas específicos surgidos durante el proceso de desarrollo de software. En Eclipse no es muy agradable depurar y se caracteriza por ser un alto consumidor de recursos del sistema.

1.6.7.1 Netbeans IDE 7.2

Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas. Dispone de soporte para el desarrollo de aplicaciones web, creación de aplicaciones compatibles con teléfonos móviles y sus funcionalidades son ampliables mediante la instalación de paquetes. Además dispone de herramientas de concepción visual para crear y manipular componentes visuales, y para la generación de código (Ramírez, 2012).

Entre las mejoras que brinda en su versión 7.2 se encuentran un rendimiento mejorado y la experiencia de codificación con las nuevas capacidades de análisis de código estático en el editor de Java. También incluye características notables, como soporte

para pruebas de marco (TestNG⁹), refactorizaciones, diversas mejoras del depurador, soporte mejorado y actualizado para suites de componentes EJB. (Netbeans.org, 2012).

1.6.8 Java Persistence API

Java Persistence API, más conocida por sus siglas JPA, es la API de persistencia desarrollada para la plataforma Java EE e incluida en el estándar EJB3. Esta API busca unificar la manera en que funcionan las utilidades que proveen una conversión objeto-relacional. El objetivo que persigue el diseño de esta API es no perder las ventajas de la orientación a objetos al interactuar con una base de datos y permitir usar objetos regulares (conocidos como POJOs¹⁰) (Junta de Andalucía, 2012).

La conversión objeto/relacional, es decir, la relación entre entidades Java y tablas de la base de datos, se realiza mediante anotaciones en las propias clases de entidad, por lo que no se requieren ficheros descriptores XML. También pueden definirse transacciones como anotaciones JPA.

JPA es una solución ORM completa llamada así por sus siglas en inglés Object-Relational Mapping (Conversión Objeto-Relacional) y soporta la herencia y polimorfismo. También se define el lenguaje de consulta JPQL (Java Persistence Query Language). Toma las mejores ideas de las tecnologías de la persistencia de otros, definiendo un modelo de persistencia estándar para todas las aplicaciones Java.

Entre las principales ventajas que proporciona se encuentran la independencia de la base de datos, bajo acoplamiento entre negocio y persistencia, y un desarrollo rápido. Esto permite centrar los esfuerzos en optimizar las consultas que realmente lo merecen (Junta de Andalucía, 2012).

Para el uso de JPA se requiere un proveedor que lo implemente; seleccionándose EclipseLink (proyecto de Servicios de Persistencia de la Fundación Eclipse) porque presenta elevada eficiencia en cuanto a la conversión de objetos, realizando para ello un proceso de mejora denominado “tejer”, el cual tiene el propósito de alterar el código byte de Java con el objetivo de agregar instrucciones JPA optimizadas. Dicha mejora

⁹ TestNG es un framework para pruebas que trabaja con Java basado en JUnit.

¹⁰ POJO (Plain Old Java Object) es una sigla utilizada por programadores Java para enfatizar el uso de clases simples y que no dependen de un marco de trabajo en especial. Este acrónimo surge en oposición al modelo planteado por los estándares Enterprise JavaBeans (EJB) anteriores al 3.0, en los que los EJB debían implementar interfaces especiales.

presenta resultados positivos tanto en tiempo de compilación como de ejecución. Además esta implementación es muy eficiente trabajando con PostgreSQL (Rubio, 2010).

1.6.8.1 EclipseLink

EclipseLink es un marco de trabajo de persistencia completo, integral y universal. Se ejecuta en cualquier ambiente Java, lee y escribe objetos a prácticamente cualquier tipo de fuente de datos, incluyendo bases de datos relacionales, XML, o sistemas EIS ¹¹ (de Sande Tundidor, 2009). Incluye además extensiones de funciones avanzadas, que ofrecen una solución de persistencia en tiempo de ejecución centrado en los principales estándares y ha extendido funcionalidades necesarias para Java Empresarial y para el desarrollo de aplicaciones orientas a servicios (Eclipse Foundation, 2013).

1.6.9 Análisis de los Sistemas Gestores de Bases de Datos

Un Sistema Gestor de Bases de Datos (SGBD) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Para la selección del SGBD se realiza un análisis de dos de los más utilizados a nivel mundial PostgreSQL y MySQL.

Compartiendo el criterio de varios autores de que la selección de un SGDB depende de los requerimientos y necesidades específicas del software, se decide utilizar PostgreSQL porque presenta características y ventajas que son necesarias para la aplicación a desarrollar. Entre estas se encuentran la implementación de operaciones que permiten revertir cambios en la base de datos, subconsultas y transacciones, haciendo su funcionamiento mucho más eficaz. Además es capaz de soportar una mayor cantidad de peticiones simultáneas de manera correcta en entornos empresariales y ofrece soluciones en campos en las que MySQL no podría.

Mientras que MySQL carece de soporte para transacciones, subconsultas y operaciones que permiten revertir cambios en la base de datos porque no maneja la integridad referencial, haciéndolo una solución pobre para muchos campos de

¹¹Sistema de Información Ejecutiva (*Executive Information System*, **EIS** por sus siglas en inglés) es una herramienta de Inteligencia Empresarial, orientada a usuarios de nivel gerencial, que permite monitorear el estado de las variables de un área o unidad de la empresa a partir de información interna y externa a la misma.

aplicación y no es viable su uso con grandes bases de datos, a las que se acceda continuamente, ya que no implementa una buena escalabilidad (Peco Martínez, 2010).

1.6.9.1 PostgreSQL 9.1

PostgreSQL es un SGBD relacional, de código abierto que soporta una gran parte del estándar SQL y ofrece muchas características modernas entre las que se destacan (The PostgreSQL Global Development Group, 2011):

- ✓ Es una base de datos cuyas transacciones cumplen con las características: Atomicidad, Consistencia, Aislamiento y Durabilidad (The PostgreSQL Global Development Group, 2010).
- ✓ Soporta todas las características de una base de datos profesional: disparadores, procedimientos almacenados, funciones, secuencias, relaciones, reglas, tipos de datos definidos por usuarios, vistas, vistas materializadas, etc.
- ✓ Multi-Version Concurrency Control (MVCC). Esta tecnología permite una alta concurrencia donde las operaciones de lectura y escritura no se bloquean entre sí.
- ✓ Replicación asincrónica/sincrónica.
- ✓ Está disponible para Linux en todas sus variantes y Windows 32/64bit.
- ✓ Múltiples métodos de autenticación.
- ✓ Acceso encriptado vía SSL llamado así por sus siglas en inglés Secure Sockets Layer (Capa de Conexión Segura).
- ✓ Comunidades muy activas, varias comunidades en castellano.
- ✓ Completa documentación, muy bien organizada, pública y libre, con comentarios de los propios usuarios.

El trabajo con el SGBD se realizará mediante la herramienta administrativa PgAdmin III en su versión 1.14.2.

1.6.10 Tecnología EJB

La tecnología Enterprise JavaBeans (también conocidos por sus siglas EJB), es una estructura de componentes del lado del servidor que simplifica el proceso de construcción de clases empresariales. Mediante el uso de EJB, es posible crear aplicaciones escalables, confiables y seguras. Además es un estándar para el desarrollo y despliegue de componentes de servidor distribuidos en aplicaciones Java. Define un acuerdo (contrato) entre componentes y servidores de aplicaciones que permite a cualquier componente poder ejecutarse en cualquier servidor de aplicaciones compatible (Patel Sriganesh, y otros, 2006).

Un servidor de aplicaciones es una tecnología básica que proporciona la infraestructura y servicios claves a las aplicaciones alojadas en un sistema. Entre los servicios habituales de un servidor de aplicaciones se incluyen los siguientes (Microsoft TechNet, 2011):

- ✓ Agrupación de recursos (por ejemplo, agrupación de conexiones de base de datos y agrupación de objetos).
- ✓ Administración de transacciones distribuida.
- ✓ Servicios de detección de errores y estado de las aplicaciones.
- ✓ Comunicación asincrónica de programa, normalmente a través de colas de mensajes que se lleva a cabo a través del estándar de mensajería JMS.

JMS: es la solución de Sun para los sistemas de mensajes. Estos aportan una serie de mejoras a la comunicación entre aplicaciones que no tienen por qué residir en la misma máquina. JMS se sitúa como middleware¹² en medio de la comunicación de dos aplicaciones (Programación en catellano, 2011). Aporta diversas ventajas, entre las que se encuentran, la presencia de un almacén de mensajes que centraliza y simplifica el almacenamiento y la recuperación de mensajes, su arquitectura no restringe el número de clientes que pueden acceder al servicio y garantiza una comunicación asíncrona. Además el protocolo JMS es fácil de transmitir a través de servidores de seguridad y otra infraestructura de red (Oracle, 2010).

En el caso de la aplicación a desarrollar se utiliza el servidor de aplicaciones GlassFish en su versión 3.1.1, desarrollado por Sun Microsystems que implementa las tecnologías definidas en la plataforma Java EE y permite ejecutar aplicaciones que siguen esta especificación. Es gratuito y de código libre (Gutiérrez, 2009). Entre sus principales características se destacan (Rojas, 2009):

- ✓ La contribución de la comunidad: cuenta con múltiples contribuciones de los miembros de la comunidad de código abierto y Java.
- ✓ Nuevas funcionalidades de monitorización y gestión: introduce nuevas funcionalidades para sus capacidades de gestión y monitorización incluyendo un seguimiento preciso y minucioso.

¹²software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, software, redes, hardware y/o sistemas operativos. Éste simplifica el trabajo de los programadores en la compleja tarea de generar las conexiones.

✓ Flexible, extensible, personalizable: la flexibilidad y extensibilidad de Sun GlassFish Enterprise Server permite a otros proveedores aprovechar un API embebido para crear una solución integrada y personalizada dentro de una única máquina virtual Java.

1.6.11 Sistema de control de versiones

Un sistema de control de versiones es un sistema de gestión de archivos y directorios cuya principal característica es que mantiene el registro de los cambios y modificaciones que se han realizado sobre estos a lo largo del tiempo. De esta forma, el sistema es capaz de recordar las versiones antiguas de los datos, lo que permite examinar el historial de cambios o recuperar versiones anteriores de un fichero (Collins-Sussma, y otros, 2004).

1.6.11.1 Subversion (SVN)

Subversion es un sistema de control de versiones libre y de código fuente abierto. Es decir, maneja ficheros y directorios a través del tiempo. Hay un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos (Collins-Sussma, y otros, 2004). Entre las ventajas que proporciona el uso de Subversión como sistema de control de versiones se encuentran (Vindas, 2008):

- ✓ Visibilidad del cambio: facilita mantener involucrada a cada persona de su equipo en el proyecto, ya que los cambios son visibles para todos una vez que son realizados.
- ✓ Versión a versión: guardará únicamente las diferencias de una versión a otra en el servidor.
- ✓ Administración: permite al administrador del proyecto observar la productividad de su equipo de trabajo, así como visualizar el historial de cambios de un producto, permitiendo una mejor administración.

De acuerdo con las políticas de configuración establecidas por el proyecto “Sistema de Gestión de Ferias y Eventos” se decide utilizar como sistema de control de versiones SVN, mediante el cliente TortoiseSVN en su versión 1.7.0.

1.7 Métricas y técnicas de pruebas

La (IEEE Standards Board, 1994) define como métrica *una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.*

Para la validación de la solución propuesta se decide utilizar también métricas porque no sólo permiten entender mejor los atributos del software, sino que ayudan fundamentalmente a valorar su calidad.

Las métricas de software pueden clasificarse en internas o externas. Las métricas internas le proporcionan a los desarrolladores la habilidad de medir la calidad de los productos intermedios, con lo cual se puede predecir la calidad del producto final (Góngora Rodríguez, 2011).

Serán utilizadas métricas internas basadas en el estudio de la calidad del diseño orientado a objeto referenciadas por (Pressman, 2010) teniendo en cuenta que este estudio brinda un esquema sencillo de implementación y que a la vez cubre los principales atributos de calidad de software. Las métricas escogidas para la validación del diseño fueron Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC). La descripción de estas métricas puede ser consultada en el Anexo 1.

Por otro lado, las métricas externas pueden ser usadas para medir la calidad del software a través de la medición del comportamiento del sistema en su totalidad (Góngora Rodríguez, 2011).

Para la evaluación de las características de calidad funcionalidad y usabilidad serán utilizadas métricas externas durante la etapa de pruebas. Su uso permitirá calcular un índice de cumplimiento de cada una de las sub-características que las componen, para finalmente determinar el nivel (Alto, Medio, Bajo) de cada atributo en el software.

Como estrategia para realizar el procedimiento de evaluación, se utilizarán métricas definidas en la Norma ISO/IEC 9126, en el Procedimiento de Evaluación de la Usabilidad de Productos de Software de (Pérez Dima, 2011), en la Guía para Aseguramiento de la Funcionalidad en los Sistemas Informáticos de Gobierno Electrónico de (Almora Gálvez, 2012) y en el Procedimiento para evaluar la característica de funcionalidad de componentes de software de (Estrada Peña, 2012). Las métricas seleccionadas se encuentran descritas en el Anexo 2 y Anexo 3.

Las técnicas de prueba definen qué estrategia seguir en cuanto a la verificación y validación del sistema, ya que están diseñadas con el propósito de descubrir fallos (Terry Morales, 2010). Las pruebas se enfocan sobre la lógica interna del software y las funciones externas, para lo cual se definen las siguientes técnicas:

Técnica de Caja Blanca: Las pruebas de caja blanca, denominadas a veces pruebas de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Se realizan sobre las funciones internas de un módulo y son ejecutadas por los miembros del

equipo, permitiendo a los programadores darse cuenta de los errores cometidos (Pressman, 2010).

Técnica de Caja Negra: Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. Tratan de encontrar errores en la siguientes categorías: 1) funciones incorrectas o faltantes, 2) errores de interfaz gráfica de usuario, 3) errores de estructuras de datos o en el acceso a bases de datos externas, 4) errores de comportamiento o desempeño y 5) errores de inicialización y término (Pressman, 2010).

1.8 Conclusiones del capítulo

En este capítulo se realizó un análisis profundo de los principales sistemas de gestión de ferias y exposiciones que existen a nivel mundial, centrándose en los procesos de convocatoria y contratación, concluyendo que los mismos no pueden ser utilizados porque son en su mayoría aplicaciones web privativas que no se adaptan a las limitaciones que presenta la infraestructura tecnológica de la Cámara de Comercio. De acuerdo a las exigencias y características actuales de dicha institución el sistema con que cuentan actualmente (SAEFE) no satisface sus necesidades entorno a la realización de estos procesos.

A partir de un análisis exhaustivo de las ventajas y limitaciones de las metodologías de desarrollo de software y teniendo como premisa las exigencias y características del nuevo sistema, se selecciona a RUP para regir el procesos de desarrollo de la investigación, porque genera gran cantidad de artefactos que brindan al usuario mayor autonomía para mantener el sistema.

Además el análisis permitió definir una arquitectura estable y sólida, que soporte todos los requisitos especificados; la selección de un conjunto de herramientas y tecnologías para garantizar el desarrollo de una solución acorde a las necesidades del cliente y un conjunto de métricas que evaluarán la calidad de los resultados que se obtengan en cada una de las disciplinas a desarrollar.

Partiendo de los resultados anteriores se propone la metodología, arquitectura, herramientas y tecnologías que permitirán desarrollar los módulos Convocatoria y Contratación que requiere el SIGFE para la gestión de ferias y exposiciones de la Cámara de Comercio, de manera tal que contribuyan a su funcionalidad y facilidad de uso.

Capítulo 2 Descripción de la propuesta de solución

2.1 Introducción

En el presente capítulo se realiza un análisis del negocio de los módulos a desarrollar. Se explican los procesos que se desean automatizar y el dominio existente, dado por el conjunto de entidades que en este intervienen. Se describe la arquitectura del nuevo sistema, así como los requisitos funcionales y no funcionales a tener en cuenta para el desarrollo de la solución propuesta. Se elaboraron los diagramas del análisis y diseño así como las tablas necesarias que garantizan un buen diseño de la base de datos, condicionando el cumplimiento de los objetivos trazados.

2.2 Propuesta de solución

La propuesta de solución de este trabajo consiste en el análisis, diseño, implementación y prueba de los módulos Convocatoria y Contratación del SIGFE, de manera tal que contribuyan a su funcionalidad y facilidad de uso. Dicha solución contendrá las funcionalidades necesarias para que exista un correcto funcionamiento de las tareas relacionadas con los procesos a informatizar.

2.3 Arquitectura del sistema

A partir del análisis realizado en el capítulo anterior la arquitectura del sistema se basa en una combinación del estilo Cliente/Servidor, distribuida en tres niveles, y basada en el desarrollo de componentes. A continuación se realizará una descripción detallada de la arquitectura haciendo énfasis en los estilos arquitectónicos y los patrones que se utilizan.

Haciendo uso del estilo cliente/servidor existe un cliente que presenta llamadas a un servidor de aplicaciones y este le retorna el resultado de la llamada para que se encargue de la presentación de los datos en una interfaz gráfica de usuario, usando el patrón de diseño Modelo Vista Presentador. Este patrón tiene como objetivo separar la dicha interfaz de la lógica de las aplicaciones. Está compuesto básicamente por 3 componentes:

✓ La Vista está compuesta por las ventanas y controles que conforman la interfaz gráfica de usuario.

- ✓ El Modelo se ocupa de toda la lógica de negocio.
- ✓ El Presentador escucha los eventos que se producen en la Vista y ejecuta las acciones necesarias a través del modelo. Además puede tener acceso a las vistas a través de las interfaces que la Vista debe implementar.

De forma general la Vista se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones. El Modelo ignora la información mostrada al usuario y realiza toda la lógica de las aplicaciones usando las entidades del dominio y el Presentador se encarga de actualizar el Modelo cuando surge un evento en la Vista, pero también es responsable de actualizar la Vista cuando el Modelo le indica que ha cambiado (de la Torre Llorente, y otros, 2010).

Conjuntamente la arquitectura está dividida en tres niveles físicos por la distribución o localización de las capas lógicas involucradas en la solución, es decir: en la parte cliente coexisten las capas de Presentación y Lógica de Negocio Cliente, en otro nivel se ubica un servidor remoto con las capas de Lógica de Negocio Servidor, Lógica de Negocio de Acceso a Datos y Entidades Persistentes, y en otro servidor físico la capa de Datos alojada en un servidor donde se encuentra un SGBD y la base de datos. La Seguridad está implícita a través de todas las capas de la arquitectura. En la Figura 2 se muestra la distribución física de las capas en los niveles.

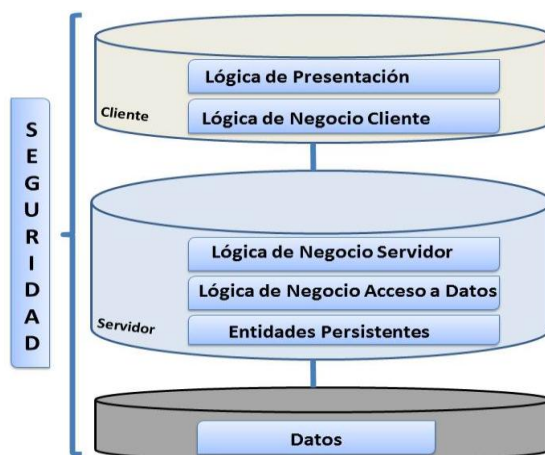


Figura 2. Distribución de las capas de la arquitectura.

A continuación se brinda una descripción detallada de las capas. Es importante tener en cuenta que las capas no están distribuidas físicamente en la misma computadora sino que están distribuidas en la red.

Capa Lógica de Presentación: es la encargada de toda la lógica de presentación que va a tener la aplicación, se encarga de presentar los datos que obtiene de la capa de

Lógica de Negocio Cliente en la interfaz gráfica de usuario. Esta capa solo se comunica con la capa de Lógica de Negocio Cliente, atendiendo a uno de los principios ya antes explicados que plantea que una capa solo se comunica con una capa vecina a ella.

Para desarrollar este proceso esta capa se apoya en un patrón de diseño llamado Modelo Vista Presentador. Fundamentalmente este patrón resuelve la separación de la interfaz gráfica de usuario y la lógica de las aplicaciones, una ventaja muy grande a la hora de la reutilización de varias de las interfaces gráficas de usuarios a lo largo del desarrollo.

Capa de Lógica de Negocio Cliente: se encarga de gestionar toda la lógica de negocio relacionada en la parte cliente, así como el acceso a las funcionalidades de los componentes en el servidor. En ella se encapsularán lógicas de acceso a dispositivos externos; lógica de comunicación con el servidor de aplicaciones para lograr la abstracción de los protocolos de comunicación y la utilización de los componentes como una llamada a objetos común y corriente, a través de la implementación de un patrón proxy.

Está implementada sobre el concepto de gestor de negocio, que agrupa las funcionalidades que se refieren a una misma entidad en un gestor con su nombre.

Capa de Lógica de Negocio Servidor: presenta una fachada a todos los componentes que se encuentran en el servidor a modo de gestores de negocio. Se encuentra alojada físicamente en un servidor de aplicaciones al cual se accede de forma remota. Presentan una interfaz y una implementación de esa interfaz la cual contiene la lógica de negocio de las funcionalidades y hace llamadas a su capa inmediata superior que es la capa de 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. Esta capa tiene mayor carga de procesamiento ya que es la encargada de procesar las peticiones que vienen desde el cliente y retornar datos formateados.

Capa de Lógica de Negocio de Acceso a Datos: es una de las capas más complejas de la arquitectura porque asocia a más de una capa para lograr su objetivo que es proveer el acceso seguro y rápido a la capa de 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. Estas operaciones se realizan en conjunto con la capa de Entidades Persistentes. En esta capa también se definen y se manejan aspectos como las propiedades de las conexiones a los SGBD, y los conjuntos de tablas que se desean

administrar, así como configuraciones para mejorar el rendimiento y la velocidad de las transacciones.

Capa Entidades Persistentes: Encargada de representar una relación entre la Base de Datos y la programación del sistema, donde cada entidad obtenida de la conversión objeto/relacional se corresponde a una tabla, a través de una especificación determinada. Ayuda a la versatilidad de la arquitectura abstrayendo del uso de un SGBD determinado, porque estas entidades se refieren a elementos genéricos del modelo de datos y no a las especificaciones de cada uno de los gestores.

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: está muy relacionada a las tecnologías que se seleccionen para la implementación, pero tiene aspectos generales que son fundamentales y deben tenerse en cuenta. La seguridad se trata de manera diferente en cada nivel físico, teniendo en cuenta sus especificaciones. En el cliente se garantiza por mecanismos de Autenticación y Autorización, que son los encargados de verificar la validez de un usuario en el sistema y de autorizar el acceso a varias o ninguna de las funcionalidades del sistema, bloqueo de operaciones no deseadas y técnicas de validación de los datos que se introducen en las aplicaciones.

En el nivel servidor se aplica por un mecanismo de publicación de los usuarios que están autenticados y autorizados en los clientes y todas las peticiones son verificadas que tengan los usuarios correctos. En el nivel de datos está protegido por el SGBD que ofrece funcionalidades para la protección de los datos, el mismo será accedido únicamente desde el servidor de aplicaciones, es decir, ningún cliente tendrá acceso al servidor de datos, así se puede aislar este de cualquier red pública y obtener mayores niveles de seguridad. Existen además, mecanismos de seguridad para la red manteniendo las conexiones seguras dentro de redes seguras.

La arquitectura se basa también en el desarrollo de software basado en componentes simplificando el mantenimiento del sistema y permitiendo alcanzar un mayor grado de reutilización, integración y seguridad.

La combinación de los estilos arquitectónicos descritos anteriormente aporta grandes ventajas a la arquitectura propuesta como el acceso centralizado a los datos; alta seguridad de los datos; clientes poco cargados de procesamiento de lógicas de negocio que permite ejecutarlos en un entorno con pocos recursos; la utilización de un

servidor de aplicaciones brinda muchas facilidades en términos de seguridad, transaccionalidad, abstracción de protocolos de red, interoperabilidad entre sistemas, alta disponibilidad de las aplicaciones y balance de carga.

La arquitectura en capas brindará descomposición de servicios que facilitará la interacción entre las capas; así como la ubicación de componentes en las mismas (Fadraga Artiles, y otros, 2012). La Figura 3 agrupa todos los conceptos descritos anteriormente, en una vista integrada para su mejor entendimiento.

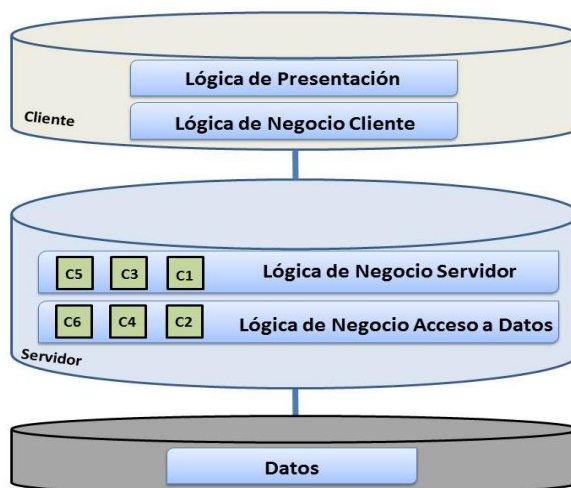


Figura 3. Vista integrada de la arquitectura de SIGFE.

2.4 Patrones de diseño

Durante el desarrollo de un software surgen con frecuencia problemas que suelen tener como factor común una misma solución. Tales soluciones cuando son definidas de modo correcto para un problema dado, describiendo sus cualidades, permitiendo su reutilización en posteriores soluciones y evitando la búsqueda reiterativa a problemas ya conocidos y solucionados anteriormente, son conocidas como patrones.

En la programación orientada a objetos, un patrón *describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma* (Ishikawa, y otros, 1977).

A continuación se mencionan algunos de los patrones que son utilizados en el diseño de los módulos.

Patrones GRASP: Es el acrónimo de General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) y se encargan de describir los principios fundamentales de la asignación de

responsabilidades a objetos (Larman, 1999). Entre los patrones GRASP utilizados se encuentran:

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.

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 el 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, gestoras de negocio servidoras.

Controlador: Sirve como intermediario entre una determinada interfaz gráfica de usuario y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que 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.

Alta cohesión: Se evidencia en el diseño de cada una de las clases del sistema garantizando que solo existan en ellas las características y funcionalidades necesarias. Entre estas clases se encuentran las clases gestoras de negocio cliente y gestoras de negocio servidoras.

Patrones GoF (Gang of Four) o Pandilla de los Cuatro: Describen las clases y objetos que se comunican entre sí, y se adaptan para resolver un problema general de diseño en un contexto particular (Hernández, 2009). A continuación se explican los patrones GoF utilizados.

Instancia única o Singleton: 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.

Iterador: Proporciona una forma de acceder secuencialmente a los elementos de un objeto compuesto por agregación sin necesidad de desvelar su representación interna. Se utiliza en las clases gestoras de acceso a datos para recorrer las estructuras de datos utilizadas, durante la conversión de las entidades de dominio a entidades persistentes. Este patrón permite garantizar una integridad en la información.

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 la facilitar la conexión entre los gestores clientes y servidores.

2.5 Modelo de negocio

2.5.1 Descripción de procesos de negocio

A partir de la descripción realizada en el epígrafe Conceptos fundamentales, a continuación se realiza la descripción de los procesos convocatoria y contratación, pues constituyen el área clave de la solución que propone este trabajo.

Tabla 2. Descripción del proceso de negocio Convocatoria.

Objetivo	Mantener un registro de los eventos que ha organizado y en los que ha participado la Cámara de Comercio de Cuba.
Evento(s) que lo genera(n)	Decisión de organizar o participar en un evento.
Pre condiciones	N/A.
Marco legal	N/A.
Clientes internos	N/A.
Clientes externos	Entidades cubanas y extranjeras interesadas.
Entradas	Datos del evento.
Flujo de eventos	
Flujo básico Convocatoria	
1. Registrar evento. Se registran en un sistema informático los datos del evento y de la entidad organizadora.	
2. Convocar evento. Se convoca, mediante los medios que decida la Cámara de Comercio de Cuba, a las entidades potencialmente interesadas en participar en el evento.	
3. Concluye el proceso.	
Pos-condiciones	
<ul style="list-style-type: none"> • Se registra un nuevo evento. • Se registra una nueva entidad organizadora. • Se creó una relación entre el evento y la entidad organizadora. 	
Salidas	
1. Registro de eventos (BD relacional).	
Asuntos pendientes	
N/A.	

Tabla 3. Descripción del proceso de negocio Contratación.

Objetivo	Registrar los acuerdos contraídos entre la entidad participante en la feria o evento y la Cámara de Comercio, así como las obligaciones de pago existentes.
Evento(s) que lo genera(n)	Recepción de una solicitud de participación en feria o exposición.
Pre condiciones	Se ha convocado al evento.
Marco legal	N/A.
Clientes internos	N/A.
Clientes externos	Entidades cubanas y extranjeras interesadas.
Entradas	Formulario de inscripción.
Flujo de eventos	
Flujo básico Contratación concluida	
4. Registrar interesado. Se registran en un sistema informático los datos de las entidades interesadas en participar en el evento, así como la solicitud de espacios y medios.	
5. Asignar espacios y medios. Se acuerda con el interesado la asignación de espacios y medios.	
6. Concluye el proceso.	
Pos-condiciones	
<ul style="list-style-type: none"> • Se registra un nuevo interesado. 	
Salidas	
2. Registro de entidad (BD relacional).	
Flujos alternos	
2.a No se alcanzó un acuerdo respecto al contrato.	
1. Concluye el sub-proceso.	
Pos-condiciones	
1. N/A.	
Salidas	
1. N/A.	
Asuntos pendientes	
N/A.	

2.5.2 Modelo de proceso de negocio

Partiendo de la descripción anterior se muestran los modelos de procesos del negocio correspondiente a la Convocatoria y Contratación de eventos.

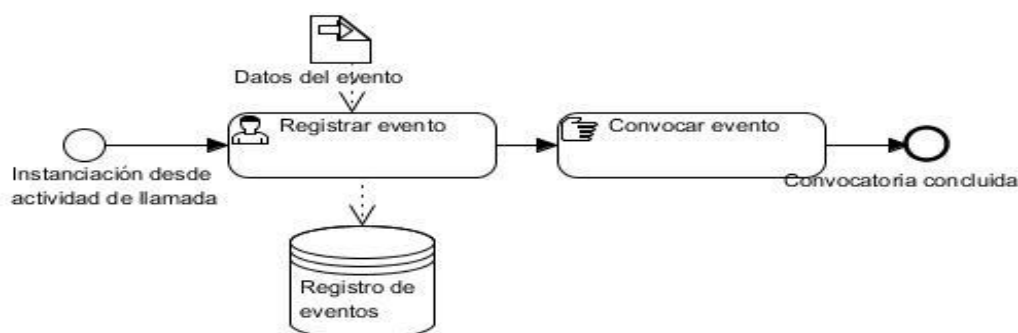


Figura 4. Modelo de proceso de negocio de la Convocatoria.

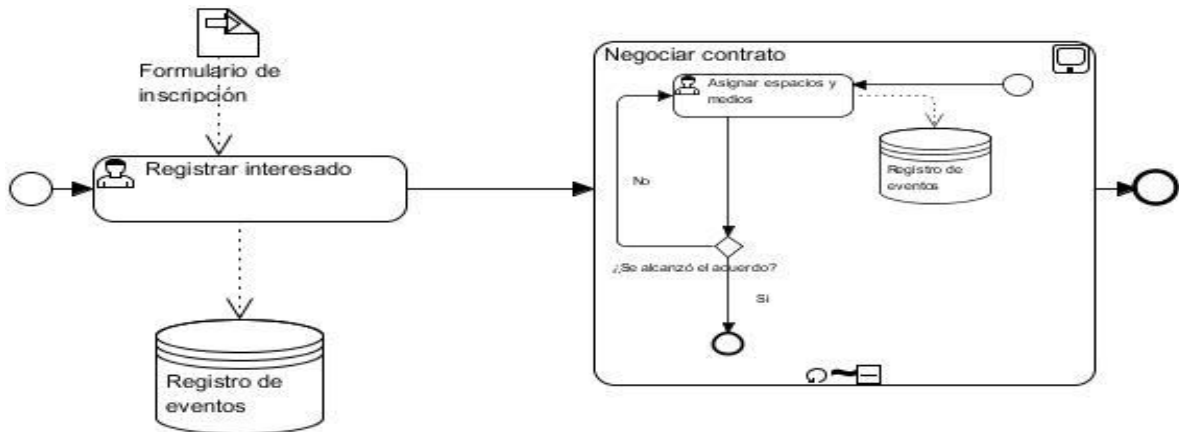


Figura 5. Modelo de proceso de negocio de la Contratación.

Dentro del proceso antes modelado se desarrolla el subproceso negociar contrato, del cual se muestra su diagrama en la Figura 6.

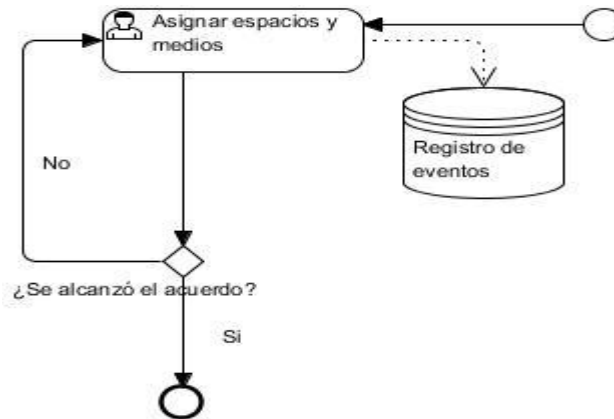


Figura 6. Modelo de proceso de negocio: Subproceso Negociar Contrato.

2.6 Especificación de los requisitos de software

Establecer una especificación de requisitos completa y consistente, constituye un paso muy importante para evitar cometer errores en la definición de los requisitos, ya que los mismos pueden resultar muy caros de corregir una vez desarrollado el sistema y de esta manera se estaría evitando que el proyecto fracase.

Según el libro: “El proceso unificado de desarrollo de software” de Ivar Jacobson, Grady Booch y James Rumbaugh la Especificación de los requisitos es *el proceso de averiguar, normalmente en circunstancias difíciles, lo que se debe construir con el propósito de guiar el desarrollo hacia el sistema correcto* (Jacobson, y otros, 2000).

Para lograr un buen proceso de elicitación de requisitos y teniendo en cuenta las características del cliente, se decidió utilizar algunas técnicas de recopilación de información como: estudio y análisis de sistemas existentes, revisión documental, talleres y tormenta de ideas.

En un primer momento se realizó un estudio crítico sobre algunas de las aplicaciones de gestión de ferias y exposiciones existentes, centrándose especialmente en el SAEFE por ser el sistema que tiene actualmente la Cámara de Comercio. La revisión documental permitió obtener información de vital importancia para el negocio en cuestión. Los talleres se llevaron a cabo en diferentes sesiones de trabajo, presididos por el jefe de proyecto, la analista principal y el arquitecto del sistema y donde además asistieron los diseñadores de la base de datos, y analistas implicados en el asunto, logrando captar en gran medida las particularidades y validaciones del negocio, así como un mejor entendimiento de los procesos que se manejan y de los módulos a implementar. La tormenta de ideas es una técnica que se combinó en los talleres donde se acumularon ideas de los participantes, anotándose y respetándose cada criterio y opinión para tener una perspectiva general de las necesidades del sistema.

2.6.1 Requerimientos funcionales

Los requerimientos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas específicas y de cómo se debe comportar en situaciones particulares (Sommerville, 2007).

A continuación se exponen algunos de los requisitos funcionales más importantes de los módulos a desarrollar. Para consultar los restantes (Ver Anexo 5).

Tabla 4. Requisitos funcionales más importantes.

Requisitos Funcionales	
Registrar evento	Eliminar área a la solicitud de recursos
Modificar evento	Eliminar recurso ferial de la solicitud de recursos
Eliminar evento	Eliminar solicitud de recursos
Consultar evento	Consultar solicitud de recursos
Buscar evento	Buscar solicitud de recursos
Registrar entidad	Registrar asignación de recursos
Modificar entidad	Modificar asignación de recursos
Eliminar entidad	Adicionar área a la asignación de recursos
Consultar entidad	Adicionar recurso ferial a la asignación de recursos
Buscar entidad	Modificar área a la asignación de recursos

Búsqueda avanzada de entidad	Modificar recurso ferial a la asignación de recursos
Registrar solicitud de recursos	Eliminar área a la asignación de recursos
Modificar solicitud de recursos	Eliminar recurso ferial de la asignación de recursos
Adicionar área a la solicitud de recursos	Eliminar asignación de recursos
Adicionar recurso ferial a la solicitud de recursos	Consultar asignación de recursos
Modificar área a la solicitud de recursos	Buscar asignación de recursos
Modificar recurso ferial a la solicitud de recursos	

2.6.2 Requerimientos no funcionales

Los requisitos no funcionales (RNF) son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. A menudo son aplicados en su totalidad al sistema y normalmente apenas se aplican a características o servicios individuales del sistema (Sommerville, 2007).

A continuación se exponen los requisitos no funcionales definidos para el sistema atendiendo a sus clasificaciones:

Usabilidad:

RNF_1. Facilidad de aprendizaje.

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

RNF_2. Facilidad de uso.

- ✓ Las interfaces de usuario indican los campos requeridos y muestran ejemplos de los valores que deben introducirse en cada campo.
- ✓ Los menús permitirán una navegación sencilla, tanto a los usuarios con conocimientos avanzados de informática como a los usuarios más inexpertos.
- ✓ La ayuda es sensible al contexto e incluye información sobre el proceso de negocio.

RNF_3. Mínimo impacto de los errores.

- ✓ El sistema permite cancelar los flujos y sus tareas.

- ✓ El sistema notifica a los usuarios los errores y sugiere cómo corregirlos.

Fiabilidad:

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

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

Eficiencia:

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

Software:

RNF_6. Servidores de base de datos locales PostgreSQL versión 9.1

RNF_7. Máquina Virtual de Java en su versión JDK 1.6.

Hardware:

RNF_8. Hardware mínimo para oficinas:

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

RNF_9. Hardware mínimo para servidores:

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

Soporte:

RNF_10. Portabilidad

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

Restricciones de diseño:

RNF_11. Restricciones de diseño

- ✓ Herramienta de modelado: 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 obtienen en los flujos de trabajo y sus diferentes fases. Las restricciones propias del diseño radican en las pautas que se establecerán, así como las diferentes relaciones que se formen durante el modelado.

- ✓ Lenguaje de programación: El software estará programado en Java, siguiendo una codificación estándar y organizada, haciendo uso de las potencialidades propias del lenguaje para implementar los diferentes procesos.
- ✓ Entorno de desarrollo integrado: El software se desarrollará sobre Netbeans 7.2 ya que contiene las herramientas para llevar a efecto la implementación de la totalidad de los componentes impuestos por la arquitectura y el diseño.
- ✓ Restricciones arquitectónicas: El diseño tomará como punto de partida la línea base de la arquitectura y las restricciones que esta impone para el desarrollo, tal es el caso de los componentes y sus conectores. Igualmente incorporará las bibliotecas de clases y marco de trabajo establecidos en dicho documento para el trabajo adecuado de la arquitectura según las cualidades que se desean obtener en ella.

Seguridad:

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

RNF_13. Las contraseñas se almacenarán en la base de datos haciendo uso del algoritmo hash MD5¹³.

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

RNF_15. Solo se podrá acceder a la aplicación desde las direcciones IP¹⁴ autorizadas.

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

Estándares aplicables:

RNF_18. Se aplicarán los estándares legales, de calidad, internacionalización y regulatorios definidos por la empresa comercial que provee el sistema. Para una mejor comprensión de los requerimientos de los módulos, se recomienda consultar el artefacto **CEGEL_SIGFE_0113_ERS_Convocatoria y Contratación**.

¹³ MD5: abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5, es un algoritmo de reducción criptográfico.

¹⁴ es un número que identifica de manera lógica y jerárquica a una interfaz de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo IP (Internet Protocol).

2.7 Definición de casos de uso del sistema

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 e implementación del sistema.

2.7.1 Definición de los actores

Tabla 5. Actores del sistema.

Actor	Justificación
Usuario	Es el usuario común del sistema.
Organizador de feria	Es el encargado de registrar y consultar la información relativa a la organización, ejecución y seguimiento a las ferias y exposiciones comerciales.
Diseñador de feria	Es el encargado de configurar, asignar y negociar los espacios y recursos feriales.

2.7.2 Listado de casos de uso

Por la importancia que presenta el caso de uso *Gestionar Solicitud de recursos* será analizado a lo largo del capítulo. En la Tabla 6 se detalla brevemente el mismo. Para consultar los demás, ver Anexo 10.

Tabla 6. Caso de uso *Gestionar Solicitud de recursos*.

CU_1	Gestionar Solicitud de recursos
Actor	Caso de uso Extendido
Descripción	Permite gestionar solicitudes de recursos (áreas y recursos feriales) a las entidades participantes en los eventos. Dicha gestión consiste en registrar, modificar, consultar, buscar y eliminar una solicitud de recursos.
Referencia	RF_15, RF_16, RF_17, RF_18, RF_19, RF_20, RF_21, RF_22, RF_23, RF_24, RF_25, CU_3

2.7.3 Diagrama de casos de uso del sistema

Los diagramas de casos de uso documentan el comportamiento de un sistema desde el punto de vista del usuario. Por lo tanto los casos de uso determinan los requisitos funcionales del sistema, es decir, representan las funciones que un sistema debe ejecutar. Su ventaja principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente. La Figura 7 muestra el diagrama de casos de uso del sistema correspondiente a los módulos a desarrollar.

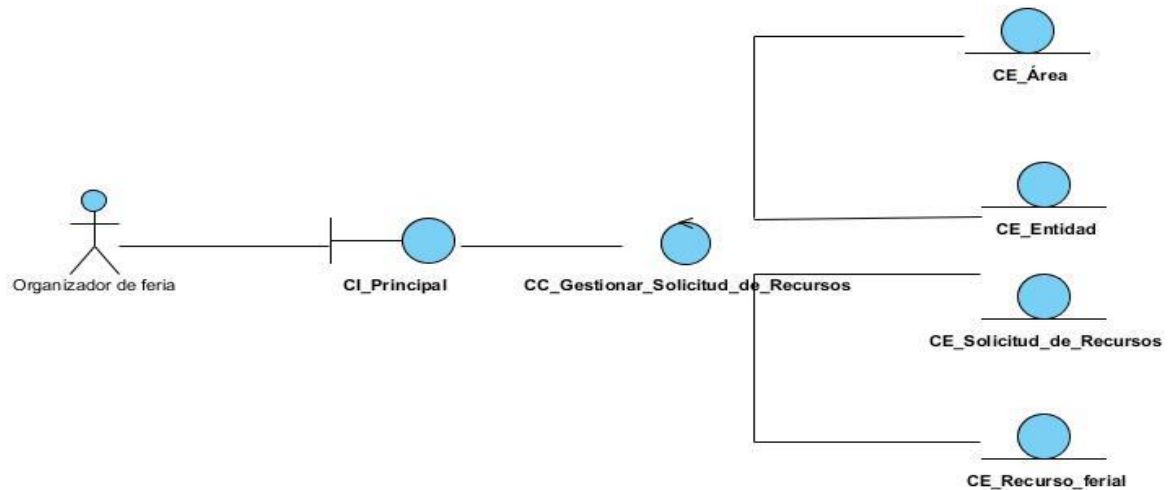


Figura 8. Diagrama de clases del análisis *Gestionar Solicitud de recursos*.

2.9 Diagrama de secuencia

Un diagrama de secuencia es un diagrama de interacción que destaca la organización temporal de los mensajes (mensaje simple, síncrono, asíncrono, y/o de retorno). Este tipo de diagrama se destaca por tener la línea de vida y el foco de control que representa el período de tiempo durante el cual un objeto ejecuta una acción. A continuación se muestra el diagrama de secuencia correspondiente al escenario *Registrar Solicitud de recursos* (Ver Figura 9). Los demás diagramas del caso de uso pueden ser observados en los anexos (Ver Anexo 6).

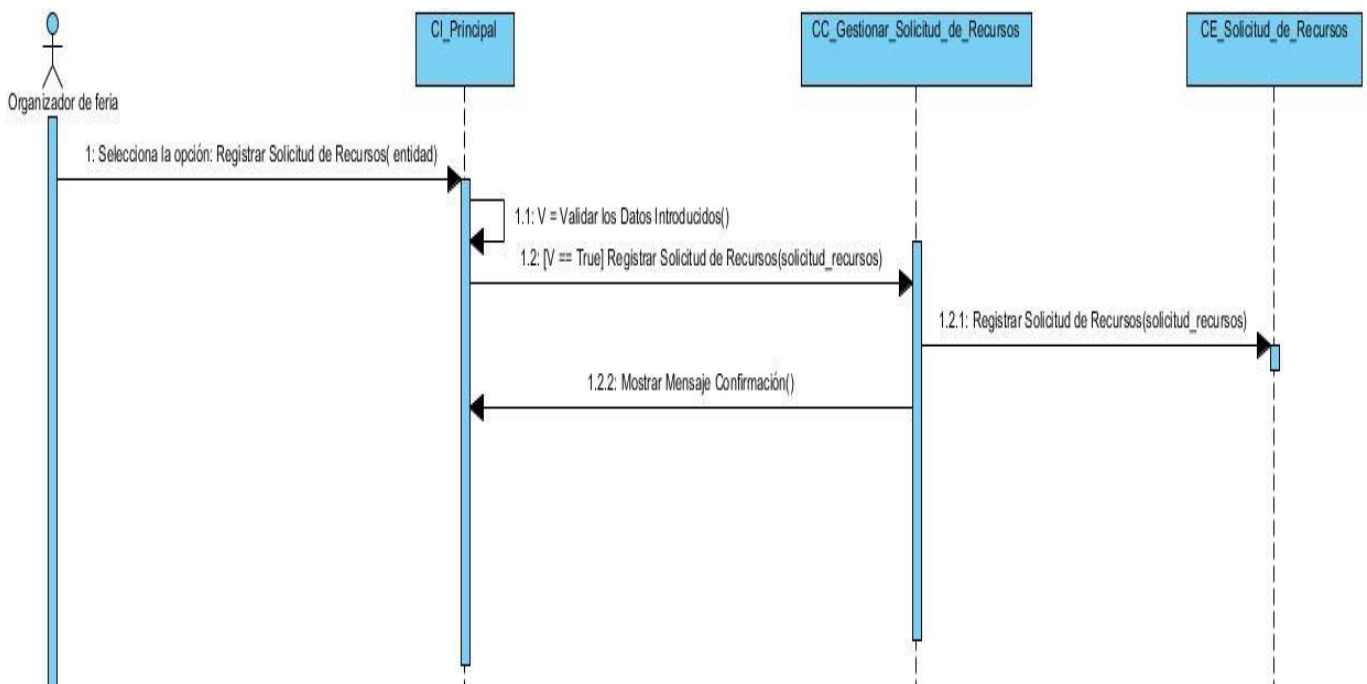


Figura 9. Diagrama de secuencia *Registrar Solicitud de recursos*.

2.10 Diagrama de clases de diseño

El diagrama de clases del diseño es una abstracción del Modelo de Implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. Representa a los casos de uso en el dominio de la solución. A continuación se muestra el diagrama de clases del diseño del caso de uso analizado. Para mayor información se puede consultar el artefacto **CEGEL_SIGFE_0121_Modelo de diseño_Convocatoria y Contratación**.

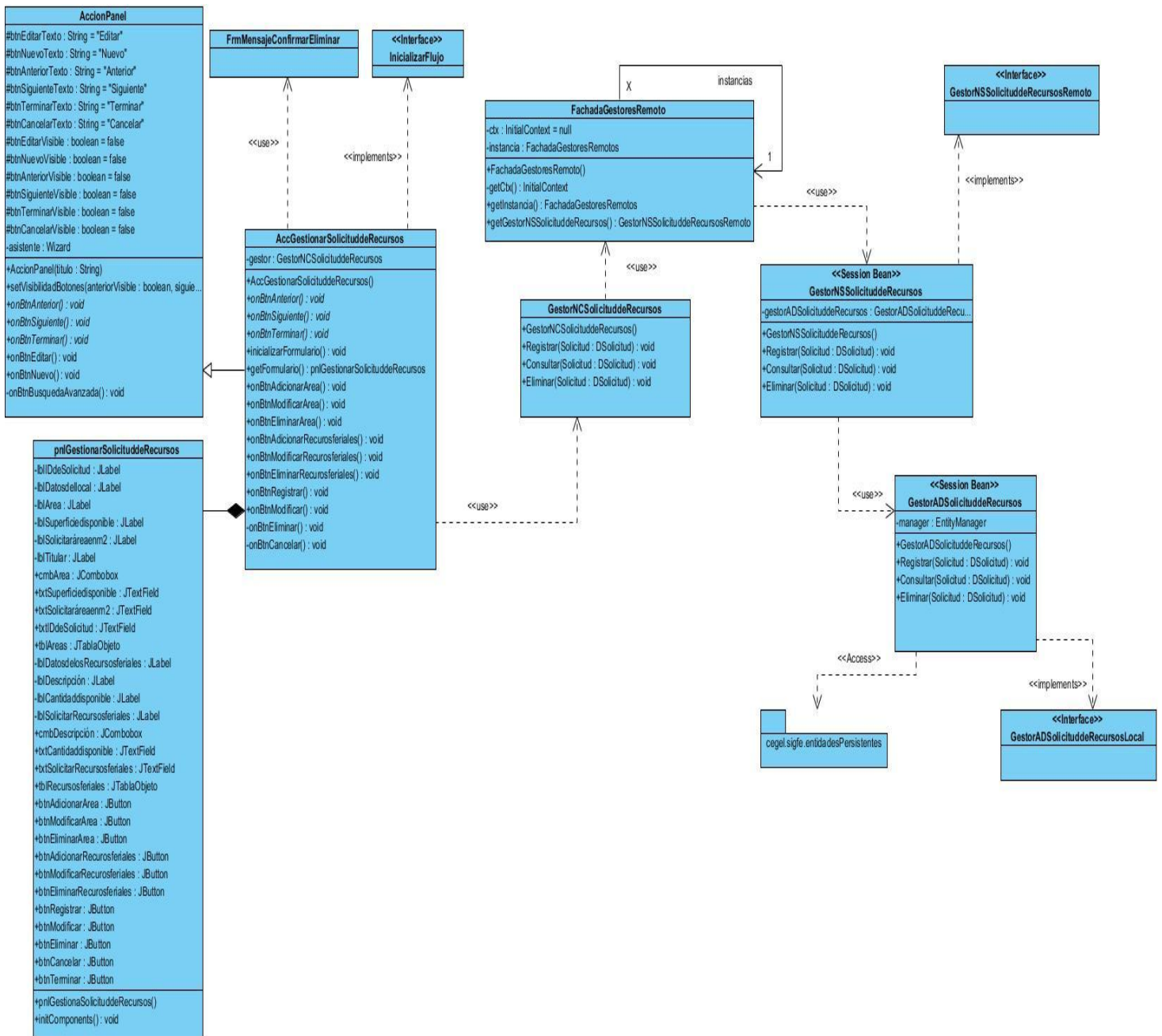


Figura 10. Diagrama de clases de diseño *Gestionar Solicitud de recursos*.

2.11 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 sistema.

- ✓ 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 asociados a una misma cosa se agruparán utilizando paneles con título.
- ✓ Cuando el formulario tiene varios componentes, el nombre de las etiquetas se escribe encima 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 debajo, a una distancia de 8 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. Todo alineado a la izquierda, y en caso de que aparezcan unos debajo de los otros se debe tratar de que todos los componentes tengan un mismo tamaño (el del mayor componente).
- ✓ En caso de que el formulario tenga pocos componentes, se pondrá primero la etiqueta y al lado el componente, tanto los componentes como las etiquetas con las mismas pautas anteriores. Además se alinean las etiquetas a la izquierda y los componentes también a la izquierda.
- ✓ Las listas desplegables deben mostrar como texto inicial –Selecione--. Los ítems para seleccionar deben escribirse en formato de oración y deben aparecer en orden alfabético, a no ser que representen un flujo, en ese caso aparecerían de acuerdo al orden correspondiente. Cuando haya una opción Otros, estará ubicada al final.
- ✓ Cuando haya que introducir una fecha debe ponerse un componente para seleccionar la misma, evitar las entradas manuales del usuario. Los botones siempre estarán situados en la parte inferior derecha de los formularios, 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.

2.12 Diseño de la base de datos

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

2.12.1 Modelo de datos

El modelo de datos es usado para describir la representación lógica y física de la información persistente manejada por el sistema. La Figura 11 muestra el submodelo que agrupa las tablas asociadas al caso de uso analizado.

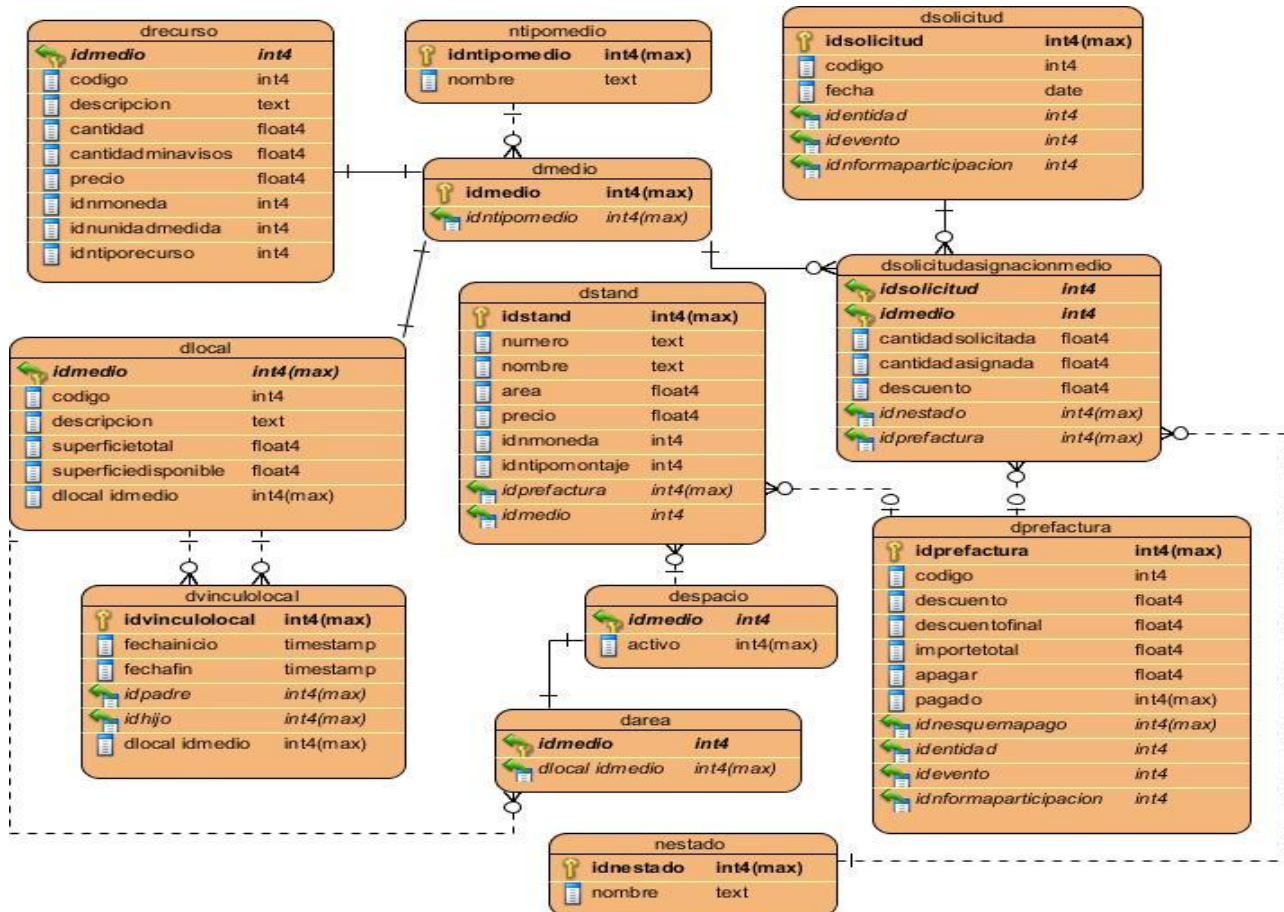


Figura 11. Submodelo físico de la base de datos: Sección Solicitud.

Para la confección del Modelo de Datos se tuvieron en cuenta patrones de diseño de bases de datos de acuerdo con (Blaha, 2010), entre los que se encuentran:

- ✓ Patrón llave subrogada: El patrón llave subrogada es utilizado para definir una llave primaria independiente del contexto para las entidades *dRonda*, *dActividad*, *dEvento*, *dLocal*, *nCategoriaParticipante* y *nOrganismo* en lugar de emplear el atributo identificador que poseen según el negocio. Para definir la columna identidad de cada entidad se utilizan valores enteros secuenciales.
- ✓ Patrón árbol cambiante en el tiempo: El empleo de este patrón ofrece una solución reusable y robusta para almacenar los datos históricos referentes a la división político administrativa así como su evolución en el tiempo. De esta manera se garantiza que










los datos relacionados con una entidad no se pierdan incluso cuando esta cambie o deja de existir. Se utiliza en la tabla *ddpa* para el tratamiento de la división política administrativa y en la entidad *ddirección* para el manejo de las direcciones de los participantes y entidades.

✓ Patrón árbol fuertemente codificado: La estructura definida por el Departamento de Ferias y Eventos de la Cámara de Comercio para realizar los procesos de solicitud y asignación de espacios está bien definida. Es por ello que se decide utilizar este patrón para representar la jerarquía existente entre las entidades *dlocal*, *darea* y *despacio*, así como para asegurar la secuencia de tipos en cada nivel.

2.12.2 Descripción de las tablas

La descripción de las tablas muestra de forma sintetizada el objetivo que persigue la representación en la base de datos de cada entidad. En la Tabla 7 se documentan las tablas de la base de datos relacionadas con el caso de uso analizado.

Tabla 7. Tablas de la base de datos asociadas a la sección Solicitud.

Nombre	Descripción
 drecurso	Entidad que almacena la información de los recursos, tales como su disponibilidad, cantidad existente, etc.
 dsolicitud	Entidad que almacena la solicitud de espacios y recursos feriales que realiza una entidad en un evento determinado a la Cámara de Comercio.
 dlocal	Almacena los datos de los locales como nombre local, superficie total, superficie disponible, etc.
 dstand	Representa los datos de los stands que se crean para ser asignados a las entidades en los eventos.
 dsolicitudasignacionmedio	Entidad que almacena la información del proceso de solicitud y asignación de recursos.
 despacio	Almacena el espacio solicitado por la entidad en un evento determinado.
 dprefactura	Representa la información referente a las pre-facturas generadas para las entidades inscritas en un evento.
 darea	Tabla que almacena los datos de las áreas para asignar a una entidad en un evento (superficie total, disponible, y el local al que pertenece).
 nestado	Nomenclador que almacena el estado de la solicitud (solicitado, asignado, pre-facturado).

2.13 Validación del diseño del sistema

A continuación se muestran los resultados obtenidos tras la aplicación de las métricas definidas en el capítulo para la validación del diseño de la propuesta de solución.

Tabla 8. Aplicación la métrica TOC.

No.	Clases	Cantidad de atributos	Cantidad de operaciones	Tamaño
1	Acc_Panel	6	8	Pequeño
2	Acc_GestionarEntidad	7	19	Medio
3	Acc_RegistrarEntidad	7	15	Medio
4	Acc_ModificarEntidad	7	12	Pequeño
5	Acc_BuscarEntidad	7	13	Pequeño
6	Acc_ConsultarEntidad	7	11	Pequeño
7	Acc_BusquedaAvanzadaEntdad	7	15	Medio
8	FachadaGestoresRemoto	2	5	Pequeño
9	GestorNegocioClienteEntidad	0	7	Pequeño
10	GestorNegocioServidorEntidad	1	7	Pequeño
11	GestorAccesoDatosEntidad	1	7	Pequeño
12	Acc_GestionarSolicitudRecursos	7	20	Medio
13	GestorNegocioClienteSolicituddeRecursos	0	4	Pequeño
14	GestorNegocioServidorSolicituddeRecursos	1	4	Pequeño
15	GestorAccesoDatosSolicituddeRecursos	1	4	Pequeño
16	Acc_GestionarAsignacionRecursos	7	22	Medio
17	GestorNCAsignacionRecursos	0	4	Pequeño
18	GestorNSAsignacionRecursos	1	4	Pequeño
19	GestorAccesoDatoAsignacionRecursos	1	4	Pequeño
20	Dentidad	8	11	Pequeño
21	DSolicitudRecursos	6	10	Pequeño
22	DSolicitud_Asignacion	8	11	Pequeño
		Total =92	Total = 217	

Se trabajó con un total de 22 clases para un promedio de 4.18 atributos por clases y 9.86 operaciones, lo que arrojó como resultado los datos que en la Figura 12 se muestran:

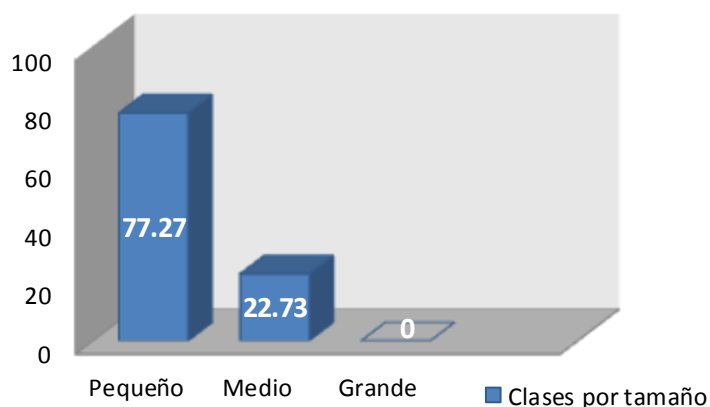


Figura 12. Resultados de la métrica TOC aplicada al diseño de la solución.

El gráfico anterior muestra que el 77.27 % de las clases tienen tamaño pequeño, lo que implica un resultado positivo según los parámetros de calidad propuestos para esta métrica, arrojando como resultado que en el diseño realizado las clases tienen bajas responsabilidades, aumentando la reutilización y facilitando la implementación y las pruebas.

Tabla 9. Aplicación de la métrica RC.

Clases	Valor RC	Acoplamiento	Complejidad de Mantenimiento	Reutilización	Cantidad de Pruebas
Acc_Panel	6	Alto	Alta	Baja	Alta
Acc_GestionarEntidad	3	Alto	Media	Media	Media
Acc_RegistrarEntidad	2	Medio	Baja	Alta	Baja
Acc_ModificarEntidad	2	Medio	Baja	Alta	Baja
Acc_BuscarEntidad	2	Medio	Baja	Alta	Baja
Acc_ConsultarEntidad	2	Medio	Baja	Alta	Baja
Acc_BusquedaAvanzadaEntidad	2	Medio	Baja	Alta	Baja
FachadaGestoresRemoto	2	Medio	Baja	Alta	Baja
GestorNegocioClienteEntidad	7	Alto	Alta	Baja	Alta
GestorNegocioServidorEntidad	2	Medio	Baja	Alta	Baja
GestorAccesoDatosEntidad	1	Bajo	Baja	Alta	Baja
Acc_GestionarSolicitudRecursos	3	Alto	Media	Media	Media
GestorNegocioClienteSolicituddeRecursos	2	Medio	Baja	Alta	Baja
GestorNegocioServidorSolicituddeRecursos	2	Medio	Baja	Alta	Baja
GestorAccesoDatosSolicituddeRecursos	1	Bajo	Baja	Alta	Baja
Acc_GestionarAsignacionRecursos	3	Alto	Media	Media	Media
GestorNCAsignacionRecursos	2	Medio	Baja	Alta	Baja
GestorNSAsignacionRecursos	2	Medio	Baja	Alta	Baja
GestorAccesoDatoAsignacionRecursos	1	Bajo	Baja	Alta	Baja
Dentidad	1	Bajo	Baja	Alta	Baja
DSolicitudRecursos	1	Bajo	Baja	Alta	Baja
DSolicitud_Asignacion	1	Bajo	Baja	Alta	Baja
	50				

Los resultados alcanzados luego de aplicar la métrica se evidencian en la Tabla 10.

Tabla 10. Resultados de los atributos de calidad de la métrica RC.

Cantidad de Clases: 22	Ninguno	Bajo	Medio	Alto
Acoplamiento	0%	50%	27%	23%
Complejidad de mantenimiento	0%	77%	14%	9%

Cantidad de pruebas	0%	77%	14%	9%
Reutilización	0%	9%	14%	77%

Considerando que la media de relaciones entre clases es 2.27 y que un alto porcentaje de las clases (77.27%) tienen un número de relaciones menor que este valor, se puede afirmar que el modelo presenta un bajo nivel de colaboraciones por clase. De igual forma muestra un bajo nivel de complejidad de mantenimiento, una baja cantidad de pruebas y un alto nivel de reutilización, evidenciando la realización de un mínimo de pruebas sobre el modelo, la posibilidad de reutilización a gran escala y de un mantenimiento poco complejo.

2.14 Conclusiones del capítulo

En este capítulo se abordaron las principales actividades desarrolladas en los flujos análisis y diseño. Se detallaron las características fundamentales de la propuesta de solución para comprender mejor las acciones a desarrollar por los módulos Convocatoria y Contratación.

Con el análisis y diseño de la solución se obtuvieron 55 requisitos funcionales para el sistema modelado y se generaron los artefactos definidos por la metodología utilizada, obteniéndose entre ellos la Especificación de Requisitos de Software, los diagramas de Clases del Análisis, Secuencia, Prototipos de Interfaz gráfica de usuario, entre otros.

Estos artefactos, unido al buen uso de los patrones de diseño: experto, alta cohesión, bajo acoplamiento, Fachada, Proxy y Singleton permitieron sentar las bases para la implementación del software con un alto grado de calidad.

Capítulo **3** Implementación y prueba

3.1 Introducción

La implementación comienza luego del resultado del diseño y construyendo el sistema en términos de componentes, es decir, ficheros de código fuente y ejecutables. Gran parte de la arquitectura del sistema es definida durante el diseño, siendo el propósito fundamental de la implementación el desarrollar la arquitectura y el sistema como un todo.

Este capítulo está orientado a la implementación y prueba de la solución. Se hará referencia al modelo de implementación, el cual está formado por los diagramas de componentes y el de despliegue. Además se verifica la calidad del resultado de la implementación, mediante los artefactos generados durante el flujo de trabajo de pruebas, exponiendo los resultados obtenidos por diferentes tipos de pruebas realizadas a los módulos.

3.2 Modelo de implementación

El Modelo de Implementación representa la composición física de la implementación en términos de subsistemas de implementación y elementos de implementación. Describe cómo los elementos de diseño se implementan en componentes (Pressman, 2005).

Según los mismos autores (Jacobson, y otros, 2000), se considera el artefacto más significativo del flujo de trabajo de Implementación, debido a la importancia que tiene para los desarrolladores comprender el funcionamiento del sistema desde el punto de vista de componentes y sus relaciones. Está compuesto por los diagramas de despliegue y componentes.

3.2.1 Diagrama de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Desde el punto de vista del diagrama de componentes se tienen en

consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo (Rosales Velazquez, 2009). A continuación se muestra el diagrama de componentes de los módulos. Para una mejor comprensión del mismo, ver Anexo 11, donde se encuentra dividido en las capas de la arquitectura.

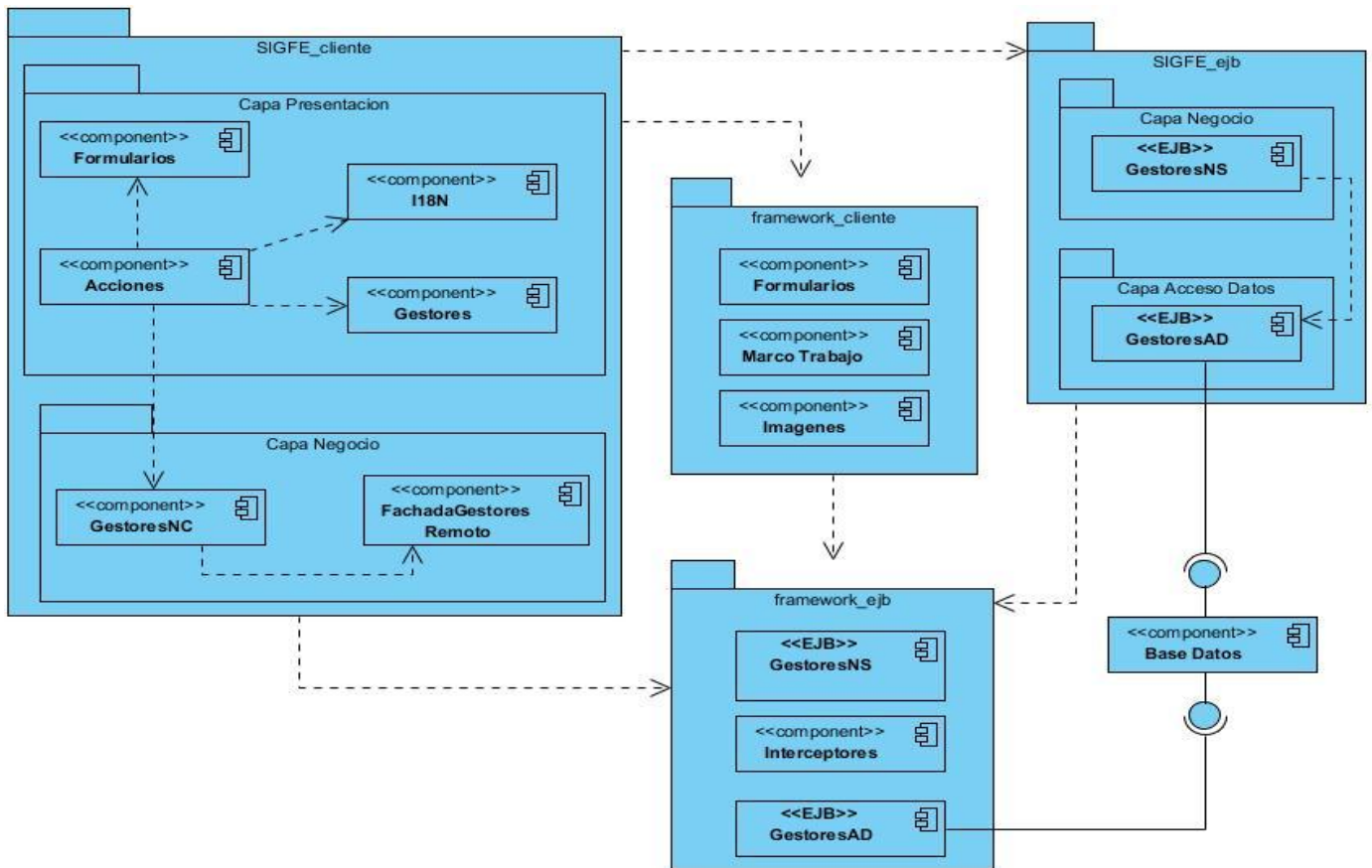


Figura 13. Diagrama de componentes de los módulos.

3.2.2 Diagrama de despliegue

Los diagramas de despliegue muestran las relaciones físicas entre los componentes del hardware y el software, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes del software. También un diagrama de despliegue es un grafo de nodos, unidos por conexiones de comunicación, donde un nodo puede contener instancias de componentes, un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria (Marca Huallpara, y otros, 2010). En la Figura 14 se muestra el diagrama de despliegue del sistema.

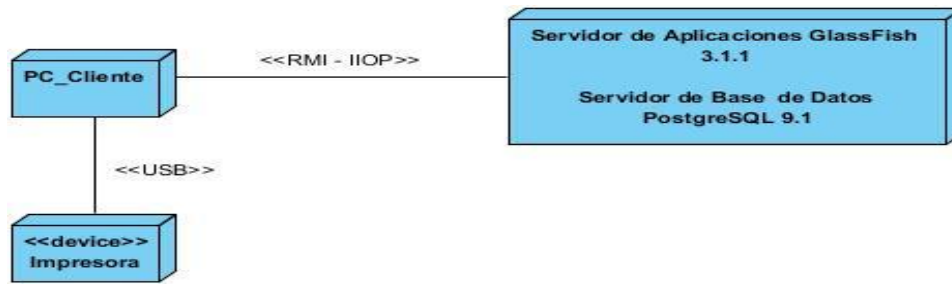


Figura 14. Diagrama de despliegue del sistema.

- ✓ **Nodo PC Cliente:** Representa las computadoras que utilizarán los usuarios para interactuar con la aplicación. Establece comunicación con el servidor de Aplicaciones y con el servidor de Base de Datos a través del protocolo RMI-IIOP (Remote Method Invocation) – (Internet Inter-ORB Protocol) que proporciona comunicación remota en aplicaciones distribuidas e interoperabilidad de distintos lenguajes con objetos CORBA¹⁵.
- ✓ **Nodo Impresora:** Representa las impresoras, que serán utilizadas para la impresión de catálogos, pre-facturas, programas de eventos, entre otros documentos que se necesiten.
- ✓ **Nodo Servidor:** Representa un servidor físico, en el que se encuentran virtualizados el servidor de aplicaciones GlassFish en su versión 3.1.1 y el Servidor de Base de datos PostgreSQL en su versión 9.1.

3.3 Estándares de codificación

Los estándares de codificación definen un conjunto de reglas para escribir el código, los cuales permiten obtener un código de alta calidad y cumplir con las buenas prácticas establecidas en la Ingeniería de Software. Además posibilita que el software que se obtiene sea fácil de comprender. A continuación se describen los estándares de codificación que se utilizarán para el desarrollo del sistema.

3.3.1 Nomenclatura para los paquetes, métodos y clases

- ✓ Los paquetes deberán comenzar con el nombre del sistema (sigfe-) seguido por el subsistema, luego el módulo y por último la capa lógica a la que pertenece.

¹⁵ **Common Object Request Broker Architecture** (CORBA) es un estándar definido por el Object Management Group (OMG) que permite que diversos componentes de software escritos en múltiples lenguajes de programación y que corren en diferentes computadoras puedan trabajar juntos.

- ✓ Se utilizará la notación Camello¹⁶ en los nombres de las entidades persistentes, entidades de dominio, propiedades y funcionalidades del sistema.
- ✓ Las clases persistentes comenzarán con las siglas definidas en la base de datos para cada tabla.
- ✓ Los formularios de tipo ventanas emergentes 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.
- ✓ Las clases interceptoras de validación terminan con el prefijo Interceptor.
- ✓ Las excepciones terminan con el sufijo Exception.

3.3.2 Nomenclatura para los componentes de formularios

- ✓ Los botones (JButton) comienzan con las siglas btn.
- ✓ Los campos de texto (JTextField) comienzan con las siglas txt.
- ✓ Los campos de fecha (Date) comienzan con las siglas fch. Para el componente JDateChooser con las siglas dtc.
- ✓ Para el componente JDayChooser con las siglas dyc.
- ✓ Las listas desplegadas (JComboBox) comienzan con las siglas cmb.
- ✓ Las áreas de texto (JTextArea) comienzan con las siglas txa.
- ✓ Las etiquetas (JLabel) comienzan con las siglas lbl.
- ✓ Los cuadros de chequeo (JCheckBox) comienzan con las siglas chb.
- ✓ Los botones de opción (JRadioButton) comienzan con las siglas rbt.
- ✓ Las tablas (JTable) comienzan con las siglas tbl.

3.4 Validación de la solución

Como parte del proceso de desarrollo del software, la fase de pruebas añade valor al producto que se maneja: todos los programas tienen errores y esta tiene como objetivo específico encontrarlos. Las pruebas de software son los procesos que verifican y revelan la calidad de un producto de software. Constituyen un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa, involucrando las operaciones del sistema bajo condiciones controladas y evaluando

¹⁶ es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en esta notación, se asemejan a las jorobas de un camello.

los resultados, es por eso que la realización de las mismas es un factor de vital importancia. Para la validación de la solución propuesta se utilizarán los métodos de prueba de Caja Blanca y Caja Negra.

Entre las técnicas de Caja Blanca será utilizada la técnica “Camino Básico” que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Para aplicar esta prueba se han definido una serie de pasos a seguir que a continuación se describen:

1. **Notación del grafo de flujo:** usando el código como base se realiza la representación del grafo de flujo (grafo del programa¹⁷), mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
 - **Nodo:** cada círculo, denominado nodo, representa una o más sentencias procedimentales.
 - **Arista:** las flechas del grafo de flujo, denominadas aristas, representan flujo de control y son análogas a las flechas del diagrama de flujo.
 - **Región:** Las áreas delimitadas por aristas y nodos se denominan regiones.
2. **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes¹⁸ del conjunto básico de un programa y proporciona el límite superior de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez. Se calcula de tres formas:
 - El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
 - La complejidad ciclomática, $V(G)$, de un grafo de flujo G se define como:
 $V(G) = A - N + 2$, donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
 - La complejidad ciclomática, $V(G)$, de un grafo de flujo G también se define como
 $V(G) = P + 1$, donde P es el número de nodos predicado contenidos en el grafo de flujo G .

¹⁷ Sirve como herramienta útil para ilustrar el método.

¹⁸ Camino del programa que introduce un nuevo conjunto de sentencias de proceso o una nueva condición. Está constituido por al menos una arista que no haya sido recorrida anteriormente a la definición del camino.

3. **Determinar un conjunto básico de caminos linealmente independientes:** El valor de $V(G)$ permite determinar el número de caminos linealmente independientes de la estructura de control del programa.
4. **Obtención de casos de prueba:** se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico (Aguilera Revuelta, y otros, 2012).

Aplicación de la prueba:

La Figura 15 muestra el código tomado como base para realizar el procedimiento anteriormente descrito, correspondiendo este al método *Registrar Evento*.

```

public void RegistrarEvento() // Opcion Aceptar
{
    if(datosValidos() && !recursos.isEmpty() && existeEntidad) { 1
        String Codigo = this.getFormulario().txtCodigo.getText(); 2
        String Nombre = getFormulario().txtNombre.getText(); 2
        String Siglas = getFormulario().txtSiglas.getText(); 2
        Date FechaInicio = getFormulario().txtFechaInicio.getDate(); 2
        Date FechaFin = getFormulario().txtFechaFin.getDate(); 2
        Ntipoevento TipoEvento = (Ntipoevento) getFormulario().cmbTipoEvento.getObjetoSeleccionado(); 2
        Dlocal Local = (Dlocal) getFormulario().cmbLocal.getObjetoSeleccionado(); 2
        Nestadoevento estado = (Nestadoevento) getFormulario().cmbEstado.getObjetoSeleccionado(); 2
        evento = new Devento(); 2
        evento.setCodigo(Codigo); 2
        evento.setNombre(Nombre); 2
        evento.setSiglas(Siglas); 2
        evento.setFechainicio(FechaInicio); 2
        evento.setFechafin(FechaFin); 2
        evento.setIdlocal(Local); 2
        evento.setIdntipoevento(TipoEvento); 2
        evento.setIdnestadoevento(estado); 2
        evento.setDeventorecursoList(recursos); 2
        GestorNCEvento.insertarEvento(evento); 2
        GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia().getMensaje("SIGFE.guardado")); 2
        accion.ActualizarLista(); 2
        GestorMarcoTrabajo.getInstancia().mostrarFormularioExistente(GestorMarcoTrabajo.getInstancia().getPanelExistente(AccGestionarEvento.class.getName())); 2
    }
    else if (recursos.isEmpty()) { 3
        GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia().getMensaje("AccRegistrarEvento.mensajeRecursos")); 4
    }
    else if (existeEntidad) { 5
        GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia().getMensaje("AccGestionarEventos.registrarEntidad")); 6
    }
    else { 7
        GestorMensajes.MensajeInformacion(Cliente_I18n.getInstancia().getMensaje("SIGFE.vacios")); 8
    }
} 9

```

Figura 15. Código de la funcionalidad *Registrar Evento*.

Se realiza el grafo de flujo partiendo del código tomado.

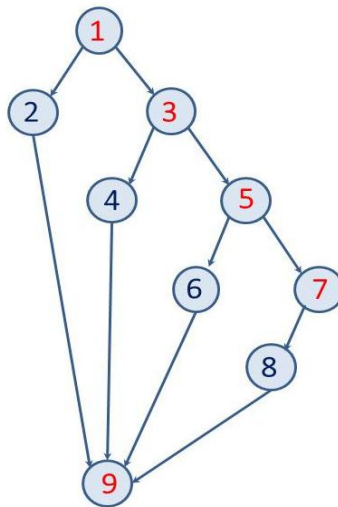


Figura 16. Grafo de flujo asociado al método *RegistrarEvento*.

Luego se calcula la **complejidad ciclomática**:

$$1. V(G) = 4 \quad 2. V(G) = A - N + 2 = 11 - 9 + 2 = 4 \quad 3. V(G) = P + 1 = 3 + 1 = 4$$

El valor de $V(G)$ da el número de **caminos linealmente independientes** de la estructura de control del programa, por lo que se definen los siguientes 4 caminos:

Camino 1: 1 - 2 - 9.

Camino 3: 1 - 3 - 5 - 6 - 9.

Camino 2: 1 - 3 - 4 - 9.

Camino 4: 1 - 3 - 5 - 7 - 8 - 9.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, realizando al menos un caso de prueba por cada camino. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que entran al procedimiento.
- **Resultados Esperados:** Se expone el resultado que se espera devuelva el procedimiento.

A continuación se muestra el diseño de uno de los casos de prueba realizado, para consultar los demás ver Anexo 9.

Caso de prueba para el camino básico 1:

Camino 1: 1 - 2 - 9.

Descripción:

- Los datos de entrada son correctos y cumplen con el formato indicado.

Condición de ejecución:

- El código será 002.
- El nombre será Feria Internacional de La Habana.
- Las siglas serán FIHAB.
- La Fecha de Inicio será el 1 de Mayo del 2013.
- La Fecha de Fin será el 21 de Mayo del 2013.
- El tipo de Evento será Feria Internacional.
- El local será Pabexpo.
- El estado será Abierto.
- La entidad organizadora será METALCUBA.
- Los recursos asignados serán TV Haier de 21 " y DVD Sony.

Entrada:

codigo = 002

nombre = Feria Internacional de La Habana

siglas = FIHAB

fechalnicio = Mayo 1, 2013

fechaFin = Mayo 21, 2013

tipoEvento = Feria Internacional

local = Pabexpo

estado = Abierto

entidadOrganizadora = METALCUBA

recursos = ([TV Haier 21", 5, 5, 1, 300.00, CUP][DVD Sony, 5, 5, 1, 10.00, CUC])

Resultados esperados:

- Se espera que se registre el evento satisfactoriamente y se le notifique al usuario mediante un mensaje de Aviso.

Luego de aplicar los distintos casos de pruebas derivados de la prueba de caja blanca “Camino Básico”, se pudo comprobar que el flujo de trabajo de la función analizada está correcto pues según las condiciones de entrada especificadas, se obtienen los resultados esperados.

Entre las técnicas de Caja Negra se empleó “Partición de Equivalencia”, técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba (Pressman, 2010). A continuación se muestra el caso de prueba correspondiente al escenario Registrar Solicitud (Ver Tabla 11).

Tabla 11. Caso de prueba: Escenario Registrar Solicitud.

Escenario	Descripción	ID de Solicitud	Áreas	Recursos	Respuesta del sistema	Flujo central
EC 1.1 Registrar Solicitud de Recursos correctamente	Al insertar los datos requeridos se registra la solicitud de recursos correctamente	V 1	V Pabellón Central	V Televisor LCD	El sistema permite registrar una solicitud de recursos .	1- Selecciona la Opción " Registrar" en la interfaz Gestionar Solicitud de Recursos. 2- Valida los datos introducidos según el formato definido y los campos obligatorios 3- Registra la solicitud en la Base de Datos y muestra un mensaje de confirmación que la solicitud de recursos se a registrado satisfactoriamente.
EC 1.2: Registrar Solicitud de Recursos: Campos Incorrectos.	Cuando se introduce algún campo incorrecto el sistema debe de lanzar un mensaje indicando el error.	I Cod13#	V Pabellón Central	V Televisor LCD	El sistema lanza una alerta de aviso indicando que el campo ID de Solicitud no está en el formato correcto.	
EC 1.3: Registrar Solicitud de Recursos: Campos Incompletos.	Cuando se insertan los datos y algún campo queda incompleto el sistema debe de lanzar un mensaje indicando el error.	I (Vacío)	NA	NA	El sistema lanza una alerta indicando que el campo ID de solicitud es obligatorio.	
		NA	I (Vacío)	NA	El sistema lanza una alerta indicando que es obligatorio contar con al menos un área en el listado de área.	
		NA	NA	I (Vacío)	El sistema lanza una alerta indicando que es obligatorio contar con al menos un recurso en el listado de recursos.	

Tabla 12. Descripción de las variables.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1.	ID de Solicitud	Campo texto	No	Solo admite números.
2.	Areas	Tabla	No	Solo admite Areas
3.	Recursos	Tabla	No	Solo admite Recursos.

Durante la ejecución del caso de prueba correspondiente a este escenario se obtuvieron 5 no conformidades, las cuales fueron resueltas en una segunda iteración.

Resultados de las pruebas realizadas

Luego de aplicados los métodos de prueba a los casos de uso críticos, es válido señalar que los resultados obtenidos han sido satisfactorios. Cada una de las no

conformidades detectadas fue debidamente atendida logrando un correcto comportamiento de las funcionalidades ante diferentes situaciones (entradas válidas y no válidas).

A continuación se presenta el número de no conformidades detectadas tras cada iteración de pruebas realizadas:

- ✓ En la primera iteración se detectaron 20 No Conformidades (NC), de las cuales 14 fueron por error de aplicación y 6 en la documentación.
- ✓ En la segunda iteración se detectaron 6 NC, entre ellas 1 por error de aplicación y 5 en la documentación.
- ✓ En la tercera iteración se corrigieron las no conformidades detectadas, para un 100% de satisfacción.

Todas las pruebas realizadas permiten afirmar que los requisitos funcionales han sido correctamente implementados.

Para la evaluación de las características de calidad funcionalidad y usabilidad son utilizados los procedimientos de evaluación que se describen a continuación.

Para medir la característica de calidad *funcionalidad*, se decide seguir el procedimiento descrito por (Estrada Peña, 2012) en “Procedimiento para evaluar la característica de funcionalidad de componentes de software”. En el mismo se establecen los siguientes pasos lógicos para lograr la medición:

1- Proceso de medición: Permite a los evaluadores determinar los pesos de cada una de las sub-características y los datos de las demás variables implicadas en la solución.

1.1- 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 19 del Anexo 7)

1.2- Recopilación de datos: Se colocan los datos de las variables implicadas en las fórmulas de medición de las métricas. (Ver Tabla 20 del Anexo 7)

2- Ejecución de la medición: Se realizan dos actividades fundamentales, una es el cálculo de las métricas y la otra es dar la evaluación correspondiente según los resultados obtenidos.

2.1- Calcular Métricas: Se calculan las métricas basándose en las fórmulas y los datos introducidos. (Ver Tabla 21 del Anexo 7)

2.2- Evaluación por cada métrica: Una vez que el evaluador posea los resultados de las métricas, pasará a evaluarlas cualitativamente mediante la tabla de escala. (Ver Tabla 22 del Anexo 7). Los resultados se ubican en la tabla Medición de la calidad funcional para establecer la evaluación alcanzada en cada métrica. (Ver Tabla 23 del Anexo 7)

3- Conclusión de la evaluación: Se obtiene la evaluación exacta y total de la funcionalidad, tomándose en consideración los resultados de cada una de las sub-características.

3.1- Evaluación por sub-característica: El evaluador dará una medición cuantitativa y cualitativa por sub-característica, calculando el por ciento de las métricas y evaluándolo de bien, regular o mal según la tabla de escala. Dicho cálculo del por ciento se realiza mediante la fórmula y las variables que se muestran en la Tabla 24 del Anexo 7.

3.2- Evaluación de la funcionalidad: Se muestra la evaluación final de la funcionalidad, calculando un promedio entre todas las sub-características.

Aplicación del procedimiento:

1- Proceso de medición:

Nivel de relevancia: El nivel de relevancia, en el caso del sistema a evaluar está establecido por la importancia de cada subcaracterística en el funcionamiento del sistema.

✓ Idoneidad: Alta

✓ Interoperabilidad: No Aplica

✓ Precisión: Media

✓ Seguridad: Alta

2- Ejecución de la medición:

✓ Idoneidad

a) Adecuacion funcional

$X = 1 - 10/55$

$X = 1 - 0.18$

$X = 0.82$

Nivel: Alto

Se evaluaron 55 funcionalidades, de las cuales 10 presentaron problemas.

Nota: Como se emplea una sola métrica, esta es la evaluación final de la sub-característica.

✓ Precisión

a) Precisión

$$X = 1 - (A/T) \quad X = 1 - (0/3) \quad X = 1 \quad \text{Nivel: Alto}$$

En un tiempo de operación de 3 segundos el sistema realizó con la precisión requerida las operaciones relacionadas con la asignación de recursos.

✓ Seguridad

a) Capacidad de revisión de cuentas de usuario

$$X = A/B \quad X = 10/10 \quad X = 1 \quad \text{Nivel: Alto}$$

Se realizaron 10 accesos al sistema, los cuales fueron registrados como trazas, en el historial de acceso que se almacena en la base de datos.

b) Capacidad de control de acceso

$$X = A/B \quad X = 4/4 \quad X = 1 \quad \text{Nivel: Alto}$$

Se realizaron 4 intentos con el objetivo de acceder sin autorización al sistema:

- 1- Cualquier usuario y contraseña
- 2- Usuario correcto y contraseña incorrecta
- 3- Usuario incorrecto y contraseña correcta.
- 4- Ambos campos vacíos.

Todos los intentos fallaron sin brindar información sensible.

3- Conclusión de la evaluación:

Evaluación de la sub-característica *Idoneidad*:

$$P_{Sub} = (0.82) * 100 \quad P_{Sub} = 82 \%$$

Evaluación de la sub-característica *Precisión*:

$$P_{Sub} = (1) * 100 \quad P_{Sub} = 100 \%$$

Evaluación de la sub-característica *Seguridad*:

$$P_{Sub} = (1 + 1)/2 * 100 \quad P_{Sub} = 100 \%$$

Evaluación de la característica de calidad *Funcionalidad*:

$$P_{Fun} = (82 + 100 + 100) / 3 \quad P_{Fun} = (282) / 3 \quad P_{Fun} = 94 \%$$

Analizando los resultados obtenidos por cada sub-característica evaluada, se llega a la conclusión que los módulos Convocatoria y Contratación están funcionando al 94 %, obteniendo un alto nivel de funcionalidad. En la Figura 17 se muestra el resultado general del procedimiento.

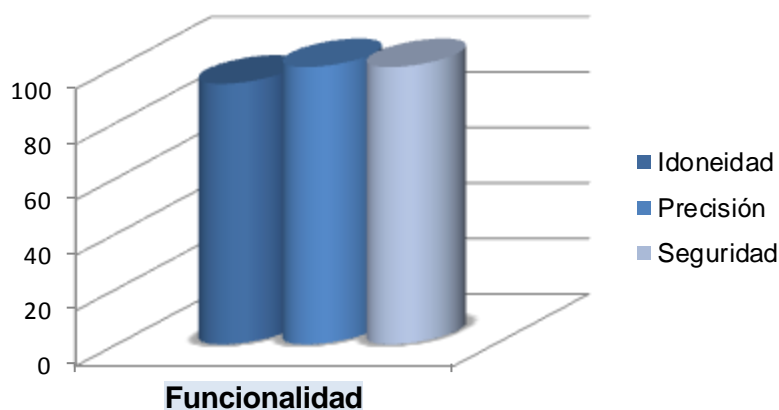


Figura 17. Funcionalidad de los módulos Convocatoria y Contratación.

Para medir la característica de calidad *Usabilidad*, se decide seguir el procedimiento descrito por (Pérez Dima, 2011) en “Evaluación de la Usabilidad de productos software”. En el mismo se establecen las siguientes actividades:

1- Definir las sub-características a evaluar.

2- Seleccionar a partir de las normas utilizadas, las estrategias a utilizar para el procedimiento.

3- Selección de usuarios para la aplicación de las métricas.

4- Aplicar métricas: Para la ejecución de esta actividad es necesario registrar los resultados obtenidos en cada métrica por cada usuario (Ver Tabla 25 del Anexo 8) y en correspondencia con el valor establecido en la Tabla Escala (Ver Tabla 26 del Anexo 8), determinar el nivel de cada sub-característica por usuarios.

5- Ejecutar Evaluación: Para ello el evaluador registra los datos y obtiene el nivel general de cada sub-característica (Ver Tabla 27 del Anexo 8), así como el grado de usabilidad existente en el software (Ver Tabla 28 del Anexo 8).

Aplicación del procedimiento

1- Las sub-características a evaluar son:

- | | |
|--------------------|----------------|
| ✓ Comprensibilidad | ✓ Operabilidad |
| ✓ Cognoscibilidad | ✓ Atracción |

2- Como estrategia para realizar el procedimiento de evaluación, se utilizarán métricas definidas en la Norma ISO/IEC 9126 y en el Procedimiento de Evaluación de la Usabilidad de Productos de Software de (Pérez Dima, 2011). (Ver Anexo 2).

3- Para la aplicación del procedimiento se seleccionaron tres usuarios, los cuales aplicaron las métricas y teniendo en cuenta el valor establecido en la tabla escala (Ver Tabla 26) evaluaron cada sub-característica.

4- Aplicación de las métricas:

Tabla 13. Evaluación de sub-características por usuarios.

Usuario #1: Leydis Rojas Elissaet.						
Sub-características	Métricas	Medición (Fórmula)	Interpretación del Valor Obtenido	Nivel Requerido	Nivel Requerido	Nivel Requerido
Comprensibilidad	Comprensibilidad de función	$X = A/B$ $X = 10/10$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Cognoscibilidad	Efectividad de la documentación y/o el sistema de ayuda.	$X = A/B$ $X = 1/1$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Operabilidad	Personalización.	$X = A/B$ $X = 2/2$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Atracción	Interacción por atracción.	$X = A/B$ $X = 7/10$ $X = 0.7$	$0 \leq X \leq 1$ $0 \leq 0.7 \leq 1$	Alto	Alto	0.7
Usuario #2: Robin Sencial Terrero						
Sub-características	Métricas	Medición (Fórmula)	Interpretación del Valor Obtenido	Nivel Requerido	Nivel Requerido	Nivel Requerido
Comprensibilidad	Comprensibilidad de función	$X = A/B$ $X = 11/11$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Cognoscibilidad	Efectividad de la documentación y/o el sistema de ayuda.	$X = A/B$ $X = 2/2$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1

Operabilidad	Personalización.	$X = A/B$ $X = 1/1$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Atracción	Interacción por atracción.	$X = A/B$ $X = 9/10$ $X = 0.9$	$0 \leq X \leq 1$ $0 \leq 0.9 \leq 1$	Alto	Alto	0.9
Usuario #3: Yoandris Velázquez Romero.						
Sub-características	Métricas	Medición (Fórmula)	Interpretación del Valor Obtenido	Nivel Requerido	Nivel Requerido	Nivel Requerido
Comprensibilidad	Comprensibilidad de función	$X = A/B$ $X = 8/8$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Cognoscibilidad	Efectividad de la documentación y/o el sistema de ayuda.	$X = A/B$ $X = 1/1$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Operabilidad	Personalización.	$X = A/B$ $X = 1/1$ $X = 1$	$0 \leq X \leq 1$ $0 \leq 1 \leq 1$	Alto	Alto	1
Atracción	Interacción por atracción.	$X = A/B$ $X = 8/10$ $X = 0.8$	$0 \leq X \leq 1$ $0 \leq 0.8 \leq 1$	Alto	Alto	0.8

5- Después de terminar la aplicación de las métricas se realiza el proceso de evaluación de cada subcaracterística.

Tabla 14. Evaluación final de cada sub-característica.

Sub-características	Suma de los resultados de cada usuario por métricas	Resultado	Nivel Obtenido
Comprensibilidad	$\sum_1^3 X = 1 + 1 + 1$	$3/3 = 1$	Alto
Cognoscibilidad	$\sum_1^3 X = 1 + 1 + 1$	$3/3 = 1$	Alto

Operabilidad	$\sum_1^3 X = 1 + 1 + 1$	$3/3 = 1$	Alto
Atracción	$\sum_1^3 X = 0.7 + 0.9 + 0.8$	$2.4/3 = 0.8$	Alto

$$V_{usab} = \frac{\sum \text{ResultG}}{\text{CantSub}} = \frac{1 + 1 + 1 + 0.8}{4} = \frac{3.8}{4} = 0.95 * 100 = 95\%$$

Analizando los resultados obtenidos por cada sub-característica evaluada, se llega a la conclusión que los módulos Convocatoria y Contratación presentan un elevado nivel de usabilidad (95%). En la Figura 18 se muestra el resultado general del procedimiento.

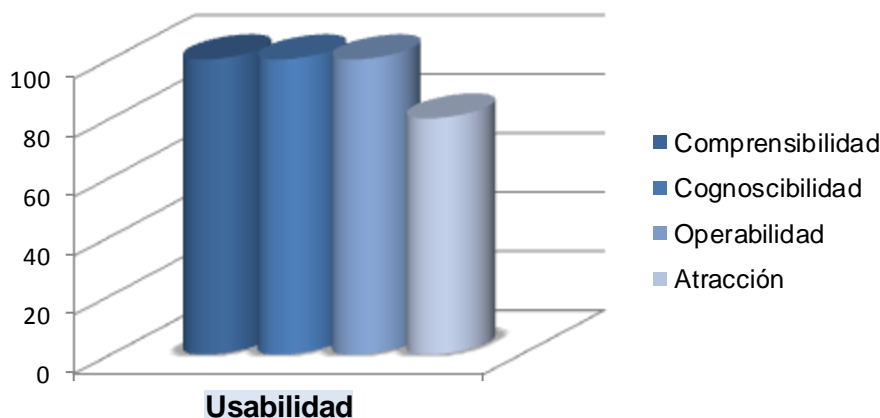


Figura 18. Usabilidad de los módulos Convocatoria y Contratación.

3.5 Conclusiones del capítulo

Para lograr una vista estática del sistema se realizaron los diagramas de componentes y despliegue, describiendo cómo los elementos del Modelo de Diseño se implementan en términos de componentes y cómo se organizan de acuerdo a los nodos físicos en el Modelo de Despliegue.

Con el objetivo de evaluar la solución desarrollada se aplicaron los métodos, niveles y tipos de prueba expuestos anteriormente, así como procedimientos de evaluación de funcionalidad y usabilidad, arrojando resultados satisfactorios que validan la calidad del producto construido.

Conclusiones

Con el propósito de darle cumplimiento al objetivo general y a la problemática planteada en el presente trabajo, se han llevado a cabo satisfactoriamente cada una de las tareas que fueron trazadas al comienzo del mismo.

- ✓ El estudio de los principales sistemas de gestión de ferias y exposiciones que existen en el ámbito nacional e internacional, centrándose en los procesos de convocatoria y contratación, demostró la necesidad de construir un software que realice estos procesos atendiendo a las exigencias actuales y que se adapte a las limitaciones tecnológicas de la Cámara de Comercio. A partir de un análisis exhaustivo se seleccionaron la metodología, tecnologías y herramientas que contribuyeron al desarrollo de una solución acorde a las necesidades del cliente.
- ✓ Como resultado del proceso de elicitación de requisitos se definió el ámbito del sistema y se identificaron las funcionalidades requeridas, que fueron agrupadas en casos de uso para su posterior implementación.
- ✓ Con el análisis y diseño de la solución se generaron los artefactos definidos por la metodología utilizada, que unidos al buen uso de los patrones de diseño, permitieron implementar los módulos Convocatoria y Contratación.
- ✓ Con el objetivo de evaluar la solución desarrollada se aplicaron métodos, niveles y tipos de prueba que certifican la calidad del producto construido. Además al aplicar los procedimientos de evaluación para obtener el nivel de funcionalidad y usabilidad en la propuesta de solución, se pudo concluir que la misma presenta altos niveles de las características de calidad evaluadas (94% de funcionalidad y 95% de usabilidad), demostrando así el cumplimiento del objetivo general de la investigación.

Recomendaciones

Los objetivos trazados en el presente trabajo de diploma fueron alcanzados satisfactoriamente, sin embargo se recomienda:

- ✓ 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.
- ✓ Continuar el estudio de los procesos convocatoria y contratación asociados a la gestión de ferias y exposiciones, con el propósito de detectar nuevas necesidades e incorporarle funcionalidades a la solución.
- ✓ Extender la búsqueda de nuevas tecnologías informáticas para perfeccionar los módulos en futuras versiones.

Bibliografía

Almora Gálvez, Yeleny. 2012. *Guía para Aseguramiento de la Funcionalidad en los Sistemas Informáticos de Gobierno Electrónico.* La Habana : s.n., 2012.

Álvarez Román, Santiago y Castañón Rodríguez, Álvaro. 2005. *Mono, el encuentro de los dos mundos.* Salamanca : s.n., 2005.

Arquitectura de Software- IEEE 1471-2000. IEEE, 1471-2000.

Bodoff, Stephanie. 2004. *The J2EE Tutorial.* Boston : Addison-Wesley, 2004. ISBN 0-321-24575-X.

Bradshaw Venzant, Maikel y Rodríguez Martini, Jorge Luis . 2010. *Análisis, diseño e implementación del Portal WAP del SIIPOL Móvil.* La Habana : s.n., 2010.

Bueno, Abel. 2012. Amiando (Aplicaciones Web). [En línea] 2012. [Citado el: 4 de noviembre de 2012.] <http://amiando.softonic.com/aplicaciones-web>.

Cámara de Comercio de la República de Cuba. 2009. *Portal Cámara de Comercio de la República de Cuba.* [En línea] 2009. [Citado el: 14 de octubre de 2012.] http://www.camaracuba.cu/index.php?option=com_content&view=article&id=44&Itemid=54.

Catalá Matienzo, Ivette. 2010. *Ingeniería de Requerimientos aplicados a la plataforma de Infodrez 2.0.* La Habana : s.n., 2010.

Claro Sánchez, Ana Margarita y Rodríguez Abreu, Ariel. 2011. *Módulo "Cuento" del producto Mis Mejores Cuentos.* La Habana : s.n., 2011.

Collins-Sussma, Ben, W. Fitzpatrick, Brian y Pilato, C Michael. 2004. *Control de Versiones con Subversion.* California : s.n., 2004. pág. 1.

de Sande Tundidor, Jose Angel . 2009. EclipseLink implementación de JPA | Tecnologías de la Sociedad de la Información. [En línea] 28 de mayo de 2009. [Citado el: 1 de febrero de 2013.] <http://tundidor.com/blog/?p=180>.

Doblemente S.L. 2007. Software para organización de eventos y congresos · Doblemente · Soluciones de comercio electrónico. [En línea] 2007. [Citado el: 4 de noviembre de 2012.] <http://www.doblemente.com/es/productos/inscribete--inscripcion-a-congresos,c984f99cd3618b67.html>.

Eclipse Foundation. 2013. EclipseLink - Eclipsepedia. [En línea] 29 de enero de 2013. [Citado el: 1 de febrero de 2013.] <http://wiki.eclipse.org/EclipseLink>.

Estrada Peña, Yuneisis. 2012. *Procedimiento para evaluar la característica de funcionalidad de componentes de software.* La Habana : s.n., 2012.

Fadraga Artiles, Lissuan y Lizama Mué, Yadira. 2012. *Propuesta de una Arquitectura de referencia para el desarrollo de aplicaciones empresariales.* 2012.

Góngora Rodríguez, Asnier Enrique. 2011. *Catálogo automatizado de métricas de calidad para evaluar los productos en las pruebas.* La Habana : s.n., 2011.

Guadalinfo. 2011. Programación de Juegos para Móviles . [En línea] 2011. [Citado el: 10 de noviembre de 2012.] http://www.edukanda.es/mediatecaweb/data/zip/1305/page_59.htm.

- Gutiérrez, Michael. 2009.** Servidor de Aplicaciones Java EE en CentOS(Glassfish). [En línea] 20 de agosto de 2009. [Citado el: 1 de febrero de 2013.] <http://michael-gutierrez.blogspot.com/2009/08/servidor-de-aplicaciones-java-ee-en.html>.
- Haines, Steven. 2009.** Java Reference Guide | SwingX | InformIT. [En línea] 6 de noviembre de 2009. [Citado el: 1 de febrero de 2013.] <http://www.informit.com/guides/content.aspx?g=java&seqNum=528>.
- Hernández Orallo, Enrique. 2002.** *El Lenguaje Unificado de Modelado (UML)*. s.l. : Madrid, 2002.
- Hernández, Jimmy . 2009.** PATRONES DE DISEÑO. [En línea] mayo de 2009. [Citado el: 22 de enero de 2013.] <http://patronesdediseno.blogspot.com/2009>.
- IEEE Standards Board. 1994.** IEEE Standard Glossary of Software Engineering Terminology. Nueva York : IEEE Computer Society Press, 1994. 978-0-7381-0391-4.
- Ishikawa, Sara, Silverstein, Murray y Alexander, Christopher. 1977.** *A Pattern Language: Towns, Buildings, Construction*. California : Oxford University Press, 1977. 9780195019193.
- J.D. Meier, Alex Home,David Hill. 2002.** *Application Architecture, Patterns And Practice*. Estados Unidos : s.n., 2002.
- Jacobson, Ivar, Booch, Grady y Rambaugh, James. 2000.** *El Proceso Unificado de Desarrollo de Sostware*. Madrid : Addison Wesley, 2000.
- Junta de Andalucía. 2012.** JPA | Marco de Desarrollo de la Junta de Andalucía. [En línea] 12 de noviembre de 2012. [Citado el: 1 de febrero de 2013.] <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/96>.
- . 2012. Referencia JPA | Marco de Desarrollo de la Junta de Andalucía. [En línea] 12 de noviembre de 2012. [Citado el: 1 de febrero de 2013.] <http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/176>.
- Ken, Arnold, Gosling, James y Holmes, David . 2001.** *El lenguaje de Programación Java™*. s.l. : Addison-Wesley, 2001. 9788478290451.
- Kruchten, P. 2000.** *The Rational Unified Process: An Introduction*. s.l. : Addison Wesley, 2000.
- Lara Bello, Rafael. 2010.** *Diseño e Implementación del Subsistema Cartas de Créditos Del Proyecto SAGEB*. Ciudad Habana : s.n., 2010.
- Larman, Craig. 1999.** *UML y Patrones*. Mexico : PRENTICE HALL, 1999. 970-17-0261-1.
- Manzanedo del Campo, Miguel Ángel , García Peñalvo, Francisco José y Cruzado Nuño, Ignacio . 1999.** *Guía de Iniciación al Lenguaje JAVA*. Zamora : s.n., 1999.
- Marca Huallpara, Hugo y Quisbert Limachi, Susana Nancy. 2010.** *TRABAJO DE INVESTIGACIÓN Y EXPOSICIÓN "Diagrama de Despliegue"*. La Habana : s.n., 2010.
- Martínez Prieto, Ana Belén . 2012.** Programar Sencillo: ¿Qué lenguaje de programación escoger? [En línea] 15 de enero de 2012. [Citado el: 24 de enero de 2013.] <http://programarsencillo.blogspot.com/2012/01/que-lenguaje-de-programacion-escoger.html>.
- Microsoft Corporation. 2009.** Microsoft Application Architecture Guide. Patterns & Practices. 2nd Edition. Washington, D.C. : Microsoft Press, 2009. 9780735627109.

- Microsoft TechNet. 2011.** Microsoft TechNet. [En línea] 2011. [Citado el: 23 de noviembre de 2012.] <http://technet.microsoft.com/es-ES/>.
- Netbeans.org. 2012.** New NetBeans IDE 7.2 Available - The Smarter and Faster Way to Code. [En línea] 24 de julio de 2012. [Citado el: 1 de febrero de 2013.] <http://netbeans.org/community/news/show/1562.html>.
- Ofi Eventos. 2010.** Ofi Eventos. [En línea] 2010. [Citado el: 4 de noviembre de 2012.] <http://ofi-eventos.programas-gratis.net/>.
- Oracle. 2010.** Guía del administrador de negocio de Sun Identity Manager 8.1. [En línea] 2010. [Citado el: 1 de febrero de 2013.] <http://docs.oracle.com/cd/E19957-01/821-0062/byaui/index.html>.
- Paredes Galarza, Luis Vicente y Vargas Jácome, Silvana Lizette. 2011.** *ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA INFORMÁTICO DE LABORATORIO CLÍNICO PARA EL CENTRO DE DIAGNÓSTICO CLÍNICO C.D.C.* Quito : s.n., 2011.
- Patel Sriganesh, Rima, Brose , Gerald y Silverman, Micah. 2006.** *Mastering Enterprise Java Beans 3.0.* Chichester : Wiley Publishing, 2006. págs. 3-9.
- Peco Martínez, Daniel. 2010.** PostGreSQL vs. MySQL. [En línea] febrero de 2010. [Citado el: 2013 de febrero de 1.] http://danielpecos.com/docs/mysql_postgres/index.html.
- Pérez Chirino, Yosbel. 2010.** *Herramienta para la gestión de torneos ajedrecísticos, versión 2.0.* La Habana : s.n., 2010.
- Pérez Dima, Yusmary. 2011.** *Evaluación de la Usabilidad de productos software.* La Habana : s.n., 2011.
- PerfectTablePlan. 2009.** PerfectTablePlan. *PerfectTablePlan: software de gestión de eventos.* [En línea] 2009. [Citado el: 4 de noviembre de 2012.] <http://www.programastop.com/perfecttableplan-software-de-gestion-de-eventos-bodas-banquetes-ceremonias/>.
- Périsé, Marcelo Claudio. 2001.** *Proyecto Informático. Una Metodología Simplificada.* Argentina : s.n., 2001. 987-43-2947-5.
- Pressman, Roger S. 2010.** *Software Engineering. A Practitiones's Approach. Seventh Edition.* Nueva York : McGraw-Hill, 2010. 978-0-07-337397-7.
- Pressman, Roger S. 2005.** *Ingeniería del Software: Un Enfoque Práctico. Sexta Edición.* 2005. pág. 900.
- Programación en catellano. 2011.** Programación en catellano- Introducción a JMS (Java Message Service). [En línea] 2011. [Citado el: 6 de 12 de 2012.] http://www.programacion.com/articulo/introduccion_a_jms_java_message_service_142.
- Ramírez, Iván. 2012.** Aplicación para desarrollo en Java. [En línea] 22 de octubre de 2012. [Citado el: 6 de diciembre de 2012.] <http://netbeans-ide.softonic.com/>.
- RegOnline. 2010.** Software de gestión. [En línea] 2010. [Citado el: 4 de noviembre de 2012.] http://www.regonline.com.es/__articles/products/software~de~gesti%C3%B3n~de~eventos.
- Rojas, Elisabeth. 2009.** GlassFish Enterprise Server v3. [En línea] 15 de diciembre de 2009. [Citado el: 1 de febrero de 2013.]

http://www.muycomputerpro.com/2009/12/15/actualidadnoticiasglassfish-enterprise-server-v3_we9erk2xxdawdxuz0-93wneniqrqncvixpjz7pg7axsebl2ncu2_2l5xbx_weyk/.

Rosales Velazquez, Norlen. 2009. *SOFTWARE EDUCATIVO PICO MESTRO*. La Habana : s.n., 2009.

Rubio, Daniel. 2010. Optimizing JPA Performance: An EclipseLink, Hibernate, and OpenJPA Comparison. [En línea] 20 de julio de 2010. [Citado el: 1 de febrero de 2013.] <http://java.dzone.com/articles/jpa-performance-optimization?page=0,0>.

Rumbaugh, James , Jacobson, Ivar y Booch, Grady. 2000. *El Lenguaje Unificado de Modelado. Manual de Referencia*. Massachusetts : Addison Wesley, 2000.

Sommerville, Ian. 2005. *Ingeniería Del Software*. s.l. : Sexta edición., 2005.

—. **2007.** *Ingeniería de Software*. 8va Edición. Boston : Addison-Wesley, 2007. 9780321313799.

Szyperski, C. 1998. *Component Software. Beyond Object-Oriented Programming*. 1998.

The PostgreSQL Global Development Group. 2011. *PostgreSQL 9.1.0 Documentation*. California : Regents of the University of California, 2011.

—. **2010.** *Tutorial de PostgreSQL*. California : Regents of the University of California, 2010.

Vega Miniet, Yanet. 2012. *Descripción de Procesos de Negocio*. La Habana : s.n., 2012.

—. **2012.** Proyecto Técnico. Proyecto de Gestión de Ferias y Eventos. 2012.

Velázquez Corona, Dianick . 2008. *Análisis y Diseño del Módulo Resultados de la Colección Multisaber*. La Habana : s.n., 2008.

Villarroel González, Luis Ramiro y Montalvo Yépez , César Alejandro. 2008. *Aplicación de la metodología MSF v4.0 a la definición e implementación de arquitecturas orientadas a objetos en visual studio .net 2005, caso práctico g5 sharing files*. Sangolqui : s.n., 2008.

Vindas, Rolando. 2008. Intergraphic Designs. Intergraphic Designs. [En línea] 29 de septiembre de 2008. [Citado el: 22 de enero de 2013.] <http://www.intergraphicdesigns.com/blog/2008/09/29/resumen-sobre-ventajas-de-utilizar-subversion/>.

Visual Paradigm. 2009. Visual Paradigm. [En línea] 21 de febrero de 2009. [Citado el: 23 de noviembre de 2012.] <http://www.visual-paradigm.com/>.

Yáñez Martínez , Susana . 2008. *BPMN 2.0. Ejemplo*. México : s.n., 2008.

Anexos

Anexo 1. Métricas para evaluar el diseño.

Tamaño Operacional de clase (TOC)

El tamaño general de una clase puede medirse determinando las siguientes medidas:

- El total de operaciones (tanto heredadas como privadas de la instancia), que se encapsulan dentro de la clase.
- El número de atributos (tanto heredados como privados de la instancia), encapsulados por la clase.

Estos dos valores son sumados de acuerdo a la clase que se analiza y el resultado es tomado como el valor de TC.

Valores grandes de Tamaño de Clase (TC) representan gran responsabilidad de la clase. Esto implica la reducción de la reutilización de la clase y complica la implementación y las pruebas. De forma general, operaciones y atributos deben ser ponderados al determinar el tamaño de la clase. Para valores pequeños de TC para una clase existe mayor posibilidad de que la clase pueda ser reutilizada.

Tabla 15. Interpretación de los resultados de la métrica TOC.

Parámetros de Calidad	Valores Grandes de TC
Reutilización	Reduce la reutilización de la clase
Implementación	Complica la implementación
Complejidad de las pruebas	Hace compleja las pruebas del sistema
Responsabilidad	La clase debe tener bastante responsabilidad

Tabla 16. Medidas o umbrales de la métrica TOC.

TOC	Umbral
Pequeño	$TC \leq 20$
Medio	$20 < TC \leq 30$
Grande	$TC > 30$

Relaciones entre Clases (RC)

Está dado por el número de relaciones de uso entre clases y evalúa los siguientes atributos de calidad:

- ✓ **Acoplamiento:** Un aumento del RC implica un aumento del Acoplamiento de la clase.
- ✓ **Complejidad de mantenimiento:** Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
- ✓ **Reutilización:** Un aumento del RC implica una disminución en el grado de reutilización de la clase.
- ✓ **Cantidad de pruebas:** Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 17. Criterios de evaluación de la métrica RC.

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	$RC = 0$
	Bajo	$RC = 1$
	Medio	$RC = 2$
	Alto	$RC > 2$
Complejidad de Mantenimiento	Baja	$RC \leq \text{Promedio}$
	Media	$\text{Promedio} < RC \leq 2 * \text{Promedio}$
	Alta	$RC > 2 * \text{Promedio}$
Reutilización	Baja	$RC > 2 * \text{Promedio}$
	Media	$\text{Promedio} < RC \leq 2 * \text{Promedio}$
	Alta	$RC \leq \text{Promedio}$
Cantidad de pruebas	Baja	$RC \leq \text{Promedio}$
	Media	$\text{Promedio} < RC \leq 2 * \text{Promedio}$
	Alta	$RC > 2 * \text{Promedio}$

Anexo 2. Métricas Externas de Usabilidad.

Métrica de Comprensibilidad: esta métrica le indica al evaluador cómo un nuevo usuario podría comprender si el software es idóneo para la aplicación a la cual lo destina – y cómo el software puede ser usado para una tarea en particular.

- *Nombre de la métrica:* Comprensibilidad de función.
- *La métrica propone medir:* ¿Qué proporción de funciones del producto, será el usuario capaz de entender correctamente?

- *Método de aplicación:* Cuente el número de funciones de la interfaz del usuario donde los propósitos son fácilmente comprendidos por el usuario y se comparan con el número de funciones disponibles para el usuario.

- *Medición:* $X = A/B$ donde:

A: Número de funciones de la interfaz gráfica de usuario cuyo propósito es correctamente descrito por el usuario.

B: Número de funciones disponibles de la interfaz gráfica de usuario.

- *Interpretación del valor obtenido:* $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

- *Escala:* Absoluta

Métrica de Cognoscibilidad: esta métrica le mostrará al evaluador cómo el sistema puede contribuir al usuario a aprender sobre su aplicación.

a) *Nombre de la métrica:* Efectividad de la documentación y/o el sistema de ayuda.

b) *La métrica propone medir:* ¿Qué proporción de tareas pueden ser completadas correctamente después de usar la documentación del usuario y/o la ayuda del sistema?

c) *Método de aplicación:* Cuente el número de tareas exitosamente completadas después de acceder a la ayuda y/o a la documentación del sistema y se compara con el número total de tareas probadas.

d) *Medición:* $X = A/B$ donde:

A: Número de tareas exitosamente completadas después de acceder a la ayuda y/o a la documentación del sistema.

B: Número total de tareas probadas.

e) *Interpretación del valor obtenido:* $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

f) *Escala:* Absoluta

Métrica de Operabilidad: esta métrica le mostrará al evaluador la posibilidad de operar y controlar el sistema por el usuario.

a) *Nombre de la métrica:* Personalización.

La métrica propone medir: ¿El usuario puede personalizar fácilmente los procedimientos de operación para su conveniencia? ¿Qué proporción de las funciones se pueden personalizar?

b) *Método de aplicación:* Realizar la comprobación de usuario y observar el comportamiento del usuario.

c) *Medición:* $X = A/B$ donde:

A: Número de funciones personalizadas con éxito.

B: Número de intentos de personalizar.

d) Interpretación del valor obtenido: $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

e) Escala: Absoluta

Métrica de Atracción: esta métrica le permitirá al evaluador valorar la apariencia del software y va a estar influenciada por factores tales como el color en la pantalla y su diseño.

a) Nombre de la métrica: Interacción por atracción

b) La métrica propone medir: ¿Cuán atractiva es el sistema para el usuario?

c) Método de aplicación: Se aplicará el cuestionario de Interacción por satisfacción. Ver Anexo 4.

d) Medición: Resultados del cuestionario con posterior uso de la interfaz gráfica de usuario.

$X = A/B$ donde:

A: Número de preguntas respondidas satisfactoriamente.

B: Número total de preguntas.

e) Interpretación del valor obtenido: $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

f) Escala: Valorativa

Anexo 3. Métricas Externas de Funcionalidad

Métrica de Idoneidad: esta métrica le indica al evaluador cuán adecuada, completa, correcta ha sido la implementación de las funcionalidades, permitiéndole establecer un criterio cuantitativo y cualitativo sobre la idoneidad del sistema a evaluar

a) Nombre de la métrica: Adecuación Funcional

b) La métrica propone medir: ¿Cuán adecuada es la función evaluada?

c) Método de aplicación: Cuente el número de funciones que son adecuadas para realizar las tareas específicas y compararlas con el número de funciones.

d) Medición: $X = 1 - A/B$ donde:

A: Número de funciones en las cuales se detectaron problemas en la evaluación.

B: Número de funciones evaluadas: constituyen la cantidad total de los exámenes que se le realizaron al sistema.

e) Interpretación del valor obtenido: $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

f) *Escala:* Absoluta

Métrica de Precisión: esta métrica le indica al evaluador con qué frecuencias se encuentran cálculos y resultados inexactos en el sistema evaluado, permitiéndole establecer un criterio cuantitativo y cualitativo sobre la exactitud del mismo.

a) *Nombre de la métrica:* Precisión

b) *La métrica propone medir:* ¿Con qué frecuencia los evaluadores encuentran resultados con la precisión inadecuada?

c) *Método de aplicación:* Registre el número de resultados con una precisión inadecuada.

d) *Medición:* $X = 1 - A/T$ donde:

A: Número de resultados incorrectos encontrados por los evaluadores con un nivel de precisión diferente del requerido.

T: Tiempo de operación: es el tiempo que se demora el sistema en mostrar los resultados.

e) *Interpretación del valor obtenido:* $X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

f) *Escala:* Valorativa

Métrica de Interoperabilidad: esta métrica permite evaluar si existe un correcto intercambio de datos entre los sistemas y si son correctas las funciones de interfaces para una transferencia de datos específica, facilitándole al evaluador establecer un criterio cuantitativo y cualitativo sobre la interoperabilidad del sistema. En el caso de la solución a evaluar esta subcaracterística no aplica porque no existirá un intercambio de datos con otros sistemas.

Métrica de Seguridad: con la aplicación de estas métricas se conocerá si el sistema posee la capacidad de control de acceso a los datos y de los usuarios al sistema, así como un el nivel de prevención de la corrupción de los datos que presenta, permitiéndole al evaluador establecer un criterio cualitativo y cuantitativo sobre la seguridad del sistema.

a) *Nombre de la métrica:* Capacidad de revisión de cuentas de acceso.

b) *La métrica propone medir:* ¿Cuán completo es el rastro de auditoría del acceso del usuario al sistema y a los datos?

c) *Método de aplicación:* Evaluar la cantidad de accesos de los usuarios que el sistema guardó en la Base de Datos durante la evaluación.

d) *Medición:* $X = A/B$ donde:

A: Número de accesos del usuario al sistema y a los datos que el sistema registró en la Base de Datos, durante la evaluación.

B: Número de accesos del usuario al sistema y a los datos durante la evaluación.

e) Interpretación del valor obtenido: $0 \leq X \leq 1$

NOTA: A mayor cercanía al 1 resultará mejor.

f) Escala: Absoluta

a) Nombre de la métrica: Capacidad de control de acceso

b) La métrica propone medir: ¿Cuán controlable es el acceso al sistema?

c) Método de aplicación: Contar el número de operaciones ilegales detectadas y compararlas con el número de operaciones ilegales, como en las especificaciones.

d) Medición:

$$X = A/B \text{ donde:}$$

A: Número de operaciones ilegales que el sistema rechaza.

B: Número total de operaciones ilegales intentadas para acceder al sistema.

g) Interpretación del valor obtenido: $0 < X \leq 1$

NOTA: A mayor cercanía al 1 resulta mejor.

e) Escala: Absoluta

Anexo 4. Cuestionario

Cuestionario de Interacción por satisfacción

Participante: _____

Sexo: Femenino: _____ Masculino: _____

Edad: _____

Fecha: ____/____/____

Preguntas	SI	NO
1) ¿El sistema le permite realizar las tareas solicitadas?	<input type="radio"/>	<input type="radio"/>

2) ¿La navegación a través del sistema resulta sencilla?	<input type="radio"/>	<input type="radio"/>

3) ¿La apariencia general del sistema resulta sencilla?

4) ¿La representatividad de los iconos del sistema respecto a la función de los mismos es la adecuada?

5) ¿La estructura y organización del sistema es la adecuada?

6) ¿Le han parecido claros y representativos los nombres y descripciones que aparecen en el sistema?

7) ¿La combinación de colores/fondos son visualmente agradables?

8) ¿El sistema facilita la identificación de elementos en lugar de tener que recordarlos?

9) En general. ¿Le fue fácil realizar las tareas solicitadas?

10) ¿Piensas que necesitarás la ayuda de alguien para manejar el sistema?

Opcional:

11) Si usted desearía dar otra opinión sobre el sistema en general puede hacerlo en la siguiente sección:

Anexo 5. Requerimientos de los Módulos

Tabla 18. Requisitos de los Módulos Convocatoria y Contratación.

Requisitos Funcionales de los Módulos:		
RF_1 Registrar evento	RF_20 Modificar recurso ferial a la solicitud de recursos	RF_39 Modificar recurso ferial a la asignación de recursos
RF_2 Modificar evento	RF_21 Eliminar área a la solicitud de recursos	RF_40 Eliminar área a la asignación de recursos
RF_3 Eliminar evento	RF_22 Eliminar recurso ferial de la solicitud de recursos	RF_41 Eliminar recurso ferial de la asignación de recursos
RF_4 Consultar evento	RF_23 Eliminar solicitud de recursos	RF_42 Eliminar asignación de recursos
RF_5 Buscar evento	RF_24 Consultar solicitud de recursos	RF_43 Consultar asignación de recursos
RF_6 Registrar entidad	RF_25 Buscar solicitud de recursos	RF_44 Buscar asignación de recursos
RF_7 Modificar entidad	RF_26 Adicionar cuenta bancaria de una entidad	RF_45 Acreditar personalidad
RF_8 Eliminar entidad	RF_27 Modificar cuenta bancaria de una entidad	RF_46 Modificar personalidad
RF_9 Consultar entidad	RF_28 Eliminar cuenta bancaria de una entidad	RF_47 Eliminar personalidad
RF_10 Buscar entidad	RF_29 Acreditar participante	RF_48 Buscar personalidad
RF_11 Búsqueda avanzada de entidad	RF_30 Modificar participante	RF_49 Búsqueda avanzada de personalidad
RF_12 Registrar datos de contactos	RF_31 Buscar participante	RF_50 Buscar Solicitud Asignación
RF_13 Modificar datos de contacto	RF_32 Búsqueda avanzada de participante	RF_51 Consultar Solicitud Asignación
RF_14 Eliminar datos de contacto	RF_33 Consultar participante	RF_52 Asignar Recursos a un evento.
RF_15 Registrar solicitud de recursos	RF_34 Registrar asignación de recursos	RF_53 Modificar Asignación de Recursos de un evento.
RF_16 Modificar solicitud de recursos	RF_35 Modificar asignación de recursos	RF_54 Eliminar Asignación de Recursos de un evento.
RF_17 Adicionar área a la solicitud de recursos	RF_36 Adicionar área a la asignación de recursos	RF_55 Buscar recursos asignados.
RF_18 Adicionar recurso ferial a la solicitud de recursos	RF_37 Adicionar recurso ferial a la asignación de recursos	
RF_19 Modificar área a la	RF_38 Modificar área a la	

solicitud de recursos	asignación de recursos	
-----------------------	------------------------	--

Anexo 6. Diagramas de Secuencia: CU *Gestionar Solicitud de recursos.*

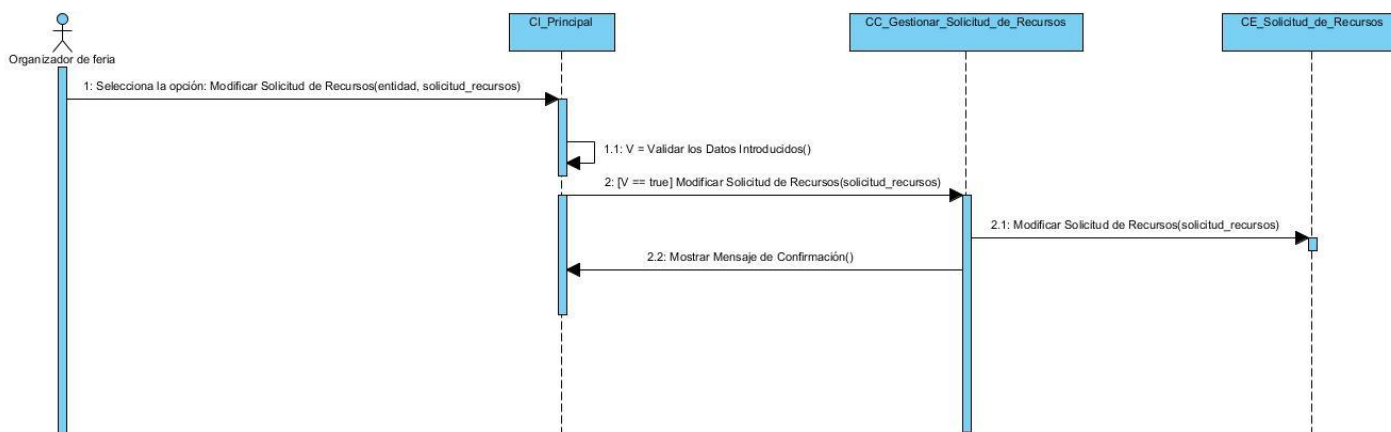


Figura 19. Diagrama de Secuencia *Modificar Solicitud de recursos.*

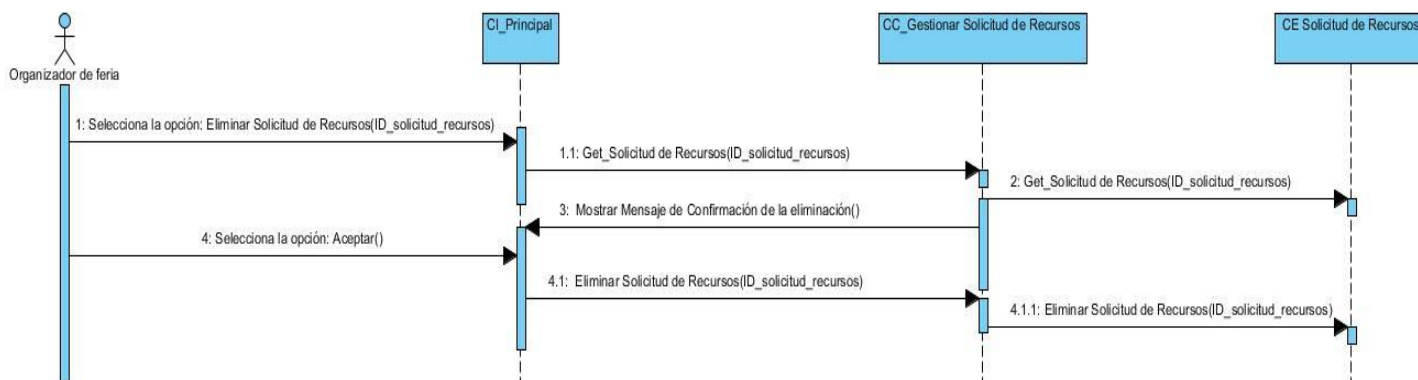


Figura 20. Diagrama de Secuencia *Eliminar Solicitud de recursos.*

Anexo 7. Procedimiento para medir la característica de calidad *funcionalidad.*

Tabla 19. Medición de la calidad funcional para establecer nivel de relevancia.

Sub-características	Nivel de relevancia	Nombre de la métrica
Idoneidad	Alta	Adecuación funcional

Tabla 20. Medición de la calidad funcional para recopilar datos.

Nombre de la	Métrica	Variables	Datos
--------------	---------	-----------	-------

métrica			
Adecuación funcional	$X = 1 - A/B$	A: Número de funciones en las cuales se detectaron problemas.	
		B: Número de funciones evaluadas.	
Compleitud de la implementación funcional	$X = 1 - A/B$	A: Número de funciones perdidas detectadas.	
		B: Número de funciones descritas en la especificación de requisitos.	

Tabla 21. Medición de la calidad funcional para cálculo de métricas.

Nombre de la métrica	Métrica	Variables	Datos	Resultados
Adecuación funcional	$X = 1 - A/B$	A: Número de funciones en las cuales se detectaron problemas.	1	0.9
		B: Número de funciones evaluadas.	10	
Compleitud de la implementación funcional	$X = 1 - A/B$	A: Número de funciones perdidas detectadas.	0	1
		B: Número de funciones descritas en la especificación de requisitos.	3	

Tabla 22. Escala para la evaluación.

Rango	Porciento	Nivel
$0.8 \leq X \leq 1$	80% - 100%	Alto
$0.5 \leq X \leq 0.8$	50% - 80%	Medio
$X < 0.5$	Menor o igual que 49%	Bajo

Tabla 23. Medición de la calidad funcional para establecer la evaluación en cada métrica.

Nombre de la métrica	Métrica	Variables	Datos	Resultados	Evaluación
Adecuación funcional	$X = 1 - A/B$	A: Número de funciones en las cuales se detectaron problemas.			
		B: Número de funciones evaluadas.			

Completitud de la implementación funcional	$X = 1 - A/B$	A: Número de funciones perdidas detectadas.			
		B: Número de funciones descritas en la especificación de requisitos.			

Tabla 24. Cálculo del porcentaje por sub-característica.

Fórmula para el cálculo del porcentaje	Variables
$P_{sub} = \frac{\sum ResultM}{CantM} * 100$	<p>Psub: Porcentaje alcanzado por sub-característica.</p> <p>ResultM: Resultados de cálculos de las métricas.</p> <p>CantM: Cantidad de métricas calculadas.</p>

Anexo 8. Procedimiento para medir la característica de calidad *usabilidad*.

Tabla 25. Aplicación de las métricas por usuarios.

Usuario:						
Sub-características	Métricas	Medición (Fórmula)	Interpretación del Valor Obtenido	Nivel Requerido	Nivel Obtenido	Resultado Real

Tabla 26. Escala para la Evaluación.

Rango	Nivel
$0.6 < X \leq 1$	Alto
$0.3 \leq X \leq 0.6$	Medio
$0 \leq X < 0.3$	Bajo

Tabla 27. Cálculo de los resultados generales.

Sub-características	Suma de los resultados de cada usuario por métricas	Resultado	Nivel Obtenido

Tabla 28. Obtención del Nivel de *Usabilidad* existente en el software.

Fórmula del cálculo del nivel de Usabilidad	Variables
$\text{Vusab} = \frac{\sum \text{ResultG}}{\text{CantSub}}$	<p>Vusab: Valor de usabilidad alcanzado</p> <p>ResultG: Resultados generales de cada una sub-característica.</p> <p>CantM: Cantidad de sub-características evaluadas.</p>

Anexo 9. Casos de Prueba: Técnica Camino Básico

Caso de prueba para el camino básico 2:

Camino 2: 1 - 3 - 4 - 9.

Descripción:

- Los datos de entrada cumplirán con el siguiente requisito: los recursos asignados estarán vacíos.

Condición de ejecución:

- El código será 002.
- El nombre será Feria Internacional de La Habana.
- Las siglas serán FIHAB.
- La Fecha de Inicio será el 1 de Mayo del 2013.
- La Fecha de Fin será el 21 de Mayo del 2013.
- El tipo de Evento será Feria Internacional.
- El local será Pabexpo.
- El estado será Abierto.
- La entidad organizadora será METALCUBA.
- Los recursos asignados estarán vacíos.

Entrada:

codigo = 002

nombre = Feria Internacional de La Habana

siglas = FIHAB

fechaInicio = Mayo 1, 2013

fechaFin = Mayo 21, 2013

tipoEvento = Feria Internacional

local = Pabexpo

estado = Abierto

entidadOrganizadora = METALCUBA

recursos = ()

Resultados esperados:

- Se espera que se le notifique al usuario mediante un mensaje de Aviso que debe asignarle recursos al evento para que pueda ser registrado.

Caso de prueba para el camino básico 3:

Camino 3: 1 - 3 - 5 - 6 - 9.

Descripción:

- Los datos de entrada cumplirán con el siguiente requisito: la entidad organizadora será nula.

Condición de ejecución:

- El código será 002.
- El nombre será Feria Internacional de La Habana.
- Las siglas serán FIHAB.
- La Fecha de Inicio será el 1 de Mayo del 2013.
- La Fecha de Fin será el 21 de Mayo del 2013.
- El tipo de Evento será Feria Internacional.
- El local será Pabexpo.
- El estado será Abierto.
- La entidad organizadora será nula.
- Los recursos asignados serán TV Haier de 21 " y DVD Sony.

Entrada:

codigo = 002

nombre = Feria Internacional de La Habana

siglas = FIHAB

fechaInicio = Mayo 1, 2013

fechaFin = Mayo 21, 2013

tipoEvento = Feria Internacional

local = Pabexpo

estado = Abierto

entidadOrganizadora = nulo

recursos = ([TV Haier 21", 5, 5, 1, 300.00, CUP][DVD Sony, 5, 5, 1, 10.00, CUC])

Resultados esperados:

- Se espera que se le notifique al usuario mediante un mensaje de Aviso que debe gestionar la entidad organizadora del evento para que pueda ser registrado.

Caso de prueba para el camino básico 4:

Camino 4: 1 - 3 - 5 - 7 - 8 - 9.

Descripción:

- Los datos de entrada cumplirán con el siguiente requisito: el campo código será nulo.

Condición de ejecución:

- El código será nulo.
- El nombre será Feria Internacional de La Habana.
- Las siglas serán FIHAB.
- La Fecha de Inicio será el 1 de Mayo del 2013.
- La Fecha de Fin será el 21 de Mayo del 2013.
- El tipo de Evento será Feria Internacional.
- El local será Pabexpo.
- El estado será Abierto.
- La entidad organizadora será METALCUBA.
- Los recursos asignados serán TV Haier de 21 " y DVD Sony.

Entrada:

codigo = nulo

nombre = Feria Internacional de La Habana

siglas = FIHAB

fechaInicio = Mayo 1, 2013

fechaFin = Mayo 21, 2013

tipoEvento = Feria Internacional

local = Pabexpo

estado = Abierto

entidadOrganizadora = METALCUBA

recursos = ([TV Haier 21", 5, 5, 1, 300.00, CUP][DVD Sony, 5, 5, 1, 10.00, CUC])

Resultados esperados:

- Se espera que se le notifique al usuario mediante un mensaje de Aviso que existen campos que no cumplen con el formato indicado.

Anexo 10. Casos de uso de los Módulos Convocatoria y Contratación.

Tabla 29. Listado de Casos de Uso

CU_2	Gestionar evento
Actor	Organizador de feria
Descripción	Permite realizar una gestión de los eventos para dar comienzo al proceso de convocatoria. Dicha gestión está enmarcada en registrar un evento; modificarlo en caso de que se desee actualizar algún dato; eliminarlo y consultarlo.
Referencia	RF_1, RF_2, RF_3, RF_4
CU_3	Buscar evento
Actor	Caso de uso Extendido
Descripción	Permite buscar un evento en el sistema. Muestra los datos del evento o los eventos que coinciden con los parámetros de búsqueda establecidos.
Referencia	RF_5, CU_2
CU_4	Gestionar Entidad
Actor	Caso de uso Extendido
Descripción	Permite adicionar, modificar, eliminar y consultar una entidad.
Referencia	RF_6, RF_7, RF_8, RF_9, RF_11, CU_2
CU_5	Buscar Entidad
Actor	Caso de uso Incluido
Descripción	Permite buscar una entidad inscrita en el sistema. Muestra los datos de la(s) entidad(es) que coinciden con los parámetros de búsqueda especificados.

Referencia	RF_10, CU_4
CU_6	Gestionar Cuenta Bancaria
Actor	Caso de uso Incluido
Descripción	Permite adicionar, modificar y eliminar cuentas bancarias a una entidad.
Referencia	RF_26, RF_27, RF_28, CU_4
CU_7	Gestionar Participante
Actor	Caso de uso Extendido
Descripción	Permite acreditar (registrar), modificar, eliminar, visualizar y buscar un participante.
Referencia	RF_29, RF_30, RF_31, RF_32, RF_33, CU_4
CU_8	Gestionar Personalidad
Actor	Caso de uso Extendido
Descripción	Permite acreditar, modificar, eliminar, visualizar y buscar una personalidad en un evento.
Referencia	RF_45, RF_46, RF_47, RF_48, RF_49, CU_2
CU_9	Gestionar Asignación de Recursos
Actor	Diseñador de feria
Descripción	Permite registrar, modificar y eliminar una asignación de recursos.
Referencia	RF_34, RF_35, RF_36, RF_37, RF_38, RF_39, RF_40, RF_41, RF_42, RF_43, RF_44
CU_10	Buscar solicitud _ asignación de recursos
Actor	Caso de uso Incluido
Descripción	Permite buscar una solicitud o asignación de recursos en el sistema.
Referencia	RF_50, CU_9
CU_11	Consultar solicitud _ asignación de recursos
Actor	Caso de uso Extendido
Descripción	Permite mostrar un listado de las solicitudes y asignaciones registradas en el sistema
Referencia	RF_51, CU_10
CU_12	Gestionar Datos de Contacto
Actor	CU Incluido

Descripción	Permite registrar, modificar y eliminar datos de contacto de una entidad, sucursal o persona.
Referencia	RF_12, RF_13, RF_14, CU_4
CU_13	Gestionar Datos de Contacto
Actor	CU Extendido
Descripción	Permite registrar, modificar y eliminar datos de contacto de una entidad, sucursal o persona.
Referencia	RF_12, RF_13, RF_14, CU_7
CU_14	Gestionar Recursos Evento
Actor	CU Extendido
Descripción	Permite registrar, modificar y eliminar los recursos asignados a un evento.
Referencia	RF_52, RF_53, RF_54, RF_55, CU_2

Anexo 11. Diagrama de componentes divididos en capas.

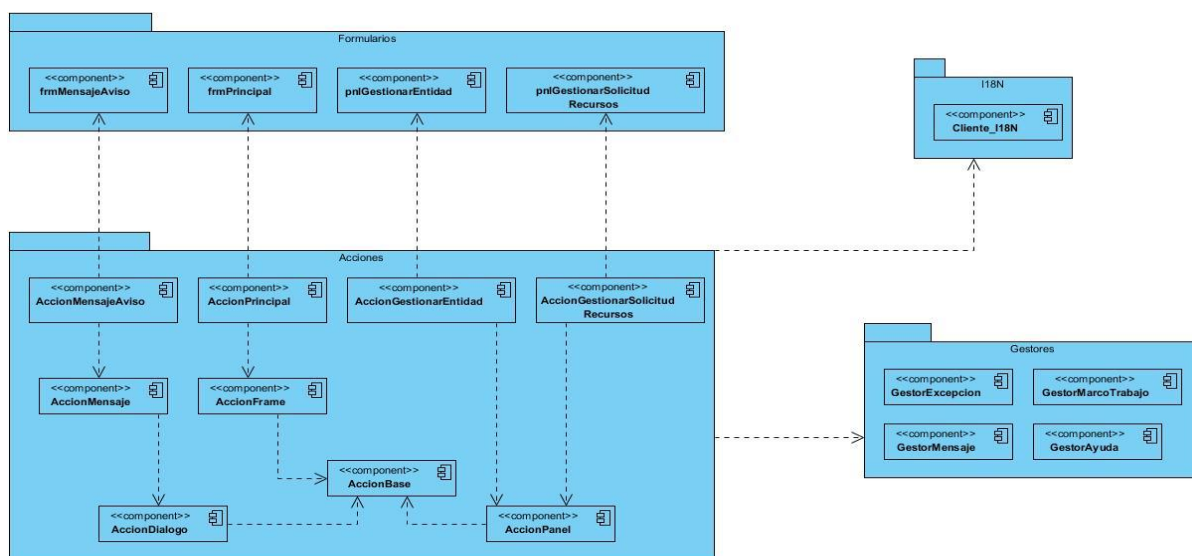


Figura 21. Diagrama de componentes de la capa de Presentación.

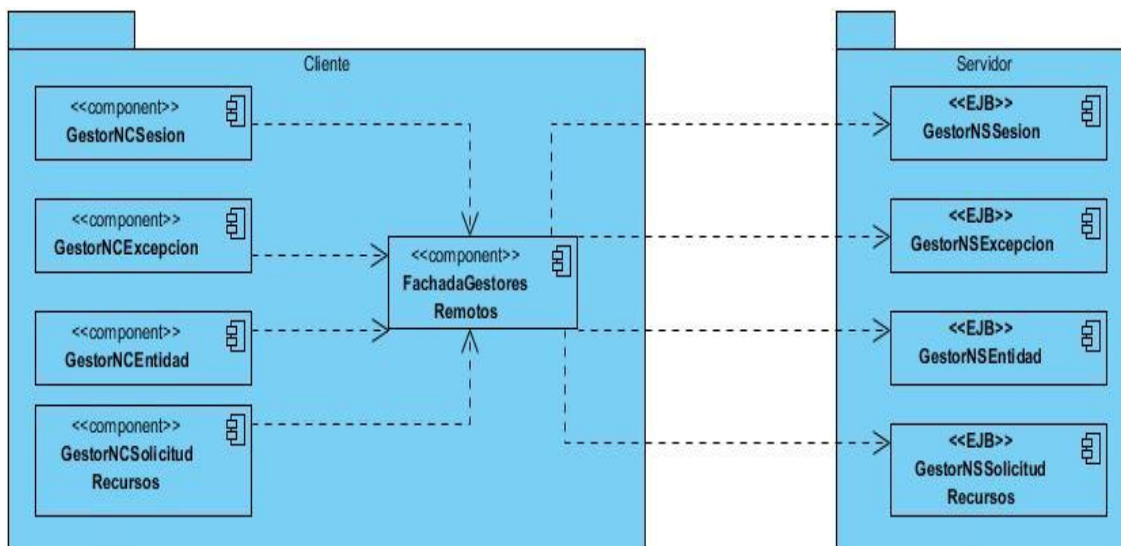


Figura 22. Diagrama de componentes de la capa de Lógica de Negocio.

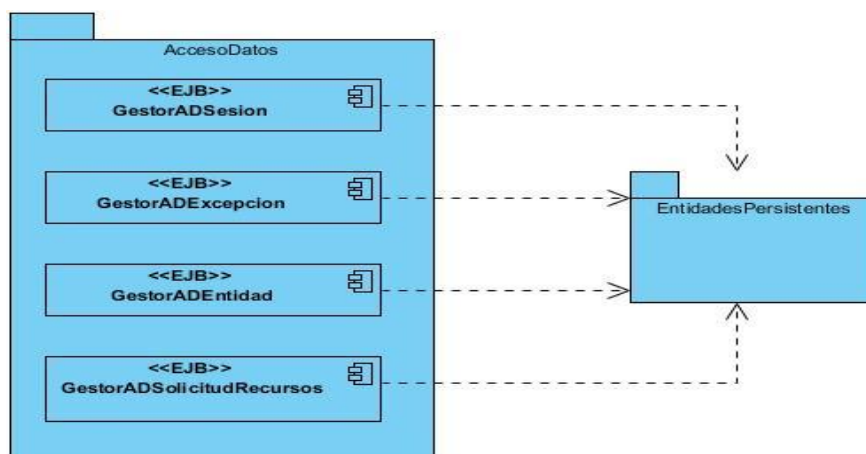


Figura 23. Diagrama de componentes de la capa Acceso a Datos.